ETL off a AWS SQS

Table of Contents

Summary:

This is Python's Command Line Interface (i.e., code) for receiving messages from AWS SQS through docker and writing the required data into the PostgreSQL database.

Necessary Materials Required:

In order to use the code, the following are needed:

1) Pycharm, VS code or any other Python environment including json, datetime, subprocess and psycopg2 libraries.

2) Docker:

a) https://docs.docker.com/get-docker/

After installing Docker successfully, run the following docker pull commands in your terminal or command prompt:

docker pull fetchdocker/data-takehome-postgres

docker pull fetchdocker/data-takehome-localstack

3) PostgreSQL:

https://www.postgresql.org/download/

4) AWS CLI:

pip install awscli-local

How to run containers on terminal?

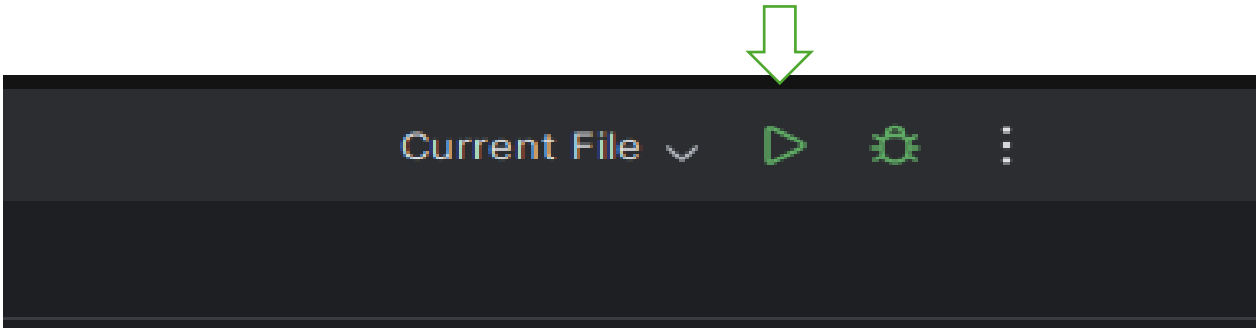
Commands for the terminal:

docker run --name localstack -d -p 4556:4556 fetchdocker/data-takehome-localstack

docker run --name postgres -d -p 5432:5432 fetchdocker/data-takehome-postgres

```
 docker run --name localstack -d -p 4556:4556 fetchdocker/data-takehome-localstack
1cb85b57050ee2697ea04704d95c983283df0b8875e0cbcf
 docker run --name postgres -d -p 5432:5432 fetchdocker/data-takehome-postgres
f9f005bb26e6cbdbc4c06975d88044b92b39de47f64f3ebb
```

Running the Python Code

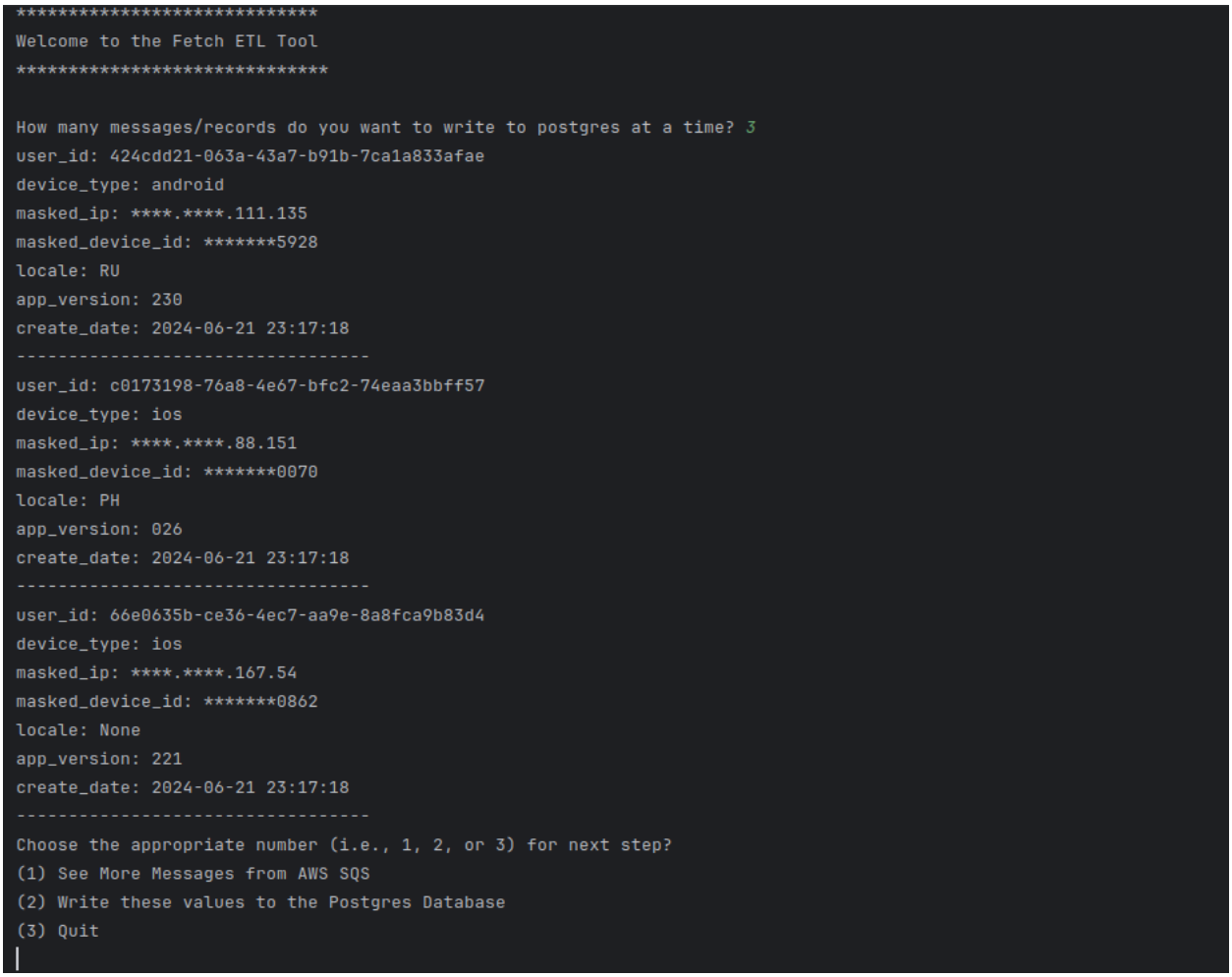



```
(.venv) adiach@adiach:~/PycharmProjects/read_aqs_queue$ python Fetch_ETL.py
*****************************
Welcome to the Fetch ETL Tool
*****************************

How many messages/records do you want to write to postgres at a time?
```

After running the code or typing 'python Fetch_ETL.py' in the terminal, it will show the output shown above with message-'Welcome to the Fetch ETL Tool'.

There will also be a question for you to choose how many messages do you want to process at the moment. After typing the number of messages, you will be shown the entries with the appropriate fields that can be written to the PostgreSQL database.



As shown in the screenshot above, it will output the following:

1) user_id
2) device_type
3) masked_ip
4) masked_device_id
5) locale

6) app_version

7) create_date (The time at which the entry was created)

After the data entries, you will then have the following options :

1) See More Messages from AWS SQS:

```
---------------------------------
user_id: 66e0635b-ce36-4ec7-aa9e-8a8fca9b83d4
device_type: ios
masked_ip: ****.****.167.54
masked_device_id: *******0862
locale: None
app_version: 221
create_date: 2024-06-21 23:23:56
---------------------------------
Choose the appropriate number (i.e., 1, 2, or 3) for next step?
(1) See More Messages from AWS SQS
(2) Write these values to the Postgres Database
(3) Quit
1

How many messages/records do you want to write to postgres at a time? 1
user_id: 181452ad-20c3-4e93-86ad-1934c9248903
device_type: android
masked_ip: ****.****.6.245
masked_device_id: *******3099
locale: ID
app_version: 096
create_date: 2024-06-21 23:23:59
---------------------------------
Choose the appropriate number (i.e., 1, 2, or 3) for next step?
(1) See More Messages from AWS SQS
(2) Write these values to the Postgres Database
(3) Quit
```

2) Write these values to the Postgres Database

```
How many messages/records do you want to write to postgres at a time? 1
user_id: 181452ad-20c3-4e93-86ad-1934c9248903
device_type: android
masked_ip: ****.****.6.245
masked_device_id: *******3099
locale: ID
app_version: 096
create_date: 2024-06-22 00:40:11
---
Choose the appropriate number (i.e., 1, 2, or 3) for next step?
(1) See More Messages from AWS SQS
(2) Write these values to the Postgres Database
(3) Quit
2
Successfully inserted 1 data entry records in the database.
Deleted message for user 181452ad-20c3-4e93-86ad-1934c9248903 from the SQS queue.

Process finished with exit code 0
```

3) Quit

```
How many messages/records do you want to write to postgres at a time? 1
user_id: 5bc74293-3ca1-4f34-bb89-523887d0cc2f
device_type: ios
masked_ip: ****.****.230.101
masked_device_id: *******2799
locale: PT
app_version: 228
create_date: 2024-06-22 00:50:35
---
Choose the appropriate number (i.e., 1, 2, or 3) for next step?
(1) See More Messages from AWS SQS
(2) Write these values to the Postgres Database
(3) Quit
3
*****************************
Thank you for using this tool
*****************************

Process finished with exit code 0
```

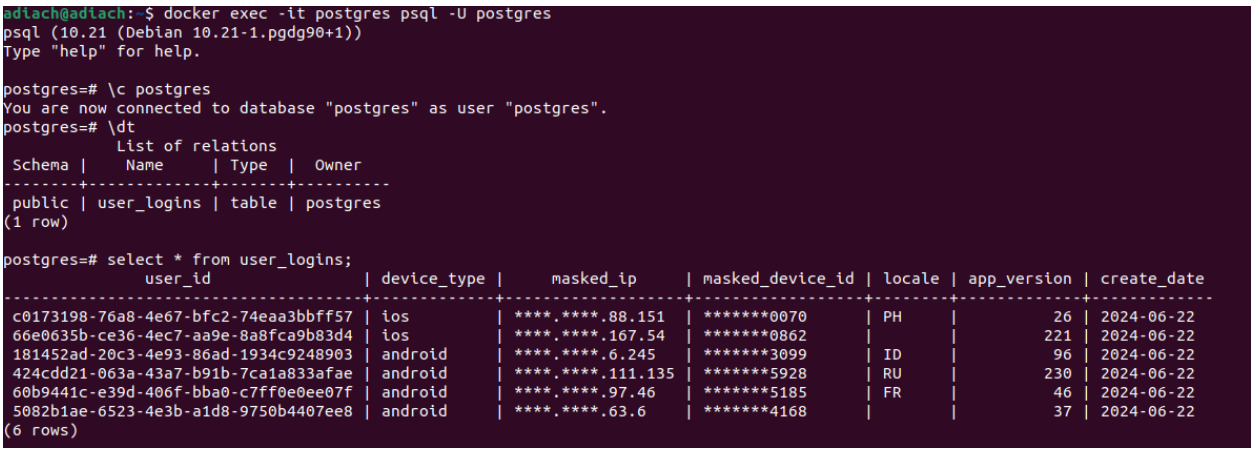
Now, in order to check the status of the database, go to the terminal and run the following:

1) docker exec -it postgres psql -U postgres

2) \c postgres

3) \dt

4) select * from user_logins;



```
adiach@adiach:~$ docker exec -it postgres psql -U postgres
psql (10.21 (Debian 10.21-1.pgdg90+1))
Type "help" for help.

postgres=# \c postgres
You are now connected to database "postgres" as user "postgres".
postgres=# \dt
           List of relations
 Schema |    Name     | Type  |  Owner
--------+-------------+-------+----------
 public | user_logins | table | postgres
(1 row)

postgres=# select * from user_logins;
               user_id               | device_type |     masked_ip     | masked_device_id | locale | app_version | create_date
-------------------------------------+-------------+-------------------+------------------+--------+-------------+-------------
 c0173198-76a8-4e67-bfc2-74eaa3bbff57 | ios         | ****.****.88.151  | *******0070      | PH     |          26 | 2024-06-22
 66e0635b-ce36-4ec7-aa9e-8a8fca9b83d4 | ios         | ****.****.167.54  | *******0862      |        |         221 | 2024-06-22
 181452ad-20c3-4e93-86ad-1934c9248903 | android     | ****.****.6.245   | *******3099      | ID     |          96 | 2024-06-22
 424cdd21-063a-43a7-b91b-7ca1a833afae | android     | ****.****.111.135 | *******5928      | RU     |         230 | 2024-06-22
 60b9441c-e39d-406f-bba0-c7ff0e0ee07f | android     | ****.****.97.46   | *******5185      | FR     |          46 | 2024-06-22
 5082b1ae-6523-4e3b-a1d8-9750b4407ee8 | android     | ****.****.63.6    | *******4168      |        |          37 | 2024-06-22
(6 rows)
```

Thought Process:

AWS SQS Queue has the messages stored in it that are of interest and we want to write it to the PostgreSQL database. The following is the summary of the thought process:

1) Run the docker commands to set up the postgres database and then see the database for the expected data values.

2) Run the docker commands to set up localstack and then see the contents of the queue through awslocal sqs commands.

3) The message has Id, receipt handle, MD5OfBody and Body. The body has the contents that we are interested in.

4) Write a code that connects to AWS SQS Queue locally using Localstack to fetch the information in each message. Using the subprocess library, run the same command as in terminal to fetch the message and its content

5) After connecting to the queue through the python code, see the output and bifurcate data according to its use case. Receipt handle and Body are important ones.

6) Body has contents that can be cleaned and ordered. After cleaning and ordering properly the output of the body, mask the IP and device ID such that it can be identified by an analyst for future

use. The app version was not in integer format and hence, decimals were removed from it for writing it to postgres to maintain the data integrity.

7) Now that the body's data is clean and usable, connect to the postgres database and now insert the values in the database.

8) After data records have been written/inserted to the database, use the Receipt Handle for deletion of messages as it ensures that each message is deleted only after the record has been written to the database.

9) Now, there are unique data records in the database. After all the values have been processed in the AWS SQS Queue and written to the database, there are no more messages to process.

Questions

How would you deploy this application in production?

For deployment, containerization tool such as Docker is a good idea. Packaging the application using Docker and thereafter, using Amazon ECS for managing and scaling these containers.

What other components would you want to add to make this production ready?

To make it production ready, we can implement logging system for troubleshooting and monitoring such that we can be alerted of the bugs or threats. Security checks to ensure all the incoming data is safe and does not violate the data constraints and maintains integrity. Identity and Access Management tools to ensure the information is being accessed by appropriate user.

How can this application scale with a growing dataset?

We can modify the application to handle multiple messages concurrently for parallel processing. Bifurcating dataset based on work load. For frequently accessed data, implement a better caching layer and for very large datasets, switch to batch processing model.

How can PII be recovered later on?

Replace masking with encryption for PII as well as a secure method of decrypting the PII, implement a robust system for managing and securing these encryption keys. Also, IAM roles as to who can access this information and under what circumstances.

What are the assumptions you made?

The assumptions that I made are:

1) The data volume is predictable as to how much of the information is frequently needed and how much can be used for batch processing

2) The data has a specific schema and constraints.

3) Data usage is bifurcated appropriately in order to introduce IAM roles for administrators as well as consumers

4) Application in production uses simple scaling and security techniques.