



# Simple Queue Service

1. Now, in this lab, I want to give an example of a program that can send messages onto the queue.
2. You can get the program from GitHub.
3. This is a program in Dotnet that can go ahead and send messages onto an Amazon simple queue.
4. Now you will get the program in a zip file, so unzip it and open it in Visual Studio.
5. Then there you need to change the URL. You can get this URL from SQS.

The screenshot shows the AWS SQS console with a single queue named 'demo-sqs-queue'. The queue details are as follows:

- Name: demo-sqs-queue
- Type: Standard
- ARN: arn:aws:sqs:ap-south-1:878893308172:demo-sqs-queue
- URL: https://sqs.ap-south-1.amazonaws.com/878893308172/demo-sqs-queue
- Dead-letter queue: -

The screenshot shows a Visual Studio code editor with a C# file named 'Program.cs' containing the following code:

```
Order.cs Program.cs  What's New?
SendMessages
1  using Amazon.SQS.Model;
2  using Amazon.SQS;
3  using SendMessages;
4  using Newtonsoft.Json;
5
6  string queueURL = "https://sqs.ap-south-1.amazonaws.com/878893308172/demo-sqs-queue";
7  var sqsClient = new AmazonSQSClient();
8
9  for (int i=1; i<=5;i++)
10 {
11     Order order=new Order();
12     order.orderId = i;
13     order.productName = $"Product{i}";
14     order.quantity = (i * 5);
15     await SendMessage(sqsClient,queueURL, JsonConvert.SerializeObject(order));
16 }
17
18 1 reference
19  static async Task SendMessage(
20      IAmazonSQS sqsClient, string queueUrl, string messageBody)
21  {
22      SendMessageResponse msgResponse =
23          await sqsClient.SendMessageAsync(queueUrl, messageBody);
24      Console.WriteLine("Message added to queue");
25  }
26
27
```

The line `string queueURL = "https://sqs.ap-south-1.amazonaws.com/878893308172/demo-sqs-queue";` is highlighted with a red box.

6. After that you need to create an IAM user.
7. In terms of permissions, I have assigned an inbuilt policy in AWS to allow full access onto the simple queue service. So, this will allow this user to interact with SQS.

The screenshot shows the AWS IAM 'Permissions policies' page with one policy listed:

Policy name	Type	Attached via
AmazonSQSFullAccess	AWS managed	Directly

8. And then in the command prompt, when it comes on to the AWS CLI, I have configured the credentials so Visual Studio can actually go ahead and make use of these same credentials.
9. Once everything is setup go to visual studio and run the program in your local machine.
10. Now you will go back to your queue and click on send and receive messages. There you will see that you have 5 messages available.
11. Now click on Poll for messages.

The screenshot shows the 'Receive messages' interface for an SQS queue. At the top, there are three buttons: 'Edit poll settings', 'Stop polling', and 'Poll for messages'. Below these, there are four input fields: 'Messages available' (5), 'Polling duration' (30), 'Maximum message count' (10), and 'Polling progress' (0% / 0 receives/second). A search bar labeled 'Search messages' is present. The main area is titled 'Messages (0)' and contains a table header with columns: ID, Sent, Size, and Receive count. A message body states 'No messages. To view messages in the queue, poll for messages.' At the bottom is a large 'Poll for messages' button.

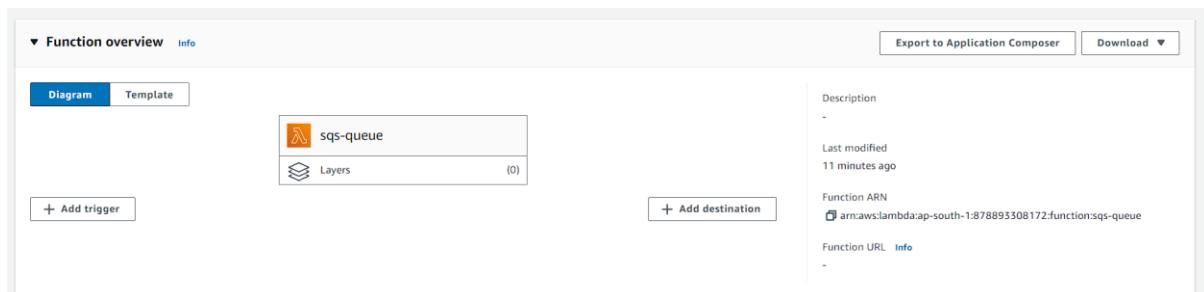
12. You will see all 5 IDs for messages. You can open them and see what's inside.

This screenshot is identical to the one above, showing the 'Receive messages' interface with 5 messages available. The 'Poll for messages' button is located at the bottom of the page.

13. Now you are going to create a Lambda function which is going to receive the messages from queue.
14. So, navigate to Lambda and create a function, choose the runtime environment as Python.
15. Now before that go to IAM and add permission for SQS in your role.

The screenshot shows the 'Other permissions policies' section in the IAM console. It lists three matches for the search term 'sqS'. The policies are: 'AmazonSQSFullAccess' (AWS managed), 'AmazonSQSReadOnlyAccess' (AWS managed), and 'AWSLambdaSQSQueueExecutionRole' (AWS managed). The 'AWSLambdaSQSQueueExecutionRole' policy is currently selected, indicated by a checked checkbox. At the bottom right are 'Cancel' and 'Add permissions' buttons.

16. Now in your function you need to add a trigger as SQS. Then you need to change the code.



## Add trigger

**Trigger configuration** [Info](#)

SQS

aws event-source-mapping polling queue

SQS queue

Choose or enter the ARN of an SQS queue.

X C

Activate trigger

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

17. Once the trigger is added now you need to change the code. Then deploy the code.

```
import json

def lambda_handler(event, context):
    if 'Records' in event:
        for message in event['Records']:
            order_details = json.loads(message['body'])
            order_id = order_details['orderId']
            product_name = order_details['productName']
            quantity = order_details['quantity']

            print(order_id)
            print(product_name)
            print(quantity)
    else:
        print("No records found in the event.)
```

The screenshot shows the AWS Lambda function editor. The top navigation bar has 'Tools' and 'Window' tabs, with 'Test' selected. Below the tabs are three cards: 'lambda\_function' (selected), 'Environment Var', and 'Execution results'. The code area contains the following Python script:

```

1 import json
2
3 def lambda_handler(event, context):
4     if 'Records' in event:
5         for message in event['Records']:
6             order_details = json.loads(message['body'])
7             order_id = order_details['orderId']
8             product_name = order_details['productName']
9             quantity = order_details['quantity']
10
11             print(order_id)
12             print(product_name)
13             print(quantity)
14     else:
15         print("No records found in the event.")
16

```

18. After that you need to go to CloudWatch. Then open the log groups.

19. In there you will see multiple logs generated for you SQS. Now you can see your messages in your logs instead of SQS.

The screenshot shows the CloudWatch Log Stream interface. The top navigation bar includes 'Log streams', 'Tags', 'Anomaly detection', 'Metric filters', 'Subscription filters', 'Contributor Insights', and 'Data protection'. The 'Log streams' tab is selected. The main area displays a table of log streams:

	Log stream	Last event time
<input type="checkbox"/>	2024/02/13/[LATEST]ff6be46a6c0d4a2ba24b561d1fa6c388	2024-02-13 11:00:13 (UTC+05:30)
<input type="checkbox"/>	2024/02/13/[LATEST]cb889453db9a4ea6a80c623af48efe31	2024-02-13 11:00:13 (UTC+05:30)
<input type="checkbox"/>	2024/02/13/[LATEST]8fb9f5af4605409dbaf4e2892254e5c8	2024-02-13 11:00:13 (UTC+05:30)
<input type="checkbox"/>	2024/02/13/[LATEST]9cc3a93ec004fdc861439d8735a6ba2	2024-02-13 11:00:13 (UTC+05:30)
<input type="checkbox"/>	2024/02/13/[LATEST]89dde78342c7430e9bd15341a5a0ceae	2024-02-13 11:00:13 (UTC+05:30)

Below this, a detailed log table shows the execution of a Lambda function:

	Timestamp	Message
		No older events at this moment. <a href="#">Retry</a>
▶	2024-02-13T11:00:13.564+05:30	INIT_START Runtime Version: python:3.12.v18 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:776a3759221679a634181f858871d5514dc74a176f78bc535f822a93..
▶	2024-02-13T11:00:13.645+05:30	START RequestId: 8b64e46c-e2c2-572b-b3ce-ba29e25deada Version: \$LATEST
▶	2024-02-13T11:00:13.646+05:30	1
▶	2024-02-13T11:00:13.646+05:30	Product1
▶	2024-02-13T11:00:13.646+05:30	5
▶	2024-02-13T11:00:13.647+05:30	END RequestId: 8b64e46c-e2c2-572b-b3ce-ba29e25deada
▶	2024-02-13T11:00:13.647+05:30	REPORT RequestId: 8b64e46c-e2c2-572b-b3ce-ba29e25deada Duration: 1.91 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 34 MB Init Duration: 80..
▶	2024-02-13T11:00:13.664+05:30	START RequestId: 7d49b2f0-016e-5fff-aa6c-ecb501f11f38 Version: \$LATEST
▶	2024-02-13T11:00:13.664+05:30	3
▶	2024-02-13T11:00:13.664+05:30	Product3
▶	2024-02-13T11:00:13.664+05:30	15
▶	2024-02-13T11:00:13.686+05:30	END RequestId: 7d49b2f0-016e-5fff-aa6c-ecb501f11f38
▶	2024-02-13T11:00:13.686+05:30	REPORT RequestId: 7d49b2f0-016e-5fff-aa6c-ecb501f11f38 Duration: 22.03 ms Billed Duration: 23 ms Memory Size: 128 MB Max Memory Used: 34 MB
▶	2024-02-13T11:00:13.690+05:30	START RequestId: e87fd6a-9d22-51d5-8d5d-802f9fdbda940 Version: \$LATEST
▶	2024-02-13T11:00:13.690+05:30	5

A yellow 'Back to top' button is located at the bottom right of the log table.