XKCD Comics Case Study

Adam Boros

- API Documentation - https://xkcd.com/json.html
- Docker Setup - https://github.com/borosadi/jet_xkcd_webcomics
- Airflow DAG
- dbt Transformations, DWH Model
- Data Quality Checks
- Further Improvements

XKCD Comics API documentation

If you want to fetch comics and metadata automatically, you can use the JSON interface. The URLs look like this:

https://xkcd.com/info.0.json (current comic)

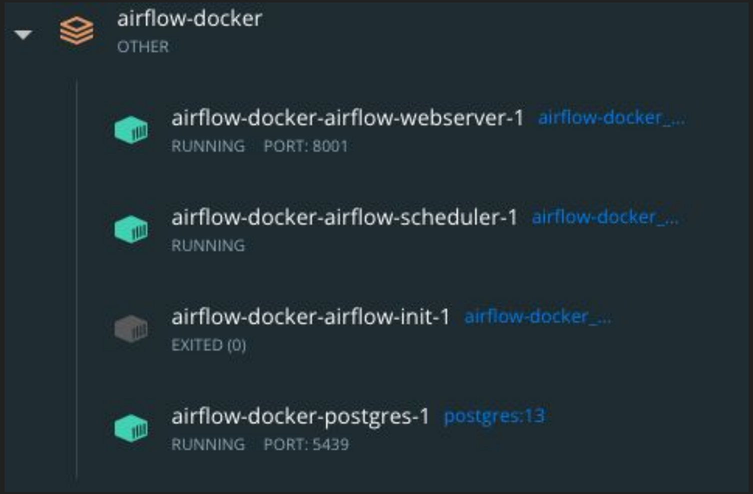
or:

https://xkcd.com/614/info.0.json (comic #614)

Those files contain, in a plaintext and easily-parsed format: comic titles, URLs, post dates, transcripts (when available), and other metadata.

Docker Setup

- git clone https://github.com/borosadi/jet_xkcd_webcomics.git
- cd airflow-docker
- docker compose up -d
- Airflow server: localhost:8080
- Postgres server: localhost:5439



airflow-docker
OTHER

airflow-docker-airflow-webserver-1 airflow-docker_...
RUNNING PORT: 8001

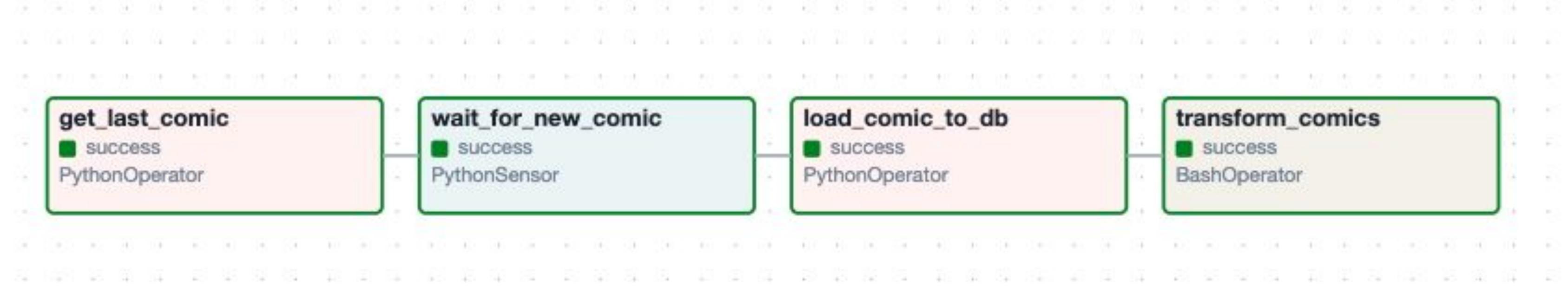
airflow-docker-airflow-scheduler-1 airflow-docker_...
RUNNING

airflow-docker-airflow-init-1 airflow-docker_...
EXITED (0)

airflow-docker-postgres-1 postgres:13
RUNNING PORT: 5439

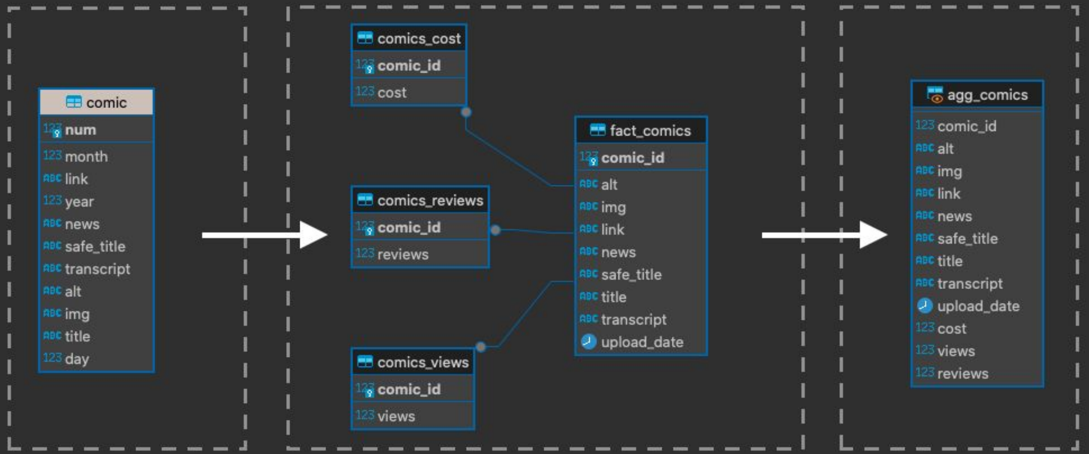
Airflow DAG

- get latest comic from database
- checking new comics on API
- load new comics to database
- run dbt transformations and test



dbt Transformations, DWH Model

- create fact tables from stage with enforced schema
- incremental materialization
- create aggregated view from fact tables



Stage table with raw data from API

Fact tables loaded from stage

Aggregate view generated from facts

Data Quality Checks

- valid date fields, year, month, day values (date(year, month, day) failure)
- consistency of urls in image field (regex string pattern), available link (check for response 200)
- uniqueness and null value test of “num”/comic_is (primary key)
- relationship test making sure that cost, views and reviews have corresponding comics (foreign keys)
- volume test, looking for sudden changes in daily data volumes (comics/day)
- freshness test, alert if comic is not available on the expected days

Further improvements

- unittest
- error handling
- logging
- async API requests
- split dbt transformations into different tasks
- implement dbt macros for automated data quality checks

Thank You!