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: Kubernetes-Native Workflow Orchestration

Class Duration

21 hours of live training delivered over 3 days.

Student Prerequisites

- Practical experience with Python
- Working knowledge of Airflow basics: DAGs, operators, tasks, and the Web UI
- Familiarity with containerization and basic Kubernetes concepts (Pods, Deployments, Services)
- Basic Linux command line skills

Target Audience

This is an intermediate-to-advanced course intended for developers already working with Apache Airflow. The course targets Airflow 3.2 with explicit callouts for teams still running 2.x. This course assumes the DAG-authoring fundamentals from Apache Airflow Programming: Developing, Configuring, and Automating Workflows.

Description

This intensive three-day course takes developers already comfortable with Airflow basics and goes deep on running Airflow 3.2 on Kubernetes. Students deploy Airflow to a local Minikube cluster with the official Helm chart, learn the new Airflow 3 architecture (API Server, Task SDK, Execution API, DAG Processor isolation), and master the Kubernetes Executor and KubernetesPodOperator for launching arbitrary Pods into the cluster as part of a DAG. The Celery Executor is covered as a comparison, along with the new Multiple Executors Concurrently feature that replaces the removed CeleryKubernetesExecutor, including when to choose each. A DAG authoring refresher introduces Assets and DAG Versioning as first-class topics, with notes on Airflow 2 differences for teams not yet upgraded. Advanced topics

include dynamic task mapping (and how to avoid overwhelming the control plane with high-fanout mapped tasks), inter-task data flow with the Object Storage XCom backend, sensors and deferrable operators, and writing maintainable custom operators, hooks, plugins, and provider packages under the airflow.sdk namespace. The course concludes with production best practices, common problems and concrete mitigations, OpenLineage and OpenTelemetry observability, and operating Airflow at scale including multi-team support and the Edge Executor for remote work.

Learning Outcomes

- Understand the Airflow 3 architecture (API Server, Task SDK, Execution API, DAG Processor isolation, and the airflow.sdk namespace) and author DAGs using Assets, DAG Versioning, the TaskFlow API, and the Operators API.
- Deploy Airflow 3.2 to a Minikube cluster using the official Helm chart and a custom container image.
- Operate the Kubernetes Executor effectively, including pod templates, resource controls, scheduling, and debugging.
- Use the KubernetesPodOperator (apache-airflow-providers-cncf-kubernetes) to launch arbitrary Pods into the cluster.
- Configure and operate the Celery Executor and use Multiple Executors Concurrently as the replacement for CeleryKubernetesExecutor.
- Use dynamic task mapping responsibly to avoid overwhelming the scheduler and control plane, and manage inter-task data flow with Airflow 3 external XCom storage and the Object Storage XCom backend.
- Handle external dependencies efficiently with sensors and deferrable operators backed by the Triggerer, and extend Airflow with custom operators, hooks, plugins, and provider packages.
- Apply best practices, recognize common production problems, implement concrete mitigations, and instrument Airflow with OpenLineage and OpenTelemetry for lineage and tracing.

Training Materials

Students receive comprehensive courseware, including reference documents, code samples, and lab guides. All students receive a downloadable MP4 recording of the training.

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

Software Requirements

- Free, personal GitHub account to access the courseware
- Permission to install Python, Visual Studio Code, Docker Desktop, Minikube, kubectl, and Helm
- Permission to install Python packages and VS Code extensions, and to download Docker images
- If students are unable to configure a local environment, a cloud-based environment can be provided

Training Topics

Airflow 3 Architecture Review

- API Server (FastAPI + React UI) replaces the Flask Webserver
- Task SDK and the Execution API: workers no longer touch the metadata DB
- DAG Processor separation (software-level guards; hard isolation is still WIP)
- Scheduler, Triggerer, and metadata database
- Executor types overview: Local, Celery, Kubernetes, Edge, and Multiple Executors Concurrently
- When to choose the Kubernetes Executor
- Airflow 2 to 3 migration notes (for teams not yet upgraded)

DAG Authoring Refresher (Airflow 3 Edition)

- DAG anatomy and lifecycle
- Operators API and TaskFlow API with airflow.sdk imports
- Tasks, schedules, and DAG runs
- Assets and asset-aware scheduling (renamed from Datasets, AIP-74/75)
- Asset Partitions (new in 3.2, AIP-76)
- DAG Bundles (Git, local) and how they enable DAG Versioning
- The @asset decorator
- DAG Versioning (new in Airflow 3)
- Jinja templating in DAGs
- Async PythonOperator (new in Airflow 3.2)
- Best practices and common authoring pitfalls

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

Airflow on Minikube

- Setting up Minikube for Airflow 3
- kubectl and Helm essentials for Airflow operators
- The official Apache Airflow Helm chart (Airflow 3 compatible)
- Required infrastructure: PostgreSQL and optional Redis
- API Server and Task SDK components on Kubernetes
- Building and using custom container images
- Installing providers and extra dependencies
- airflow db migrate and the Airflow 2 to 3 upgrade path
- Live upgrades and configuration changes
- Common setup issues and mitigations

Kubernetes Executor Deep Dive

- How the Kubernetes Executor works (now in the cncf.kubernetes provider)
- Worker Pod lifecycle with Task SDK
- Pod templates and pod_template_file
- Per-task resource requests and limits
- Affinity, tolerations, and node selectors
- Pod cleanup and handling failed Pods
- Execution API communication and network security
- Debugging worker Pods with kubectl logs and exec
- Common issues: image pull, resource starvation, Pod eviction
- Best practices for the Kubernetes Executor

Celery Executor and Multiple Executors Concurrently

- When to use Celery Executor vs Kubernetes Executor
- CeleryKubernetesExecutor was removed in Airflow 3.0: use Multiple Executors Concurrently instead
- Redis as the Celery broker
- Running Celery workers on Minikube
- Monitoring Celery workers with Flower
- KEDA autoscaling for Celery workers

KubernetesPodOperator

- PodOperator vs Kubernetes Executor: when to use which
- apache-airflow-providers-cncf-kubernetes provider package (v10.x or later)

- Creating arbitrary Pods in the cluster as DAG tasks
- Volume mounts and Persistent Volume Claims
- Environment variables and Secrets
- Resource requests and limits
- Image pull secrets and private registries
- In-cluster vs out-of-cluster Pods
- RBAC and service account configuration
- Common gotchas: timeouts, image pull failures, logs lost after cleanup
- Best practices for PodOperator at scale

Dynamic Task Mapping

- Mapping patterns: .expand() and .expand_kwargs()
- Mapped tasks in the API Server UI
- Common problem: overwhelming the scheduler and control plane with high-fanout mapping
- Mitigations: max_active_tis_per_dag, max_active_tasks, pools, chunking, and batching
- When mapped tasks are the wrong tool

XCom and Inter-Task Communication (Airflow 3 Edition)

- Airflow 3 XCom architecture: external storage with DB pointers, accessed via Execution API
- XCom push and pull patterns
- Cross-task data flow with the TaskFlow API
- Common problem: too much data between tasks
- The Object Storage XCom backend (apache-airflow-providers-common-io): the canonical custom backend
- Writing custom XCom backends
- Best practices for inter-task data flow

Variables, Connections, and Secrets

- Storing and retrieving Variables
- Managing Connections
- Secret backends (Kubernetes Secrets, HashiCorp Vault)
- Best practices for secret handling

Sensors and Deferrable Operators

- Sensors: poke vs reschedule mode

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

- The worker-slot problem with long-running sensors
- Deferrable operators and the Triggerer process
- Async triggers and the asyncio model
- Writing custom deferrable operators
- Common use cases for sensors and deferrable operators
- Best practices and common pitfalls

Custom Operators, Hooks, Plugins, and Providers

- When to write custom code vs use existing operators
- The Airflow plugin architecture (airflow.sdk namespace)
- Writing a custom operator
- Writing a custom hook
- Provider packages: structure, distribution, and the Apache Airflow Registry (website catalog) with AIP-95 provider lifecycle stages
- Packaging and publishing a provider to PyPI
- Testing custom operators and hooks
- Best practices for maintainable custom code

Best Practices, Common Problems, and Mitigations

- DAG design best practices
- Idempotency and retry safety
- Avoiding top-level code in DAG files
- Backfills, reruns, and catchup
- Performance anti-patterns and how to fix them
- Common problem catalog: symptoms, causes, and mitigations
- Production readiness checklist

Observability

- Log configuration and external log storage
- Metrics with StatsD and Grafana dashboards
- OpenLineage: built-in lineage (apache-airflow-providers-openlineage)
- OpenLineage events: producing and consuming lineage across pipelines
- OpenTelemetry tracing in Airflow 3 (custom spans via airflow.sdk.observability)
- Notifications and alerting

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

Operating Airflow at Scale

- Scheduler performance tuning
- Metadata database tuning
- Worker autoscaling strategies (KEDA)
- Multi-team isolation (Airflow 3.2; logical/perimeter separation, not hard isolation)
- Edge Executor for remote workers
- Capacity planning
- Cost optimization

