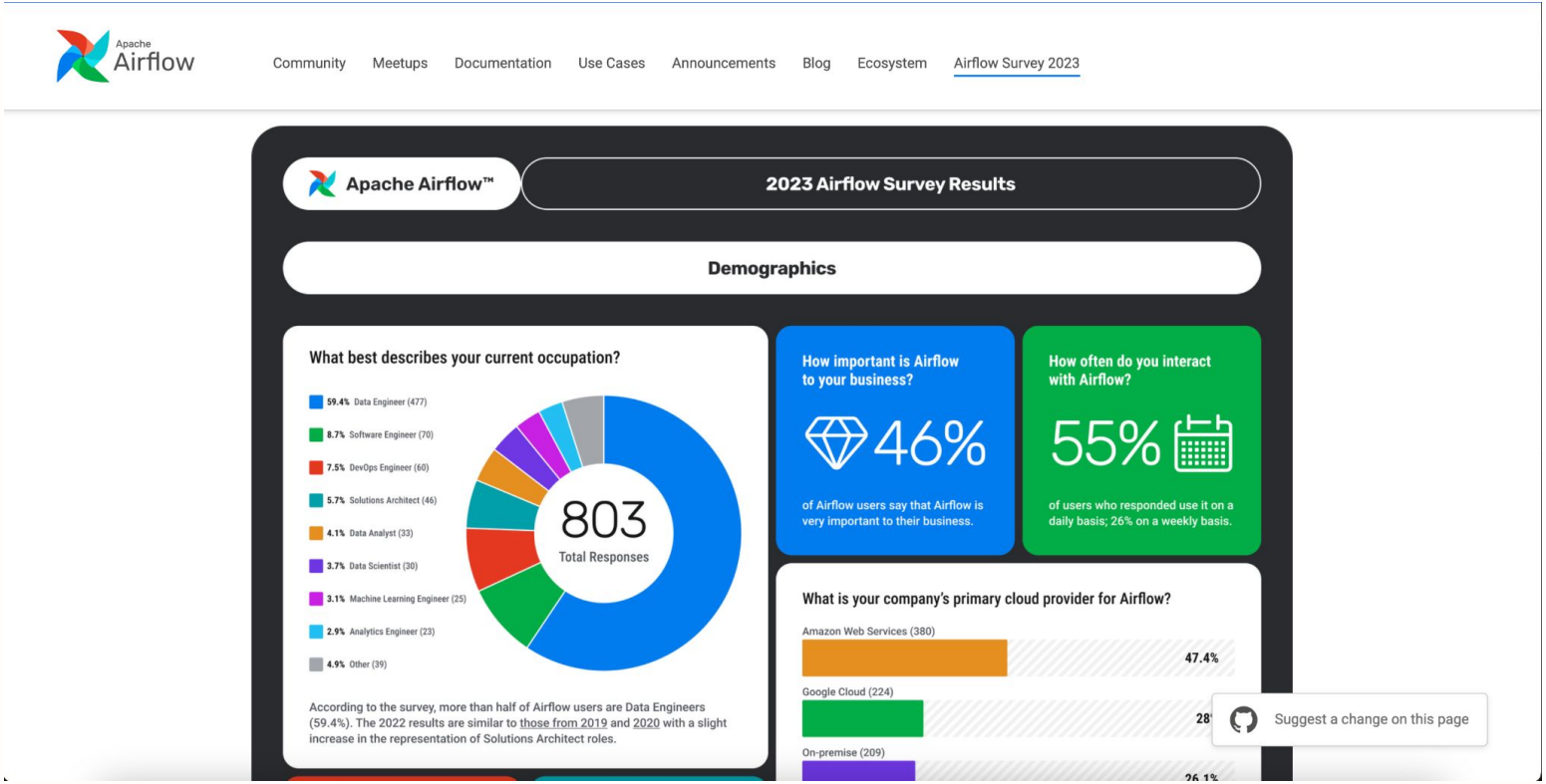
Airflow-as-an-Engine

Lessons from Open-Source Applications Built On Top of Airflow

Ian Moritz, Product @ Astronomer

How do teams interact with Airflow?

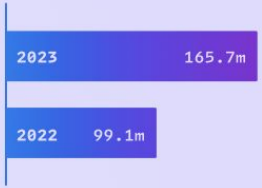
The annual Airflow Survey has some opinions...



....one is that the Airflow adoption is incredible...

67%

year-over-year increase in number of Airflow downloads with over 165.7 million[1]



This exponential growth signifies the platform's appeal and its essential role in addressing the complex data orchestration needs of organizations.

2.8k

Apache Airflow contributors outpacing stalwarts Apache Spark and Apache Kafka[2]

This robust community support ensures that Apache Airflow continues to evolve, adapt, and meet the evolving needs of users.

33k

GitHub stars[3]

This acknowledgment further reinforces Airflow's significance in the open-source ecosystem and the trust placed in it by developers worldwide.

92%

of users would recommend Airflow[4]

It's not just numbers and stars; it's also about the community that stands behind Apache Airflow. When asked about their willingness to recommend Airflow, 92% of respondents expressed their likelihood to do so.

Source: 2024 State of Apache Airflow, 2023 & 2022 Airflow Survey

...and that teams using Airflow use it more and more over time

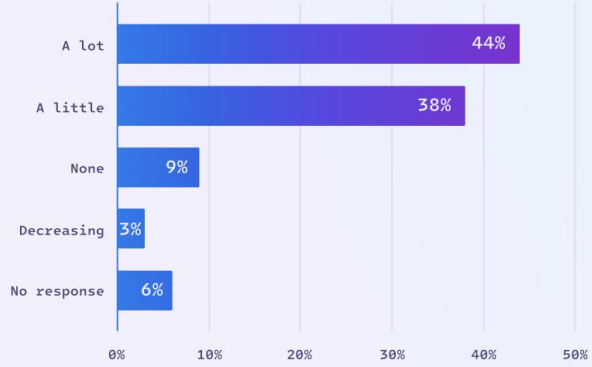
How Teams are Using Airflow

A notable increase in the number of use cases for which teams are using Airflow shows that teams are growing in their confidence of the platform and expanding their vision of what it can accomplish.

82%

of respondents say their team's Airflow use cases have increased over time, with the majority of those saying use cases have increased substantially.

How have your team's use cases with Airflow grown over time?

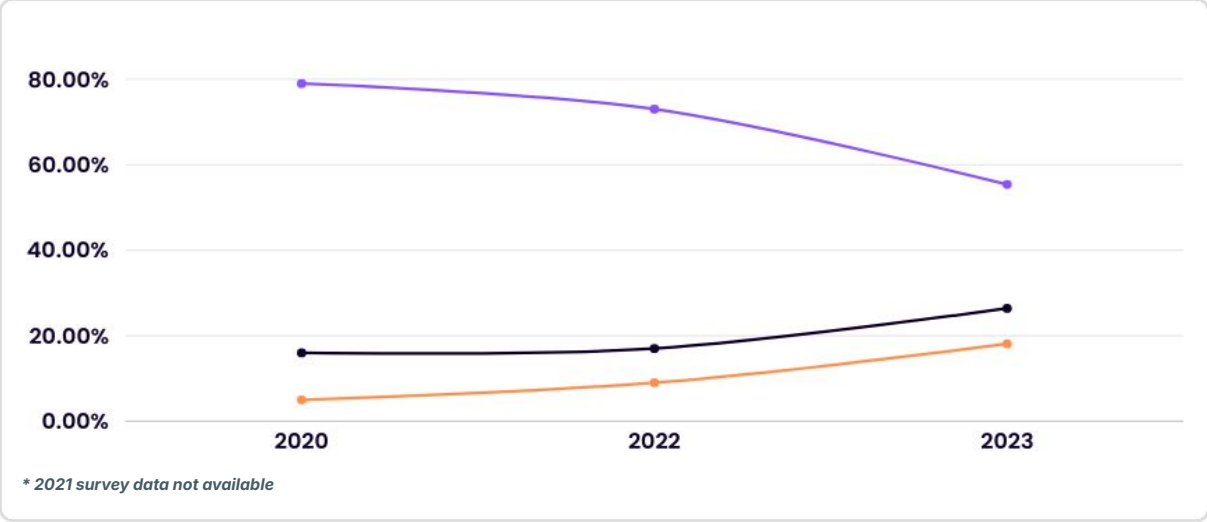


Source: 2023 Apache Airflow Survey, n=761

Source: 2024 State of Apache Airflow, 2023 & 2022 Airflow Survey

But we have a surprising finding from the survey about the users of Airflow!

How often do users interact with Airflow?



Source: 2023, 2022, 2020 Airflow Survey

Who are these users?

Data Engineers

Airflow & data engineering experts who productionize data

~59% of Airflow users

Engineering Professionals

Software, BI, DevOps, Analytics engineers

~33% of Airflow users

Data Professionals

Data analysts, sometimes data scientists

~8% of Airflow users

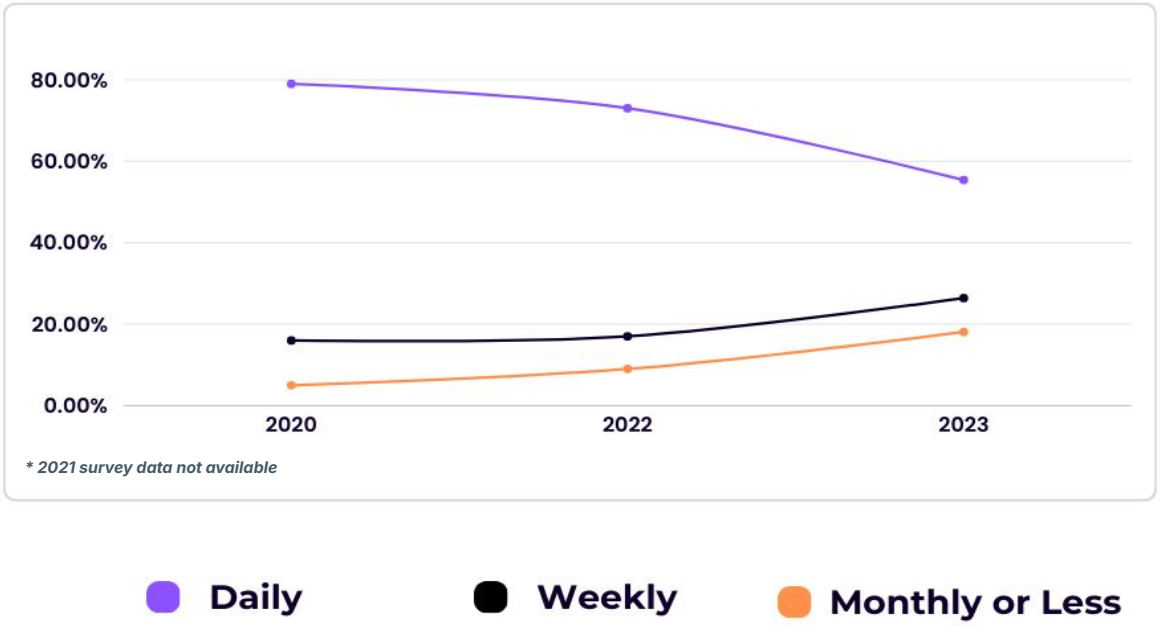
Source: 2024 Airflow Survey

How often do users interact with Airflow?

Of Airflow users, the percentage of daily active users has decreased

Users who interact with Airflow daily decreased from 79% to 55% from 2020 → 2023

Weekly went from 16% to 26%, and Monthly or less went from 5% to 18% in the same period



Source: 2023, 2022, 2020 Airflow Survey

As Airflow adoption grows, some new users are *interacting with it less*

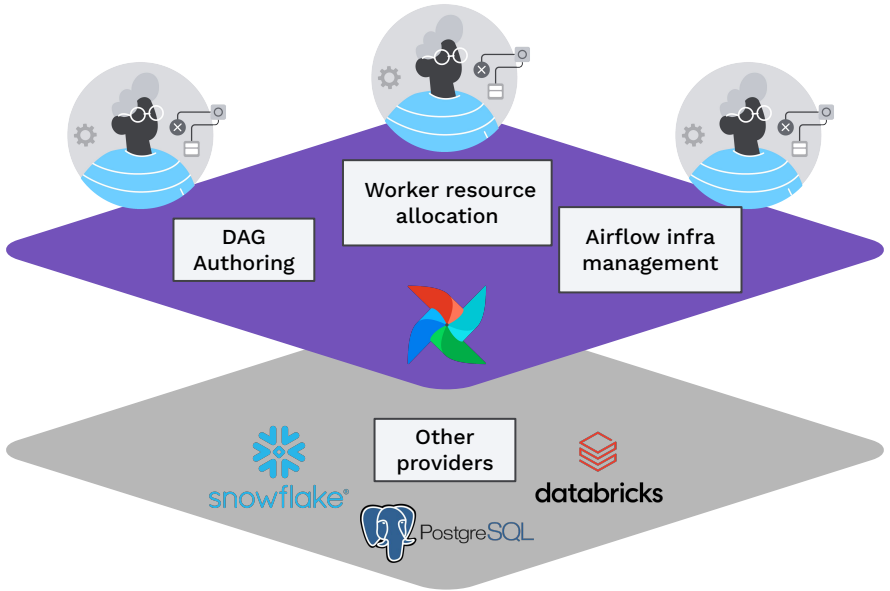
For the audience:
any theories why?

One idea...

Beyond its use as an orchestration platform, Airflow is increasingly an orchestration engine

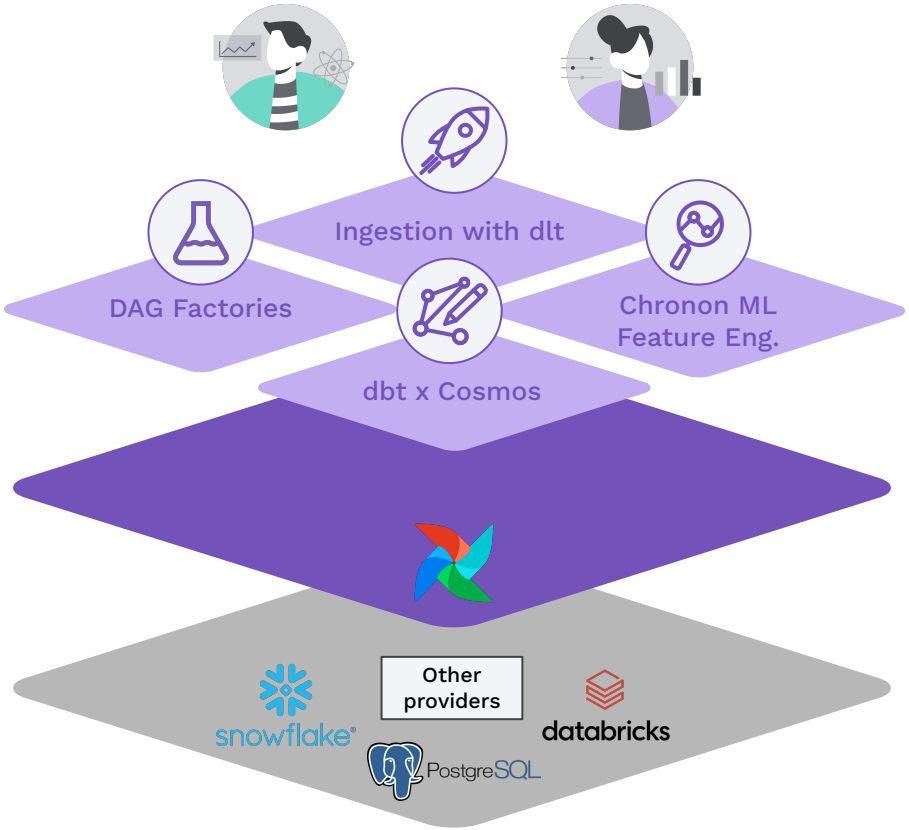
Airflow as a Platform: Users directly interact with Airflow primitives

- **DAG authoring & operator management**
  Directly author DAGs and their tasks using operators, sensors, etc.
- **DevOps**
  Set up CI/CD, deployment strategies, development lifecycles
- **Monitoring & Infra Planning**
  Configure the characteristics of your Airflow deployments, like executors, worker queues, components resources, etc.



Airflow as an Engine: Airflow is a “black box” orchestrator to users

- **Use-case first**
  Users spend most of their time in other tools, and then “deploy” their code to run on Airflow
- **Infrequent direct DAG authoring**
  User frameworks compile configuration to DAGs instead of directly authoring DAGs
- **Airflow is frequently “abstracted away”, user tools like the web server are used less frequently**
  Sometimes, users don’t even know their tools use Airflow under the hood



Three example OSS projects with the engine pattern

Astronomer Cosmos - Data Transformation

Run and observe dbt in Airflow

Just one initialization DAG – that's it

dlt - Data Ingestion

General-purposes framework for data ingestion between sources and sinks

Generate DAGs with their CLI, instead of writing them

Chronon - MLOps

Framework for ML feature management & observability

DAGs generators are provided for you, only occasional changes needed

Astronomer Cosmos

Use and observe dbt in Airflow

Just one initialization DAG – that's it

dbt Core

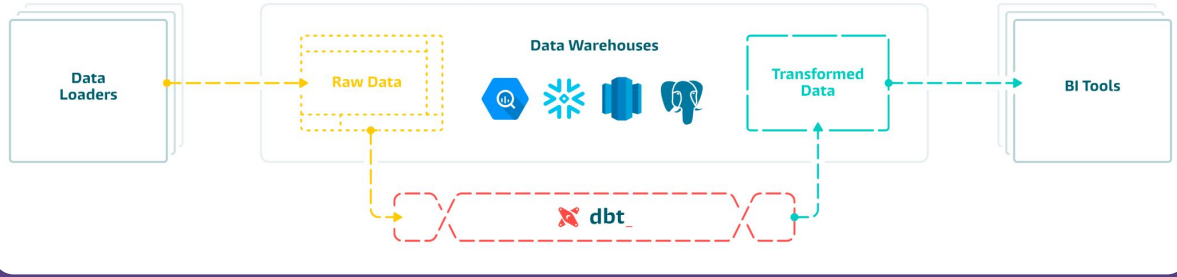
dbt (Data Build Tool) Core is an **open-source** tool for **data transformations** and analysis, using **SQL**

Growing in popularity as a standard for sql analysts and data mart builders

250 Contributors

6K Total Commits

6.5K GitHub Stars



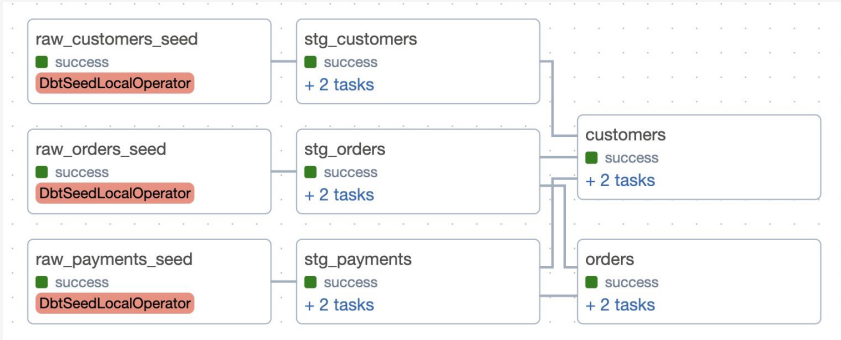
dbt is the T in ELT. Organize, cleanse, denormalize, filter, rename, and pre-aggregate the raw data in your warehouse so that it's ready for analysis.

Cosmos

Google Translate

Text | Images | Documents | Websites

Detect language | dbt | Polish | English | Apache Airflow | Portuguese



Send feedback

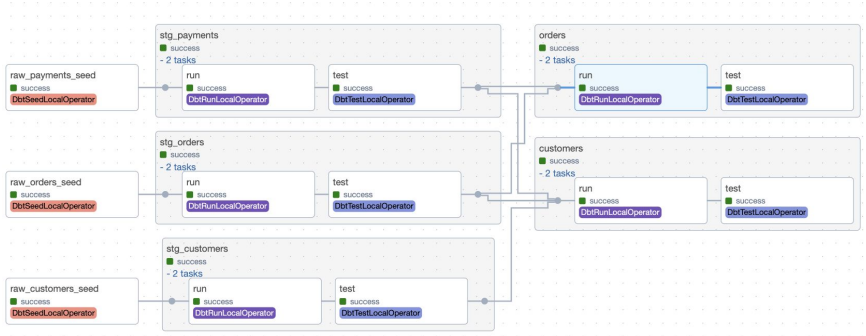
Just **~15 lines of initial code** lets you translate from dbt to Cosmos

```python
import os
from datetime import datetime
from pathlib import Path
from cosmos import DbtDag, ProjectConfig, ProfileConfig
from cosmos.profiles import PostgresUserPasswordProfileMapping

DEFAULT_DBT_ROOT_PATH = Path(__file__).parent / "dbt"
DBT_ROOT_PATH = Path(os.getenv("DBT_ROOT_PATH", DEFAULT_DBT_ROOT_PATH))

profile_config = ProfileConfig(
    profile_name="jaffle_shop",
    target_name="dev",
    profile_mapping=PostgresUserPasswordProfileMapping(
        conn_id="airflow_db",
        profile_args={"schema": "public"},
    ),
)
basic_cosmos_dag = DbtDag(
    project_config=ProjectConfig(
        DBT_ROOT_PATH / "jaffle_shop",
    ),
    profile_config=profile_config,
    schedule_interval="@daily",
    start_date=datetime(2023, 1, 1),
    catchup=False,
    dag_id="basic_cosmos_dag",
)
```

Apache Airflow



With Cosmos, you're less frequently writing DAGs after this initialization step

dlt

General-purposes framework for data ingestion between sources and sinks

Generate DAGs with their CLI, instead of writing them

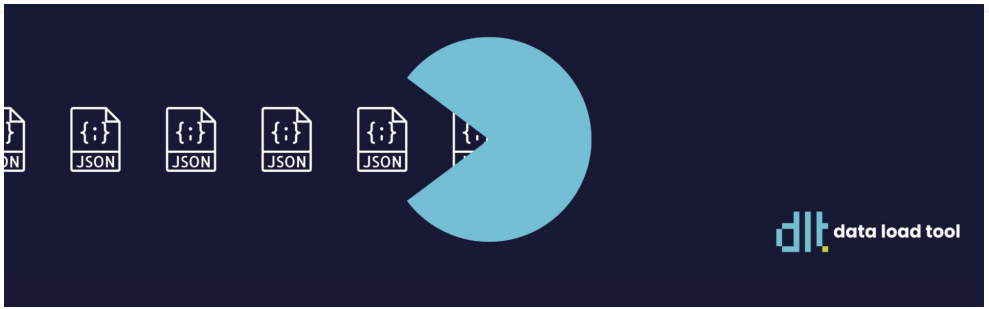
dlt is an open-source Python library to load data from various and often messy data sources into well-structured, live datasets

Growing as a Pythonic way to perform data loads *anywhere* you write Python

65 Contributors

3K Total Commits

2.3K GitHub Stars



dlt is the L in ELT. Load data from different SaaS products, databases, and APIs into destinations

dlt provides tools to **generate & deploy Airflow DAGs** from source code

A single CLI command...

```
dlt deploy my_pipeline.py airflow-composer
```

...generates your DAG

```python
import dlt
from airflow.decorators import dag
from dlt.common import pendulum
from dlt.helpers.airflow_helper import PipelineTasksGroup

# Modify the dag arguments
default_task_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': 'test@test.com',
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 0,

}

@dag(
    schedule=None,
    start_date=pendulum.datetime(2021, 1, 1),
    catchup=False,
    max_active_runs=1,
    default_args=default_task_args
)
def load_data():
    # Set `use_data_folder` to True to store temporary data on the `data` bucket.
    # Use only when it does not fit on the local storage
    tasks = PipelineTaskGroup("pipeline_name", use_data_folder=False, wipe_local_data=True)

    # Import your source from pipeline script
    from pipeline_or_source_script import source

    # Modify the pipeline parameters
    pipeline = dlt.pipeline(
        pipeline_name='pipeline_name',
        dataset_name='dataset_name',
        destination='duckdb',
        dev_mode=False # must be false if we decompose
    )
    # Create the source, the "serialize" decompose option
    # will convert dlt resources into Airflow tasks.
    # Use "none" to disable it.
    tasks.add_run(
        pipeline,
        source(),
        decompose="serialize",
        trigger_rule="all_done",
        retries=0,
        provide_context=True
    )
```

Chronon

Framework for ML feature management & observability

DAGs are provided for you, only occasional changes needed

chronon

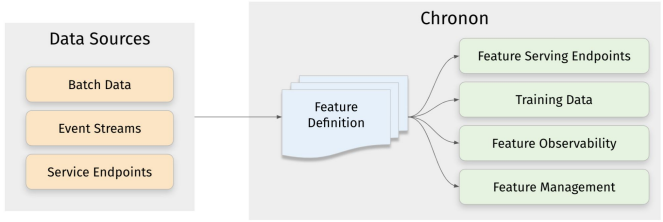
Chronon is an open source feature platform that allows Machine Learning teams t**o easily build, deploy, manage and monitor data pipelines** for machine learning.

Backed by Airbnb and Stripe, Chronon is a fairly new open-source project

22 Contributors

850 Total Commits

704 GitHub Stars



Orchestration for Chronon **involves running the various jobs to compute batch and streaming feature computation**, as well on online/offline consistency measurement.

Chronon **constructs dynamic DAGs for you**– users just add their own Airflow deployment

```python
import helpers
from constants import CHRONON_PATH, GROUP_BY_BATCH_CONCURRENCY
from airflow.models import DAG
from datetime import datetime, timedelta


def batch_constructor(conf, mode, conf_type, team_conf):
    return DAG(
        helpers.dag_names(conf, mode, conf_type),
        **helpers.dag_default_args(),
        default_args=helpers.task_default_args(
            team_conf,
            conf["metaData"]["team"],
            retries=1,
            retry_delay=timedelta(minutes=1),
        ),

    )


def streaming_constructor(conf, mode, conf_type, team_conf):
    return DAG(
        helpers.dag_names(conf, mode, conf_type),
        default_args=helpers.task_default_args(
            team_conf,
            conf["metaData"]["team"],
            retries=1,
            retry_delay=timedelta(seconds=60),
            queue='silver_medium',
        ),
        start_date=datetime.strptime("2022-02-01", "%Y-%m-%d"),
        max_active_runs=1,
        dagrun_timeout=timedelta(minutes=20),
        schedule_interval=timedelta(minutes=20),
        catchup=False,
    )
```

What do these autogenerated DAGs do?

Enable backfills

Kick off streaming jobs

Consistency & data quality checks

....and more

What are some lessons we can learn from Airflow-as-an-engine use cases?

1) Not everyone is going to directly write DAGs... but they'll still use Airflow!

Uber

Introducing uWorc

We built uWorc, or Universal Workflow Orchestrator, on our guiding principles. It includes a simple drag and drop interface that can manage the entire life cycle of a batch or streaming pipeline, without having to write a single line of code.

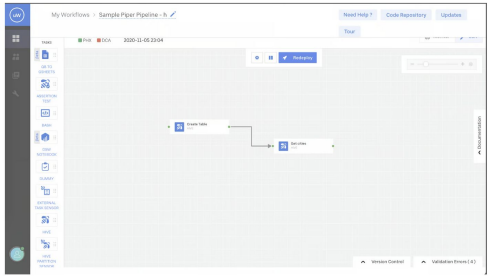


Figure 1: uWorc has a drag and drop workflow editor.

gusty

```
# ---
# dependencies:
#   - same_dag_task
# external_dependencies:
#   - another_dag: another_task
#   - a_whole_dag: all
# python_callable: say_hello
# ---

def say_hello():
  phrase = "hello world"
  print(phrase)
```

Dag Factory

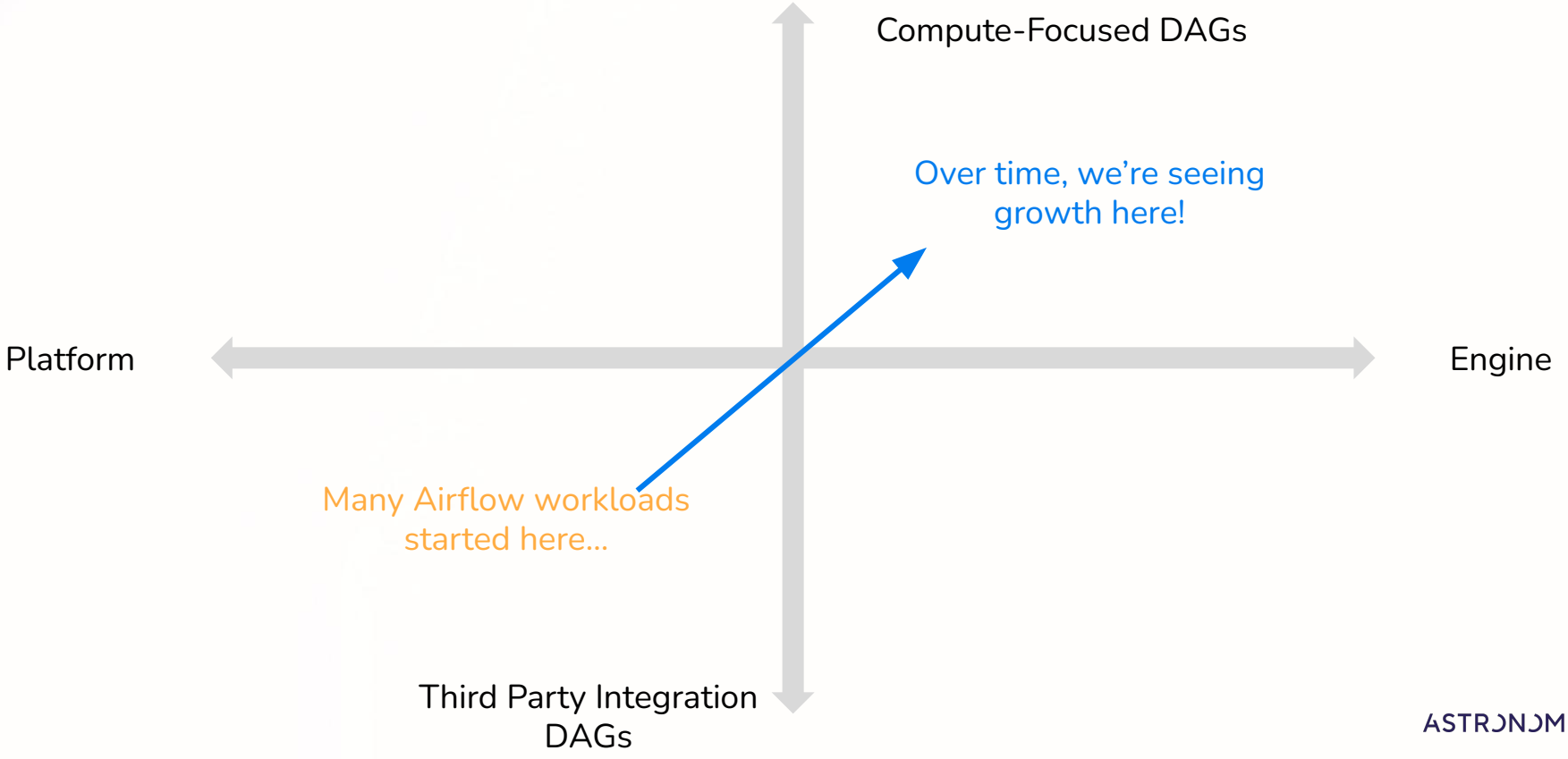
1. To install *dag-factory*, run the following pip command in your Apache Airflow® env

```
pip install dag-factory
```

2. Create a YAML configuration file called `config_file.yml` and save it within your

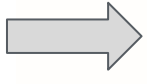
```
example_dag1:
  default_args:
    owner: 'example_owner'
    retries: 1
    start_date: '2024-01-01'
  schedule_interval: '0 3 * * *'
  catchup: False
  description: 'this is an example dag!'
  tasks:
    task_1:
      operator: airflow.operators.bash_operator.BashOperator
      bash_command: 'echo 1'
    task_2:
      operator: airflow.operators.bash_operator.BashOperator
      bash_command: 'echo 2'
      dependencies: [task_1]
    task_3:
      operator: airflow.operators.bash_operator.BashOperator
      bash_command: 'echo 3'
      dependencies: [task_1]
```

2) Airflow is maturing as a compute platform, allowing it to directly manage previously third-party jobs



2) Airflow is maturing as a compute platform, allowing it to directly manage previously third-party jobs

dbt Cloud »
DbtCloudRunJobOperator



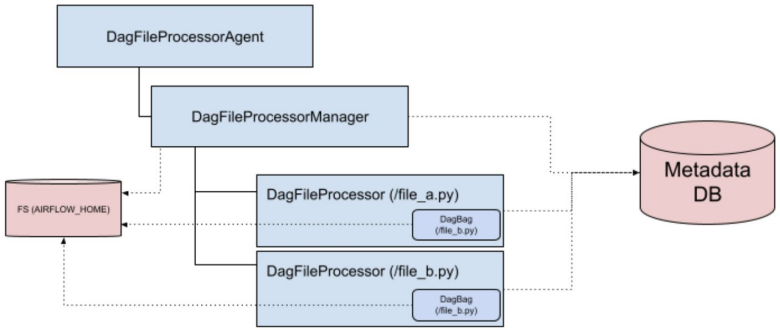
Cosmos

Fivetran »
FivetranOperator

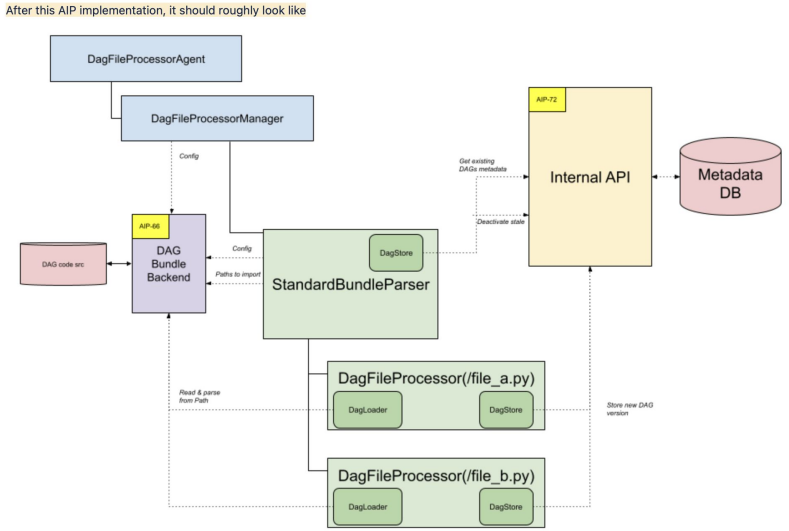
Some third-party job operators (especially for transformation & ingestion workloads) can instead rely on Airflow to provide compute

3) There are high-potential ways of getting new workloads onto Airflow with Airflow 3

Airflow Today



AIP-85: Pluggable bundle parsing



New ways of parsing DAGs can remove the need to write classic Python DAGs for our workflows

Beyond its use as an orchestration platform, Airflow is increasingly an orchestration engine

Thank you for watching!

To reach out, I’m always available at
ian.moritz@astronomer.io

Appendix

Sources

https://dlthub.com/blog/on-orchestrators
https://cwiki.apache.org/confluence/display/AIRFLOW/AIP-85+Extendable+DAG+parsing+controls
https://airbyte.com/blog/data-orchestration-trends