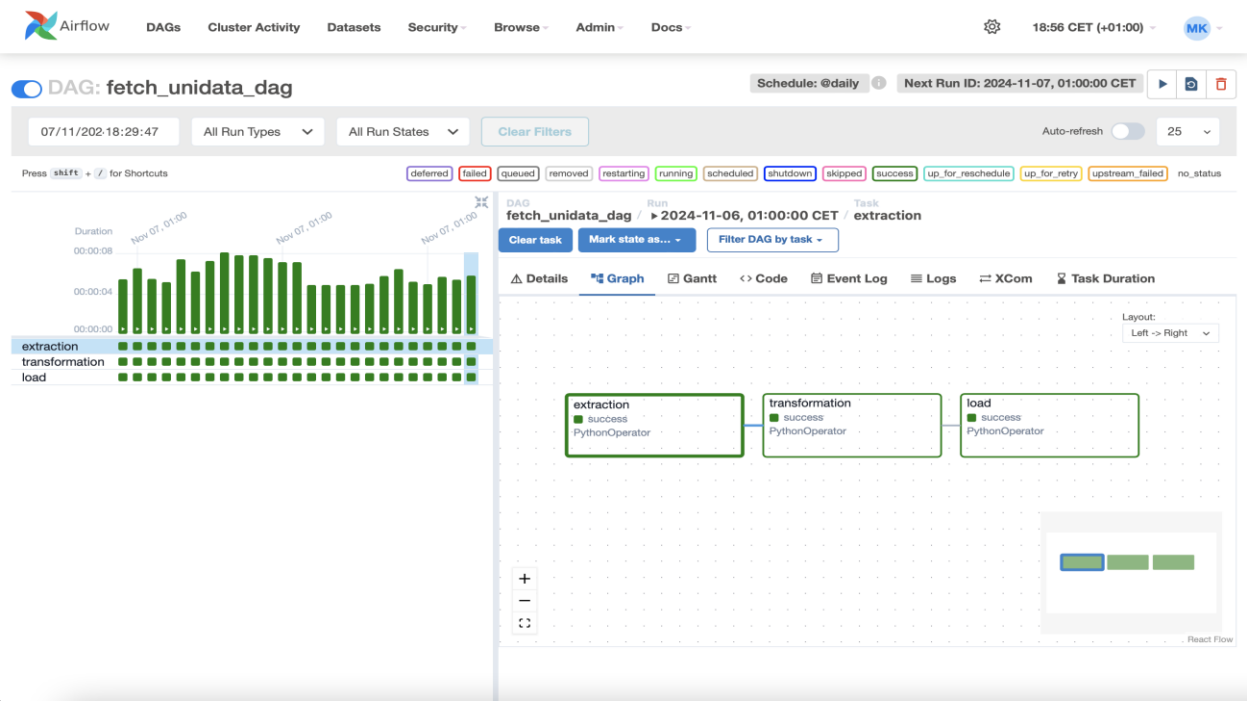
ETL Pipeline for Real-Time Data Analytics

Objective:

This ETL (Extract, Transform, Load) Data pipeline is designed to perform real time data analysis. In this pipeline data is extracted from external source (API) , transformed and then ingested into Postgres database.



Tools and Technologies Used:

- **Apache Airflow**:
- Docker
- Python
- PostgreSQL
- Postgres Hook
- DAGs and Python Operator
- Datetime and Time delta
- Visual Studio Code

Step#1: Defining Libraries:

```
#Libraries/dependencies
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.providers.postgres.hooks.postgres import
PostgresHook #for connection airflow with postgres database
from datetime import datetime
from datetime import timedelta
import requests
from airflow.models.xcom import XCom
```

Step#2: Defining and initializing the DAG:

```
initalizing the DAG

default_args = {
'Owner' : 'airflow',
'depend_on_past':False,
}

dag = DAG(
'fetch_unidata_dag',
default_args=default_args,
start_date=datetime(2023,1,1),
schedule = '@daily',
catchup=False,
)
```

Step#3: Defining python functions:

EXTRACTION:

```
#step1:Data extraction from API:

def data_extraction (**kwargs):
url =
"http://universities.hipolabs.com/search?country=United+States"
response = requests.get(url)
response.raise_for_status()
data = response.json()
if data is None or not data:
raise ValueError("API response is empty or None")
print(data)
kwargs['ti'].xcom_push(key='raw_data', value=data)
```

TRANSFORMATION:

```
#step2:Data transformation:
def data_transformation(**kwargs):
#need to pull data from xcom:
raw_data = kwargs['ti'].xcom_pull(task_ids='extraction',
key='raw_data')
print(f"Raw data received from XCom: {raw_data}")
if raw_data is None or not raw_data:
raise ValueError("No data received from 'data_extraction' task.")
#getting only specified data
transformed_data = [
{
'domains': item.get('domains', []), # Defaulting to empty list if
'domains' is missing
```

```
'state_province': item.get('state-province', ''),
'country': item.get('country', ''),
'name': item.get('name', ''),
'web_pages': item.get('web_pages', [])
}
for item in raw_data
]
print(transformed_data)

kwargs['ti'].xcom_push(key='transformed_data',
value=transformed_data)
```

LOAD:

```
#step3: Data load in postgres database:
def data_loading (**kwargs):
#step31 #getting data transformed data from xcom from
data_transformation task:

my_data = kwargs['ti'].xcom_pull(key=
'transformed_data',task_ids='transformation')
print(f"Data pulled from XCom: {my_data}")

if not my_data:
raise ValueError("No data received from 'data_transformation'
task.")

#step32 #creating a connection of airflow with postgres data_base
using postgres hook:
pg_hook = PostgresHook(postgres_conn_id='my_database_conn')
connection = pg_hook.get_conn()
```

```
cursor = connection.cursor()
try:
conn = hook.get_conn() # Try to get the connection
if conn:
print("Connection successful!")
else:
print("Failed to connect to database.")
except Exception as e:
print(f"Error connecting to database: {e}")

#step33 create a table with postgres command and execute with
cursor:
create_table_query = """
CREATE TABLE IF NOT EXISTS my_table (
domains TEXT[],
state_province varchar(255),
country varchar(255),
name varchar(255),
web_pages TEXT
);"""

cursor.execute(create_table_query)
#step34 insert data into this newly created table and execute it with
cursor:
insert_query="""
INSERT INTO my_table
(domains,state_province,country,name,web_pages)
values(%s,%s,%s,%s,%s)
"""
for record in my_data:
```

```
cursor.execute(insert_query,(record['domains'],record['state_provin
ce'],record['country'],record['name'],record['web_pages']))
#step 35 after commiting the data in table.close the cursor and
connection:
connection.commit()
cursor.close()
connection.close()
```

Step#4: Defining python Operators:

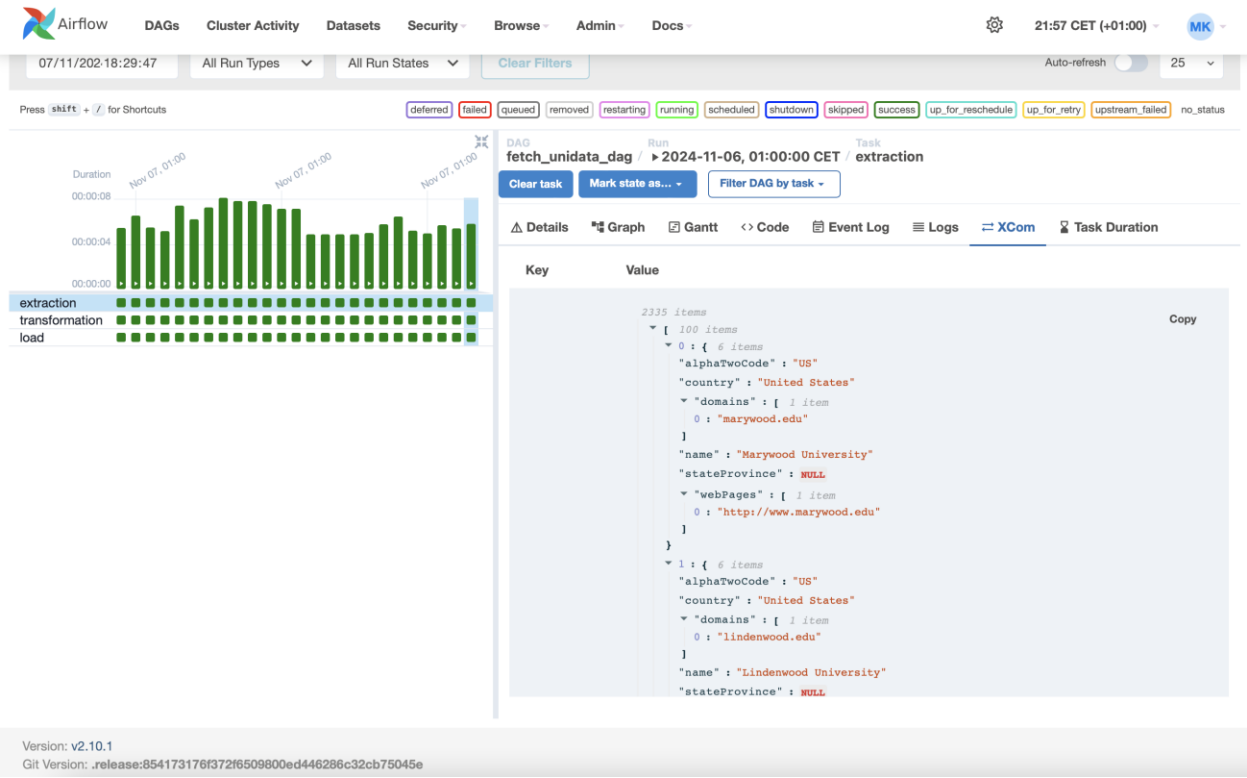
```
#defining python operators for carrying the tasks
task_extraction = PythonOperator(
task_id = 'extraction',
python_callable = data_extraction,
dag=dag,
)
task_transformation = PythonOperator(
task_id = 'transformation',
python_callable = data_transformation,
dag=dag,
)

task_load = PythonOperator(
task_id = 'load',
python_callable = data_loading ,
dag=dag,
)
```

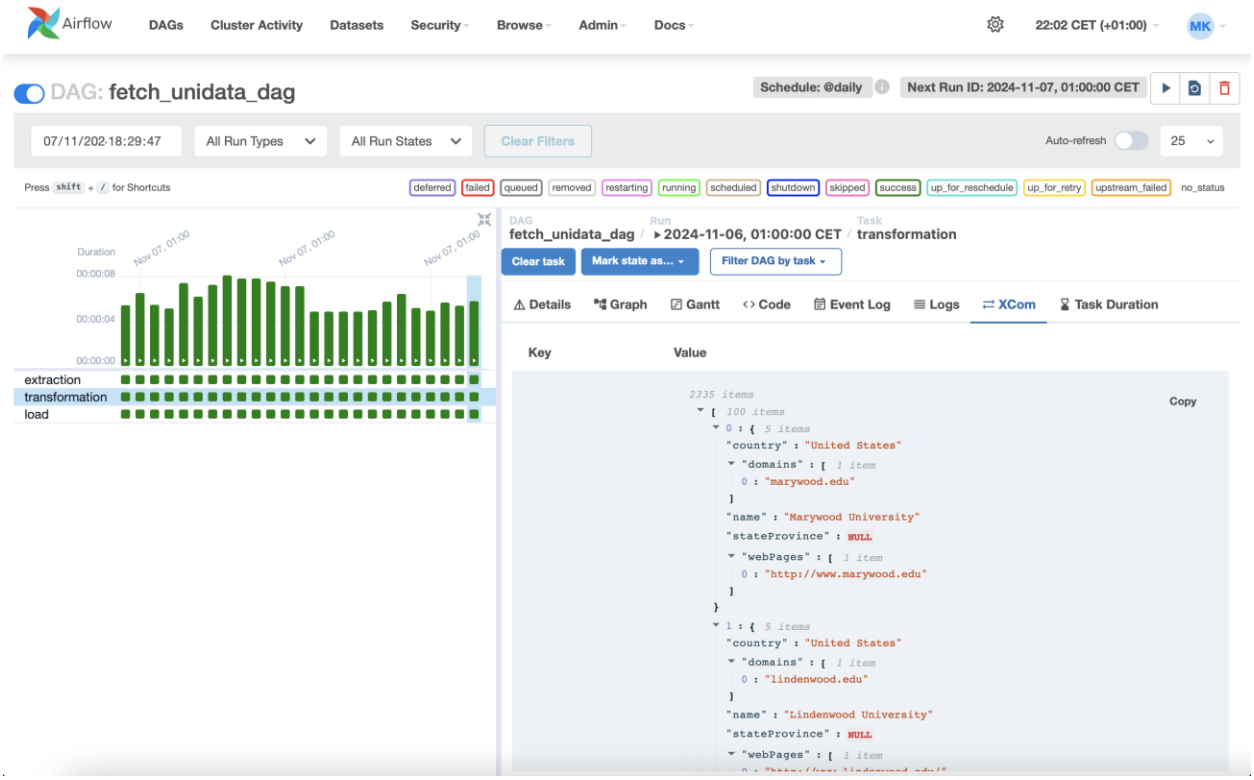
Step#5: Task Flow:

```
task_extraction >> task_transformation >> task_load
```

During Extraction, Data In Xcom:



During Transformation, Data In Xcom:



Task Duration: