UNUSUAL VENTURES

TECH SPOTLIGHT

The Rise of Workflow Orchestration Tools

Technology is always evolving. Some advances make us turn our heads, but others rise above and fundamentally change the way we live and work. In this series we highlight an emerging technology that we believe has an unrivaled opportunity for startups.

BY JORDAN SEGALL

Table of Contents

Introduction

At Unusual Ventures, we believe that Data Engineering will play a critical role in enabling companies to leverage data, analytics, and machine learning within their organization.

Data Engineers are the current fastest growing job with 50% YoY growth. Since 2018, there have been four times as many job postings for data engineering roles than that of Data Scientist roles.[1]

While Data Scientists build analyses and ML models on top of big data, Data Engineers support Data Scientists and other organizations by building and maintaining the data infrastructure and pipelines. Data workflow orchestration has become an instrumental part of a Data Engineer's workflow, with Airflow being one of the most commonly required skills among 11% of all job postings.

At the same time, various market trends—such as increased sophistication of data pipelines, event-driven and real-time use cases, and the increasing use of Machine Learning (ML)—have required an influx of new tools designed for modern architectures.

We anticipate significant growth in data engineering tools to support today's infrastructures and use cases. Workflow orchestration tools are a core opportunity in this space and this whitepaper explores how founders can take advantage of them.

Overview

- **The evolution of workflow orchestration tools and what prompted them**
- **New opportunities for emerging players**
- **A map of contemporary solutions in the market**

1 **Source:** Dice 2020 Tech Job Report

Workflow orchestrator origins

A brief history: 2013-2015

Companies have always required workflow management tools even prior to the existence of workflow orchestrators. The standard solution was to write cron jobs to schedule recurring jobs, which worked well back when data pipelines were relatively small.

However, as data and batch processing became increasingly prevalent across organizations, the complexity of data pipelining and requirements surpassed the capabilities of cron jobs. Scripts began to call other scripts, jobs and dependencies became harder to manage. Troubleshooting failed jobs became more difficult and resulted in a convoluted system of interconnected jobs and datasets.

Coinciding with the explosion of data were the releases of popular data warehouse solutions like Amazon Redshift (2012), Snowflake (2012), and BigQuery (2010). As companies began to adopt data warehouses, there became a greater need to routinely perform extract, transform, and load (ETL) processes to transform and load data into the warehouse.

As a result, in order to collect, process, and store data, developers needed to use data workflow systems to author and organize sets of tasks and dependencies—called workflows—to routinely execute in a distributed manner. Many companies built and open-sourced their own workflow systems to satisfy their individual use cases. For example, Spotify created Luigi, LinkedIn created Azkaban, Pinterest made Pinball, and Airbnb made Airflow—all between 2013-2015.

The birth of DAGs

Each of these orchestration tools are fundamentally based on Directed Acyclic Graphs (DAG). A DAG is a collection of tasks set up to run in a specified order that reflects their dependencies. At the time, a required property of DAGs for many solutions (including Airflow) was to run its jobs at a specific scheduled time (we'll return to this later in the piece).

For the first time, DAGs allowed organizations to track their workflows and identify where in the graph a process failed.

The most popular tool to emerge during this period was Airflow, created at Airbnb and inducted into the Apache ecosystem. Airflow remains to this day the de facto data pipeline orchestration tool. It is used by numerous Fortune 500 companies.

Sizable community of Airflow users

16,000 GitHub Stars

7,000 Slack members

1,100 contributors

Notably, while 97% of its users utilize it as part of ETL pipelines, an increasing number of users (30% today) are also using it as part of ML pipelines or automating DevOps. However, Airflow was never built to be used in such a manner. It was designed to run static workflows on a fixed schedule.

As a result, additional tools were required and people began hacking together solutions to fit their needs.

2 **Source:** https://airflow.apache.org/blog/airflow-survey/

A change in data needs

> "My 5-person Data Engineering team at DigitalOcean spent over 80% of our time debugging Airflow, preventing the release of products from teams that relied on the Data Engineering team's support." – *Former Data Scientist at DigitalOcean*

In explaining the emergence of modern workflow orchestration tools, let us first examine various market trends and why Airflow (as an example representing old orchestrators) was unable to adapt to changing needs of modern, data-centric organizations.

Event-driven architectures

Airflow is strictly dependent on time, requiring a DAG execution date. As companies have migrated to event-driven architectures, such as serverless, numerous workflows have changed from a previously batch/scheduled model, to kicking off upon activation. Airflow cannot support this.

This is further compounded by the rise of real-time use cases and stream processing outside of batch processing. DAGs on Airflow cannot run off schedule or be parameterized with input. They have limited branching logic.

Enhanced machine learning

ML pipelines have different requirements than the ETL pipelines that Airflow was primarily designed for. Data scientists still use Airflow to orchestrate their pipelines, but implement significant workarounds to do so.

For example, sharing state is prohibitively difficult in Airflow where each step in the workflow typically ends with data residing in a datastore. Airflow has a feature called XComs that allows tasks to exchange small pieces of metadata with each other by using admin access to write executables into an Airflow metadata database.

This functionality is insecure and highly inefficient, and often creates dependencies between tasks that the Scheduler is unaware of, breaking the DAG. For example, if a Data Scientist creates a 10GB data frame and passes it through 10 tasks of a DAG, then 100GB of permanent data is written to Airflow's database with each run. In short, Airflow cannot sufficiently pass data from one task to another in a manner that Data Scientists need.

Enhanced security/compliance

Facing increasing pressure to comply with GDPR, CCPA, and other data security legislation, organizations require commercial solutions for their workflow orchestration systems.

Most organizations lack the resources to manually secure open source orchestration solutions like Airflow and instead opt for solutions where security is a first-class feature or for managed services.

Improved latency in complex data pipelines

Airflow's highly optimized API and architecture have failed to keep up with the overall sophistication of data pipelines. It was architected to couple its scheduler with its workflow functionality. That means the scheduler checks all Task dependencies to determine which tasks to run, reparses the DAG folder every few seconds, sets the final DAG states in the DB, and checks the DAG schedules, etc.

This introduces several flaws, such as latency between determining when a task meets its dependencies and when it executes. It requires a scheduling service to run locally and a user to instantiate a database, while not enabling tasks to communicate with one another as described previously.

As a result, Airflow's scheduler's latency takes 10 seconds to run any task (5 seconds to mark as queued and 5 seconds to submit for execution), regardless of the size of the Dask/Spark cluster. If there are a small number of tasks, then this latency would be fine, but the number of tasks are only growing:

Growing pipelines demand sophisticated solutions

62% of companies have at least 2 DAG with >20 tasks

27% of companies have 100-10,000 tasks in any given DAG

92% companies increase microservices each year[3]

As a result of increasingly complex DAGs and an increasing number of database systems and associated dependencies, old workflow systems designed with a particular use case in mind—regularly timed tasks for batch processing—often need significant resources for maintenance or custom code to conform to today's modern architectures and needs.

Today, 40% of Airflow's users are developers, architects, and Data Scientists developing workarounds to execute a ML pipeline upon a trigger event and pass data between thousands of tasks in a low-latency manner.

3 **Source:** Lightstep Microservice Trend Report

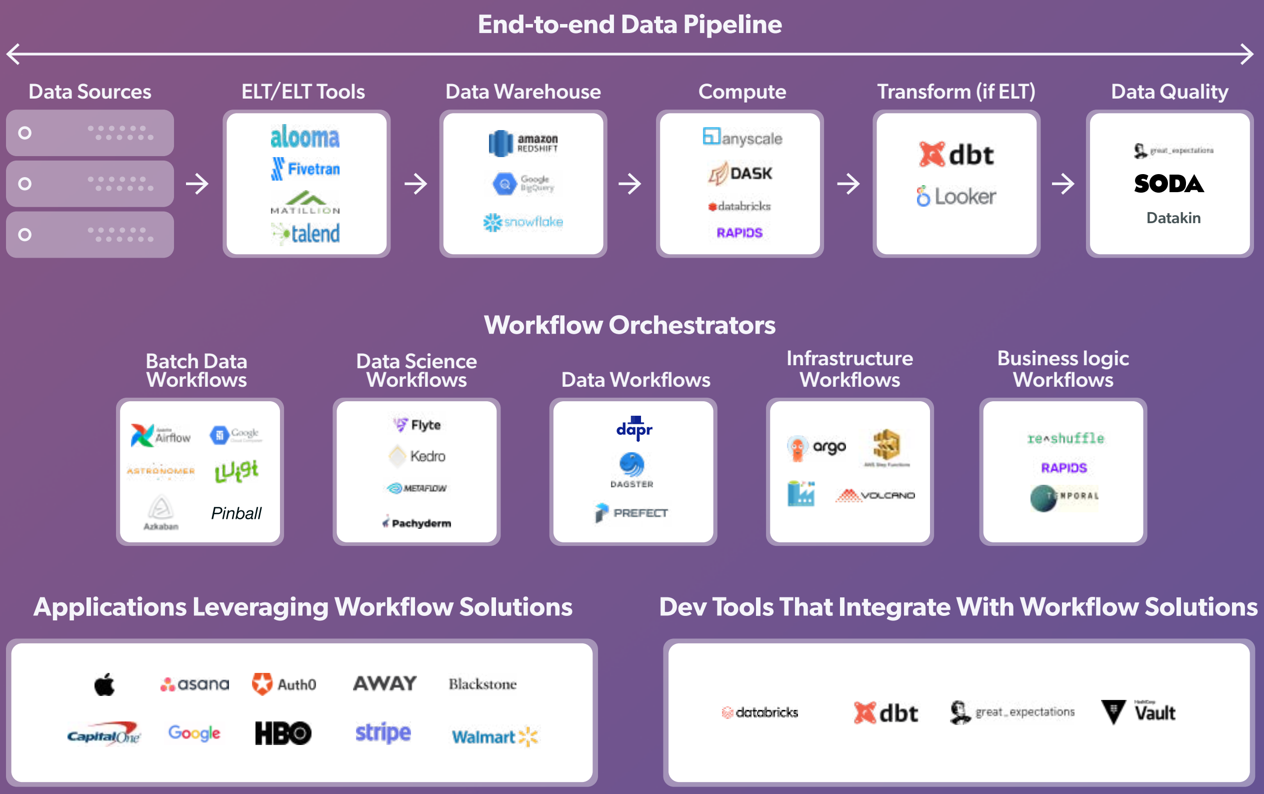
Current market landscape

The evolution of data engineering ETL jobs and the inefficiencies of older solutions has given rise to various workflow orchestration products. These tools orchestrate a particular kind of workflow or offer a more modern, generalized orchestration that improves upon its predecessors.

The market map below consists of startups and open source packages across various types of orchestration tools. The top layer highlights how orchestration tools can play a role in creating and monitoring data pipelines across each step of the process. The middle layer demonstrates the various types of workflow solutions that tackle different use cases. Finally, the bottom layer illustrates the prevalence of orchestration solutions across varied use cases, as well as how orchestration tools can integrate with other developer tools.

To better illustrate the current state of the industry, we have featured one technology from each category to describe in further detail on the next page to provide a better picture of the types of technologies emerging in the workflow orchestration space.

Workflow Orchestrators



Batch data

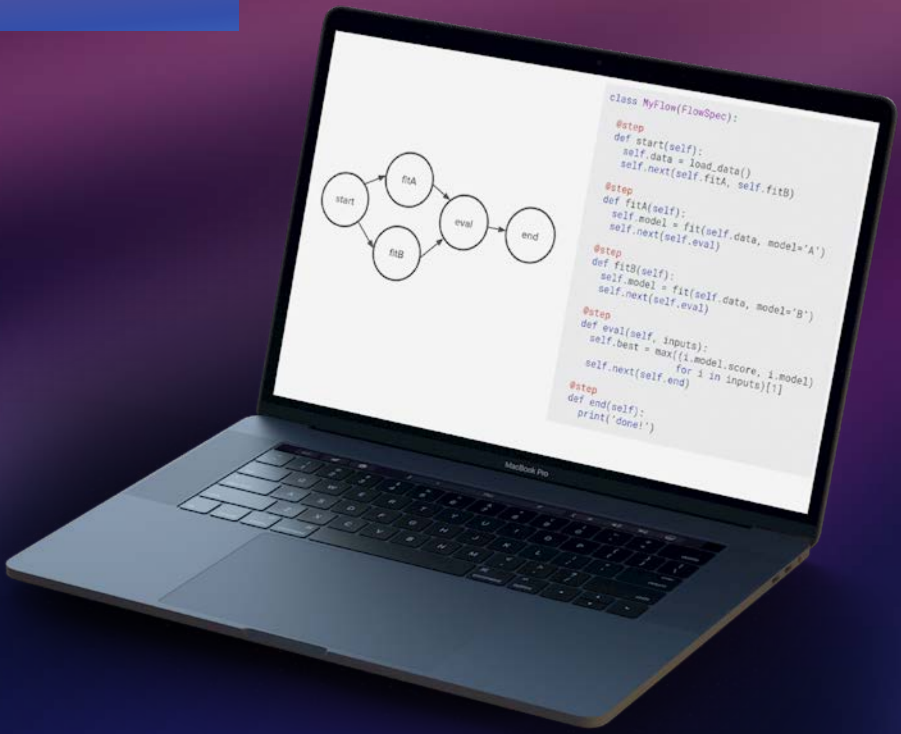
Apache Airflow

We've already covered Apache Airflow in detail. However, while many of its flaws were discussed, it is still the most widely used solution and sufficient for many companies.

For example, Airbnb uses Airflow to compute engagement metrics, search ranking results, and A/B testing framework logic. It also uses it for ETL'ing data into its data warehouse, performing email targeting, and cleaning up data infrastructure.

Robinhood evaluated numerous competitors like Pinball (sparse documentation), Azkaban (uses properties files to define workflows, which is much more difficult than workflows as code), and Luigi (lacks a scheduler and relies on cron) before choosing to use Airflow. It was selected for its batch processing jobs at set schedules, including data analysis, metric aggregations, and computations like dividend payouts.

If your needs are limited in nature to standard scheduled batch jobs, then Airflow is a solid choice due to the massive community and documentation available from a support perspective.



```
class MyFlow(FlowSpec):

  @step
  def start(self):
    self.data = load_data()
    self.next(self.fitA, self.fitB)

  @step
  def fitA(self):
    self.model = fit(self.data, model='A')
    self.next(self.eval)

  @step
  def fitB(self):
    self.model = fit(self.data, model='B')
    self.next(self.eval)

  @step
  def eval(self, inputs):
    self.best = max((i.model.score, i.model)
                    for i in inputs)[1]
    self.next(self.end)

  @step
  def end(self):
    print('done!')
```

Data science

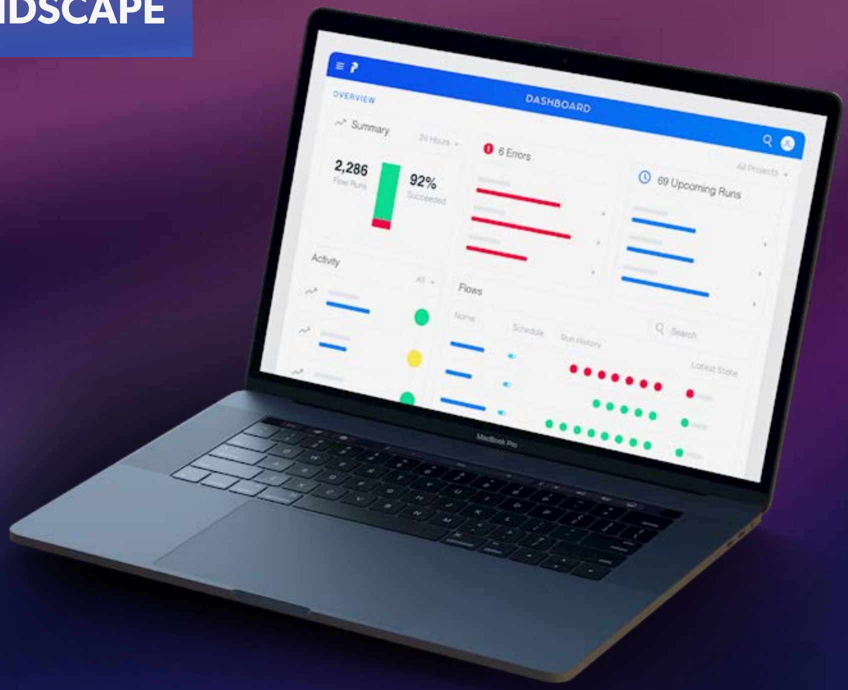
Metaflow

With its increasing use of data science, Netflix created Metaflow to manage the challenge of Data Scientists. It was built to ensure production workloads were packaged for containers to execute on servers, return model results as containers for consumption by business applications, and store and version their data.

Metaflow allows Data Scientists to structure their ML workflows as a DAG consisting of familiar DS-specific steps (e.g., “fit”, “eval”). It allows for the passing of state and variables between steps, with models and data acting as Python instance variables, and provides first class support for running locally or distributed in the cloud.

It also manages the serialization, de-serialization, and persistence of objects between DAGs and stores results after each node executes, allowing users to restart flows from the point of failure, reproduce past results, and inspect each node within a workflow.

Metaflow intends to allow users who write DAGs using their FlowSpec to transition their Metaflow DAGs to production schedulers like Airflow. Metaflow is an example of the “verticalization” of workflow solutions like Airflow—which were originally designed for batch processing—to more role-specific use cases like machine learning for Data Scientists.



Data workflows

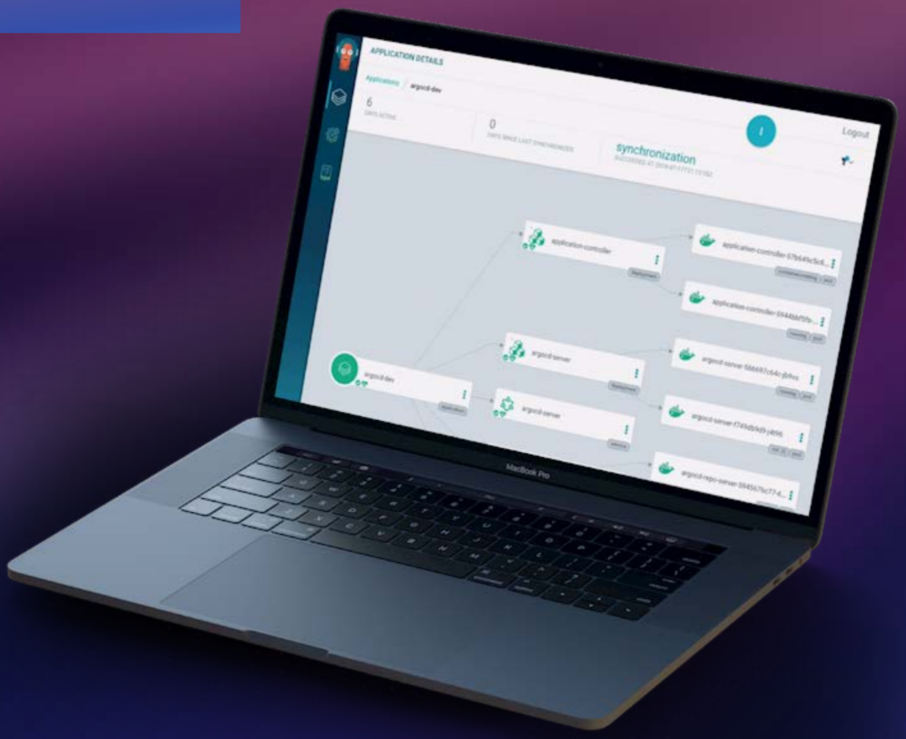
Prefect

Prefect was designed from the ground-up to solve many of Airflow's inefficiencies and to support any type of workflow beyond batch-based ETL. It treats workflows as standalone objects that can be run at any time for any reason, allowing for event-driven architectures and non-schedule based flows with input parameters. It treats every Prefect task just like a function, where they can be parameterized and call one another, and every function can be converted to a Prefect task with one line of code to easily convert scripts into Prefect DAGs.

Unlike Airflow, Prefect's scheduler is decoupled from its workflow logic and execution, freeing it from its ancestor's latency issues. As a result, the flow itself can act dynamically and outsource execution details to systems like Spark/Dask.

There are numerous other improvements to Airflow, with first class features, including versioning and local execution without the need for a centralized executor, and an enhanced UI for debugging failures in execution.

Finally, Prefect's unique hybrid execution model allows companies to leverage Prefect's technology on-premise, gaining the security benefits of such an offering, while using Prefect's managed cloud service. Data remains within the company's network, while only metadata required for the cloud orchestrator to function and manage state is passed between the on-prem and cloud environments. This model is particularly well suited for organizations with strict compliance standards, such as banks and healthcare companies.



Infrastructure workflows

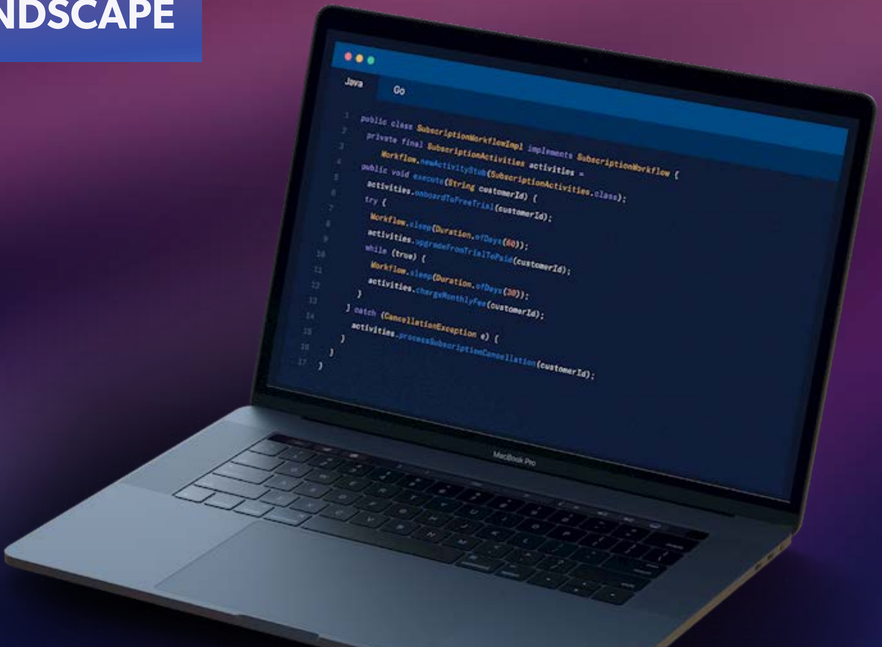
Argo

Argo is a container-native workflow engine for Kubernetes used by companies, such as Adobe, Capital One, BlackRock, and Datadog. Argo was designed for container-based workloads to run on any Kubernetes cluster. Unlike standard DAGs, each step in an Argo DAG is implemented as a container.

Similar to Kubernetes, Argo workflows are also designed with YAML as Kubernetes CRDs (Custom Resource Definitions). Argo includes a native artifact manager for state management between steps and portability of outputs, with versionable workflows.

While Argo can be used for running and orchestrating Spark jobs on Kubernetes for machine learning or data processing, it also includes many other extensions. Argo CD allows developers to build and run CI/CD pipelines natively on Kubernetes and includes a UI for declaring version-controlled application definitions, configurations, and environments.

Argo Rollouts provides additional deployment capabilities including blue-green, canary, experimentation, and progressive delivery to Kubernetes. Argo Events is an event-based dependency manager for Kubernetes that allows developers to define event-based triggers and dependencies from numerous event sources prior to executing workflows or business logic.



```
public class SubscriptionWorkflowImpl implements SubscriptionWorkflow {
  private final SubscriptionActivities activities =
      Workflow.newActivityStub(SubscriptionActivities.class);
  public void execute(String customerId) {
    activities.onboardToFreeTrial(customerId);
    try {
      Workflow.sleep(Duration.ofDays(60));
      activities.upgradeFromTrialToPaid(customerId);
      while (true) {
        Workflow.sleep(Duration.ofDays(30));
        activities.chargeMonthlyFee(customerId);
      }
    } catch (CancellationException e) {
      activities.processSubscriptionCancellation(customerId);
    }
  }
}
```

Business-logic workflow

Cadence/Temporal.io

Temporal.io is the commercial company behind Cadence, an open source solution developed at Uber. It is a tool for orchestrating asynchronous long-running business logic in a fault-tolerant and stateful manner.

In a traditional distributed architecture with numerous microservices, business logic workflows can extend beyond a simple request-reply logic, requiring tracking of complex state across stateless services and responding to asynchronous events and external dependencies with management of queuing systems. This complexity naturally leads to low availability as certain components with the system crash.

Cadence is a fault-oblivious stateful programming model, such that the states of the workflows are immune to service failures and the complexity of the distributed system is abstracted away to allow developers to focus on business logic.

As a result, while Cadence can be used for workflows like those that Airflow supports, it is also a more generalizable workflow solution for dealing with asynchronous business logic. With Cadence/Temporal, a user simply implements the business logic of the workflow, and the workflow is oblivious to failures and downtime of any worker processes in the flow.

Cadence also provides visibility into the execution state of such workflows, allowing for debugging and visibility into downstream processes with scalability to hundreds of millions of concurrent executions. In some use cases, Temporal provides financial transaction reliability, infrastructure automation and provisioning, ML model life-cycle management, reactive loyalty programs, and customer support.

Closing thoughts

Modern infrastructure and application needs have changed drastically since the introduction of Airflow in 2014. Standard batch processing jobs, though still prominent, have evolved into sophisticated distributed applications of thousands of microservices that have event driven architectures, real time use cases and stream processing, and machine learning-based data intensive workloads.

While Airflow remains a powerful tool and sufficient for standard batch jobs, it can no longer support the myriad needs of modern architectures.

Airflow and similar DAGs of the 2013-2015 era paved the way for tools designed to solve specific use cases in the areas of enhanced ETL, data science, infrastructure, DevOps, and distributed application business logic. At Unusual Ventures, we're excited about tools that better enable developers to orchestrate data in an increasingly complex ecosystem, and look forward to seeing the next generation of workflow orchestration tools emerge.

Contact

If you think workflow orchestration is interesting, are working on unusual tech (orchestrators and more), or simply want to chat, we want to hear from you. You can follow us on **Twitter**, Connect on **LinkedIn**, or simply shoot us an email.

UNUSUAL VENTURES

Jordan Segall
@jordan_segall
jordan@unusual.vc