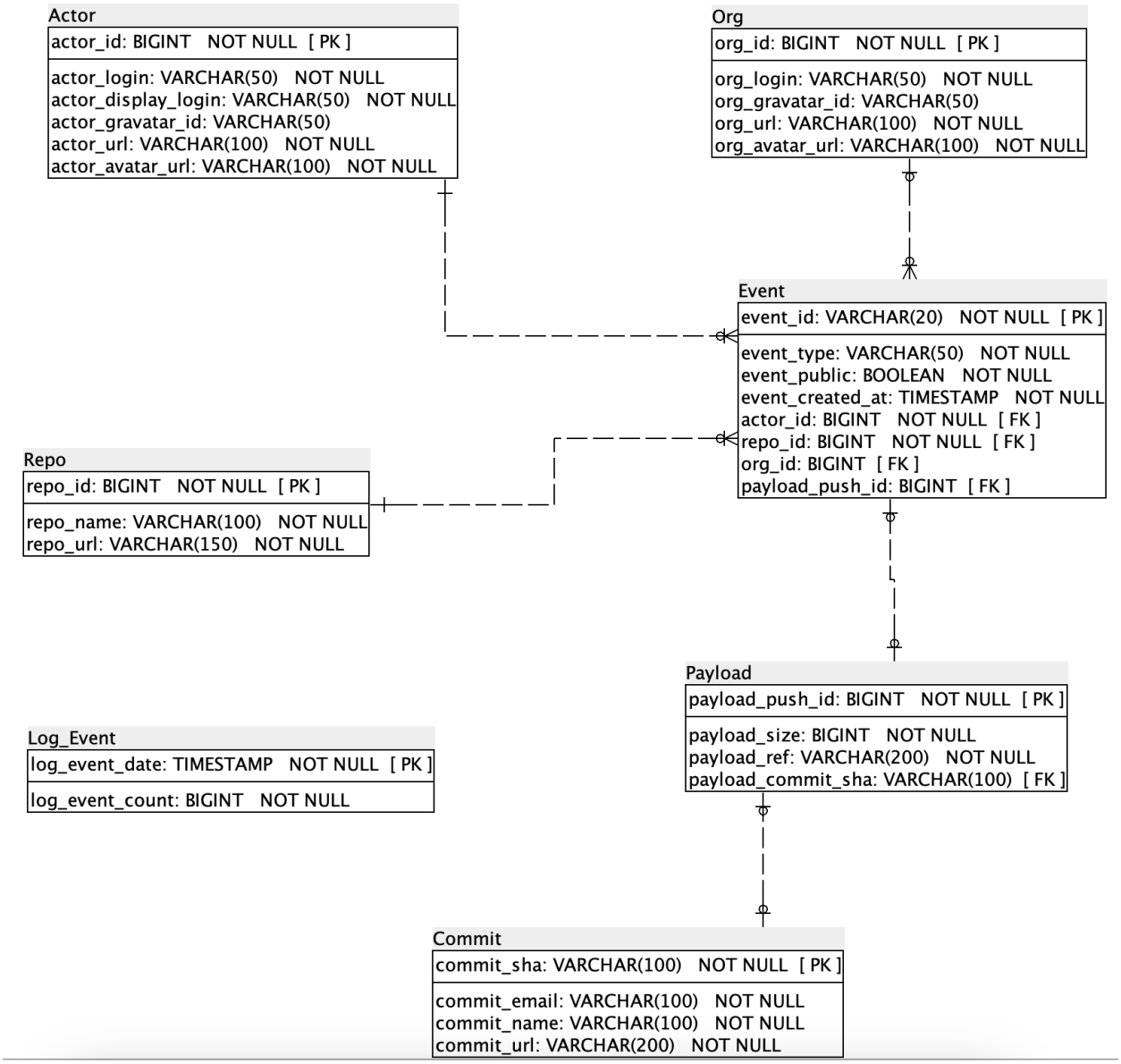
Lab5 – Project Airflow

Data model



** Table **Log_Event** is used to collect the log for loading of table **Event**.

DS525 - Chin Lertvipada - 64199130039

Project implementation instruction

1. Reach the project repository '/swu-ds525/05-creating-and-scheduling-data-pipelines'

   : *$ cd 05-creating-and-scheduling-data-pipelines*

```
(base) JC@Napchins-MacBook-Air swu-ds525 % cd 05-creating-and-scheduling-data-pipelines
(base) JC@Napchins-MacBook-Air 05-creating-and-scheduling-data-pipelines %
```

2. Setup Environment (*** ONLY FOR LINUX on 1ST TIME SETUP ***)

   : $ mkdir -p ./dags ./logs ./plugins

   : $ echo -e "AIRFLOW_UID=$(id -u)" > .env

```
mkdir -p ./dags ./logs ./plugins
echo -e "AIRFLOW_UID=$(id -u)" > .env
```

3. Prepare the environment workspace thru 'docker-compose.yml'

   : $ docker-compose up

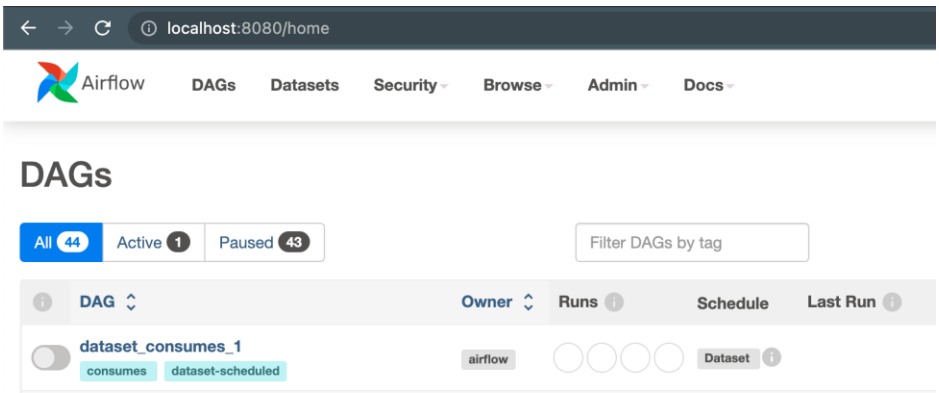
```
docker-compose up
```

   These services will be up

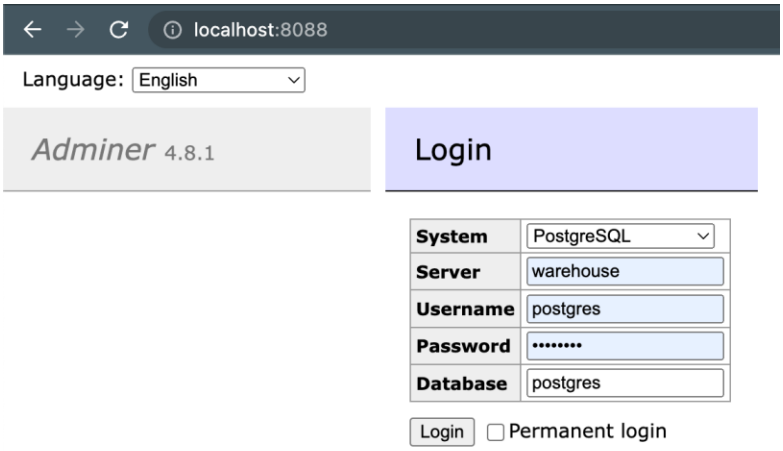
   - Apache Airflow : for task scheduling
   - Postgres : for database
   - Adminer : for Postgres access thru web service

4. Open Airflow and Postgres thru web service

   Airflow :



Postgres : password = ‘postgres’

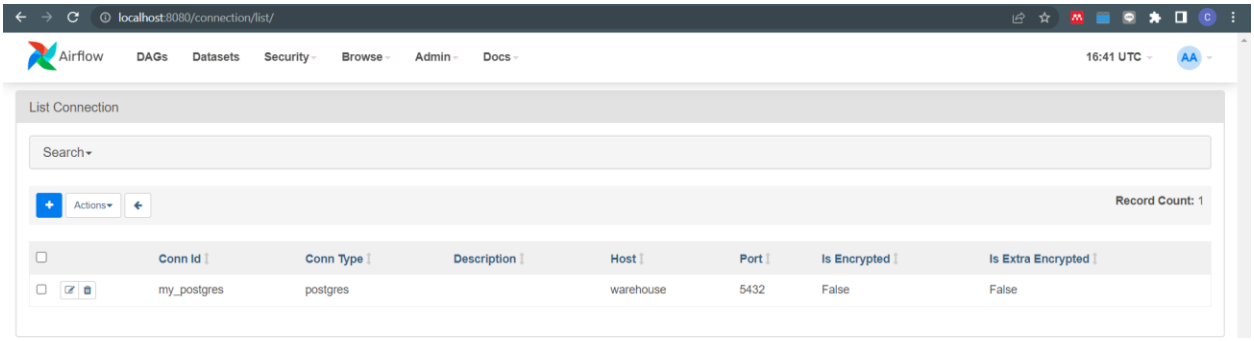


** below are Postgres connection information, setup in **docker-compose.yaml**

```yaml
warehouse:
  image: postgres:13
  environment:
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: postgres
    POSTGRES_DB: postgres
  volumes:
    - warehouse-db-volume:/var/lib/postgresql/data
```

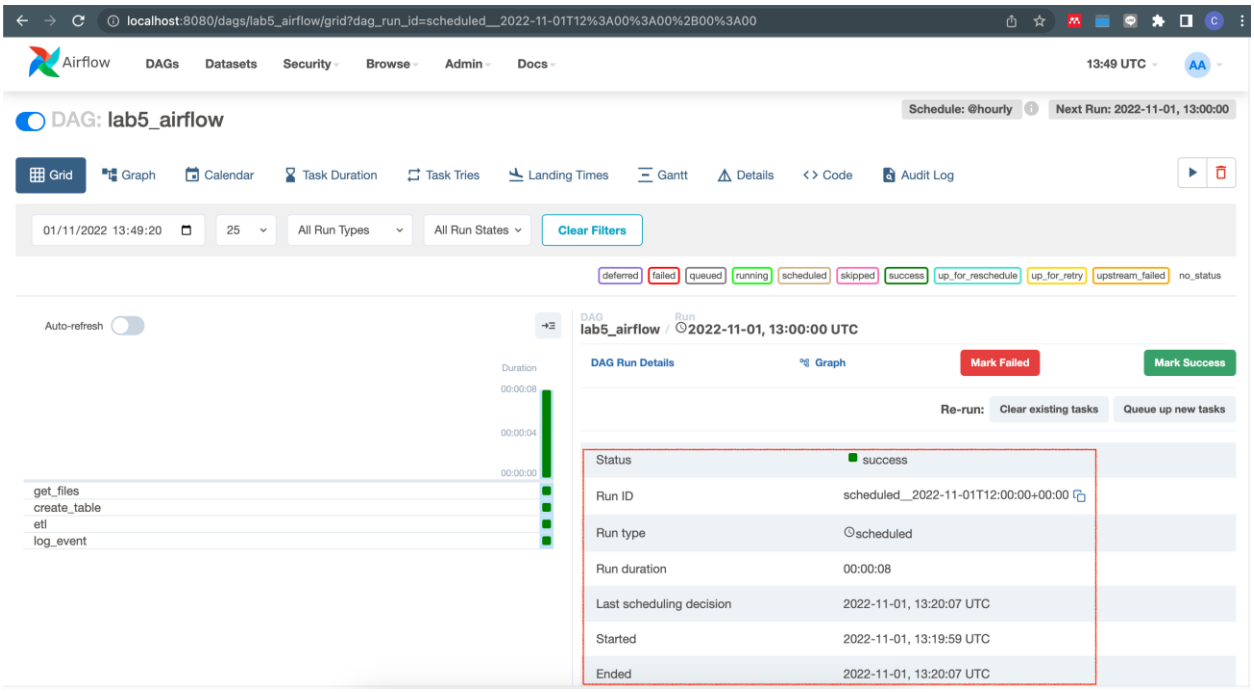
5. Setup Database connection for Airflow to access Postgres

Information as below :



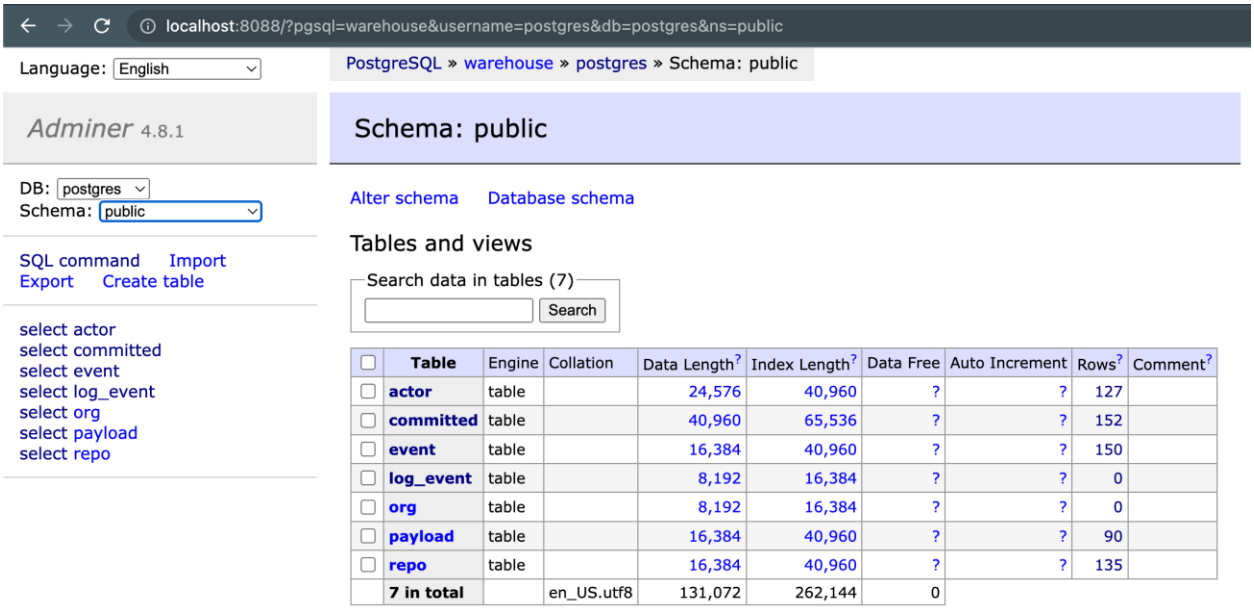
6. Run & Check Airflow schedule

   Frequency: Hourly



** Schedule was run successfully

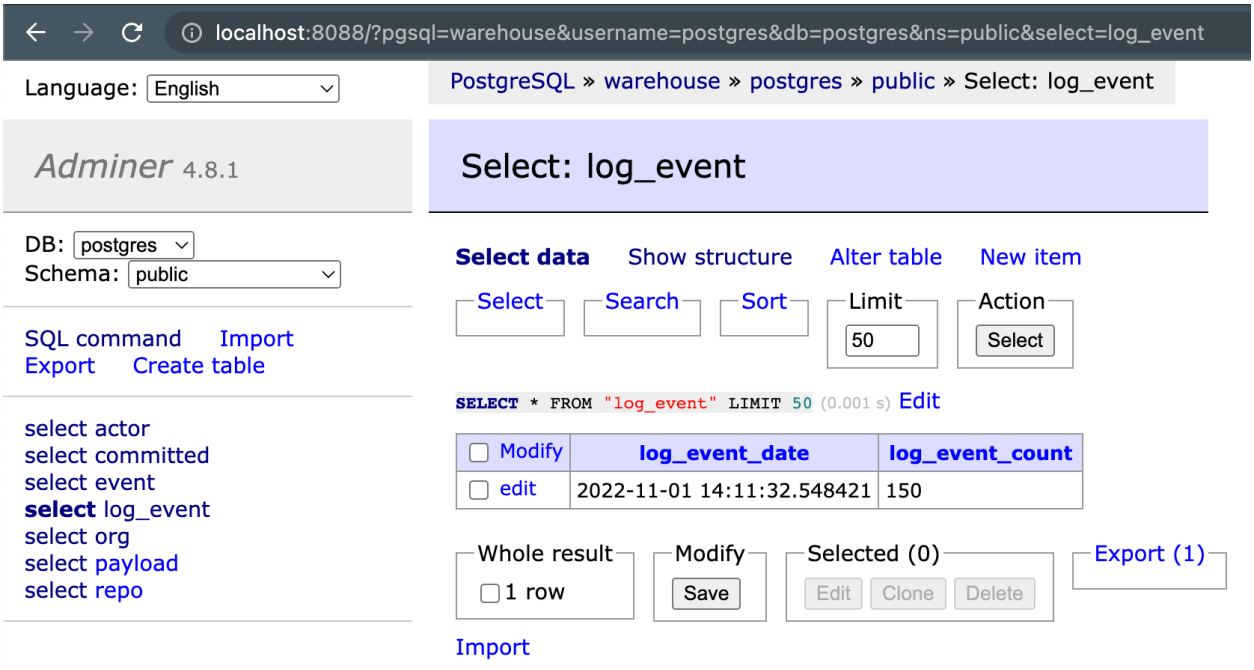
7. Check table creation and data loading in Postgres



** Tables were created and data were loaded properly

8. Check data in tables

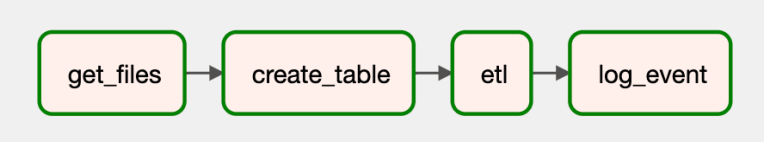
Table **log_event** : the timestamp and record count were inserted



DS525 - Chin Lertvipada - 64199130039

Appendix

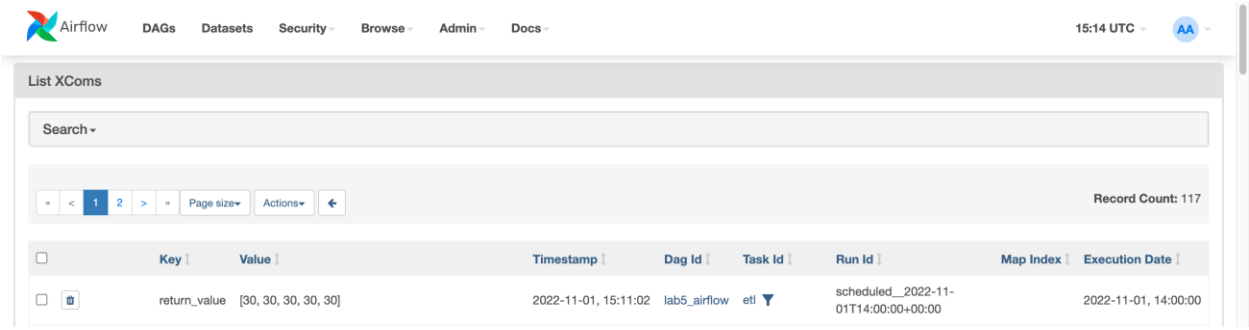
1. Workflow



   a. Get data files
   b. Create tables if not exist
   c. Loading the data if not exist
   d. Log the loading information (timestamp of loading & record count) for table **Event**

2. Using Xcoms to store returned output of task

   Xcoms variable:



   Access Xcoms variable:

```
def _log_event(**context):
    ti = context['ti']
    event_cnt = ti.xcom_pull(task_ids = 'etl', key = 'return_value')
```

3. DAG schedule

```
with DAG(
    'lab5_airflow',
    start_date = timezone.datetime(2022, 11, 1),
    schedule = '@hourly',
    tags = ['lab5'],
    catchup = False,
) as dag:
```

   Start: 2022-11-01

   Frequency: Hourly