Apache Airflow and Development Environment Setup Guide

Contents

---

1. Apache Airflow Setup on macOS with Docker

Prerequisites

- **Docker**: Ensure Docker is installed on your macOS machine.

Step-by-Step Guide

1. **Create Airflow Directory**
   - Open the terminal.
   - Create a directory for your Airflow setup:

     ```
     bash
     Copy code
     mkdir airflow-docker
     cd airflow-docker
     ```

2. **Create `docker-compose.yaml` File**
   - Inside the `airflow-docker` directory, create a `docker-compose.yaml` file:

     ```
     bash
     Copy code
     nano docker-compose.yaml
     ```

   - Copy and paste the following content:

     ```
     yaml
     Copy code
     services:
       postgres:
         image: postgres:13
         environment:
           POSTGRES_USER: airflow
           POSTGRES_PASSWORD: airflow
           POSTGRES_DB: airflow
     ```

```
    volumes:
      - postgres-db-volume:/var/lib/postgresql/data
    networks:
      - airflow-net

  webserver:
    image: apache/airflow:2.6.0
    depends_on:
      - postgres
    environment:
      AIRFLOW__CORE__EXECUTOR: LocalExecutor
      AIRFLOW__CORE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
      AIRFLOW__CORE__FERNET_KEY: YOUR_FERNET_KEY
      AIRFLOW__CORE__LOAD_EXAMPLES: 'false'
    volumes:
      - ./dags:/opt/airflow/dags
      - ./logs:/opt/airflow/logs
      - ./plugins:/opt/airflow/plugins
    ports:
      - "8080:8080"
    command: webserver
    networks:
      - airflow-net

  scheduler:
    image: apache/airflow:2.6.0
    depends_on:
      - webserver
      - postgres
    environment:
      AIRFLOW__CORE__EXECUTOR: LocalExecutor
      AIRFLOW__CORE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
      AIRFLOW__CORE__FERNET_KEY: YOUR_FERNET_KEY
      AIRFLOW__CORE__LOAD_EXAMPLES: 'false'
    volumes:
      - ./dags:/opt/airflow/dags
      - ./logs:/opt/airflow/logs
      - ./plugins:/opt/airflow/plugins
    command: scheduler
    networks:
      - airflow-net

networks:
  airflow-net:

volumes:
  postgres-db-volume:
```

3. **Generate a Fernet Key**
   - Airflow requires a Fernet key for encryption. Generate one using Python:

     ```
     bash
     Copy code
     ```

```
python3 -c "from cryptography.fernet import Fernet;
print(Fernet.generate_key().decode())"
```

   - Copy the generated key and replace `YOUR_FERNET_KEY` in `docker-compose.yaml`.

4. **Create Required Directories**
   - Inside `airflow-docker`, create the following directories:

     ```
     bash
     Copy code
     mkdir dags logs plugins
     ```

5. **Initialize Airflow Database**
   - Run the following command to initialize the Airflow database:

     ```
     bash
     Copy code
     docker-compose run webserver airflow db init
     ```

6. **Start Airflow Services**
   - To start the Airflow services:

     ```
     bash
     Copy code
     docker-compose up
     ```

   - Access the Airflow Web UI at http://localhost:8080 with default credentials:
     - **Username:** airflow
     - **Password:** airflow
   - To run Airflow in the background:

     ```
     bash
     Copy code
     docker-compose up -d
     ```

7. **Stopping the Services**
   - Stop the running Airflow services when done:

     ```
     bash
     Copy code
     docker-compose down
     ```

Additional Commands and Troubleshooting

- **Rebuild Containers**: If you update `docker-compose.yaml`, rebuild with:

```
bash
Copy code
docker-compose up --build
```

- **View Logs**: View logs for a specific service (e.g., webserver):

  ```
  bash
  Copy code
  docker-compose logs webserver
  ```

- **Port Conflict**: If port 8080 is in use, change the mapping:

  ```
  yaml
  Copy code
  ports:
    - "8081:8080"
  ```

---

2. Setting Up a Local Python Development Environment

Step-by-Step Guide

1. **Install Python and Virtualenv**
   - **Install Virtualenv** (if not already installed):

     ```
     bash
     Copy code
     pip install virtualenv
     ```

2. **Create and Activate a Virtual Environment**
   - Create a virtual environment:

     ```
     bash
     Copy code
     python3 -m venv mlops_env
     ```

   - Activate the virtual environment:
     - **macOS/Linux:** `source mlops_env/bin/activate`
     - **Windows:** `.\mlops_env\Scripts\activate`
3. **Install Required Packages**
   - Inside the virtual environment, install packages:

     ```
     bash
     Copy code
     pip install pandas scikit-learn google-cloud-storage
     ```

   - Create a `requirements.txt` file for project dependencies:

     ```
     bash
     Copy code
     pip freeze > requirements.txt
     ```

---

3. Project Collaboration with GitHub and VS Code

Step-by-Step Guide

1. **Install VS Code and Git**
    - **Install Visual Studio Code (VS Code)** from here.
    - **Install Git** for version control.
    - *(Optional)* Install **GitHub Desktop** for a GUI-based interface.
2. **Configure VS Code with GitHub**
    - **Install Extensions in VS Code:**
        - GitHub Pull Requests and Issues
        - Live Share (for real-time collaboration)
        - Language-specific extensions, e.g., Python.
    - **Clone a GitHub Repository in VS Code:**

1. Open VS Code and go to **Command Palette** (Cmd+Shift+P on macOS).
2. Select **Git: Clone** and enter the GitHub repository URL.
3. Choose a folder to save the repository locally.

3. **Basic Git Commands for Collaboration**
    - **Create a Branch:** Create a new branch for each feature/task:

```
bash
Copy code
git checkout -b feature-branch
```

    - **Commit Changes:** Save changes with a meaningful message:

```
bash
Copy code
git add .
git commit -m "Your commit message"
```

    - **Push Changes:** Push local changes to the GitHub repository:

```
bash
Copy code
git push origin feature-branch
```

    - **Pull Changes:** Pull latest changes from the main branch:

```
bash
Copy code
git pull origin main
```

    - **Create a Pull Request:** After pushing changes, go to GitHub and create a Pull Request to allow teammates to review changes before merging.
4. **Managing Access and Permissions in GitHub**

- **Add Collaborators**: Go to **Settings > Collaborators & Teams** to add members.
- **Branch Protection Rules**: Set up branch protection to require PR review before merging. Go to **Settings > Branches > Add Rule**.

---

4. Integrating with Google Cloud Platform (GCP)

Step-by-Step Guide

1. **Create Google Cloud Service Account**
   - Go to **IAM & Admin > Service Accounts** in the GCP Console.
   - **Create a New Service Account:**
     - Name it (e.g., `mlops-service-account`).
     - Assign a role (e.g., Owner for GCS access).
     - **Generate a JSON key** and download it to your project folder.
2. **Set Environment Variable for Service Account Key**
   - Set the environment variable to use the downloaded JSON key:
     - **macOS/Linux:**

       ```
       bash
       Copy code
       export GOOGLE_APPLICATION_CREDENTIALS="/path/to/your-
       service-account-key.json"
       ```

     - **Windows:**

       ```
       cmd
       Copy code
       set GOOGLE_APPLICATION_CREDENTIALS="C:\path\to\your-
       service-account-key.json"
       ```

3. **Install Google Cloud SDK**
   - **Install the Google Cloud SDK on macOS:**

     ```
     bash
     Copy code
     curl https://sdk.cloud.google.com | bash
     exec -l $SHELL
     ```

   - **Initialize and Authenticate SDK:**

     ```
     bash
     Copy code
     gcloud init
     gcloud auth application-default login
     ```

This setup guide provides an end-to-end configuration for setting up Apache Airflow, local development, collaboration via GitHub, and integration with Google Cloud Platform, creating a robust foundation for managing data workflows and collaborative development.