Orchestrating the Future

Navigating Today's Data Workflow Challenges with Airflow and Beyond

Budapest Data + ML Forum
June 2024

Agenda

- Orchestrator – *The What & Why?*
- What is Apache Airflow?
  - Why is Airflow the Industry Standard for Data Professionals?
  - Evolution of Airflow
- Today's Data Workflow Challenges
  - How Airflow addresses them – *Real world case studies*
- The Future of Airflow

Orchestrator

The What & Why?

What is Orchestration? Who is an Orchestrator?

ASTRONOMER

Why Orchestration?

ASTRONOMER

Orchestration in Engineering!

Workflow Orchestrator

Automates and manages interconnected tasks across various systems to streamline complex business processes. E.g Running bash script everyday to update packages on a laptop.

Data Orchestrator

Automates and manages interconnected tasks that deal with **data** across various systems to streamline complex business processes. E.g ETL for a BI dashboard.

What is Apache Airflow?

What is Apache Airflow?

ASTRONOMER

A Workflow Orchestrator, most commonly used for Data Orchestration

Official Definition:

A platform to programmatically author, schedule and monitor workflows

Key Features of Airflow

Python Native

The language of data professionals (Data Engineers & Scientists). DAGs are defined in code: allowing more flexibility & observability of code changes when used with git.

Common Interface

Between Data Engineering, Data Science, ML Engineering and Operations.

Integrates with Toolkit

All data sources, all Python libraries, TensorFlow, SageMaker, MLFlow, Spark, Ray, etc.

Monitoring & Alerting

Built in features for logging, monitoring and alerting to external systems.

Extensible

Standardize custom operators and templates for common DS tasks across the organization.

Pluggable Compute

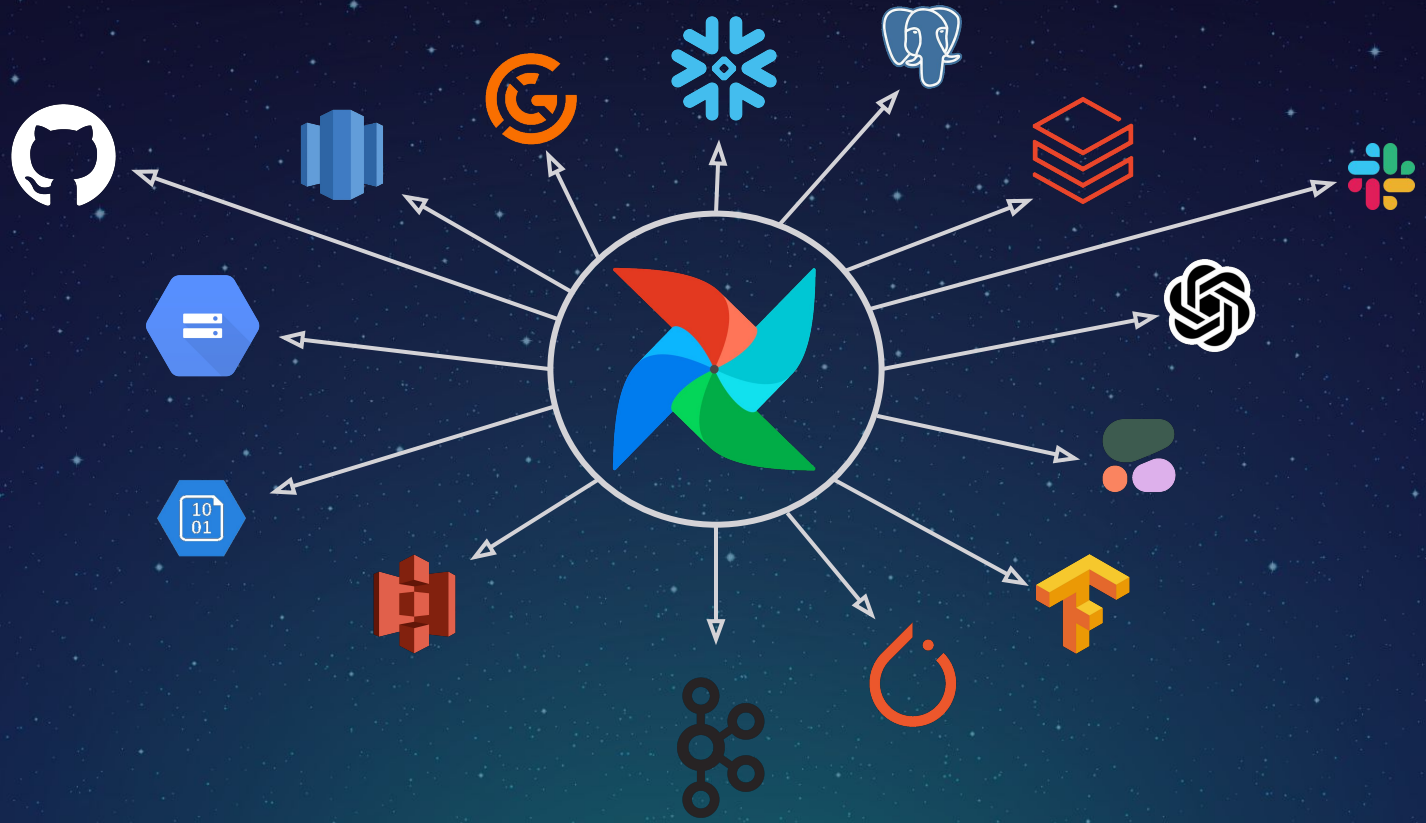
GPUs, Kubernetes, EC2, VMs etc.

Data Agnostic

But data aware.

Cloud Native

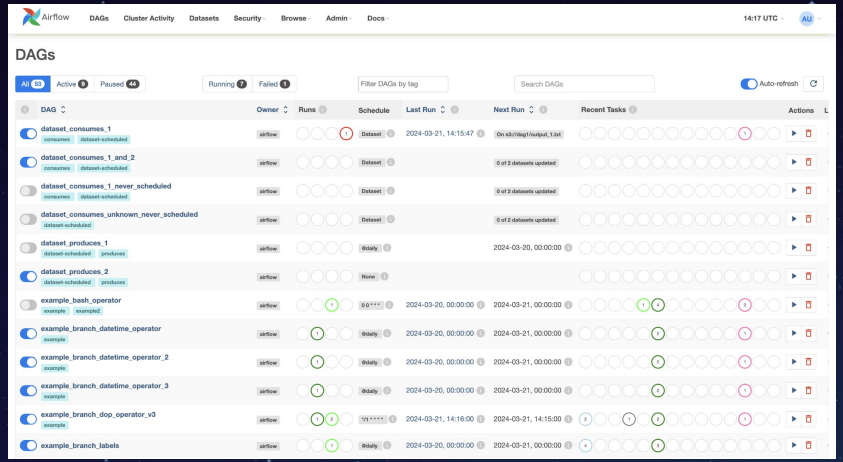
But cloud neutral.

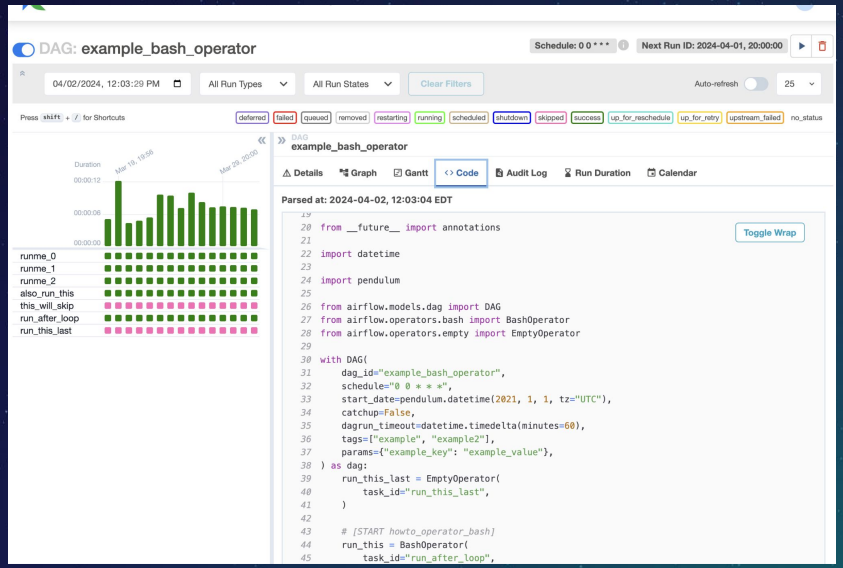


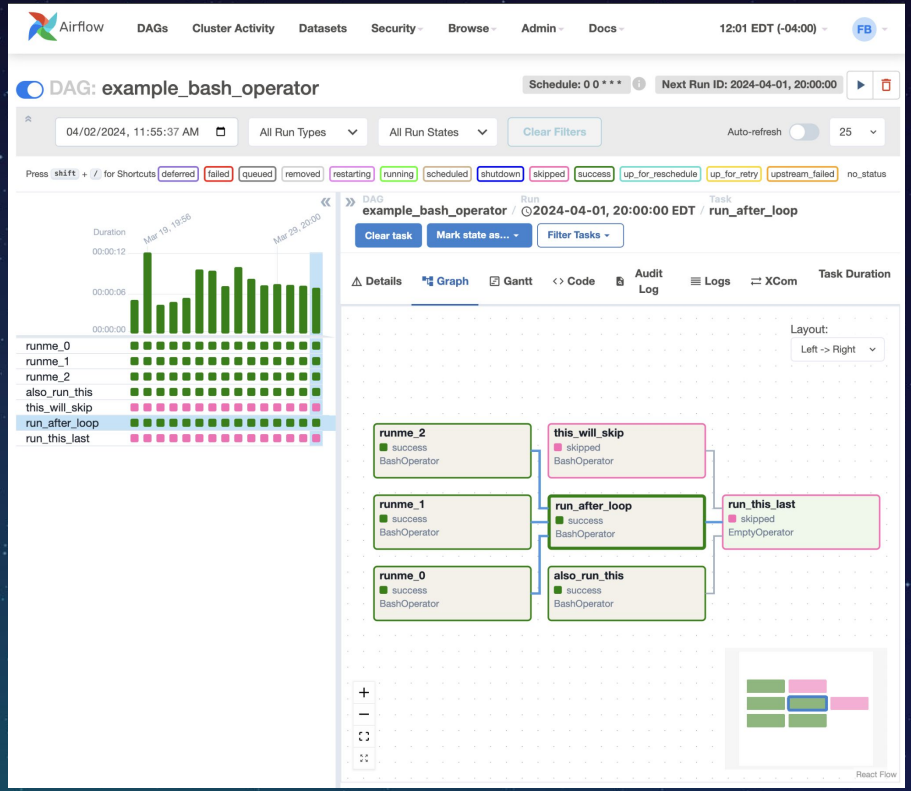
Example DAG

```python
import pendulum

from airflow import DAG
from airflow.decorators import task

with DAG(
    dag_id="etl_dag",
    schedule_interval="0 0 * * *",
    start_date=pendulum.datetime(2022, 6, 1),
) as dag:
    @task
    def extract():
        return [301.27, 433.21, 502.22]

    @task
    def transform(orders: list):
        return sum(orders)

    @task
    def load(total_order_value: float):
        print(f"Total order value is: {total_order_value}")

    order_data = extract()
    order_summary = transform(order_data)
    load(order_summary)
```


```
Total order value is: 1236.7
```


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
14:17 UTC
DAGs
All 55
Active 9
Paused 44
Running 7
Failed 1
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
dataset_consumes_1
dataset_consumes_1_and_2
dataset_consumes_1_never_scheduled
dataset_consumes_unknown_never_scheduled
dataset_produces_1
dataset_produces_2
example_bash_operator
example_branch_datetime_operator
example_branch_datetime_operator_2
example_branch_datetime_operator_3
example_branch_dop_operator_v3
example_branch_labels


DAG: example_bash_operator
Schedule: 0 0 * * *
Next Run ID: 2024-04-01, 20:00:00
04/02/2024, 12:03:29 PM
All Run Types
All Run States
Clear Filters
Auto-refresh
25
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
runme_0
runme_1
runme_2
also_run_this
this_will_skip
run_after_loop
run_this_last
DAG
example_bash_operator
Details
Graph
Gantt
Code
Audit Log
Run Duration
Calendar
Parsed at: 2024-04-02, 12:03:04 EDT
Toggle Wrap
from __future__ import annotations
import datetime
import pendulum
from airflow.models.dag import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.empty import EmptyOperator
with DAG(
dag_id="example_bash_operator",
schedule="0 0 * * *",
start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
catchup=False,
dagrun_timeout=datetime.timedelta(minutes=60),
tags=["example", "example2"],
params={"example_key": "example_value"},
) as dag:
run_this_last = EmptyOperator(
task_id="run_this_last",
)
# [START howto_operator_bash]
run_this = BashOperator(
task_id="run_after_loop",


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
12:01 EDT (-04:00)
FB
DAG: example_bash_operator
Schedule: 0 0 * * *
Next Run ID: 2024-04-01, 20:00:00
04/02/2024, 11:55:37 AM
All Run Types
All Run States
Clear Filters
Auto-refresh
25
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
runme_0
runme_1
runme_2
also_run_this
this_will_skip
run_after_loop
run_this_last
DAG
example_bash_operator
Run
2024-04-01, 20:00:00 EDT
Task
run_after_loop
Clear task
Mark state as...
Filter Tasks
Details
Graph
Gantt
Code
Audit Log
Logs
XCom
Task Duration
Layout:
Left -> Right
runme_2
success
BashOperator
this_will_skip
skipped
BashOperator
runme_1
success
BashOperator
run_after_loop
success
BashOperator
run_this_last
skipped
EmptyOperator
runme_0
success
BashOperator
also_run_this
success
BashOperator
React Flow

Why is Airflow the Industry Standard for Data Professionals?

Apache Airflow

ASTRONOMER

The Community

25M

Monthly Downloads

2.9K

Contributors

35K

GitHub Stars

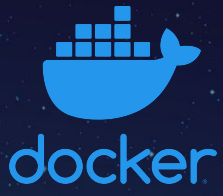
47K

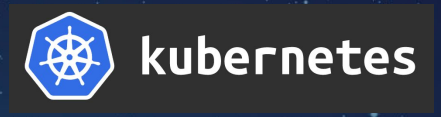
Slack Community

Under ...

THE APACHE® SOFTWARE FOUNDATION

Integrations





And .....

90+ Providers

- Airbyte
- Alibaba
- Amazon
- Apache Beam
- Apache Cassandra
- Apache Drill
- Apache Druid
- Apache Flink
- Apache HDFS
- Apache Hive
- Apache Iceberg
- Apache Impala
- Apache Kafka
- Apache Kylin
- Apache Livy
- Apache Pig
- Apache Pinot
- Apache Spark
- Apprise
- ArangoDB
- Asana
- Atlassian Jira
- Celery
- Cloudant
- CNCF Kubernetes
- Cohere
- Common IO
- Common SQL
- Databricks
- Datadog
- dbt Cloud
- Dingding
- Discord
- Docker
- Elasticsearch
- Exasol
- FAB (Flask-AppBuilder)
- Facebook
- File Transfer Protocol (FTP)
- GitHub
- Google
- gRPC
- Hashicorp
- Hypertext Transfer Protocol (HTTP)
- IBM Cloudant
- Influx DB
- Internet Message Access Protocol (IMAP)
- Java Database Connectivity (JDBC)
- Jenkins
- Microsoft Azure
- Microsoft SQL Server (MSSQL)
- Microsoft PowerShell Remoting Protocol (PSRP)
- Microsoft Windows Remote Management (WinRM)
- MongoDB
- MySQL
- Neo4j
- ODBC
- OpenAI
- OpenFaaS
- OpenLineage
- Open Search
- Opsgenie
- Oracle
- Pagerduty
- Papermill
- PgVector
- Pinecone
- PostgreSQL
- Presto
- Qdrant
- Redis
- Salesforce
- Samba
- Segment
- Sendgrid
- SFTP
- Singularity
- Slack
- SMTP
- Snowflake
- SQLite
- SSH
- Tableau
- Tabular
- Telegram
- Teradata
- Trino
- Vertica
- Weaviate
- Yandex
- Zendesk

Docker Image

Explore / apache/airflow

apache/airflow Sponsored OSS ☆521

Pulls 1B+

By The Apache Software Foundation • Updated 3 days ago

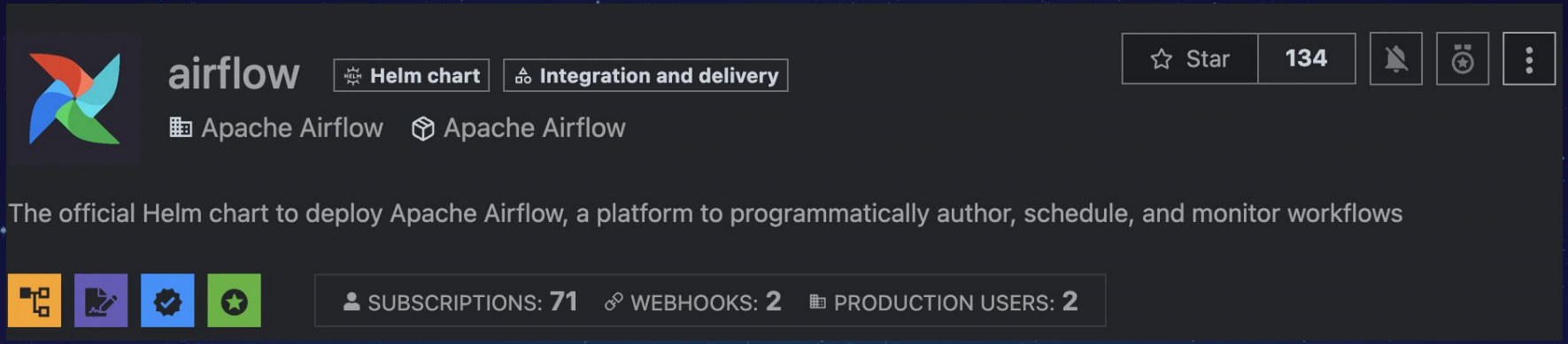
Apache Airflow

IMAGE

DATA SCIENCE | INTEGRATION & DELIVERY | MACHINE LEARNING & AI

```
docker pull apache/airflow
```

Helm Chart

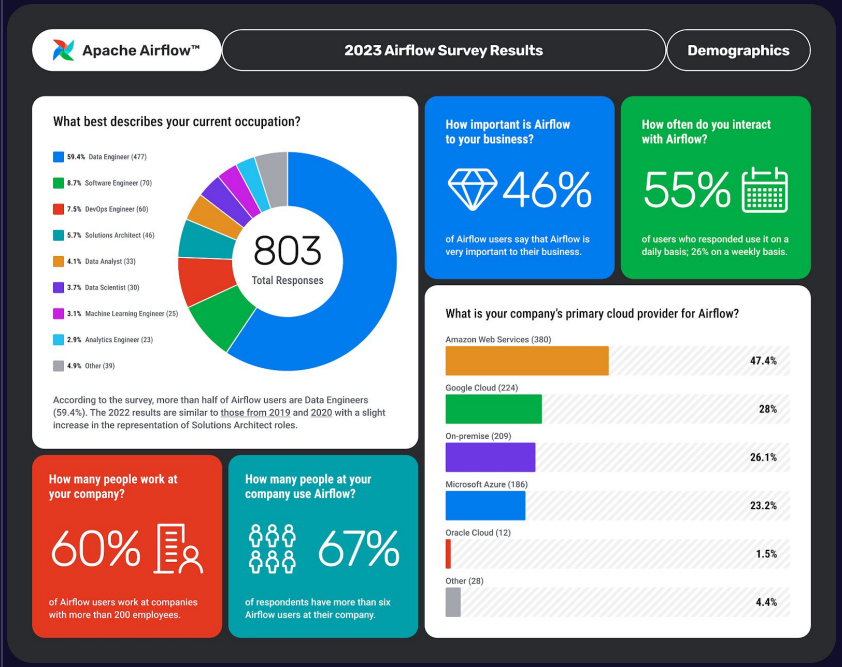



```
helm repo add apache-airflow https://airflow.apache.org/
helm install my-airflow apache-airflow/airflow
```

Managed Airflow Vendors

Airflow Survey and State of Apache Airflow report

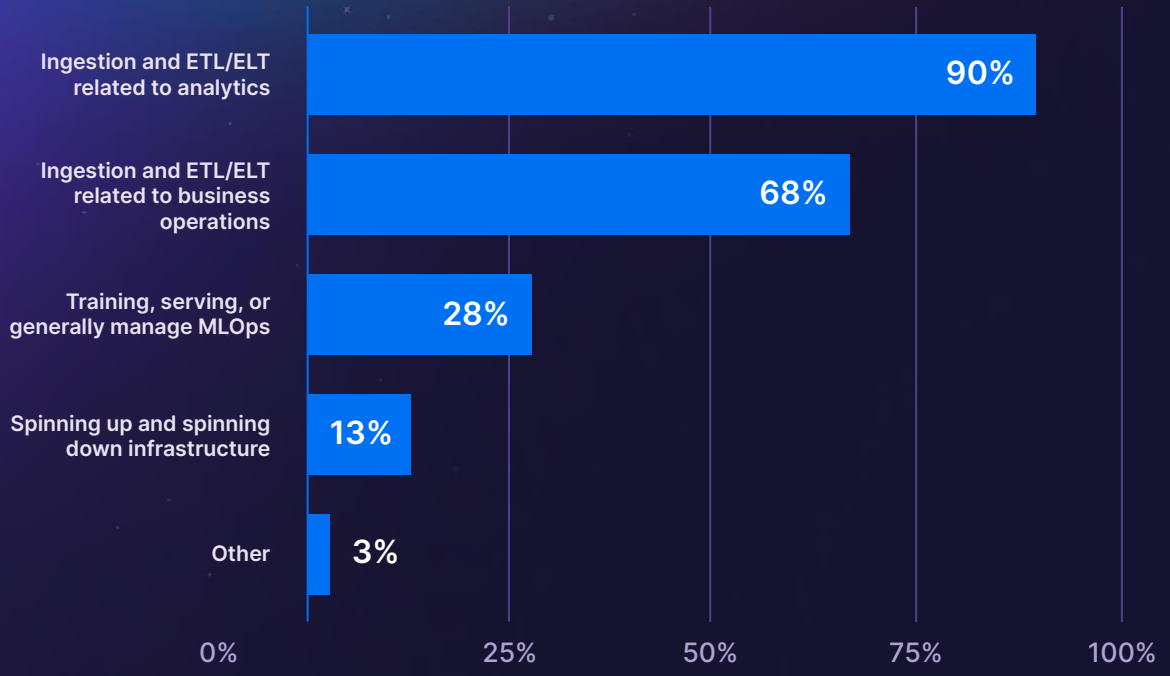
Infographic:
https://airflow.apache.org/survey/



Report:
https://www.astronomer.io/state-of-airflow/

ASTRONOMER

Use cases for Airflow



Source: 2023 Apache Airflow Survey, n=797

90%

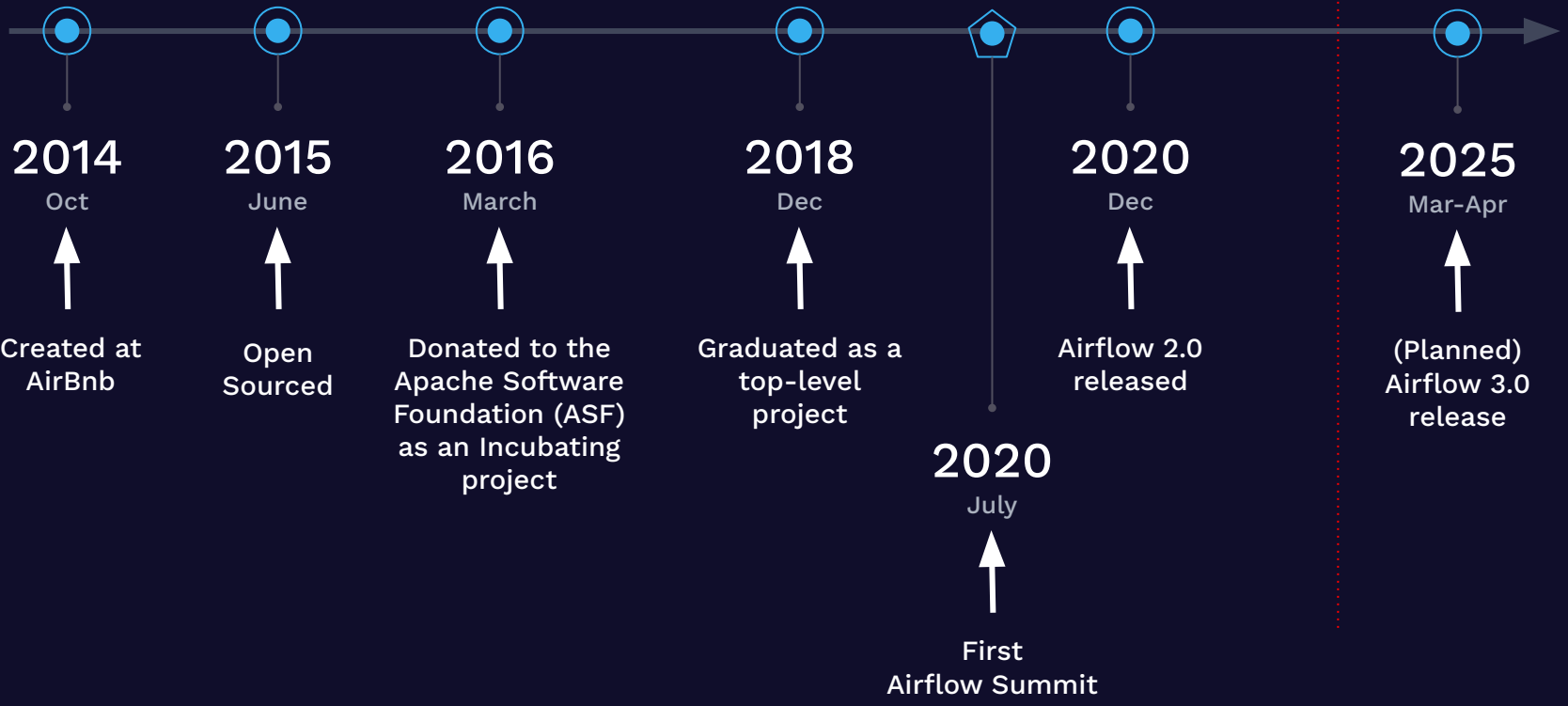
of Apache Airflow usage is dedicated to ingestion and ETL/ELT tasks associated with analytics, followed by **68%** for business operations.

Additionally, there's a growing adoption for MLOps (28%) and infrastructure management (13%), highlighting its versatility across various data workflow tasks.

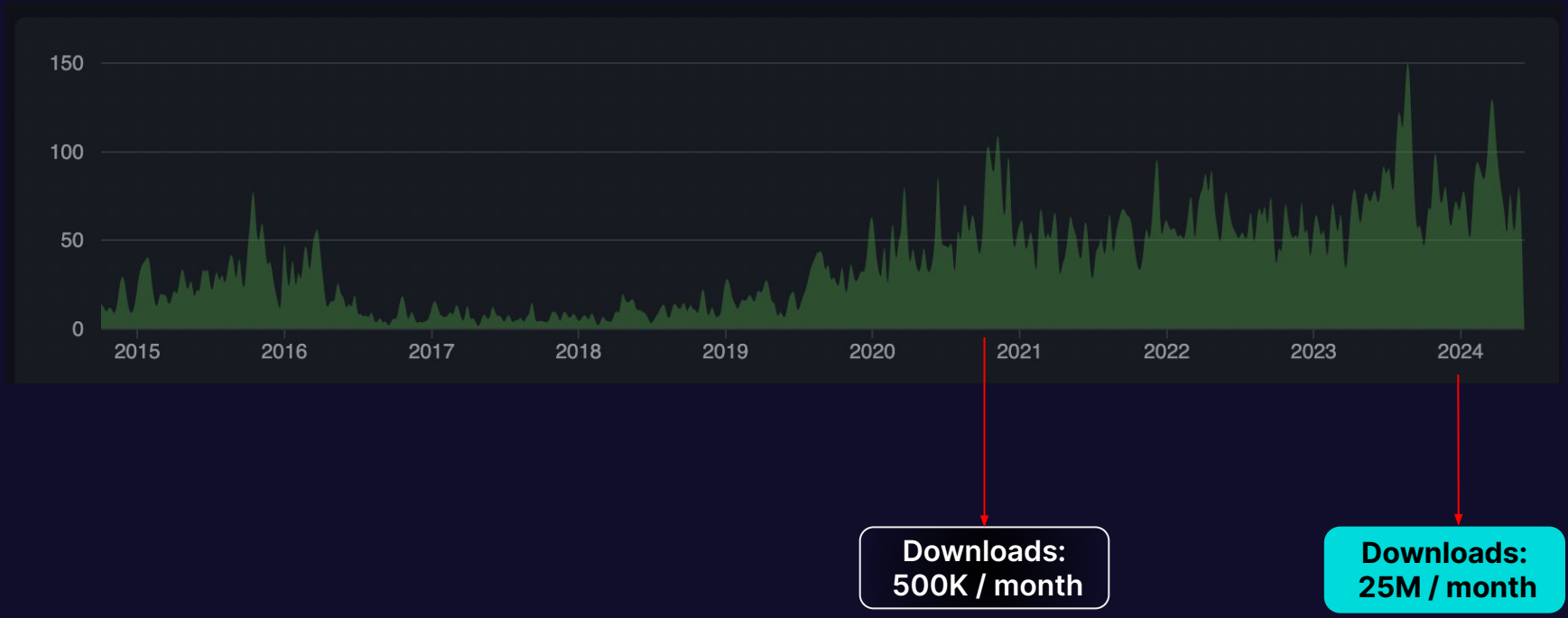
ASTRONOMER

The Evolution of Airflow

Timeline: Major Milestones



Code Contributions & downloads continue to grow!



ASTRONOMER

Today’s Data Workflow Challenges

Today’s Data Workflow challenges

Increasing Data Volumes

Businesses generates more data than ever. Handling this data & its quality is critical.

Intelligent Infrastructure

Infrastructure must be elastic & flexible to optimize for a modern workloads.

Complexity in Data Workflows

Modern workflows need handling data from multiple sources that require managing complex deps & dynamic schedules.

Need for near Real-time Processing

Data Workflows are being used to drive critical business decisions in near real-time & hence requiring reliability & performance guarantees.

Today’s Data Workflow challenges

Additional Interfaces

Net-new teams- from ML to AI - want to get the best out of Airflow without learning a new framework.

Cost Reduction

Tight budgets have pushed teams to efficiently utilize the resources to drive operational costs down.

Platform Governance

Visibility, auditability, & lineage across a data platform is need-to-have.

Licensing & Security in OSS

OSS projects owned by a single company have changed licenses too often in recent past.

How does Airflow address these challenges?

Case Study: Texas Rangers

Company: A professional baseball team in Major League Baseball (MLB), based in Arlington, Texas. The Rangers won their first World Series championship in 2023.

Goal: Use data to gain unfair advantage, Moneyball style! Data to be collected: real-time game data streaming, comprehensive player health reporting, predictive analytics of everything from pitch spin to hit trajectory, and more

Challenge: Scalability issues due to **volume & unprecedented rate of data** & infra bottleneck in their live game analytics pipeline. This impacted the *timely delivery of analytics* to their team and affected their competitive edge.

Case Study: Texas Rangers

Solution: Use Airflow's **worker queues** to create dedicated worker pools for CPU-intensive tasks while other tasks used cheaper workers. Using **Data-aware Scheduling**, they were able to start their DAGs when data was available instead of time-based scheduling.

Result:

Improved Scalability

Using worker queues, DAG completion time reduced by 80% (from 20 mins to 3 mins)

Increased Efficiency

Optimizing compute resources allowed processing of 4 additional DAGs in parallel, enabling immediate post-game analytics delivery for a competitive edge.

Case Study: Bloomberg

Company: Bloomberg is a leading source for financial & economic data: Equities, bonds, Index, Mortgages, currencies, etc. Founded in 1981 with subscribers in 170+ countries.

Goal: Deliver a diverse array of information, news & analytics to facilitate decision-making

Challenge: Maintaining custom pipelines for **diverse datasets** of different domains is expensive & time consuming. Their engineers lacked domain knowledge to aggregate data into client insights & their domain experts lack skills to maintain data pipelines in Production.

Case Study: Bloomberg

Solution: Configuration-driven ETL platform leveraging Airflow & dynamic DAGs. User-defined configs are translated into **Dynamic DAGs** determining tasks & their dependencies with success/failure actions.

Source: https://airflowsummit.org/sessions/2023/airflow-at-bloomberg-leveraging-dynamic-dags-for-data-ingestion/

Result: The Data Platform teams now supports 1600+ DAGs, 700+ datasets, 200+ users, 11 different product teams, 10k+ weekly file ingestions

The Future of Apache Airflow

Airflow 3

Make Airflow the foundation for Data, ML, and Gen AI orchestration for the next 5 years.

1. Enable secure remote task execution across network boundaries.
2. Integrate data awareness needed for governance and compliance
3. Enable non-python tasks, for integration with any language
4. Enable Versioning of Dags and Datasets
5. Single command local install for learning and experimentation.