React-Based Portfolio Website on AWS Cloud Architecture

Introduction

This project showcases the development of a React-based portfolio website leveraging AWS cloud architecture. The solution integrates modern web development practices with cloud engineering principles to deliver a scalable, secure, and responsive portfolio application. The project utilizes React (via Vite.js) for frontend development and several AWS services for hosting, security, and functionality.

Project Overview

Technology Stack

- **Frontend**: React with Vite.js, Bootstrap for styling and responsiveness.
- **Backend**: Serverless architecture powered by AWS Lambda and API Gateway.
- **Hosting and Distribution**: Amazon S3 for static site hosting, Amazon CloudFront for content delivery.
- **Domain Management**: Amazon Route 53 for DNS configuration and custom domains.
- **Security**: AWS Certificate Manager (ACM) for SSL/TLS certificates, CloudFront for HTTPS enforcement.
- **Automation**: GitHub Actions CI/CD pipeline for seamless deployment.
- **Interactivity**: AI-powered chatbot using Amazon Lex, with context managed in DynamoDB.
- **Contact Handling**: API Gateway, Lambda, and Amazon SES for handling contact form submissions.

Development Process

1. Frontend Development

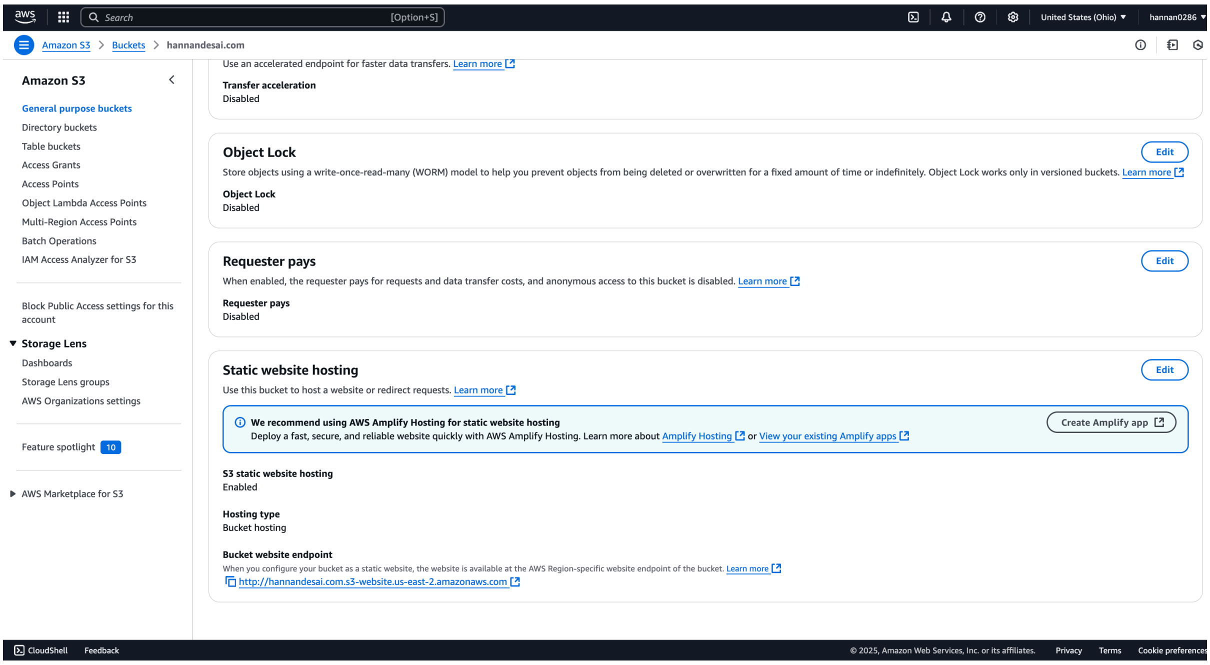
The website frontend was developed using React, a powerful JavaScript library for building user interfaces. Vite.js was used as the build tool, providing faster development and optimized production builds. Key steps include:

- **Project Initialization**: The project was bootstrapped using `npm create vite@latest`. React was selected as the framework, and dependencies were installed using `npm install`.

- **Component-Based Design**: The site features reusable components for scalability and maintainability.
- **Styling**: Bootstrap was integrated for responsive and professional styling.
- **Production Build**: The production build was generated using npm run build, outputting optimized static assets.

2. Hosting on AWS S3 and CloudFront

- **Static Hosting**: The built React application was hosted on Amazon S3, leveraging its scalability and cost-efficiency for static websites.



Use an accelerated endpoint for faster data transfers. Learn more

Transfer acceleration
Disabled

Object Lock

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. Learn more

Object Lock
Disabled

Requester pays

When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. Learn more

Requester pays
Disabled

Static website hosting

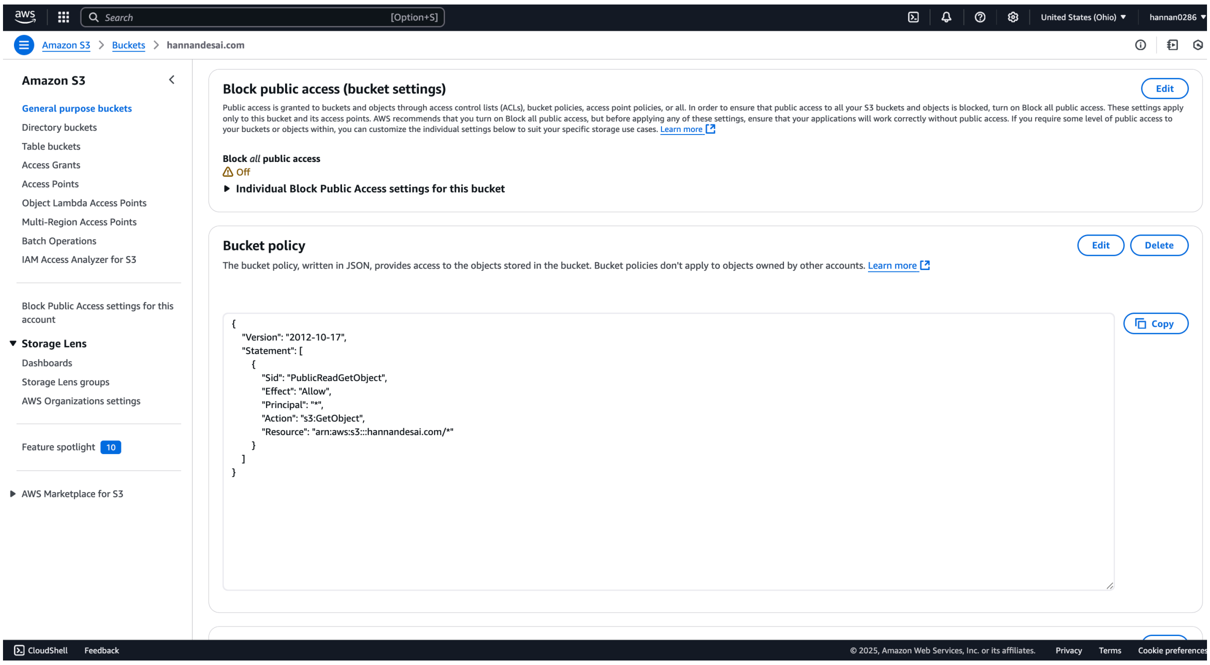
Use this bucket to host a website or redirect requests. Learn more

We recommend using AWS Amplify Hosting for static website hosting
Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about Amplify Hosting or View your existing Amplify apps

S3 static website hosting
Enabled

Hosting type
Bucket hosting

Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Learn more
http://hannandesai.com.s3-website.us-east-2.amazonaws.com



Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

Block *all* public access
⚠ Off

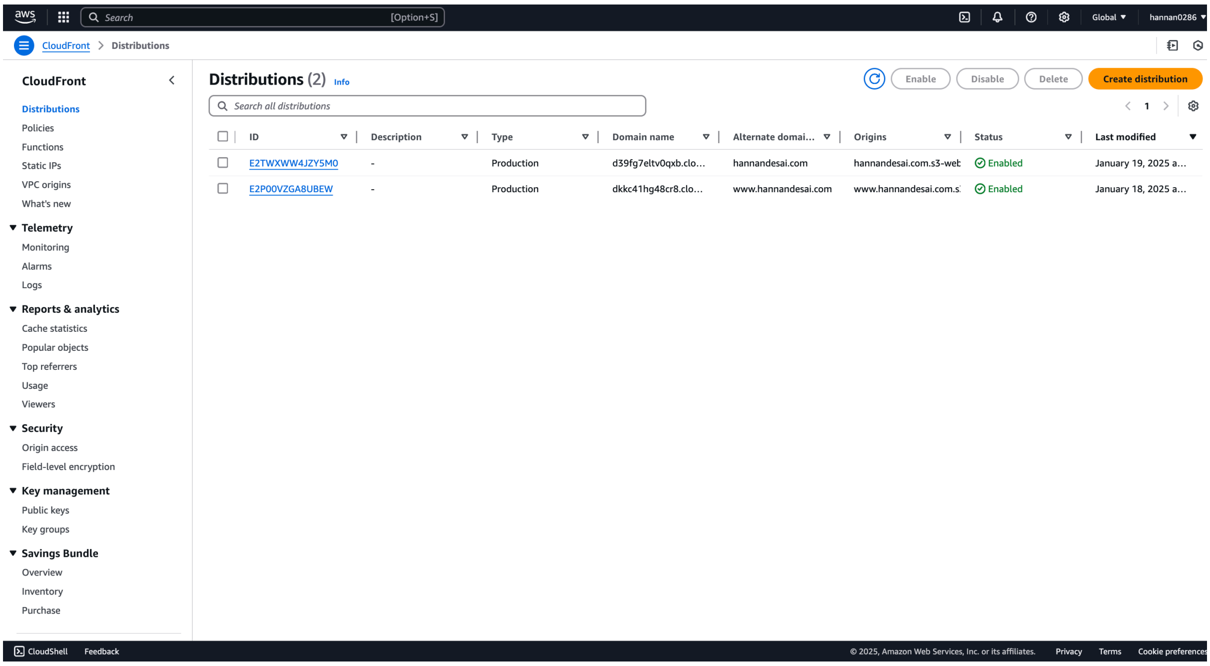
▶ **Individual Block Public Access settings for this bucket**

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::hannandesai.com/*"
    }
  ]
}
```

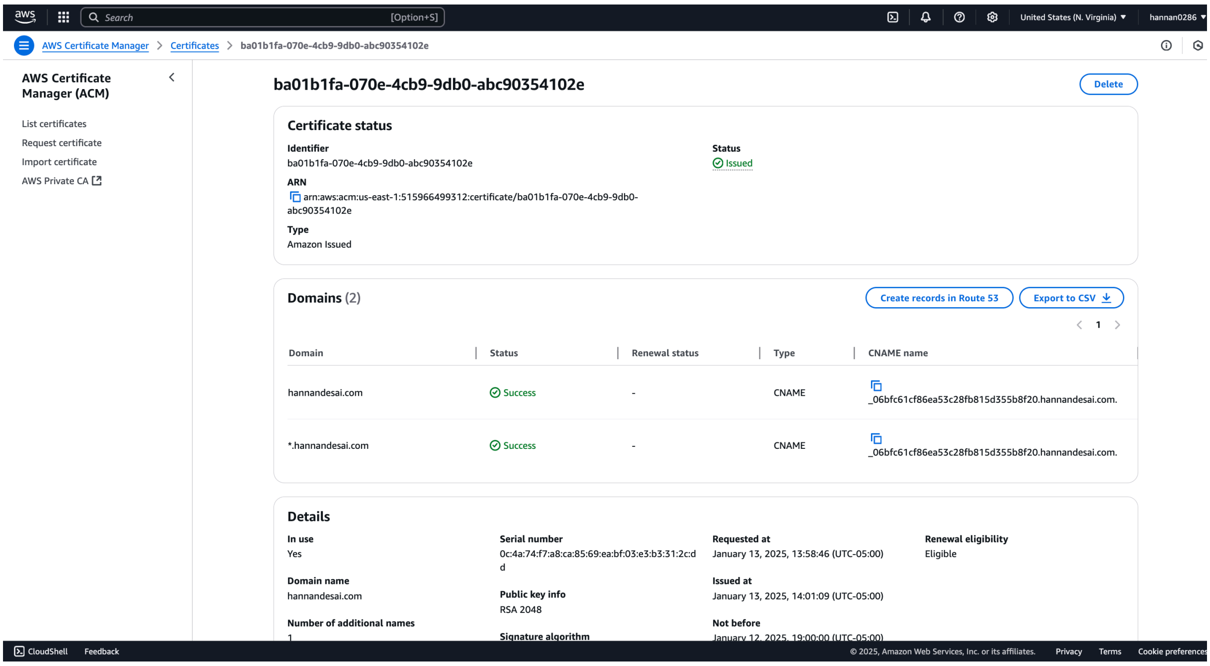
- **Content Delivery**: Amazon CloudFront was configured as a Content Delivery Network (CDN), ensuring low latency and high availability through globally distributed edge locations.



Distributions (2)

☐	ID	Description	Type	Domain name	Alternate domai...	Origins	Status	Last modified
☐	E2TWXWW4JZY5M0	-	Production	d39fg7eltv0qxb.clo...	hannandesai.com	hannandesai.com.s3-wet	✓ Enabled	January 19, 2025 a...
☐	E2P00VZGA8UBEW	-	Production	dkkc41hg48cr8.clo...	www.hannandesai.com	www.hannandesai.com.s:	✓ Enabled	January 18, 2025 a...

- **HTTPS Security**: AWS Certificate Manager provided SSL/TLS certificates to enforce HTTPS for secure data transmission.



ba01b1fa-070e-4cb9-9db0-abc90354102e

Certificate status

Identifier
ba01b1fa-070e-4cb9-9db0-abc90354102e

Status
✓ Issued

ARN
arn:aws:acm:us-east-1:515966499312:certificate/ba01b1fa-070e-4cb9-9db0-abc90354102e

Type
Amazon Issued

Domains (2)

Domain	Status	Renewal status	Type	CNAME name
hannandesai.com	✓ Success	-	CNAME	_06bfc61cf86ea53c28fb815d355b8f20.hannandesai.com.
*.hannandesai.com	✓ Success	-	CNAME	_06bfc61cf86ea53c28fb815d355b8f20.hannandesai.com.

Details

In use
Yes

Domain name
hannandesai.com

Number of additional names
1

Serial number
0c:4a:74:f7:a8:ca:85:69:ea:bf:03:e3:b3:31:2c:dd

Public key info
RSA 2048

Signature algorithm

Requested at
January 13, 2025, 13:58:46 (UTC-05:00)

Issued at
January 13, 2025, 14:01:09 (UTC-05:00)

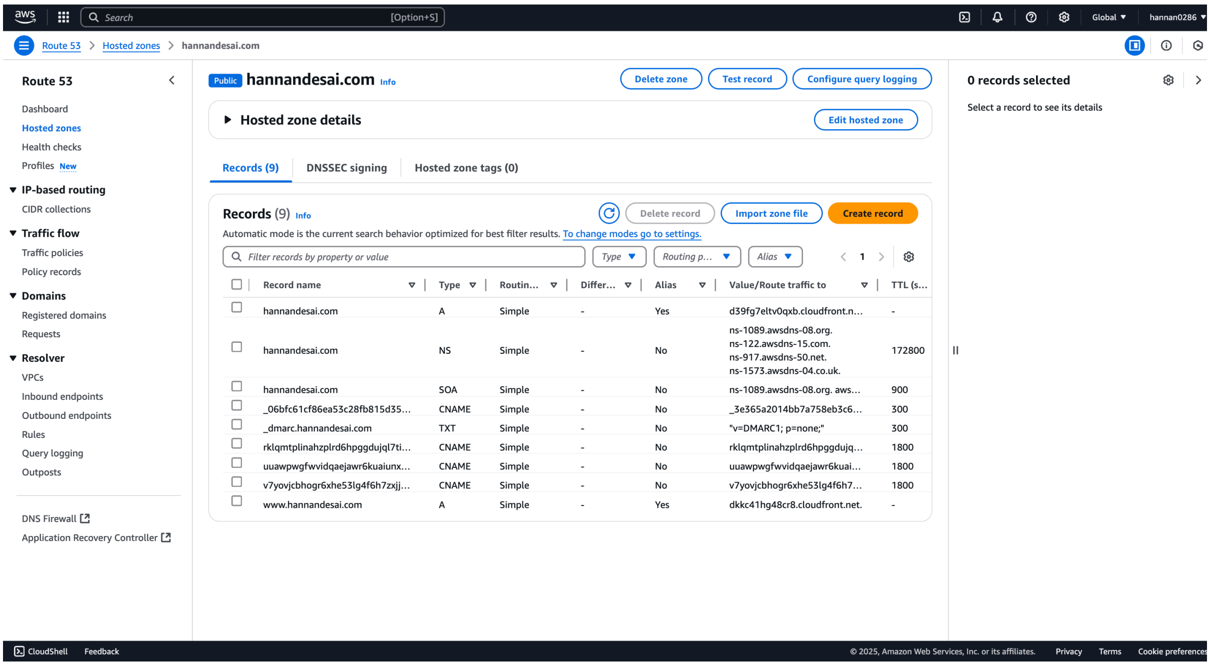
Not before
January 12, 2025, 19:00:00 (UTC-05:00)

Renewal eligibility
Eligible

3. Domain Setup with Route 53

- **Custom Domain**: Amazon Route 53 was used to register and configure the custom domain for the portfolio website.

- **DNS Configuration**: DNS records were updated to route traffic to the CloudFront distribution, enabling secure and professional access to the site.



hannandesai.com

Hosted zone details

0 records selected

Select a record to see its details

Records (9) Info

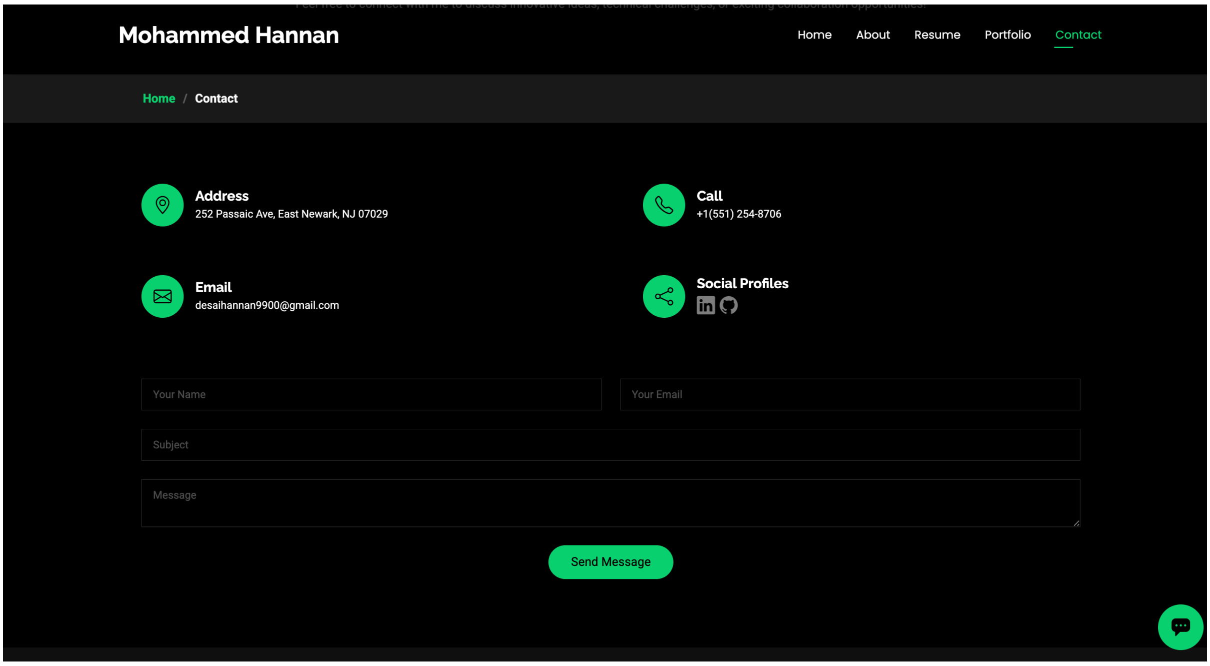
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Filter records by property or value

□	Record name	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...
□	hannandesai.com	A	Simple	-	Yes	d39fg7eltv0qxb.cloudfront.n...	-
□	hannandesai.com	NS	Simple	-	No	ns-1089.awsdns-08.org. ns-122.awsdns-15.com. ns-917.awsdns-50.net. ns-1573.awsdns-04.co.uk.	172800
□	hannandesai.com	SOA	Simple	-	No	ns-1089.awsdns-08.org. aws...	900
□	_06bfc61cf86ea53c28fb815d35...	CNAME	Simple	-	No	_3e365a2014bb7a758eb3c6...	300
□	_dmarc.hannandesai.com	TXT	Simple	-	No	"v=DMARC1; p=none;"	300
□	rklqmtplinahzplrd6hpggdujql7ti...	CNAME	Simple	-	No	rklqmtplinahzplrd6hpggdujq...	1800
□	uuawpwgfwvidqaejawr6kuaiunx...	CNAME	Simple	-	No	uuawpwgfwvidqaejawr6kuai...	1800
□	v7yovjcbhogr6xhe53lg4f6h7zxjj...	CNAME	Simple	-	No	v7yovjcbhogr6xhe53lg4f6h7...	1800
□	www.hannandesai.com	A	Simple	-	Yes	dkkc41hg48cr8.cloudfront.net.	-

4. Interactive Features and Serverless Backend

- **Contact Form**:
  - Users submit messages through a form integrated into the React frontend.



Feel free to connect with me to discuss innovative ideas, technical challenges, or exciting collaboration opportunities.

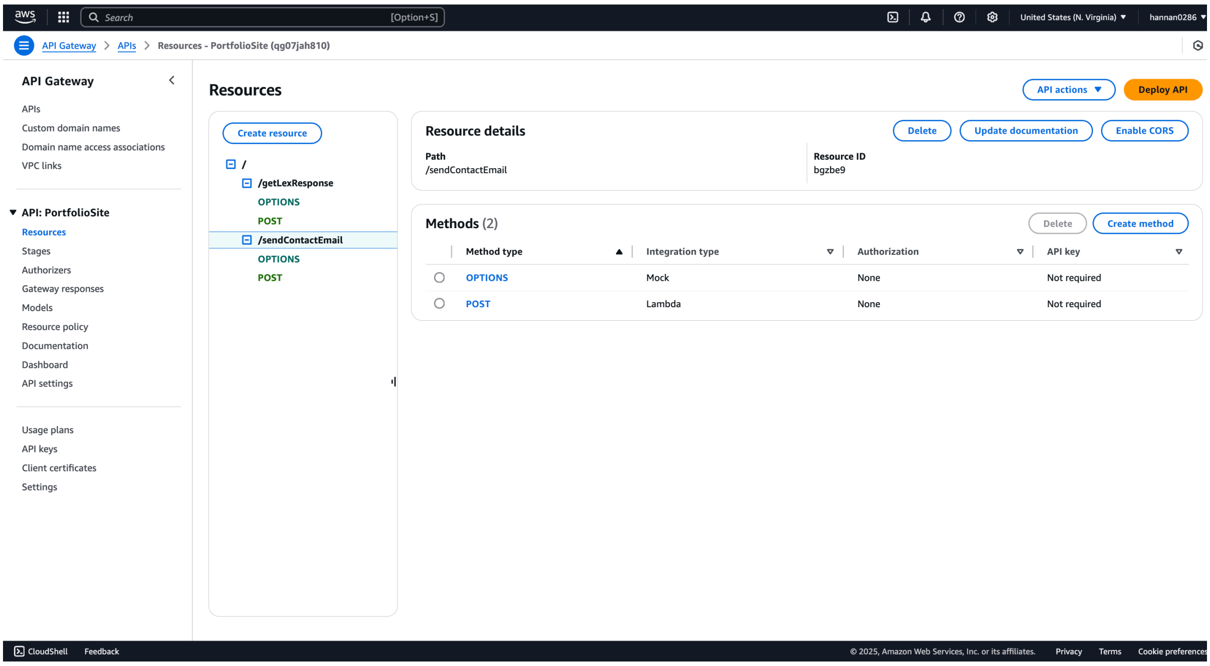
Address
252 Passaic Ave, East Newark, NJ 07029

Call
+1(551) 254-8706

Email
desaihannan9900@gmail.com

Social Profiles

- API Gateway routes form submissions to a Lambda function.



Resource details

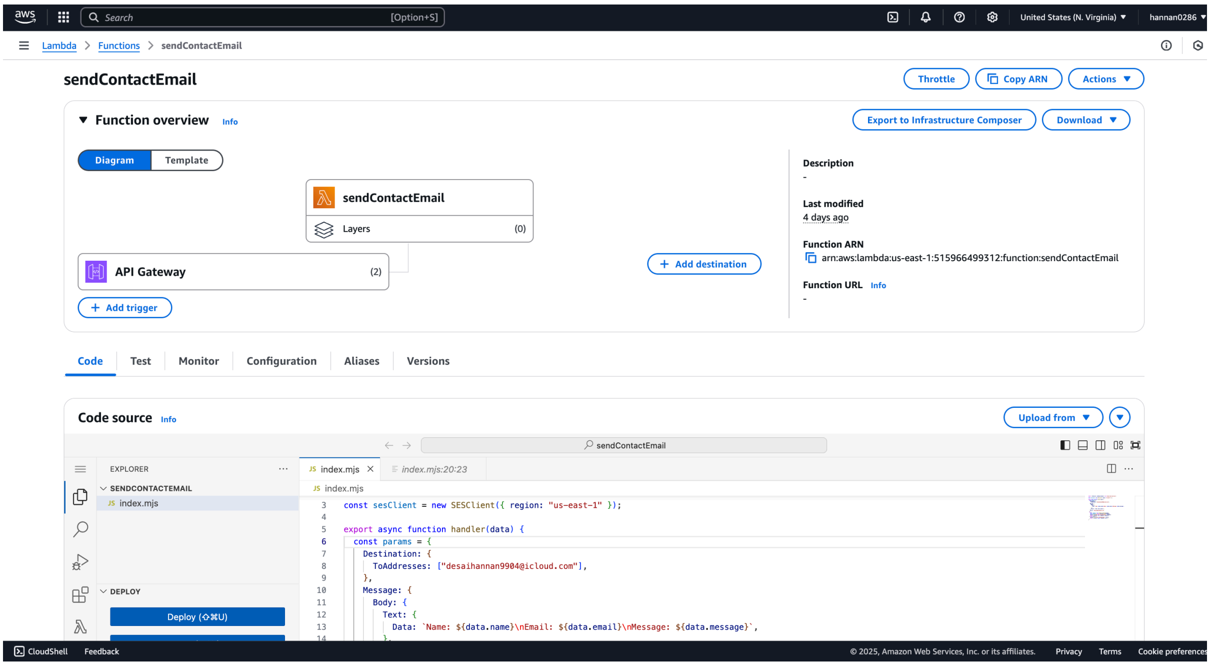
Path
/sendContactEmail

Resource ID
bgzbe9

Methods (2)

Method type	Integration type	Authorization	API key
OPTIONS	Mock	None	Not required
POST	Lambda	None	Not required

- Lambda processes the input and triggers Amazon SES to send the message to the designated inbox.



sendContactEmail

Function overview Info

Description
-

Last modified
4 days ago

Function ARN
arn:aws:lambda:us-east-1:515966499312:function:sendContactEmail

Function URL Info
-

Code source Info

```javascript
const sesClient = new SESClient({ region: "us-east-1" });

export async function handler(data) {
  const params = {
    Destination: {
      ToAddresses: ["desaihannan9904@icloud.com"],
    },
    Message: {
      Body: {
        Text: {
          Data: `Name: ${data.name}\nEmail: ${data.email}\nMessage: ${data.message}`,
        },
```

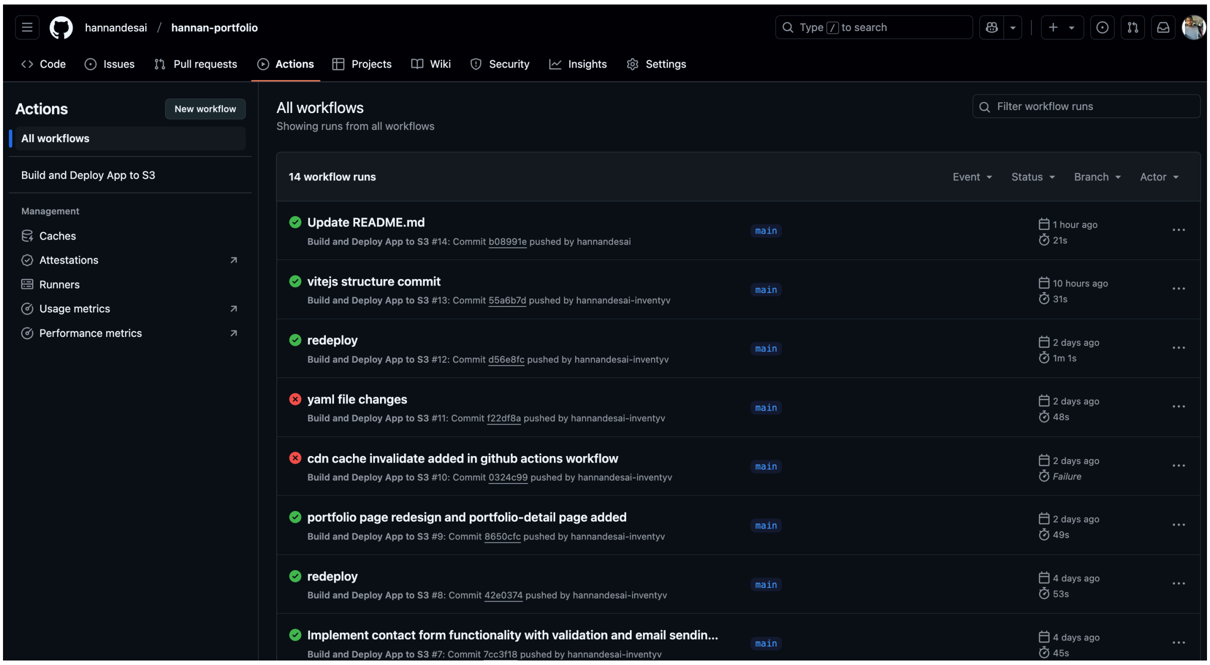
- Workflow: User Input -> API Gateway -> Lambda -> SES -> Email.

- **AI Chatbot**:
  - An Amazon Lex-powered chatbot was integrated for real-time user interaction.
  - Messages from the chatbot UI are sent via API Gateway to a Lambda function, which interacts with Amazon Lex.

   - Chat history and context are managed using DynamoDB to enhance conversational logic and user experience.
   - Architecture: Chatbot UI -> API Gateway -> Lambda -> Lex -> DynamoDB.

5. Automation with GitHub Actions

- A GitHub Actions pipeline was configured to automate deployments. Upon every push to the main branch, the pipeline:
   - Builds the React application.
   - Deploys the updated assets to the S3 bucket.
   - Invalidates CloudFront cache to reflect the latest changes immediately.
- This automation ensures efficient and error-free updates.



All workflows

Showing runs from all workflows

14 workflow runs

Update README.md
Build and Deploy App to S3 #14: Commit b08991e pushed by hannandesai
main
1 hour ago
21s

vitejs structure commit
Build and Deploy App to S3 #13: Commit 55a6b7d pushed by hannandesai-inventyv
main
10 hours ago
31s

redeploy
Build and Deploy App to S3 #12: Commit d56e8fc pushed by hannandesai-inventyv
main
2 days ago
1m 1s

yaml file changes
Build and Deploy App to S3 #11: Commit f22df8a pushed by hannandesai-inventyv
main
2 days ago
48s

cdn cache invalidate added in github actions workflow
Build and Deploy App to S3 #10: Commit 0324c99 pushed by hannandesai-inventyv
main
2 days ago
Failure

portfolio page redesign and portfolio-detail page added
Build and Deploy App to S3 #9: Commit 8650cfc pushed by hannandesai-inventyv
main
2 days ago
49s

redeploy
Build and Deploy App to S3 #8: Commit 42e0374 pushed by hannandesai-inventyv
main
4 days ago
53s

Implement contact form functionality with validation and email sendin...
Build and Deploy App to S3 #7: Commit 7cc3f18 pushed by hannandesai-inventyv
main
4 days ago
45s

Architectural Diagram

The following architecture was implemented for the portfolio website:

1. **Frontend**:
   - Developed in React (hosted on S3 with CloudFront as CDN).
2. **Backend**:
   - Serverless architecture utilizing Lambda for contact form handling and chatbot interaction.
   - API Gateway as the entry point for backend services.
3. **Database**:
   - DynamoDB for storing chatbot context and interaction history.
4. **Security**:
   - ACM for HTTPS.
   - IAM roles for secure resource access.

5. **Automation**:
    - CI/CD pipeline using GitHub Actions.

Conclusion

This project demonstrates the integration of modern frontend development practices with robust cloud architecture. By leveraging React, AWS services, and serverless design, the portfolio website achieves high performance, security, and scalability. The documented process serves as a reference for developers exploring the intersection of web development and cloud engineering.

Explore AWS and React to create your own scalable solutions!

Happy Building!