MEDIRECT

DATA ENGINEERING ASSESSMENT

Python Developer

ENVIRONMENT SETUP

Apache Airflow
Postgres
Jenkins

APACHE AIRFLOW

Installation Procedure

- Setting up of Apache Airflow v2.7.2 docker stack on a local environment (macOS) was carried by following through the official installation guides.
- The official *docker-compose.yml* file includes the provision of CeleryExecutor rather than the LocalExecutor. The latter executor is capable of running tasks sequentially – one task instance at a time and must reside on the same machine as the scheduler.
- Minor changes applied to docker yaml file:
  - Enabling port forwarding for PostgreSQL database server to hosting machine i.e TCP 5432:5432. This is required to provisioning the database and database tables via Azure Data Studio.
  - Addition of a dockerfile to airflow-common to enable the provision of addition of python libraries via requirements.txt
  - Explicit use of a docker env file - docker compose up airflow-init --env-file docker-compose.env (rather than hidden .env file)
- Inclusion of *airflow.sh* wrapper script to invoke commands on the web server via docker compose/terminal CLI.

  e.g Creation of a new application user having admin rights.

```
./airflow.sh users create --username mark --password mark --firstname Mark --lastname Grech --
role Admin --email grechmarkj@gmail.com
```

APACHE AIRFLOW METASTORE

Installation Procedure

- The creation and initialisation of the Airflow's metastore took place automatically during the initial setup process.
- Environment variables pertaining to the metastore were left unchanged.

```
AIRFLOW__DATABASE__SQL_ALCHEMY_CONN
AIRFLOW__DATABASE__SQL_ALCHEMY_SCHEMA
```

- However, there were an instance where `airflow db migrate` command had to to explicitly invoked.
- Specification of project folder. By default, docker compose creates a project folder in the same directory where the yaml file resides.

```
AIRFLOW_PROJ_DIR=/Volumes/DATA2/Docker/airflow_stack_official/
```

- Avoid the loading of demo DAGS

```
AIRFLOW__CORE__LOAD_EXAMPLES=False
```

- User UID is set in response to the warning message reported during the provision of containers. Documentation states that the uid denotes the user to run containers as and would could affect the mounting of shared volumes.

```
AIRFLOW_UID=50000
```

AIRFLOW CONFIGURATION

- The database and API endpoints are defined as Airflow Variables and Connections.
- Defined programmatically via shell script - `init_connections.sh`
  - Deletion of existing objects
  - Creation of connections and variables.
- **List of Connections**
  - `api_endpoint_openholidaysapi_org` – airflow connection to openholidaysapi.org REST API endpoint
  - `api_endpoint_openerapi_com_v6` – airflow connection to open.er-api.com REST API endpoint
- **List of Variables**
  - `Environment` – Denotes the type of airflow setup - development/uat/pre-pod/production. Within DAG files the environment variable is read. In a non-prod setting the data returned by tasks are included in the log file for debugging purposes.

AZURE DATA STUDIO / POSTGRES DB

- Installation of PostgreSQL plugin since only MS SQL server is supported by default.
- Testing connectivity to local database instance using default administrative credentials.
- Creation of *db_currency_exchange* database and database objects via the DDL script (*db_init.sql*).
- The creation of the underlying database objects was performed outside of the data pipeline. The rationale behind this approach is based on the separation of data manipulation and database definition tasks. Data pipelines are generally complex resulting in additional maintenance/support efforts should DDL scripts be embedded in the pipelines.
- Security. Following the least privileges principle. The creation of an application database user having restricted permissions on the default database schema (*airflow_app_usr*).

JENKINS

- The provision of a Jenkins docker container to enable CI/CD.
- Reverted to the official and latest version of Jenkins Docker container.

  => `docker-compose-Jenkins.yaml`
- Installation of SSH Agent plugin + other standard plugins.

APACHE AIRFLOW

airflow.sh info

```
Apache Airflow
version                 | 2.7.2
executor                | CeleryExecutor
task_logging_handler    | airflow.utils.log.file_task_handler.FileTaskHandler
sql_alchemy_conn        | postgresql+psycopg2://airflow:airflow@postgres/airflow
dags_folder             | /opt/airflow/dags
plugins_folder          | /opt/airflow/plugins
base_log_folder         | /opt/airflow/logs
remote_base_log_folder |


System info
OS                  | Linux
architecture        | x86_64
uname               | uname_result(system='Linux', node='b081596f7361', release='6.4.16-linuxkit', version='#1 SMP PREEMPT_DYNAMIC Tue Oct 10 20:42:40 UTC 2023',
                    | machine='x86_64', processor='')
locale              | ('en_US', 'UTF-8')
python_version      | 3.8.18 (default, Oct 11 2023, 23:57:43)  [GCC 10.2.1 20210110]
python_location     | /usr/local/bin/python


Tools info
git                 | NOT AVAILABLE
ssh                 | OpenSSH_8.4p1 Debian-5+deb11u2, OpenSSL 1.1.1w  11 Sep 2023
kubectl             | NOT AVAILABLE
gcloud              | NOT AVAILABLE
cloud_sql_proxy     | NOT AVAILABLE
mysql               | mysql  Ver 8.0.34 for Linux on x86_64 (MySQL Community Server - GPL)
sqlite3             | 3.34.1 2021-01-20 14:10:07 10e20c0b43500cfb9bbc0eaa061c57514f715d87238f4d835880cd846b9ealt1
psql                | psql (PostgreSQL) 16.0 (Debian 16.0-1.pgdg110+1)


Paths info
airflow_home        | /opt/airflow
system_path         | /root/bin:/home/airflow/.local/bin:/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
python_path         | /home/airflow/.local/bin:/usr/local/lib/python38.zip:/usr/local/lib/python3.8:/usr/local/lib/python3.8/lib-dynload:/home/airflow/.local/lib/pytho
                    | n3.8/site-packages:/usr/local/lib/python3.8/site-packages:/opt/airflow/dags:/opt/airflow/config:/opt/airflow/plugins
airflow_on_path     | True


Providers info
apache-airflow-providers-amazon           | 8.7.1
apache-airflow-providers-celery           | 3.3.4
apache-airflow-providers-cncf-kubernetes  | 7.6.0
apache-airflow-providers-common-sql       | 1.7.2
apache-airflow-providers-daskexecutor     | 1.0.1
apache-airflow-providers-docker           | 3.7.5
apache-airflow-providers-elasticsearch    | 5.0.2
apache-airflow-providers-ftp              | 3.5.2
apache-airflow-providers-google           | 10.9.0
apache-airflow-providers-grpc             | 3.2.2
apache-airflow-providers-hashicorp        | 3.4.3
apache-airflow-providers-http             | 4.5.2
apache-airflow-providers-imap             | 3.3.2
apache-airflow-providers-microsoft-azure  | 7.0.0
apache-airflow-providers-mysql            | 5.3.1
apache-airflow-providers-odbc             | 4.0.0
apache-airflow-providers-openlineage      | 1.1.0
apache-airflow-providers-postgres         | 5.6.1
apache-airflow-providers-redis            | 3.3.2
apache-airflow-providers-sendgrid         | 3.2.2
apache-airflow-providers-sftp             | 4.6.1
apache-airflow-providers-slack            | 8.1.0
apache-airflow-providers-snowflake        | 5.0.1
apache-airflow-providers-sqlite           | 3.4.3
apache-airflow-providers-ssh              | 3.7.3

(base) mark@imac-9490m Docker_official %
```

APACHE AIRFLOW

```
docker ps
```

```
(base) mark@imac-9490m ~ % docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS                    PORTS                    NAMES
249842710fca   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   22 hours ago   Up 25 minutes (healthy)   8080/tcp                 docker_official-airflow-triggerer-1
1da51203c2c0   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   22 hours ago   Up 25 minutes (healthy)   8080/tcp                 docker_official-airflow-worker-1
a2f7d3c21595   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   22 hours ago   Up 25 minutes (healthy)   8080/tcp                 docker_official-airflow-scheduler-1
2dc893bafcdb   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   22 hours ago   Up 25 minutes (healthy)   0.0.0.0:8080->8080/tcp   docker_official-airflow-webserver-1
22e30d20f4e4   postgres:13            "docker-entrypoint.s…"   22 hours ago   Up 25 minutes (healthy)   0.0.0.0:5432->5432/tcp   docker_official-postgres-1
01c3ded706dd   redis:latest           "docker-entrypoint.s…"   22 hours ago   Up 25 minutes (healthy)   6379/tcp                 docker_official-redis-1
```

PYTHON TRANSFORMATION JOBS

DAGS

DAGS

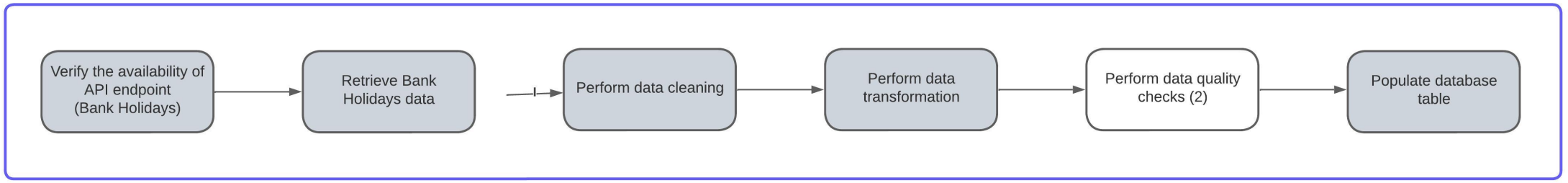
1. Two separate pipelines were designed considering that each pipeline differ in execution frequencies.
2. It is understood that the bank holidays pipeline needs to be executed on a yearly basis. Data pertaining to the specified county is retrieved in a single request.
3. In contrast, the most recent currency exchange rates would need to be retrieved at least on a daily basis.
4. The bank holiday data set includes a start date and an end date – seemingly identical. This means that the all public holiday spans over a single day. However, this assumption was taken into account and an informational entry was included whenever multi-day holidays are encountered. Such a message would assist in troubleshooting and data quality initiatives.
5. The bank holiday data set was transformed such that holidays spanning over multiple days are represented in the form of multiple records.

DAGS

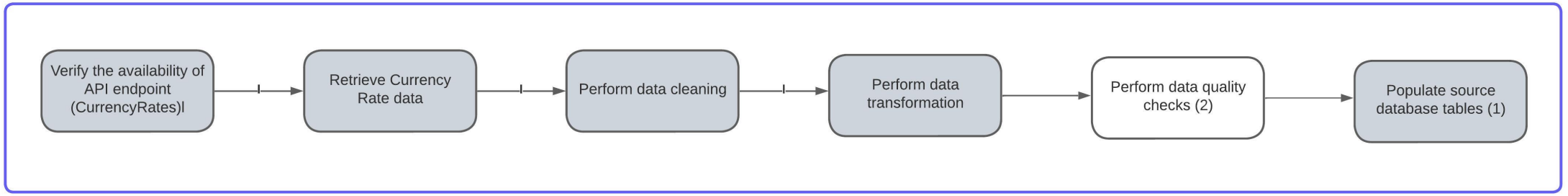
6. Data exchange between tasks operators occurs through XCOM. Objects were relayed in form of a dict. This approach is thought to be more extensible vis-à-vis other data structures such as lists or tuples whenever additional is required to the passed from one task to another.

DAGS

public_holidays_dag



currency_rate_dag

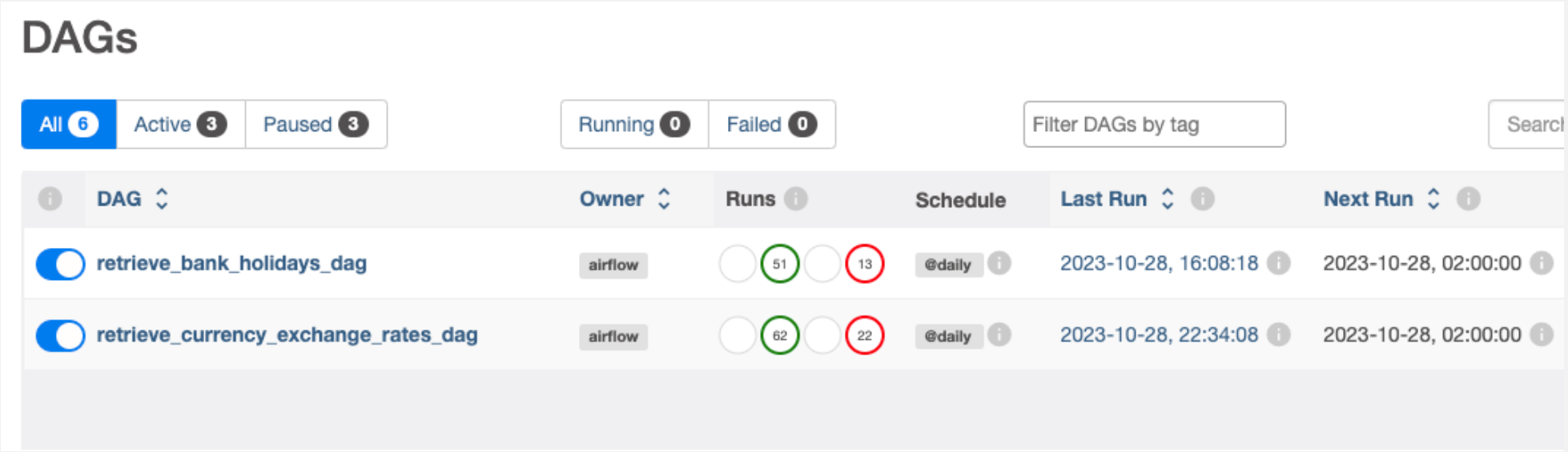


1. Source tables are populated in a single database transaction

2. A task can be added to a data pipeline that would perform data quality checks prior to persisting data in DB

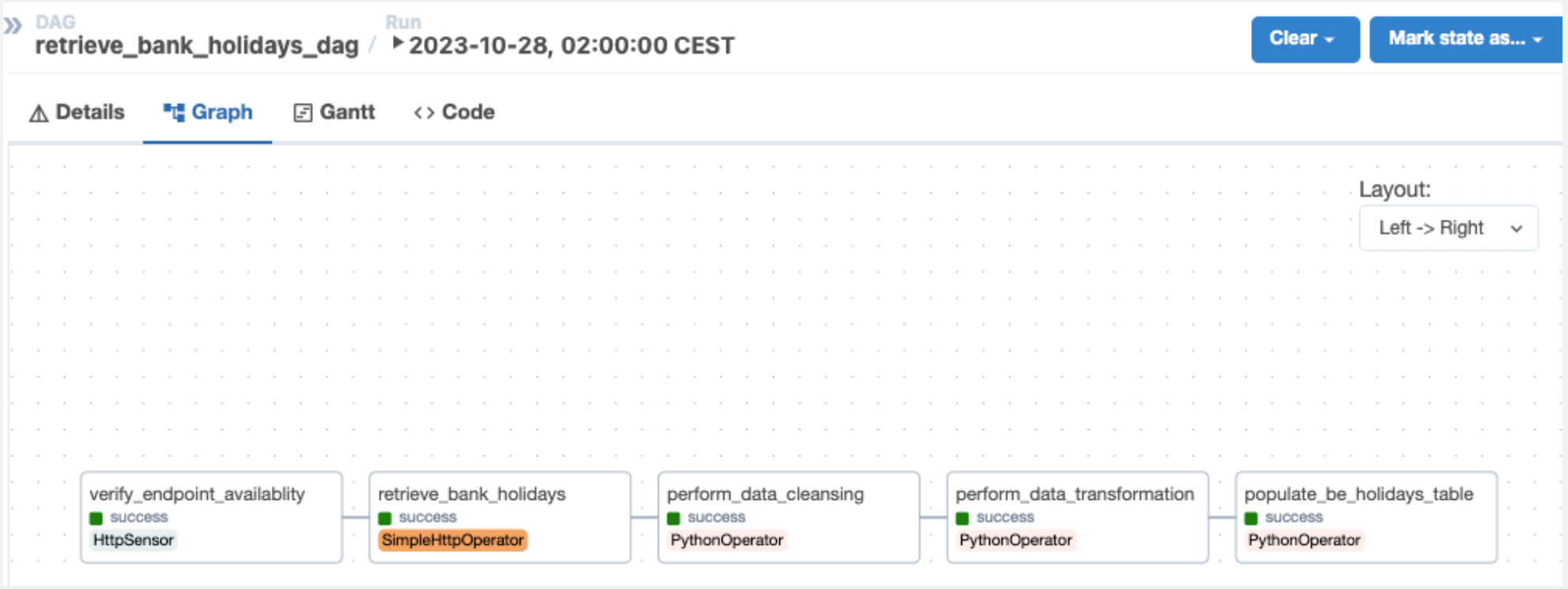
DAGS

The two data pipelines as shown in Airflow's DAGs GUI.



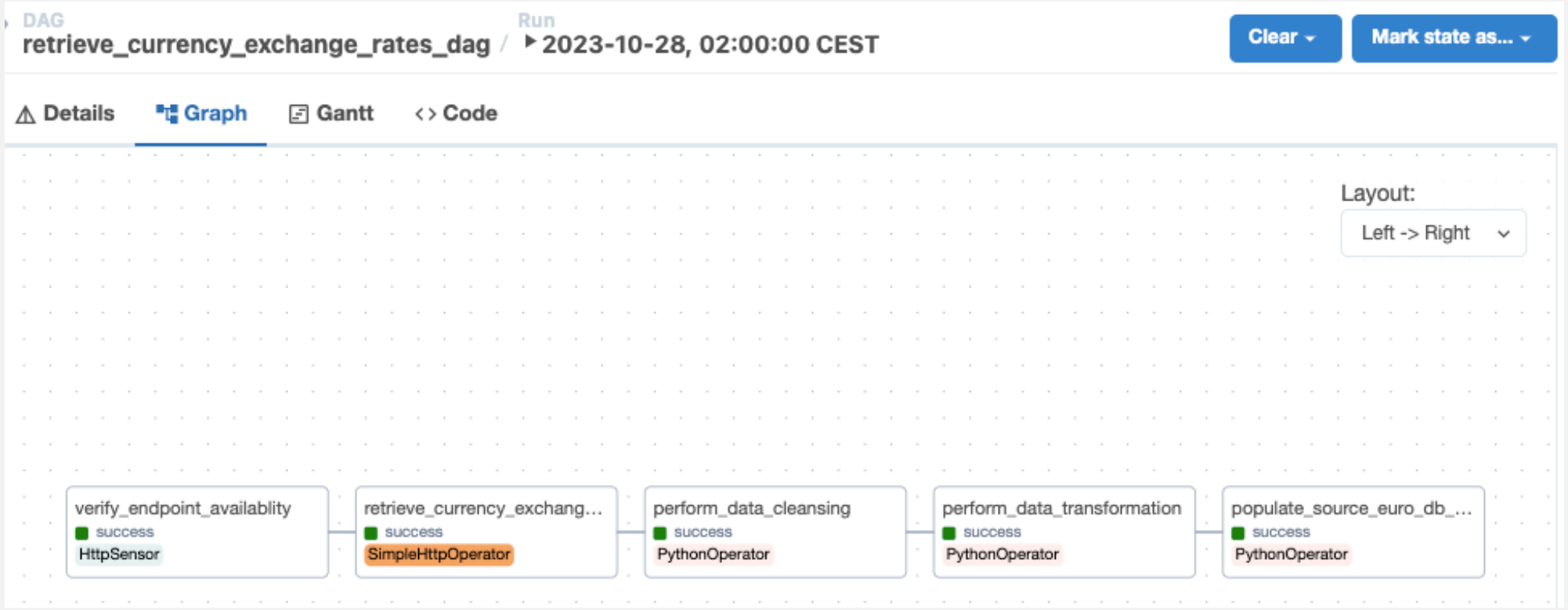
DAGS - GRAPHS

1. The structure of `retrieve_bank_holidays_dag` as illustrated in Airflow's DAGs GUI.



DAGS - GRAPHS

2. The structure of `retrieve_currency_exchange_rates_dag` as illustrated in Airflow's DAGs GUI.



SQLALCHEMY

6. Persistence of data to PostgreSQL database has been implemented based on a PythonOperator rather than a PostgresOperator.

7. Implementation is based on SQLAlchemy ORM as the library in question is natively supported by Airflow.

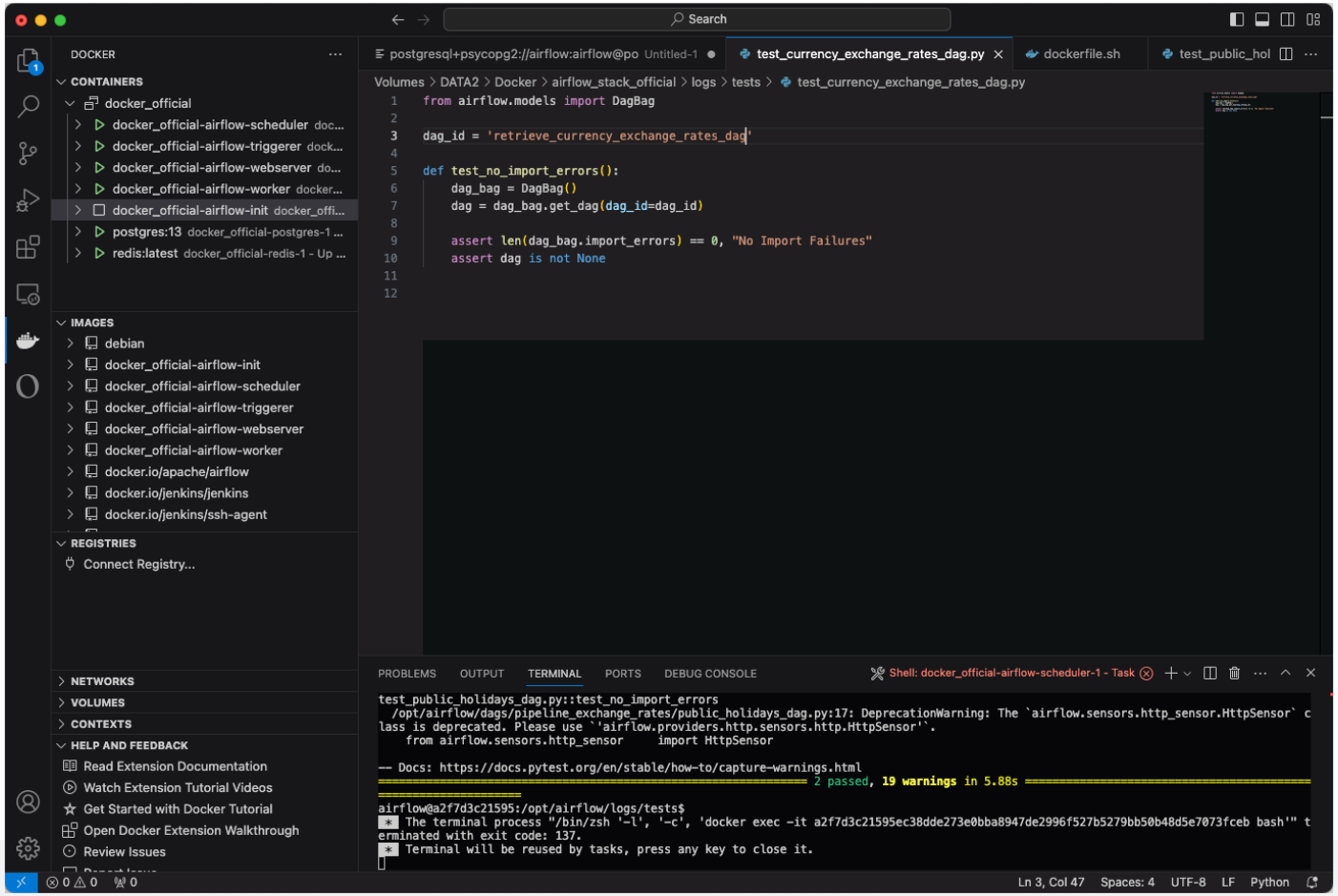
8. Implementation follows SQLAlchemy Declarative Mapping approach. Three classes were defined, shared between the two data pipelines and placed outside the DAG files as separate python modules/files. (ref: db_currency_exchange.py).

9. No relationships between the three table classes were defined in the absence of foreign key constraints.

UNIT TESTING

TESTING

1. Installation of pyTest library on airflow-webserver through *dockerfile* and *requirements.txt*
2. Initiation a terminal shell on airflow-webserver container by initiating a terminal session via VS code IDE and docker extension



TESTING

3. Only a single test was created that verifies the importation of libraries within a DAG.
4. Other tests will follow suite as highlighted in the future works section.

DELIVERABLES

DELIVERABLES

1. Kindly refer to *deliverables.xlsx*.
2. Deliverables reside in a designated code repository hosted on Git Hub.
3. URL to repo: https://github.com/tmpdeproject/assessment.git

SALIENT POINTS

SALIENT POINTS

Being relatively new to Apache Airflow, below are a few points worth mentioning during the undertaking of this task.

- Definition of a separate sets of environment configuration file sets for each dev/pre-prod/prod environment.
- Hardcoding of variables within a DAG file. Instead revert to airflow's variables, connection and machine's environment variables.
- Making use of *HTTPSensor* operator to verify the health of a REST API endpoint prior to the actual invocation as a separate task.
- Sensitive information should be securely stored in Airflow backend and retrieved via a unique connection id. If a fernet key is specified, passwords in the connection configuration and the variable configuration are encrypted at rest. Fernet is an implementation of symmetric authenticated cryptography.
- Never specify date time now() as a start date parameter for a task as it would never get executed.
- The transfer of data between tasks should occur via Airflow XCOM. However, in case of large messages such data should be placed on a shared file system, Azure BLOB or S3 repository.
- When designing an Airflow task, two important principles should be kept in mind: atomicity and idempotency.

FUTURE WORK ENHANCEMENTS

FUTURE WORK / ENHANCEMENTS

- Belgian public holidays DAG. Testing of existing database records prior to the insertion of database records.
- Possibly the inclusion of frameworks such as Great Expections to ensure data quality.
- Develop a comprehensive set of unit tests (pyTest scripts) aimed to check for errors in the actual DAG .py scripts and more importantly to verify the functionality and edge-cases of each data pipeline operator.
- Possibly the use of conditional task operators such as *BranchOperator*.
- Configuration of Jenkins and source code repository to support CI/CD.
- In general, a deep dive into Apache Airflow. Worth mentioning that a substantial amount of knowledge was gained over the past days.

THANK YOU