BIG DATA

Студент группы МПИ-22-2-1
Королев Никита

Отчет по лабораторной работе № 3

docker-compose.yaml × new_dag.py data24.csv

```yaml
---
version: '3.8'
x-airflow-common:
  &airflow-common
  # In order to add custom dependencies or upgrade provider packages you can use your extended image.
  # Comment the image line, place your Dockerfile in the directory where you placed the docker-compose.yaml
  # and uncomment the "build" line below, Then run `docker-compose build` to build the images.
  image: ${AIRFLOW_IMAGE_NAME:-apache/airflow:2.6.0}
  # build: .
  environment:
    &airflow-common-env
    AIRFLOW__CORE__EXECUTOR: CeleryExecutor
    AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow
    # For backward compatibility, with Airflow <2.3
    AIRFLOW__CORE__SQL_ALCHEMY_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow
    AIRFLOW__CELERY__RESULT_BACKEND: db+postgresql://airflow:airflow@postgres/airflow
    AIRFLOW__CELERY__BROKER_URL: redis://:@redis:6379/0
    AIRFLOW__CORE__FERNET_KEY: ''
    AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION: 'true'
    AIRFLOW__CORE__LOAD_EXAMPLES: 'true'
    AIRFLOW__API__AUTH_BACKENDS: 'airflow.api.auth.backend.basic_auth,airflow.api.auth.backend.session'
    # yamllint disable rule:line-length
    # Use simple http server on scheduler for health checks
    # See https://airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/logging-monitoring/check-h
    # yamllint enable rule:line-length
    AIRFLOW__SCHEDULER__ENABLE_HEALTH_CHECK: 'true'
    # WARNING: Use _PIP_ADDITIONAL_REQUIREMENTS option ONLY for a quick checks
    # for other purpose (development, test and especially production usage) build/extend Airflow image.
    _PIP_ADDITIONAL_REQUIREMENTS: ${_PIP_ADDITIONAL_REQUIREMENTS:-}
  volumes:
    - ${AIRFLOW_PROJ_DIR:-.}/dags:/opt/airflow/dags
    - ${AIRFLOW_PROJ_DIR:-.}/logs:/opt/airflow/logs
    - ${AIRFLOW_PROJ_DIR:-.}/plugins:/opt/airflow/plugins
  user: "${AIRFLOW_UID:-50000}:0"
  depends_on:
    &airflow-common-depends-on
    redis:
      condition: service_healthy
    postgres:
      condition: service_healthy
```

Рис. 1 – конфигурация файла docker-compose.yml (взята дефолтная конфигурация с официального сайта Apache Airflow)

```
/users/iratewarrior/misis/big_data/lab33 ▷ docker ps -a

CONTAINER ID   IMAGE                  COMMAND                  CREATED             STATUS                       PORTS                    NAMES
49e93b43056a   apache/airflow:2.6.0   "/usr/bin/dumb-init …"   About an hour ago   Up About an hour (healthy)   8080/tcp                 lab33-airflow-scheduler-1
a21e994433cb   apache/airflow:2.6.0   "/usr/bin/dumb-init …"   About an hour ago   Up About an hour (healthy)   0.0.0.0:8080->8080/tcp   lab33-airflow-webserver-1
4e6e4035da7f   apache/airflow:2.6.0   "/usr/bin/dumb-init …"   About an hour ago   Up About an hour (healthy)   8080/tcp                 lab33-airflow-worker-1
2472ae9868c3   apache/airflow:2.6.0   "/usr/bin/dumb-init …"   About an hour ago   Up About an hour (healthy)   8080/tcp                 lab33-airflow-triggerer-1
bd849758d43a   apache/airflow:2.6.0   "/bin/bash -c 'funct…"   About an hour ago   Exited (0) About an hour ago                          lab33-airflow-init-1
dcacedb39324   postgres:13            "docker-entrypoint.s…"   About an hour ago   Up About an hour (healthy)   0.0.0.0:5433->5432/tcp   lab33-postgres-1
d74b90cb868e   redis:latest           "docker-entrypoint.s…"   About an hour ago   Up About an hour (healthy)   6379/tcp                 lab33-redis-1
/users/iratewarrior/misis/big_data/lab33 ▷
```

Рис. 2 – после поднятия docker compose список работающих контейнеров

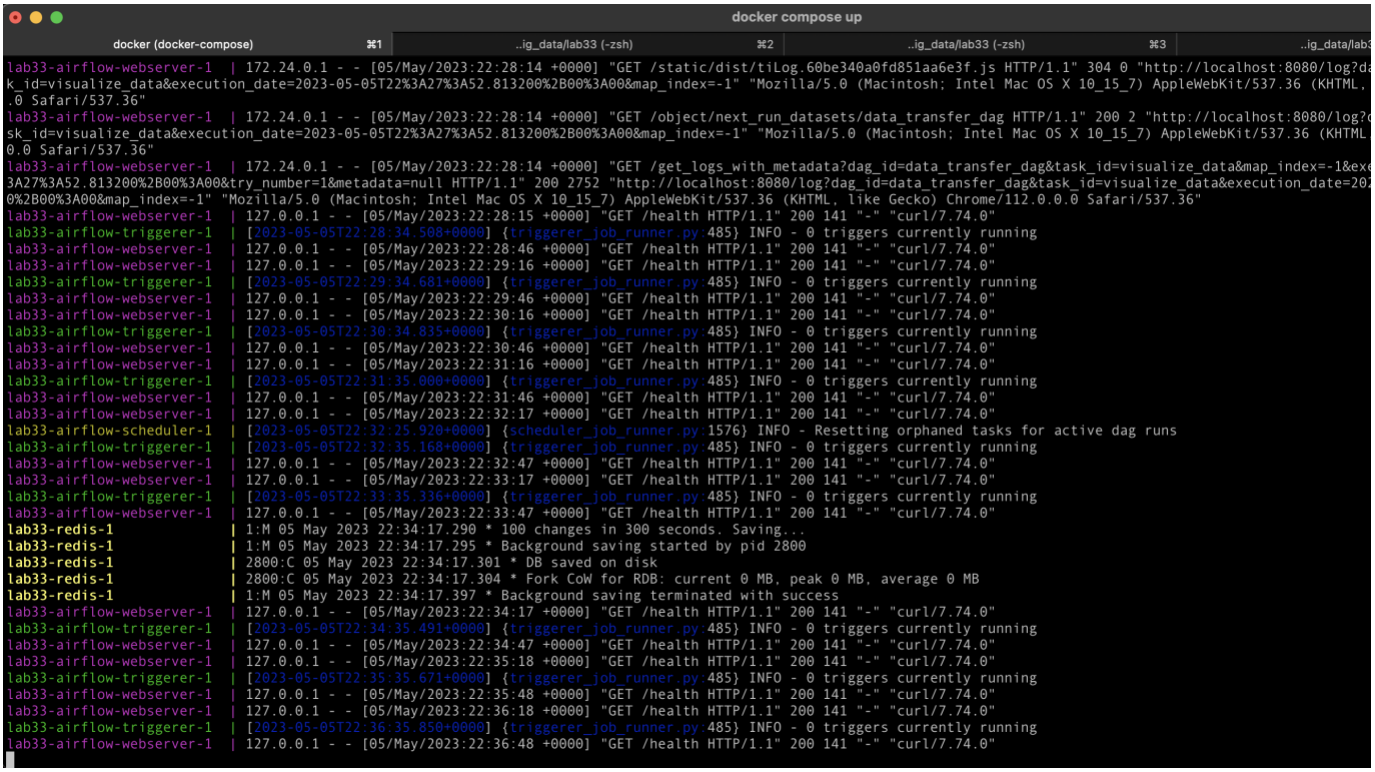

Рис. 3 – работа Airflow в консоли

docker-compose.yaml | new_dag.py | data24.csv

```python
from airflow.models import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime
import pandas as pd
import psycopg2

# Функция для чтения данных из CSV файла
def read_csv_data():
    df = pd.read_csv('/opt/airflow/dags/data24.csv')
    return df

# Функция для записи данных в базу данных Postgres
def write_to_postgres():
    conn = psycopg2.connect(
        host='172.24.0.3',
        port='5432',
        database='airflow',
        user='airflow',
        password='airflow'
    )
    cursor = conn.cursor()

    # Прочитать данные из CSV файла
    df = read_csv_data()

    for i in df.values:
        cursor.execute("INSERT INTO public.lab34 (id, name, surname, age, height, weight, run_100m) VALUES (%s, %s, %s, %s, %s, %s, %s);",
                       [i[0], i[1], i[2], i[3], i[4], i[5], i[6]])

    conn.commit()
    cursor.close()
    conn.close()
```

Рис. 4.1 – конфигурация DAG для Apache Airflow

docker-compose.yaml | new_dag.py | data24.csv

```
33
34  # Функция для визуализации данных в консоли
    1 usage
35  def visualize_data():
36      conn = psycopg2.connect(
37          host='172.24.0.3',
38          port='5432',
39          database='airflow',
40          user='airflow',
41          password='airflow'
42      )
43      # Выберем топ-3 бегунов по результатам бега 100 метров
44      query = """
45          SELECT name, surname, run_100m
46          FROM public.lab34
47          ORDER BY run_100m
48          LIMIT 3;
49          """
50      # Заполнение данных в DataFrame
51      df = pd.read_sql(query, conn)
52
53      conn.close()
54
55      # Выводим результаты в консоль
56      print("Top 3 runners by 100m run:")
57      print(df)
58
59  # Создание DAG
60  dag = DAG(
61      'data_transfer_dag',
62      description='Перенос данных из CSV в базу данных Postgres',
63      schedule_interval=None,
64      start_date=datetime(2023, 5, 5),
65      catchup=False
66  )
```

Рис. 4.2 – конфигурация DAG для Apache Airflow

docker-compose.yaml | new_dag.py | data24.csv

```
58
59  # Создание DAG
60  dag = DAG(
61      'data_transfer_dag',
62      description='Перенос данных из CSV в базу данных Postgres',
63      schedule_interval=None,
64      start_date=datetime(2023, 5, 5),
65      catchup=False
66  )
67
68  # Оператор для чтения данных из CSV файла
69  read_csv_operator = PythonOperator(
70      task_id='read_csv_data',
71      python_callable=read_csv_data,
72      dag=dag
73  )
74
75  # Оператор для записи данных в базу данных Postgres
76  write_to_postgres_operator = PythonOperator(
77      task_id='write_to_postgres',
78      python_callable=write_to_postgres,
79      dag=dag
80  )
81
82  visualize_data = PythonOperator(
83      task_id='visualize_data',
84      python_callable=visualize_data,
85      dag=dag
86  )
87
88
89
90  # Установка порядка выполнения задач
91  read_csv_operator >> write_to_postgres_operator >> visualize_data
92
```

Рис. 4.3 – конфигурация DAG для Apache Airflow

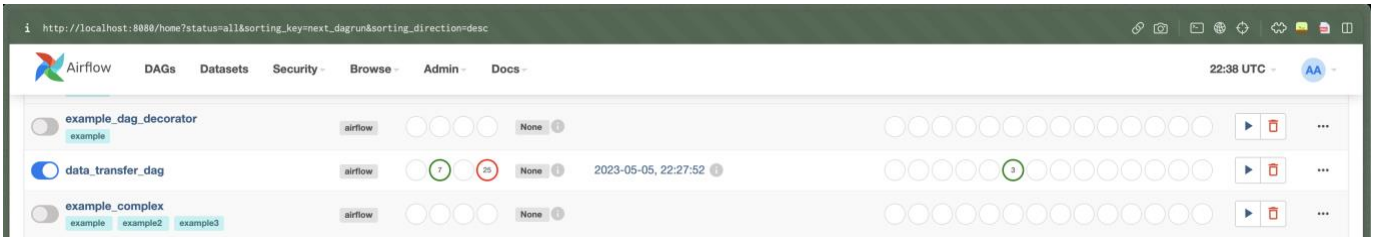

Рис. 5 – отображение DAG в UI Apache Airflow

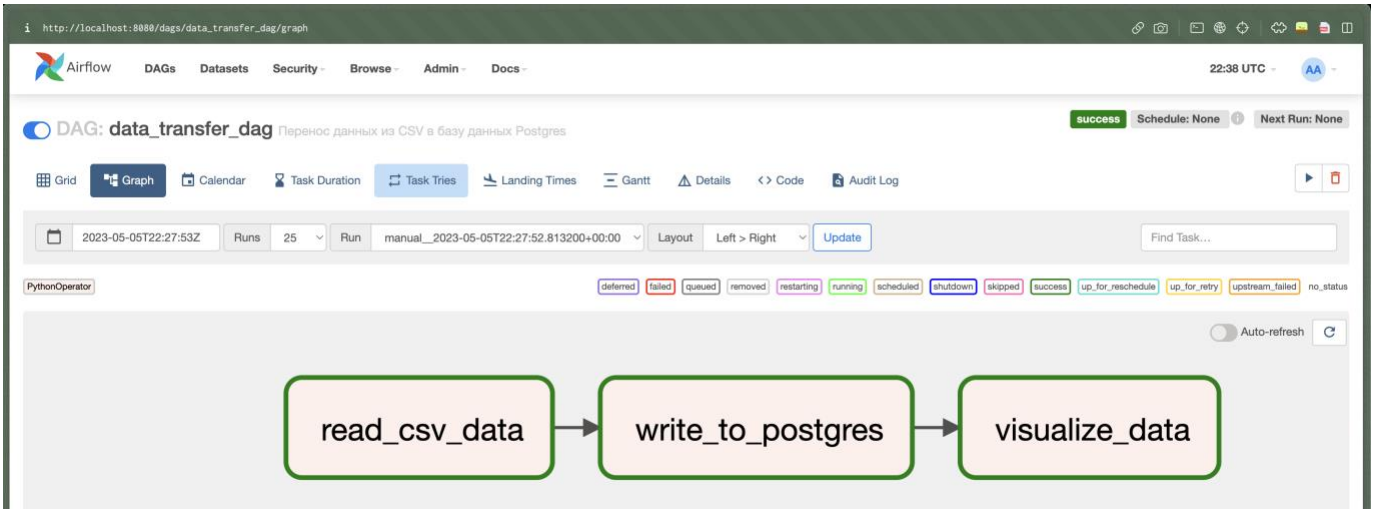

Рис. 6 – отображение графа DAG в UI Apache Airflow

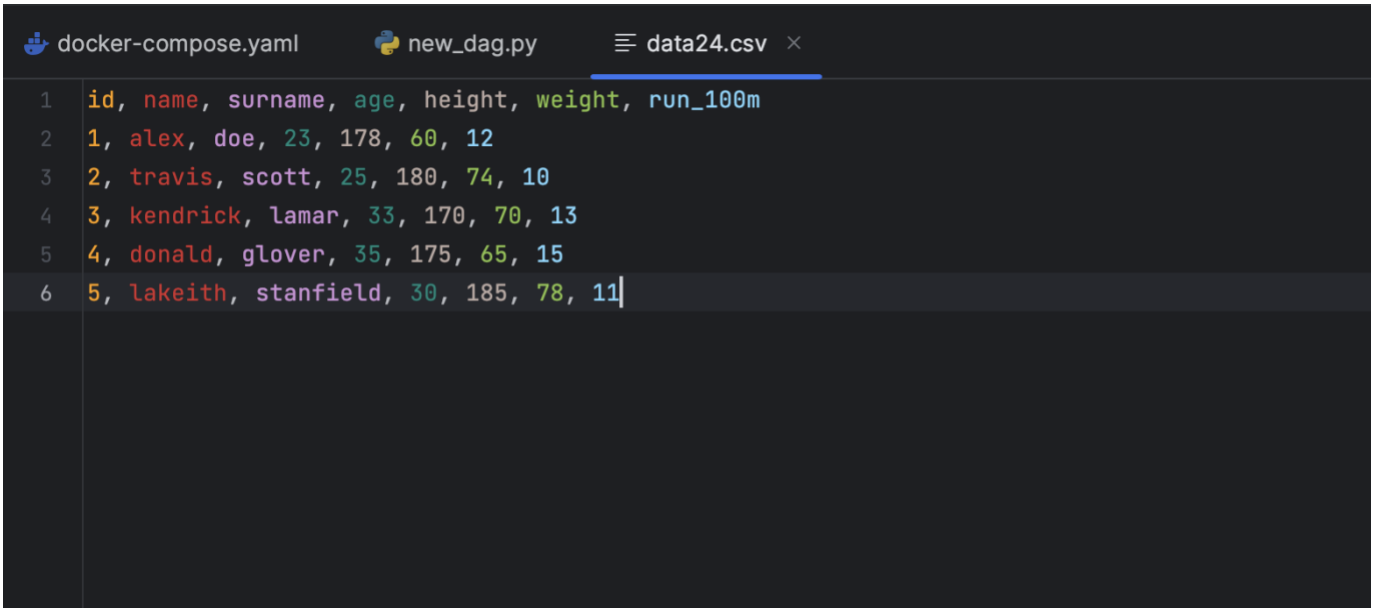

Рис. 7 – содержание CSV файла

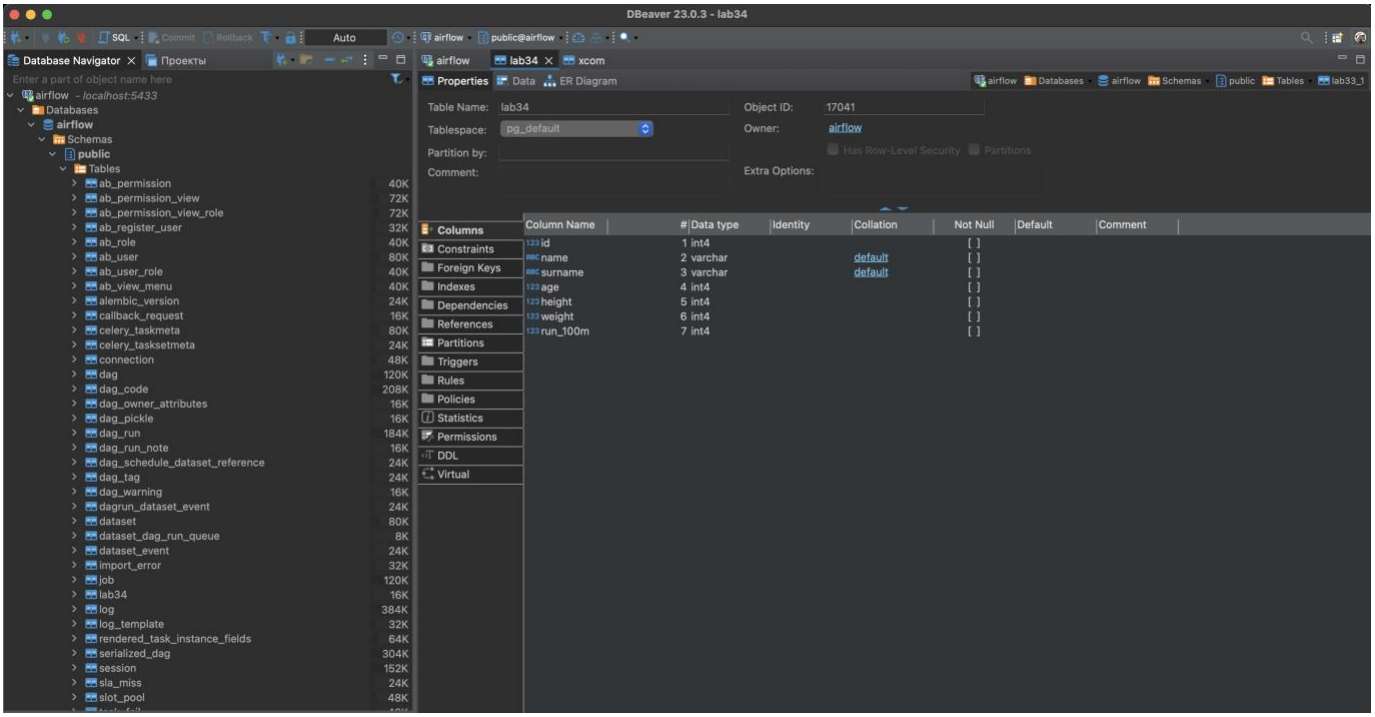

Рис. 8 – создание таблицы в БД Postgres в ПО DBeaver

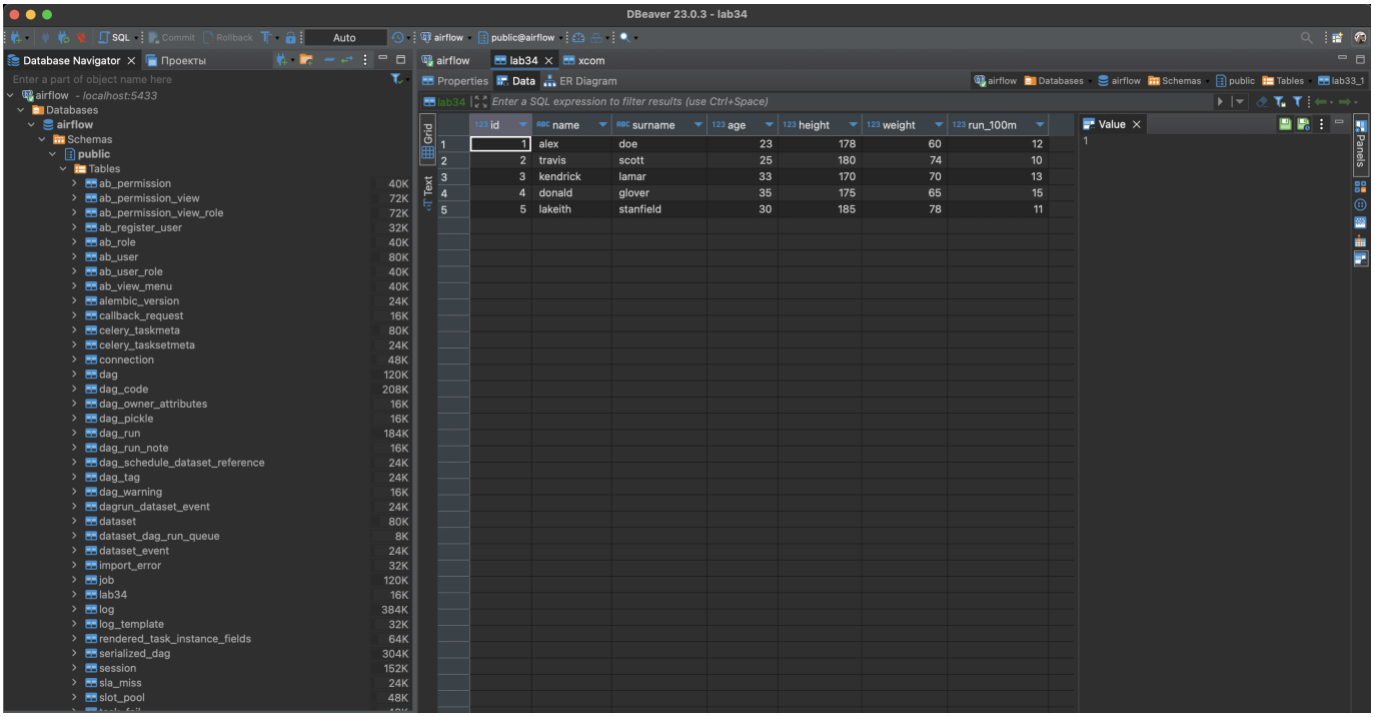

Рис. 9 –таблицы в БД Postgres заполнена данными из CSV файла после отработки DAG

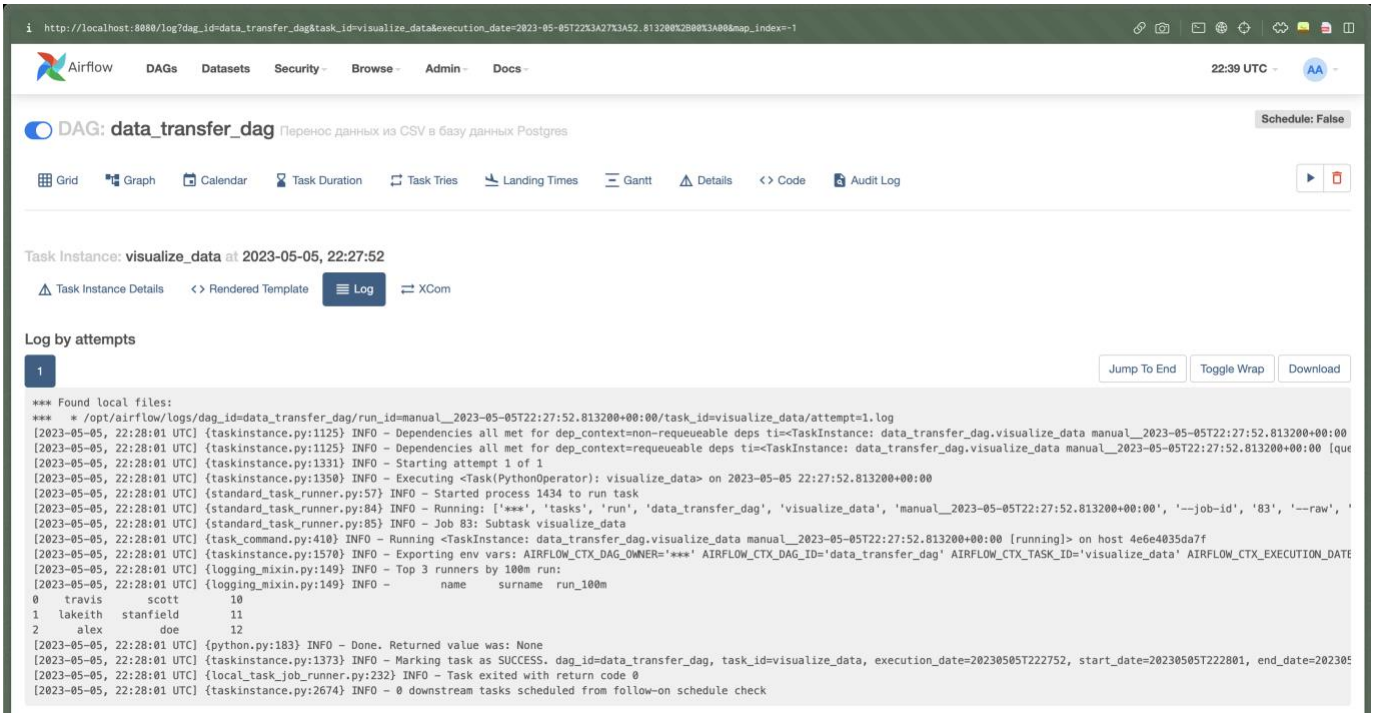

Рис. 10 – логи DAG