Lab 03: Build a Serverless Contact Form

Lab Overview

In this lab you will build a real, working serverless contact form. A visitor fills out a form on your S3-hosted website, clicks Submit, and within seconds you receive an email in your inbox. No web servers. No monthly EC2 bill. Pure event-driven cloud architecture.

◆ **PREREQ:** Complete Lab 01 (S3 Static Website Hosting) before starting this lab. You need a working S3 static website to host the contact form page.

Service	Purpose	Free Tier
Amazon S3	Hosts your static HTML contact form page	Yes
AWS Lambda	Serverless function that processes the form submission	Yes
Amazon API Gateway	Creates the HTTPS endpoint your form posts data to	Yes
Amazon SES	Simple Email Service — sends the email to your inbox	Yes
AWS IAM	Grants Lambda permission to call SES securely	Yes

■ **NOTE:** Lambda: 1M free requests/month · API Gateway: 1M calls/month · SES: 62,000 emails/month from Lambda.

■ **WARNING:** SES sandbox mode: You can only send emails to verified addresses. Verify your email in Step 1.

Prerequisites

- An active AWS account (free tier is sufficient)
- Completed Lab 01 — you have a working S3 static website
- A working email address you can verify in SES

1 Amazon Simple Email Service (SES) Verify Your Email Address

1. Search for SES in the AWS Console → click Simple Email Service
2. Left sidebar → Verified identities → Create identity
3. Identity type: Email address → enter your notification email → Create identity
4. Open your email inbox and click the verification link from AWS
5. Return to SES console and refresh — status should show Verified

✓ **TIP:** Keep the SES console tab open — you will need your verified email in Step 3.

2 AWS Identity & Access Management (IAM) Create an IAM Role for Lambda

1. Search for IAM → Left sidebar → Roles → Create role

2. Trusted entity: AWS service | Use case: Lambda → Next
3. Search for and add: AmazonSESFullAccess
4. Search for and add: AWSLambdaBasicExecutionRole (allows Lambda to write logs)
5. Click Next → Role name: LambdaSESRole → Create role

■ **NOTE:** AWSLambdaBasicExecutionRole lets Lambda write CloudWatch logs — essential for debugging.

3 AWS Lambda
Create the Lambda Function

Create the function

1. Search for Lambda → Create function → Author from scratch
2. Function name: ContactFormHandler | Runtime: Python 3.12
3. Expand Change default execution role → Use an existing role → select LambdaSESRole
4. Click Create function

Add the code — replace both email addresses with your verified SES email

```
import json, boto3
ses = boto3.client('ses', region_name='us-east-1')
SENDER_EMAIL    = 'your-verified-email@example.com'  # replace
RECIPIENT_EMAIL = 'your-verified-email@example.com'  # replace

def lambda_handler(event, context):
    try:
        body = json.loads(event.get('body', '{}'))
        name    = body.get('name', 'Unknown')
        email   = body.get('email', 'Unknown')
        message = body.get('message', 'No message')
        ses.send_email(
            Source=SENDER_EMAIL,
            Destination={'ToAddresses': [RECIPIENT_EMAIL]},
            Message={
                'Subject': {'Data': f'New contact from {name}'},
                'Body': {'Text': {'Data': f'Name: {name}\nEmail: {email}\nMessage: {message}'}}
            }
        )
        return {'statusCode': 200,
                'headers': {'Access-Control-Allow-Origin': '*'},
                'body': json.dumps({'message': 'Email sent!'}) }
    except Exception as e:
        return {'statusCode': 500,
                'headers': {'Access-Control-Allow-Origin': '*'},
                'body': json.dumps({'error': str(e)}) }
```

■ **WARNING:** Replace both 'your-verified-email@example.com' with the email you verified in Step 1.

Deploy and test

1. Click Deploy to save the code
2. Click Test → Create new test event → Name: TestContact → replace JSON with:

```
{
  "body": "{\"name\": \"Test User\", \"email\": \"test@example.com\", \"message\": \"Hello from Lambda!\"}",
  "httpMethod": "POST"
}
```

3. Click Test — you should see Execution result: succeeded

4. Check your inbox — you should have received the test email

✓ **TIP:** If you received the test email, your Lambda function is working correctly.

4 Amazon API Gateway
Create an API Gateway Endpoint

1. Search for API Gateway → Create API → HTTP API → Build
2. Add integration → Lambda → select ContactFormHandler
3. API name: ContactFormAPI → Next
4. Method: POST | Resource path: /contact → Next
5. Stage name: prod → Next → Create
6. Copy the Invoke URL from the API summary page and save it

Enable CORS

1. Left sidebar → CORS
2. Access-Control-Allow-Origin: *
3. Access-Control-Allow-Methods: POST, OPTIONS → Save

■ **NOTE:** CORS must be enabled so your S3 website can call the API endpoint.

5 Amazon S3
Add the Contact Form to Your Website

Create contact.html with the code below. Replace YOUR_API_GATEWAY_URL with the Invoke URL from Step 4.

```
<!DOCTYPE html><html lang='en'><head>
  <meta charset='UTF-8'/><title>Contact Us</title>
  <style>body{font-family:Arial,sans-serif;max-width:600px;margin:60px auto;padding:0 20px}
    input,textarea{width:100%;padding:10px;margin-bottom:12px;border:1px solid #ccc}
    button{background:#1B2A4A;color:#fff;border:none;padding:10px 24px;cursor:pointer}
  </style></head><body>
  <h1>Contact Us</h1>
  <input type='text'  id='name'    placeholder='Your name'/>
  <input type='email' id='email'   placeholder='your@email.com'/>
  <textarea id='message' rows='5'  placeholder='Your message'></textarea>
  <button onclick='submitForm()'>Send Message</button>
  <p id='status'></p>
  <script>
    const API = 'YOUR_API_GATEWAY_URL/contact'; // replace this
    async function submitForm() {
      const d = {name:document.getElementById('name').value,
                 email:document.getElementById('email').value,
                 message:document.getElementById('message').value};
      const s = document.getElementById('status');
      s.textContent='Sending...';
      try { await fetch(API,{method:'POST',
              headers:{'Content-Type':'application/json'},body:JSON.stringify(d)});
            s.style.color='green'; s.textContent='Message sent!'; }
      catch(e){ s.style.color='red'; s.textContent='Error. Try again.'; }
    }
  </script></body></html>
```

■ **WARNING:** Replace YOUR_API_GATEWAY_URL with the full Invoke URL from Step 4.

Upload to S3 and test

1. S3 → your bucket → Upload → add contact.html → Upload
2. Open your site URL and add /contact.html at the end
3. Fill in the form and click Send Message
4. Check your inbox — you should receive the email within seconds

✓ **TIP:** Congratulations! You have built a fully serverless contact form at virtually zero cost.

What You Learned

- Serverless compute — Lambda runs only when triggered, with no server to manage or pay for at idle
- Event-driven architecture — a form submit triggers a chain of AWS services automatically
- IAM roles and least privilege — Lambda granted only the permissions it needed
- Managed services — SES, API Gateway, Lambda are fully managed by AWS
- Pay-per-use pricing — you only pay when the function runs; effectively free at low volumes
- CORS — Cross-Origin Resource Sharing controls which websites can call your API

Lab Cleanup — Delete Your Resources

✗ **IMPORTANT:** Always delete lab resources when done to avoid unexpected charges.

#	Resource	How to Delete
1	Lambda Function	Lambda → Functions → ContactFormHandler → Actions → Delete
2	API Gateway	API Gateway → APIs → ContactFormAPI → Actions → Delete
3	SES Verified Identity	SES → Verified identities → select email → Delete identity
4	IAM Role	IAM → Roles → search LambdaSESRole → Delete
5	S3 contact.html	S3 → your bucket → select contact.html → Delete
6	CloudWatch Logs	CloudWatch → Log groups → delete /aws/lambda/ContactFormHandler