aws workshop studio (3)

The Amazon API Gateway Workshop > Module 3 - API Gateway REST Integrations > Amazon SQS > Build Amazon SQS Integration with AWS SAM and OpenAPI

Build Amazon SQS Integration with AWS SAM and OpenAPI

You can integrate an API method to put a message into SQS Queue using **AWS type integration**.

Use AWS SAM and OpenAPI to create an API Gateway REST API with SQS integration

1. Using AWS Cloud9 console, return to the root folder module-3/sqs

2. This code belongs in your SAM [template file template.yaml Review the code and then copy/paste it into the template.yaml file.

The Amazon API

Introduction

Getting Started

API Gateway

Integrations

Mock

Module Goals

► HTTP Integration

AWS Step Functions

Set up your AWS SAM Project

Integration with AWS SAM

Build Amazon SQS

and OpenAPI

Test Integration

AWS Lambda

▼ Amazon SQS

Amazon SNS

▶ Amazon Kinesis

Amazon DynamoDB

Amazon EventBridge

► Module 4 - Observability in API

► Module 6 - Enable fine-grained

access control for your APIs

► Module 5 - WebSocket APIs

▶ Private Integration

► Amazon S3

Clean up

Gateway

Clean up

Resources

with IaC

Gateway Workshop

▶ Module 1 - Introduction to Amazon

▶ Module 2 - Deploy your first API

▼ Module 3 - API Gateway REST

<

```
AWSTemplateFormatVersion: '2010-09-09'
      Transform: 'AWS::Serverless-2016-10-31'
      Description: >
         module3-lambda-rest-api: Sample SAM Template for module3-sqs-rest-api
      Globals:
          # Enable Logs
9
10
              MethodSettings:
11
                   - ResourcePath: "/*"
12
                    HttpMethod: "*"
                    DataTraceEnabled: True
13
                    LoggingLevel: INFO
14
15
                     MetricsEnabled: True
16
          Function:
17
              Timeout: 3
              Runtime: nodejs18.x
18
19
20
      Resources:
21
22
          # SQS Queue for Orders
23
          OrderQueue:
24
              Type: AWS::SQS::Queue
25
              Properties:
26
                  QueueName: OrderQueue
27
28
          # Example REST API Gateway Integrated with SQS
29
          APIWithSQSIntegration:
30
              Type: AWS::Serverless::Api
31
              Properties:
32
                  StageName: dev
33
                  OpenApiVersion: 3.0.3
                  DefinitionBody: # an OpenApi definition
34
35
                       "Fn::Transform":
                           Name: "AWS::Include"
36
37
                           Parameters:
38
                              Location: "openapi.yaml"
39
                   EndpointConfiguration:
40
                      Type: REGIONAL
41
           # IAM Role For APIGW Integration with SQS
42
43
          IAMRoleForSQSIntegration:
44
              Type: AWS::IAM::Role
45
              Properties:
                   AssumeRolePolicyDocument:
46
47
                      Version: "2012-10-17"
                      Statement:
48
                         - Effect: Allow
49
50
                           Principal:
51
                             Service:
52
                                apigateway.amazonaws.com
53
                           Action:
                             - 'sts:AssumeRole'
54
                   Policies:
55
                        PolicyName: PolicyForAPIGWSQSIntegration
56
                         PolicyDocument:
57
                           Version: "2012-10-17"
58
59
                           Statement:
                             - Effect: Allow
60
61
                               Action: 'sqs:SendMessage'
62
                               Resource:
63

    !GetAtt OrderQueue.Arn

64
           #Lambda Function to consumes SQS Queue
65
           SQSConsumerFunction:
66
              Type: 'AWS::Serverless::Function'
67
              Properties:
68
                   CodeUri: ./handlers
69
```

The resource OrderQueue defines a SQS Queue.

70

71

72

73

74

75

76 77

78

79

80 81

82

83

84

85

86 87

88 89

90

91

92

93 94

95 96

97 98 99

100

Outputs:

• The resource IAMRoleForSQSIntegration defines an IAM Role responsible for allowing API Gateway to place a message in SQS with the least possible privilege. • The SQSConsumerFunction resource defines the **Lambda** that will **consume** the request messages that Amazon API Gateway will put on SQS.

(i) Note that the SQSPollerPolicy and DynamoDBCrudPolicy polices, respectively, add permissions to the Lambda execution role to pull messages from the queue and save them in

Fn::Sub: https://\${APIWithSQSIntegration}.execute-api.\${AWS::Region}.amazonaws.com/dev/order

DynamoDB, respecting the principle of the **least-privilege**. To learn more visit AWS SAM policy templates ...

Review the code and then copy/paste it into the openapi.yaml file

3. This code belongs in your OpenAPI definition file openapi.yaml

Handler: queue-consumer.handler

TableName: !Ref OrderTable

QueueName: !GetAtt OrderQueue.QueueName

SAMPLE_TABLE: !Ref OrderTable

REGION: !Sub "\${AWS::Region}"

Queue: !GetAtt OrderQueue.Arn

Policies:

Environment:

Events:

OrderTable:

Properties:

Value:

PrimaryKey:

Name: id

Type: String ProvisionedThroughput:

- SQSPollerPolicy:

Variables:

SQSTrigger:

Type: SQS

Properties:

Type: AWS::Serverless::SimpleTable

ReadCapacityUnits: 2

WriteCapacityUnits: 2

Description: API Gateway Endpoint

- DynamoDBCrudPolicy:

```
openapi: "3.0.1"
2
     info:
        title: "module-3-sqs-integration"
        description: "API Gateway example for using SQS as direct service integration"
        version: "1.0"
6
     paths:
8
9
        /order:
10
           post:
11
              produces:
              - "application/json"
12
13
              responses:
14
                 "200":
15
                    description: "200 response"
16
                    schema:
17
                       $ref: "#/definitions/Empty"
18
              x-amazon-apigateway-integration:
19
                 credentials:
20
                   Fn::Sub: ${IAMRoleForSQSIntegration.Arn}
21
                 httpMethod: "POST"
22
23
                    Fn::Sub: "arn:aws:apigateway:${AWS::Region}:sqs:path/${AWS::AccountId}/${OrderQueue.QueueName}"
24
                 responses:
25
                    default:
26
                       statusCode: "200"
27
                 requestTemplates:
28
                    application/json: "Action=SendMessage&MessageBody=$input.body"
29
                 requestParameters:
30
                    integration.request.header.Content-Type: "'application/x-www-form-urlencoded'"
31
                 passthroughBehavior: "when_no_match"
32
                 type: "aws"
33
34 definitions:
35
        Empty:
           type: "object"
36
37
           title: "Empty Schema"
```

The x-amazon-apigateway-integration object [2] (line 18) specifies details of the backend integration used for this method. On line 20 we have \${IAMRoleForSQSIntegration.Arn}, responsible for associate the IAM Role, that will allow API Gateway put message on SQS, with the API

On line 23 we have the uri definition. For Amazon SQS, we override the path with the following format: example-account-id/example-sqs-queue-name. This definition "arn:aws:apigateway:\${AWS::Region}:sqs:path/\${AWS::AccountId}/\${OrderQueue.QueueName}" will result in this configuration and format.

On line 32 we have type: "aws", responsible for configuring the integration as AWS type to expose AWS service actions.

With AWS integration it is possible to have more control in the request and response workflow. In this example we are transforming the input payload before

In Line 28 we have the request transformation template definition Action=SendMessage&MessageBody=\$input.body with defines the action as SendMessage and put

the request body to SQS as **message**. In line 30 we have the Content-type header definition as application/x-www-form-urlencoded. Failure to provide this header will result in the following error:

UnknownOperationException.

1 sam build && sam deploy --guided

method.

Deploy the project 1. To deploy the Amazon API Gateway and the AWS Lambda to your AWS account, run the following commands from the application root module-3/sqs, where the

```
template.yaml file for the sample application is located:
```

configurations that you want SAM to have in order to get the guided deployment. You can configure as it is above. Stack Name: module-3-sqs-integration

The first time that you run the sam deploy --guided command, AWS SAM starts an AWS CloudFormation deployment. In this case, you need to say what are the

• AWS Region: Put the chosen region to run the workshop. e.g. us-east-1 🗖 Confirm changes before deploy: Y Allow SAM CLI IAM role creation: Y

passing it to SQS using Apache Velocity Template Language (VTL) .

 Disable rollback: N Save arguments to configuration file: Y SAM configuration file and SAM configuration environment leave blank

creates. Then, it will ask you to confirm the changes. Type y to confirm.

Configuring SAM deploy Looking for config file [samconfig.toml] : Found Reading default arguments : Success Setting default arguments for 'sam deploy' Stack Name [module-3-sqs-integration]: module-3-sqs-integration AWS Region [us-east-1]: #Shows you resources changes to be deployed and require a 'Y' to initiate deploy Confirm changes before deploy [Y/n]: y #SAM needs permission to be able to create roles to connect to the resources in your template Allow SAM CLI IAM role creation [Y/n]: y #Preserves the state of previously provisioned resources when an operation fails Disable rollback [y/N]: n Save arguments to configuration file [Y/n]: y SAM configuration file [samconfig.toml]: SAM configuration environment [default]:

2. After configuring the deployment, AWS SAM will display assets that will be created. But first, it will automatically upload the template to a temporary bucket it

```
CloudFormation stack changeset
                                                                           LogicalResourceId
                                                                                                                                                        ResourceType
                                                                                                                                                                                                                                    Replacement
 peration
                                                                            APIWithSQSIntegrationdevStage APIWithSQSIntegration
                                                                                                                                                        AWS::ApiGateway::Stage
AWS::ApiGateway::RestApi
                                                                                                                                                                                                                                    N/A
N/A
                                                                            IAMRoleForSQSIntegration
                                                                                                                                                        AWS::IAM::Role
                                                                            OrderTable
                                                                            SQSConsumerFunctionSQSTrigger
                                                                                                                                                        AWS::Lambda::EventSourceMapping
```

advertising. To accept or decline all non-essential cookies, choose "Accept" or "Decline." To make more detailed choices, choose "Customize."

Customize

aws workshop studio (3)

The Amazon API Gateway Workshop > Module 3 - API Gateway REST Integrations > Amazon SQS > **Test Integration**

After you create your API Gateway REST API with SQS integration, you can test the API.

3. In the **Resources** pane, you can select the method you want to test. Click on the POST method.

Test SQS Integration using API Gateway console

1. Open the Amazon API Gateway console and sign in.

API Gateway > APIs > Resources - module-3-sqs-integration (cg03f3foag)

Resources

⊡ ∠

Create resource

POST

2. Choose your REST API named, module-3-sqs-integration.

Client

Authorization

Request validator

Request paths (0)

NONE

None

Method request settings

/order - POST - Method execution

Method request

The Amazon API **Gateway Workshop**

<

Test Integration

- Introduction
- Getting Started ▶ Module 1 - Introduction to Amazon
- **API Gateway**
- ▶ Module 2 Deploy your first API with IaC
- ▼ Module 3 API Gateway REST
- Integrations Module Goals
- Mock
- ► HTTP Integration AWS Lambda
- ▼ Amazon SQS Build Amazon SQS

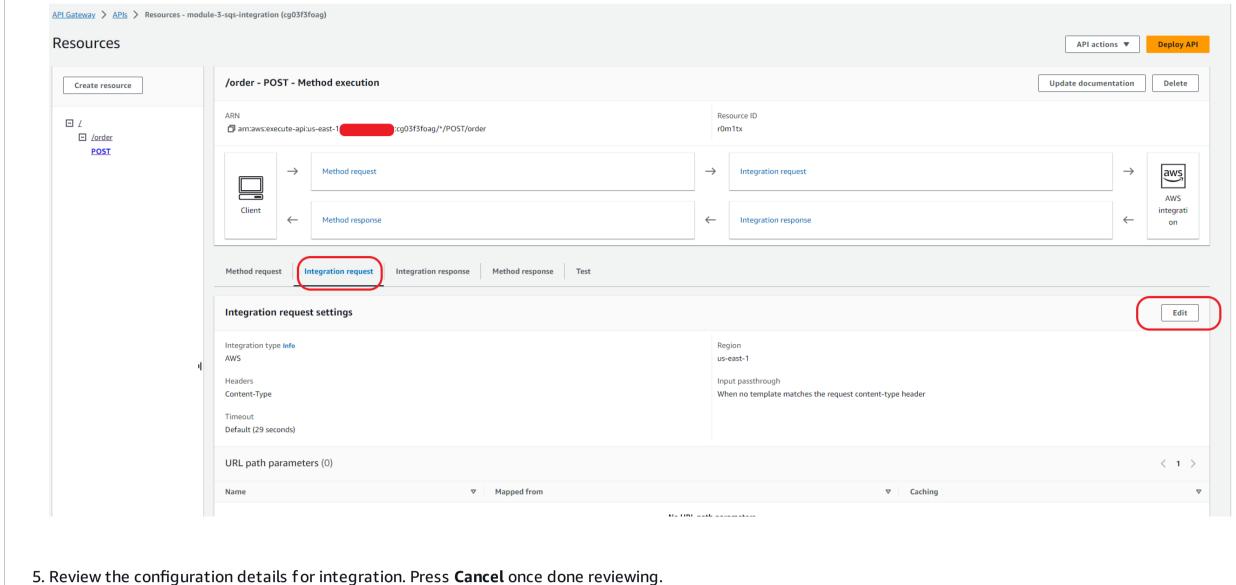
► AWS Step Functions

- Set up your AWS SAM Project Integration with AWS SAM and OpenAPI
- **Test Integration** Amazon SNS
- Amazon Kinesis ► Amazon DynamoDB
- ► Amazon EventBridge
- ▶ Private Integration
- ► Amazon S3

Clean up

- ► Module 4 Observability in API Gateway
- ► Module 5 WebSocket APIs
- ► Module 6 Enable fine-grained access control for your APIs
- Clean up
- Resources

4. Choose the **Integration request** tab, press **Edit**.



API key required

SDK operation name

Generated based on method and path

False

API actions ▼

Update documentation

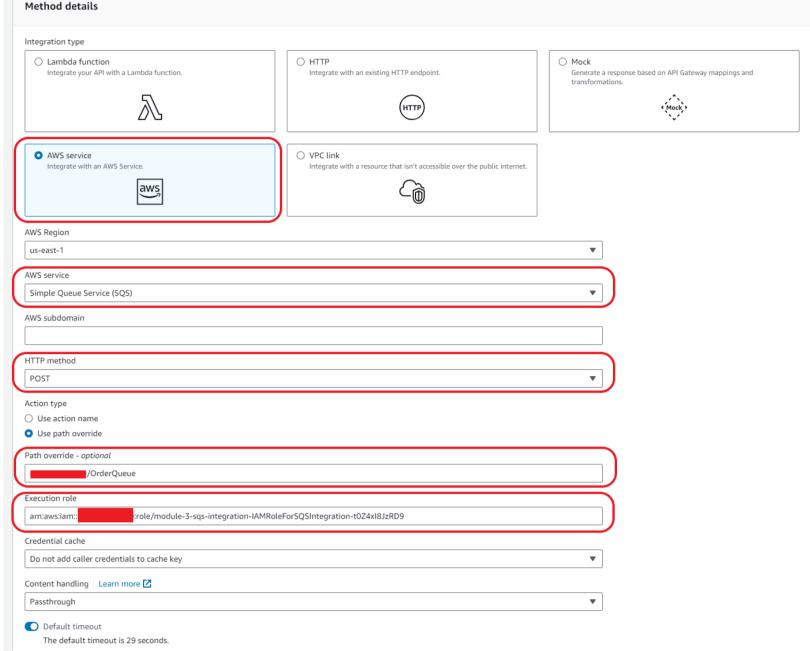
Deploy API

Edit

Method details

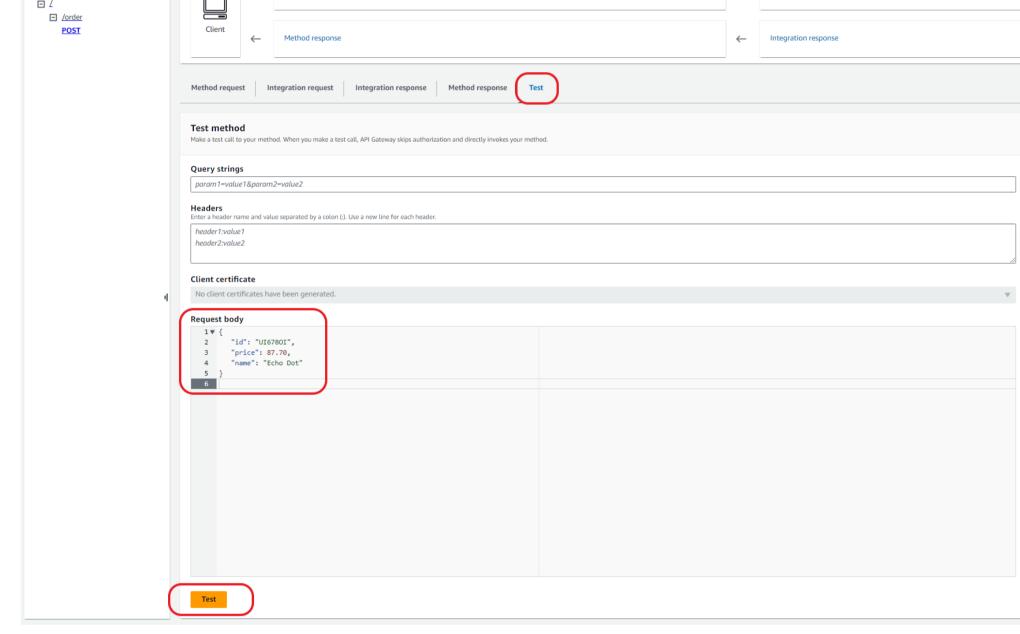
→ Method request

Edit integration request



⊡ /

6. Choose the **Test** tab.



→ Integration request

For Request Body copy: 1 {

> "id": "UI6780I", "price": 87.70, "name": "Echo Dot"

```
7. Click on Test button. Leave the other fields blank.
8. The test result are showed in the log. You can see the function called and the response:
     Create resource
                            /order - POST method test results
      /order
```

40d6-a131-6fc6c5ff6cde", "SequenceNumber": null}}} "Content-Type": "application/json", "X-Amzn-Trace-Id": "Root=1-653145c0-a12165ef71d3466e149c7ae1" Execution log for request 2fbb8865-67c3-4a6b-b400-d5cc73ea20be Thu Oct 19 15:05:36 UTC 2023 : Starting execution for request: 2fbb8865-67c3-4a6b-b400-d5cc73ea20be Thu Oct 19 15:05:36 UTC 2023 : HTTP Method: POST, Resource Path: /order Thu Oct 19 15:05:36 UTC 2023 : Method request path: {} Thu Oct 19 15:05:36 UTC 2023 : Method request query string: {} Thu Oct 19 15:05:36 UTC 2023 : Method request headers: $\{\}$ Thu Oct 19 15:05:36 UTC 2023 : Method request body before transformations: { "price": 87.70, 9. Now let's check an important detail of the integration. Let's understand the SQS message format that API Gateway puts on SQS. Let's explore SQS Lambda

11. Click on **Log Groups**, and then click on a lambda log group with the prefix module-3-sqs-integrati... Log groups (1)

By default, we only load up to 10000 log groups.

```
Q Filter log groups or try prefix search
                    Log group
                    /aws/lambda/module-3-sqs-integration-SQSConsumerFunction-kIG88QV5CnfN
Log streams
                  Metric filters
                                    Subscription filters
                                                             Contributor Insights
                                                                                                 Data protection - new
                                                                                      Tags
```

2023-03-27T16:25:41.644Z

12. Click on the most recent logstream

consumer logs.

10. Navigate to CloudWatch [in the AWS console.

```
Log streams (1)
                                   Q Filter log streams or try prefix search
                                                                                                                          Exact match Show expire
                                         Log stream
                                         2023/03/27/[$LATEST]ce7ce3a484d3434ea339ec45dd9c2af2
13. Look for INFO Received event and take a look at the Lambda input event payload (SQS Message):
               2023-03-27T13:25:41.657-03:00
                                                 2023-03-27T16:25:41.644Z 74c0a0ee-6893-57a4-8334-fc75ad50b630 INFO Received event: {"Records":[{"messageId":"1b8dd566-f47a-454c-9b92-
```

"Records": ["messageId": "1b8dd566-f47a-454c-9b92-bed296a2e96f",

```
"receiptHandle": "AQEB1rPv0XSa+CxrB05bKr2t79DPnL+J2gulGunNn7kAaTcoV4vw69r96JVSwjWeZ9wBaBQrCriPofaKmlZtRByZ6/eR3HrH+Noy53kIZL1dHLZIBwds885VQ2sX166UKR00uWXsWLkJ
                /6quKB5Y44r3T1rzqHlW5VmnZ01pt5uR47RNZH20ZmCzEmra1mdy865yh/gDHjE+oc0Y/FM39Hwlsd94NqVkghbuXm867HegWdeP6s6Dxd2iTZUa00RkJhvPUyvAnDegeVT5iwX1UhISGXG30QkOKN0QJclIZ8
               /I6HyUbFYERJq9iYoWFOz10RAhOxLZ7G1kVR0TYpLS+841GZ8hjnRQ5fWLZpxSY0NX148Lb1nJ",
                            "body": "{\n\t\"id\": \"UI6780I\", \n\t\"price\": 87.70, \n\t\"name\": \"Echo Dot\"\n}",
                           "attributes": {
                               "ApproximateReceiveCount": "1",
                               "AWSTraceHeader": "Root=1-6421c384-0c47c997ba2df2a9213e094c",
                               "SentTimestamp": "1679934340908",
                               "SenderId": "AROATMSPEL5CS7MXPIGYS:BackplaneAssumeRoleSession",
                               "ApproximateFirstReceiveTimestamp": "1679934340915"
                            "messageAttributes": {},
                            "md50fBody": "5ab83bbc02e360aa6931ce62d66a0c5c",
                            "eventSource": "aws:sqs",
                            "eventSourceARN": "arn:aws:sqs:us-east-1 @@@df@df@df@df@derQueue",
                            "awsRegion": "us-east-1"
                  ]
               2023-03-27T13:25:41.657-03:00
                                                   2023-03-27T16:25:41.657Z 74c0a0ee-6893-57a4-8334-fc75ad50b630 INFO Received message 1b8dd566-f47a-454c-9b92-bed296a2e96f from SQS:
               2023-03-27T13:25:41.657-03:00
                                                   2023-03-27T16:25:41.657Z 74c0a0ee-6893-57a4-8334-fc75ad50b630 INFO { "id": "UI6780I", "price": 87.70, "name": "Echo Dot" }
               2023-03-27T13:25:42.139-03:00
                                                   2023-03-27T16:25:42.139Z 74c0a0ee-6893-57a4-8334-fc75ad50b630 INFO Response: {"statusCode":200, "headers":{"Content-Type":"application
Test the deployed API using cURL
```

74c0a0ee-6893-57a4-8334-fc75ad50b630 INFO Received event:

2. Copy the following cURL command and paste it into the terminal window, replacing <api-id> with your APIs API ID and <region> with the region where your API is deployed. You may have copied this URL in from the CloudFormation output in the last step. You can also find the full invoke URL in the API Gateway console by navigating to **Stages > dev**.

3. The **Response Body** output should be

1. Open a new terminal window in your AWS Cloud9 environment.

curl -X POST https://<api-id>.execute-api.<region>.amazonaws.com/dev/order \ 2 -H 'Content-Type: application/json' \ 3 -d '{"id": "GF854123","price": 280.00,"name": "Karaoke"}'

```
"SendMessageResponse":
        "ResponseMetadata":
          "RequestId": "83e347ef-8943-5068-a500-1b74442e02e6"
       "SendMessageResult":
8
9
           "MD50fMessageAttributes":null,
10
           "MD50fMessageBody":"f5373bd65c4ef0163e148862fb499944",
11
           "MD50fMessageSystemAttributes":null,
12
           "MessageId": "ac4891c8-eca7-4ad3-8177-022bde2ce75c",
13
           "SequenceNumber":null
14
```

15 16 17 } (i) Tip

When testing a method using the API Gateway console, logs are not sent to CloudWatch Logs. When calling externally, for example via cURL, the logs are sent normally.

advertising. To accept or decline all non-essential cookies, choose "Accept" or "Decline." To make more detailed choices, choose "Customize."

If you agree, AWS and approved third parties will also use cookies to provide useful site features, remember your preferences, and display relevant content, including relevant

Find the payload logs **before** and **after** the **request** and **response** transformations in **CloudWatch**.

⚠ Challenge

Customize