

Proposta de solução:

Criei containers docker contendo Airflow, Postgres e Jupyter Notebook

Foi criado uma DAG chamada **importar_dados.py** que primeiro executa uma task que realiza a leitura dos arquivos gerando um arquivo csv temporário. Depois outra task faz a leitura do csv e grava no banco postgres. Por fim o arquivo de dados original é renomeado adicionando um sufixo **.prc** e o csv é removido. Esta DAG está agendada para rodar a cada minuto, no Airflow não consegui agendar num intervalo menor. Pensei em criar repetidas tasks para poder executar a cada minutos usando sleep, mas não achei que seria uma proposta elegante. Pensei em usar FileSensor para detectar quando um arquivo novo é gravado na pasta, mas mesmo assim deveria criar um loop e nunca trabalhei com estes sensores para saber do comportamento.

Com o objetivo de separar a lógica de negócio da execução foi criado o script `/dag/scripts/carga.py` que realiza o processo de ETL

Com o objetivo de realizar teste unitário foi criado o teste `/dag/test/teste_carga.py`. Devido ao prazo foi implementado apenas um teste unitário.

Depois de o projeto rodando, coloquei o arquivo `Adult.test` na pasta, após o processamento coloquei o arquivo `Adult.data` para processamento.

Passo a passo

1 – Passo: Clonar o repositório e acessar a pasta

```
cd ~/git
```

```
git clone <URI>
```

```
cd ~/git/python-dev-test
```

2 – Passo: criar o arquivo **.env** com algumas configurações

```
echo -e "AIRFLOW_UID=$(id -u)" > docker/.env && echo -e "DATA_FILES=$(pwd)/data" >> docker/.env && echo -e "JUPYTER_TOKEN=12345678" >> docker/.env
```

```
docker/.env && echo -e "JUPYTER_TOKEN=12345678" >> docker/.env
```

A terminal window titled 'rodrigobrasil@pop-os: ~/git/python-dev-test' showing the execution of commands to create the .env file. The prompt is 'rodrigobrasil@pop-os:~/git/python-dev-test\$'. The command entered is 'echo -e "AIRFLOW_UID=\$(id -u)" > docker/.env && echo -e "DATA_FILES=\$(pwd)/data" >> docker/.env && echo -e "JUPYTER_TOKEN=12345678" >> docker/.env'. The output is 'rodrigobrasil@pop-os:~/git/python-dev-test\$' followed by a cursor. The terminal has a dark background with green and blue text for the prompt and command respectively. There are window control buttons (search, menu, close) in the top right corner.

```
rodrigobrasil@pop-os:~/git/python-dev-test$ echo -e "AIRFLOW_UID=$(id -u)" > doc
ker/.env && echo -e "DATA_FILES=$(pwd)/data" >> docker/.env && echo -e "JUPYTER_
TOKEN=12345678" >> docker/.env
rodrigobrasil@pop-os:~/git/python-dev-test$
```

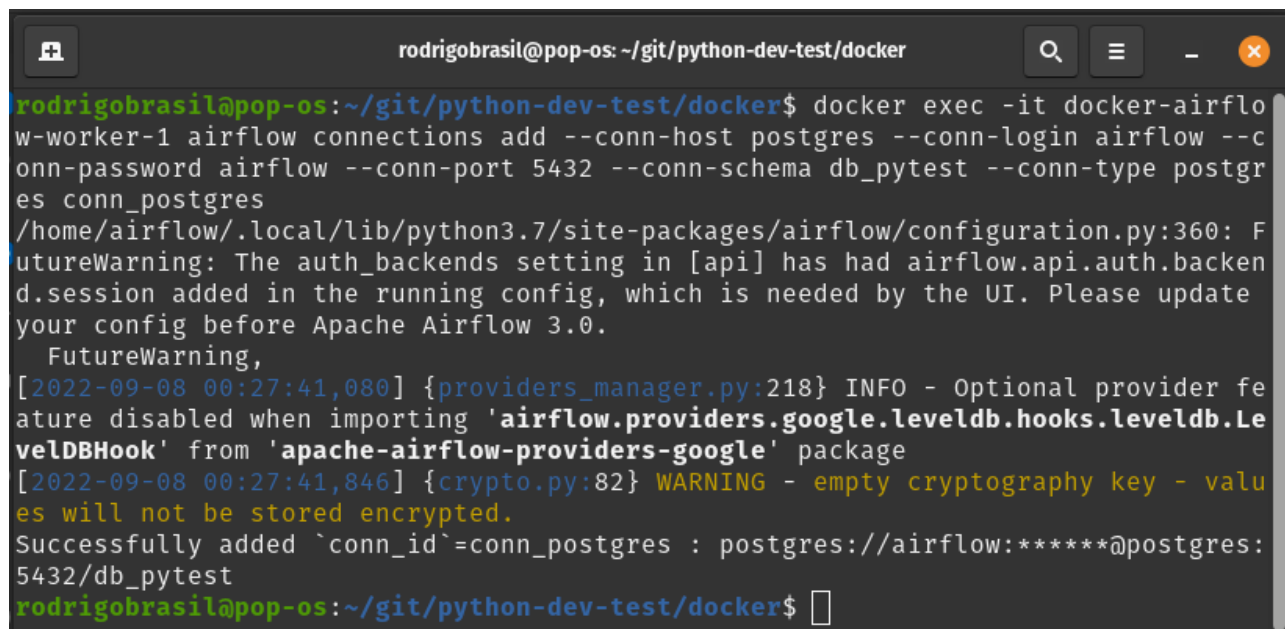
3 – Passo: acessar a pasta do container e executar o build da aplicação

```
cd docker
```

```
docker-compose up --build -d
```

4 – Passo: criar uma connection no airflow para a conexão com o banco de dados

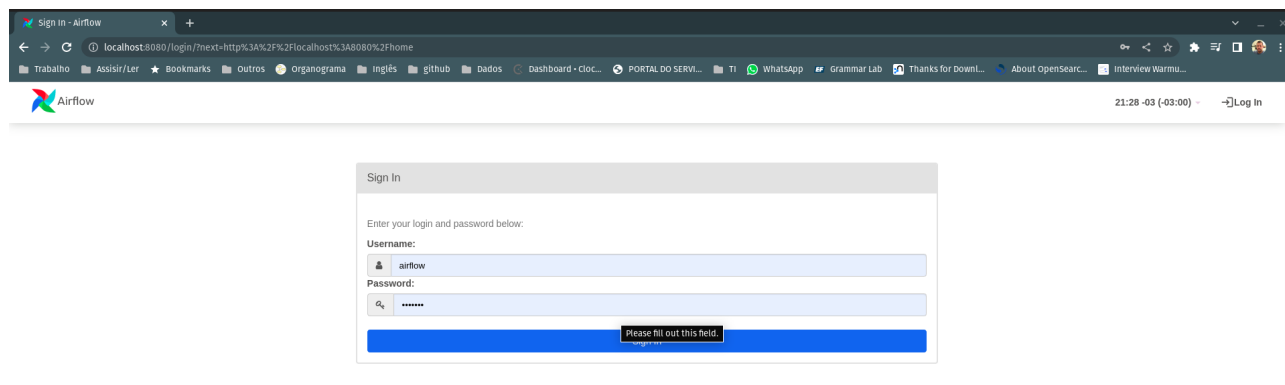
```
docker exec -it docker-airflow-worker-1 airflow connections add --conn-host postgres --conn-login airflow --conn-password airflow --conn-port 5432 --conn-schema db_pytest --conn-type postgres conn_postgres
```



```
rodrigobrasil@pop-os: ~/git/python-dev-test/docker
rodrigobrasil@pop-os:~/git/python-dev-test/docker$ docker exec -it docker-airflow-worker-1 airflow connections add --conn-host postgres --conn-login airflow --conn-password airflow --conn-port 5432 --conn-schema db_pytest --conn-type postgres conn_postgres
/home/airflow/.local/lib/python3.7/site-packages/airflow/configuration.py:360: FutureWarning: The auth_backends setting in [api] has had airflow.api.auth.backend.session added in the running config, which is needed by the UI. Please update your config before Apache Airflow 3.0.
  FutureWarning,
[2022-09-08 00:27:41,080] {providers_manager.py:218} INFO - Optional provider feature disabled when importing 'airflow.providers.google.leveldb.hooks.leveldb.LevelDBHook' from 'apache-airflow-providers-google' package
[2022-09-08 00:27:41,846] {crypto.py:82} WARNING - empty cryptography key - values will not be stored encrypted.
Successfully added `conn_id`=conn_postgres : postgres://airflow:*****@postgres:5432/db_pytest
rodrigobrasil@pop-os:~/git/python-dev-test/docker$
```

5 – Passo: Acessar o Airflow

abrir <http://localhost:8080> login airflow senha airflow



6– Passo: executar test unitário

`docker exec -it docker-airflow-worker-1 pytest /opt/airflow/dags/test/test_carga/test_carga.py`

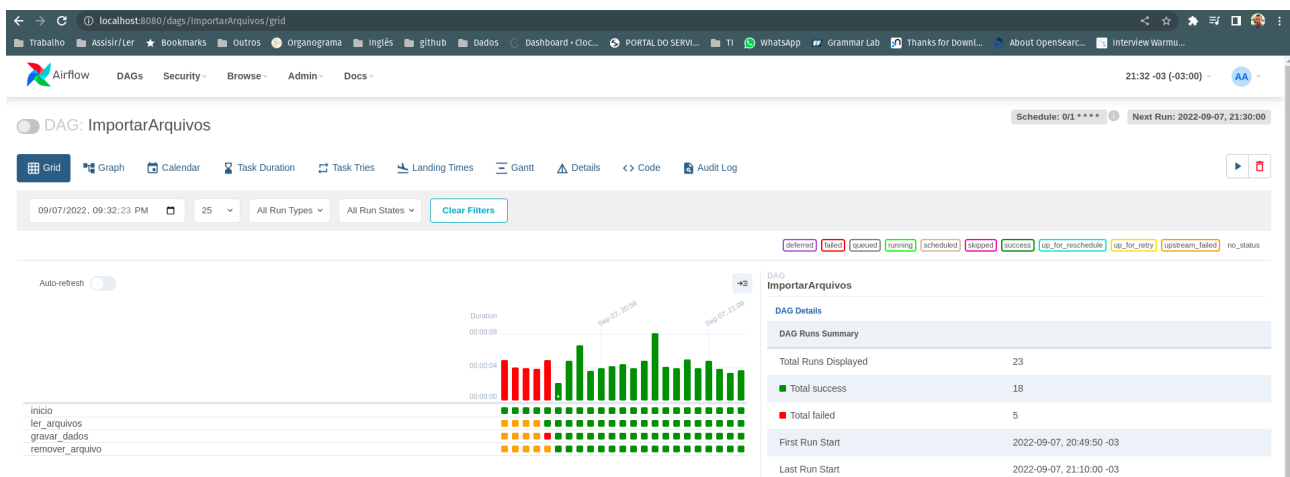
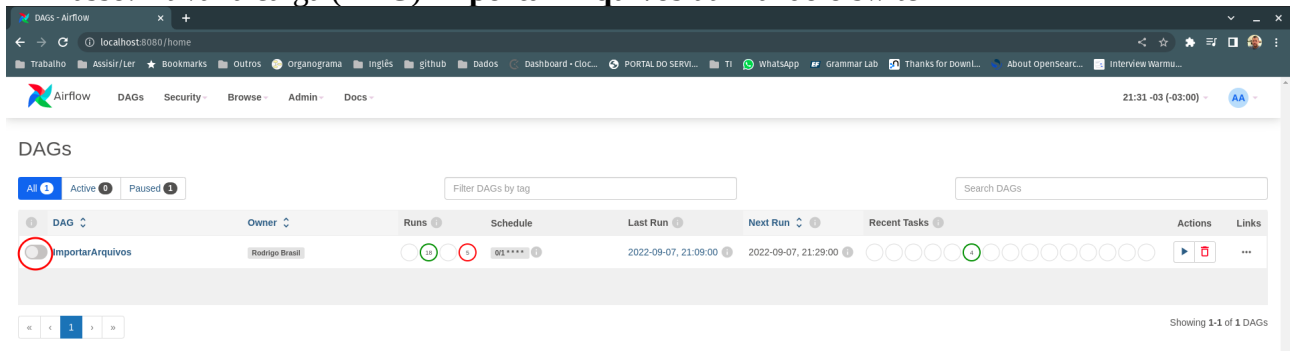
```
rodrigobrasil@pop-os: ~/git/python-dev-test/docker
rodrigobrasil@pop-os:~/git/python-dev-test/docker$ docker exec -it docker-airflow-worker-1 pytest /opt/airflow/dags/test/test_carga/test_carga.py
===== test session starts =====
platform linux -- Python 3.7.13, pytest-7.1.2, pluggy-1.0.0
rootdir: /opt/airflow
plugins: anyio-3.6.1
collected 1 item

dags/test/test_carga/test_carga.py . [100%]

===== warnings summary =====
../../home/airflow/.local/lib/python3.7/site-packages/airflow/configuration.py:360
FutureWarning: The auth_backends setting in [api] has had airflow.api.auth.backend.session added in the running config, which is needed by the UI. Please update your config before Apache Airflow 3.0.
FutureWarning,

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 1 passed, 1 warning in 0.73s =====
rodrigobrasil@pop-os:~/git/python-dev-test/docker$
```

7 – Passo: Ativar a carga (DAG) ImportarArquivos utilizando o switch



Sempre que um arquivo novo for disponibilizado na pasta python-dev-test/data com o Nome iniciando com Adult ele é processado.

Obs: Não apaguei os arquivos, após o processamento são renomeados para name.prc

8 – Passo: Acessar o jupyter notebook para conferir o resultado e abrir o arquivo **work/explorar.ipynb**

11 - Executar todos os passos