**Airflow and MLflow Setup on Ubuntu Server using Docker**

**1. Ubuntu Server and Docker Installation**

* Set up an **Ubuntu server** (hosted on a public cloud or a local VM using VirtualBox).
* Installed **Docker** and **Docker Compose** following the official Docker installation guide.
* Enabled Docker in **systemd** to start automatically upon server reboot.
* Verified the installation by running basic Docker commands like docker ps.

**2. Deploying Airflow using Docker**

* Deployed **Apache Airflow** using **Docker Compose**.
* Retrieved the official docker-compose.yaml file and modified it to **remove the worker component**, ensuring all tasks execute within the scheduler.
* Stored the modified docker-compose.yaml in the project repository.
* Initialized Airflow’s database using:

docker-compose up airflow-init

* Started Airflow services:

docker-compose up -d

* Verified running containers using:

docker ps -a

* Accessed the Airflow UI via **http://172.25.20.135:8080**.
* Logged in using the default credentials (airflow/airflow) and confirmed the UI displayed pre-existing DAGs from the Airflow community.

**3. Deploying MLflow using Docker**

* Created a separate docker-compose.yaml file for **MLflow**, stored in the project repository.
* Started MLflow services:

docker-compose up -d

* Verified the running MLflow container:

docker ps -a

* Accessed the **MLflow UI** via **http://172.25.20.135:5000** (no authentication required).

**4. Setting Up DAGs for Airflow**

* Since Airflow is running inside a Docker container on a **remote server**, **VS Code** was used to connect via **SSH**.
* DAG files were placed in the default Airflow path:

/opt/airflow/dags/

* DAGs were written in Python, with the first one named **assignment.py**, defining tasks and execution workflows.

**5. Final Setup Status**

With **Airflow** and **MLflow** fully configured and running, the environment is now ready for:

* **Executing DAGs** in Airflow for workflow automation.
* **Tracking ML experiments** in MLflow.

This setup ensures efficient orchestration of ML pipelines while maintaining version control and experiment tracking.