AWS Serverless Contact Form Project — Full Step-by-Step Documentation

Project Overview

Project Title: Contact Form API Using API Gateway + Lambda + SES

Objective: Create a serverless contact form where users can submit inquiries that automatically trigger an email to a specified recipient using AWS services.

Step 1: SES Setup

1. Log in to AWS Console → SES (Simple Email Service).
2. Verify two email addresses using **identities**:
    a. Sender email (SOURCE_EMAIL)
    b. Recipient email (DEST_EMAIL)
3. Ensure SES is in the correct region (e.g.: us-west-1).
4. No sandbox limitations if emails are verified.

Goal: SES verifies two or more emails and is ready for email delivery.

Step 2: Lambda Function Creation

2.1 Create IAM Role for Lambda Service

- Navigate to IAM Dashboard on console.
- Select Lambda as AWS Service for role.
- Role Name: Lambda-SESSendEmail-Role (or any as desired)
- Attached Policies:
    - AmazonSESFullAccess (or custom SES policy such AmazonSESReadOnlyAccess for least access privilege).
    - CloudWatchLogsFullAccess.

Goal: Ensure Lambda has attached permissions to invoke SES and write logs.

2.2 Create Lambda Function

- Function Name: ContactFormHandler (or any as desired)
- Runtime: Python 3.12 (*Runtime environment changes based on language*).
- Permissions: Use the IAM role created above for Lambda service.

2.3 Lambda Code

- Write Python code (or other, e.g.: Node.JS) in Code editor or upload zip file to:
    - Parse JSON body from API Gateway
    - Send email via SES
    - Return success or error response

2.4 Environment Variables

Key	Value
SES_REGION	us-west-1
SOURCE_EMAIL	verified_sender@example.com (type in verified email in SES)
DEST_EMAIL	verified_recipient@example.com (type in verified email in SES)

2.5 Lambda Testing

- Deploy Lambda function in Code Editor.
- Use Lambda Test event with JSON payload to test Lambda function:

```
{
  "body": "{\"name\": \"<Insert any name>\", \"email\":<\Insert verified SES DEST_EMAIL\>",
\"message\": \"Insert any message\"}"
}
```

Example using fabricated details:

```
{
  "body": "{\"name\": \"John Doe\", \"email\":\"john@example.com\", \"message\": \"Hello
from Lambda test!\"}"
}
```

- Receive a 200 OK response.
- Email successfully delivered to recipient.

Progress: Lambda is fully functional.

Step 3: API Gateway Setup

3.1 Create REST API

- Navigate to API Gateway in AWS console → APIs.
- Name: ContactFormAPI (or any as desired)
- Endpoint type: Regional

3.2 Create Resource + Method

- Create a Resource with Path: /contact (or any as desired)
- Select the resource and Create Method: POST
- Integration: Choose Lambda Function → ContactFormHandler (or name you selected) or paste its ARN.
- Enable CORS to display **OPTIONS.**

3.3 Deploy API

- Select Deploy API.
- Create a Stage → Stage: New Stage, Stage Name: dev (or any as desired, e.g.: prod)
- Endpoint URL: Use Invoke Url in POST method under the **Stages**.
  e.g.: https://4q5ln4zyoi.execute-api.us-west-1.amazonaws.com/dev/contact

3.4 Test with Postman

- Method: POST
- Headers: Content-Type: application/json
- Body (Select raw and JSON format):

```
{
 "name": "John Doe",
 "email": "johndoe@example.com",
 "message": "Test message to confirm Lambda Proxy Integration and SES email delivery."
}
```

- Receive a 200 OK response.
- Email delivered to DEST_EMAIL.

Goal: API Gateway connected and fully functional.

- Confirmed correct configuration for Lambda Proxy Integration:
- Sent test requests → received 200 OK + email.

Progress: End-to-end flow validated.

Conclusion & Next Steps

- Serverless contact form fully functional: API Gateway → Lambda → SES.
- Emails reliably delivered and logged in CloudWatch.
- Future improvements:
  - Design and host web platform to accept user response.
  - Add CAPTCHA to prevent spam
  - Deploy SES in production with domain and DKIM
  - Rate-limiting in API Gateway
  - Store submissions in DynamoDB