The Ultimate Guide to Getting Started with Apache Airflow

Powered by Astronomer

Editor’s Note

Welcome to The Ultimate Guide to Getting Started with Apache Airflow, brought to you by Astronomer. We believe that Airflow is a best-in-class open source technology for adapting to the ever-changing world of data. In this ebook, we’ve gathered and explained key Airflow concepts to help you get started using the platform. We’ve also compiled some guides and tutorials for more advanced Airflow features, as well as real-life use cases. The last section is Astronomer-specific: we provide information about getting Apache Airflow-certified and answer some frequently asked questions about adopting Airflow.

This publication is complemented by the videos, webinars, and guides available at www.astronomer.io.

Enjoy!

Table of content

What is Apache Airflow?

Apache Airflow is a way to **programmatically** author, schedule and monitor your data pipelines.

Apache Airflow was created by Maxime Beauchemin while working at Airbnb as an open-source project in late 2014. It was brought into the Apache Software Foundation's Incubator Program in March 2016 and saw growing success afterward. By January of 2019, Airflow was announced as a Top-Level Apache Project by the Foundation and is now considered the industry's leading workflow orchestration solution.

> *Leveraged by 1M+ data engineers and deployed by 1000s of companies as the unbiased data control plane, it connects business with data processing fabric.*

Apache Airflow's popularity as a tool for data pipeline automation has grown for a few reasons:

1. **Proven core functionality for data pipelining**
   Airflow's core capabilities are used by thousands of organizations in production, delivering value across scheduling, scalable task execution, and UI-based task management and monitoring.

2. **An extensible framework.**
   Airflow was designed to make integrating existing data sources as simple as possible. Today it supports over 60 providers, including AWS, GCP, Microsoft Azure, Salesforce, Slack, and Snowflake. Its ability to meet the needs of simple and complex use cases alike makes it both easy to adopt and scale.

3. **It's scalable.**
   From a few pipelines to thousands of them running every day.

4. **A large, vibrant community.**
   Airflow boasts thousands of users, and over 1,600 contributors who regularly submit features, plugins, content and bug fixes to ensure continuous momentum, and improvement. **In 2020, Airflow reached 10,000 commits, 18,000 GitHub stars, and over 13,000 members of the Slack community.**

Apache Airflow continues to grow, thanks to an active and expanding community and very deep, proven functionality. And, since it's under ASF and governed by a group of PMC members—it can live forever.

As a result, hundreds of companies — including Fortune 500s, tech giants, and early-stage startups — are adopting Airflow as their preferred tool to programmatically author, schedule, and monitor workflows. Among the biggest names, you'll find: Adobe, Bloomberg, Asana, Dropbox, Glassdoor, HBO, PayPal, Tesla, ThoughtWorks, WeTransfer, and more!

Apache Airflow Core Principles

Airflow is built on a set of core ideals that allow you to leverage the most popular open source workflow orchestrator on the market while maintaining enterprise-ready flexibility and reliability.

Flexible
Fully programmatic workflow authoring allows you you to maintain full control of the logic you wish to execute.

Extensible
Leverage a robust ecosystem of open source integrations to connect natively to any third party datastore or API.

Open Source
Get the optionality of an open-source codebase while tapping into a buzzing and action-packed community.

Scalable
Scale your Airflow environment to infinity with a modular and highly-available architecture across a variety of execution frameworks.

Secure
Integrate with your internal authentication systems and secrets managers for an platform ops experience that your security team will love.

Modular
Plug into your internal logging and monitoring systems to keep all of the metrics you care about in one place.

Why Open Source Software?

As reported by Accel,

> *in 2019 alone, over $1.3 million first-time contributors joined the open source community and 30% of all projects on GitHub were created.*

The massive increase in open source software (OSS) projects is changing both the tech scene and the business scene on a global scale. So why do we love open-source? Because it is:

Innovative
According to Wired, when Facebook decided to stop "*treating data center design like Fight Club*," companies like Microsoft, HP, and even Google followed suit. Finally, companies no longer had to waste time continually adopting new technologies to fix the shortcomings of their existing infrastructure. Now they can implement the components they need—created by top developers—and innovate elsewhere.

Reliable
We've all had "*technical difficulties*" when our technology operated unreliably. Professionally, that can cause a variety of outcomes, from mild annoyance to genuine detriment. When everybody's using the same components, everybody's optimizing the same components. As Linux Creator Linus Torvalds said, "Given enough eyeballs, all bugs are shallow" (Linus's Law). In other words, the more people who can see and test a set of code, the more likely any flaws will be caught and fixed quickly.

Diverse

It's a huge benefit to have a large number of developers look at and contribute to the same systems. Instead of a few people with the same view of the world-building components, an entire community with varying perspectives and strengths contributes to the system (and has a stake in its success and adoption), which makes the system more resilient.

Transparent

Transparent meaning people aren't tied down to a proprietary system protected in a black box. Transparent, as in the ones actually using and building on the system can see everything; they can pop open GitHub, look at the code, and trust that the code has been reviewed publicly. This gives decision-makers insight into what's going on—where components can be swapped around—and the control they need to do it.

Agile

When components are living in a community-curated open source world, they are built to "*play nicely*" with other components. This flexibility allows us (and anyone) to stay on the cutting edge. And with open-source, any organization can take a piece of code and customize it to best fit their needs.

When it comes to tech, we want to live in a world where we're free to explore, invent best-in-class technology and use it in revolutionary ways. We can do it through open source.

Apache Airflow 101

Why use data pipelines as code?

INTERVIEW

Kenten Danas

Astronomer Data Engineer

Getting started with Airflow requires understanding the concept of pipelines as code. We asked Astronomer Data Engineer Kenten Danas about the qualities and benefits of this development style.

Q: Why would you run your pipelines as code?

At Astronomer, we believe using a code-based data pipeline tool like Airflow should be a standard. There are many benefits to that solution, but a few come to mind as high-level concepts:

Firstly, code-based pipelines are extremely **dynamic.** If you can write it in code, then you can do it in your data pipeline. And that's really powerful.

Secondly, code-based pipelines are **highly extensible.** You can integrate with basically every system out there, as long as it has an API.

Finally, they are **more manageable** since everything is in code, it can integrate seamlessly into your source controls CICT and general developer workflows. So no need to manage some external things differently.

Q: What about a company that is looking at a drag-and-drop tool or other low/no-code approach? Why should they consider Airflow?

The first thing I'd say is that we never think of it as *"Airflow or"*– instead, it's always *"Airflow and"*. Because Airflow is so extensible, you can still use those drag-and-drop tools and then orchestrate and integrate them with other pipelines using Airflow. **This approach allows you to have a one-stop-shop for orchestration without necessarily having to make a giant migration effort upfront or convince everybody at one time.**

> *As those internal teams get more comfortable with Airflow and see all the benefits, they'll naturally start to transfer to Airflow.*

"Apache Airflow has become an essential part of the modern data stack. At Astronomer we live and breathe Airflow, which is why we've decided to share that knowledge with data professionals looking to get started with the tool. We truly hope you'll enjoy the content of our ebook and find it useful in your data engineering journey."

Kenten Danas
Field Engineer at Astronomer

Apache Airflow Core Concepts

✦ DAGs are the foundational structure of Airflow

A directed acyclic graph, or DAG, is a structure with a few mathematic qualities:

1. **Directed**
   If multiple dependent nodes exist, each node must have at least one defined upstream (previous) or downstream (subsequent) task.

2. **Acyclic**
   No node can be dependent on itself. This could cause an infinite loop that would be, um, it'd be bad. Don't do that.

3. **Graph**
   All nodes can be laid out in a diagram with clear relationships to each other.

Dependencies between nodes are represented by edges on the graph. As long as the graph remains directed and acyclic, there are no limits to how nodes and edges are defined, making DAGs suitable for nearly every use case.

✦ DAGs in Airflow: A powerful structure

In Airflow, a DAG is your data pipeline, and represents a set of instructions that must be completed in a specific order. This is beneficial to data orchestration for a few reasons:

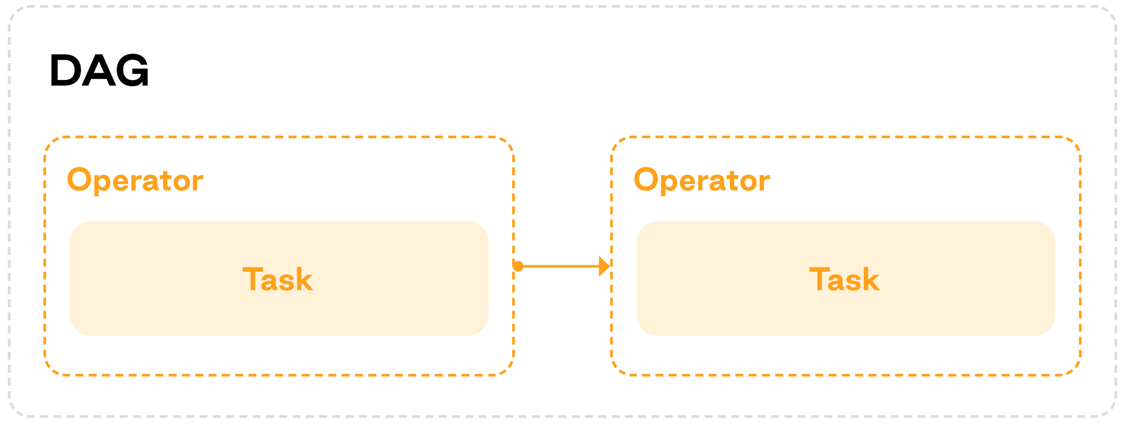
- DAG dependencies ensure that your data tasks are executed in the same order every time, making them reliable for your everyday data infrastructure.
- The graphing component of DAGs allows you to visualize dependencies in Airflow's user interface.
- Because every path in a DAG is linear, it's easy to develop and test your data pipelines against expected outcomes.

Perhaps what's most important about DAGs in Airflow is that they are 100% Python code. With just a basic understanding of Python, you can easily write and deploy data pipelines.

An Airflow DAG starts with a **task** written in Python. You can think of tasks as the nodes of your DAG: Each one represents a single action, and it can be dependent on both upstream and downstream tasks.

Tasks are wrapped in Operators, which are the building blocks of Airflow that define the behavior of a task. For example, a task wrapped in a PythonOperator will execute a Python function, while a task wrapped in a SensorOperator will wait for a signal before completing an action.

✦ Workflow

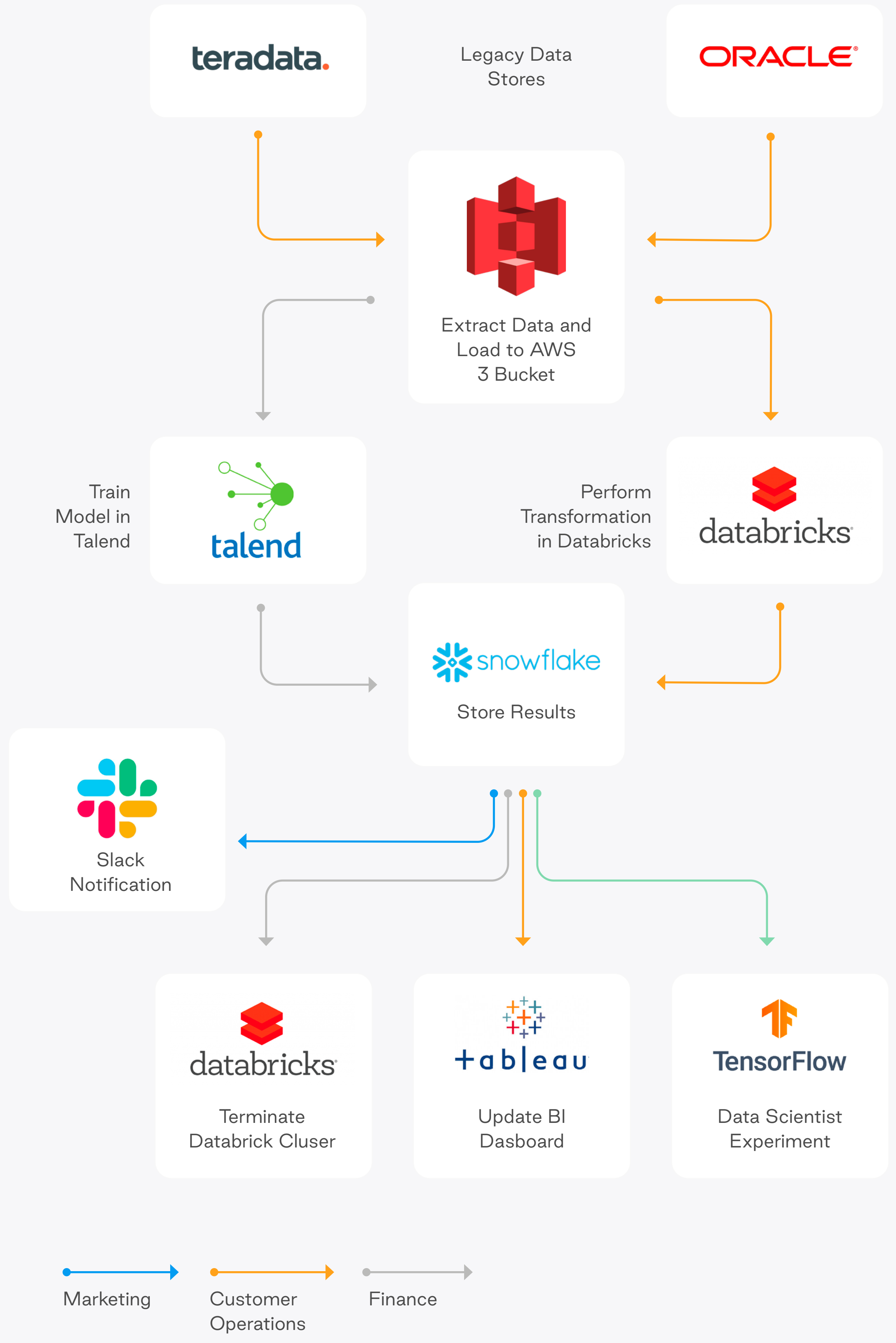


The following diagram shows how these concepts work in practice. As you can see, by writing a single DAG file in Python, you can begin to define complex relationships between data and actions.
You can see this flexibility in the following real-world example.
Using a single DAG, we are able to:

- Extract data from a legacy data store and load it into an AWS S3 bucket.
- Either train a data model or complete a data transformation, depending on the data we're using.
- Store the results of the previous action in a database.
- Send information about the entire process to various metric and reporting systems.

That's a lot of work for a single Python file! And if we need to expand on this workflow, all we need to do is add another node/task to our DAG.

A real world implementation of Apache Airflow



A few words on Airflow operators

Q: What are Airflow operators and and how to use them?

Operators are the building blocks of Airflow. You can think of them as wrappers around each task that defines how that task is going to be run. It abstracts away a lot of the code that you would otherwise have to write yourself. One of the great benefits of Airflow's community is that so many operators have been committed to cover a huge array of use cases. There are **action operators** that execute a function—e.g. a pipeline operator executes a Python function or the symbol HTTP operator executes an API call. There are **transfer operators**, which move data from source to destination, such as the S3 to GCS operator. And then there are **sensor operators** which wait for something to happen, like the external tasks sensor. Of course, if none of these is what you're looking for, you can always create your own and maybe even contribute it back to the open-source project.

Q: Users often ask about the KubernetesPodOperator. Why is it so important?

The KubernetesPodOperator spins up a pod to run a docker container and then spins down that pod when the task is complete. To use it you have to have a dockerized Airflow set up (which you would have if you were running it on Astronomer) and then you have to pass the operator a docker image for the container that you want to run.

One great thing about the KubernetesPodOperator is that it enables task-level resource configuration, and that's one way to manage custom Python dependencies. Ultimately, it allows Airflow to act as a job orchestrator, no matter what language those jobs are written in.

ASTRONOMER

Want to know more?

Check out our blog with more in-depth articles on Apache Airflow and data management.

Explore our blog

Get Started with Airflow

Want to learn more about Airflow, but not sure where to go next? We've compiled a list of Astronomer's best educational resources across the web for running Airflow at any scale. Whether you're a field engineer trying to deploy your first DAG or a CTO planning to adopt Airflow at your organization, we have you covered.

✦ Mastering Airflow

Improve your DAG writing and data orchestration skills by working through key guides and tutorials developed by the Astronomer team.

✦ Airflow Concepts

Use these resources to learn the basics of Apache Airflow

VIDEO
1. Coding Your First DAG for Beginners
SEE VIDEO ⟶

WEBINAR
2. Intro to Airflow
SEE WEBINAR ⟶

GUIDE
3. Understanding the Airflow UI
SEE GUIDE ⟶

GUIDE
4. Operators 101
SEE GUIDE ⟶

GUIDE
5. Airflow Executors Explained
SEE GUIDE ⟶

GUIDE
6. Managing Your Connections in Apache Airflow
SEE GUIDE ⟶

✦ Writing DAGs

Improve your Airflow development skills.

VIDEO
1. Introducing Airflow 2.0
Discover more about each of the key new features which arrived with this momentous Airflow release.
SEE VIDEO ⟶

GUIDE
2. DAG Writing Best Practices
How to create effective, clean, and functional DAGs.
SEE GUIDE ⟶

GUIDE
3. Using Airflow to Execute SQL
Learn to execute queries, paramaterize queries, and embed SQL-driven ETL in Apache Airflow DAGs.
SEE GUIDE ⟶

GUIDE
4. Using Task Groups in Airflow
Learn how to use Task Groups to build modular workflows in Airflow.
SEE GUIDE ⟶

✦ Third Party Support

Make the most of Airflow by connecting it to other tools.

GUIDE

1. Organizing Databricks Jobs with Apache Airflow

Process large amounts of data from your Apache Airflow DAGs using Databricks.

SEE GUIDE ⟶

GUIDE

2. Organizing Azure Container Instances with Airflow

Organize Azure containers directly from your DAGs.

SEE GUIDE ⟶

GUIDE

3. Integrating Airflow and Great Expectations

Using the Great Expectations provider natively in your Airflow DAGs.

SEE GUIDE ⟶

GUIDE

4. Integrating Airflow and Hashicorp Vault

Pull connection information from your Hashicorp Vault to use in your Airflow DAGs.

SEE GUIDE ⟶

✦ Adopting Airflow

Because Airflow is capable of fulfilling so many use cases, running Airflow at scale can often seem like a daunting task. Thankfully, Astronomer has produced both tools and educational resources for helping teams adopt Airflow at scale.

✦ Deploying Airflow

Spin up an Airflow cluster on your local machine or the cloud.

VIDEO

1. The Easy and Fast Way to run Apache Airflow in Production

Set up a local development environment using the open source Astronomer CLI.

SEE VIDEO ⟶

VIDEO

2. Running Airflow 2.0 with Docker in 5 Minutes

Quickly spin up an Airflow 2.0 environment using Docker.

SEE VIDEO ⟶

✦ Managing Airflow

Scale your existing Airflow Deployments to meet the needs of your team.

GUIDE
1. Scaling Out Airflow
How to scale out Airflow workers and the settings needed to maximize parallelism.
SEE GUIDE →

DOCUMENT
2. Access Control
How to create effective, clean, and functional DAGs.
SEE DOCUMENT →

✦ Simplifying Airflow

Learn about Astronomer's ability to simplify your Airflow management.

DOCUMENT
1. Astronomer Cloud Quickstart
Get started with Astronomer Cloud.
SEE DOCUMENT →

DOCUMENT
2. Deploy DAGs via the Astronomer CLI
How to push DAGs to your Airflow Deployment on Astronomer using the Astronomer CLI.
SEE DOCUMENT →

DOCUMENT
3. Manage User Permissions on Astronomer
Add and moderate users directly from the Astronomer UI.
SEE DOCUMENT →

Ready for more?

Check out our lineup of webinars — from introductory content to super advanced tutorials, we cover the most common topics around Apache Airflow and data management!

You can expect practitioners from our team to share their insights, best practices and know-how!

Register today!

Major features in Apache Airflow 2.0

In this chapter you'll learn about:

- **Highly available Scheduler**
- **Full REST API**
- **Smart sensors**
- **TaskFlow API**
- **Independent Providers**

In December of 2020, the Airflow community released Airflow 2.0. This momentous release delivered significant improvements, many of which were inspired by feedback from Airflow's 2019 Community Survey. The release also included hundreds of bug fixes and quality of life improvements. Here, you'll find the overview of the major features of Airflow 2.0.

A New Scheduler: Low-Latency + High-Availability

The Airflow Scheduler as a core component has been key to the growth and success of the project following its creation in 2014. As Airflow matured and the number of users running hundreds of thousands of tasks grew, we at Astronomer saw a great opportunity in driving a dedicated effort to improve upon Scheduler functionality and push Airflow to a new level of scalability.

In fact, "Scheduler Performance" was the most asked for improvement in the Community Survey. Airflow users have found that while the Celery and Kubernetes Executors allow for task execution at scale, the Scheduler often limits the speed at which tasks are scheduled and queued for execution. While effects vary across use cases, it's not unusual for users to grapple with the induced downtime and a long recovery in the case of a failure and experience high latency between short-running tasks.

It is for that reason that we're beyond ecstatic to introduce a new, refactored Scheduler with the Airflow 2.0 release. The most impactful Airflow 2.0 change in this area is the support for running multiple schedulers concurrently in an active/active model. Coupled with DAG Serialization, Airflow's refactored Scheduler is now highly available, significantly faster, and infinitely scalable. Here's a quick overview of new functionality:

1. Horizontal Scalability

If task load on 1 Scheduler increases, a user can now launch additional "replicas" of the Scheduler to increase the throughput of their Airflow Deployment.

2. Lowered Task Latency

In Airflow 2.0, even a single scheduler has proven to schedule tasks at much faster speed with the same level of CPU and Memory.

3. Zero Recovery Time

Users running 2+ Schedulers will see zero downtime and no recovery time in the case of a failure.

4. Easier Maintenance

The Airflow 2.0 model allows users to make changes to individual schedulers without impacting the rest and inducing downtime.

The Scheduler's zero recovery time and readiness for scale eliminate it as a single point of failure within Apache Airflow. Given the significance of this change, our team recently published "The Airflow 2.0 Scheduler", a blog post that dives deeper into the story behind Scheduler improvements alongside an architecture overview and benchmark metrics.

For more information on how to run more than 1 Scheduler concurrently, refer to the official documentation on the Airflow Scheduler.

Full REST API

Data engineers have been using Airflow's "Experimental API" for years, most often for triggering DAG runs programmatically. With that said, the API has historically remained narrow in scope and lacked critical elements of functionality, including a robust authorization and permissions framework. Airflow 2.0 introduces a new, comprehensive REST API that sets a strong foundation for a new Airflow UI and CLI in the future. Additionally, the new API:

- Makes for easy access by third-parties
- Is based on the Swagger/OpenAPI Spec
- Implements CRUD (Create, Update, Delete) operations on all Airflow resources and
- Includes authorization capabilities (parallel to those of the Airflow UI)

These capabilities enable a variety of use cases and create new opportunities for automation. For example, users now can programmatically set Connections and Variables, show import errors, create Pools, and monitor the status of the Metadata Database and Scheduler.

For more information, reference Airflow's REST API documentation.

Smart Sensors

In the context of dependency management in Airflow, it's been common for data engineers to design data pipelines that employ Sensors. Sensors are a special kind of Airflow Operator whose purpose is to wait on a particular trigger, such as a file landing at an expected location or an external task completing successfully. Although Sensors are idle for most of their execution time, they nonetheless hold a "worker slot" that can cost significant CPU and memory. The "Smart Sensor" introduced in Airflow 2.0 is a foundational feature that:

- Executes as a single, "long running task"
- Checks the status of a batch of Sensor tasks
- Stores sensor status information in Airflow's Metadata DB

This feature was proposed and contributed by Airbnb based on their experience running an impressively large Airflow Deployment with tens of thousands of DAGs. For them, Smart Sensors reduced the number of occupied worker slots by over 50% for concurrent loads in peak traffic.

To learn more, refer to Airflow documentation on Smart Sensors.

TaskFlow API

While Airflow has historically shined in scheduling and running idempotent tasks, it lacked a simple way to pass information between tasks. Let's say you are writing a DAG to train some set of Machine Learning models. The first set of tasks in that DAG generates an identifier for each model and the second set of tasks outputs the results generated by each of those models. In this scenario, what's the best way to pass output from the first set of tasks to the latter?

Historically, XComs have been the standard way to pass information between tasks and would be the most appropriate method to tackle the use case above. As most users know, however, XComs are often cumbersome to use and require redundant boilerplate code to set return variables at the end of a task and retrieve them in downstream tasks.

With Airflow 2.0, we're excited to introduce the TaskFlow API and Task Decorator to address this challenge. The TaskFlow API implemented in 2.0 makes DAGs significantly easier to write by abstracting the task and dependency management layer from users. Here's a breakdown of incoming functionality:

1. **A framework that automatically creates PythonOperator tasks from Python functions and handles variable passing.** Now, variables such as Python Dictionaries can simply be passed between tasks as return and input variables for cleaner and more efficient code.

2. **Task dependencies are abstracted and inferred as a result of the Python function invocation.** This again makes for much cleaner and more simple DAG writing for all users.

3. **Support for Custom XCom Backends.** Airflow 2.0 includes support for a new `xcom_backend` parameter that will allow users to pass even more objects between tasks. Out-of-the-box support for S3, HDFS, and other tools is coming soon.

It's worth noting that the underlying mechanism here is still XCom and data is still stored in Airflow's Metadata Database, but the XCom operation itself is hidden inside the PythonOperator and is completely abstracted from the DAG developer. Now, Airflow users can pass information and manage dependencies between tasks in a standardized Pythonic manner for cleaner and more efficient code.

To learn more, refer to **Airflow documentation on the TaskFlow API** and the **accompanying tutorial**.

Independent Providers

One of Airflow's signature strengths is its sizable collection of community-built Operators, Hooks, and Sensors—all of which enable users to integrate with external systems like AWS, GCP, Microsoft Azure, Snowflake, Slack, and many more.

Providers have historically been bundled into the core Airflow distribution and versioned alongside every Apache Airflow release. As of Airflow 2.0, they are now split into their own **airflow/providers** directory such that they can be released and versioned independently from the core Apache Airflow distribution. Cloud service release schedules often don't align with the Airflow release schedule and either result in incompatibility errors or prohibit users from being able to run the latest versions of certain providers. The separation in Airflow 2.0 allows the most up-to-date versions of

Provider packages to be made generally available and removes their dependency on core Airflow releases.

It's worth noting that some operators, including the Bash and Python Operators, remain in the core distribution given their widespread usage.

To learn more, refer to [Airflow documentation on Provider Packages].

To support the providers framework and help the Airflow community get more out of it Astronomer launched Astronomer Registry. It's designed to help anyone from a major technology company to an individual developer writing DAGs publish a provider from any public repository. For more on the Registry, read the announcement post.

Get the Airflow news straight into your inbox!

Sign up for Astronomer's newsletter and never miss an update on all things data!

Sign up

Frequently Asked Questions

INSIGHTS
Viraj Parekh
Field CTO Astronomer

We have gathered some of the most common questions that we hear during our tech talks. Check out our Field CTO Viraj Parekh's insights

Q: Why should I be using Apache Airflow vs. any other similar tool?

Here you'll find the most important reasons for using Apache Airflow:

1. **Proven core functionality for data pipelining**—Airflow's core capabilities are used by thousands of organizations in production, delivering value across scheduling, scalable task execution and UI-based task management and monitoring.

2. **An extensible framework**—Airflow was designed to make integrating existing data sources as simple as possible. Today it supports over 70 providers, including AWS, GCP, Microsoft Azure, Salesforce, Slack, and Snowflake. Its ability to meet the needs of simple and complex use cases alike makes it both easy to adopt and scale.

3. **It's scalable**—From a few pipelines to thousands of them running everyday.

4. **It's got a large and vibrant community** — Airflow has thousands of users and over 1,600 contributors who regularly submit features and provide the continuous improvement of the tool. In 2020, Airflow reached 10,000 commits and 18,000 GitHub stars and over 13,000 members of the Slack community.

So, if you want something that's flexible, standardized, and scalable that has the ongoing support of the open-source community— Airflow is perfect for you.

Q: Is Apache Airflow hard to learn?

Apache Airflow is not difficult to learn and implement because of a tool like Astronomer. Plus, everyone uses it, which means there are a lot of resources and documentation that can help you get started.

Q: What companies use Apache Airflow?

Walmart, Tinder, JP Morgan, Tesla, Revolut....just to name a few! Currently, hundreds of companies around the world, from various industries like healthcare, AI, fintech, or e-commerce, use Apache Airflow as part of their modern data stack to help them make sense of their data and drive better business decisions. It's time you become one of them!

Q: How functional data pipelines can improve my business?

Data pipelines help users with standard information. There's table stakes stuff where you can check what your sales were yesterday, what are your basic metrics for the day, etc. Data pipelines empower those users who are looking for baseline insights into the business.

Once you get past the basics, they can be used to give you a competitive advantage. Data pipelines expand into machine learning, AI, and other sorts of business intelligence around your consumers to try and predict the future.

Hear from the Experts

Check out our podcast for community interviews focused on the fundamental use cases driving Airflow forward. Available wherever you get your podcasts.

Check the podcast

Airflow Certification

Level Up Your Airflow Game with Astronomer!

INTERVIEW

Marc Lamberti

Head of Customer Training at Astronomer

Get a glimpse of the Apache Airflow Certification in the interview with Marc Lamberti, Head of Customer Training at Astronomer.

In this chapter you'll learn:

- **About the Astronomer Certification for Apache Airflow Fundamentals**
- **Why is the exam for you**
- **What are the key benefits**

Q: What is the goal of the Astronomer Certification for Apache Airflow Fundamentals?

The goal of the course and exam is for engineers to be able to show that they have the ability to create functional data pipelines; they know how to get started with Airflow, how to run a data pipeline, run different tasks and monitor them. It gives you the fundamental knowledge to be able to move forward in your Apache Airflow career and proves to companies that you are able to jump on Airflow right away.

Q: Who is the Airflow tutorial for?

I recommend the certification for Airflow users who have been using the tool for at least 3-4 months, and are looking to prove and enhance their basic knowledge and skills of the project. However, the preparation course is not only designed for passing the certification but also for people who are entirely new to Apache Airflow. Therefore, if you're a beginner and want to learn how to get started with Airflow in only 3 hours—the prep course is for you.

Q: What are the key benefits of the course?

First of all, it's a great way to assess and evaluate your skills, to know where you are at. Secondly, it will help you refresh your knowledge of Airflow, go back to some fundamentals you might have missed or forgotten about. And lastly, the certification can help you show the companies that you are able to use Airflow on a daily basis—it's the ultimate proof of your skills.

Q: What do people usually ask about during the course?

Usually, people are worried about whether they know enough to pass the certification and you might be asking the same question. Let me tell you, there's everything you need in the Apache Airflow preparation course. I made it with engineers like you in mind. Also, people ask if they can retry taking the certification if they fail the first time. Luckily for them, they can retake the exam after one month for free. And finally, they ask when the next certification will come. For now, I can only say we're working on it, so stay tuned;)

If you are interested in the certification, check out the details here!

Apache Airflow Fundamentals

The Astronomer Certification for Apache Airflow Fundamentals exam assesses an understanding of the basics of the Airflow architecture and the ability to create basic data pipelines for scheduling and monitoring tasks.

Concepts Covered:

- User Interface
- Scheduling
- Backfill & Catchup
- DAG Structure
- Architectural Components
- Use Cases

Get Certified

Airflow in Business

How Cloud Agronomics Processed 5x More Data with Astronomer

Check out the story of Cloud Agronomics to see the power of Airflow in business!

Key project highlights:

- **5x bigger data volume being processed in the same time period**
- **4x more structured data pipelines within the organization**
- **10x higher YoY Annual Recurring Revenue in 2021 compared to 2020**

Astronomer product values:

- Developer productivity — eliminating the issue of undifferentiated heavy lifting by providing automation tools.
- Day-2 operations — adding scalability, diagnostics, automation, and observability.
- Business impact — helping the client deliver the right data fast to their customers.
- Continuous innovation — allowing the client to stay on top of Airflow advancements.
- Mission critical — uninterrupted business operations thanks to 24/7 support.

✦ Ground–truth datasets for agriculture

Cloud Agronomics is a tech company that uses hyperspectral imagery, satellite data, and machine learning to challenge the way agriculture has been done for millennia. With the help of Astronomer, they've built the ag industry's largest cloud–based hyperspectral data pipeline. Here at Astronomer, we believe in supporting amazing startups like Cloud Agronomics in their data management journey as it allows them to grow and change the world for the better.

The agriculture industry has only recently joined the game of cloud computing and AI due to the lack of reliable data. The company's goal is to fill that gap and help agribusinesses produce sustainable, carbon–negative, resilient, and nutritious food. By using aircraft to gather proprietary data, the Cloud Agronomics platform provides actionable metrics on soil and plant productivity, allowing customers to make better, more sustainable decisions, as well as quantify changes to their field's Soil Organic Carbon (SOC) content.

> *"We try to give food a story — how it was planted, where it came from, how it got to the shelf in your grocery store. People should know how sustainable that journey was",*

explains Benjamin Leff, VP, Engineering at Cloud Agronomics.

✦ The challenge – leveraging big data

In the AgTech world, due to weather conditions and the nature of the local biosphere, you're relying on ever–changing, dynamic data. In order to deliver the right information quickly to their customers, Cloud Agronomics needed a way to build reliable and scalable data pipelines.

"Timing is extremely relevant to our business," says Benjamin, *"we have about 48 hours between flying the aircraft and delivering insights to our customers."*

Cloud Agronomics deals with terabytes of data and hundreds of thousands of files — called "Datacubes" — at a time. They have to be processed, analyzed, and delivered to customers on time. If they're not, and there's a change in plant nutrient uptake, the information delivered becomes irrelevant, putting the farmers' crops and businesses in danger.

To solve these problems, the client looked into different solutions that would:

- Allow their developers to be more productive (eliminating the undifferentiated heavy lifting)
- Enhance operational efficiency
- Easily manage infrastructure that gives them insights into possible issues
- Simply work (be out–of–the–box and easily integrated)

Apache Airflow caught their attention as a lot of Cloud Agronomics' data engineers are Python experts. Additionally, Airflow has a massive community behind it, and therefore, easily accessible support. After looking into many similar tools, choosing Apache Airflow was a no–brainer as it turned out to be the most mature, scalable, flexible, and simplest one to use.

✦ Astronomer on Microsoft Azure

Getting Airflow up and running and managing it on your own is difficult, which is why the company came to us for help. Having already been using our SaaS cloud service (Astronomer Cloud), they wanted to leverage the power of Microsoft Azure and their partnership with Microsoft to run the Astronomer Platform in their own cloud.

The switch went smoothly as our platform runs natively on Azure Kubernetes Service (AKS), which made it easy for the Cloud Agronomics workloads to run as KuberneretesPodOperators. This gives them the flexibility to run whatever docker images they like, all coordinated from Airflow. The client was also able to easily optimize their Airflow environment to fit their use case by switching to the Celery Executor.

Additionally, Cloud Agronomics gets ongoing support for their data engineering team with the Astronomer Platform.
"Astronomer makes Airflow so easy to use and get started with," says Benjamin, *"The abstractions they have provided on top of that — nobody holds a candle."*

> *"Astronomer's expertise in working with advanced Airflow features and integrations such as Kubernetes functionality helped jumpstart our transition to Airflow. It would have been significantly harder to approach these problems on our own,"*

says Annie Weinmann, Data Engineer at Cloud Agronomics.
Airflow allows Cloud Agronomics to automate their data pipelines and work management to provide a reliable way to operate their business at scale. They can now process discrete, smaller batches of data, saving compute time and making reprocessing easier, ultimately saving them time and money.

"Thanks to the infrastructure created by Airflow, we transformed our data pipelines from systems that didn't allow retries and restarts throughout the pipeline to a flexible, scalable, and reliable pipeline that can easily handle and adapt to unexpected changes," Annie confirms, *"As an added bonus, Airflow's GUI makes it easy to visualize exactly what's going on in our pipeline, too."*

Thanks to the Astronomer platform, instead of spending countless hours managing their data pipelines, they can focus on putting those resources somewhere else: into growing the company and providing better services for their customers.

"Unless you want to spend a couple of million dollars rolling your own solution, and dealing with all the problems that come with it, use Airflow and Astronomer. They've already put the great work in," sums up Benjamin.

ASTRONOMER

Thank you!

Hope you had a great time reading our introductory guide. We would love to hear your feedback, so make sure to follow us on Twitter and Linkedin!

Start building your next-generation data platform today.

Get Astronomer

Experts behind:

Viraj Parekh | Field CTO Astronomer
Kenten Danas | Field Engineer Astronomer
Marc Lamberti | Head of Customer Training Astronomer