Project in AWS
Practice Lab

Triggering AWS Lambda from Amazon SQS

Andra-Diana Popescu

2025

ABOUT THIS LAB

In this hands-on AWS lab, you will learn how to trigger a Lambda function using SQS. This Lambda function will process messages from the SQS queue and insert the message data as records into a DynamoDB table.

LEARNING OBJECTIVES

- Create the Lambda Function
- Create the SQS Trigger
- Copy the Source Code into the Lambda Function
- Log In to the EC2 Instance and Test the Script
- Confirm Messages Were Inserted into the DynamoDB Table

AWS Documentation about Lambda and SQS:

https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

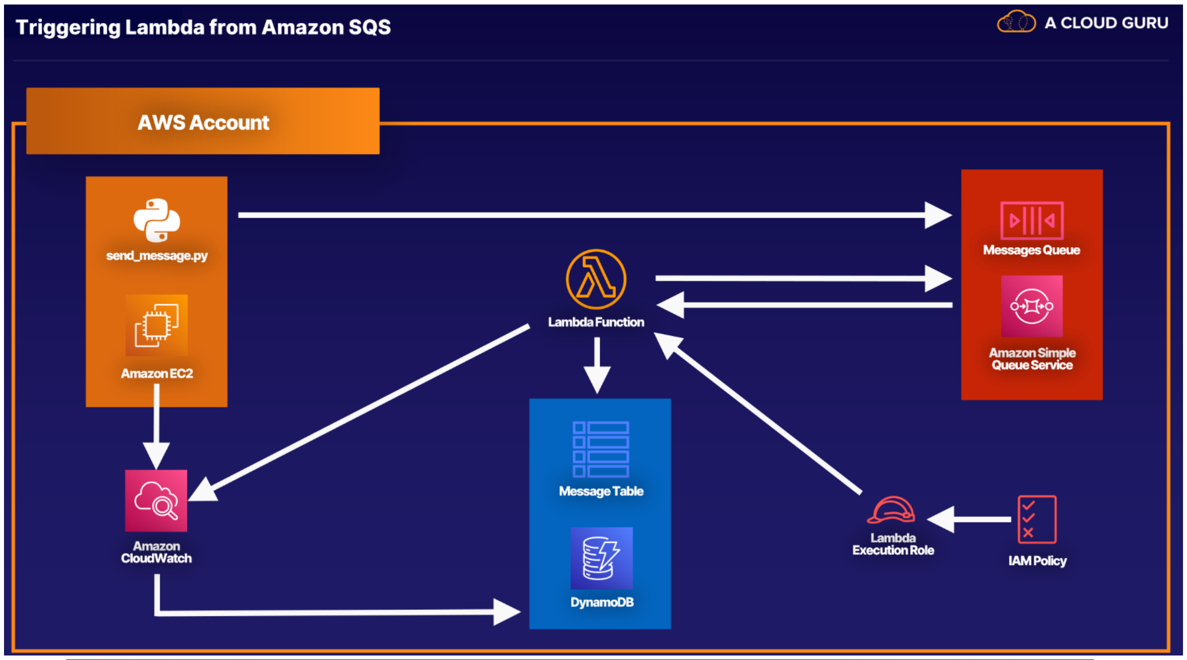
https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html

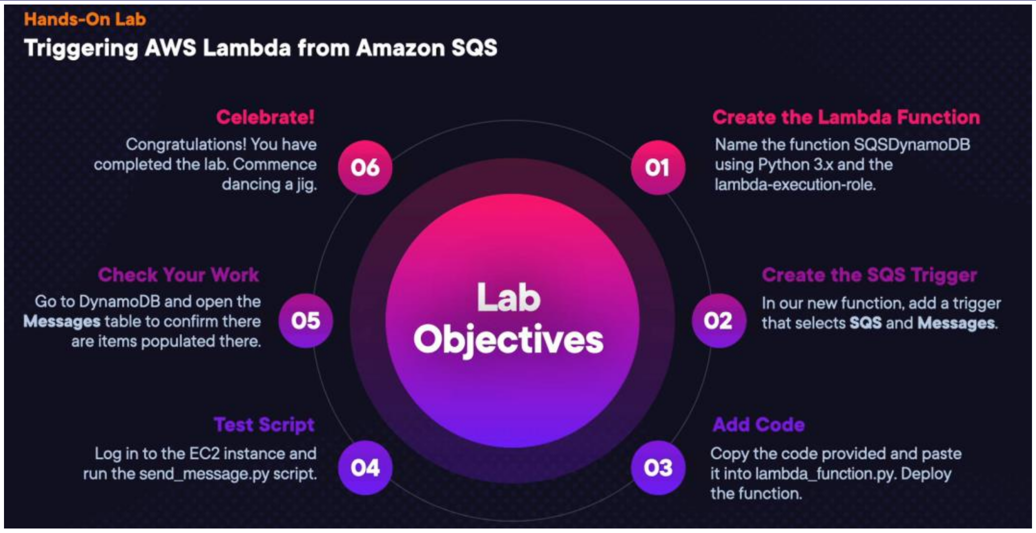
https://aws.amazon.com/sqs/faqs/#topic-0

Source: https://learn.acloud.guru/course/certified-solutions-architect-associate/

Table of Contents

Lab Diagrams




Hands-On Lab

Triggering AWS Lambda from Amazon SQS

Lab Objectives

01 **Create the Lambda Function**
Name the function SQSDynamoDB using Python 3.x and the lambda-execution-role.

02 **Create the SQS Trigger**
In our new function, add a trigger that selects **SQS** and **Messages**.

03 **Add Code**
Copy the code provided and paste it into lambda_function.py. Deploy the function.

04 **Test Script**
Log in to the EC2 instance and run the send_message.py script.

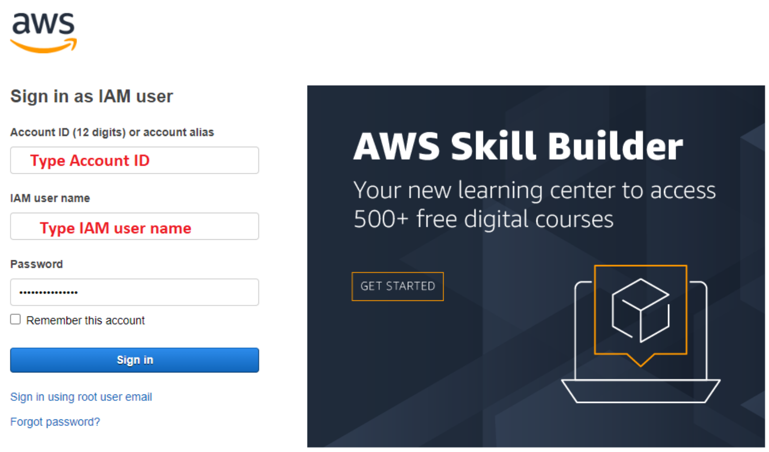
05 **Check Your Work**
Go to DynamoDB and open the **Messages** table to confirm there are items populated there.

06 **Celebrate!**
Congratulations! You have completed the lab. Commence dancing a jig.

We have the AWS account in **us-east-1** Region. In this lab, you are a solutions architect that has been tasked with improving how messages are handled in your environment. The solution should be event-driven and require minimal human interactions.

This lab will walk you through utilizing a message queue in SQS that is already created to trigger Lambda to send message data to your DynamoDB table. Your EC2 instance will simulate messages going into the queue using a Python script that's already on it. Both your EC2 instance and your Lambda function will be sending metrics and logs to CloudWatch throughout the lab. Permissions necessary to allow SQS, Lambda, and DynamoDB to talk to one another have already been set up as well.

Log in to your AWS account


Sign in as IAM user

Account ID (12 digits) or account alias

Type Account ID

IAM user name

Type IAM user name

Password

Remember this account

Sign in

Sign in using root user email

Forgot password?

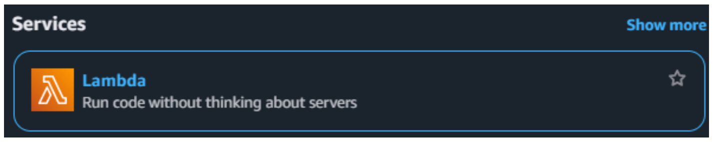
AWS Skill Builder

Your new learning center to access 500+ free digital courses

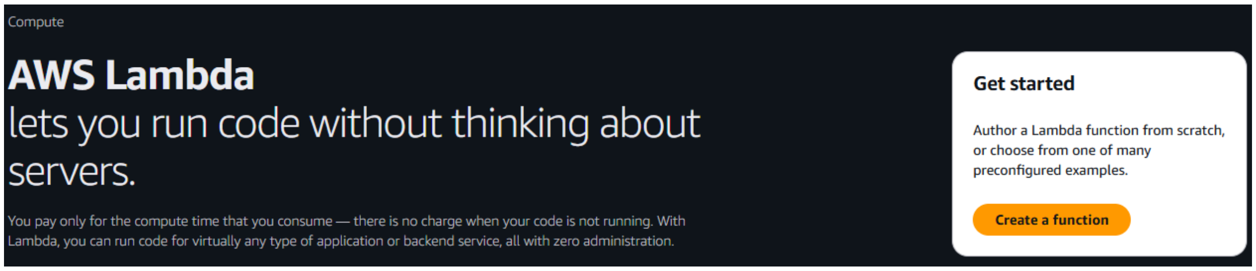
GET STARTED

1. Create the Lambda Function

1. Once you are logged into the AWS Management Console, navigate to **Lambda**.



2. Click the **Create function** button.


Compute

AWS Lambda lets you run code without thinking about servers.

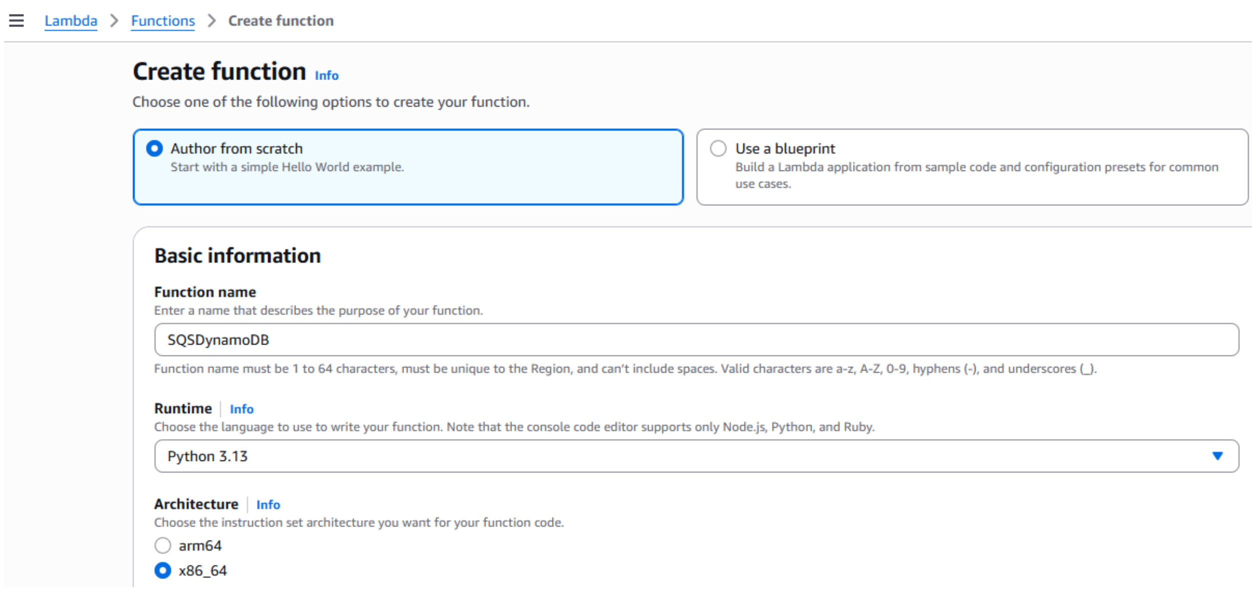
You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

Get started

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

Create a function

3. On the **Create function** page, select **Author from scratch**.
4. Under **Basic Information**, set the following parameters for each field:
   a. ***Function name***: Enter ***SQSDynamoDB***.
   b. ***Runtime***: Select ***Python 3.13*** from the dropdown menu.
   c. ***Architecture***: Select ***x86_64***.



☰ Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

[✓ Author from scratch] Start with a simple Hello World example.

Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.

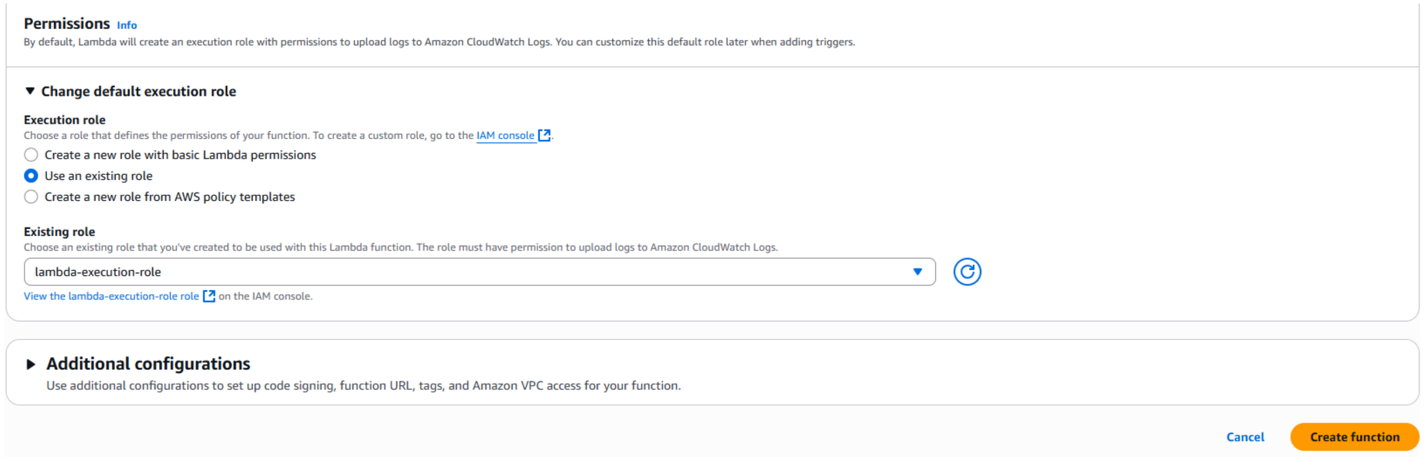
Basic information

Function name
Enter a name that describes the purpose of your function.
SQSDynamoDB
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.13

Architecture Info
Choose the instruction set architecture you want for your function code.
arm64 [✓ x86_64]

5. Under **Permissions**, expand **Change default execution role**.
6. Select **Use an existing role**.
7. Under **Existing role**, select **lambda-execution-role** from the dropdown menu.
8. Click the **Create function** button.



Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

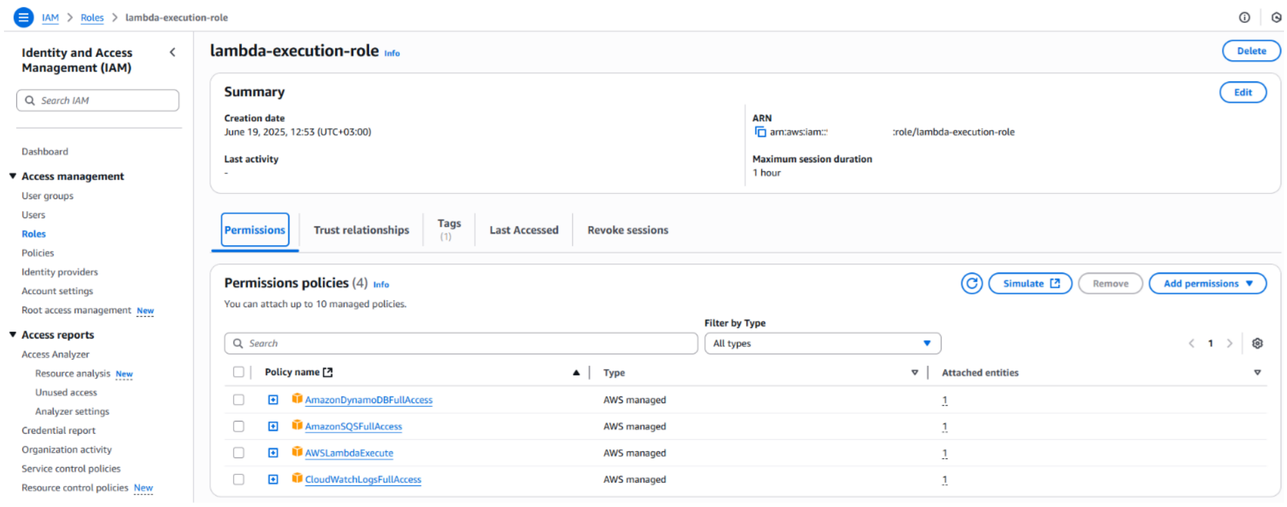
▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
[✓ Use an existing role]
Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
lambda-execution-role
View the lambda-execution-role role on the IAM console.

▶ **Additional configurations**
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

Note: Below are the permissions for **lambda-execution-role**.


IAM > Roles > lambda-execution-role

Identity and Access Management (IAM)

Search IAM

Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
Organization activity
Service control policies
Resource control policies New

lambda-execution-role Info

Delete

Summary

Edit

Creation date
June 19, 2025, 12:53 (UTC+03:00)

ARN
arn:aws:iam::   :role/lambda-execution-role

Last activity
-

Maximum session duration
1 hour

Permissions | Trust relationships | Tags (1) | Last Accessed | Revoke sessions

Permissions policies (4) Info

Simulate | Remove | Add permissions

You can attach up to 10 managed policies.

Filter by Type

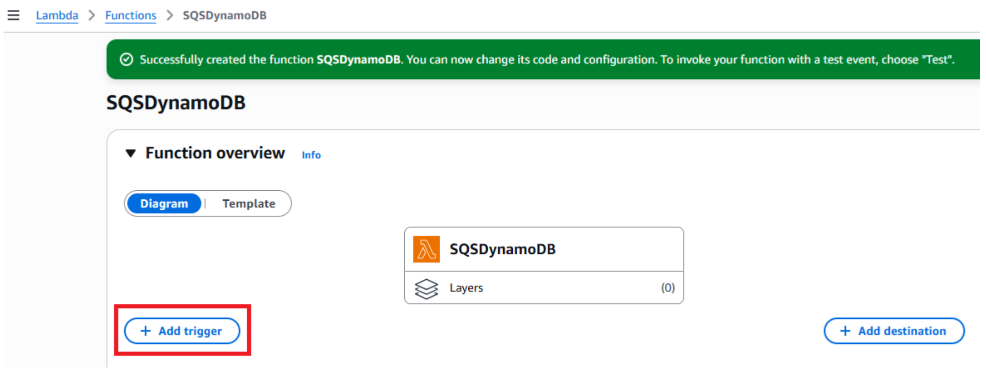
Search | All types

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
AmazonSQSFullAccess	AWS managed	1
AWSLambdaExecute	AWS managed	1
CloudWatchLogsFullAccess	AWS managed	1

Note: Now we need to link our Lambda function to SQS.

2. Create the SQS Trigger

1. Click the **+ Add trigger** button.


Lambda > Functions > SQSDynamoDB

Successfully created the function **SQSDynamoDB**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

SQSDynamoDB

Function overview Info

Diagram | Template

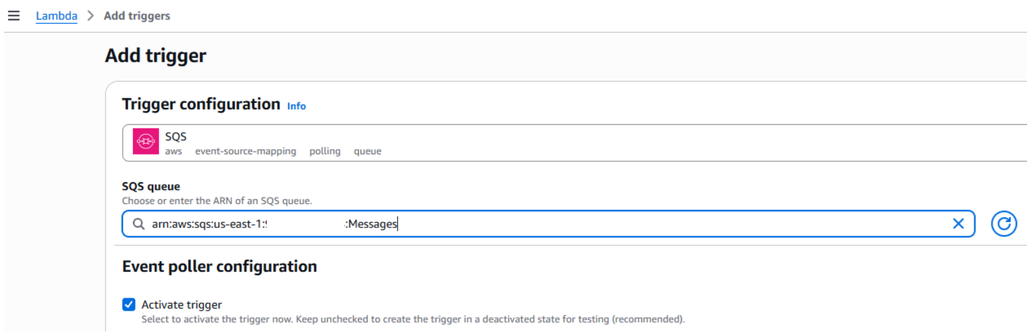
SQSDynamoDB

Layers (0)

+ Add trigger

+ Add destination

2. Under **Trigger configuration**, click the **Select a source** dropdown menu.
3. From the menu, select **SQS**.
4. Under **SQS queue**, click the search bar and select **Messages**.
5. Ensure that the checkbox next to **Activate trigger** is checked. Keep the defaults.


Lambda > Add triggers

Add trigger

Trigger configuration Info

SQS
aws event-source-mapping polling queue

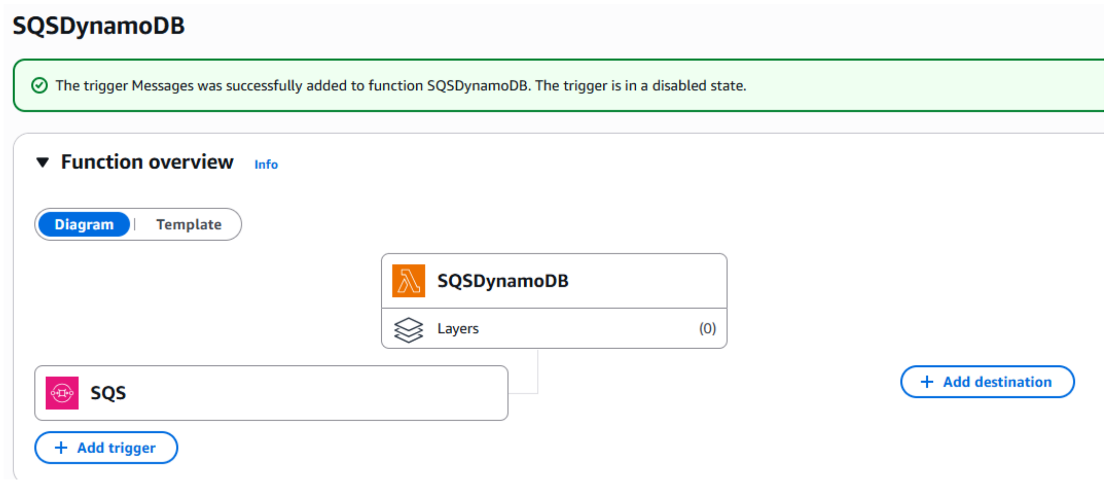
SQS queue
Choose or enter the ARN of an SQS queue.

arn:aws:sqs:us-east-1:!   :Messages

Event poller configuration

☑ Activate trigger
Select to activate the trigger now. Keep unchecked to create the trigger in a deactivated state for testing (recommended).

6. Click **Add**.

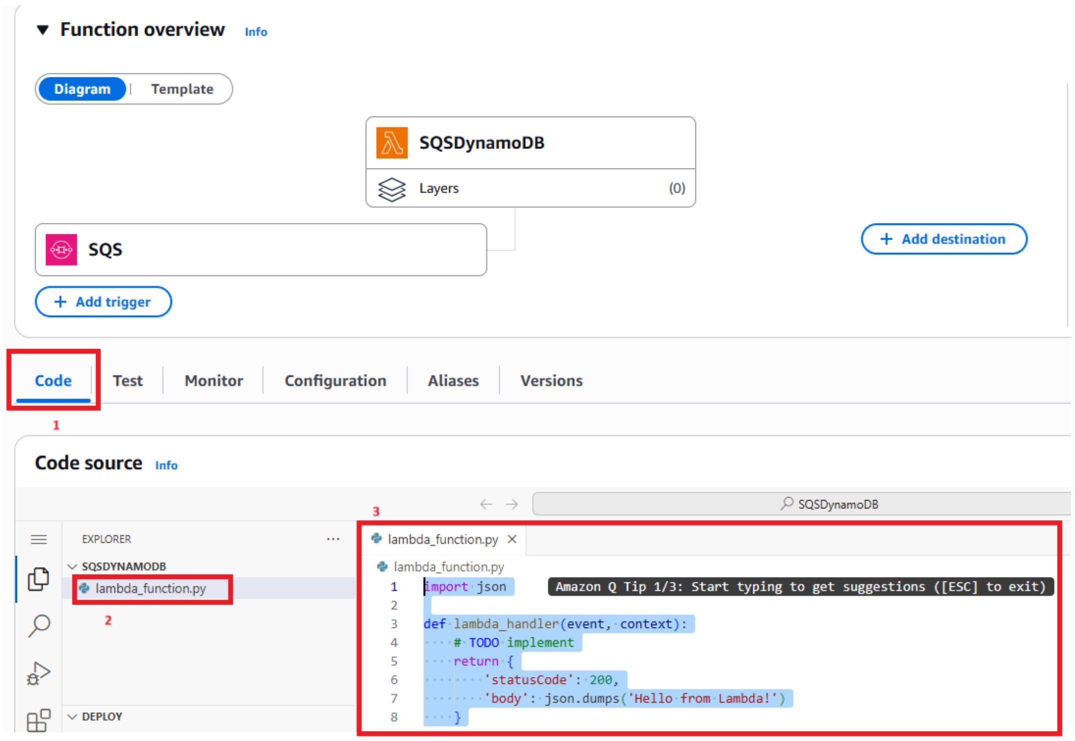


The trigger Messages was successfully added to function SQSDynamoDB. The trigger is in a disabled state.

Note: Now we're ready to configure our function.

3. Copy the Source Code into the Lambda Function

1. Under the **+ Add trigger** button, click the **Code** tab.
2. On the left side, double-click on **lambda_function.py**.
3. Delete the contents of the function.

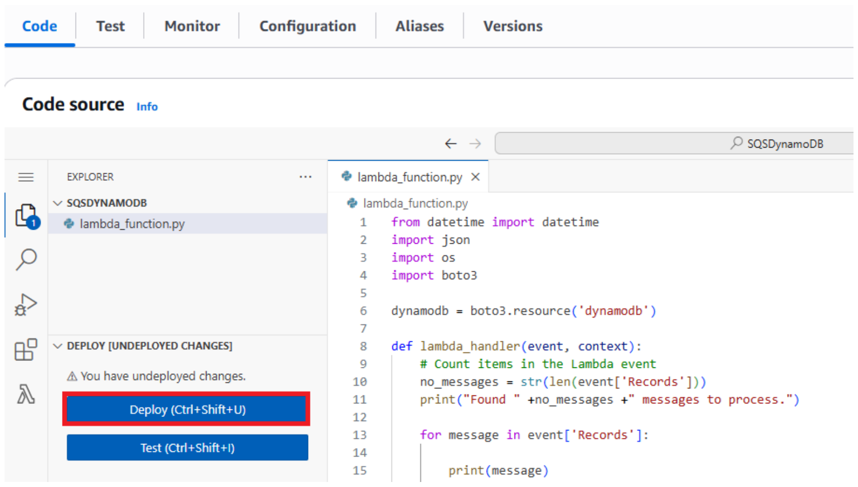


```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

4. The source code for **lambda_function.py**:

```
from datetime import datetime
import json
import os
import boto3

dynamodb = boto3.resource('dynamodb')

def lambda_handler(event, context):
    # Count items in the Lambda event
    no_messages = str(len(event['Records']))
    print("Found " +no_messages +" messages to process.")

    for message in event['Records']:

        print(message)

        # Write message to DynamoDB
        table = dynamodb.Table('Message')

        response = table.put_item(
            Item={
                'MessageId': message['messageId'],
                'Body': message['body'],
                'Timestamp': datetime.now().isoformat()
            }
        )
        print("Wrote message to DynamoDB:", json.dumps(response))
```

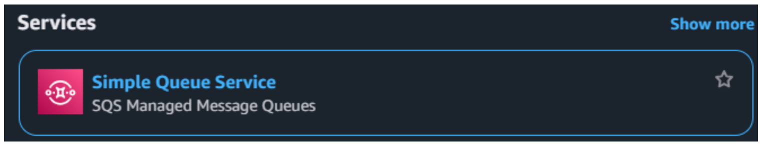
5. Copy the code. This code uses an API to let us write to a DynamoDB table. The table that it's writing to, it's named "Message", and the information that the item is actually writing is the "MessageID", "Body", "Timestamp".
6. Return to the AWS console and paste the code into the **lambda_function.py** code box.
7. Click the **Deploy** button.



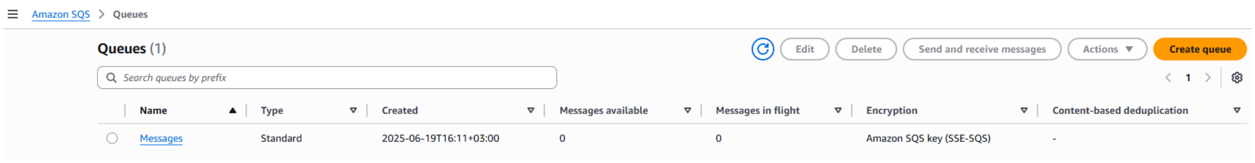
```
from datetime import datetime
import json
import os
import boto3

dynamodb = boto3.resource('dynamodb')

def lambda_handler(event, context):
    # Count items in the Lambda event
    no_messages = str(len(event['Records']))
    print("Found " +no_messages +" messages to process.")

    for message in event['Records']:

        print(message)
```

4. Log In to the EC2 Instance and Test the Script

1. In the search bar on top of the AWS console, enter ***sqs***.
2. From the search results, select **Simple Queue Service**.



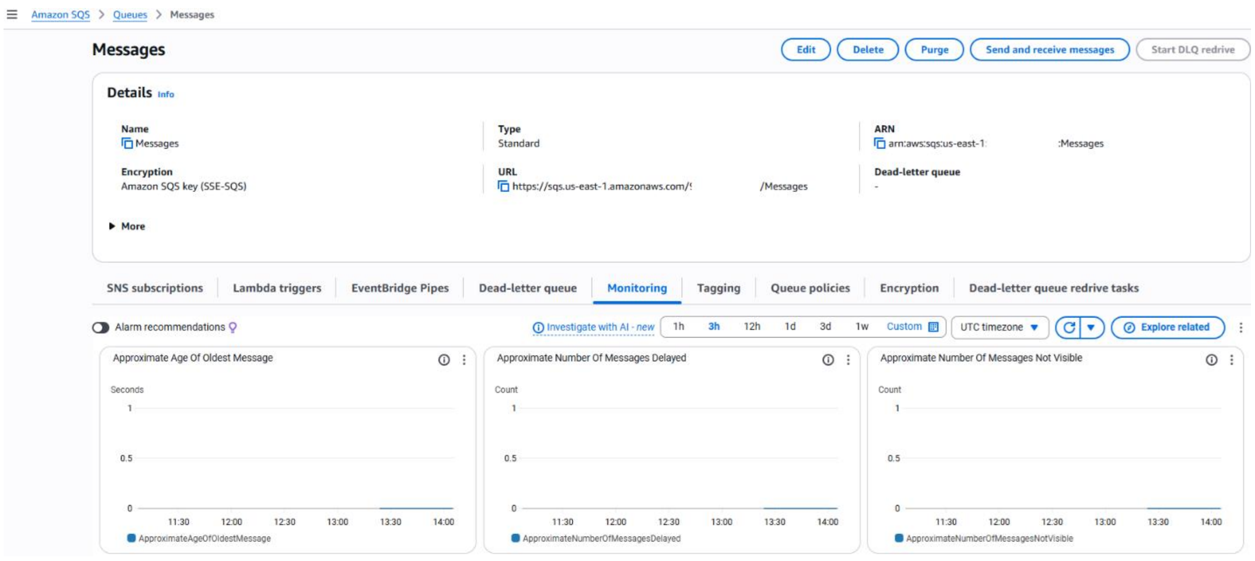
3. Click **Messages**.



Queues (1)

Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
Messages	Standard	2025-06-19T16:11+03:00	0	0	Amazon SQS key (SSE-SQS)	-

4. Click the **Monitoring** tab to monitor our SQS messages. Keep this window open.

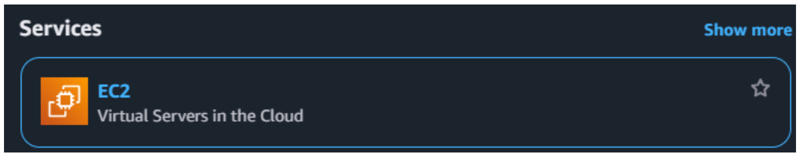

Amazon SQS > Queues > Messages
Edit
Delete
Purge
Send and receive messages
Start DLQ redrive
More
SNS subscriptions
Lambda triggers
EventBridge Pipes
Dead-letter queue
Monitoring
Tagging
Queue policies
Encryption
Dead-letter queue redrive tasks
Alarm recommendations
Investigate with AI - new
1h
3h
12h
1d
3d
1w
Custom
UTC timezone
Explore related
Approximate Age Of Oldest Message
Seconds
ApproximateAgeOfOldestMessage
Approximate Number Of Messages Delayed
Count
ApproximateNumberOfMessagesDelayed
Approximate Number Of Messages Not Visible
Count
ApproximateNumberOfMessagesNotVisible

Messages

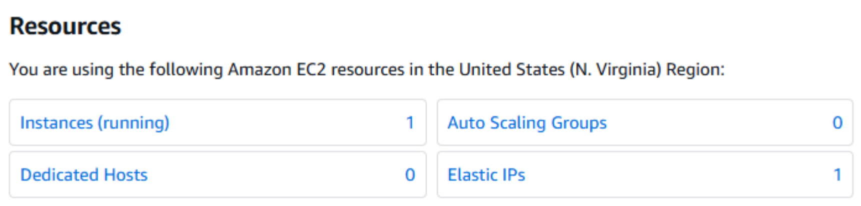
Details Info

Name	Type	ARN
Messages	Standard	arn:aws:sqs:us-east-1: :Messages
Encryption	**URL**	**Dead-letter queue**
Amazon SQS key (SSE-SQS)	https://sqs.us-east-1.amazonaws.com/ /Messages	-

5. In the search bar on top, enter ***ec2***.
6. From the search results, select **EC2** and open it in a new browser tab or window. We will start our script to simulate our messages coming through.



7. Under **Resources**, click **Instances (running)**.

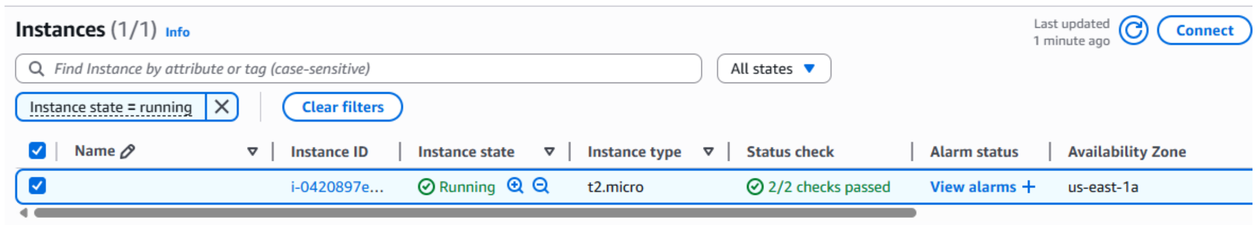


Resources

You are using the following Amazon EC2 resources in the United States (N. Virginia) Region:

Instances (running)	1	Auto Scaling Groups	0
Dedicated Hosts	0	Elastic IPs	1

8. In the existing instance available, click the checkbox next to its name.
9. Click the **Connect** button at the top.

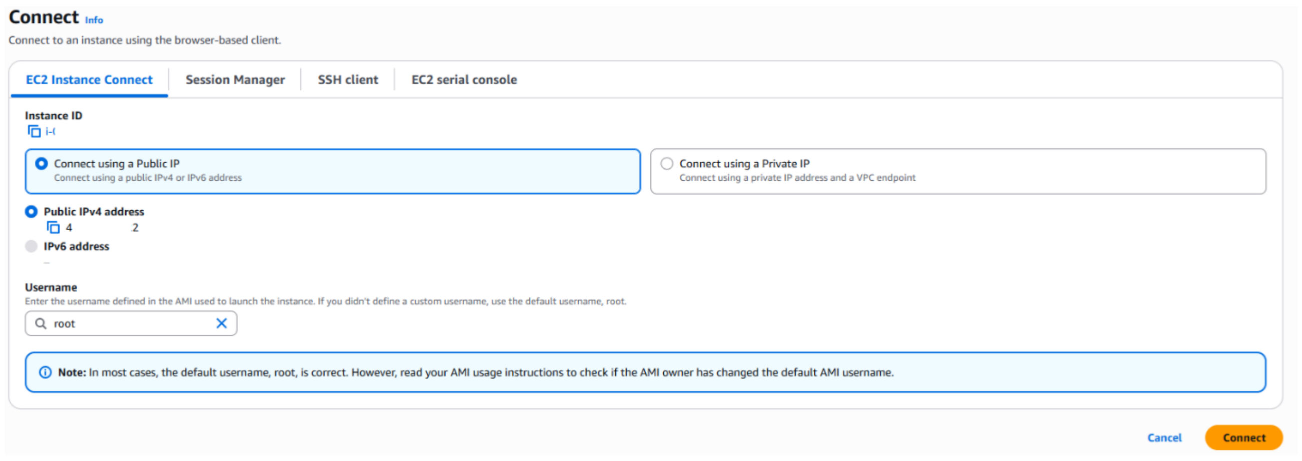


Instances (1/1) Info

Last updated 1 minute ago

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☑		i-0420897e...	✓ Running	t2.micro	✓ 2/2 checks passed	View alarms +	us-east-1a

10. Click **Connect** at the bottom to open a shell and access the command line.



Connect Info

Connect to an instance using the browser-based client.

Instance ID

i-[redacted]

● Connect using a Public IP
Connect using a public IPv4 or IPv6 address

○ Connect using a Private IP
Connect using a private IP address and a VPC endpoint

● Public IPv4 address
4[redacted]2

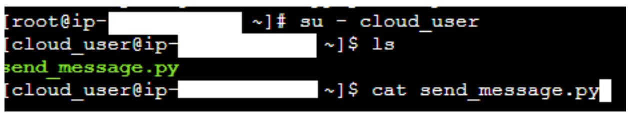
○ IPv6 address
–

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, root.

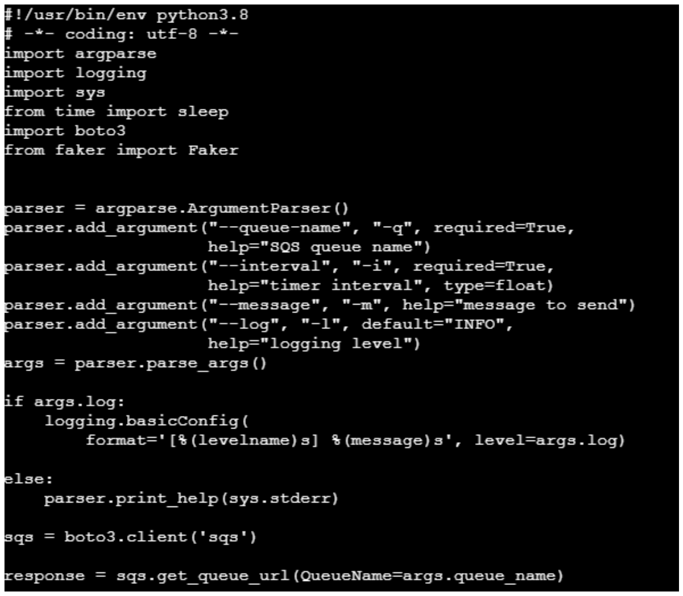
root

Note: In most cases, the default username, root, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

11. In the shell, become the **cloud_user** role: ***su - cloud_user***
12. View a list of files available to you: ***ls***
13. View the contents of the **send_message.py** file: ***cat send_message.py***



```
[root@ip-[redacted] ~]# su - cloud_user
[cloud_user@ip-[redacted] ~]$ ls
send_message.py
[cloud_user@ip-[redacted] ~]$ cat send_message.py
```



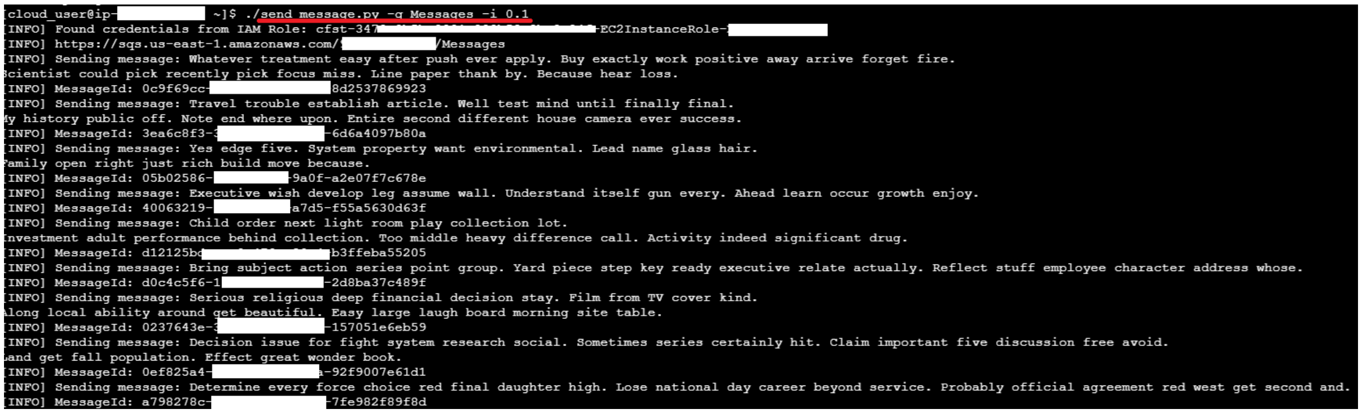
```
#!/usr/bin/env python3.8
# -*- coding: utf-8 -*-
import argparse
import logging
import sys
from time import sleep
import boto3
from faker import Faker


parser = argparse.ArgumentParser()
parser.add_argument("--queue-name", "-q", required=True,
                    help="SQS queue name")
parser.add_argument("--interval", "-i", required=True,
                    help="timer interval", type=float)
parser.add_argument("--message", "-m", help="message to send")
parser.add_argument("--log", "-l", default="INFO",
                    help="logging level")
args = parser.parse_args()

if args.log:
    logging.basicConfig(
        format='[%(levelname)s] %(message)s', level=args.log)

else:
    parser.print_help(sys.stderr)

sqs = boto3.client('sqs')

response = sqs.get_queue_url(QueueName=args.queue_name)
```

```
#!/usr/bin/env python3.8
# -*- coding: utf-8 -*-
import argparse
import logging
import sys
from time import sleep
import boto3
from faker import Faker


parser = argparse.ArgumentParser()
parser.add_argument("--queue-name", "-q", required=True,
                    help="SQS queue name")
parser.add_argument("--interval", "-i", required=True,
                    help="timer interval", type=float)
parser.add_argument("--message", "-m", help="message to send")
parser.add_argument("--log", "-l", default="INFO",
                    help="logging level")
args = parser.parse_args()

if args.log:
    logging.basicConfig(
        format='[%(levelname)s] %(message)s', level=args.log)

else:
    parser.print_help(sys.stderr)

sqs = boto3.client('sqs')

response = sqs.get_queue_url(QueueName=args.queue_name)
```

```
queue_url = response['QueueUrl']

logging.info(queue_url)

while True:
    message = args.message
    if not args.message:
        fake = Faker()
        message = fake.text()

    logging.info('Sending message: ' + message)

    response = sqs.send_message(
        QueueUrl=queue_url, MessageBody=message)

    logging.info('MessageId: ' + response['MessageId'])
    sleep(args.interval)
```

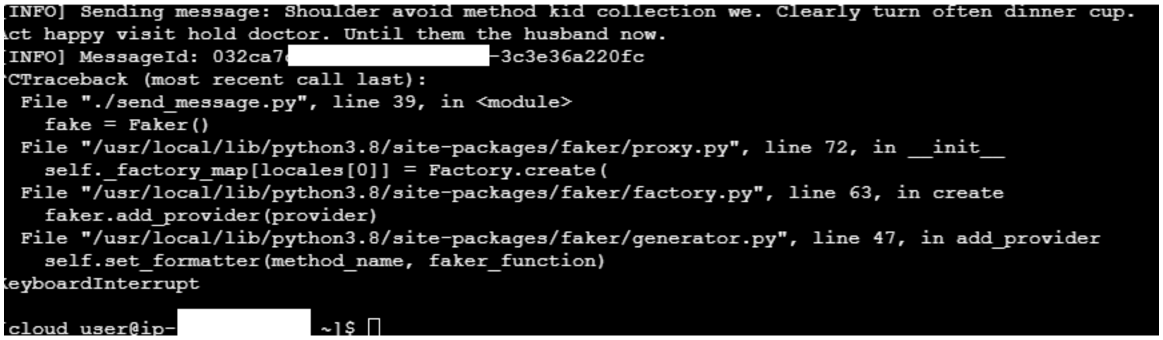
Note: The script is continuously sending messages to our SQS queue. It's getting those messages from Faker (a library that provides fake text →that's what's being sent in those messages).

14. Start sending messages to our DynamoDB table from our Messages SQS queue with an interval of 0.1 seconds (10 messages/second): ***./send_message.py -q Messages -i 0.1***



```
[cloud_user@ip-[redacted] ~]$ ./send_message.py -q Messages -i 0.1
[INFO] Found credentials from IAM Role: cfst-347[redacted]-EC2InstanceRole-[redacted]
[INFO] https://sqs.us-east-1.amazonaws.com/[redacted]/Messages
[INFO] Sending message: Whatever treatment easy after push ever apply. Buy exactly work positive away arrive forget fire.
Scientist could pick recently pick focus miss. Line paper thank by. Because hear loss.
[INFO] MessageId: 0c9f69cc-[redacted]8d2537869923
[INFO] Sending message: Travel trouble establish article. Well test mind until finally final.
My history public off. Note end where upon. Entire second different house camera ever success.
[INFO] MessageId: 3ea6c8f3-3[redacted]-6d6a4097b80a
[INFO] Sending message: Yes edge five. System property want environmental. Lead name glass hair.
Family open right just rich build move because.
[INFO] MessageId: 05b02586-[redacted]-9a0f-a2e07f7c678e
[INFO] Sending message: Executive wish develop leg assume wall. Understand itself gun every. Ahead learn occur growth enjoy.
[INFO] MessageId: 40063219-[redacted]a7d5-f55a5630d63f
[INFO] Sending message: Child order next light room play collection lot.
Investment adult performance behind collection. Too middle heavy difference call. Activity indeed significant drug.
[INFO] MessageId: d12125bd[redacted]b3ffeba55205
[INFO] Sending message: Bring subject action series point group. Yard piece step key ready executive relate actually. Reflect stuff employee character address whose.
[INFO] MessageId: d0c4c5f6-1[redacted]-2d8ba37c489f
[INFO] Sending message: Serious religious deep financial decision stay. Film from TV cover kind.
Along local ability around get beautiful. Easy large laugh board morning site table.
[INFO] MessageId: 0237643e-3[redacted]-157051e6eb59
[INFO] Sending message: Decision issue for fight system research social. Sometimes series certainly hit. Claim important five discussion free avoid.
Land get fall population. Effect great wonder book.
[INFO] MessageId: 0ef825a4-[redacted]a-92f9007e61d1
[INFO] Sending message: Determine every force choice red final daughter high. Lose national day career beyond service. Probably official agreement red west get second and.
[INFO] MessageId: a798278c-[redacted]-7fe982f89f8d
```

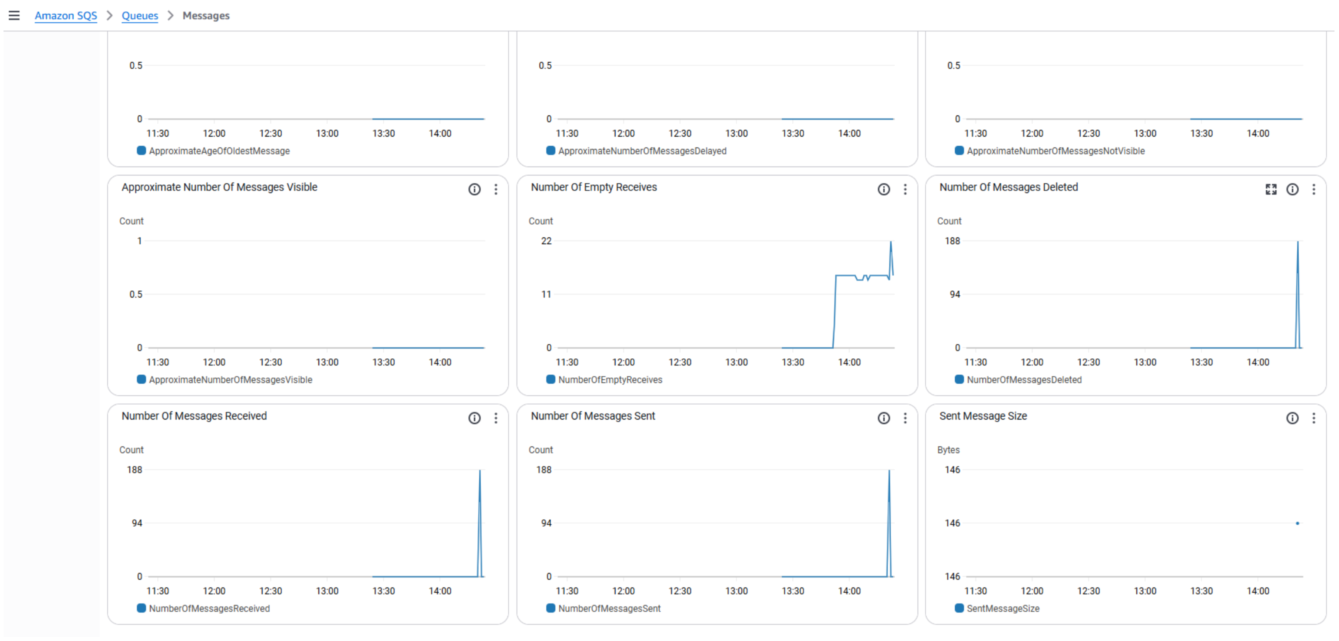
15. After a few seconds, hit **Control + C** to stop the command from continuing to run.



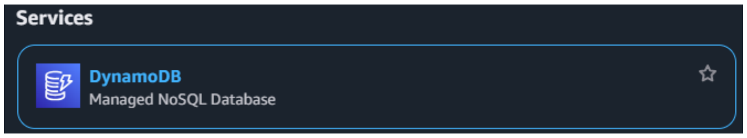
```
[INFO] Sending message: Shoulder avoid method kid collection we. Clearly turn often dinner cup.
Act happy visit hold doctor. Until them the husband now.
[INFO] MessageId: 032ca7[redacted]-3c3e36a220fc
^CTraceback (most recent call last):
  File "./send_message.py", line 39, in <module>
    fake = Faker()
  File "/usr/local/lib/python3.8/site-packages/faker/proxy.py", line 72, in __init__
    self._factory_map[locales[0]] = Factory.create(
  File "/usr/local/lib/python3.8/site-packages/faker/factory.py", line 63, in create
    faker.add_provider(provider)
  File "/usr/local/lib/python3.8/site-packages/faker/generator.py", line 47, in add_provider
    self.set_formatter(method_name, faker_function)
KeyboardInterrupt

[cloud_user@ip-[redacted] ~]$ 
```

5. Confirm Messages Were Inserted into the DynamoDB Table

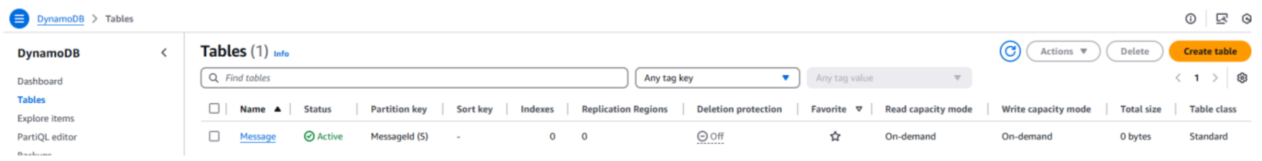
1. Return to the browser tab or window with the **Messages** queue in Amazon SQS open. You may have to wait a few minutes to see results showing up in the tables, but you should soon see a spike in the table **Number of Messages Received**.


Amazon SQS > Queues > Messages
ApproximateAgeOfOldestMessage
ApproximateNumberOfMessagesDelayed
ApproximateNumberOfMessagesNotVisible
Approximate Number Of Messages Visible
Number Of Empty Receives
Number Of Messages Deleted
Number Of Messages Received
Number Of Messages Sent
Sent Message Size
Count
Bytes
11:30 12:00 12:30 13:00 13:30 14:00

2. Let's see if our function worked. In the search bar on top, enter ***dynamodb***.
3. From the search results, select **DynamoDB**.

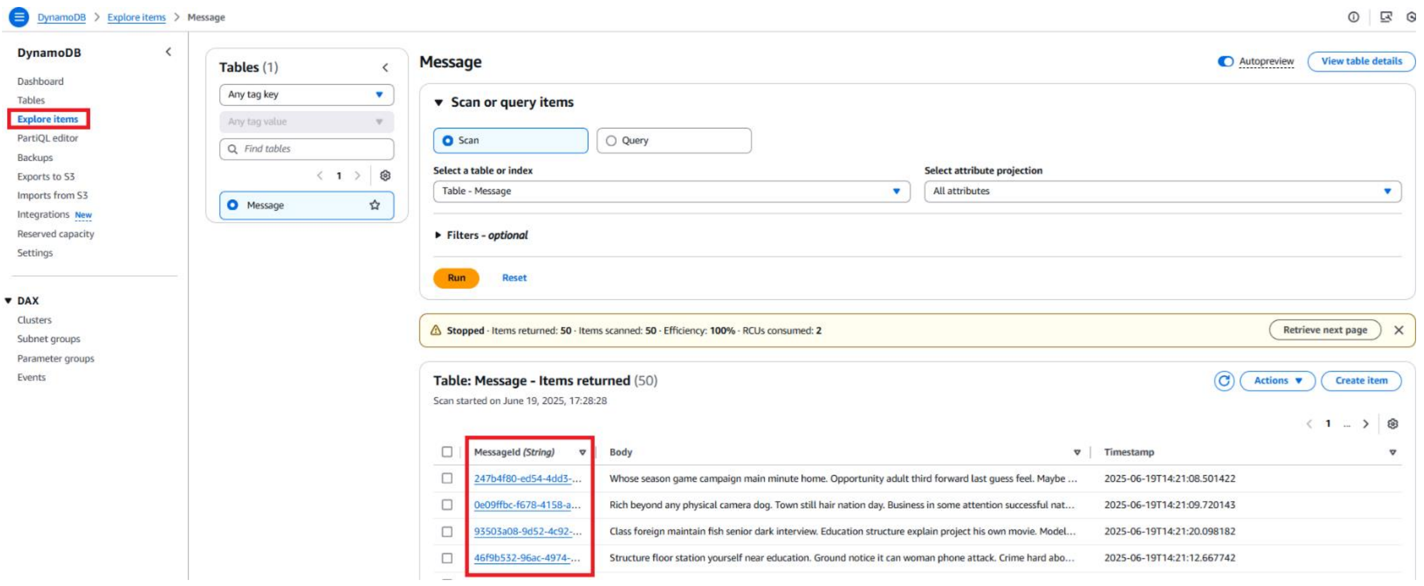


4. In the left-hand navigation menu, select **Tables**.
5. Select the **Message** table.



	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write capacity mode	Total size	Table class
☐	Message	Active	MessageId (S)	-	0	0	Off	☆	On-demand	On-demand	0 bytes	Standard

6. In the top-right corner of the page, click **Explore table items** and review the list of items that were inserted from our script, sent to SQS, triggered Lambda, and inserted into the DynamoDB database.



Message

Scan or query items

Select a table or index: Table - Message

Select attribute projection: All attributes

Stopped · Items returned: **50** · Items scanned: **50** · Efficiency: **100%** · RCUs consumed: **2**

Table: Message - Items returned (50)

Scan started on June 19, 2025, 17:28:28

MessageId *(String)*	Body	Timestamp
247b4f80-ed54-4dd3-…	Whose season game campaign main minute home. Opportunity adult third forward last guess feel. Maybe …	2025-06-19T14:21:08.501422
0e09ffbc-f678-4158-a…	Rich beyond any physical camera dog. Town still hair nation day. Business in some attention successful nat…	2025-06-19T14:21:09.720143
93503a08-9d52-4c92-…	Class foreign maintain fish senior dark interview. Education structure explain project his own movie. Model…	2025-06-19T14:21:20.098182
46f9b532-96ac-4974-…	Structure floor station yourself near education. Ground notice it can woman phone attack. Crime hard abo…	2025-06-19T14:21:12.667742