Introduction to Apache Airflow and Docker

Apache Airflow is an open-source workflow orchestration tool designed to programmatically author, schedule, and monitor workflows. Workflows are defined as Directed Acyclic Graphs (DAGs), where each node is a task.

Docker is a containerization platform that allows you to package applications and their dependencies into a single portable container. This makes it easier to distribute and deploy applications across different environments.

Why Use Airflow with Docker :

- Isolation: Each component runs in a separate container, ensuring clean dependency management.
- Portability: Move from development to production easily.
- Scalability: Add more workers and scale up with Docker Compose or Kubernetes.
- Simplified Setup: Using Docker Compose can spin up a full Airflow stack in seconds.

Architecture of Airflow with Docker

A typical Airflow Docker-based setup includes the following containers:

1. Webserver: The UI for managing and monitoring workflows.
2. Scheduler: Parses DAGs and schedules tasks.
3. Worker(s): Executes tasks, usually via Celery.
4. Database (PostgreSQL): Stores metadata (task status, DAG runs, logs).
5. Redis: Acts as a Celery message broker.
6. Flower: Web-based tool for monitoring Celery workers (optional).

These services interact using Docker networks. Airflow uses a shared volume for DAGs so that all containers can access workflow definitions.

Setting Up Airflow with Docker

Step 1: Prerequisites

- Docker and Docker Compose installed.
- Basic knowledge of Python and terminal commands.

Step 2: Clone Airflow Docker Compose Repository

git clone https://github.com/apache/airflow.git

cd airflow

Step 3: Initialize Environment

=>Open Notepad or VS Code.

=>Paste this line:

AIRFLOW_UID=50000

=>Save the file as .env in airflow dir

Step 4: Initialize Database and Services

docker-compose up airflow-init

Step 5: Start the Airflow Stack

docker-compose up

Access the Airflow UI at http://localhost:8080 with default credentials:

- Username: airflow
- Password: airflow

You can now view sample DAGs and start running your own workflows.

Real-World Usage, Tips, and Troubleshooting

Real-World Use Cases

- ETL Pipelines: Extract data from APIs, transform using pandas, load to databases.
- Machine Learning: Train models, evaluate, and deploy using DAG stages.
- Data Warehousing: Automate loading from S3/Blob storage to Snowflake/Redshift.

Tips

- Use .env files to manage credentials.
- Use external volumes for persisting logs and plugins.
- Schedule Airflow backups to avoid metadata loss.

Common Issues

- Docker Health Checks Failing: Check logs using docker-compose logs.
- Webserver Not Starting: Verify ports and database readiness.
- Scheduler Lagging: Ensure enough resources and monitor Celery queues.