Project Documentation: DuckDB, DBT, and Dagster Integration with Docker

# Table of Contents:

1. Introduction

2. Prerequisites

3. Step 1: Setting up DuckDB

4. Step 2: Integrating DBT with DuckDB

5. Step 3: Orchestrating with Dagster

6. Step 4: Dockerizing the Project

7. Step 5: Scheduling in Dagster GUI

8. Step 6: Logging in Dagster GUI

9. Conclusion

# Introduction

This project demonstrates the integration of DuckDB as the database, DBT (Data Build Tool) for transformations, and Dagster for orchestration. The entire process is Dockerized, making it easier to manage and deploy the pipeline. Additionally, the project allows scheduling and logging through the Dagster GUI for better monitoring and execution tracking.

# Prerequisites

Before starting, ensure that you have the following installed:  
  
- Docker  
- Dagster  
- DBT  
- DuckDB  
- LocalStack (for S3 integration, if necessary)  
- Python (for DBT, Dagster, and S3 interaction)  
  
You will also need access to the project's repository and the docker-compose file for orchestration.

# Step 1: Setting up DuckDB

1. Install DuckDB: Install DuckDB in your environment if it isn't already present.  
2. Create a DuckDB instance: In your project directory), set up the DuckDB file, ensuring it points to the database file (e.g., dev.duckdb).

# Step 2: Integrating DBT with DuckDB

1. Install DBT: Set up DBT in the same environment using pip install dbt.  
2. DBT Setup:  
- Create a profiles.yml file to configure your DuckDB connection.  
- Define your models, seeds, and other DBT configurations.  
3. Run DBT Models: Using DBT commands, test the integration by running your models:  
```bash  
dbt run  
```

# Step 3: Orchestrating with Dagster

1. Install Dagster: Install the required Dagster packages:  
   ```bash  
   pip install dagster dagit  
   ```  
   2. Create a Dagster Pipeline:  
   - Define your solids (functions that represent DBT tasks) and pipeline in a Python file.  
   - Set up the integration to execute DBT commands within the Dagster pipeline.  
   3. Test the Pipeline:  
   - Run the Dagster pipeline manually using:  
   ```bash  
   dagit  
   ```

4- Also run dagster-sbt

# Step 4: Dockerizing the Project

1. Dockerfile:  
- Ensure that your Docker image contains all dependencies (DBT, DuckDB, Dagster, etc.).  
- Create a Dockerfile similar to this:  
```dockerfile  
# Use an official Python image as the base

FROM python:3.10-slim

# Set environment variables

ENV PYTHONUNBUFFERED=1 \

    DBT\_PROFILES\_DIR=/app/dbtdagster \

    DAGSTER\_HOME=/app/dagster\_home \

    DBT\_PROJECT\_DIR=/app/duck\_db\_project

# Install Python dependencies

COPY requirements.txt /tmp/requirements.txt

RUN pip install --no-cache-dir -r /tmp/requirements.txt && rm /tmp/requirements.txt

# Set up the application directory

WORKDIR /app

# Create required directories

RUN mkdir -p /app/dbtdagster /app/dagster\_home /app/duck\_db\_project

# Copy project files

COPY . /app

# Set the default command to start Dagster's development server

CMD ["dagster", "dev", "--host", "0.0.0.0", "--port", "3002"]

**To Build:**

docker build -t my\_duckdb\_dbt\_dagster\_project .

**To RUN**:  
docker run -v "E:/Tasks/DuckDb/locally/duck\_db\_project:/app/dbt\_project" -e DBT\_PROJECT\_DIR="/app/dbt\_project" -p 3002:3002 my\_duckdb\_dbt\_dagster\_project

# Step 5: Scheduling in Dagster GUI

1. Access Dagster GUI: Once your Dagster container is running, open the Dagster GUI in your browser at http://localhost:3000.  
2. Schedule a Job:  
- Go to the 'Schedules' tab.  
- Click 'Create Schedule'.  
- Choose the pipeline you want to schedule, define the cron expression (e.g., daily, hourly), and save the schedule.  
3. Trigger Schedule: The job will automatically run according to the defined schedule.

# Step 6: Logging in Dagster GUI

1. Log Output:  
- In the Dagster GUI, navigate to the 'Logs' tab to view logs for your pipelines and individual solids.  
- You can see execution details, including success or failure logs for each run.  
2. Log Configuration:  
- Customize logging within the Dagster pipeline code if necessary (e.g., using the logging library in Python).

# Conclusion

This project provides a complete solution for data orchestration with DuckDB, DBT, and Dagster, all managed in a Dockerized environment. By using the Dagster GUI, users can easily schedule and monitor pipeline executions and log details, ensuring efficient pipeline management.