Apache Airflow DAG Authoring Certification Training (Astronomer v3)

OEM: Apache • Duration: 3 Days (24 hrs) • Code: ASTRO-AF3-DAG

COURSE MODULES & TOPICS

Module 1: DAG Authoring Basics

- DAG detection and Python file requirements
- start_date and scheduling mechanics
- Catchup behavior (Airflow 3 default: catchup=False) and state clearing
- Deterministic and idempotent task design principles
- default_args and task-level overrides
- depends_on_past and wait_for_downstream behaviors

Module 2: Variables & Configuration

- Variable creation methods: UI, CLI, REST API, environment variables, Secrets Backend
- Best practice: access variables inside tasks, not at DAG parsing level
- Sensitive value handling and auto-masking
- Avoiding database hits on every DAG parse

Module 3: TaskFlow API & Operators

- Template fields and Jinja rendering
- @task, @task.python, @task.virtualenv decorators (TaskFlow API)
- Custom operator creation and template_fields discovery
- XCom data exchange and size limitations
- airflow.sdk import path for Airflow 3

Module 4: Task Grouping & DAG Dependencies

- TaskGroup implementation and nested TaskGroup capabilities
- SubDAG limitations (deprecated in Airflow 3 — use TaskGroups instead)
- Cross-DAG dependencies: ExternalTaskSensor, TriggerDagRunOperator
- Pool concurrency management and pool_slot parameter

Module 5: Branching, Trigger Rules & Dynamic DAGs

- BranchPythonOperator: conditional task routing

- Trigger rules (8 types): all_success, all_failed, all_done, one_failed, one_success, none_failed, none_skipped, none_failed_or_skipped
- Dynamic DAG generation: trade-offs (harder debugging, increased parse time)
- cross_downstream() for chaining dependency lists

Module 6: Concurrency, Versioning & Pools

- Concurrency control: Airflow-level and DAG-level settings
- pool and pool_slot parameters for resource management
- priority_weight for task execution ordering
- DAG Versioning (Airflow 3 native via AIP-66)

Module 7: Sensors, SLAs, Callbacks & Advanced Scheduling

- Sensor operators: timeout, poke_interval, reschedule mode
- DAG-level timeout (dagrun_timeout) and operator-level (execution_timeout)
- Callback functions: on_success_callback, on_failure_callback, on_retry_callback
- Retry mechanisms: retries, retry_delay, retry_exponential_backoff
- SLA (Service Level Agreement): time limits with alert triggers
- Asset-driven scheduling (Airflow 3): @asset decorator replacing Datasets (AIP-74/75)
- Dynamic task mapping (Airflow 3): .expand() and .partial() for runtime task counts
- Backfill as first-class citizen: scheduler-managed, UI/API accessible (AIP-78)