

Assignment – 1 - Lambda

Problem Statement:

You work for XYZ Corporation. Your corporation wants to launch a new web-based application and they do not want their servers to be running all the time. It should also be managed by AWS. Implement suitable solutions.

Tasks To Be Performed:

1. Create a sample Python Lambda function.
2. Set the Lambda Trigger as SQS and send a message to test invocations

Create a sample Python Lambda function.

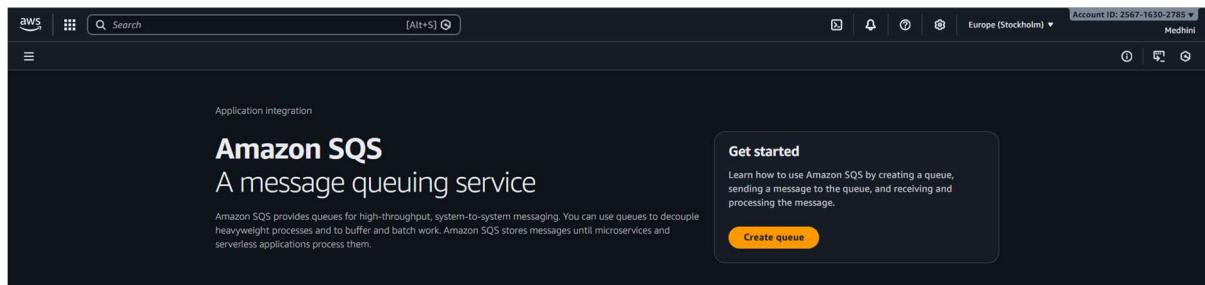
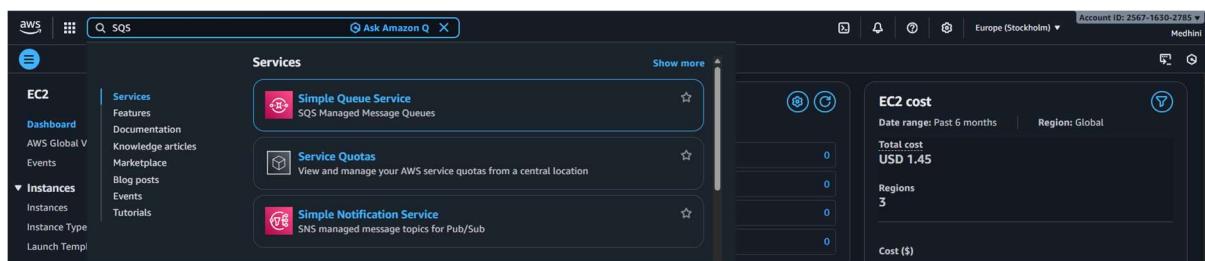
STEP 1: Create an SQS Queue

Open SQS

- Login to AWS Console
- Search SQS
- Click Create queue

Queue Settings

- Type: Standard
- Queue name: xyz-lambda-queue
- Leave all defaults
- Click Create queue



Assignment – 1 - Lambda

The screenshot shows the 'Create queue' wizard in the AWS SQS console. In the 'Details' step, the 'Type' is set to 'Standard'. The 'Name' is 'xyz-lambda-queue'. A note states: 'You can't change the queue type after you create a queue.' In the 'Configuration' step, 'Visibility timeout' is set to '30 Seconds' and 'Message retention period' is '4 Days'. Both fields have a note: 'Should be between 0 seconds and 12 hours.' and 'Should be between 1 minute and 14 days.' respectively.

The screenshot shows the 'xyz-lambda-queue' details page. A green banner at the top says: 'Queue xyz-lambda-queue created successfully. You can now send and receive messages.' Below, the queue details are listed: Name (xyz-lambda-queue), Type (Standard), ARN (arn:aws:sqs:eu-north-1:256716302785:xyz-lambda-queue), and Dead-letter queue (-). The 'Access policy' section shows a JSON policy document:

```
{ "Version": "2012-10-17", "Id": "_default_policy_ID", "Statement": [ { "Sid": "statementID", "Effect": "Allow", "Principal": "*", "Action": "SQS:SendMessage", "Resource": "arn:aws:sqs:eu-north-1:256716302785:xyz-lambda-queue", "Condition": {} } ] }
```

STEP 2: Create IAM Role for Lambda

Lambda needs permission to read messages from SQS.

Open IAM

- Go to **IAM - Roles**
- Click **Create role**

Select Trusted Entity

- Select **AWS Service**
- Choose **Lambda**
- Click **Next**

Attach Policies

Attach these policies:

Assignment – 1 - Lambda

AWSLambdaBasicExecutionRole

AmazonSQSFullAccess (for learning purpose)

Click **Next**

Role Name

- Role name: lambda-sqs-role
- Click **Create role**

The image consists of three vertically stacked screenshots of the AWS IAM Dashboard.

Screenshot 1: IAM Dashboard
The left sidebar shows "Access management" with "Roles" selected. The main area displays "Security recommendations" and "IAM resources". A red box highlights the "Roles" link in the sidebar.

Screenshot 2: Roles List
The left sidebar shows "Access management" with "Roles" selected. The main area lists existing roles: "AWSBackupDefaultServiceRole", "AWSServiceRoleForAmazonElasticFileSystem", "AWSServiceRoleForAmazonSES", "AWSServiceRoleForAutoScaling", and "AWSServiceRoleForBackup". A red box highlights the "Create role" button at the top right.

Screenshot 3: Create Role Step 2
The left sidebar shows "Step 2: Add permissions". The main area is titled "Add permissions" and shows a search bar for "Permissions policies". It lists "AmazonSQSFullAccess" under "Policy name". A red box highlights the "AmazonSQSFullAccess" policy in the list.

Assignment – 1 - Lambda

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. The 'Permissions policies' section displays a search result for 'AWSLambdaBasicExecutionRole'. The policy is described as 'Provides write permissions to CloudWatch Metrics'. The 'Set permissions boundary - optional' section is collapsed. Navigation buttons 'Cancel', 'Previous', and 'Next' are visible at the bottom.

The screenshot shows the 'Select trusted entity' step. Under 'Trusted entity type', the 'AWS service' option is selected, which is highlighted with a red box. Under 'Use case', the 'Lambda' option is selected, also highlighted with a red box. The 'Next' button is highlighted with a red box at the bottom right.

The screenshot shows the 'Name, review, and create' step. In the 'Role details' section, the 'Role name' field contains 'lambda-sqs-role'. The 'Description' field contains 'Allows Lambda functions to call AWS services on your behalf.' The 'Step 1: Select trusted entities' link is highlighted with a red box at the bottom left.

Assignment – 1 - Lambda

The screenshot shows the 'Create role' wizard in the AWS IAM console. In Step 2: Add permissions, two managed policies are attached:

Policy name	Type	Attached as
AmazonSQSFullAccess	AWS managed	Permissions policy
AWSLambdaBasicExecutionRole	AWS managed	Permissions policy

The screenshot shows the details of the 'lambda-sqs-role' created in the previous step. The role summary includes:

- Creation date: January 01, 2026, 21:28 (UTC+05:30)
- Last activity: -
- ARN: arn:aws:iam::256716302785:role/lambda-sqs-role
- Maximum session duration: 1 hour

The Permissions tab shows the attached policies:

Policy name	Type	Attached entities
AmazonSQSFullAccess	AWS managed	1
AWSLambdaBasicExecutionRole	AWS managed	1

STEP 3: Create Python Lambda Function

Open Lambda

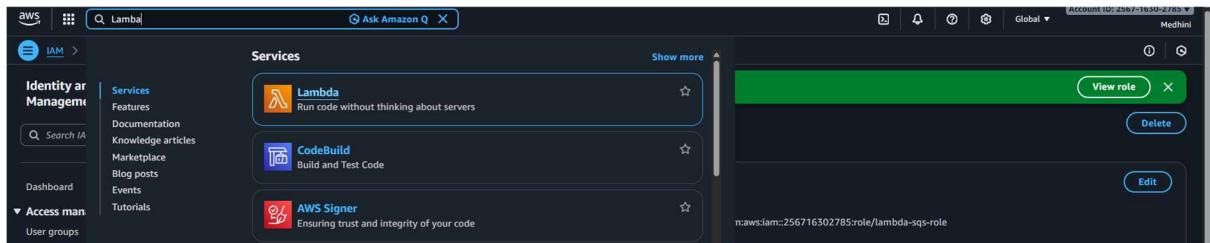
- Search **Lambda**
- Click **Create function**

Function Settings

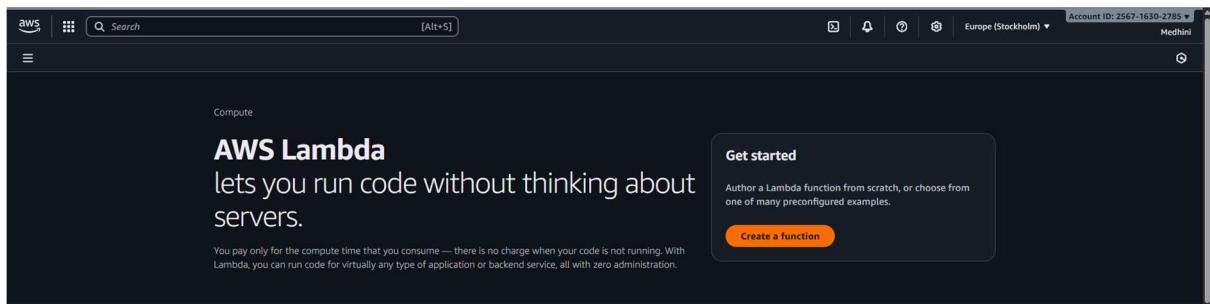
- **Author from scratch**
- **Function name:** xyz-sqs-lambda
- **Runtime:** Python 3.12
- **Execution role:** Use existing role
- Select **lambda-sqs-role**

Assignment – 1 - Lambda

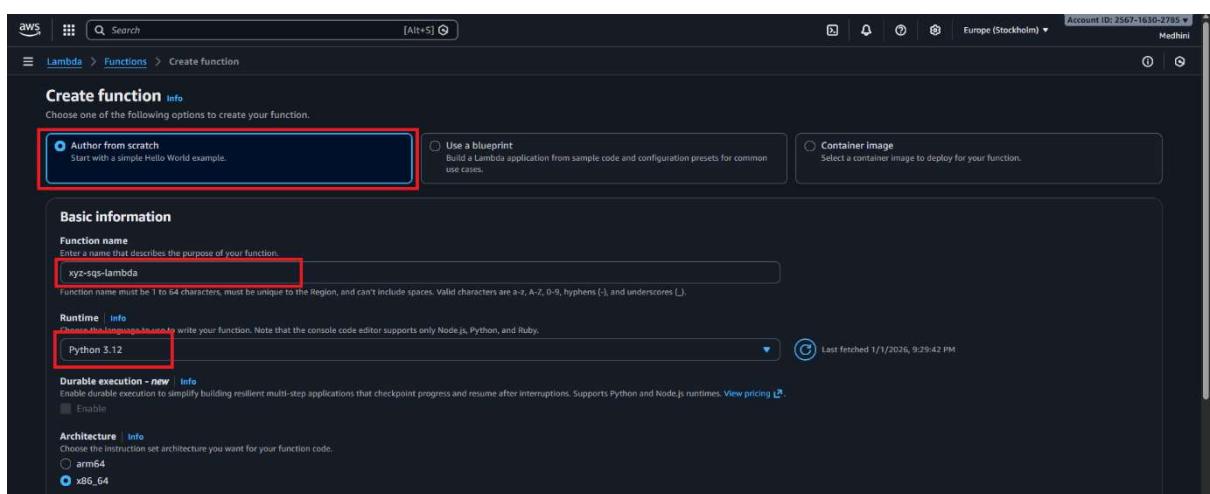
Click **Create function**



The screenshot shows the AWS Lambda service page. The Lambda icon is highlighted with a green box. A role named "nawsiam:256716302785:role/lambda-sqs-role" is listed on the right.



The screenshot shows the AWS Lambda landing page. The "Create a function" button is highlighted with a green box.

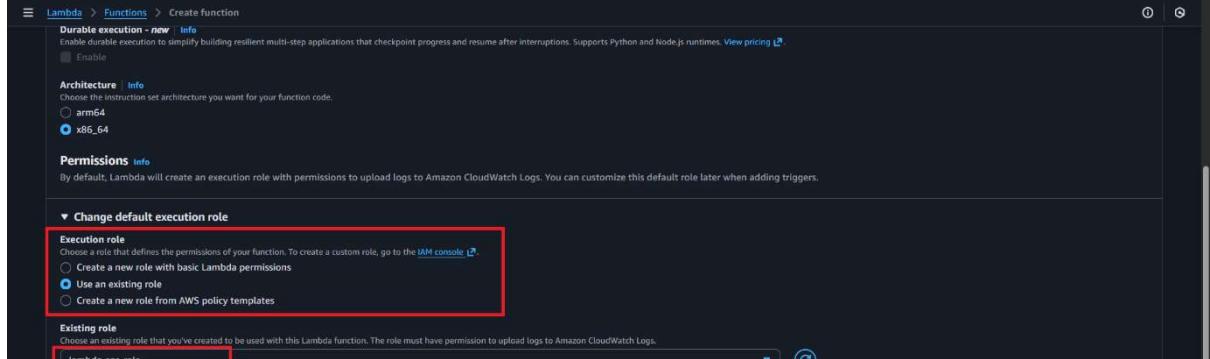


The screenshot shows the "Create function" wizard, Step 1: Basic information. The "Author from scratch" option is selected and highlighted with a red box. Other options like "Use a blueprint" and "Container Image" are shown below.

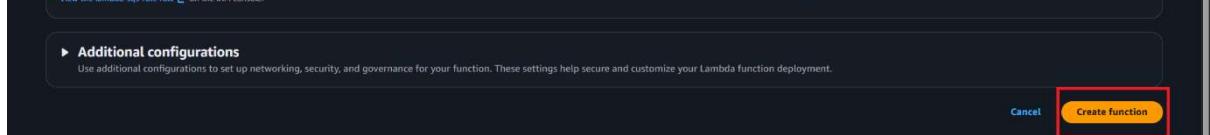


The screenshot shows the "Create function" wizard, Step 2: Durable execution. The "Durable execution - new" info section is shown, with the "Enable" checkbox highlighted with a red box.

The screenshot shows the "Create function" wizard, Step 3: Architecture. The "Architecture" dropdown is set to "x86_64" and highlighted with a red box.

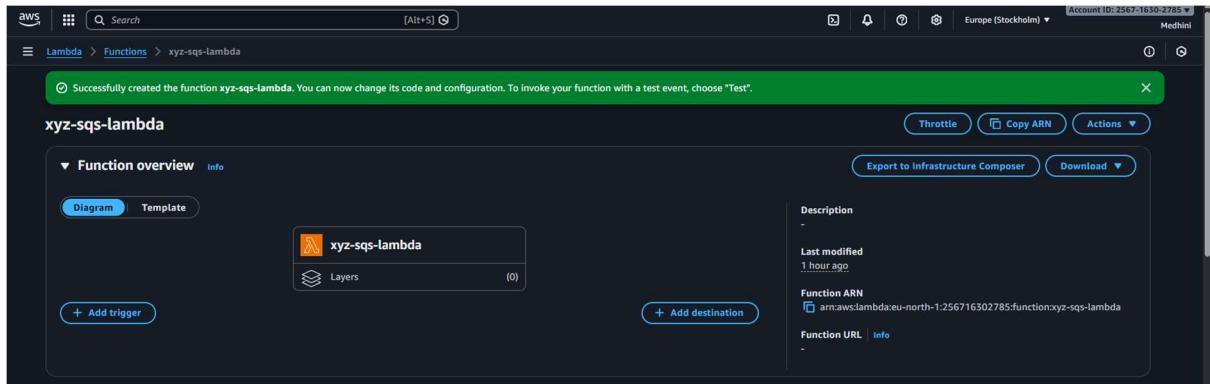


The screenshot shows the "Create function" wizard, Step 4: Permissions. The "Execution role" dropdown is set to "lambda-sqs-role" and highlighted with a red box. The "Create default execution role" section is expanded, showing options for creating a new role or using an existing one.



The screenshot shows the "Create function" wizard, Step 5: Additional configurations. The "Create Function" button is highlighted with a red box.

Assignment – 1 - Lambda



STEP 4: Add Python Code to Lambda

Replace default code with this

```
import json

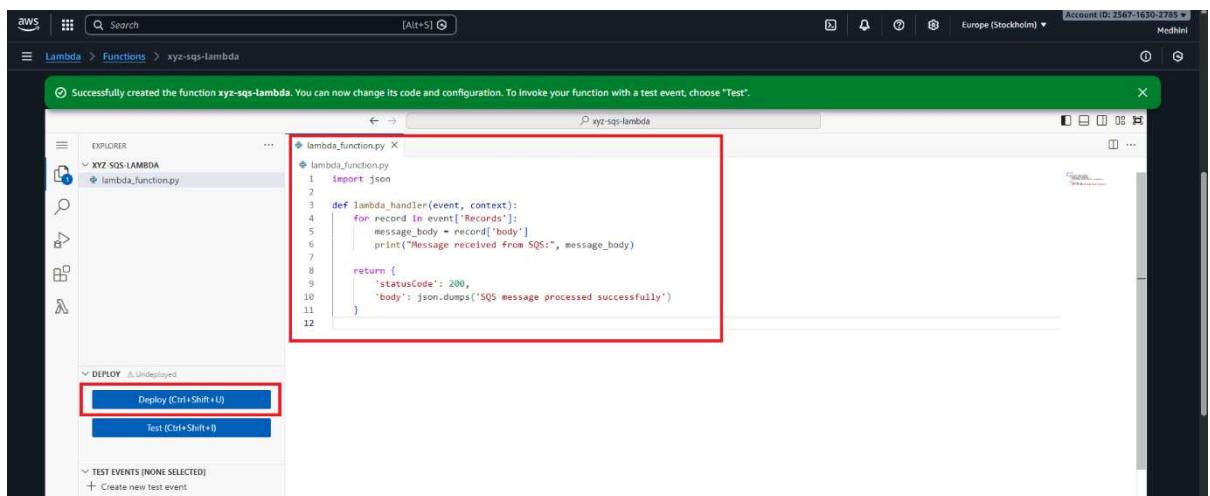
def lambda_handler(event, context):

    for record in event['Records']:
        message_body = record['body']

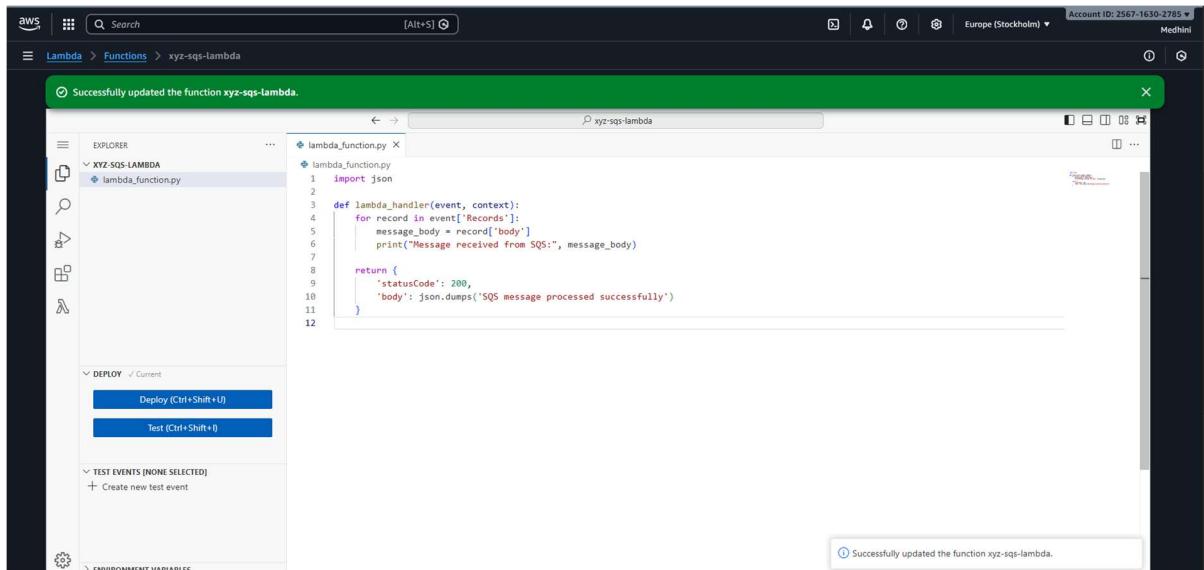
        print("Message received from SQS:", message_body)

    return {
        'statusCode': 200,
        'body': json.dumps('SQS message processed successfully')
    }
```

Click Deploy



Assignment – 1 - Lambda



The screenshot shows the AWS Lambda function editor for the function 'xyz-sqs-lambda'. The code in 'lambda_function.py' is:

```
import json

def lambda_handler(event, context):
    for record in event['Records']:
        message_body = record['body']
        print("Message received from SQS:", message_body)

    return {
        'statusCode': 200,
        'body': json.dumps('SQS message processed successfully')
    }
```

The interface includes tabs for EXPLORER, DEPLOY, and TEST EVENTS. A success message 'Successfully updated the function xyz-sqs-lambda.' is displayed at the bottom.

STEP 5: Add SQS as Lambda Trigger

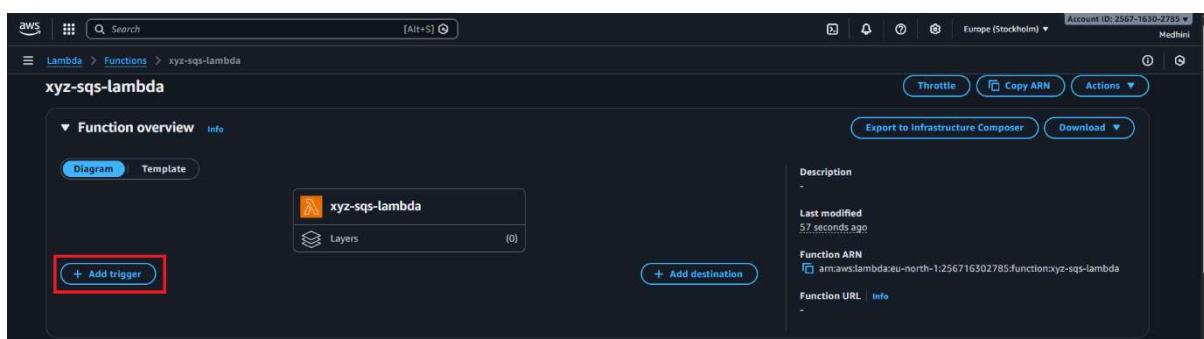
Add Trigger

- In Lambda → Click **Add trigger**
- Select **SQS**

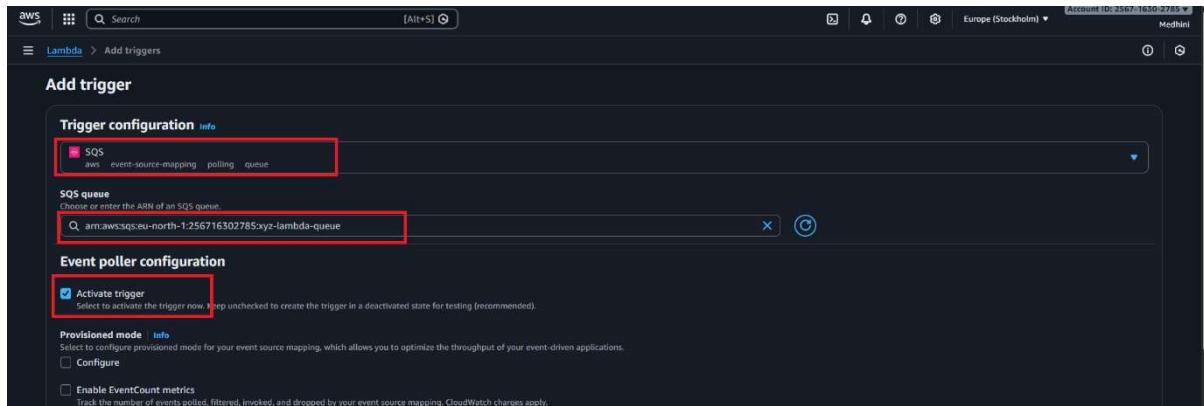
Configure Trigger

- Choose queue: xyz-lambda-queue
- Enable trigger
- Click **Add**

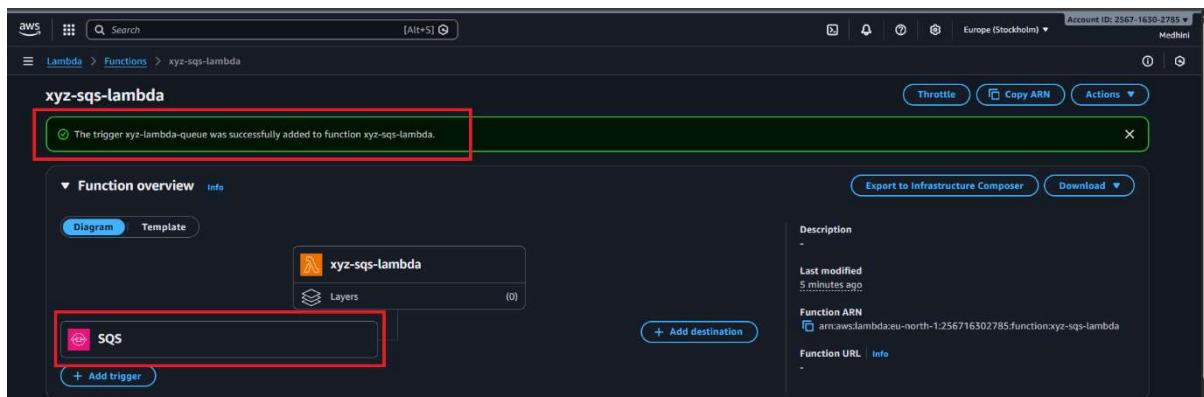
Now SQS is connected to Lambda



Assignment – 1 - Lambda



Scroll down and click on “Add”



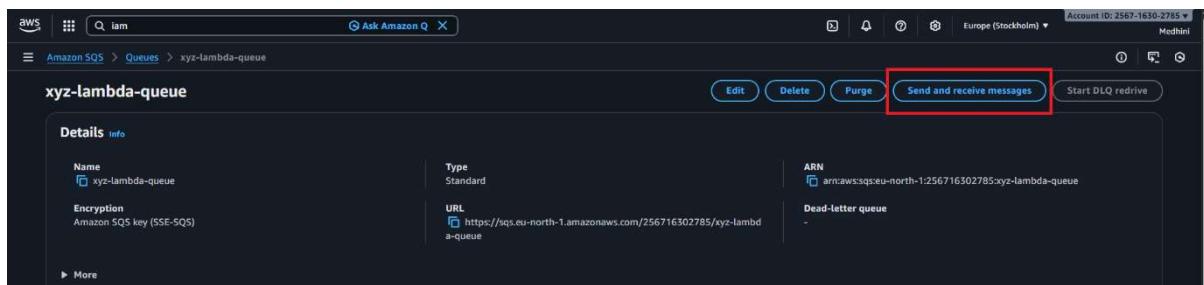
STEP 6: Send Message to SQS (Test)

Open SQS

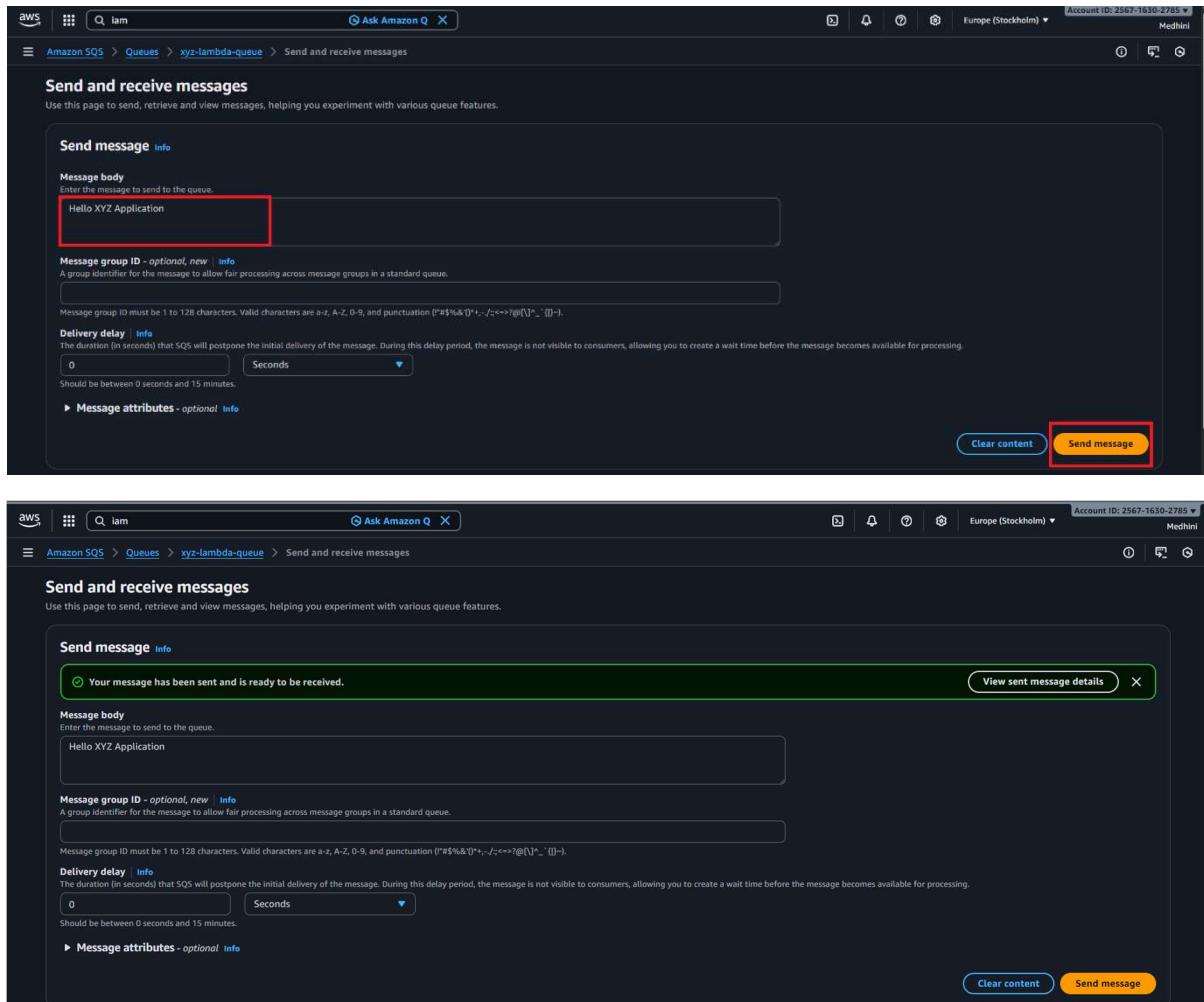
- Click xyz-lambda-queue
- Click **Send and receive messages**

Send Message

Message body example:



Assignment – 1 - Lambda

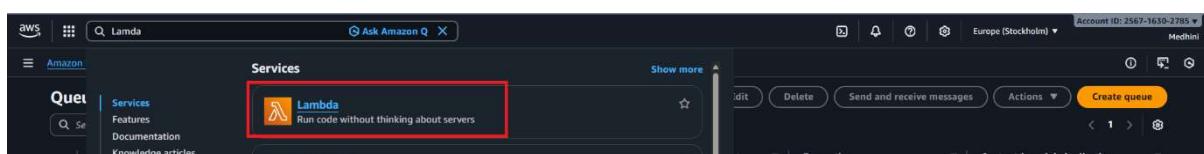


The screenshots show the AWS SQS 'Send and receive messages' interface. In the first screenshot, a message is being sent with the body 'Hello XYZ Application'. In the second screenshot, a success message is displayed: 'Your message has been sent and is ready to be received.' Both screenshots highlight the 'Send message' button.

STEP 7: Verify Lambda Execution

Go to Lambda

- Open xyz-sqs-lambda
- Click **Monitor**
- Click **View logs in CloudWatch**



The screenshot shows the AWS Lambda service page. It features a prominent 'Lambda' button with the text 'Run code without thinking about servers'. Other navigation options like 'Services', 'Features', 'Documentation', and 'Knowledge articles' are also visible.

Assignment – 1 - Lambda

The screenshot shows the AWS Lambda Function Overview page for a function named "xyz-sqs-lambda". The function has a single trigger, "SQS", which is highlighted with a red box. The "Monitor" tab is selected. On the right, there is a "Code source" section showing the file "lambda_function.py" with the code "import json".

This screenshot is identical to the one above, but the "Monitor" tab is explicitly highlighted with a red box.

The screenshot shows the AWS CloudWatch Log Management page for the log group "/aws/lambda/xyz-sqs-lambda". A log stream is selected, and its ARN is shown as "arn:awslogs:eu-north-1:256716302785:log-group:/aws/lambda/xyz-sqs-lambda:2026/01/01/[\$LATEST]a9c884be05eb4f7099498ea0d3b8a00d". This ARN is also highlighted with a red box.

Assignment – 1 - Lambda

The screenshot shows the AWS CloudWatch Log Management interface. The URL is /aws/lambda/xyz-sqs-lambda. The log events table has two columns: Timestamp and Message. The messages show the Lambda function starting, receiving a message from SQS, processing it, and then reporting back to SQS.

Timestamp	Message
2026-01-01T17:31:50.248Z	INIT_START Runtime Version: python:3.12.v101 Runtime Version ARN: arn:aws:lambda:eu-north-1:runtime:994aac32248ecf4d69d9f5e9a3a57ab3cce19d94170a61d5ecf978927e1b0f
2026-01-01T17:31:50.336Z	START RequestId: 00bea46c-8b84-5f48-8765-c5bc3a321ce5 Version: \$LATEST
2026-01-01T17:31:50.336Z	Message received from SQS: Hello XYZ Application
2026-01-01T17:31:50.338Z	END RequestId: 00bea46c-8b84-5f48-8765-c5bc3a321ce5
2026-01-01T17:31:50.338Z	REPORT RequestId: 00bea46c-8b84-5f48-8765-c5bc3a321ce5 Duration: 1.74 ms Billed Duration: 86 ms Memory Size: 128 MB Max Memory Used: 36 MB Init Duration: 84.12 ms

Lambda triggered automatically