

**Introduction to Apache Airflow**

**Course Number:** PYTH-220
**Duration:** 3 days

**Overview**

This Introduction to Airflow training course teaches software engineers and data engineers how to use Apache Airflow to orchestrate production-ready data pipelines. Students learn how to create pipelines via DAGs (Directed Acyclic Graphs), make pipelines predictable, and master their scheduling. In addition, participants learn how to abstract functionality, reuse components, and modularize and share data across tasks and pipelines with the TaskFlow API, Task Groups, and Branching. Finally, attendees learn how to scale Airflow within Celery, Kubernetes, and KEDA executors.

Throughout the course, students incrementally construct a single, real-world application.

**Prerequisites**

All attendees should have basic Python knowledge or object-oriented programming experience.

**Materials**

All Airflow training students receive comprehensive courseware.

**Software Needed on Each Student PC**

* Python 3.5 or later
* Airflow 2.1 or later

**Objectives**

* Create production-ready data pipelines in Airflow
* Build pipelines in Airflow that are able to scale to hundreds of tasks
* Enforce modularization and reusability of Airflow tasks across projects
* Scale Airflow in Kubernetes

**Outline**

* Introducing Apache Airflow
	+ What Airflow is and what does it solve?
	+ Airflow architecture
	+ How do we represent a Pipeline?
	+ Our first DAG
	+ Tasks, TaskFlow, and Operators
	+ First Pipeline
* Mastering scheduling
	+ execution\_date, start\_date and schedule\_interval
	+ Handling non-default schedule\_intervals
	+ Playing with time
* Abstracting functionality
	+ Using custom operators
	+ Creating TaskGroups vs subDAGs
	+ Sharing data with xCOMs
	+ Branching and Triggers
	+ Sensors and SmartSensors
* Executors and Scaling Airflow
	+ Abandoning SQLite for PostgreSQL
	+ Executors: Debug, Local, Celery
	+ Concurrency and parallelism
	+ Concurrency with Celery
	+ Airflow in Kubernetes, the old and new ways
	+ KEDA and HA scheduler
	+ Deploying a highly availability fault-tolerant Airflow
* Conclusion