Comandos para hacer el scafold.

docker-compose build

docker compose up -d

1.1 Archivos

create\_bucket.py

# -> Crea el bucket de trabajo en S3

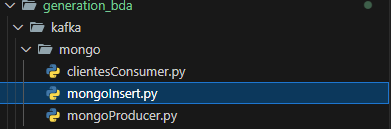
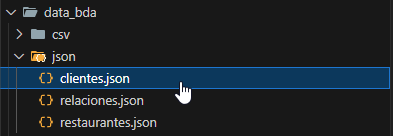
Meterse en el cluster

docker exec -it b17b7945d0d028dfc3b28a0892fb5ebbfd57e36d63ce2077ccf0b8f2e52dce65 /bin/bash

root@b17b7945d0d0:/opt/spark-apps/DataIntegration# python data\_integration.py

# -> Sube directamente los archivos al bucket habitaciones\_data.csv y hoteles.csv

2 MongoDB

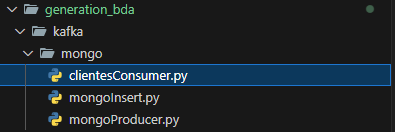
 

mongoInsert.py

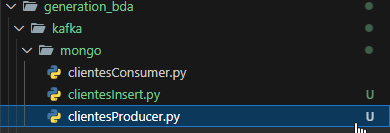
# -> inserta los registros en mongo provenientes de clientes.json

clientesConsumer.py

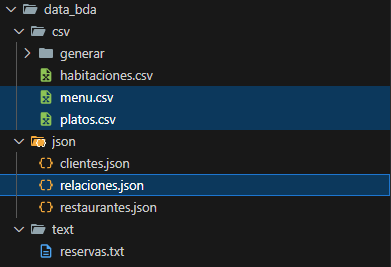
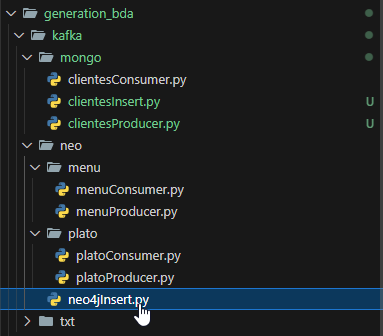
# -> Se pone a la escucha para cuando se le mande la información del Producer.



clientesProducer.py

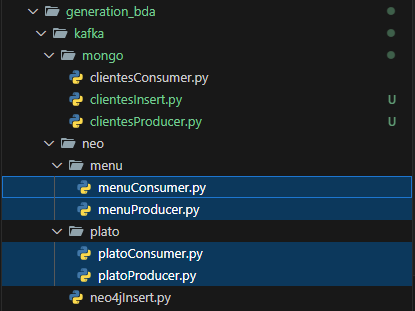
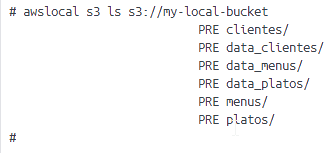
# -> Se encarga de leer desde mongo\_db y mandarla (mensaje estructurado) al Consumer. 

3 Neo4j



neo4jinsert.py

# -> inserta los registros en neo4j provenientes de menus.csv, platos.csv y relaciones.json

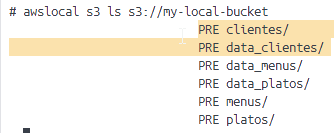
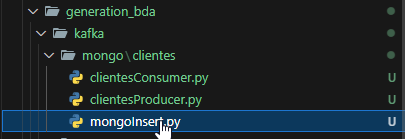
Los Producers se encargan de leer los archivos y mandarlos a los consumers.

Los Consumers se encargan de recibir los mensajes y crear en el bucket los archivos csv menus y platos.

Ya tenemos los archivos.

3.1 Postgres

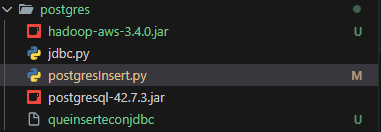
mongoInsert.py



# -> Leer los archivos y los inserta en mongo\_db

clientesConsumer.py

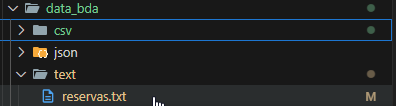
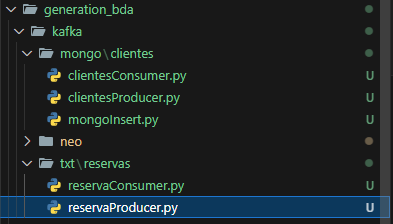
clientesProducer.py

3.1 Postgres empleados. csv y hoteles.csv

postgresinsert.py

# -> inserta en postgres empleados y hoteles

3.2 Kafka



Producer

# -> Lee el .txt y manda el mensaje al Consumer

Consumer

# \_> Lee el mensaje e inserta los registros en postgres.

Consultas con Spark:

Consulta1 Clientes.