**🗓️ WEEK 1 — Core Airflow Foundations**

**Goal:** Understand how Airflow works, its components, and how to run simple DAGs inside Docker.

**🧩 DAY 1 — Understanding Airflow Architecture**

**Focus:** The big picture.

**What to Learn**

* Components:
  + **Scheduler:** Triggers and schedules tasks.
  + **Webserver:** UI and monitoring.
  + **Worker:** Executes tasks.
  + **Metadata DB:** Stores DAG runs, task states.
* DAG (Directed Acyclic Graph) = workflow of tasks.
* How tasks depend on each other.

**Activity (Hands-on)**

1. Draw (on paper or digitally) how Airflow components connect.
2. Open the UI in your browser and explore sections: *DAGs*, *Browse → Task Instances*, *Admin → Connections*.

**🧩 DAY 2 — Setting Up Airflow with Docker**

**Focus:** Environment setup and control.

**What to Do**

* Use the **official docker-compose.yaml** from Airflow’s docs.
* Run:
* curl -LfO 'https://airflow.apache.org/docs/apache-airflow/2.9.0/docker-compose.yaml'
* docker-compose up -d
* Access UI → [http://localhost:8080](http://localhost:8080/)  
  Default: user = airflow, pass = airflow

**Learn**

* Folder structure (dags/, logs/, plugins/)
* How to restart Airflow
* docker-compose down
* docker-compose up -d

**Goal:** You can start/stop and access Airflow UI from Docker.

**🧩 DAY 3 — Creating Your First DAG**

**Focus:** Understand DAG structure and scheduling.

**Learn**

* DAG structure:
* from airflow import DAG
* from airflow.operators.python import PythonOperator
* from datetime import datetime
* def greet():
* print("Hello Airflow!")
* with DAG(
* dag\_id="hello\_airflow",
* start\_date=datetime(2024, 1, 1),
* schedule\_interval="@daily",
* catchup=False
* ) as dag:
* task1 = PythonOperator(task\_id="greet\_task", python\_callable=greet)

**Practice**

* Save this in dags/hello\_airflow.py
* Open Airflow UI → Turn it ON → Trigger manually.
* See logs and task instance.

**Goal:** You can create and run a basic DAG.

**🧩 DAY 4 — Dependencies and Scheduling**

**Focus:** Control execution order.

**Learn**

* >> and << for task order.
* Example:
* task1 >> task2
* DAG parameters:
  + start\_date
  + schedule\_interval (e.g. "@daily", "0 6 \* \* \*")
  + catchup=False (avoid backfilling)

**Practice**

* Create 3 tasks (extract, transform, load)
* Chain them → extract >> transform >> load
* Trigger DAG and observe order.

**Goal:** You can define dependencies and understand schedules.

**🧩 DAY 5 — Variables, Connections, Configs**

**Focus:** Make DAGs flexible.

**Learn**

* **Variables** (Admin → Variables) store config values.
* **Connections** (Admin → Connections) store credentials (S3, DB, API).
* Access them in code:
* from airflow.models import Variable
* my\_var = Variable.get("my\_config")

**Practice**

* Create variable greet\_name = “Kabir”.
* Use it in your DAG’s greet() function.
* Print the value from code.

**🧩 DAY 6 — Airflow CLI Basics**

**Focus:** Command-line control.

**Learn**

* Common commands:
* docker exec -it airflow-scheduler airflow dags list
* docker exec -it airflow-scheduler airflow dags trigger hello\_airflow
* docker exec -it airflow-scheduler airflow tasks list hello\_airflow
* docker exec -it airflow-scheduler airflow tasks test hello\_airflow greet\_task 2025-01-01

**Practice**

* Trigger and test DAGs via CLI.

**Goal:** You can interact with Airflow without UI.

**🧩 DAY 7 — Review & Mini Project**

**Focus:** Combine it all.

**Mini Project**  
Build a small “Daily Logger DAG”:

1. Three tasks:
   * start\_log → print start time
   * process\_data → simple Python print
   * end\_log → print “Pipeline done!”
2. Add retry logic:
3. retries=2,
4. retry\_delay=timedelta(minutes=1)

**Outcome:** You now fully understand how Airflow runs, schedules, and executes tasks locally.

**🗓️ WEEK 2 — Operators, Sensors & XComs**

**Goal:** Learn to connect real tasks, manage data flow, and create functional DAGs.

**🧩 DAY 8 — Built-in Operators Overview**

**Learn**

* **Common Operators**
  + PythonOperator
  + BashOperator
  + EmailOperator
  + DummyOperator
* Understand their roles.

**Practice**

* Create a DAG with a mix:
  + Bash task: echo "Start"
  + Python task: print a message
  + Email task (optional if SMTP configured)

**🧩 DAY 9 — PythonOperator in Depth**

**Focus:** Real Python tasks inside Airflow.

**Practice**

* Create a DAG that:
  1. Reads a CSV file (local or from a URL)
  2. Calculates basic stats
  3. Logs them using PythonOperator

**Goal:** Comfortable writing logic inside Python tasks.

**🧩 DAY 10 — Bash & File Operators**

**Learn**

* Run shell scripts using BashOperator
* Use templated variables:
* bash\_command="echo {{ ds }}"

**Practice**

* Build a DAG that zips a file and prints date.

**🧩 DAY 11 — Sensors (Waiting for Events)**

**Focus:** Automation that waits.

**Learn**

* **Sensors** wait for conditions:
  + FileSensor
  + S3KeySensor
  + ExternalTaskSensor
* Syntax example:
* from airflow.sensors.filesystem import FileSensor
* wait\_for\_file = FileSensor(
* task\_id='wait\_for\_input',
* filepath='/opt/airflow/data/input.csv',
* poke\_interval=30,
* timeout=600
* )

**Practice**

* Add a FileSensor to your ETL DAG to wait until a file appears.

**🧩 DAY 12 — XComs (Passing Data Between Tasks)**

**Learn**

* How tasks share data:
* return value # push
* ti.xcom\_pull(task\_ids='previous\_task') # pull

**Practice**

* Task1 → generates a number
* Task2 → reads that number and doubles it
* Task3 → logs final output

**🧩 DAY 13 — Mini ETL Pipeline**

**Goal:** Combine everything.

**Project**

* DAG: simple\_etl
* Steps:
  1. Download a CSV file (PythonOperator)
  2. Transform it (PythonOperator)
  3. Save cleaned file locally or to S3 (BashOperator)
  4. Use XCom to pass filename between steps.

**Outcome**  
You now understand real-world DAG logic and communication between tasks.

**🧩 DAY 14 — Review & Debug**

**Focus:** Testing and troubleshooting.

**What to Do**

* Use CLI:
* airflow dags test simple\_etl 2025-01-01
* Check task logs, XCom values, and DAG structure.
* Document what worked and what failed.

**Goal:** You can build, debug, and monitor a functional Airflow workflow.

✅ **After Week 2:**  
You’ll be able to:

* Write fully functional Airflow pipelines
* Use Operators, Sensors, and XComs
* Handle dependencies and retries
* Test and debug with confidence