# MLOPS (DA5402) - Assignment 2
# News Scraping Pipeline with Apache Airflow

Gokulakrishnan B
Roll No: DA24M007

February 10, 2025

## 1 Introduction

This report explains the implementation of the news scraping pipeline using apache airflow. This pipeline has two DAGs. One scraps the web, get the url of top stories, scrapes it and then store them in database after checking for duplicates, other sends email whenever new records are being added to the database.

## 2 Implementation Overview

The solution is implemented as two separate but interconnected DAGs:

- DAG 1: News Scraping Pipeline - Handles the core functionality of scraping, processing, and storing news articles
- DAG 2: Email Notification System - Manages state tracking and email notifications for new entries

The DAGs communicate through a status file, implementing a lightweight yet effective triggering mechanism.

## 3 DAG 1: News Scraping Pipeline

This DAG implements the core scraping functionality through four interconnected modules, each handling a specific aspect of the data pipeline:

### 3.1 Module 1: Homepage Scraping

This module initiates the scraping process by accessing the Google News homepage. It uses requests library for HTTP communication and implements proper error handling for network issues. The scraped content is passed between tasks using Airflow's XCom feature, ensuring reliable data transfer between modules.

```
def module_1(url,ti):
    response = requests.get(url)
    ti.xcom_push(key="home_page",value=response.text)
```

## 3.2 Module 2: Top Stories Extraction

The second module focuses on parsing the homepage content to locate the "Top stories" section. It employs BeautifulSoup for HTML parsing and implements pattern matching to reliably identify the correct link.

```
def module_2(pattern,ti):
    response = ti.xcom_pull(key="home_page",task_ids="module_1")
    soup = BeautifulSoup(response, 'html.parser')
    links = soup.find_all('a')
    for link in links:
        if pattern.lower() in link.get_text().lower():
            href = link.get('href')
            if href:
                return f"https://news.google.com{href[1:-1]}"
```

## 3.3 Module 3: Article Extraction

This module handles the complex task of extracting article details from the Top Stories page. It handles lazyloading by refresing the page multiple times, configured by the user.

```
# Extract publication date and handle missing data
time_element = article.find('time')
publication_date = time_element.get('datetime') if time_element else
    None

# Smart thumbnail extraction with lazy loading support
figure = article.find('figure')
if figure and figure.find('img'):
    img = figure.find('img')
    thumbnail_url = img.get('src') or img.get('data-src')
```

## 3.4 Module 4: Database Operations

This module manages data persistence and implements sophisticated duplicate detection:

- Uses Jaccard similarity for fuzzy matching of headlines
- Implements efficient batch processing
- Maintains referential integrity between images and headlines
- Tracks successful insertions for notification purposes

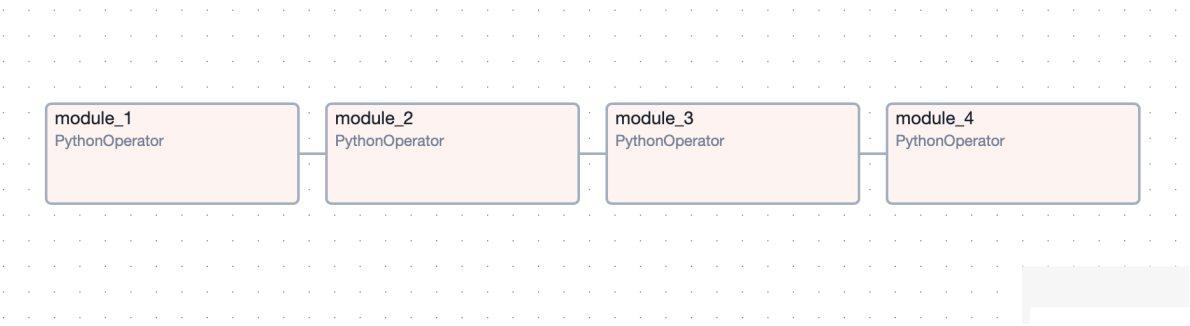


Figure 1: DAG 1 graph view

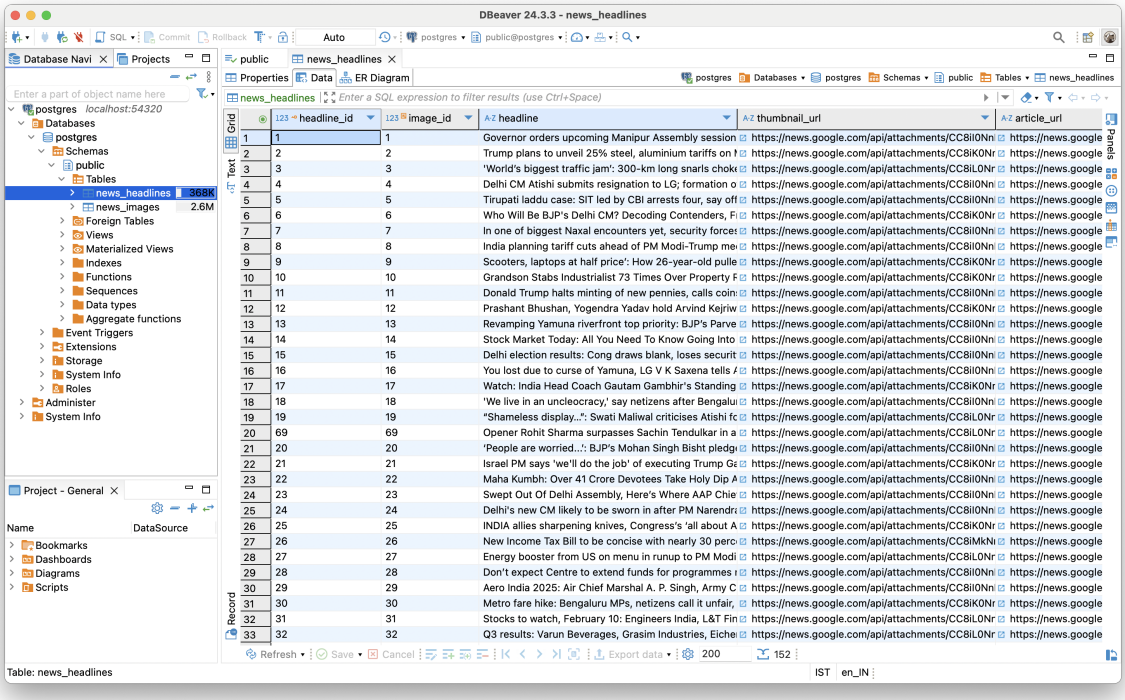


Figure 2: Headlines table

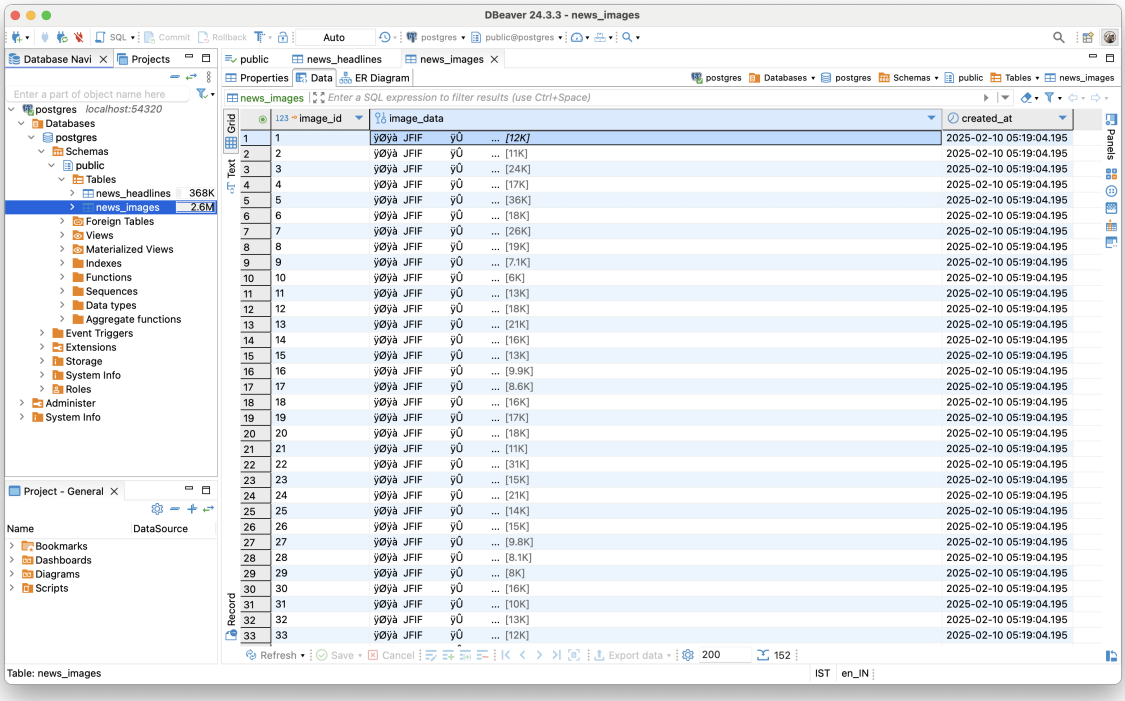


Figure 3: thumbnail(image) table

# 4 DAG 2: Email Notification System

Airflow is configured with the SMTP protocol to be able to send emails. Here, it is triggered by the successful execution of DAG1. Then it checks for the status file under the run directory. if exists, it checks the value in it, if not 0, it queries that much number of entries from database and headline id of latest of insert is also remembered to fetch only the new emails. Then the status file is deleted if the email is successfully sent, so that we can check for the successful execution of next iteration of dag 1.

Key components:

```
wait_for_status = FileSensor(
    task_id='wait_for_status_file',
    fs_conn_id='file',
    filepath='/opt/airflow/dags/run/status',
    poke_interval=30,
)
```

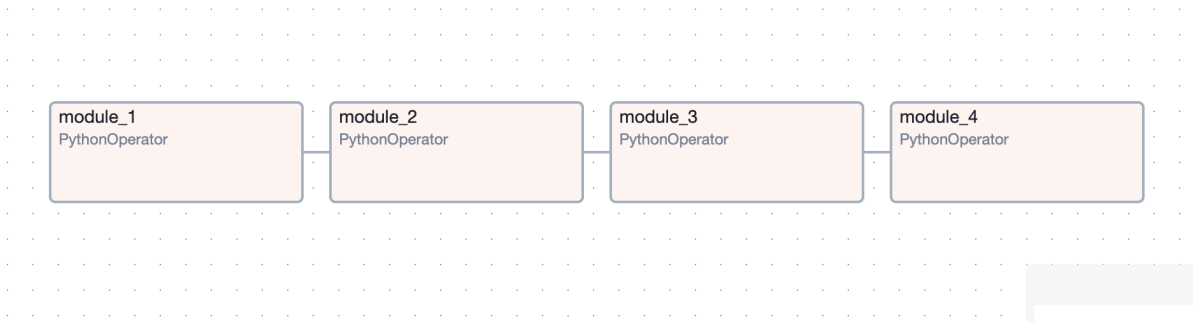


Figure 4: DAG 2 graph view

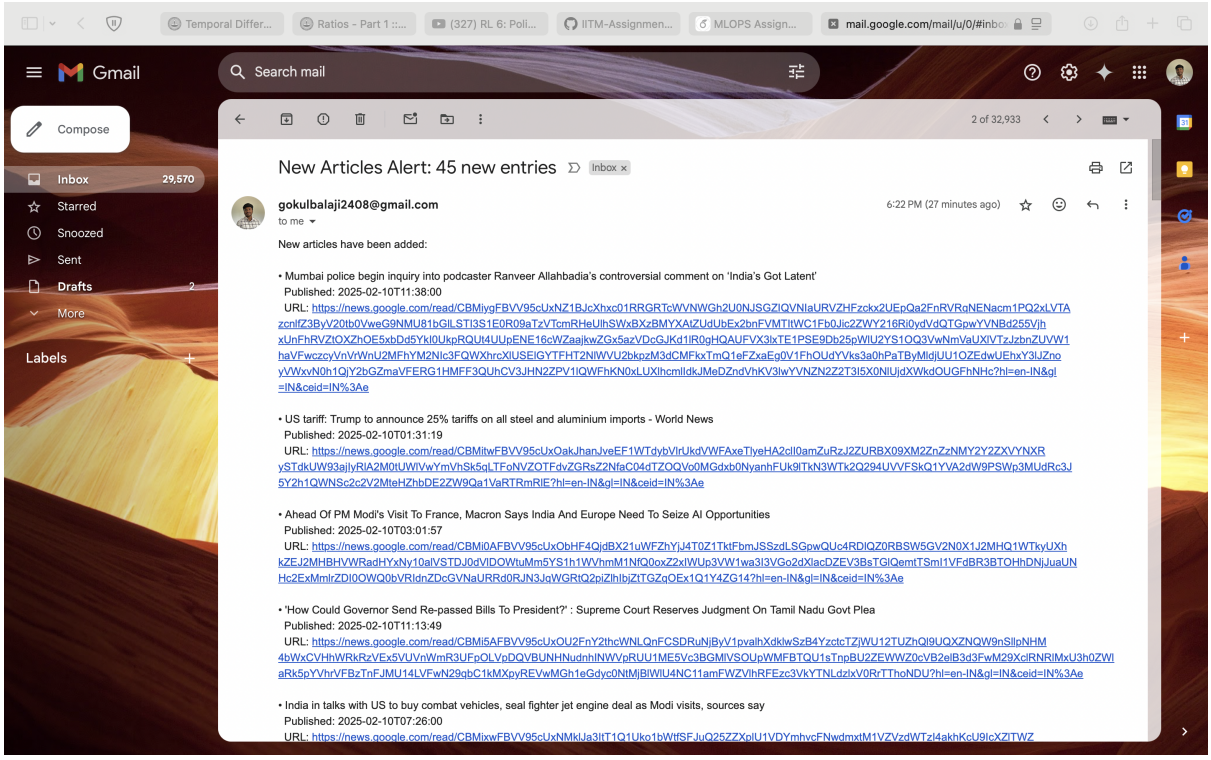


Figure 5: Email screenshot

# 5 Workflow Integration

The integration between the two DAGs is achieved through multiple mechanisms:

- File-based triggering using status files
- State persistence using JSON for tracking processed entries
- External Task Sensor for DAG synchronization
- Shared database access for data consistency

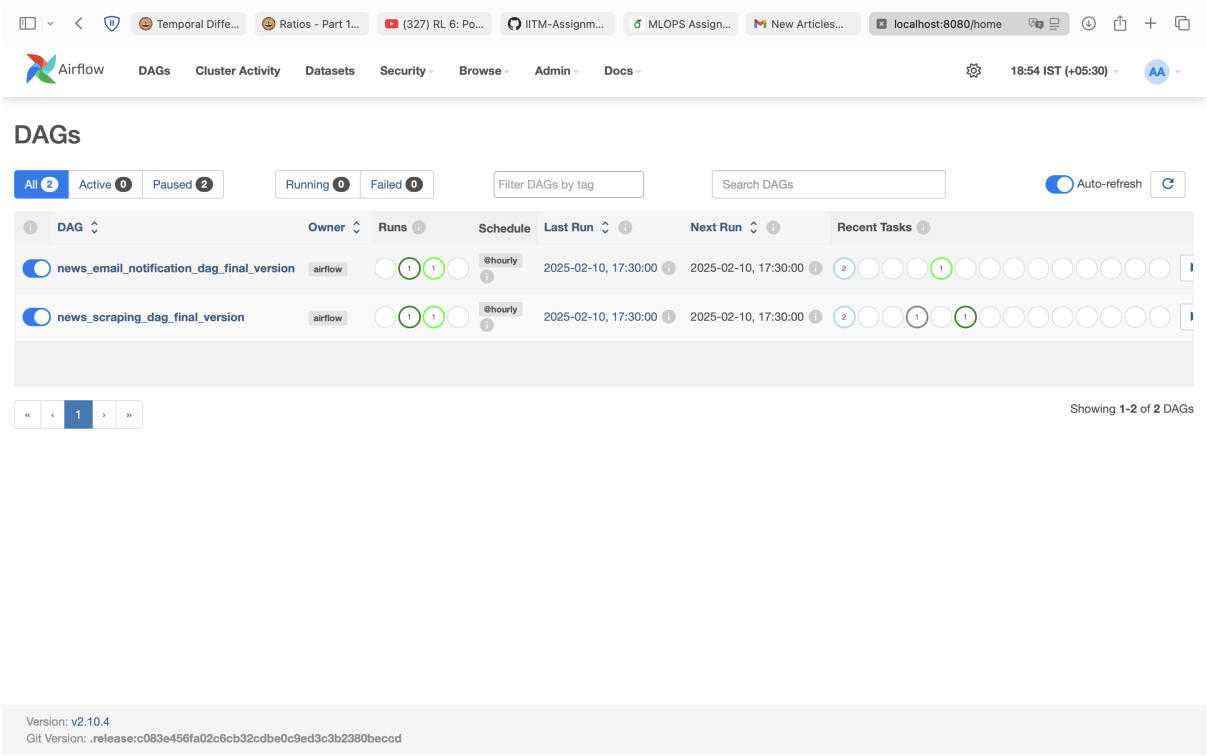


Figure 6: Airflow Dashboard showing both DAGs

# 6 Database Implementation

Two main tables:

```sql
CREATE TABLE news_images (
    image_id SERIAL PRIMARY KEY,
    image_data BYTEA NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE news_headlines (
    headline_id SERIAL PRIMARY KEY,
    image_id INTEGER REFERENCES news_images(image_id),
    headline TEXT NOT NULL,
    thumbnail_url TEXT,
    article_url TEXT,
    publication_date TIMESTAMP,
    scrape_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(headline, article_url)
);
```

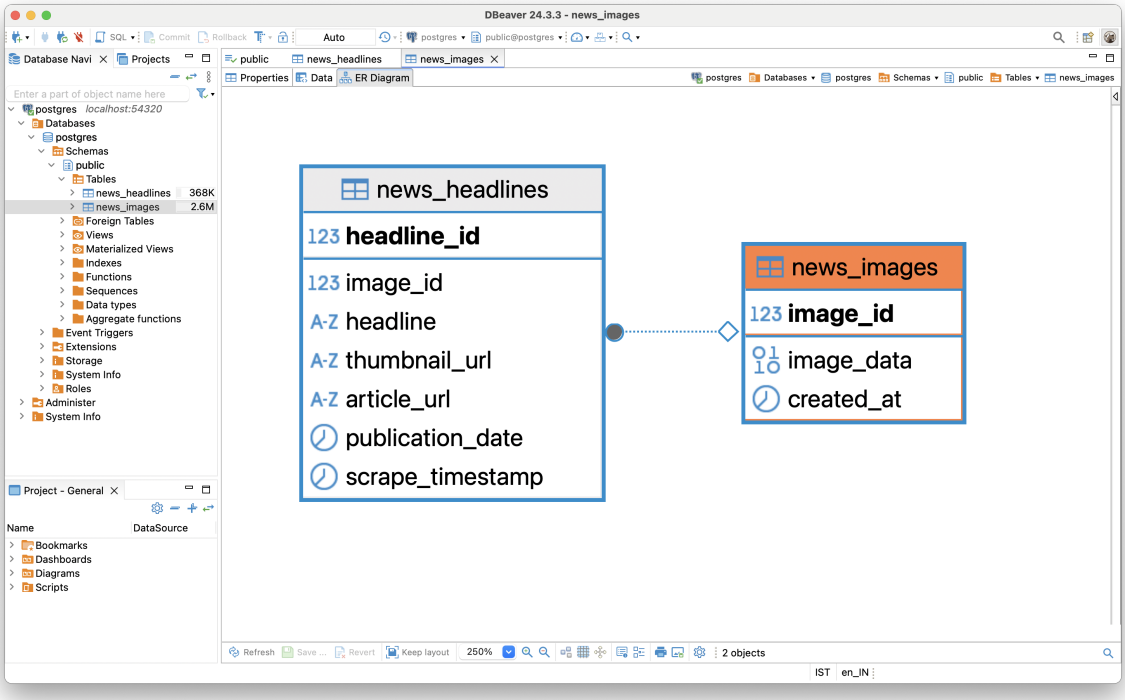


Figure 7: Database ER Diagram

# 7 Conclusion

The implementation successfully creates an automated news scraping and notification system using Apache Airflow. The system demonstrates the power of workflow orchestration and the benefits of using appropriate tools for automation tasks.