Serverless Contact Form on AWS - Project

# 1. Project Title

Serverless Contact Form using AWS Lambda, API Gateway, DynamoDB, and SES

# 2. Project Overview

A serverless contact form where users can submit their details (Name, Email, Message). The system: - Stores messages in DynamoDB. - Sends email notifications using Amazon SES. - Uses AWS Lambda for serverless backend logic. - Provides a frontend form hosted on S3.

3. Architecture Diagram

A diagram of a computer application

AI-generated content may be incorrect.

# 4. AWS Services Used

| Service | Purpose |
| --- | --- |
| S3 | Hosting frontend HTML form |
| API Gateway | Exposes HTTP API endpoint |
| Lambda | Backend function to handle form submissions |
| DynamoDB | Stores contact messages |
| SES | Sends notification emails |

# 5. Implementation Steps

## Step 1: Create DynamoDB Table

* Table name: ContactMessages
* Partition key: id (String)
* On-demand capacity

## A screenshot of a computer AI-generated content may be incorrect.

## Step 2: Create Lambda Function

* Name: ContactFormHandler
* Runtime: Python 3.9
* Role: Lambda with permissions for DynamoDB and SES

A screenshot of a computer

AI-generated content may be incorrect.

**Code Snippet:**

import json  
import boto3  
import uuid  
  
dynamodb = boto3.resource('dynamodb')  
ses = boto3.client('ses', region\_name="ap-south-1")  
table = dynamodb.Table('ContactMessages')  
  
def lambda\_handler(event, context):  
 body = json.loads(event['body'])  
 message\_id = str(uuid.uuid4())  
 table.put\_item(Item={  
 'id': message\_id,  
 'name': body['name'],  
 'email': body['email'],  
 'message': body['message']  
 })  
 ses.send\_email(  
 Source="your-verified-email@domain.com",  
 Destination={'ToAddresses': ["your-verified-email@domain.com"]},  
 Message={'Subject': {'Data': 'New Contact Form Submission'},  
 'Body': {'Text': {'Data': f"Name: {body['name']}\nEmail: {body['email']}\nMessage: {body['message']}"}}}  
 )  
 return {'statusCode': 200, 'headers': {"Access-Control-Allow-Origin": "\*"},  
 'body': json.dumps({'message': 'Form submitted successfully!'})}

## Step 3: Verify Email in SES

* Sender and recipient emails must be verified

A screenshot of a computer

AI-generated content may be incorrect.

## Step 4: Create API Gateway HTTP API

* Route: POST /contact
* Integration: Lambda (ContactFormHandler)
* A screenshot of a computer

  AI-generated content may be incorrect.Deploy stage: dev

## Step 5: Frontend HTML Form

A screenshot of a computer

AI-generated content may be incorrect.

**Code Snippet:**

<form id="contactForm">  
 <input type="text" name="name" placeholder="Your Name" required><br>  
 <input type="email" name="email" placeholder="Your Email" required><br>  
 <textarea name="message" placeholder="Your Message" required></textarea><br>  
 <button type="submit">Send</button>  
</form>  
  
<script>  
document.getElementById("contactForm").addEventListener("submit", async (e) => {  
 e.preventDefault();  
 const formData = {  
 name: e.target.name.value,  
 email: e.target.email.value,  
 message: e.target.message.value  
 };  
 const res = await fetch("https://n87uwdwjqb.execute-api.ap-south-1.amazonaws.com/dev/contact", {  
 method: "POST",  
 headers: {"Content-Type": "application/json"},  
 body: JSON.stringify(formData)  
 });  
 const data = await res.json();  
 alert(data.message);  
});  
</script>

# 6. Testing

**Steps:** 1. Open the HTML page locally or from S3.

2. Fill the form and submit.

3. Check DynamoDB for new items.

4. Check SES email for notification.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

# 7. Challenges & Solutions

| Issue | Solution |
| --- | --- |
| CORS error when testing locally | Enable CORS in API Gateway & host HTML on S3 |
| “Not Found” error on URL | Ensure /contact route exists and stage is deployed |
| SES sandbox email limitation | Verify sender & recipient emails |

# 8. Conclusion

This project demonstrates a fully serverless architecture: - No servers to manage - Scalable using AWS Lambda - Fast deployment of API and frontend - Automated notifications via SES