News Extractor Pipeline

1. Project Overview

This project implements an automated ETL (Extract, Transform, Load) pipeline designed to ingest news headlines from five major Indonesian news portals: Kompas, Detik, Tempo, CNN Indonesia, and Liputan6.

The system is containerized using Docker and orchestrated by Apache Airflow, ensuring reproducibility and scalability. The pipeline is scheduled to run daily, processing data from raw HTML/XML sources into a structured SQLite database.

2. Project Files Structure

dags/	# Folder for DAG script
└── news_scraper_dag.py	# Airflow DAG script for ETL pipelina
data/	# Folder to save scraping results
└── news_articles.db	# SQLite of scraping result
└── hasil_berita.csv	# CSV files of scraping result
logs/	# Airflow Logging
docker-compose.yaml	# Docker Compose setup for Airflow
Dockerfile	# Custom Airflow image with required Python packages
requirements.txt	# List of required python packages
check_query.py	# Python script to query from SQLite and save to CSV

3. Architecture and Tech Stack

The solution is built upon a microservices architecture managed by Docker Compose:

- **Orchestration:** Apache Airflow 2.9.0 (LocalExecutor).
- **Language:** Python 3.x (utilizing requests, BeautifulSoup4, pandas).
- **Database:** SQLite (Persistent storage via Docker Volumes).
- **Containerization:** Docker & Docker Compose.

4. Pipeline Detail

a. Pipeline DAG Structure

The Airflow DAG consists of four sequential tasks: init_db_task -> extract_task -> transform_task -> load_task

b. Extract Layer (Scraping)

The goal is to extract headlines, URLs, and publication dates. The strategy is using a Hybrid Extraction Strategy was implemented to handle different anti-bot protections and structure variability.

1. RSS Feed Parsing (Standard Strategy):

   Used for Detik, Tempo, CNN Indonesia, and Liputan6. RSS feeds provide structured XML data (<item>, <title>, <pubDate>) which is more stable and less prone to blocking than HTML scraping. To implement this method I used BeautifulSoup(content, 'xml') to parse the feeds.

2. Pagination & Pattern Matching (Kompas Strategy):

   Kompas often blocks standard requests or has unstable RSS feeds. To handle this use a robust scraping method targeting the Index page (indeks.kompas.com). The script iterates through multiple pages (?page=1 to ?page=3) to ensure a high volume of data. Instead of relying on fragile CSS classes, the script scans for all <a> tags containing the pattern /read/ (news article URL signature) and uses Regex to extract titles from nested <h2> or <div> tags within the link.

c. Transform Layer (Cleansing)

The goal is to cleanse data and standardize schema. To implement this strategy, The data is processed using pandas

1. Deduplication

   Duplicate articles are removed based on the url column. This prevents overlapping data from multiple pipeline runs.

2. Date Parsing

   Sources use mixed date formats (e.g., RSS uses RFC 822 English, while HTML uses Indonesian 09 Desember 2025). A function smart_parse_date() was created. It first attempts to parse standard RSS formats. If that fails, it uses a dictionary mapping (e.g., 'Desember': '12') to translate Indonesian months and convert the string into a valid ISO 8601 format.

3. URL Sanitization

Tracking parameters (e.g., ?utm_source=...) are stripped from URLs to ensure clean, unique identifiers.

d. Load Layer (Storage)

1. Schema:
   - id (INTEGER PRIMARY KEY AUTOINCREMENT)
   - source (TEXT)
   - title (TEXT)
   - url (TEXT UNIQUE)
   - published_at (TEXT - ISO 8601)
   - scraped_at (TEXT - ISO 8601)
2. Idempotency**:** The SQL statement INSERT OR IGNORE is used. This ensures that if the pipeline re-runs, existing articles (identified by the UNIQUE url) are skipped, preserving data integrity without duplication.

5. Orchestration (Airflow)

- **DAG ID**: news_etl_pipeline_v1
- **Schedule**: 0 7 * * * (Setiap jam 7 pagi pada menit 0 tepat)
- **Data Passing:** Airflow XCom is used to pass data payloads (list of dictionaries) between the Extract, Transform, and Load tasks.

6. How to Run

1. Create folder **data/** and **logs/**
2. Ensure Docker Desktop is running.
3. Navigate to the project root directory.
4. Execute the following command to start the environment: **docker-compose up –d**
5. Access the Airflow UI at http://localhost:8080.
6. Login with credentials (default: admin/admin).
7. Toggle the news_etl_pipeline_v1 DAG to ON and trigger it manually to verify execution.
8. Output data can be verified in data/news_articles.db.