Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

# Apache Airflow 3 for Developers

## Class Duration

21-28 hours of live training delivered over 3-4 days. The longer format adds depth on Kubernetes execution, custom providers, and testing.

## Student Prerequisites

- Practical experience with Python
- Basic Linux command line skills
- Familiarity with containers
- Basic Kubernetes concepts are helpful for the Kubernetes module
- No prior Airflow experience is required

## Target Audience

Python developers, data engineers, and DevOps engineers who build and maintain pipelines with Apache Airflow 3. Teams moving DAGs from Airflow 2 get notes on what changed. Platform teams that deploy and upgrade Airflow itself can pair this with Apache Airflow 3 Operations and Administration.

## Description

This course teaches developers to build pipelines with Apache Airflow 3. Students learn the Airflow 3 architecture, set up a local environment, and write DAGs with the Task SDK. They schedule work with timetables, assets, and asset watchers, and use DAG bundles for versioning. Later modules cover dynamic task mapping, deferrable operators, XCom, and running tasks on Kubernetes. The course ends with custom providers, the Common AI provider, testing, and observability.

## Learning Outcomes

- Explain the Airflow 3 architecture, including the Task SDK and Execution API.
- Write DAGs with the TaskFlow API and the Operator API.
- Schedule workflows with timetables, assets, and asset watchers.
- Manage DAG changes with DAG bundles and DAG versioning.
- Apply dynamic task mapping without overloading the scheduler.

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Wait on external systems with sensors and deferrable operators.
- Keep XCom data small and offload large values to object storage.
- Run tasks on Kubernetes with the KubernetesExecutor and KubernetesPodOperator.
- Write custom operators, hooks, and provider packages.
- Call LLMs and agents from DAGs with the Common AI provider.
- Test DAGs and instrument them with OpenLineage and OpenTelemetry.

## Training Materials

Comprehensive courseware is distributed online at the start of class, including reference documents and code samples. All students receive a downloadable MP4 recording of the training.

## Software Requirements

- Free, personal GitHub account to access the courseware
- Courseware targets Apache Airflow 3.4 on Python 3.11 to 3.14
- Permission to install Python, Visual Studio Code, Docker Desktop or Podman, Minikube or kind, kubectl, and Helm
- Permission to install Python packages and VS Code extensions, and to download container images
- If students are unable to configure a local environment, a cloud-based environment can be provided

## Training Topics

### What is Apache Airflow?

- Workflow orchestration and workflows as code
- Airflow vs. cron, Celery, and event pipelines
- DAGs, tasks, and DAG runs
- Airflow 3 services
- The Task SDK and Execution API
- Executor types
- Changes from Airflow 2

### Local Development Environment

- Virtual environments and the constraints file
- Standalone mode for local development

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Running services separately
- PostgreSQL as the metadata database
- Exploring the Airflow UI

**Authoring DAGs with the Task SDK**

- The airflow.sdk authoring interface
- TaskFlow API with @dag and @task
- The Operator API
- Task dependencies
- Jinja templating and params
- The runtime context
- Using connections and variables in code
- Async Python tasks
- Task groups and task loops
- Human-in-the-loop tasks
- Top-level code pitfalls
- Non-idempotent tasks and hidden state

**Scheduling, Assets, and DAG Versioning**

- Cron expressions and timetables
- Catchup and backfills
- Asset-based scheduling and the @asset decorator
- Event-driven scheduling with asset watchers
- Asset partitions and partition mappers
- Git and local DAG bundles
- DAG versioning
- Rerunning on the original or latest bundle version
- Task and asset state stores

**Dynamic Tasks, Branching, and Deferrable Operators**

- Dynamic task mapping
- expand, partial, and expand_kwargs
- High-fanout mapping and scheduler load
- Limiting fan-out with max_active_tis_per_dag
- Pools, chunking, and batching
- Branching and trigger rules
- Sensor poke and reschedule modes

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 

- Sensors and worker slots
- Deferrable operators, async triggers, and the Triggerer
- Writing custom deferrable operators
- Retries and pluggable retry policies

**XCom and Inter-Task Data**

- How XCom works in Airflow 3
- XCom through the Execution API
- XCom push and pull
- TaskFlow return values
- Passing too much data between tasks
- The Object Storage XCom backend
- Size thresholds for offloading
- Writing custom XCom backends

**Running Airflow Tasks on Kubernetes**

- Deploying to Minikube or kind with Helm
- Building custom container images
- How the KubernetesExecutor works
- Worker Pod lifecycle with the Task SDK
- Pod templates
- Per-task resource requests and limits
- Affinity, tolerations, and node selectors
- Pod cleanup and failed Pods
- Debugging with kubectl logs and exec
- KubernetesPodOperator for container tasks
- Volumes and environment variables
- Secrets and image pull secrets
- RBAC and service accounts
- In-cluster vs. out-of-cluster Pods
- KubernetesPodOperator vs. KubernetesExecutor
- Image pull failures
- Resource starvation and Pod eviction
- Lost task logs

**Celery Executor and Mixed Executors**

- CeleryExecutor vs. KubernetesExecutor

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Multiple executors instead of CeleryKubernetesExecutor
- Routing tasks to an executor or queue
- KEDA autoscaling for Celery workers

**Variables, Connections, and Secrets**

- Storing and retrieving variables
- Managing connections
- Secrets backends
- Handling secrets in DAG code

**Providers, Custom Operators, and Hooks**

- Custom code vs. existing operators
- Provider packages
- AWS and Azure providers
- Writing custom operators and hooks
- Writing Airflow plugins
- Packaging and publishing a provider
- The Common AI provider
- LLM tasks with @task.llm
- Agent tasks with @task.agent
- Experimental language SDKs for Go and Java

**Testing and Debugging DAGs**

- Testing DAGs with dag.test()
- Unit testing tasks, operators, and hooks
- DAG integrity checks in CI
- Debugging with task logs

**Observability and Production Practices**

- Notifications, callbacks, and Deadline Alerts
- Lineage with OpenLineage
- OpenTelemetry metrics and traces
- Idempotent, rerun-safe tasks
- Performance anti-patterns
- Common problems and fixes

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved 