COURSE TABLE OF CONTENTS

Apache Airflow Fundamentals Certification Training (Astronomer v3)

OEM: Apache • Duration: 2 Days (16 hrs) • Code: ASTRO-AF3-FUND

COURSE MODULES & TOPICS

Module 1: Introduction to Orchestration & Airflow

- What is workflow orchestration and why Apache Airflow exists
- When Airflow is appropriate — and when it is not (not for streaming or data processing)
- Core use cases for Airflow in modern data stacks

Module 2: Airflow Architecture & Basics

- Core components: DAG Parser, API Server (formerly Web Server), Scheduler, Executor, Worker, Metadata DB
- DAG Parser scans DAGs folder every 5 minutes (dag_dir_list_interval)
- Scheduler reads Metadata DB and schedules Task Instances
- Executor pushes Task Instances to queue; Worker executes and reports status
- Default timezone: UTC
- New in Airflow 3: airflow.sdk stable import namespace for DAG, @dag, @task

Module 3: Local Development Environment

- Setting up Airflow locally with Astro CLI
- Docker-based local setup and airflow standalone command
- Environment configuration and managing dependencies

Module 4: The Airflow UI

- DAG list view, Grid view, Graph view, Calendar view, Gantt view
- Monitoring DAG and Task states from the UI
- Triggering, pausing, and clearing DAG runs
- Testing connections via Admin!' Connections!' TEST

Module 5: DAGs 101

- Required parameter: dag_id; valid DAG declaration syntax (DAG(), context manager, @dag decorator)
- default_args dictionary — applying parameters across all tasks
- Recommended DAG parameters: description, tags, schedule, start_date, catchup, max_active_runs

- DAG run states: queued → running → success/failed; logical_date, run_id
- Task dependency notation: >> (forward), << (reverse); chain() for lists

Module 6: DAG Scheduling

- schedule parameter (renamed from schedule_interval in Airflow 2)
- Cron expressions, timedelta objects, @once, @continuous
- Airflow 3 key change: catchup=False is now the default
- Assets (formerly Datasets): event-driven scheduling via @asset decorator (AIP-74/75)
- Backfill as first-class citizen — scheduler-managed via CLI, UI, REST API (AIP-78)
- DAG Versioning — Airflow 3 tracks structural DAG changes over time (AIP-66)

Module 7: Connections 101

- Connection components: conn_id, connection type, host, port, login, password, extra
- Creation methods: UI, CLI, environment variables (AIRFLOW_CONN_...), REST API, Secrets Backend
- Encryption enabled by default; env var connections not visible in UI but still accessible

Module 8: XComs 101

- Purpose: passing small metadata/data between tasks (must be JSON-serializable)
- Stored in Metadata DB via API Server; pull requires task_id, run_id, key
- Size limits by database: SQLite (2GB), Postgres (1GB), MySQL (64KB)

Module 9: Variables 101

- JSON key-value store with Key/ID, Value, optional Description
- Creation methods: REST API, CLI, UI, environment variables
- Sensitive keywords auto-hidden: access_token, api_key, password

Module 10: Sensors

- Function: checks a condition and waits poke_interval seconds before rechecking
- Key parameters: poke_interval, timeout (default 1 week), mode
- Mode poke: holds worker slot (use when poke_interval "d 5 minutes)
- Mode reschedule: releases worker between checks (uses up_for_reschedule state)

Module 11: Debugging DAGs

- DAG not showing: check refresh intervals, dag_id uniqueness, .airflowignore
- DAG not running: verify unpause status, date ranges, parallelism limits
- airflow tasks test: runs a single task without dependencies or XCom storage
- DAG deletion: removes all run history and task instances from Metadata DB

Module 12: Airflow CLI

- airflow db init, airflow users create, airflow standalone, airflow info

- airflow tasks test — run single task without dependencies
- airflow dags backfill — historical run execution
- Import/export: connections, pools, users, variables