

School of Computer Science, Engineering and Applications (SCSEA)

B. Tech TY (CCSA)

Subject: Cloud Architecture And Protocol

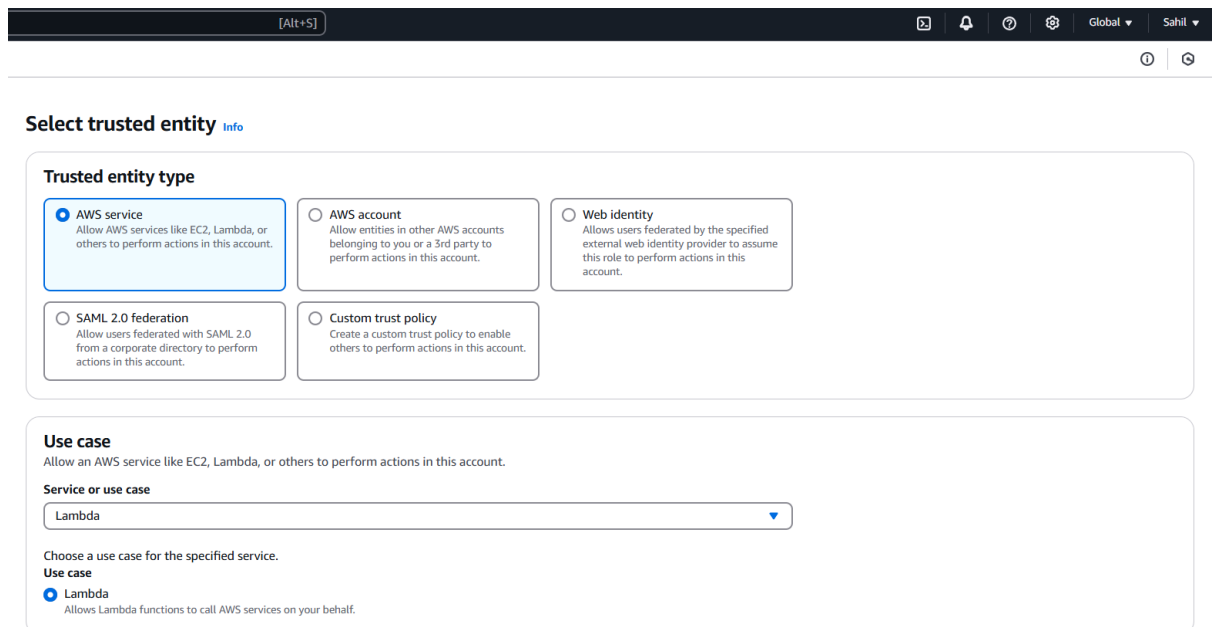
Name of the Student: Sahil S. Mandawgade

PRN: 20220802265

Title of Practical: 11. Serverless Event-Driven Architecture: Amazon SQS Event Processing with AWS Lambda

Step 1: IAM Role Creation for Lambda Function.

- Go to IAM in AWS Console.
- Create a new IAM Role with proper name.
 - Select 'Trusted Entity Type' as 'AWS Service'.
 - Select 'Use Case' as 'Lambda'.
 - Attach "AmazonSQSFullAccess" policy.
 - Set name of role as 'Role_SQS_2265'.
 - Review and click on 'Create role'.



The screenshot shows the AWS IAM console interface for creating a new role. The 'Select trusted entity' step is active, displaying five options for the 'Trusted entity type':

- AWS service** (selected): Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account**: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity**: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation**: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy**: Create a custom trust policy to enable others to perform actions in this account.

Below the trusted entity type selection, the 'Use case' section is visible, showing the 'Service or use case' dropdown set to 'Lambda' and the 'Use case' dropdown set to 'Lambda' (selected), which allows Lambda functions to call AWS services on your behalf.

School of Computer Science, Engineering and Applications (SCSEA)

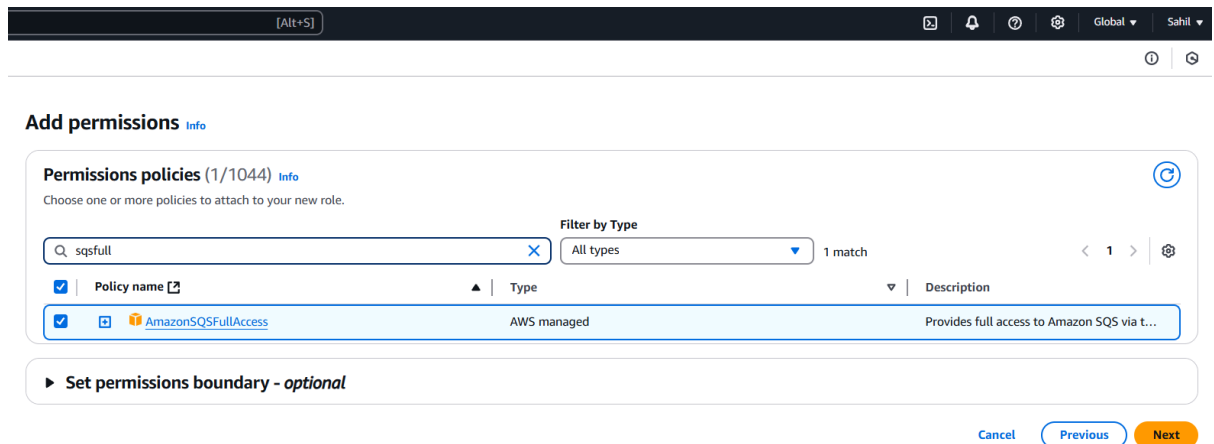
B. Tech TY (CCSA)

Subject: Cloud Architecture And Protocol

Name of the Student: Sahil S. Mandawgade

PRN: 20220802265

Title of Practical: 11. Serverless Event-Driven Architecture: Amazon SQS Event Processing with AWS Lambda




[Alt+S] [Icons] Global Sahil

Add permissions Info

Permissions policies (1/1044) Info

Choose one or more policies to attach to your new role.

Filter by Type: All types 1 match

<input checked="" type="checkbox"/>	Policy name	Type	Description
<input checked="" type="checkbox"/>	 AmazonSQSFullAccess	AWS managed	Provides full access to Amazon SQS via t...

► Set permissions boundary - optional

Cancel Previous Next

Step 2: Create an SQS Queue

- Go to SQS in AWS Console.
- Select "Standard" queue.
- Set name as 'Lab11-Queue-2265'.
- Set the visibility timeout to 1 minute.
- Create the queue.

School of Computer Science, Engineering and Applications (SCSEA)

B. Tech TY (CCSA)

Subject: Cloud Architecture And Protocol

Name of the Student: Sahil S. Mandawgade

PRN: 20220802265

Title of Practical: 11. Serverless Event-Driven Architecture: Amazon SQS Event Processing with AWS Lambda

Amazon SQS > Queues > Create queue

Create queue

Details

Type

Choose the queue type for your application or cloud infrastructure.



Standard [Info](#)

At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering



FIFO [Info](#)

First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing



You can't change the queue type after you create a queue.

Name

Lab11-Queue-2265

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

Configuration [Info](#)

Set the maximum message size, visibility to other consumers, and message retention.

Visibility timeout [Info](#)

1

Minutes



Should be between 0 seconds and 12 hours.

Message retention period [Info](#)

4

Days



Should be between 1 minute and 14 days.

Step 3: Create AWS Lambda Function

- Go to Lambda in AWS Console.
- Click on 'Create function'
- Select 'Author from scratch'.
- Set name as 'Func-SQS-2265'.
- Set Runtime as 'Python'.
- Under 'Change default execution role':
 - Use existing execution role created in Step 1 i.e. 'Role_SQS_2265'.
- Click on 'Create function'.

School of Computer Science, Engineering and Applications (SCSEA)

B. Tech TY (CCSA)

Subject: Cloud Architecture And Protocol

Name of the Student: Sahil S. Mandawgade

PRN: 20220802265

Title of Practical: 11. Serverless Event-Driven Architecture: Amazon SQS Event Processing with AWS Lambda

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.

☐ Cont
Select

Basic information

Function name

Enter a name that describes the purpose of your function.

Func-SQS-2265

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.

☒ x86_64

☐ arm64

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.

Role_SQS_2265

[View the Role_SQS_2265 role](#) on the IAM console.

Step 4: Paste and Deploy Lambda Code

- Go to Code Section of the Function.
- Paste your Python code into the 'lambda_function.py'.
- The Code:

```
import json
```

```
import boto3
```

```
def lambda_handler(event, context):
```



School of Computer Science, Engineering and Applications (SCSEA)

B. Tech TY (CCSA)

Subject: Cloud Architecture And Protocol

Name of the Student: Sahil S. Mandawgade

PRN: 20220802265

Title of Practical: 11. Serverless Event-Driven Architecture: Amazon SQS Event Processing with AWS Lambda

```
try:
    sqsClient = boto3.client("sqs", region_name="us-east-1")
    try:
        response = sqsClient.receive_message(
            QueueUrl="<your Queue URL>",
            AttributeNames=['All'],
        )
        print(response)
        return response
    except Exception as e:
        print("Get queue message failed because ", e)
except Exception as e:
    print("Client connection to SQS failed because ", e)
```

- In our custom code change our working region (here it is “**ap-south-1**”).
- Also **add** the **SQS queue URL** in our code.
- Save and Deploy the Lambda function.

School of Computer Science, Engineering and Applications (SCSEA)

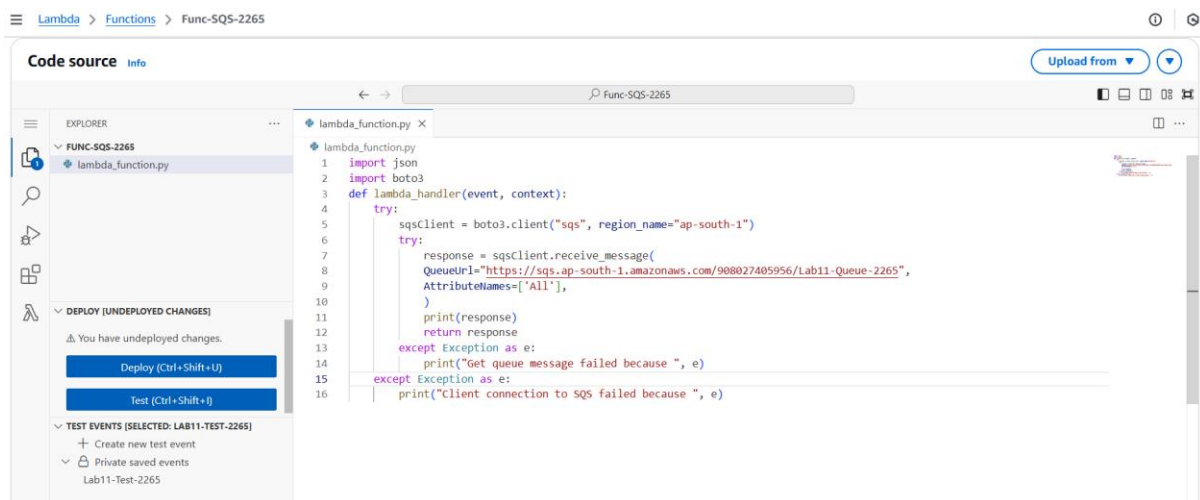
B. Tech TY (CCSA)

Subject: Cloud Architecture And Protocol

Name of the Student: Sahil S. Mandawgade

PRN: 20220802265

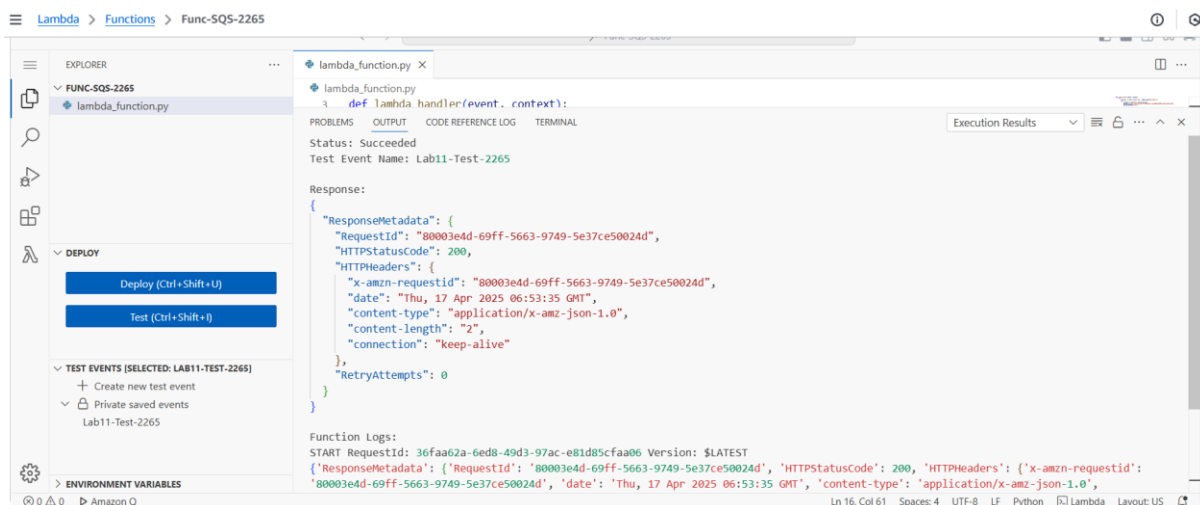
Title of Practical: 11. Serverless Event-Driven Architecture: Amazon SQS Event Processing with AWS Lambda



```
1 import json
2 import boto3
3 def lambda_handler(event, context):
4     try:
5         sqsClient = boto3.client("sqs", region_name="ap-south-1")
6         try:
7             response = sqsClient.receive_message(
8                 QueueUrl="https://sqs.ap-south-1.amazonaws.com/908027405956/Lab11-Queue-2265",
9                 AttributeNames=["All"],
10            )
11            print(response)
12            return response
13        except Exception as e:
14            print("Get queue message failed because ", e)
15    except Exception as e:
16        print("Client connection to SQS failed because ", e)
```

Step 5: Test Lambda Function Without Trigger

- In the Code section, Click on 'Test'.
- Create a new test event 'Lab11-Test-2265'.
- Save the test event.
- Verify the output.



```
Execution Results
Status: Succeeded
Test Event Name: Lab11-Test-2265

Response:
{
  "ResponseMetadata": {
    "RequestId": "80003e4d-69ff-5663-9749-5e37ce50024d",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "80003e4d-69ff-5663-9749-5e37ce50024d",
      "date": "Thu, 17 Apr 2025 06:53:35 GMT",
      "content-type": "application/x-amz-json-1.0",
      "content-length": "2",
      "connection": "Keep-alive"
    },
    "RetryAttempts": 0
  }
}
```

Step 6: Send Message to SQS and Test it.

School of Computer Science, Engineering and Applications (SCSEA)

B. Tech TY (CCSA)

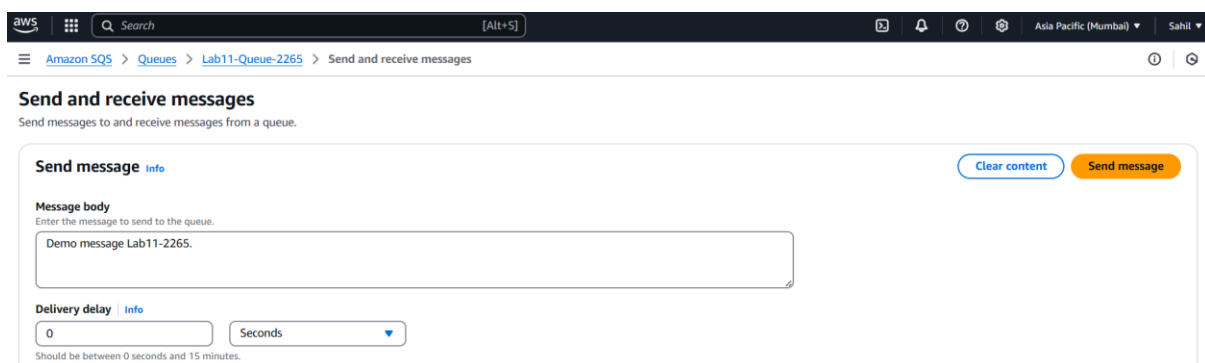
Subject: Cloud Architecture And Protocol

Name of the Student: Sahil S. Mandawgade

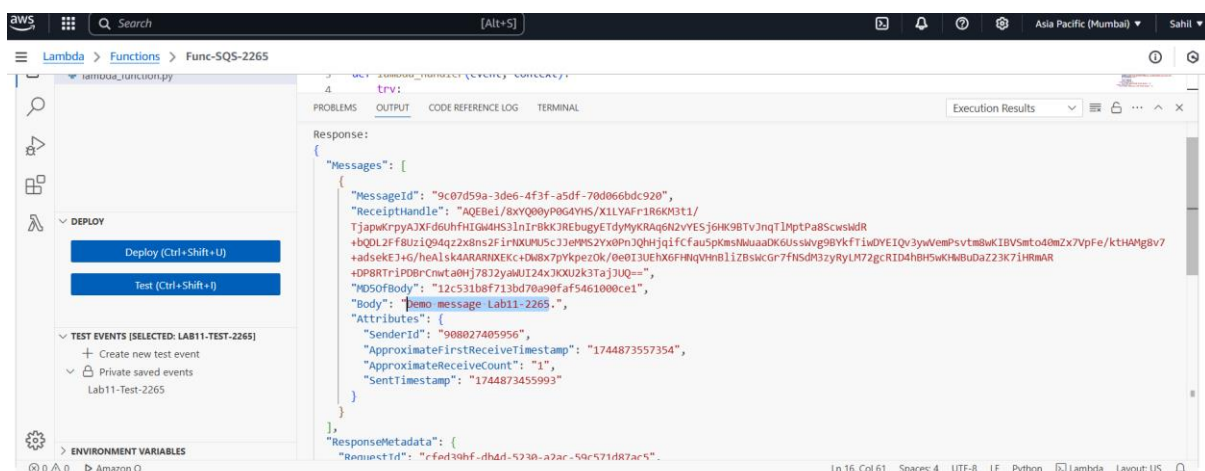
PRN: 20220802265

Title of Practical: 11. Serverless Event-Driven Architecture: Amazon SQS Event Processing with AWS Lambda

- Go to your SQS queue.
- Click on "Send and receive messages".
- In the 'Message body', enter a message like " Demo message Lab11-2265."
- In the first message, set 'Delivery delay' as '0 seconds'.
- Click on 'Send Message'.



- In Lambda, after we click on 'Test' we can see that our message has arrived.



Deletion : 'Purge' the Queue and delete the Function.