

## Test Amazon SQS using AWS Lambda

### 1. Create an Amazon SQS queue

1. Login to AWS console - <https://console.aws.amazon.com>
2. Type **SQS** in the Services search box and select **Amazon SQS**
3. On the **SQS** console home, click **Create New Queue** or click **Get started**
4. Enter **<Your\_Name>-LabQueue.fifo** in the **Queue Name** textbox.
5. Select **FIFO Queue** under **\*\*What type of queue do you need?**
6. Scroll down and click **Quick-Create Queue**
7. Your queue is now successfully created.
8. Copy the **URL** from the **Details** tab at the bottom of the screen and save it in a text editor. We will use this URL in the Lambda function we will create later.

### 2. Create a Python based Lambda function to send messages to SQS

1. Navigate to AWS Lambda
2. Click on **Create function**
3. Select **Author from scratch**
4. Name the function as **<Your\_Name>-sendmessage**
5. Select **Python 3.6** as the runtime
6. Click **Create function**
7. Replace the default code with the code in the block below

```
import json
import boto3

def lambda_handler(event, context):

    sqs_queue_url="<QUEUE_URL>"

    sqs_client = boto3.client('sqs')

    msg =
    sqs_client.send_message(QueueUrl=sqs_queue_url,MessageBody=event,MessageGroupId='
    mygroup',MessageDeduplicationId='dedupeID',
    )
```

8. Replace **<QUEUE\_URL>** with the URL you saved into the text editor earlier
9. Click on **Save** at the top of the screen to save the code changes you just made.
10. Scroll down to the **Execution role** section on the Lambda console. You will see that a new IAM Role has been created for the Lambda function with some basic permissions as


shown in the screenshot below.

**Execution role**

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role ▼

**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.

service-role/sendmessage-role-9hung6pn ▼ 

[View the sendmessage-role-9hung6pn role](#) on the IAM console.

11. For the Lambda function to be able to send messages to the SQS queue, it needs to have necessary permissions. Click on **View the sendmessage-role-...** link which will open a new tab and take you to the IAM console.
12. Click on **Attach policies**
13. Type **sqs** in the search textbox and select **AmazonSQSFullAccess** policy checkbox
14. Click **Attach policy**. Once the policy is saved, close the browser tab.

### 3. Create a Python based Lambda function to read messages from the Queue

1. Navigate to **AWS Lambda** console and click **Create function**
2. Name the function as **readmessage**
3. Select **Python 3.6** as the runtime
4. Expand **Choose or create an execution role** by clicking on it
5. Select **Use an existing role** under **Execution role** section

6. Select the same role you created earlier for the **<Your\_Name>-sendmessage** lambda function. See screenshot below for details

**Basic information**

Function name

Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)

Choose the language to use to write your function.

Python 3.6 ▼

Permissions [Info](#)

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role ▼

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.

service-role/sendmessage-role-9hung6pn ▼

[View the sendmessage-role-9hung6pn role](#) on the IAM console.

↻

7. Click **Create function**
8. Replace the default code with the code below and click the **Save** button at the top right to save the changes

```
import json
import boto3

def lambda_handler(event, context):
    # Retrieve messages from an SQS queue
    sqs_queue_url = "<QUEUE_URL>"
    sqs_client = boto3.client('sqs')

    msg = sqs_client.receive_message(QueueUrl=sqs_queue_url)

    print(msgs)
```

9. Replace **<QUEUE\_URL>** with the URL of the queue you created earlier in the exercise

## Test everything

### Send message to the SQS queue

1. Navigate to the <Your\_Name>-sendmessage Lambda function page
2. Click on the drop down near the **Test** button at the top right and select **Configure test events**
3. In the new popup, select **Create new test event**
4. Select **Hello World** template.
5. Name the event as <Your\_Name>-newmessage
6. Clear the textbox with sample input json and enter any string within quotes. See screenshot below for details

**Configure test event** ×

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☒ Create new test event  
☐ Edit saved test events

Event template

Hello World ▼

Event name

newmessage

```
1 "hello from python"
```

7. Click **Save** at the bottom of the screen
8. Simply click on the **Test** button on the <Your\_Name>-sendmessage Lambda function home page to send the message to SQS
9. You should see the user interface saying **Execution result:succeeded**. Click on it to see details of the execution.

### Check the message in SQS queue

1. Navigate to Amazon SQS home page and select **LabQueue.info**
2. Click on **Send and receive messages** and click **Poll for messages**
3. You should be able to see the message that you just sent from the Lambda function.

### Read the message from the SQS queue

1. Navigate to the <Your\_Name>-readmessage Lambda function page
2. Click on the drop down near the **Test** button at the top right and select **Configure test events**
3. In the new popup, select **Create new test event**
4. Select **Hello World** template.

5. Name the event as **readmessage**
6. Clear the textbox with sample input json and replace it with empty quotes

**Configure test event** ✕

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☒ Create new test event  
☐ Edit saved test events

Event template

Hello World ▼

Event name

readmessage

1	""
---	----

7. Click **Save** at the bottom of the screen
8. Simply click on the **Test** button on the <Your\_Name>-readmessage Lambda function home page to send the message to SQS
9. You should see the user interface saying **Execution result:succeeded**. Expanding it will show the details of the execution along with the content of the SQS message in the **Log output** section.