

Introducing Apache Airflow® 3

The Next Evolution
in Orchestration

3.0

Airflow 3 Overview

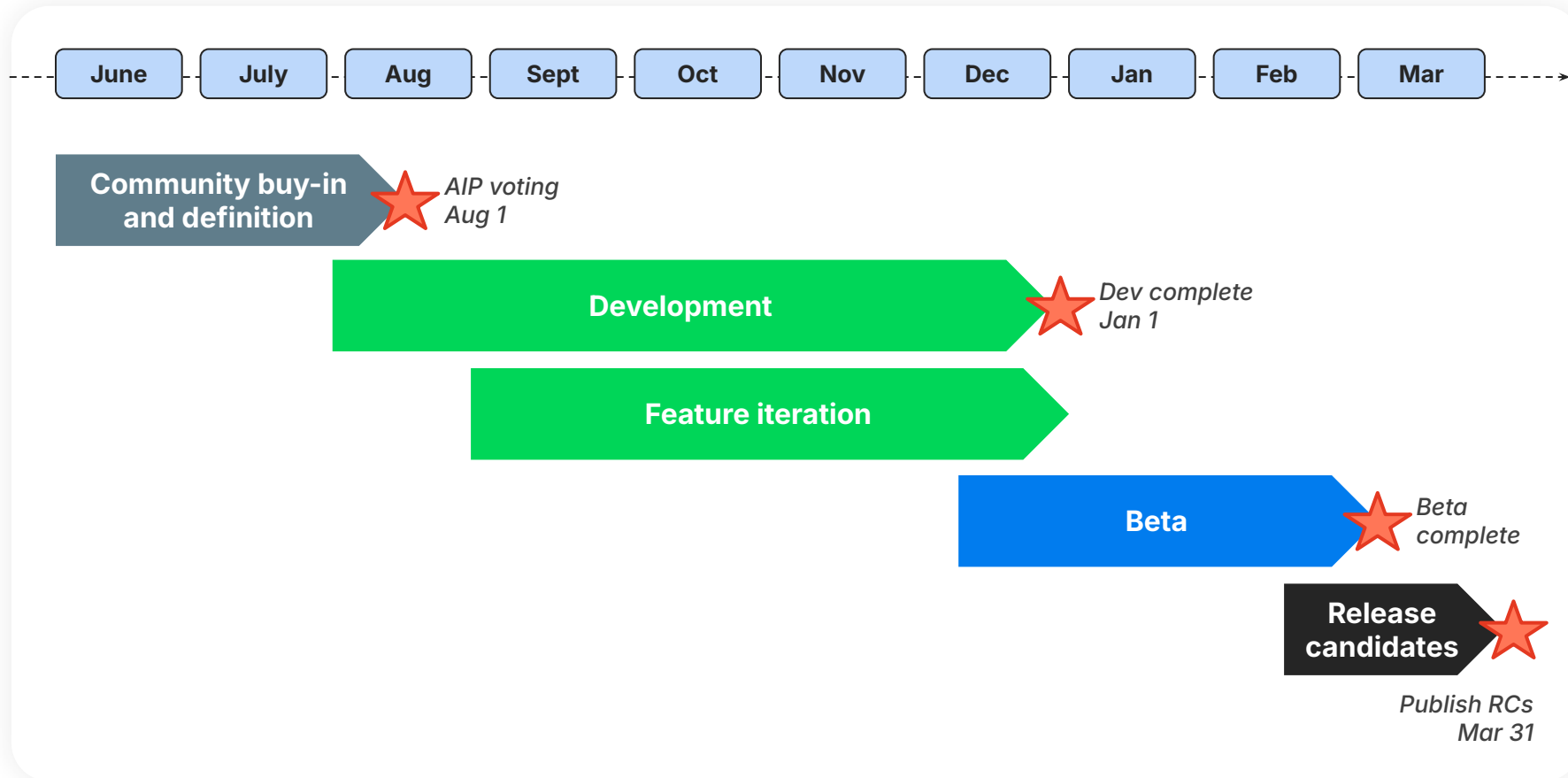
Vikram Koka

Chief Strategy Officer

Astronomer

3.0

Airflow 3.0 timeline (dated Sep 2024)



Airflow 3.0 released April 22nd, 2025!



- ③ 330+ developers worldwide, 500+ commits / month
- ③ 13 AIPs (Airflow Improvement Proposals)
- ③ Already, the third most downloaded Airflow version!

What does this mean for you?



Better user experience

- Dag Versioning, New UI, Backfills



Improved security posture

- Task Isolation, Updated Architecture



Run tasks anywhere, at any time and in any language

- Data Assets, Event driven

Airflow 3.1 released Sept 25, 2025!



1400+ commits into 3.1 (after 3.0)



163 developers



Accelerated development on the 3.0 foundation!

What does Airflow 3.1 bring?



Improved User Experience

- UI plugins, Translations (17 so far)



Expanded Event Driven Scheduling

- Supports: SQS, Apache Kafka, Redis, Azure (WIP)



Improved authoring

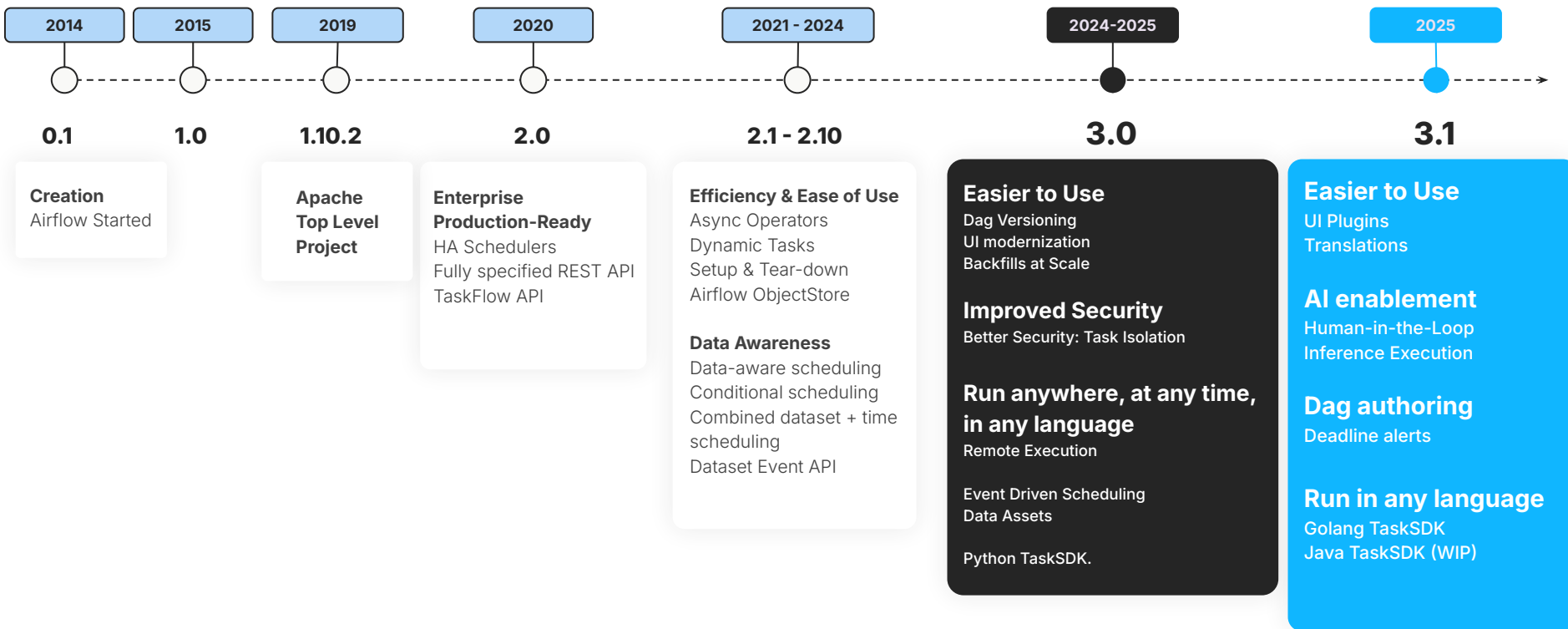
- Deadline Alerts



AI enablement

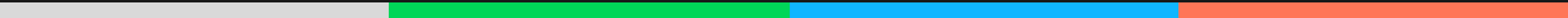
- Inference Execution
- Human-in-the-loop!

Airflow release timeline



Airflow expanding personas

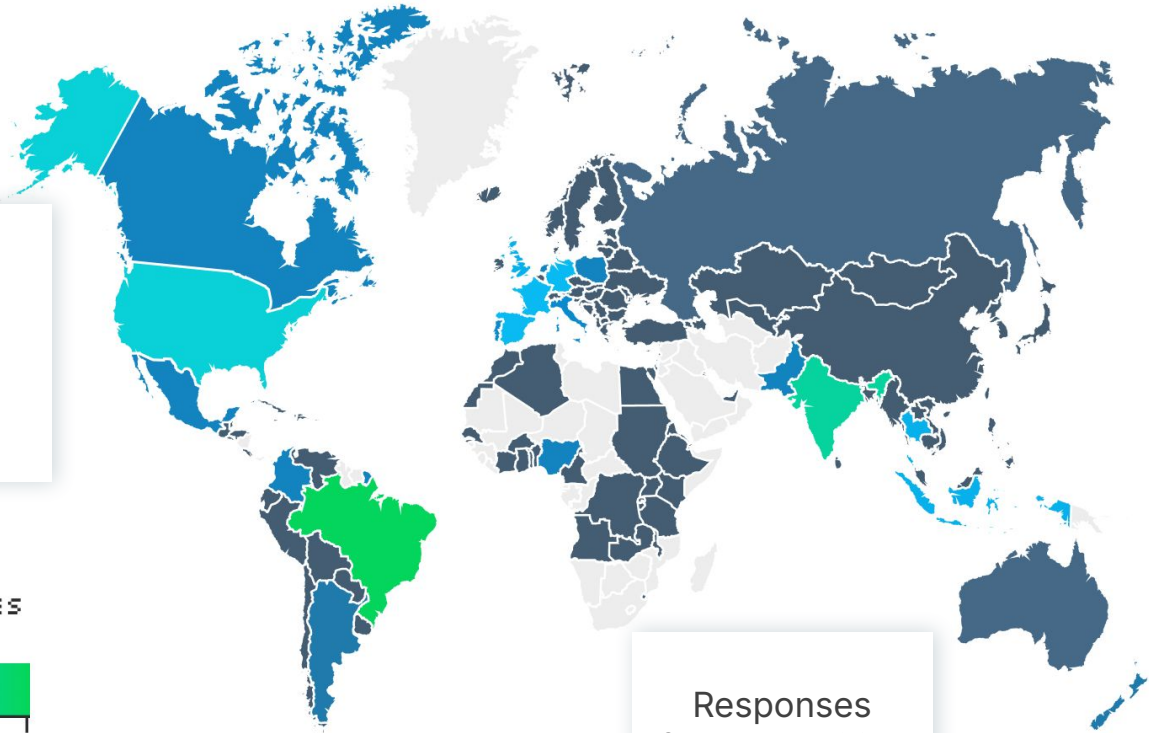
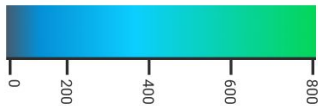
Key Community Insights from the Apache Airflow Survey



5,255
responses
(5x up on 2023)

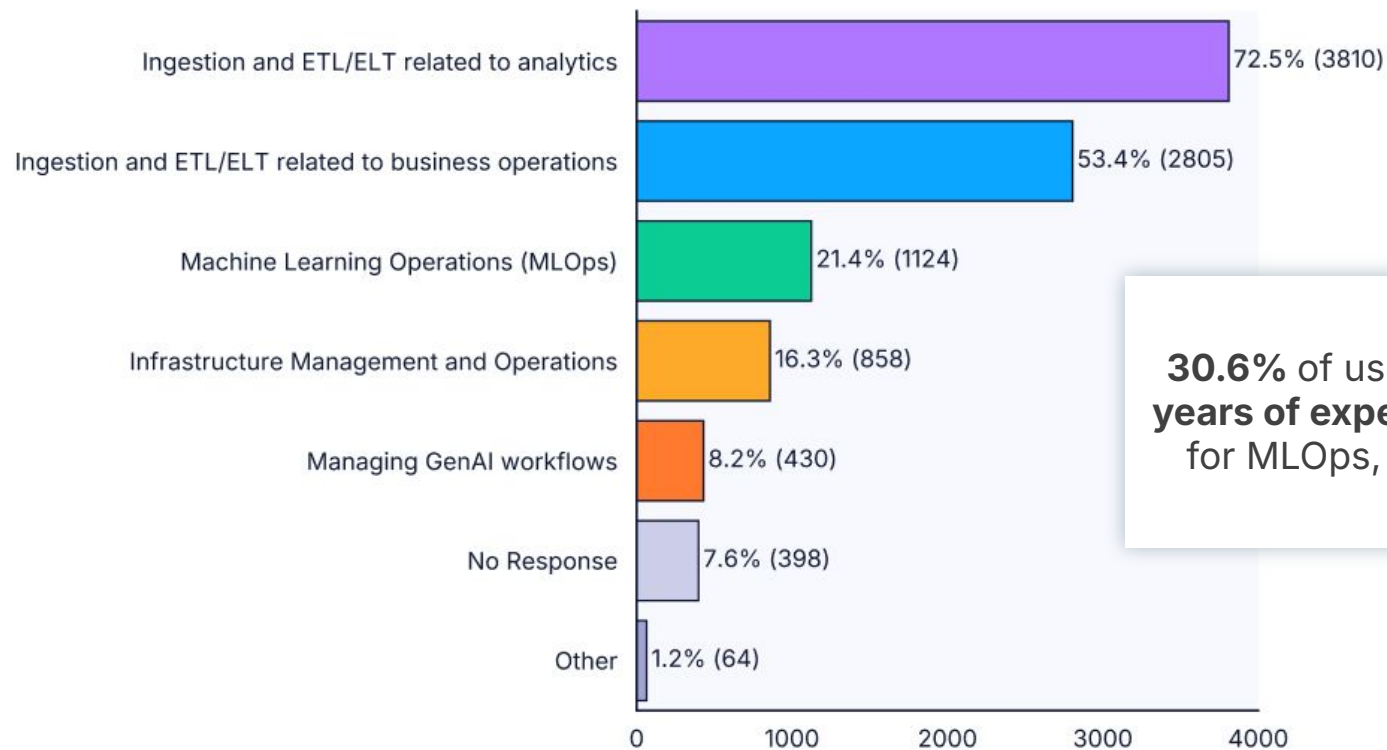
SURVEY RESPONSES

N=5255



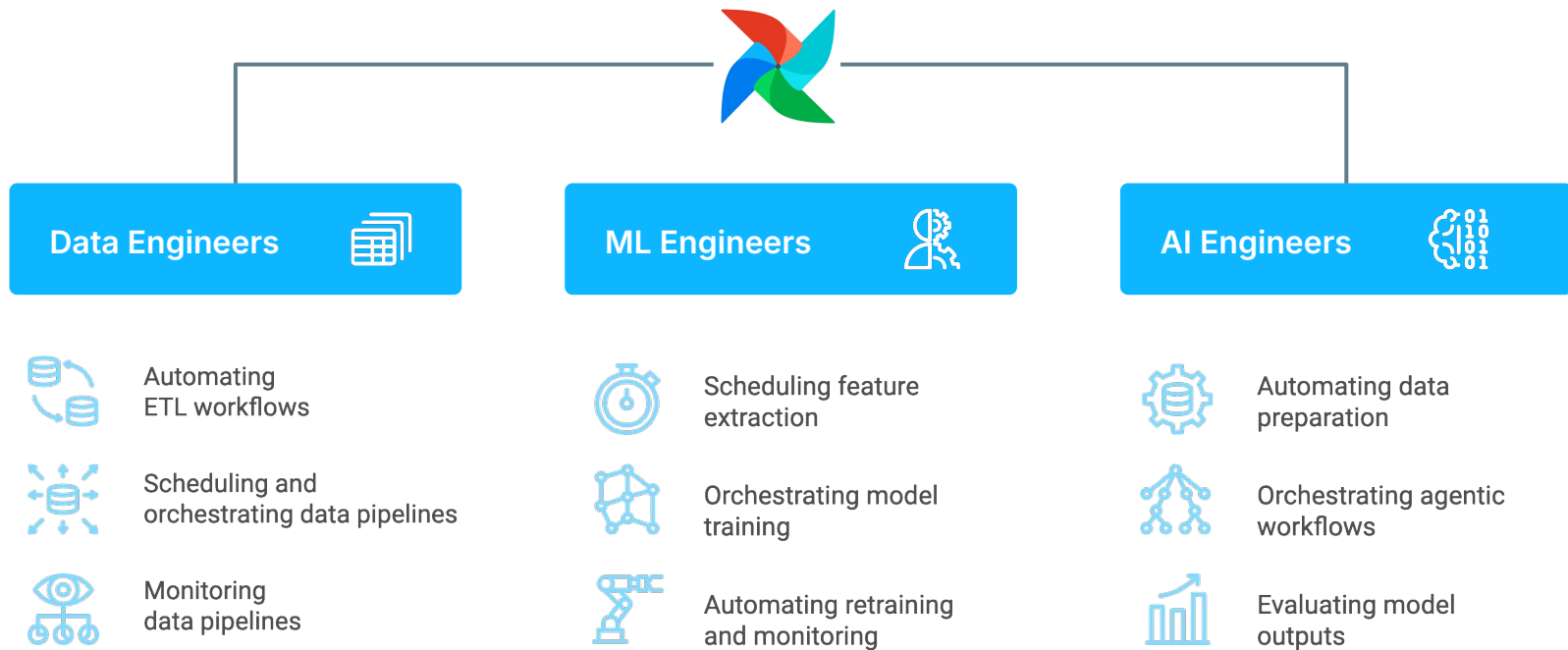
Responses
from 534 cities
in 116 countries

Use Cases Insight from the State of Airflow survey



30.6% of users with 5 or more years of experience use Airflow for MLOps, 13.3% for GenAI

Airflow is a “core technology” to a growing range of engineers and personas



Summit sessions reflect this



Around a third of use Case sessions about AI



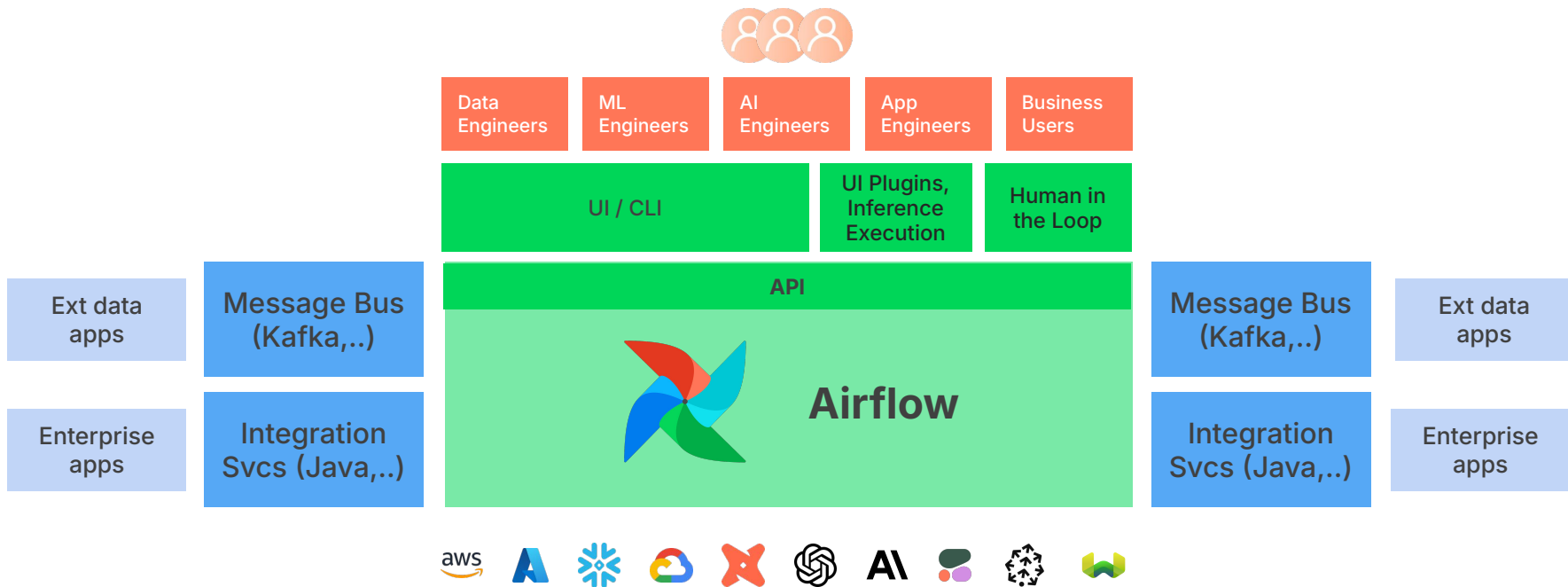
Foundational AI companies:  **OpenAI** **ANTHROPIC**

Enterprise	Financial	Healthcare	Consumer	Tech	Gov
IBM / Red Hat	Financial Times	Kaiser Permanente	Pinterest	King / Microsoft	INTRVL (voting)
SAP	Weav	iKang	Duolingo		

Note: Only a representative selection show above

Airflow ecosystem

Expanded with Airflow 3



UX features

Jed Cunningham

Daniel Standish

Brent Bovenzi

Pierre Jeambrun

3.0



UX features

Slides intro

Slide on API + React being a foundation - (Pierre - 3 mins)

UX features via demo (Brent - 10 mins)

Demo new features Released since Airflow 3.0 in 3.1:

Internationalization, Gantt, Calendar (Brent - 2 mins) and Plugins (Pierre - 5 mins)

Dag Versioning - Jed Cunningham

Backfills - Daniel Standish

Whole New Stack

The entire API and UI were rebuilt for Airflow 3. FastAPI & React

Thanks to our amazing community!

Pierre Jeambrun
Staff Software Engineer
Astronomer

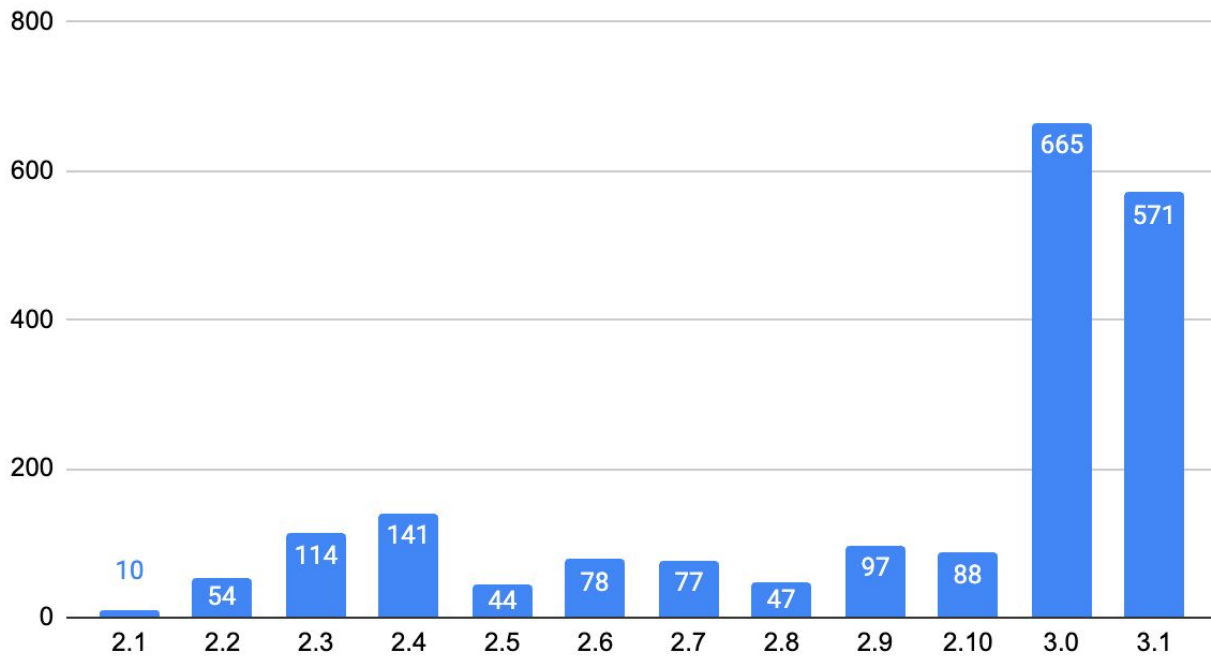
FastAPI Backend

- Reliable, modern, easy to maintain
- Remove dependencies (Flask AppBuilder, Connexion)
- Code first API design
- Inherit FastAPI features (automatic documentation, data validation, native async and more)

React UI

- ② Single Page Application, responsive
- ② Great React ecosystem (community, tools and packages)
- ② Flexible Deployment
- ② Opening opportunities for internationalization and more complex features

UI Pull Requests merged into main between release dates



New Dashboard

Brent Bovenzi

Engineer
Astronomer

✖ 5

Failed Dags >

📄 14

Dag Import Errors >

📈 0

Running Dags >

🔄 25

Active Dags >

☆ First 10 favorite Dags



Health

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✓ Dag Processor

Pool Slots

✓ 128

Manage Pools

History

📅 Last 24 Hours

2025-09-28 10:26:39 - 2025-09-29 10:26:39

📊 52

Dag Runs

👉 0

Queued

0.00%

📈 0

Running

0.00%

✓ 43

Success

82.69%

✖ 9

Failed

17.31%

📄 200

Task Instances

✓ 177

Success

88.50%

📄 21

Asset Events

Newest First

2025-09-29 08:45:29

📄 s3://consuming_2_task/asset_other_unknown.txt

Source: consume_1_or_both_2_and_3_with_asset_expressions

2025-09-29 08:45:29

📄 s3://consuming_2_task/asset_other_unknown.txt

Source: consume_1_or_2_with_asset_expressions

2025-09-29 08:45:29

📄 s3://consuming_1_task/asset_other.txt

Source: asset_consumes_1

2025-09-29 08:45:28

📄 loaded_data

Source: loaded_data

2025-09-29 08:45:22

📄 transformed_data

Source: transformed_data

Triggered Dag Run: ✓ loaded_data

Updated Dag Page



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Stats

5

Failed Dags >

14

Dag Import Errors >

0

Running Dags >

25

Active Dags >

☆ First 10 favorite Dags

extracted_data

latest_only

Health

Pool Slots

Manage Pools

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✓ Dag Processor

✓ 128

History

Last 24 Hours

2025-09-28 10:30:36 - 2025-09-29 10:30:36

52

Dag Runs

0

Queued

0.00%

0

Running

0.00%

43

Success

82.69%

9

Failed

17.31%

200

Task Instances

177

Success

88.50%

21

Asset Events

Newest First

2025-09-29 08:45:29

s3://consuming_2_task/asset_other_unknown.txt

Source: consume_1_or_both_2_and_3_with_asset_expressions

2025-09-29 08:45:29

s3://consuming_2_task/asset_other_unknown.txt

Source: consume_1_or_2_with_asset_expressions

2025-09-29 08:45:29

s3://consuming_1_task/asset_other.txt

Source: asset_consumes_1

2025-09-29 08:45:28

loaded_data

Source: loaded_data

2025-09-29 08:45:22

transformed_data

Source: transformed_data

Triggered Dag Run:

loaded_data

Dag <> Asset Graphs

✖ 5

Failed Dags >

📄 14

Dag Import Errors >

🔄 0

Running Dags >

🔌 25

Active Dags >

☆ First 10 favorite Dags



🛡️ Health

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✓ Dag Processor

🔌 Pool Slots



📅 History

📅 Last 24 Hours ▾

2025-09-28 10:33:33 - 2025-09-29 10:33:33



📄 21 Asset Events

Newest First ▾

2025-09-29 08:45:29

📄 s3://consuming_2_task/asset_other_unknown.txt

Source: consume_1_or_both_2_and_3_with_asset_expressions

2025-09-29 08:45:29

📄 s3://consuming_2_task/asset_other_unknown.txt

Source: consume_1_or_2_with_asset_expressions

2025-09-29 08:45:29

📄 s3://consuming_1_task/asset_other.txt

Source: asset_consumes_1

2025-09-29 08:45:28

📄 loaded_data

Source: loaded_data

2025-09-29 08:45:22

📄 transformed_data

Source: transformed_data

Triggered Dag Run: ✓ loaded_data

Support for 17 Languages



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Dags Dag Runs Task Instances

Search Dags

Advanced Search ⌘+K

All

Failed

Queued

Running

Success

Required Actions

All

Filter by tag

All

150 Dags 14

Sort by Latest Run Start Date...



nested_groups

☒

▶

☆

🗑️

Schedule	Latest Run	Next Run	
0 0 * * *	2025-09-29 10:16:13 ✖	2025-09-29 20:00:00	

loaded_data

☒

▶

☆

🗑️

Schedule	Latest Run	Next Run	
transformed_data	2025-09-29 08:45:22 ✔		

consume_1_or_both_2_and_3_with_asset_expressions

☒

▶

☆

🗑️

Schedule	Latest Run	Next Run	
0 of 3 assets updated	2025-09-29 08:45:16 ✔		

consume_1_or_2_with_asset_expressions

☒

▶

☆

🗑️

Schedule	Latest Run	Next Run	
0 of 2 assets updated	2025-09-29 08:45:16 ✔		

asset_consumes_1 🔗 asset-scheduled, consumes

☒

▶

☆

🗑️

Schedule	Latest Run	Next Run	
s3://dag1/output_1.txt	2025-09-29 08:45:16 ✔		

Calendar View



Dag

sleep_dag_fail



Home



Dags



Assets



Browse



Admin



Plugins



Docs



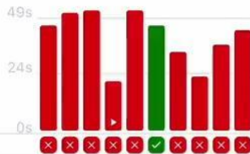
User



Options ▾



sleep_for_5

sleep_dag_fail ☐

Schedule

0 0,6,12,18 ** MON,WED,FRI

Latest Run

2025-09-29 08:00:00 ✖

Next Run

2025-09-29 14:00:00

Owner

airflow

Tags

Latest Dag Version

v3

☆ Favorite Dag



Overview

Runs

Tasks

Calendar

Backfills

Audit Log

Code

Details



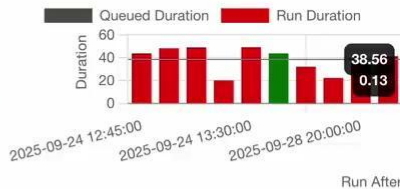
Last 24 Hours ▾

2025-09-28 10:50:40 - 2025-09-29 10:50:40

2 Failed Tasks

2 Failed Runs

Last 10 Dag Runs



Recent Failed Task Logs

Wrap



sleep_for_5 2025-09-29 08:00:00

Show Logs



View full logs



sleep_for_5 2025-09-28 20:00:00

Show Logs



View full logs

New Ways to Filter



Airflow



Stats

✖ 5 Failed Dags >

📄 14 Dag Import Errors >

🔄 0 Running Dags >

🔌 25 Active Dags >

☆ First 10 favorite Dags



Health

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✓ Dag Processor

Pool Slots

✓ 128

Manage Pools

History

Last 24 Hours

2025-09-28 10:51:27 - 2025-09-29 10:51:27

📊 52 Dag Runs

📄 0 Queued

0.00%

🔄 0 Running

0.00%

✓ 43 Success

82.69%

✖ 9 Failed

17.31%

📄 200 Task Instances

📄 21 Asset Events

Newest First

- 2025-09-29 08:45:29

📄 s3://consuming_2_task/asset_other_unknown.txt

Source: consume_1_or_both_2_and_3_with_asset_expressions
- 2025-09-29 08:45:29

📄 s3://consuming_2_task/asset_other_unknown.txt

Source: consume_1_or_2_with_asset_expressions
- 2025-09-29 08:45:29

📄 s3://consuming_1_task/asset_other.txt

Source: asset_consumes_1
- 2025-09-29 08:45:28

📄 loaded_data

Source: loaded_data
- 2025-09-29 08:45:22

Plugins

- Backward compatibility layer for Airflow 2 plugins in Airflow 3 (requires fab provider)
- Have access to the powerful FastAPI interface via sub applications and middlewares
- Extend the UI through External Views and React Applications registration

Pierre Jeambrun
Staff Software Engineer
Astronomer

IFrames



Home



Dags



Assets



Browse



Admin



Security



Plugins



Docs



User

Welcome to Your New React App!

This project was bootstrapped with the Airflow React Plugin tool.

Toggle Theme

Welcome

Stats

✖ 1

Failed Dags >

🔴 5

Dag Import Errors >

🟢 0

Running Dags >

🟢 2

Active Dags >

☆ First 10 favorite Dags

No favorites yet. Click the star icon next to a Dag in the list to add it to your favorites.

Health

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✖ Dag Processor

Pool Slots



Manage Pools

History

📅 Last 24 Hours ▾ 2025-09-29 10:20:37 - 2025-09-30 10:20:37

1 Dag Run



1 Task Instance



0 Asset Events

Newest First ▾

No Asset Events found.

External Link



Home

Dags

Assets

Browse

Admin

Security

Plugins

Docs

User

To exit full screen, press and hold **esc** **pp!**

This project was bootstrapped with the Airflow React Plugin tool.

Toggle Theme

Welcome

Stats

✕ 1

Failed Dags >

🔴 5

Dag Import Errors >

🟢 0

Running Dags >

🟡 2

Active Dags >

☆ First 10 favorite Dags

No favorites yet. Click the star icon next to a Dag in the list to add it to your favorites.

Health

✓ MetaDatabase ✓ Scheduler ✓ Triggerer ✓ Dag Processor

Pool Slots

✓ 128

Manage Pools

History

Last 24 Hours 2025-09-29 10:25:00 - 2025-09-30 10:25:00

1 Dag Run

0 Queued 0.00%

0 Running 0.00%

0 Success 0.00%

1 Failed 100.00%

1 Task Instance

1 Failed 100.00%

0 Asset Events

Newest First

No Asset Events found.

React App



Home



Dags



Assets



Browse



Admin



Security



Plugins



Docs



User

Welcome to Your New React App!

This project was bootstrapped with the Airflow React Plugin tool.

[Toggle Theme](#)

Welcome

Stats

✕ 1

Failed Dags >

📄 5

Dag Import Errors >

🔄 0

Running Dags >

🔄 2

Active Dags >

☆ First 10 favorite Dags

No favorites yet. Click the star icon next to a Dag in the list to add it to your favorites.

🛡️ Health

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✓ Dag Processor

🔗 Pool Slots

✓ 128

[Manage Pools](#)

📅 History



Last 24 Hours



2025-09-29 10:25:00 - 2025-09-30 10:25:00



1 Dag Run



0 Queued

0.00%



0 Running

0.00%



0 Success

0.00%



1 Failed

100.00%



1 Task Instance



1 Failed

100.00%



0 Asset Events

Newest First

No Asset Events found.



Home



Dags



Assets



Browse



Admin



Security



Plugins



Docs



User

Welcome to Your New React App!

This project was bootstrapped with the Airflow React Plugin tool.

[Toggle Theme](#)

Welcome

Stats

✕ 1

[Failed Dags >](#)

📄 5

[Dag Import Errors >](#)

🔄 0

[Running Dags >](#)

🔄 2

[Active Dags >](#)

☆ First 10 favorite Dags

No favorites yet. Click the star icon next to a Dag in the list to add it to your favorites.

Health

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✓ Dag Processor

Pool Slots

✓ 128

[Manage Pools](#)

History



Last 24 Hours



2025-09-29 10:28:57 - 2025-09-30 10:28:57

📊 1 **Dag Run**

⏸ 0

Queued

0.00%

🔄 0

Running

0.00%

✓ 0

Success

0.00%

✕ 1

Failed

100.00%

📄 1

Task Instance

✕ 1

Failed

100.00%

📋 0 **Asset Events**

Newest First



No Asset Events found.



Navbar Categories



Home



Dags



Assets



Browse



Admin



Security



Plugins



Docs



User

Welcome to Your New React App!

This project was bootstrapped with the Airflow React Plugin tool.

Toggle Theme

Welcome

Stats

✕ 1

Failed Dags >

🔴 5

Dag Import Errors >

🟢 0

Running Dags >

🟡 2

Active Dags >

☆ First 10 favorite Dags

No favorites yet. Click the star icon next to a Dag in the list to add it to your favorites.

Health

✓ MetaDatabase ✓ Scheduler ✓ Triggerer ✓ Dag Processor

Pool Slots

✓ 128

Manage Pools

History

📅 Last 24 Hours ▾ 2025-09-29 10:27:26 - 2025-09-30 10:27:26

1 Dag Run

0 Queued

0.00%

0 Running

0.00%

0 Success

0.00%

✕ 1 Failed

100.00%

1 Task Instance

✕ 1 Failed

100.00%

0 Asset Events

Newest First ▾

No Asset Events found.



Destinations

Welcome to Your New React App!

This project was bootstrapped with the Airflow React Plugin tool.

Toggle Theme

Welcome

Stats

✖ 1

Failed Dags >

📄 5

Dag Import Errors >

🔄 0

Running Dags >

🔧 2

Active Dags >

☆ First 10 favorite Dags

No favorites yet. Click the star icon next to a Dag in the list to add it to your favorites.

Health

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✓ Dag Processor

Pool Slots



Manage Pools

History

📅

Last 24 Hours

▼

2025-09-29 10:28:57 - 2025-09-30 10:28:57



📋 0

Asset Events

Newest First ▼

No Asset Events found.



React App Placement

Plugins Takeaways

-  Airflow 2 plugins are still supported, not a blocker to upgrade to Airflow 3
-  Flexible Interface
-  Can't wait to see what the community builds!

Dag Versioning

Jed Cunningham

Principal Software Engineer
Astronomer

Dag Versioning

Execution

UI

Dag Bundle

collection of Dags and other files

Types of Dag Bundles

Versioned vs not

Run task using a specific
version of
Dag code

Dag code from **many sources**

"Code" vs Dag

Bundle Versions + DagRuns

V1 Bundle

V2 Bundle





Dag
demo



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User



Options ▾



sleep
hello

Search Dags ****K**

Trigger

demo ☐

Favorite Dag



Schedule

Latest Run

Next Run

Owner
airflow

Tags

Latest Dag Version
v1

Overview

Runs

Tasks

Calendar

Audit Log

Code

Details

Parsed at:

2025-09-29

09:32:30

Parse

Duration:

0.03s

dagDetails.bundleVersion:

3eeb907b73ffce1f0280ad290c4173c3fc

d540e4

v1 2025-09-29 09:32:30 ▾

Copy

Wrap

```
1 from airflow.sdk import DAG
2 from airflow.providers.standard.operators.bash import BashOperator
3
4 with DAG(dag_id="demo"):
5     sleep = BashOperator(task_id="sleep", bash_command="sleep 60")
6     hello = BashOperator(task_id="hello", bash_command="echo 'Hello Airflow Summit!'")
7
8     sleep >> hello
9
```



Home

Dags

Assets

Browse

Admin

Plugins

Docs

User

Dag demo Dag Run 2025-09-29 09:34:03



Options



2025-09-29 09:34:03 Success

Add a note

Clear Dag Run

Mark Dag Run as...

Logical Date	Run Type	Start Date	End Date	Duration	Triggering User Name
2025-09-29 09:34:03	manual	2025-09-29 09:34:05	2025-09-29 09:34:18	00:00:12	Anonymous

Dag Version(s)

v2

Task Instances Asset Events Audit Log Code Details

Search Tasks

All States

Task ID	Map Index	State	Start Date	End Date	Try Number	Pool
sleep		Success	2025-09-29 09:34:05	2025-09-29 09:34:16	1	default_
hello_new		Success	2025-09-29 09:34:16	2025-09-29 09:34:17	1	default_

Backfills

Daniel Standish
Staff Software Engineer
Astronomer

What is backfill in Airflow?



Re-run tasks over dates

Why:

- data correction
- new pipelines
- large data migrations

Backfills in Airflow 2



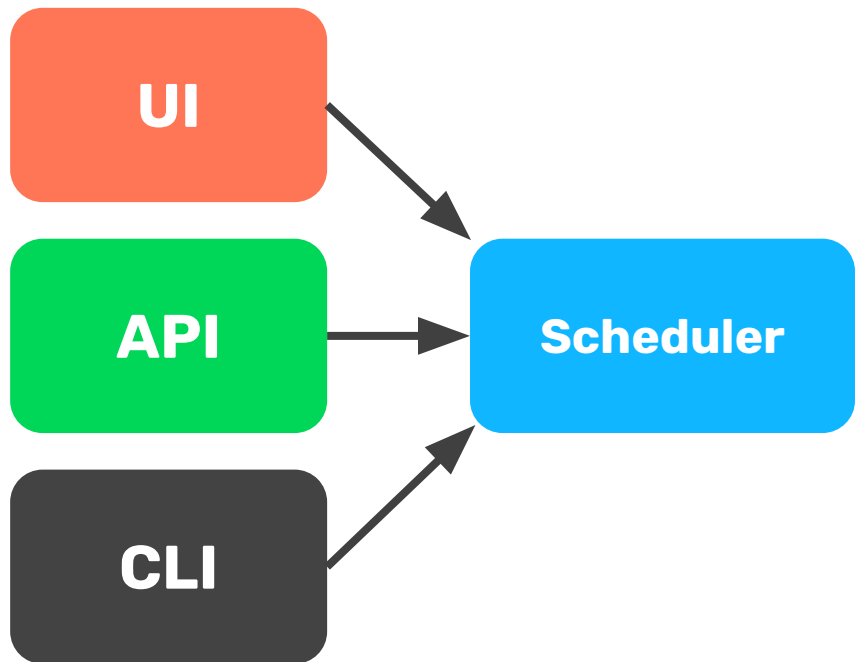
```
airflow dags backfill my_dag --from-date '1900-01-01'
```

CLI only → less accessible, less reliable

Goals for Airflow 3

-  Make backfills easier to run and observe
-  Give users more control over how they are run

Backfills in Airflow 3



- Runs managed by core scheduler
- Feature parity across all interfaces



Home



Dags



Assets



Browse



Admin



Docs



User

Dag
example_backfill_01



Options ▾



example_backfill_01 ☒

☆ Favorite Dag



Schedule

Latest Run

Next Run

Owner

Tags

Latest Dag Version

📅 0 0 ***

2025-09-25 17:00:00

2025-09-26 17:00:00

airflow

v1

Overview

Runs

Tasks

Calendar

Backfills

Audit Log

Code

Details



Last 24 Hours ▾

2025-09-25 16:37:44 - 2025-09-26 16:37:44

0

Failed Tasks

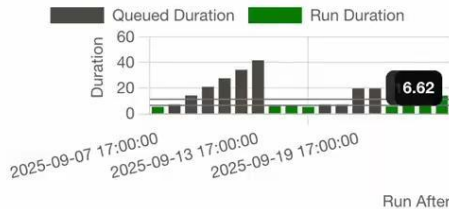


0

Failed Runs



Last 18 Dag Runs



Backfills in Airflow 3: wrap up

- Backfills are easier to use
- More control over execution
- Feature parity across UI / API / CLI

Run Tasks Anywhere

Kaxil Naik
Amogh Desai
Jens Scheffler
Buğra Öztürk

3.0

Run Tasks Anywhere: Task Isolation

3.0

Kaxil Naik

Sr Director of Engineering @ Astronomer
Airflow Committer & PMC Member

Amogh Desai

Sr Software Engineer @ Astronomer
Airflow Committer & PMC Member

The Incident

my_dag.py



Generate Python task for Snowflake cleanup



files > dags > my_dag.py > cleanup_snowflake_connections > delete_snowflake_co

```
1 from airflow.decorators import dag, task
2 from airflow.models import Connection
3 from airflow.utils.session import create_session
4 from datetime import datetime
```

I'm cleaning up my Airflow environment and have a bunch of old Snowflake test connections I want to remove. Can you help me write a task to delete all connections daily?



I'll help you create a task to clean up those Snowflake connections with a daily schedule.

Generate Python task for Snowflake cleanup



I'm cleaning up my Airflow environment and have a bunch of old Snowflake test connections I want to remove. Can you help me write a task to delete all connections daily?



```
16         ).delete()
17
18         print(f"Deleted {deleted_count} Snowflake connections")
19         return deleted_count
20
21     delete_snowflake_connections()
22
23 cleanup_snowflake_connections()
```

and uses the `@task` decorator

- Logs the deletion count so you can see how many connections were removed

The DAG will run automatically every day and clean up all Snowflake connections regardless of their connection ID.

Review Changes



The Incident

my_dag.py •

▶ ▾ □ ...

files > dags > my_dag.py > cleanup_snowflake_connections > delete_snowflake_co

```
1 from airflow.decorators import dag, task
2 from airflow.models import Connection
3 from airflow.utils.session import create_session
4 from datetime import datetime
5
6 @dag(
7     schedule='@daily',
8     start_date=datetime(2024, 1, 1)
9 )
10 def cleanup_snowflake_connections():
11     @task
12     def delete_snowflake_connections():
13         with create_session() as session:
14             deleted_count = session.query(Connection).filter(
15                 Connection.conn_type == 'snowflake'
16             ).delete()
17
18         print(f"Deleted {deleted_count} Snowflake connections")
19         return deleted_count
20
21     delete_snowflake_connections()
22
23 cleanup_snowflake_connections()
```

Generate Python task for Snowflake cleanup

+ ↻ 📄 ...

I'm cleaning up my Airflow environment and have a bunch of old Snowflake test connections I want to remove. Can you help me write a task to delete all connections daily?

↩

I'll help you create a task to clean up those Snowflake connections with a daily schedule.

Read my_dag.py

my_dag.py +24 -1 ✓

... 📄 ↕

Perfect! I've created a daily scheduled DAG that will clean up your Snowflake connections. The code:

- Runs daily with `schedule='@daily'`
- Uses `session.query(Connection).filter(Connection.conn_type == 'snowflake').delete()` to remove all connections with `conn_type 'snowflake'`
- Imports `Connection` from `airflow.models`, `create_session` from `airflow.utils.session`, and uses the `@task` decorator
- Logs the deletion count so you can see how many connections were removed

The DAG will run automatically every day and clean up all Snowflake connections regardless of their connection ID.

☰ Review Changes

👍 💬 📄 ...

The Incident

my_dag.py

▶ ▢ ...

Generate Python task for Snowflake cleanup

+ ⓘ 🔍 ...

@task

```
def delete_snowflake_connections():
```

```
    with create_session() as session:
```

```
        deleted_count = session.query(Connection).filter(
            Connection.conn_type == 'snowflake'
        ).delete()
```

```
    session.commit()
```

```
    print(f"Deleted {deleted_count} Snowflake connections")
```

```
    return deleted_count
```

25 Snowflake connections DELETED !!

The Incident



Ash Berlin-Taylor 15:24

Ummmm wait, what!? Where did all our connections go? 🚫 Every dag is failing! 🚫

Calling an incident !

B *I* ☹️ | 🔗 ½☰ ☰☰ | ☰☰ </> </>

aah, I think it was my change -- sorry 😬

+ Aa 😊 @ | 📺 🎤 | 📝



The Real Damage

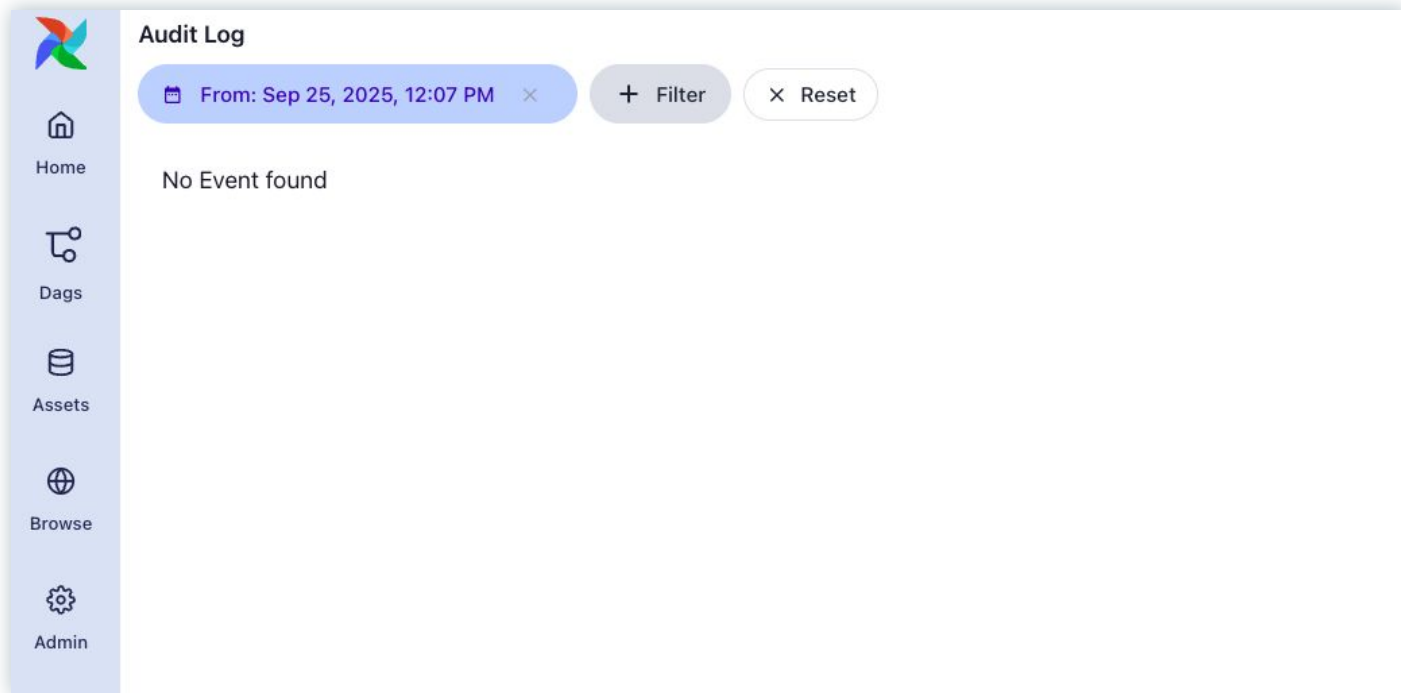
Security

Compliance

Performance

Operations

The Real Damage



The Real Damage

airflow-maintenance-dags

A series of DAGs/Workflows to help maintain the operation of Airflow

DAGs/Workflows

- [backup-configs](#)
 - A maintenance workflow that you can deploy into Airflow to periodically take backups of various Airflow configurations and files.
- [clear-missing-dags](#)
 - A maintenance workflow that you can deploy into Airflow to periodically clean out entries in the DAG table of which there is no longer a corresponding Python File for it. This ensures that the DAG table doesn't have needless items in it and that the Airflow Web Server displays only those available DAGs.
- [db-cleanup](#)
 - A maintenance workflow that you can deploy into Airflow to periodically clean out the DagRun, TaskInstance, Log, XCom, Job DB and SlaMiss entries to avoid having too much data in your Airflow MetaStore.

The Real Damage



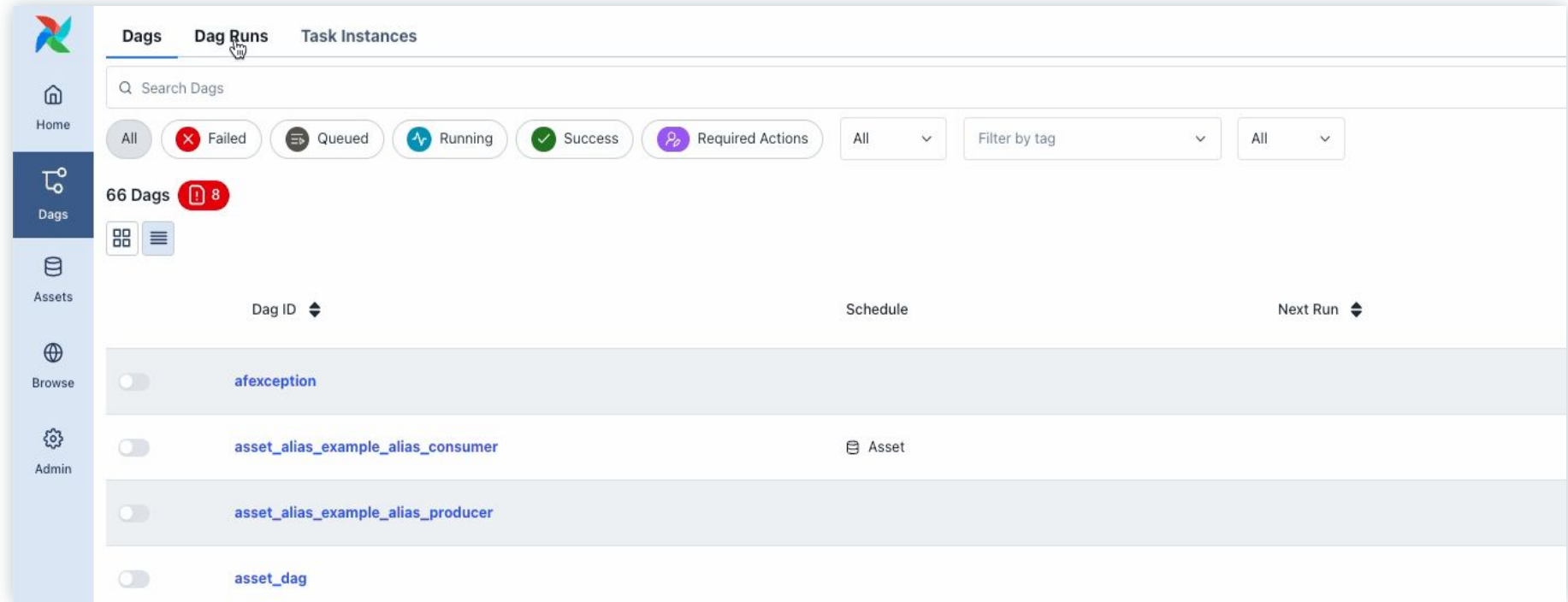
```
airflow trigger_dag --conf  
'{"maxDBEntryAgeInDays": 30}' airflow-db-cleanup
```

The Real Damage



```
airflow trigger_dag --conf  
'{"maxDBEntryAgeInDays": 0}' airflow-db-cleanup
```

The Real Damage



The screenshot displays the Apache Airflow web interface, specifically the 'Dags' tab. The interface is organized into a sidebar on the left and a main content area. The sidebar contains navigation links: Home, Dags (selected), Assets, Browse, and Admin. The main content area features a header with tabs for 'Dags', 'Dag Runs', and 'Task Instances'. Below the tabs is a search bar labeled 'Search Dags'. A row of status filters is present, including 'All', 'Failed' (highlighted with a red circle), 'Queued', 'Running', 'Success', and 'Required Actions'. Below the filters, it shows '66 Dags' with a red badge indicating '8' failed DAGs. A table of DAGs is displayed with columns for 'Dag ID', 'Schedule', and 'Next Run'. The first DAG is 'afexception'. The second DAG is 'asset_alias_example_alias_consumer', which is marked as an 'Asset'.

Dag ID	Schedule	Next Run
☐ afexception		
☐ asset_alias_example_alias_consumer	Asset	
☐ asset_alias_example_alias_producer		
☐ asset_dag		

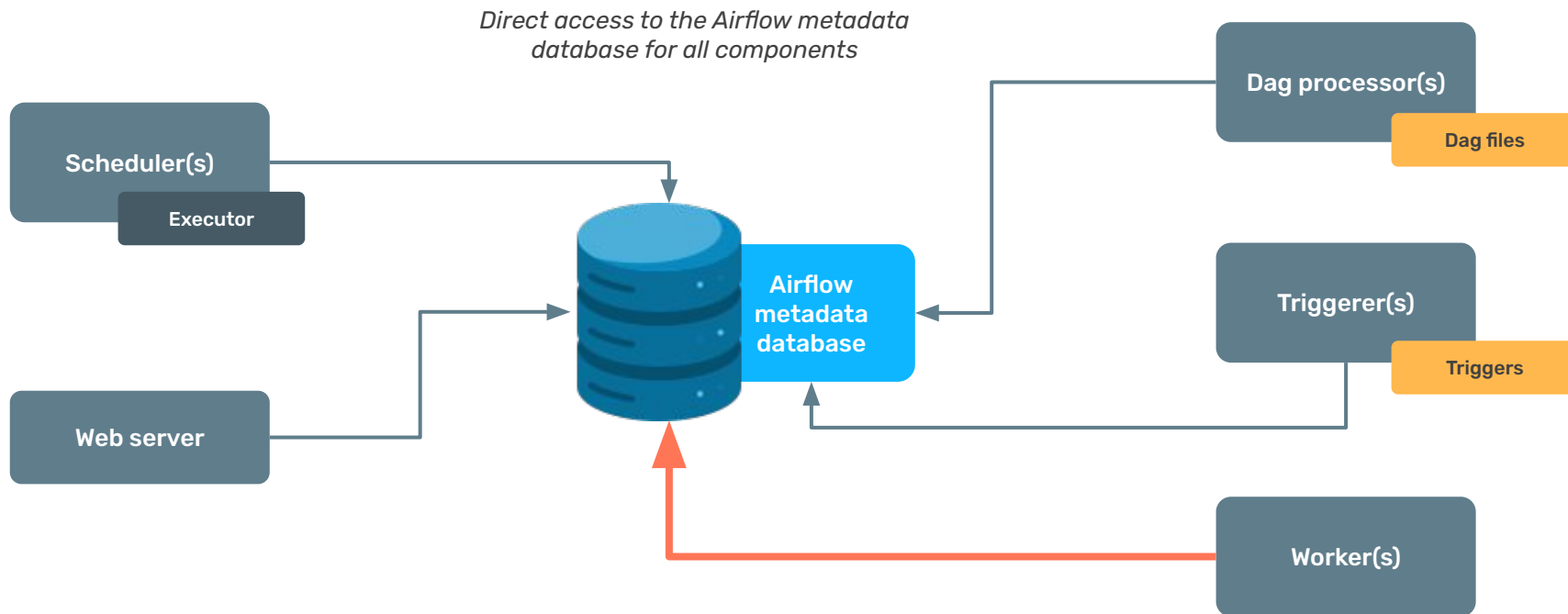
Security?

Performance?

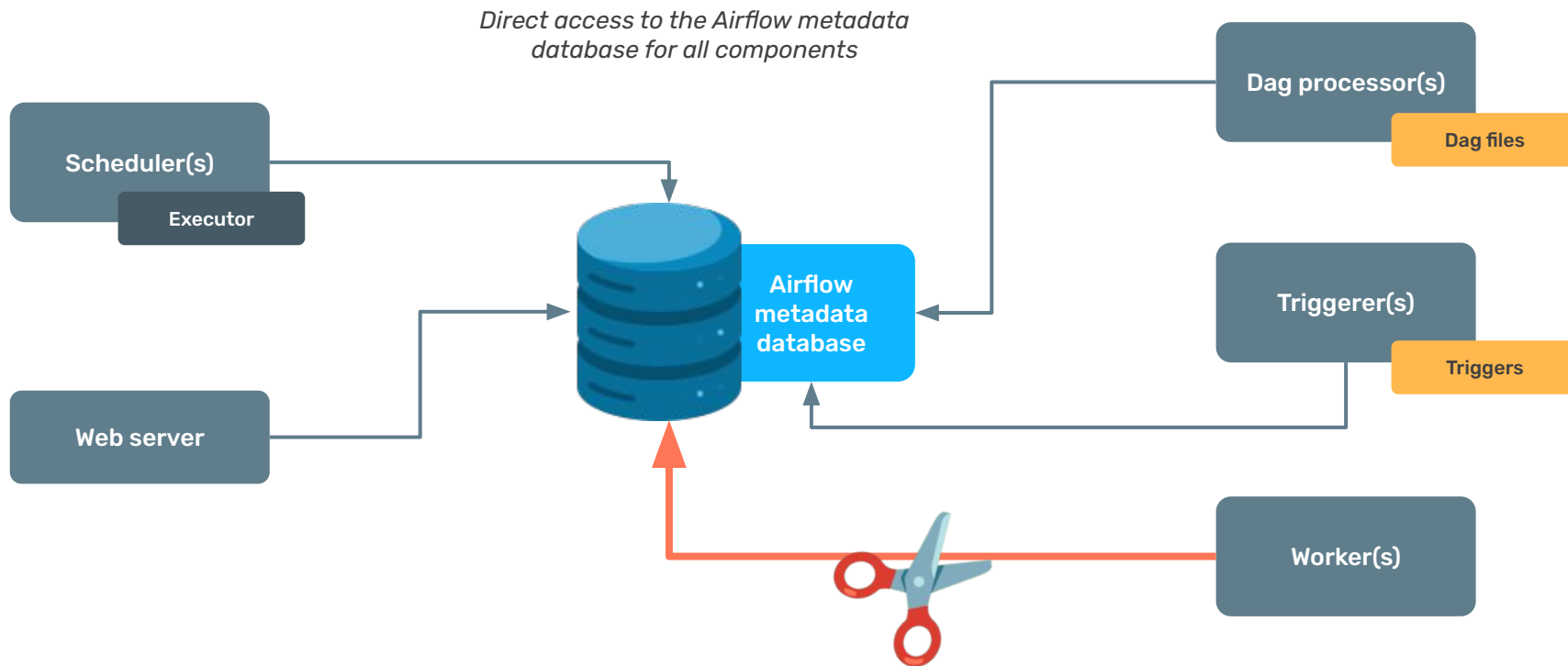
We needed a fundamental change.

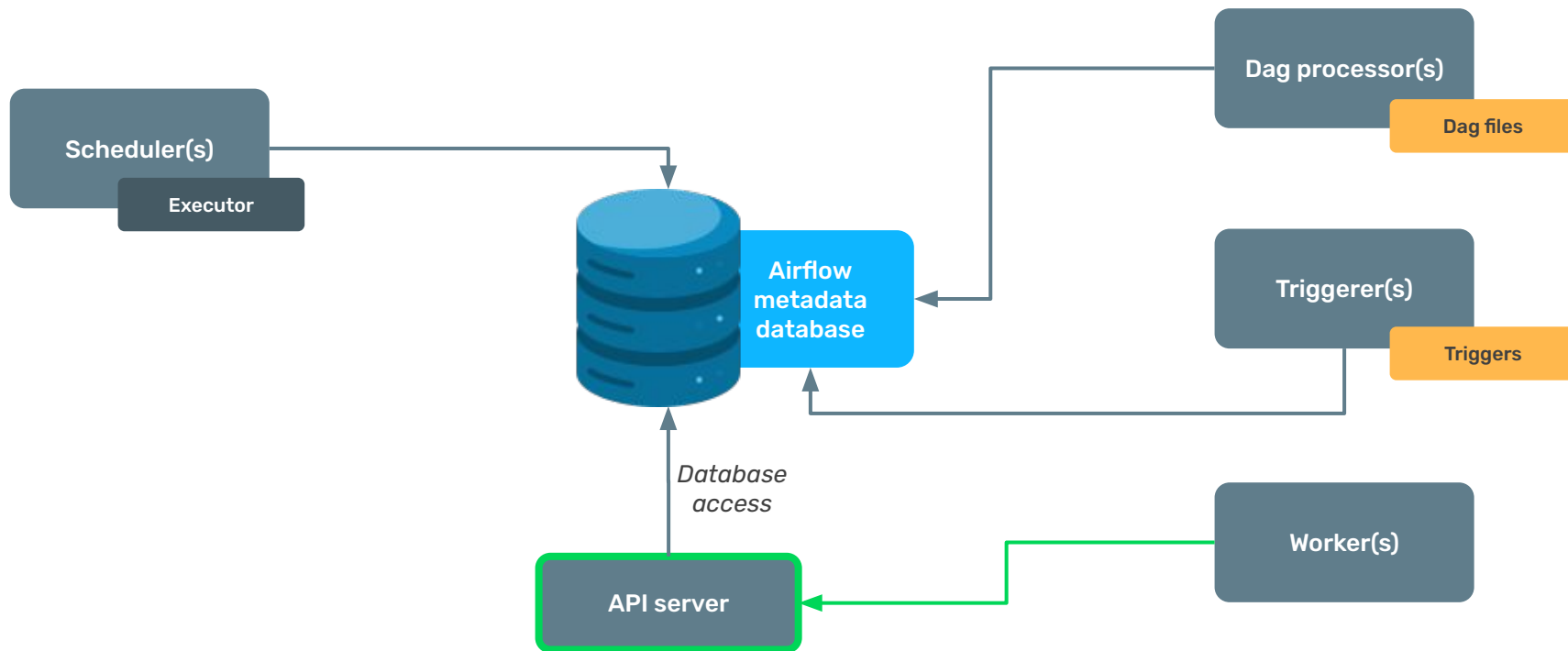
Not a band-aid, but **surgery**.

Airflow 2.x: Everything in one place

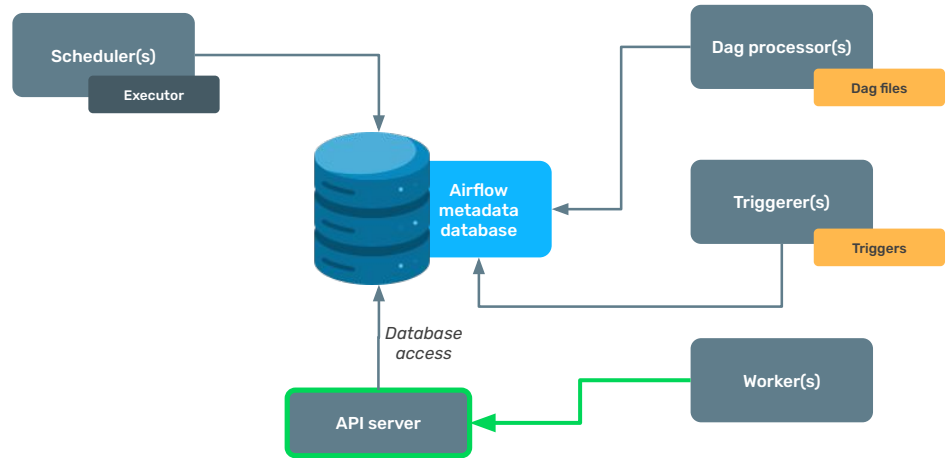
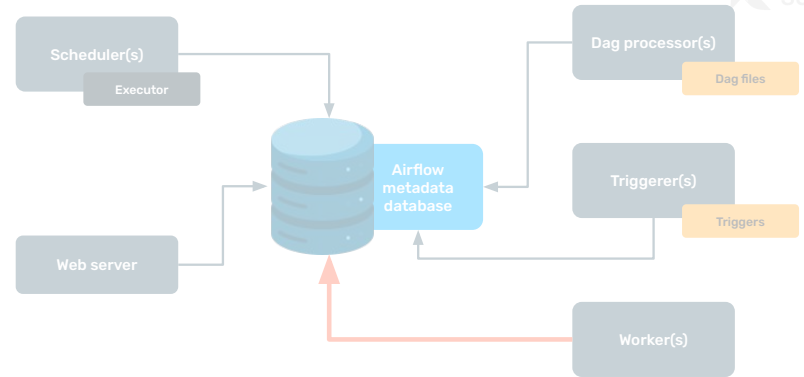


Airflow 2.x: Everything in one place

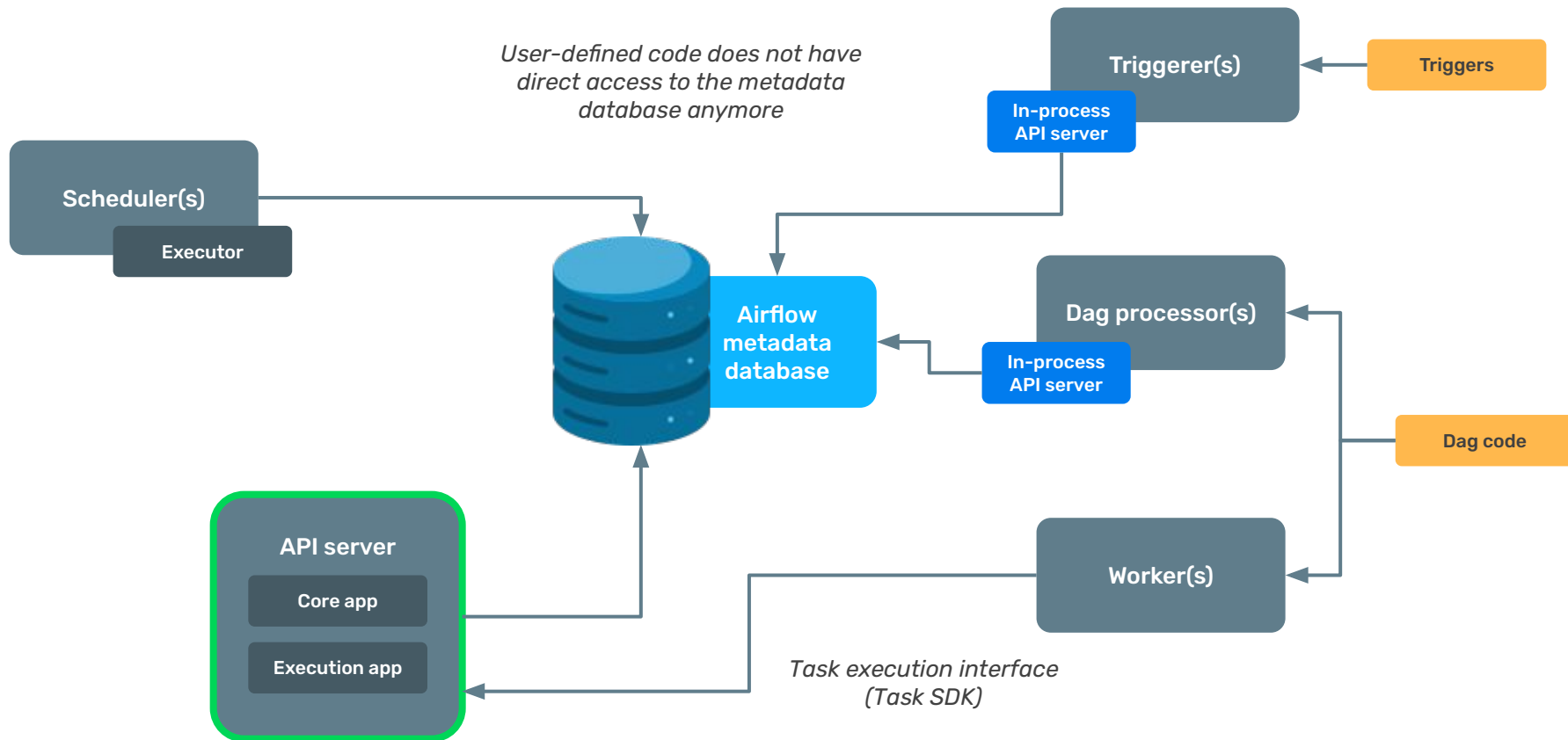




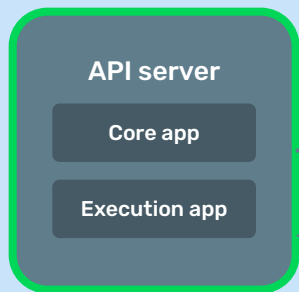
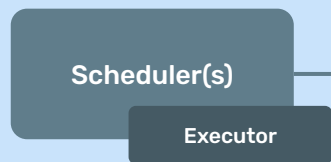
But What About Dag Processor & Triggerer?



Airflow 3 Architecture: Completing the Isolation



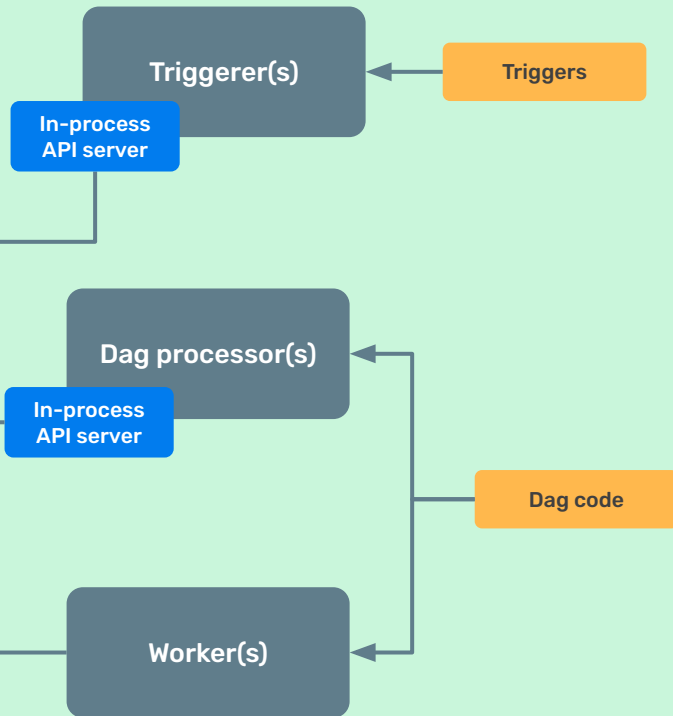
Airflow Server



Airflow
metadata
database

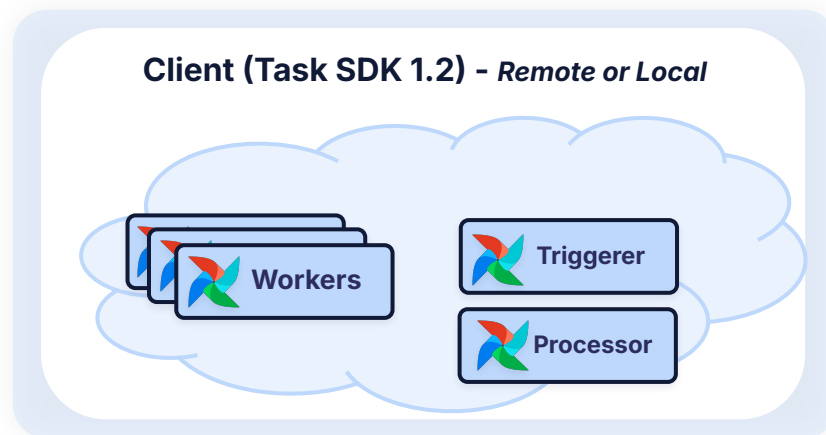
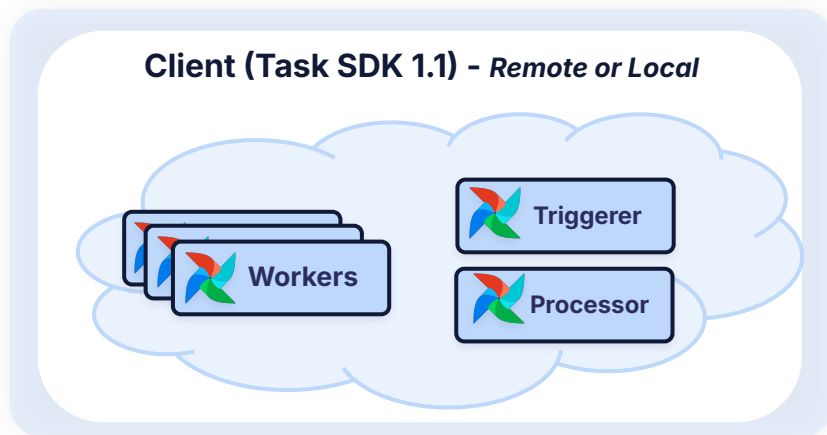
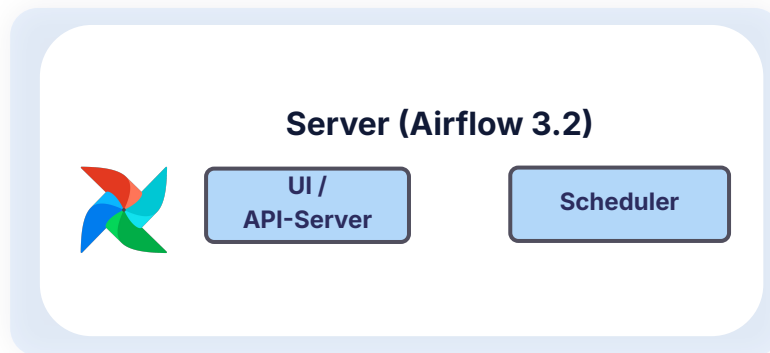
Task execution interface
(Task SDK)

Airflow Client





Enables - *Independent Upgrades*



Run Tasks Anywhere:

Edge Executor

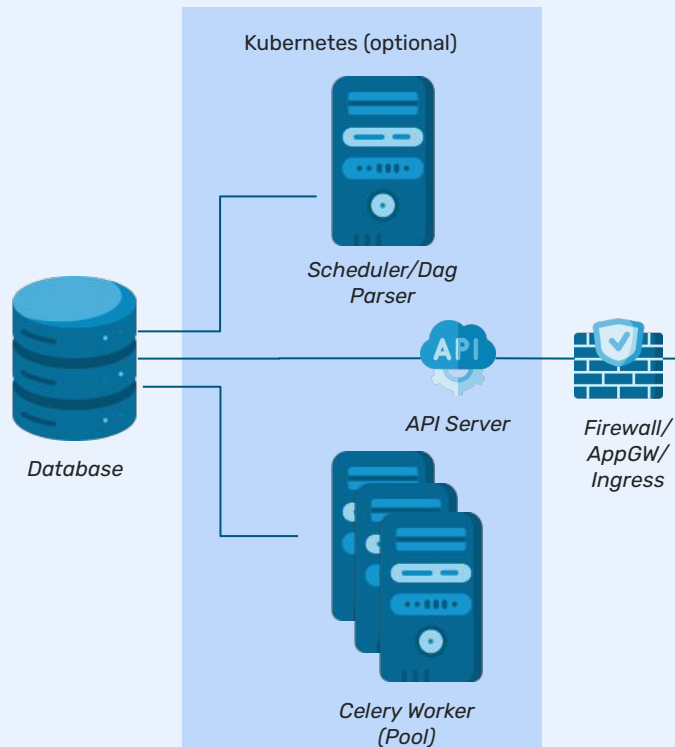
3.0

Jens Scheffler

Cluster Technical Architect @ Bosch
Airflow Committer & PMC Member

Edge Executor

Airflow Main Deployment/Cloud

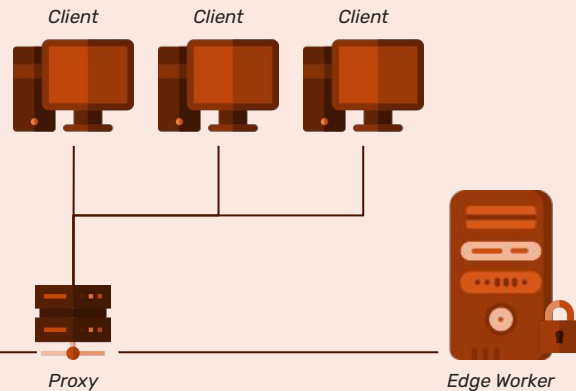


HTTPS



User

Corporate Network



Remote Site



Edge Worker



On-Prem System

Edge Executor - with Airflow 3.1 including UI Plugin



Home



Dags



Assets



Browse



Admin

Edge Worker

Edge Jobs

Worker Name	State	Queues	First Online	Last Heartbeat	Active Jobs	System Information	Operations
bosch.ffm.ad-alliance.biz	idle	• production	1 year ago	1 second ago	0	- airflow_version: 3.1.0 - edge_provider_version: 1.3.1 - concurrency: 8 - free_concurrency: 8	   
dev-abt.de.bosch.com	maintenance mode	• dev	1 year ago	5 seconds ago	0	- airflow_version: 3.1.0 - edge_provider_version: 1.3.1 - concurrency: 8 - free_concurrency: 8	[2025-09-27 20:05] - admin put node into maintenance mode Comment: Moved to production, yeah!   
	offline	• arctis	5 months ago	1 hour ago	0	- airflow_version: 3.1.0 - edge_provider_version: 1.3.1 - concurrency: 8 - free_concurrency: 8	

Variables

Pools

Providers

Plugins

Connections

Config

Edge Worker

Edge Worker Jobs

Edge Executor - with Airflow 3.1 including UI Plugin



Home



Dags



Assets



Browse



Admin

Edge Worker

Edge Jobs

Worker Name State Queues First Online Last Heartbeat Active Jobs System Information Operations

bosch.ffm.ad-alliance.biz

idle

• production

1 year ago

1 second ago

0

- airflow_version: 3.1.0
- edge_provider_version: 1.3.1
- concurrency: 8
- free_concurrency: 8



dev-abt.de.bosch.com



Home



Dags



Assets



Browse



Edge Worker

Edge Jobs

Dag ID	Run ID	Task ID	Map Index	Try Number	State	Queue	Queued DTTM
integration_test	summit_2025-10-07	my_setup	-	1	queued	default	3 seconds ago
integration_test	summit_2025-10-07	docker_task	-	1	queued	default	3 seconds ago
integration_test	summit_2025-10-07	docker_operator_task	-	1	queued	default	3 seconds ago

Variables

Pools

Providers

Plugins

Connections

Config

Edge Worker

Edge Worker Jobs



Start

Dags

Datensets
(Assets)

Browsen

Verwaltung

Doku

Benutzer

Dag ID	Run ID	Task ID	Map Index	Try Number	State	Queue	Queued DTTM	Edge Worker	Last Update
integration_test	summit__2025-10-07	my_setup	-	1	queued	default	8 minutes ago	seattle	8 minutes ago
integration_test	summit__2025-10-07	docker_task	-	1	queued	default	8 minutes ago	seattle	8 minutes ago
integration_test	summit__2025-10-07	docker_operator_task	-	1	queued	default	8 minutes ago	seattle	8 minutes ago

Airflow CTL

Buğra Öztürk

Apache Airflow Committer

Senior Data Engineer

Mollie

3.1

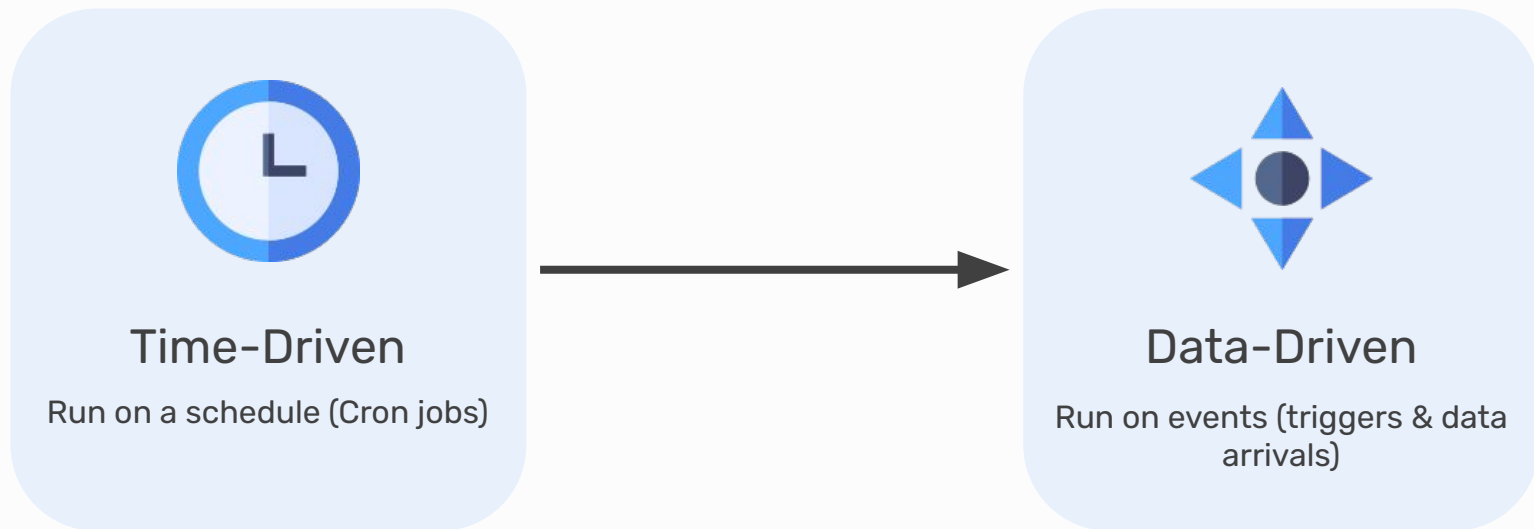


Run Tasks at any Time

Constance Martineau
Vincent Beck

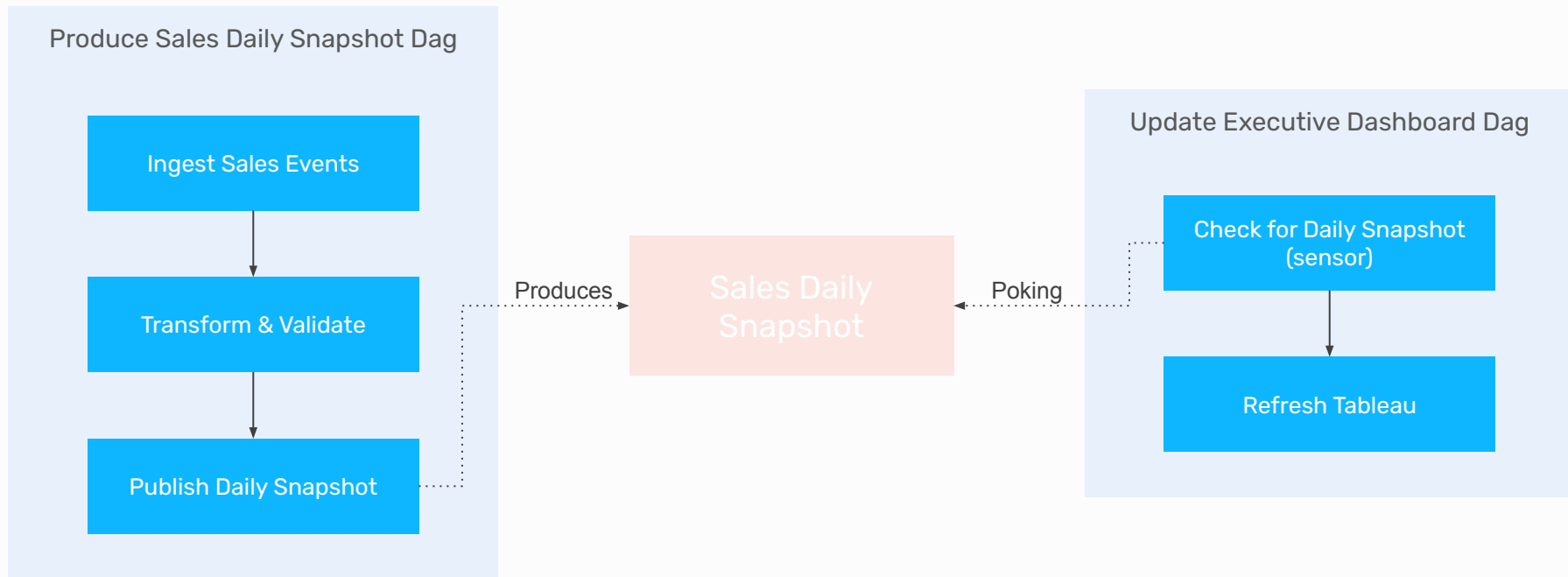
3.1

From Time-Driven to Data-Driven

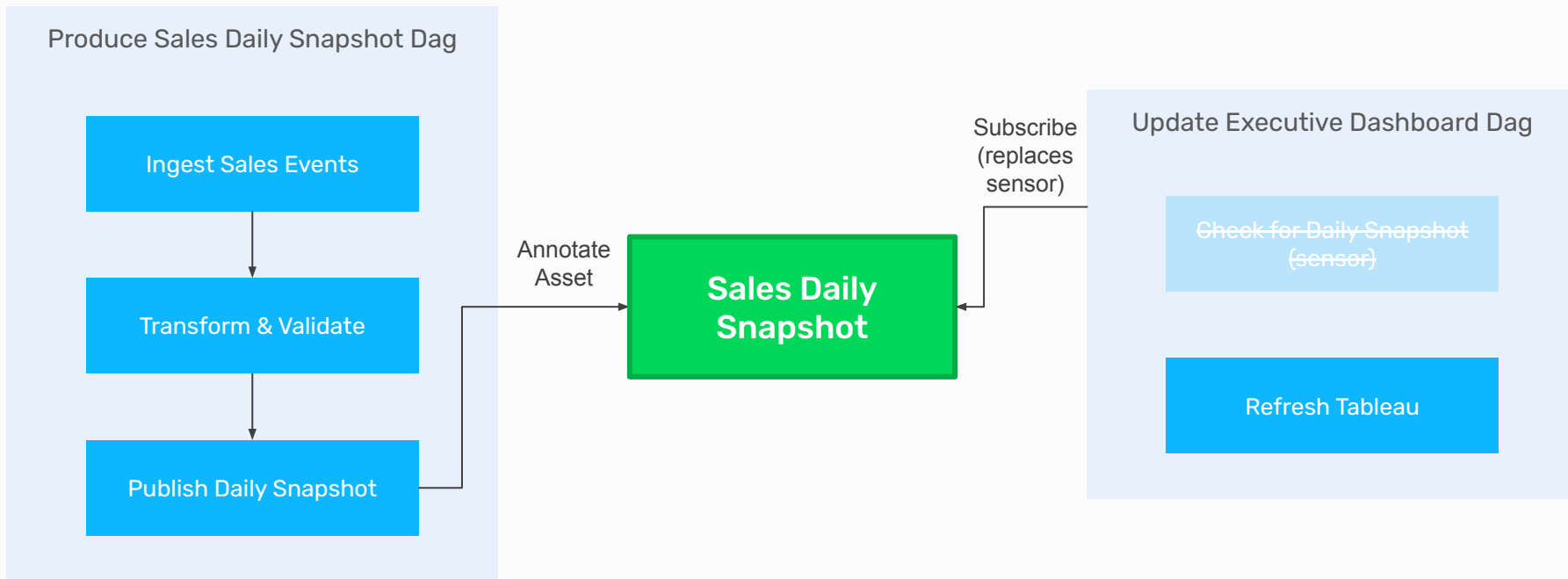


Stop waiting for midnight. Start reacting in (near) real time.

Assets: Orchestrate Around Data



Assets: Orchestrate Around Data



Assets: Orchestrate Around Data

```
from airflow.sdk import asset
```

```
@asset(schedule='@daily')
def sales_daily_snapshot():
    # ingest sales events
    # transform & validate
    # publish daily snapshot
```

```
@asset(schedule=[sales_daily_snapshot])
def executive_dashboard():
    # refresh tableau
```

Equivalent (and matter of
personal preference)



```
from airflow.sdk import Asset, dag, task

sales_daily_snapshot = Asset(name='sales_daily_snapshot')

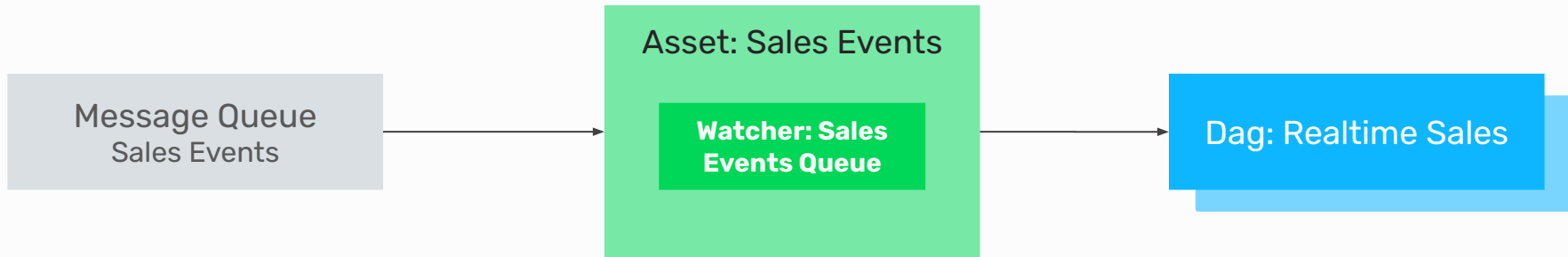
@dag(schedule='@daily')
def produce_sales_daily_snapshot_dag():
    @task(outlets=[sales_daily_snapshot])
    def sales_daily_snapshot_task():
        # ingest sales events
        # transform & validate
        # publish daily snapshot
        sales_daily_snapshot_task()
    sales_daily_snapshot_dag()

@dag(schedule=sales_daily_snapshot)
def update_executive_dashboard_dag():
    @task(outlets=Asset(name='executive_dashboard'))
    def refresh_tableau():
        # refresh tableau
        refresh_tableau()
    update_executive_dashboard_dag()
```

Asset as the focus, not the process

But isn't that first Dag still
running on a schedule?

Event-Driven Scheduling (Asset Watchers)



Queue → Watcher (polls for new events) → Asset → Dag

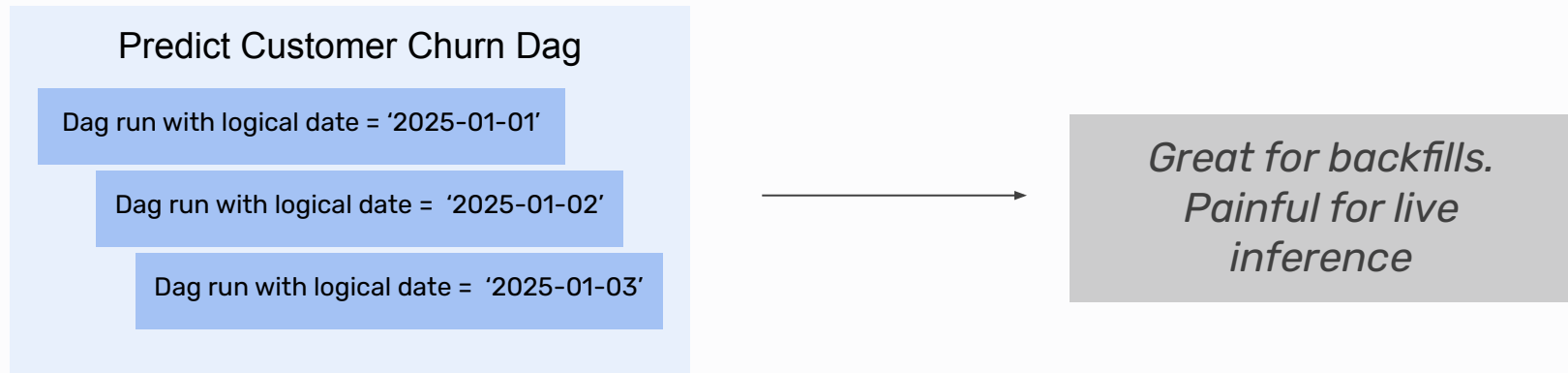
Event-Driven Scheduling (Asset Watchers)

```
from airflow.sdk import Asset, AssetWatcher, dag
from airflow.providers.common.messaging.triggers.msg_queue import MessageQueueTrigger

# Define a trigger that listens to an external message queue (AWS SQS in this case)
trigger = MessageQueueTrigger(scheme='sqs', queue='sales-events')
# Define an asset that watches for messages on the queue
sales_events_asset = Asset('sales_events', watchers=[AssetWatcher(name='sqs_sales_queue', trigger=trigger)])

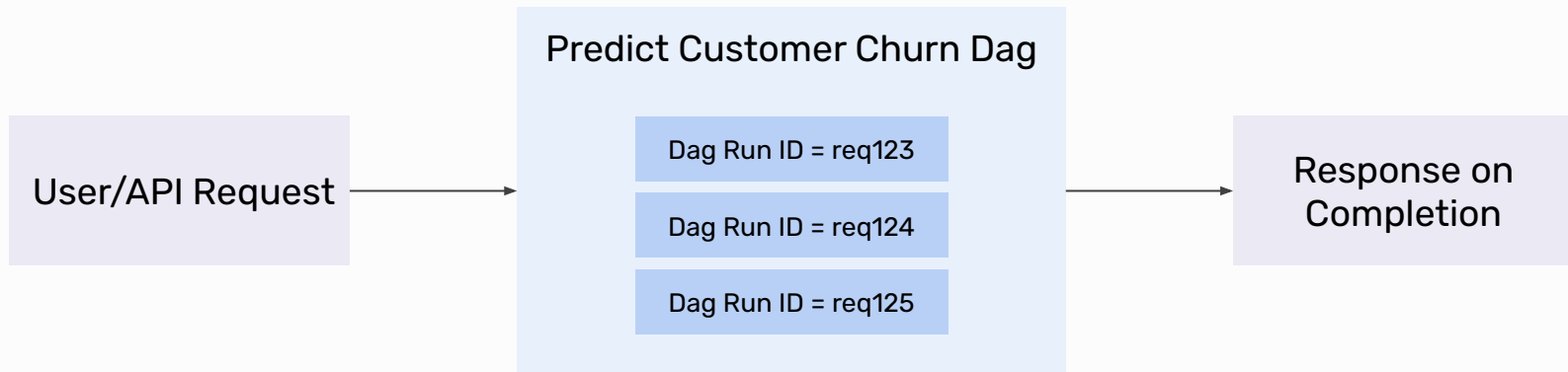
@dag(schedule=[sales_events_asset])
def realtime_sales():
    # rest of dag logic
    ...
    realtime_sales()
```

Inference Execution: The Old Way



Traditional Dags have **fixed Logical Dates**

Inference Execution: The New Way



Synchronous Execution:

Run → Predict → Respond, all in one flow

API Response Example

```
{
  "state": "success",
  "results": {
    "prediction": "Customer likely to churn"
  }
}
```

Event-Driven Scheduling Demo



Welcome



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Stats

✕ 0

Failed Dags >

~ 0

Running Dags >

⚡ 1

Active Dags >

☆ First 10 favorite Dags

No favorites yet. Click the star icon next to a Dag in the list to add it to your favorites.

Health

✓ MetaDatabase

✓ Scheduler

✓ Triggerer

✓ Dag Processor

Pool Slots

[Manage Pools](#)

✓ 128

History



Last 24 Hours



2025-09-28 22:41:30 - 2025-09-29 22:41:30

0 Dag Runs

0 Queued

0%

0 Running

0%

0 Success

0%

0 Failed

0%

0 Task Instances

0 Asset Events

Newest First



No Asset Events found.



Dag
example_keynote1



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Options ▾



example_keynote1  

☆ Favorite Dag



Schedule

Latest Run

Next Run

Owner

Tags

Latest Dag Version

 keynote_kafka_topic1

airflow

v1

Overview

Runs

Tasks

Calendar

Backfills

Audit Log

Code

Details

Parsed at: 2025-09-29 22:41:46 Parse Duration: 0.07s

v1 2025-09-29 22:41:15 ▾

 Copy

Wrap

```
1 from __future__ import annotations
2
3 from airflow.providers.common.messaging.triggers.msg_queue import MessageQueueTrigger
4 from airflow.sdk import task, chain, DAG, Asset, AssetWatcher
5
6 trigger = MessageQueueTrigger(
7     scheme="kafka",
8     topics=["airflow-keynote1"],
9 )
10 asset = Asset("keynote_kafka_topic1", watchers=[
11     AssetWatcher(name="keynote_kafka_topic_watcher1", trigger=trigger)
12 ])
13
14 with DAG(
15     dag_id="example_keynote1",
16     schedule=[asset],
17 ):
18     @task
19     def my_task():
20         print("Hello world")
21
22     chain(my_task())
23
```



Dag
example_keynote1



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Options ▾



example_keynote1



Schedule

Latest Run

Next Run

Owner

Tags

Latest Dag Version

keynote_kafka_topic1

airflow

v1

☆ Favorite Dag



Overview

Runs

Tasks

Calendar

Backfills

Audit Log

Code

Details

Parsed at: 2025-09-29 22:41:46 Parse Duration: 0.07s

v1 2025-09-29 22:41:15 ▾

Copy

Wrap

```
1 from __future__ import annotations
2
3 from airflow.providers.common.messaging.triggers.msg_queue import MessageQueueTrigger
4 from airflow.sdk import task, chain, DAG, Asset, AssetWatcher
5
6 trigger = MessageQueueTrigger(
7     scheme="kafka",
8     topics=["airflow-keynote1"],
9 )
10 asset = Asset("keynote_kafka_topic1", watchers=[
11     AssetWatcher(name="keynote_kafka_topic_watcher1", trigger=trigger)
12 ])
13
14 with DAG(
15     dag_id="example_keynote1",
16     schedule=[asset],
17 ):
18     @task
19     def my_task():
20         print("Hello world")
21
22     chain(my_task())
23
```



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Dag ☐ example_keynote1 / Dag Run 2025-09-29 22:42:36



Options ▾



2025-09-29 22:42:36 ✓ Success

Add a note

Clear Dag Run

Mark Dag Run as... ▾

☰

Run Type	Start Date	End Date	Duration	Dag Version(s)
asset_triggered	2025-09-29 22:42:36	2025-09-29 22:42:38	2.13s	v1

Task Instances **Asset Events** Audit Log Code Details

1 Source Asset Event

2025-09-29 22:42:36

keynote_kafka_topic1

Source: Trigger

Triggered Dag Run: ✓ example_keynote1

▼ { 1 item

```
"payload" : "New message"
```



Dag

example_keynote2



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User



Options



my_task



example_keynote2



Search Dags

**K



Trigger



Favorite Dag



Schedule

Latest Run

Next Run

Owner

Tags

Latest Dag Version

keynote_kafka_topic2

airflow

v1

Overview

Runs

Tasks

Calendar

Backfills

Audit Log

Code

Details

Parsed at: 2025-09-29 23:01:28 Parse Duration: 0.09s

v1 2025-09-29 22:41:15

Copy

Wrap

```
1 from __future__ import annotations
2
3 from airflow.providers.common.messaging.triggers.msg_queue import MessageQueueTrigger
4 from airflow.sdk import task, chain, DAG, Asset, AssetWatcher
5
6 trigger = MessageQueueTrigger(
7     scheme="kafka",
8     topics=["airflow-keynote2"],
9 )
10 asset = Asset("keynote_kafka_topic2", watchers=[
11     AssetWatcher(name="keynote_kafka_topic_watcher2", trigger=trigger)
12 ])
13
14 with DAG(
15     dag_id="example_keynote2",
16     schedule=[asset],
17 ):
18     @task
19     def my_task(triggering_asset_events):
20         for event in triggering_asset_events[asset]:
21             print(event.extra["payload"])
22
23     chain(my_task())
24
```



Dag
example_keynote2



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User



Options ▾



my_task



example_keynote2



Favorite Dag



Schedule

Latest Run

Next Run

Owner

Tags

Latest Dag Version

keynote_kafka_topic2

airflow

v1

Overview

Runs

Tasks

Calendar

Backfills

Audit Log

Code

Details

Parsed at: 2025-09-29 23:01:28 Parse Duration: 0.09s

v1 2025-09-29 22:41:15 ▾

Copy

Wrap

```
1 from __future__ import annotations
2
3 from airflow.providers.common.messaging.triggers.msg_queue import MessageQueueTrigger
4 from airflow.sdk import task, chain, DAG, Asset, AssetWatcher
5
6 trigger = MessageQueueTrigger(
7     scheme="kafka",
8     topics=["airflow-keynote2"],
9 )
10 asset = Asset("keynote_kafka_topic2", watchers=[
11     AssetWatcher(name="keynote_kafka_topic_watcher2", trigger=trigger)
12 ])
13
14 with DAG(
15     dag_id="example_keynote2",
16     schedule=[asset],
17 ):
18     @task
19     def my_task(triggering_asset_events):
20         for event in triggering_asset_events[asset]:
21             print(event.extra["payload"])
22
23     chain(my_task())
24
```



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Dag ☐ example_keynote2 / Dag Run ☒ 2025-09-29 23:01:55 / Task my_task



Options ▾



my_task 2025-09-29 23:01:56 ✓ Success

Operator	Start Date	End Date	Duration	Dag Version
@task	2025-09-29 23:01:56	2025-09-29 23:01:58	1.61s	v1

Logs Rendered Templates XCom Asset Events Audit Log Code Details

All Log Levels ▾ All Sources ▾

```
Log message source details sources=["/root/airflow/logs/dag_id=example_keynote2/run_id=asset_triggered__2025-09-30T03:01:55
2 [2025-09-29 23:01:57] INFO - DAG bundles loaded: dags-folder source=airflow.dag_processing.bundles.manager.DagBundlesM
3 [2025-09-29 23:01:57] INFO - Filling up the DagBag from /files/dags/example_keynote2.py source=airflow.dag_processing.
4 [2025-09-29 23:01:58] INFO - My payload source=task.stdout
5 [2025-09-29 23:01:58] INFO - Done. Returned value was: None source=airflow.task.operators.airflow.providers.standard.d
```



Home



Dags



Assets



Browse



Admin



Plugins



Docs

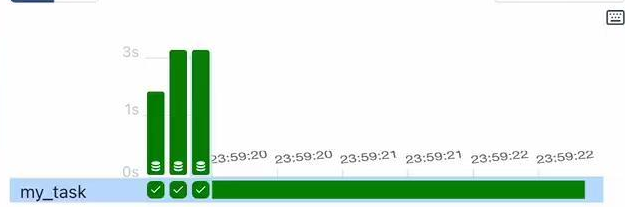


User

Dag ☐ example_keynote2 / Dag Run ☒ 2025-09-29 23:59:19 / Task my_task



Options ▾



Search Dags **K Trigger

my_task 2025-09-29 23:59:20 Success

Add a note Clear Task Instance Mark Task Instance as...

Operator	Start Date	End Date	Duration	Dag Version
@task	2025-09-29 23:59:20	2025-09-29 23:59:23	2.96s	v1

Logs Rendered Templates XCom Asset Events Audit Log Code Details

All Log Levels ▾ All Sources ▾ Log Settings ↗

Log message source details sources=["/root/airflow/logs/dag_id=example_keynote2/run_id=asset_triggered__2025-09-30T03:59:19"]

```
2 [2025-09-29 23:59:21] INFO - DAG bundles loaded: dags-folder source=airflow.dag_processing.bundles.manager.DagBundlesM
3 [2025-09-29 23:59:21] INFO - Filling up the DagBag from /files/dags/example_keynote2.py source=airflow.dag_processing.
4 [2025-09-29 23:59:23] INFO - New message source=task.stdout
5 [2025-09-29 23:59:23] INFO - Done. Returned value was: None source=airflow.task.operators.airflow.providers.standard.d
```



Dag
example_keynote3



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Options ▾



example_keynote3

☆ Favorite Dag



Schedule	Latest Run	Next Run	Owner	Tags	Latest Dag Version
keynote_kafka_topic3			airflow		v1

Overview Runs Tasks Calendar Backfills Audit Log Code Details

```
1 from __future__ import annotations
2
3 from airflow.providers.common.messaging.triggers.msg_queue import MessageQueueTrigger
4 from airflow.sdk import task, chain, DAG, Asset, AssetWatcher
5
6 kafka_trigger = MessageQueueTrigger(
7     scheme="kafka",
8     topics=["airflow-keynote3"],
9 )
10 sqs_trigger = MessageQueueTrigger(
11     scheme="sqs",
12     queue="https://sqs.us-east-1.amazonaws.com/722404908466/Test"
13 )
14 asset = Asset("keynote_kafka_topic3", watchers=[
15     AssetWatcher(name="keynote_kafka_topic_watcher3", trigger=kafka_trigger),
16     AssetWatcher(name="keynote_sqs_topic_watcher3", trigger=sqs_trigger)
17 ])
18
19 with DAG(
20     dag_id="example_keynote3",
21     schedule=[asset],
22 ):
23     @task
24     def my_task(triggering_asset_events):
25         for event in triggering_asset_events[asset]:
26             print(event.extra["payload"])
27
28     chain(my_task())
29
```



Dag
example_keynote3



Home



Dags



Assets



Browse



Admin



Plugins



Docs



User

Options ▾



example_keynote3 ☐

Search Dags ****K**

▶ Trigger

☆ Favorite Dag



Schedule	Latest Run	Next Run	Owner	Tags	Latest Dag Version
keynote_kafka_topic3			airflow		v1

Overview Runs Tasks Calendar Backfills Audit Log Code Details

```
1 from __future__ import annotations
2
3 from airflow.providers.common.messaging.triggers.msg_queue import MessageQueueTrigger
4 from airflow.sdk import task, chain, DAG, Asset, AssetWatcher
5
6 kafka_trigger = MessageQueueTrigger(
7     scheme="kafka",
8     topics=["airflow-keynote3"],
9 )
10 sqs_trigger = MessageQueueTrigger(
11     scheme="sqs",
12     queue="https://sqs.us-east-1.amazonaws.com/722404908466/Test"
13 )
14 asset = Asset("keynote_kafka_topic3", watchers=[
15     AssetWatcher(name="keynote_kafka_topic_watcher3", trigger=kafka_trigger),
16     AssetWatcher(name="keynote_sqs_topic_watcher3", trigger=sqs_trigger)
17 ])
18
19 with DAG(
20     dag_id="example_keynote3",
21     schedule=[asset],
22 ):
23     @task
24     def my_task(triggering_asset_events):
25         for event in triggering_asset_events[asset]:
26             print(event.extra["payload"])
27
28     chain(my_task())
29
```

What's Next: Expanding the Asset Model



Partitions & Watermarks → Define evolving assets by da, hour or key; track freshness and history



Targeted Updates → Refresh only what's new or stale, instead of reprocessing everything



Asset Validations → Validation and status surfaced directly in the UI



At-a-Glance Visibility → Validation and status surfaced directly in the UI



End-to-End Event Tracking → Trace events from message queue → asset → downstream dags

Run Tasks in any Language

Ash Berlin Taylor

Member of Apache Airflow PMC; and
Director, Airflow Engineering
Astronomer

3.1

Tasks in any language: Goal



Enable integrations with applications

- Pull / push data from/to existing apps
- Expand Airflow's role as "integration platform"
- Enable non-python teams to leverage Airflow



First Language: Golang, Java to follow

- Stepping stone for compiled language support



Non-goal at this time:

- Writing entire workflows i.e. Dags in other languages

Introducing Go SDK: 1.0.0-beta1



Airflow Tasks in Go



Full power of Airflow

- Connections, Variables, XCom and more



Feels like natural part of Go ecosystem

Polyglot Dag Demo

Airflow

x

+

←

→

↺

http://localhost:28080/plugin/edge_worker

150%

Home

Dags

Assets

Browse

Admin

Docs

User

Edge Worker

Edge Jobs

Worker Name	State	Queues	First Online	Last Heartbeat	Active Jobs	System Information	Operations
a5a5a7840990	idle	• default	4 hours ago	3 seconds ago	0	- airflow_version: 3.2.0 - edge_provider_version: 1.3.1 - concurrency: 8 - free_concurrency: 8	



Edge Worker

Edge Jobs

Worker Name

State

Queues

First Online

Last Heartbeat

Active Jobs

System Information

Operations

a5a5a7840990

 idle

- default

4 hours ago

2 seconds ago

0

- airflow_version: 3.2.0
- edge_provider_version: 1.3.1
- concurrency: 8
- free_concurrency: 8



ip-10-6-31-1

 idle

- seattle

3 seconds ago

3 seconds ago

0

- airflow_version: 3.2.0
- concurrency: 16
- edge_provider_version: 1.3.1
- free_concurrency: 16
- go_version: go1.24.6



 Home

[Home](#)

Dags



Assets



Browse



Admin



Docs



User

⌂

Home

📅

Dags

🗄️

Assets

🌐

Browse

⚙️

Admin

📖

Docs

👤

User

🌈

Airflow

⌵

🔍

Search Dags

⌨️

▶️

Trigger

Dag

multi_language

Dag Run

2025-10-03 21:08:23

Task

load

Options

⌵

6s

3s

0s

extract

transform

load

✓

✓

✓

📄

load 2025-10-03 21:08:30

✓ Success

📝

Add a note

🔄

Clear Task Instance

🏷️

Mark Task Instance as...

⌵

Operator	Start Date	End Date	Duration	Dag Version
@task	2025-10-03 21:08:30	2025-10-03 21:08:32	1.79s	v1

📄

Logs

📄

Rendered Templates

📄

XCom

📄

Asset Events

📄

Audit Log

📄

Code

📄

Details

1

from airflow.sdk import dag, task

2

3

@task.stub(queue="seattle")

4

def extract(): ...

5

6

@task.stub(queue="seattle")

7

def transform(order_data_dict: dict): ...

8

9

@dag()

10

def multi_language():

11

12

@task()

13

def load(order_summary: float):

14

print(f"Total order value is: {order_summary['total_order_value']:.2f}")

15

16

order_data = extract()

17

order_summary = transform(order_data)

18

load(order_summary)

19

20

multi_language()

21

multi_language

Dag Run

2025-10-03 21:08:23

Options

2025-10-03 21:08:23

Success

Add a note

Clear Dag Run

Mark Dag Run as...

Logical Date	Run Type	Start Date	End Date	Duration	Triggering User Name	Dag Version(s)
2025-10-03 21:08:23	manual	2025-10-03 21:08:26	2025-10-03 21:08:32	6.35s	admin	v1

Task Instances

Asset Events

Audit Log

Code

Details

Parsed at: 2025-10-03 21:06:34

Parse Duration: 0.05s

v1

2025-10-03 17:29:41

Copy

Wrap

```
1 from airflow.sdk import dag, task
2
3 @task.stub(queue="seattle")
4 def extract(): ...
5
6 @task.stub(queue="seattle")
7 def transform(order_data_dict: dict): ...
8
9 @dag()
10 def multi_language():
11
12     @task()
13     def load(order_summary: float):
14         print(f"Total order value is: {order_summary['total_order_value']:.2f}")
15
16     order_data = extract()
17     order_summary = transform(order_data)
18     load(order_summary)
19
```

Home

Dags

Assets

Browse

Admin

Docs

User

extract

transform

load

multi_language

2025-10-03 21:08:23

Options

2025-10-03 21:08:23

Success

Add a note

Clear Dag Run

Mark Dag Run as...

Logical Date	Run Type	Start Date	End Date	Duration	Triggering User Name	Dag Version(s)
2025-10-03 21:08:23	manual	2025-10-03 21:08:26	2025-10-03 21:08:32	6.35s	admin	v1

Task Instances

Asset Events

Audit Log

Code

Details

```
1 from airflow.sdk import dag, task
2
3 @task.stub(queue="seattle")
4 def extract(): ...
5
6 @task.stub(queue="seattle")
7 def transform(order_data_dict: dict): ...
8
9 @dag()
10 def multi_language():
11
12     @task()
13     def load(order_summary: float):
14         print(f"Total order value is: {order_summary['total_order_value']:.2f}")
15
16     order_data = extract()
17     order_summary = transform(order_data)
18     load(order_summary)
19
20 multi_language()
21
```

Home

Dags

Assets

Browse

Admin

Docs

User

extract

transform

load

Home

Dags

Assets

Browse

Admin

Docs

User

Dag

multi_language

Dag Run

2025-10-03 21:08:23

Task

extract

Options

extract

transform

load

extract 2025-10-03 21:08:27

Success

Add a note

Clear Task Instance

Mark Task Instance as...

Operator	Start Date	End Date	Duration	Dag Version
@task.stub	2025-10-03 21:08:27	2025-10-03 21:08:28	1.21s	v1

Logs

Rendered Templates

XCom

Asset Events

Audit Log

Code

Details

All Log Levels

Log Settings

Log message source details sources=["http://host.docker.internal:8793/log/dag_id=multi_language/run_id>manual__2025-10-03T20-08-25.751290+00-00/task_id=extract/attempt=1.log"]

2 INFO - Task succeeded

[Home](#)

Dags



Assets



Browse



Admin



Docs



User



Options ▾



extract
transform
load

 Add a note

↺ Clear Task Instance

Mark Task Instance as... ▾

Operator	Start Date	End Date	Duration	Dag Version
@task.stub	2025-10-03 21:08:29	2025-10-03 21:08:29	0.04s	v1

+ Filter

Key	Dag	Run ID	Task ID	Map Index	Value
return_value	multi_language	manual__2025-10-03T20:08:25.751290+00:00	transform	-1	<pre> { 1 item " total_order_value " : 1236.7 } </pre>

Compiled tasks: are stubbed

Operator "stub"
definition in calling Dag

Still need a python file

```
from airflow.sdk import task, DAG

@task.stub
def extract(output: str, tables: list[str]):
    ...

with DAG("keynote"):
    ...
    extract("loc1", get_tables())
```

Compiled tasks: are registered

Real operator code in
Golang in separate file

Compiled into Go
binary



```
func (m *myBundle) RegisterDags(dagbag v1.Registry) error {  
    dag := dagbag.AddDag("keynote_dag")  
    dag.AddTask(extract)  
  
    return nil  
}  
  
func task1(ctx context.Context, client sdk.Client) (any, error) {  
    conn, err := client.GetConnection(ctx, "test_http")  
    if err != nil  
        return err  
    }  
  
    return map[string]any{  
        "go_version": runtime.Version(),  
        "host": conn.Host,  
    }, nil  
}
```

Which language we should build next?

Come to the next session in this
room

Closing Remarks

Vikram Koka

Chief Strategy Officer

Astronomer

3.1

Airflow 3 is already being widely used

- ① Hundreds of Enterprise deployments running Airflow 3
- ① Summit sessions include: Uber, Qualcomm, and DataDog
- ① Most have taken: Upgrade, then expand approach

Airflow 3: Foundation for more contributions



Rewritten React-based UI:

- UI plugins (Airflow as a platform)
- Translations, expanded contributors



Data Assets:

- Watchers / Common Msg Interface: Event Driven Scheduling

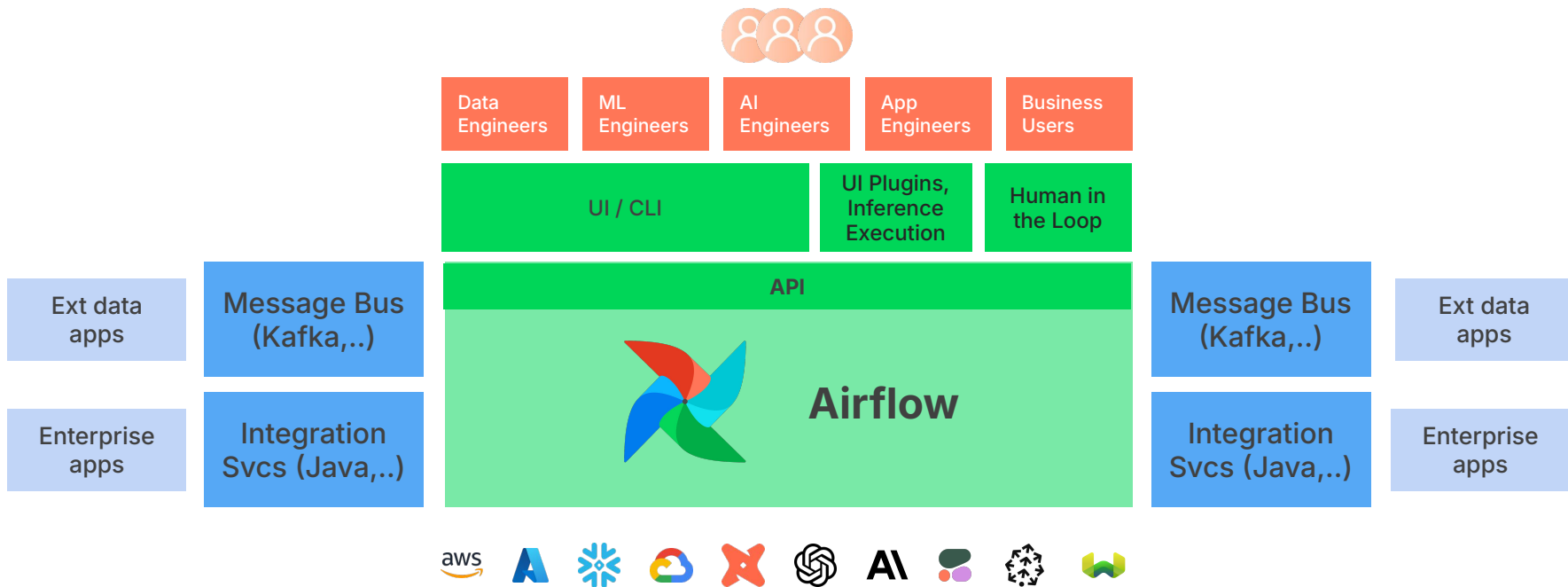


Task SDK in additional languages

Around 3,500 contributors now

Airflow ecosystem

Expanded with Airflow 3



Upgrading to Airflow 3



Min Versions:

- Airflow 2.7 and Python 3.10
- Database cleanup: Can save a lot of time!



Pre-upgrade: use the checks

- Dag Code: `ruff check dags/ --select AIR3`
- For Airflow config, use `airflow config update` from Airflow 2.11



Upgrade:

- Database upgrade
- Startup scripts: web server to api server

Note: Upgrade guide at [Upgrading to Airflow 3](#)

TODO: Airflow 3 - Expert opinions



Best practices:

- Use Assets and Tasks together for best results. Use Assets as interface between Airflows, rather than using "Trigger Dag".
- Use common.abc interfaces whenever possible



Anti-patterns:

- Storing credentials in Airflow db, use Secrets back-end instead
- Using the Airflow metadatabase for large data sharing between Tasks. XComs are for passing small data, ideally references

Dive Deeper into Airflow 3!



Tues Oct 7 • 11:30AM

Security made us do it: Airflow's new Task Execution Architecture

Columbia A

Tues Oct 7 • 12:15PM

Airflow That Remembers: The Dag Versioning Era is here!

Columbia A

Tues Oct 7 • 2:00PM

Unlocking Event-Driven Scheduling in Airflow 3: A New Era of Reactive Data Pipelines

Columbia A

Tues Oct 7 • 4:15PM

EdgeExecutor / Edge Worker - The new option to run anywhere

Columbia C

Wed Oct 8 • 12:30PM

Airflow 3 UI is not enough? Add a Plugin!

Columbia A

Weds Oct 8 • 3:45PM

Assets: Past, Present, Future

Columbia A

Dive Deeper into Airflow 3!



Wed Oct 8 • 10:30AM

**Get started with Airflow 3.0
Workshop**

Room 301

Wed Oct 8 • 12:30PM

**Airflow 3 UI is not enough?
Add a Plugin!**

Columbia A

Weds Oct 8 • 3:45PM

Assets: Past, Present, Future

Columbia A

The 2025 Apache Airflow® Survey is here!



**Fill it out for
a free Airflow 3
Fundamentals or Dag
Authoring in Airflow
3 certification code**



Congratulations

New members of Project Management Committee



Vincent Beck



Wei Lee

New committer



Guan Ming

Questions?

3.1