**Airflow**

**Name: Priyeshwar**

**Mail:** [**priyesh2664@gmail.com**](mailto:priyesh2664@gmail.com)

**What is Airflow?**

Apache Airflow is an open-source workflow orchestration tool used by data engineers to programmatically author, schedule, and monitor workflows (pipelines). Workflows are defined in Python code and represented as Directed Acyclic Graphs (DAGs).

**Key Features of Apache Airflow**

**Workflow Management**

* **DAGs (Directed Acyclic Graphs):** Workflows are represented as DAGs ensuring tasks run in the correct sequence without cycles.
* **Task Dependencies:** Define upstream and downstream relationships clearly.

**Scheduling & Execution**

* **Scheduler & Executors:** Tasks are scheduled and distributed across executors (Local, Celery, Kubernetes, etc.).
* **Flexible Scheduling:** Cron-like expressions or time intervals; supports backfilling.

**Monitoring & Management**

* **Web UI:** Visual interface for monitoring DAG runs, logs, and task states.
* **Logging:** Centralized logs stored locally or in external systems (S3, GCS, Elasticsearch).
* **Alerts & Retries:** Automatic retries, failure callbacks, and alerts on errors.

**Scalability & Integration**

* **Horizontal Scalability:** Distribute workloads across multiple workers.
* **Integrations:** Works with AWS, GCP, Azure, Spark, Hive, Presto, Kubernetes, and many other tools.
* **Event-based Triggers:** Sensors for external jobs, file arrivals, and system events.

**Security**

* **Role-Based Access Control (RBAC):** Manage permissions by roles.
* **Authentication:** Integration with LDAP, OAuth, and other authentication systems.

**Steps to Build and Run an Airflow Pipeline**

**Step 1: Install and Initialize Airflow**

* Install Airflow (e.g., with pip install apache-airflow).
* Initialize the Airflow database:
* airflow db init

**Step 2: Create the DAG File**

* Write your DAG definition in a .py file.
* Save it inside the Airflow dags/ folder (default: ~/airflow/dags/).

**Example DAG file: ~/airflow/dags/process\_employees.py**

import datetime

import pendulum

import os

import requests

from airflow.decorators import task, dag

from airflow.providers.postgres.hooks.postgres import PostgresHook

from airflow.providers.postgres.operators.postgres import PostgresOperator

@dag(

dag\_id="process\_employees",

schedule="0 0 \* \* \*",

start\_date=pendulum.datetime(2021, 1, 1, tz="UTC"),

catchup=False,

dagrun\_timeout=datetime.timedelta(minutes=60),

)

def ProcessEmployees():

create\_employees\_table = PostgresOperator(

task\_id="create\_employees\_table",

postgres\_conn\_id="tutorial\_pg\_conn",

sql="""

CREATE TABLE IF NOT EXISTS employees (

"Serial Number" NUMERIC PRIMARY KEY,

"Company Name" TEXT,

"Employee Markme" TEXT,

"Description" TEXT,

"Leave" INTEGER

);""",

)

create\_employees\_temp\_table = PostgresOperator(

task\_id="create\_employees\_temp\_table",

postgres\_conn\_id="tutorial\_pg\_conn",

sql="""

DROP TABLE IF EXISTS employees\_temp;

CREATE TABLE employees\_temp (

"Serial Number" NUMERIC PRIMARY KEY,

"Company Name" TEXT,

"Employee Markme" TEXT,

"Description" TEXT,

"Leave" INTEGER

);""",

)

**Step 3: Create the task**

@task

def get\_data():

data\_path = "/opt/airflow/dags/files/employees.csv"

os.makedirs(os.path.dirname(data\_path), exist\_ok=True)

url = "https://raw.githubusercontent.com/apache/airflow/main/airflow-core/docs/tutorial/pipeline\_example.csv"

response = requests.get(url)

with open(data\_path, "w") as file:

file.write(response.text)

postgres\_hook = PostgresHook(postgres\_conn\_id="tutorial\_pg\_conn")

conn = postgres\_hook.get\_conn()

cur = conn.cursor()

with open(data\_path, "r") as file:

cur.copy\_expert(

"COPY employees\_temp FROM STDIN WITH CSV HEADER DELIMITER AS ',' QUOTE '"'",

file,

)

conn.commit()

@task

def merge\_data():

query = """

INSERT INTO employees

SELECT \*

FROM (

SELECT DISTINCT \*

FROM employees\_temp

) t

ON CONFLICT ("Serial Number") DO UPDATE

SET

"Employee Markme" = excluded."Employee Markme",

"Description" = excluded."Description",

"Leave" = excluded."Leave";

"""

try:

postgres\_hook = PostgresHook(postgres\_conn\_id="tutorial\_pg\_conn")

conn = postgres\_hook.get\_conn()

cur = conn.cursor()

cur.execute(query)

conn.commit()

return 0

except Exception as e:

return 1

[create\_employees\_table, create\_employees\_temp\_table] >> get\_data() >> merge\_data()

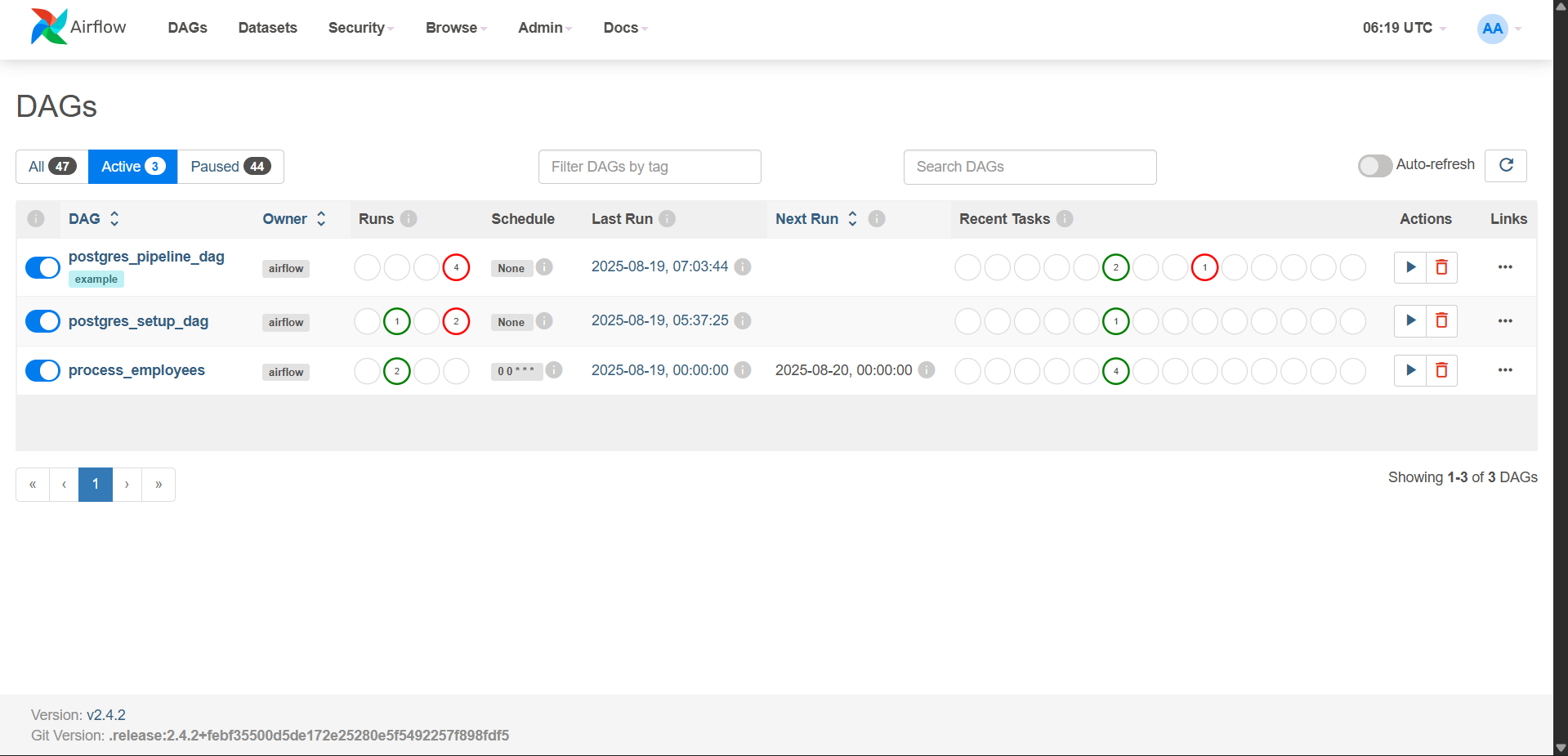
dag = ProcessEmployees()

**Step 4: Start Airflow Services**

* Start the scheduler:
* airflow scheduler
* Start the webserver:
* airflow webserver -p 8080

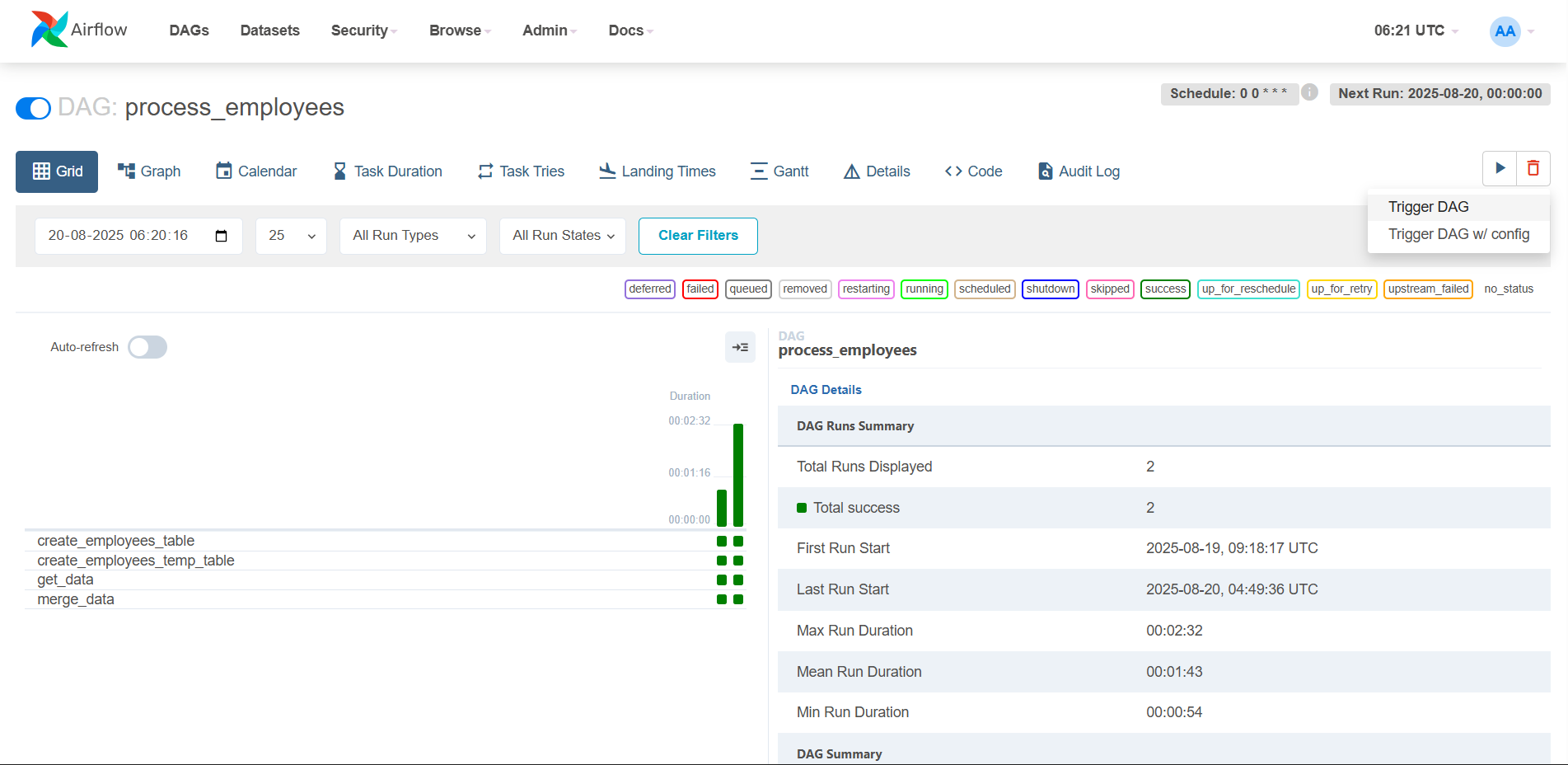
**Step 5: Access the Airflow UI**

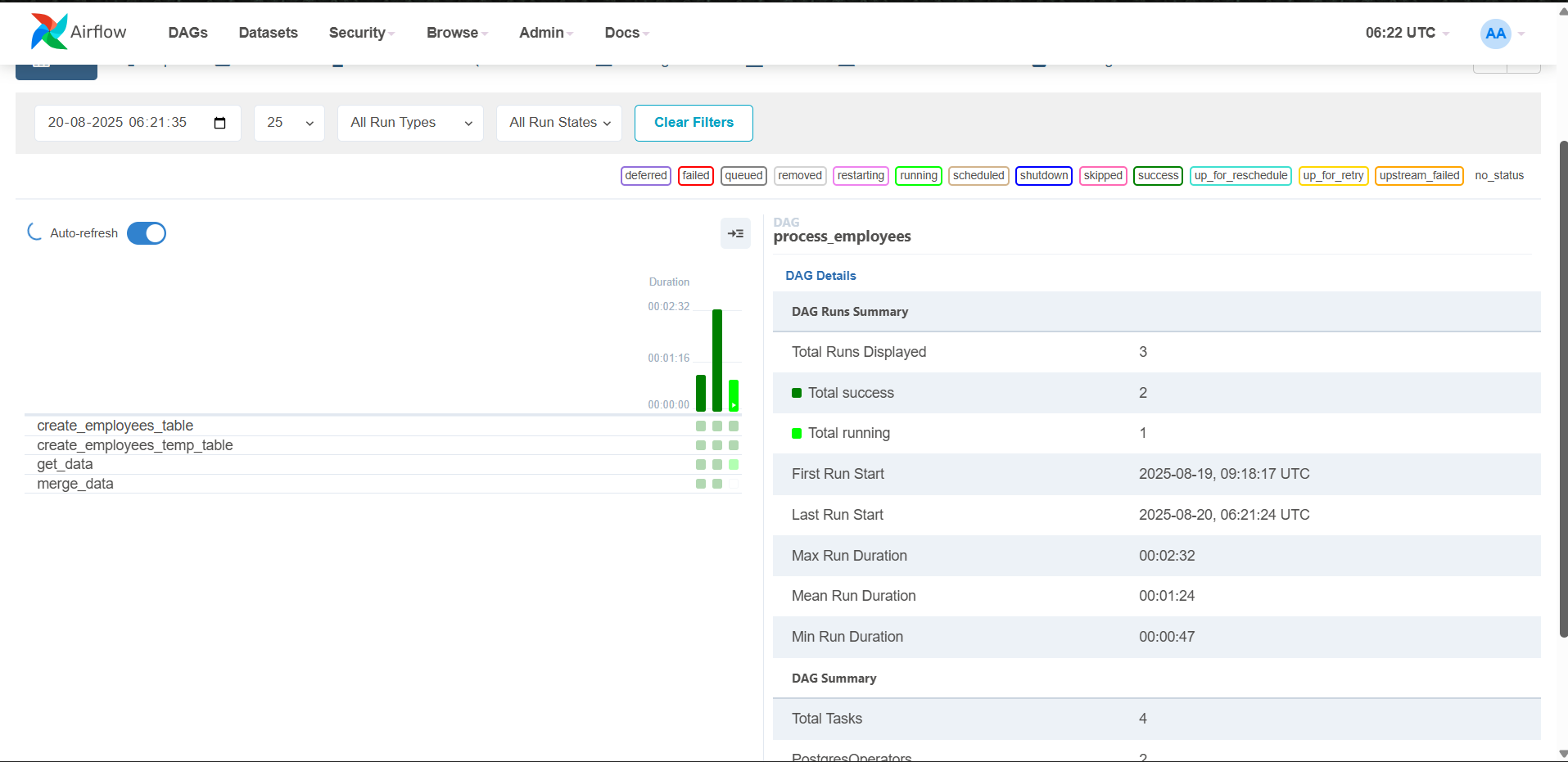
* Open your browser and go to: [http://localhost:8080](http://localhost:8080/).
* Log in with the user created earlier.



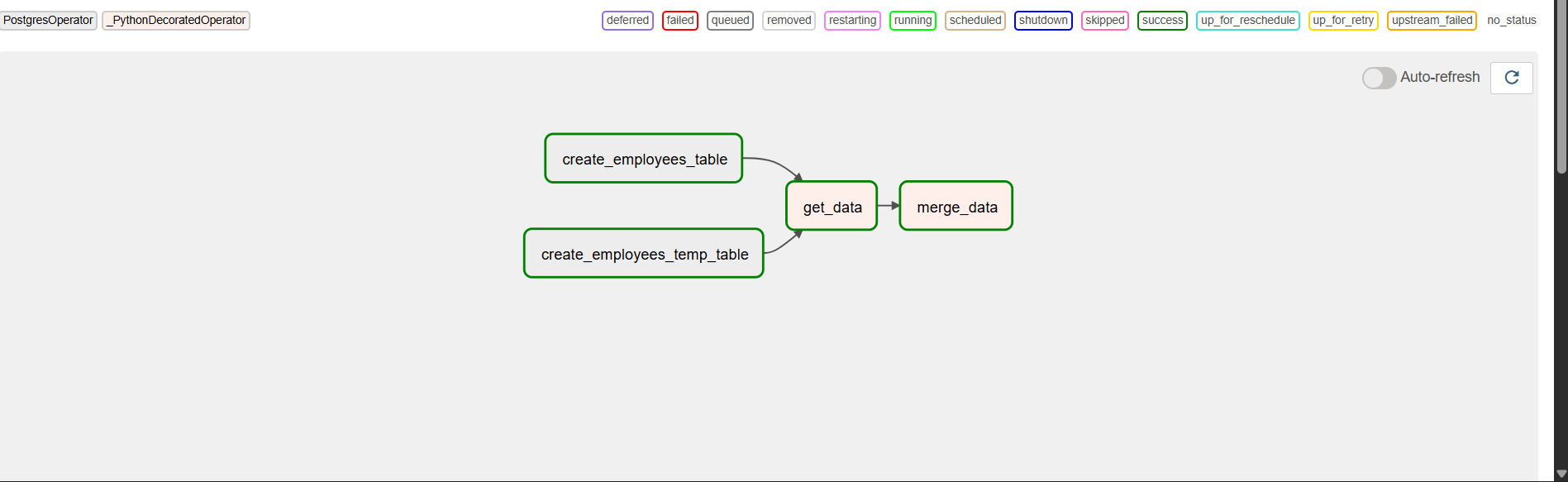
**Step 6: Trigger and Monitor the DAG**

* Enable the DAG in the UI.
* Trigger a run manually or wait for the schedule.

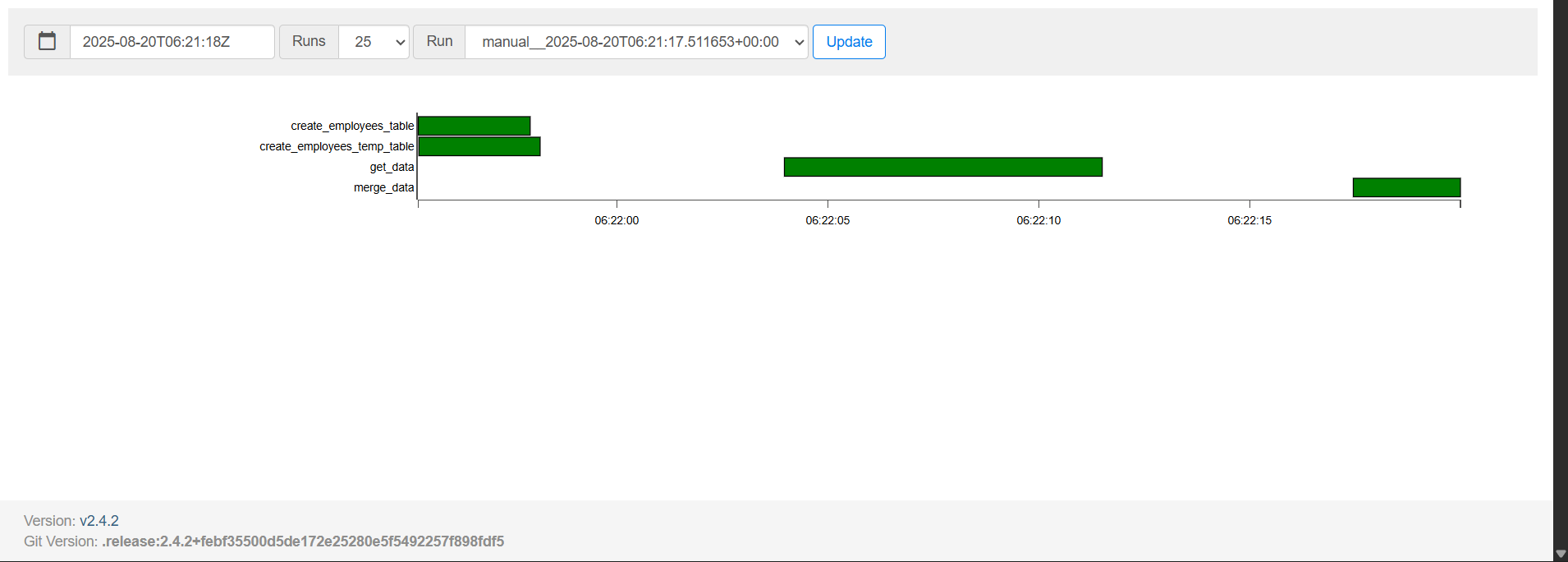




* **Graph View:** Visual DAG representation showing task dependencies.



* **Gantt View:** Duration and overlap of tasks.



* View logs and retries for each task.