AWS Lambda + SNS + SQS Integration Guide

Step-by-Step Guide: Integrating React Form with AWS Lambda + SNS + SQS

Introduction

This guide walks you through building a React frontend that sends a user message to AWS Lambda, which then flows through SNS to SQS and triggers Processor Lambdas.

Step-by-Step Setup

Step 1: Set Up AWS Lambda Function

1. Go to AWS Console > Lambda > Create Function.
2. Name: PublishToSNSFunction.
3. Runtime: Python 3.13.
4. Create a role with AWSLambdaBasicExecutionRole and AmazonSNSFullAccess.
5. Set environment variable: SNS_TOPIC_ARN = <Your SNS Topic ARN>.

Lambda Code:

```
import boto3
import os
import json

def lambda_handler(event, context):
    sns_client = boto3.client('sns')
    topic_arn = os.environ.get('SNS_TOPIC_ARN')

    if event['httpMethod'] == 'OPTIONS':
        return {
            'statusCode': 200,
            'headers': {
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST,GET",
                "Access-Control-Allow-Headers": "Content-Type"
            },
            'body': json.dumps('Preflight OK')
        }
```

```
    if 'body' not in event or not event['body']:
        return {
            'statusCode': 400,
            'headers': {
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST,GET",
                "Access-Control-Allow-Headers": "Content-Type"
            },
            'body': json.dumps({'error': 'Missing message body'})
        }

    try:
        body = json.loads(event['body'])
        user_message = body.get('message', 'Default message')

        response = sns_client.publish(
            TopicArn=topic_arn,
            Message=user_message,
            Subject='User Submitted Message'
        )

        return {
            'statusCode': 200,
            'headers': {
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST,GET",
                "Access-Control-Allow-Headers": "Content-Type"
            },
            'body': json.dumps({'messageId': response['MessageId']})
        }

    except Exception as e:
        print(f"Error occurred: {e}")
        return {
            'statusCode': 500,
```

```
            'headers': {
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST,GET",
                "Access-Control-Allow-Headers": "Content-Type"
            },
            'body': json.dumps({'error': 'Internal server error'})
        }
```

Step 2: Attach API Gateway to Lambda

1. Go to Lambda > Triggers > Add Trigger.
2. Select API Gateway > Create New API > REST API.
3. Deploy and note down the Invoke URL.

Example: https://your-api-id.execute-api.region.amazonaws.com/default/PublishToSNSFunction

Step 3: Create React Frontend

App.js Example Code:

```
import { useState } from 'react';

function App() {
  const [message, setMessage] = useState('');
  const [response, setResponse] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();

                                                const             apiUrl             =
'https://your-api-id.execute-api.region.amazonaws.com/default/PublishToSNSFunction';

    try {
      const res = await fetch(apiUrl, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ message })
      });
```

```
      const data = await res.json();
      setResponse(`Message sent successfully! ID: ${data.messageId}`);
    } catch (err) {
      setResponse('Error sending message.');
      console.error(err);
    }
  };

  return (
    <div>
      <h2>Send a Message to AWS Lambda</h2>
      <form onSubmit={handleSubmit}>
        <textarea value={message} onChange={(e) => setMessage(e.target.value)} />
        <button type="submit">Send Message</button>
      </form>
      {response && <p>{response}</p>}
    </div>
  );
}

export default App;
```

How It Works

1. React submits form data to API Gateway.
2. API Gateway invokes Lambda.
3. Lambda publishes message to SNS.
4. SNS fans-out to SQS queues.
5. SQS triggers Processor Lambda functions.
6. Processor Lambdas log received messages.

Final Checklist

- [x] Lambda handles POST and OPTIONS.

AWS Lambda + SNS + SQS Integration Guide

- [x] API Gateway integrated with Lambda Proxy.
- [x] CORS enabled via Lambda responses.
- [x] React connects successfully to API Gateway.

Congratulations!

You have built a fully serverless React-to-AWS backend integration.