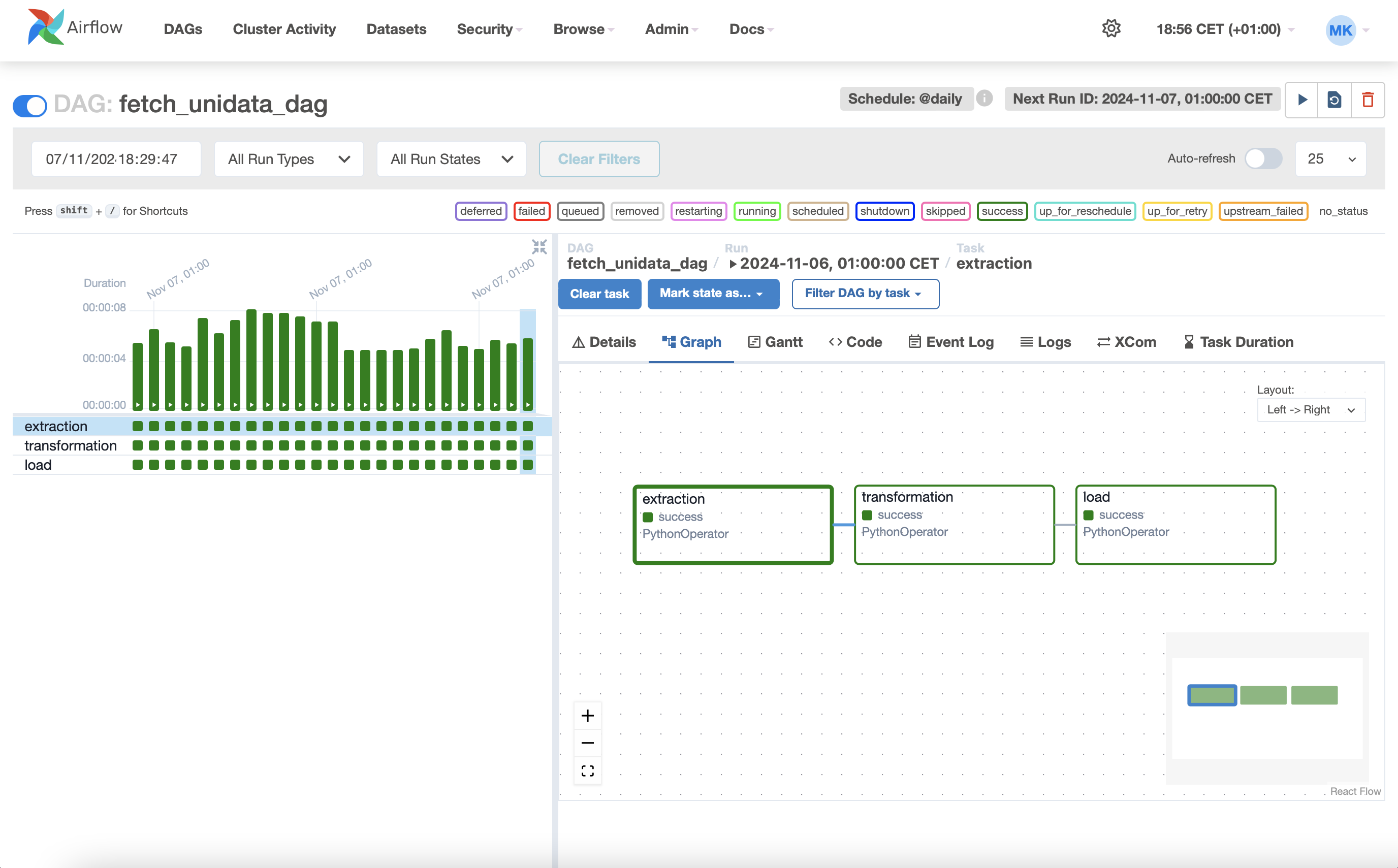
**ETL Pipeline for Real-Time Data Analytics**

**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\_province'],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 ,

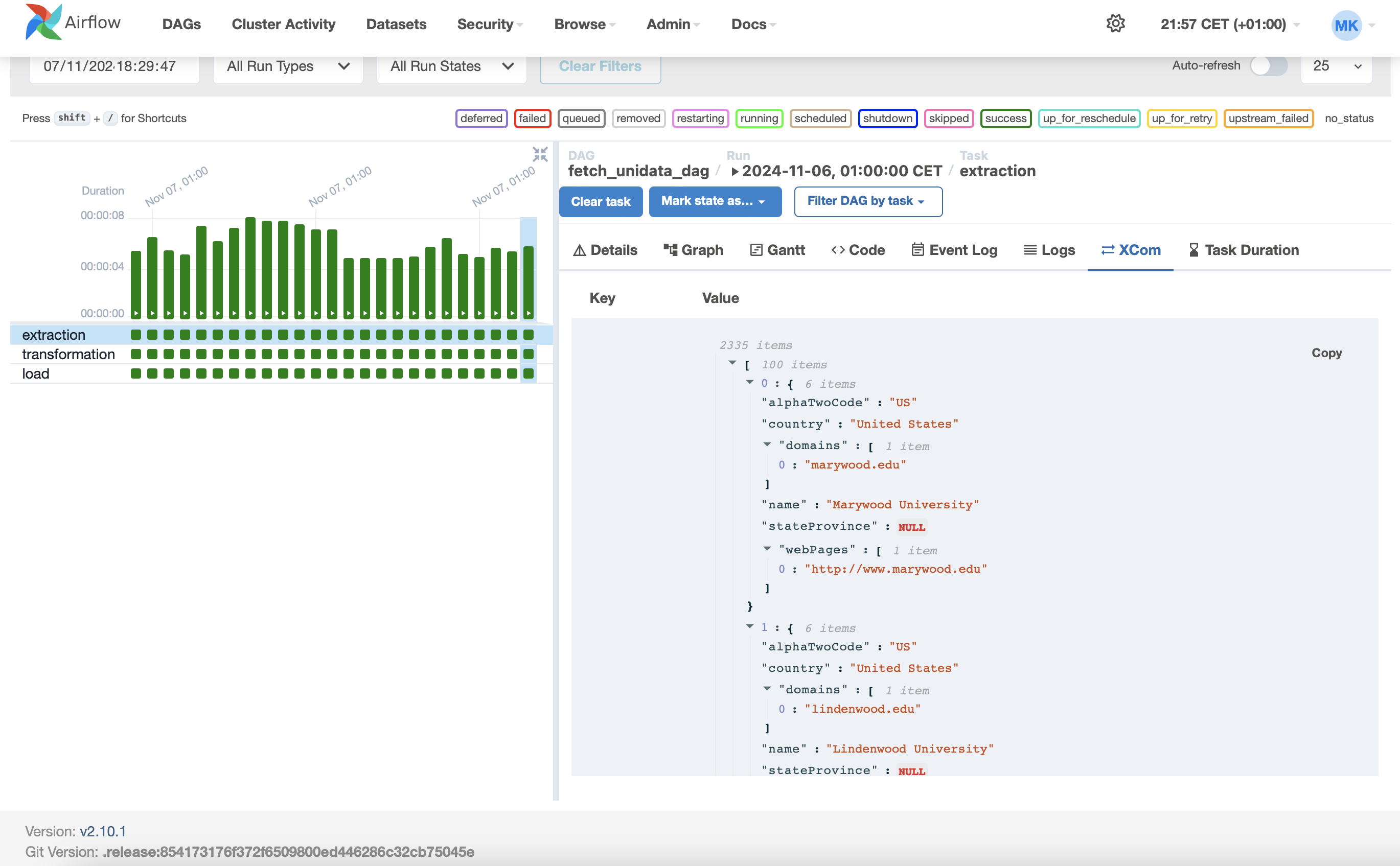
dag=dag,

)

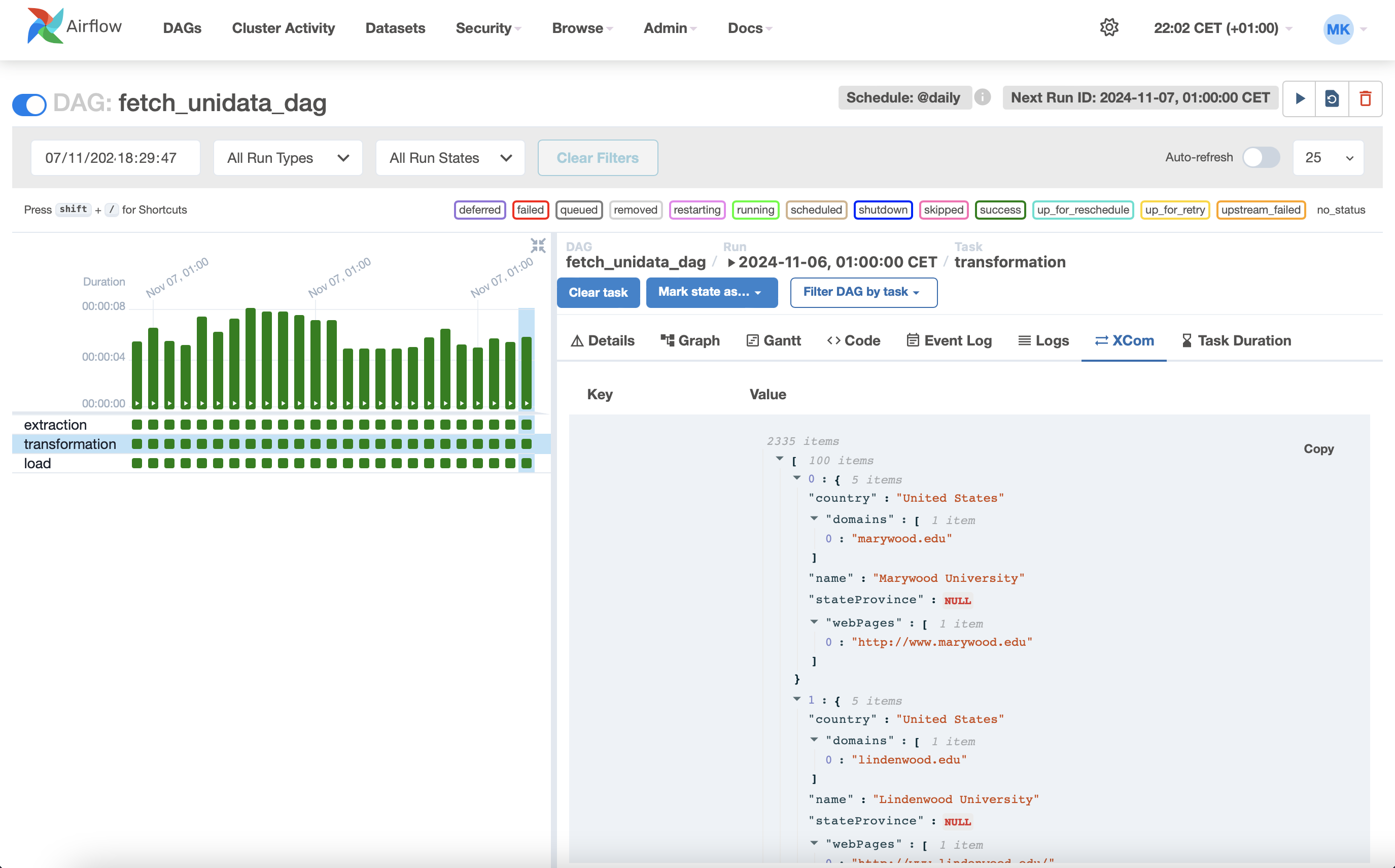
**Step#5: Task Flow:**

task\_extraction >> task\_transformation >> task\_load

**During Extraction, Data In Xcom:**



**During Transformation, Data In Xcom:**



**Task Duration:**

