
📁 Project 01 - Static Portfolio Website on S3 with CI/CD

Overview: Host a personal portfolio using Vite on AWS S3 with automated CI/CD via GitHub Actions.

🔧 Services Used:

* **Amazon S3**: Static file hosting

* **AWS Route 53**: Custom domain DNS

* **Amazon Certificate Manager (ACM)**: SSL (HTTPS) support

* **Amazon CloudFront**: CDN for global delivery

* **GitHub Actions**: CI/CD deployment

⚙️ Configuration:

* `vite.config.js` used for building static files

* GitHub Actions workflow triggers on changes in `site` folder

* S3 bucket is configured for website hosting

* CloudFront distribution points to S3 bucket

* Route 53 record links domain to CloudFront

📁 Project 02 - Mass Emailing System using AWS Lambda and SES

Overview: Send bulk emails from a CSV using AWS Lambda and SES.

🔧 Services Used:

- **AWS Lambda**: Main email logic

- **Amazon SES**: Email delivery

- **Amazon S3**: Store `email-list.csv`

- **GitHub Actions**: CI/CD for Lambda function

⚙️ Configuration:

- CSV is fetched from S3

- Each row is parsed and passed to SES

- Lambda connected via GitHub Actions CI

- IAM permissions for S3 + SES

- Secrets stored via GitHub Secrets

📁 Project 03 - Alexa Skill for Portfolio Projects

Overview: Users can ask Alexa about your portfolio projects.

🔧 Services Used:

- **Amazon Alexa Developer Console**: Custom skill

- **AWS Lambda**: Backend logic

- **Amazon S3** (optional): Store documentation

⚙️ Configuration:

- Custom Alexa skill with intents like `ProjectInfoIntent`

- Lambda linked via ARN in Alexa console

- Response mapped to user queries

- JSON interaction model created with utterances

📁 Project 04 - Text-to-Speech Generator with Amazon Polly

Overview: Convert user-inputted text to audio and store it on S3.

🔧 Services Used:

- **Amazon Polly**: Text-to-speech generation

- **AWS Lambda**: Main function

- **Amazon S3**: Store `.mp3` files

- **API Gateway**: Front-facing API

⚙️ Configuration:

- Text sent to Lambda via API Gateway POST

- Polly returns audio stream

- Audio is uploaded to S3

- Response includes public S3 URL

📁 Project 05 - Music Recommendation API with Custom ML (Alt to Amazon Personalize)

Overview: Use Python to build and train a recommender model. Optionally run API via Lambda or Flask.

🔧 Services Used:

- **Amazon S3**: Store training data

- **Amazon SageMaker / Local Python**: Model training (temporary)

- **GitHub**: Model + API code

- **API Gateway + Lambda (optional)**: Serve predictions

⚙️ Configuration:

- Python-based collaborative filtering model

- Spotify dataset used (1GB sample)

- Once trained, model is stored

- Predict endpoint serves user-based recommendations

📁 Project 06 - Serverless Image Resizer

Overview: Upload or link to an image, resize it, and receive a public download link.

🔧 Services Used:

- **AWS Lambda**: Main resizing logic

- **API Gateway**: Accepts URL or image upload

- **Amazon S3**: Stores resized image

- **Frontend**: Built with Vite + shadcn/ui

⚙️ Configuration:

- Lambda receives image via URL or upload

- Sharp used to resize based on selected size or custom dimensions

- S3 stores resized image with public access

- Vite UI sends requests to `/resize` endpoint

- Image download link shown after processing
