AWS SQS Message Buffer Demonstration (Asynchronous Processing)

Project Overview :

I created a small SQS demo showing how messages are buffered when the consumer is down or unavailable and processed later when the customer is available.

KEY CONCEPTS COVERED :

*Asynchronous processing - The sender does not wait for the receiver to finish its work.
*Decoupling - Producer and consumer does not DEPEND on each other to be available.
*Queue depth visibility
*Buffering - Messages are stored temporarily when consumer is down or unavailable
*Reliable message delivery of atleast once
*Pushing and Polling

Project Architecture:

Steps Followed :

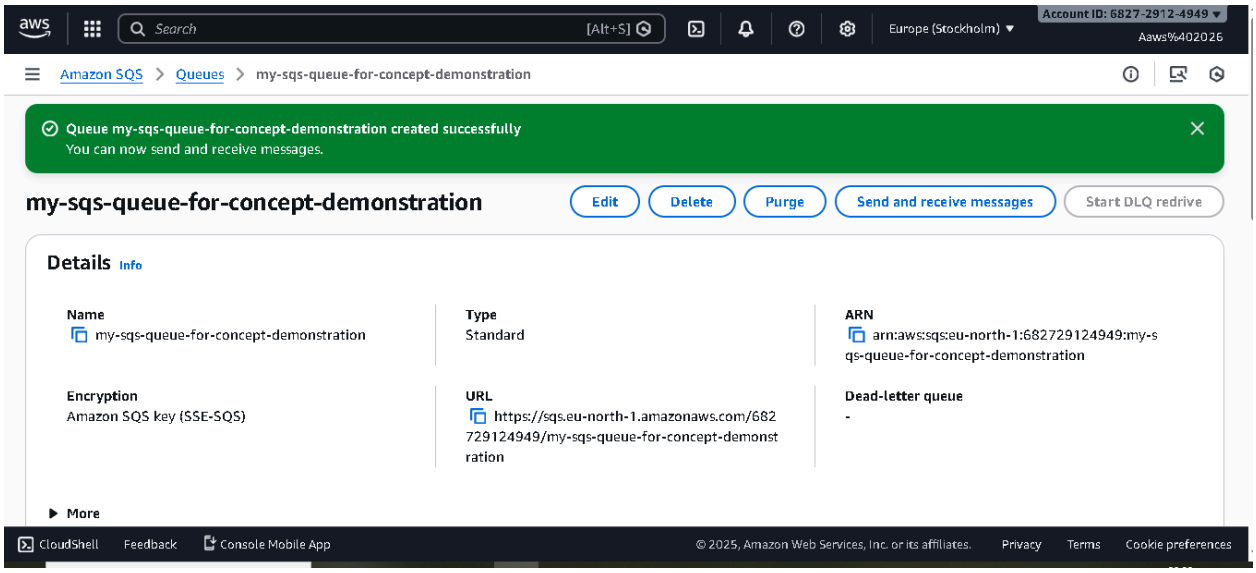
1. Created SQS Queue (AWS Console)

AWS Console → SQS--> Create queue

1.a.2 Choose Settings

Queue type: Standard

Queue name:sqs-basic
Leave everything else as default → Click Create queue



2.Copy Queue URL

After creation:

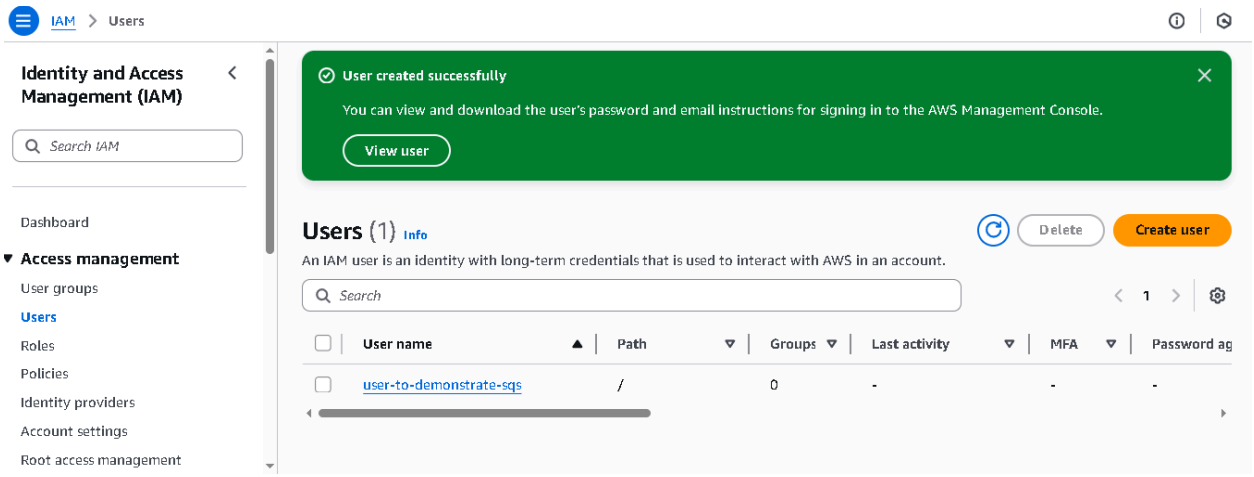
Open the queue --> Copy Queue URL (Save it somewhere)

eg URL: https://sqs.eu-north-1.amazonaws.com/682729124949/my-queue-for-demo

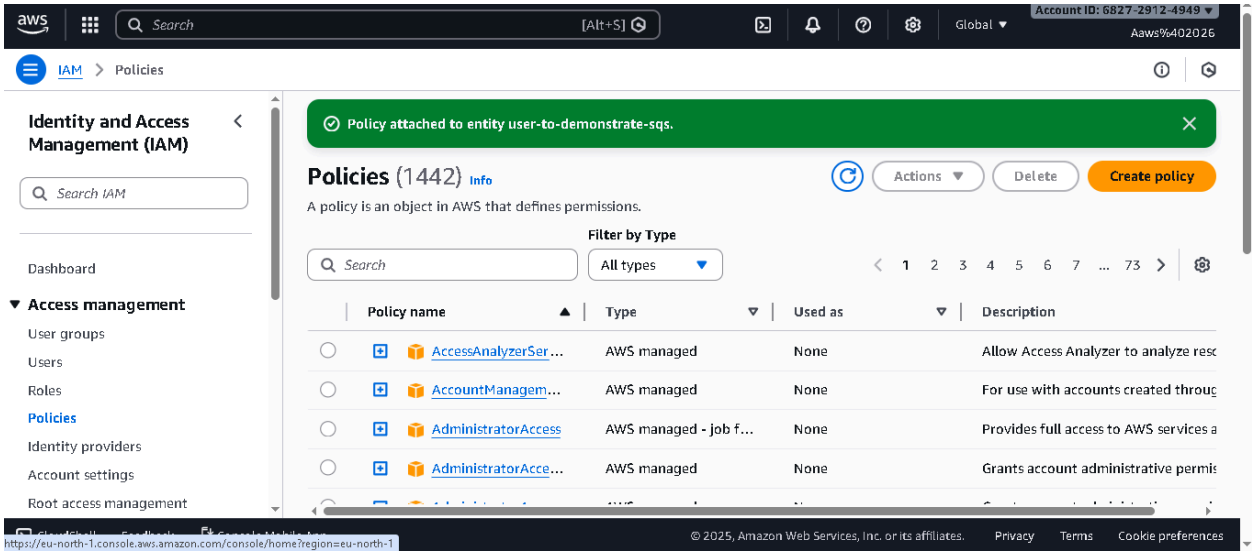
```
QUEUE URL

https://sqs.eu-north-1.amazonaws.com/682729124949/my-sqs-queue-for-concept-demonstration
```

3: IAM Permissions (code needs permission to use SQS)

IAM → Users → create User



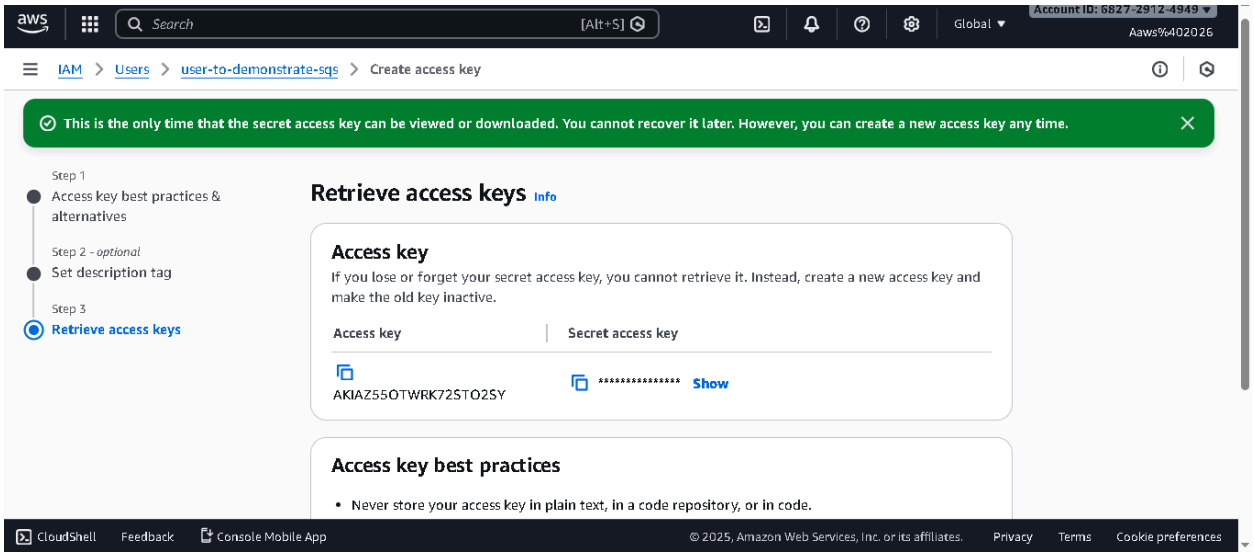
created user -> Attach policy: AmazonSQSFullAccess



3.a. Access key creation for the user credentials as we are going to use the local machine

created user -> Security credentials--> Create access key--> Choose Command Line Interface (CLI)

Download CSV file that contains Access key Id and Secret Access Key ID



4: Prepared Local Environment(prepares the local machine to talk to AWS securely, so it is not ec2 or aws console)

open the command prompt of the local machine

to install python library

type python and follow the installation steps

4.1 Install boto3(it lets python talk to aws services like SQS)

pip install boto3

to install AWS CLI in Windows command prompt

use the Windows Package Manager (winget) or a direct download command with PowerShell.
Option 1: Using Windows Package Manager (winget)

winget install Amazon.AWSCLI

5: Configured AWS CLI in local machine terminal to connect to the SQS

aws configure

Enter:
AWS Access Key ID : <your access key>

AWS Secret Access Key : <your secret key>
Default region name : region name in sqs url
Default output format : json
📌 This stores credentials in: ~/.aws/credentials

6: Created producer python file in windows terminal

notepad producerss.py

type the following code in the notepad file created above

```
import boto3

sqs = boto3.client('sqs', region_name='ap-south-1')

queue_url = "your url"
response = sqs.send_message(
    QueueUrl=queue_url,
    MessageBody="Hello from Producer"
)

print("Message sent to SQS")
```

run the created file
python (filename.py)

```
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lenovo>python --version
Python 3.13.9

C:\Users\lenovo>pip install boto3
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: boto3 in c:\users\lenovo\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (1.42.12)
Requirement already satisfied: botocore<1.43.0,>=1.42.12 in c:\users\lenovo\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from boto3) (1.42.12)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in c:\users\lenovo\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from boto3) (1.0.1)
Requirement already satisfied: s3transfer<0.17.0,>=0.16.0 in c:\users\lenovo\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from boto3) (0.16.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\lenovo\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from botocore<1.43.0,>=1.42.12->boto3) (2.9.0.post0)
Requirement already satisfied: urllib3!=2.2.0,<3,>=1.25.4 in c:\users\lenovo\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from botocore<1.43.0,>=1.42.12->boto3) (2.6.2)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.43.0,>=1.42.12->boto3) (1.17.0)

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: C:\Users\lenovo\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip

C:\Users\lenovo>aws configure
AWS Access Key ID [****************G52B]: AKIAZ55OTWRK72STO2SY
AWS Secret Access Key [****************qXqw]: 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
Default region name [ap-south-1]: eu-north-1
Default output format [json]: json

C:\Users\lenovo>aws sqs list-queues
{
    "QueueUrls": [
        "https://sqs.eu-north-1.amazonaws.com/682729124949/my-sqs-queue-for-concept-demonstration"
    ]
}
```

7: Verify AWS Access (Test)
Run:

aws sqs list-queues

✓ If queues are listed → SUCCESS
❌ If error → permission or region issue

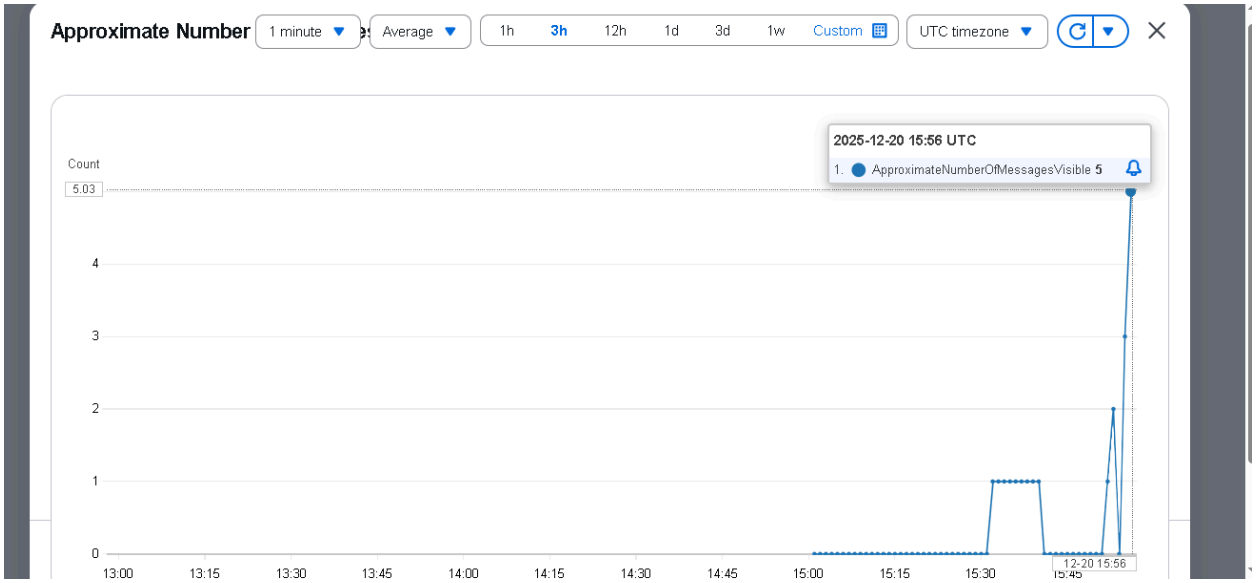
This proves: Your laptop can now talk to AWS SQS.

Internally Python calls AWS SQS, Message is stored in queue, Producer exits immediately

8 : Verify in AWS Console

Queue-->Click Send and receive messages(send about 5 messages)-->Click Poll for messages

You'll see your message ✓ in the LOCAL MACHINE
(check the monitoring tab in the created queue and see the number of messages visible group graph)



9:: Consumer Code (Receive Message)
Create consumer.py

```
import boto3
import time

sqs = boto3.client('sqs', region_name='ap-south-1')
queue_url = "PASTE_YOUR_QUEUE_URL_HERE"

while True:
```

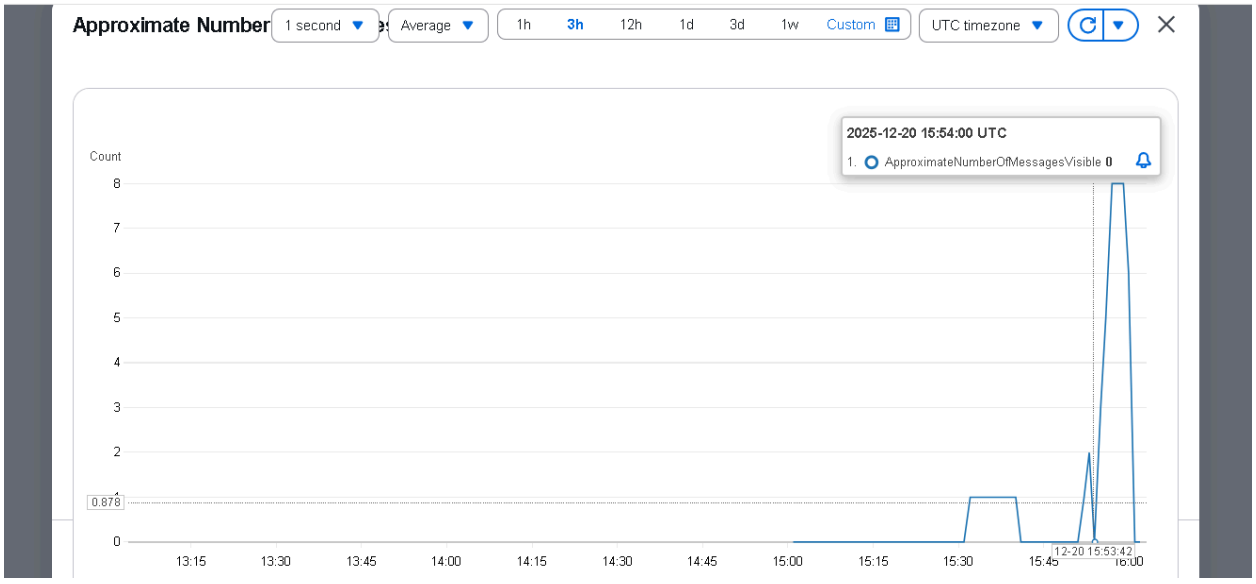
```
    response = sqs.receive_message(
        QueueUrl=queue_url,
        MaxNumberOfMessages=1,
        WaitTimeSeconds=10
    )

    messages = response.get('Messages', [])

    for message in messages:
        print("Message received:", message['Body'])

        sqs.delete_message(
            QueueUrl=queue_url,
            ReceiptHandle=message['ReceiptHandle']
        )

    time.sleep(2)
```

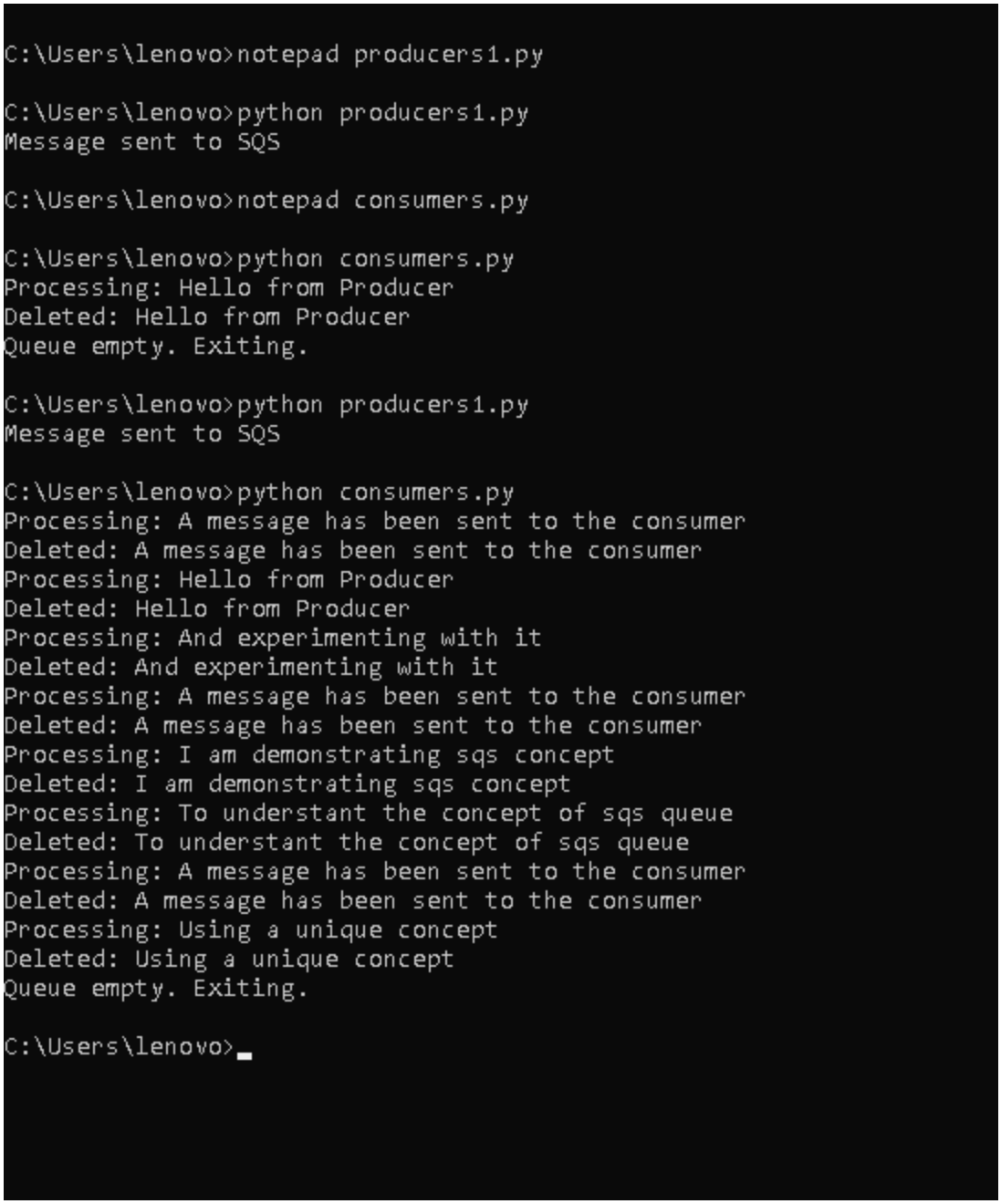
Run Consumer
python consumer.py

Internally Consumer polls SQS, Message becomes invisible (visibility timeout), Consumer processes message, Message is deleted
Queue becomes empty

📌 This proves at-least-once delivery

(check the monitoring tab in the created queue and see the number of messages visible group graph which will be 0)





Key Learnings :

- AWS SQS enables asynchronous processing and service decoupling
- Messages are safely buffered when consumers are unavailable
- Consumers can process messages later using at-least-once delivery

Result :

Messages were successfully sent to AWS SQS and stored even when the consumer was not running.
Once the consumer started, all messages were processed without loss, proving reliable asynchronous message handling.