SERVERLESS PORTFOLIO

Project Introduction

This is a full-stack application built using HTML, CSS, and JavaScript for the frontend, with Python powering the backend. The website is fully responsive, providing an optimal user experience across various screen sizes. It follows a serverless architecture, leveraging several AWS services for scalability, performance, and high availability. The deployment process is automated through a CI/CD pipeline using GitHub Actions, ensuring efficient and reliable updates from code commit to live deployment. This approach allows for continuous integration and delivery, ensuring the website remains up-to-date and performs reliably.

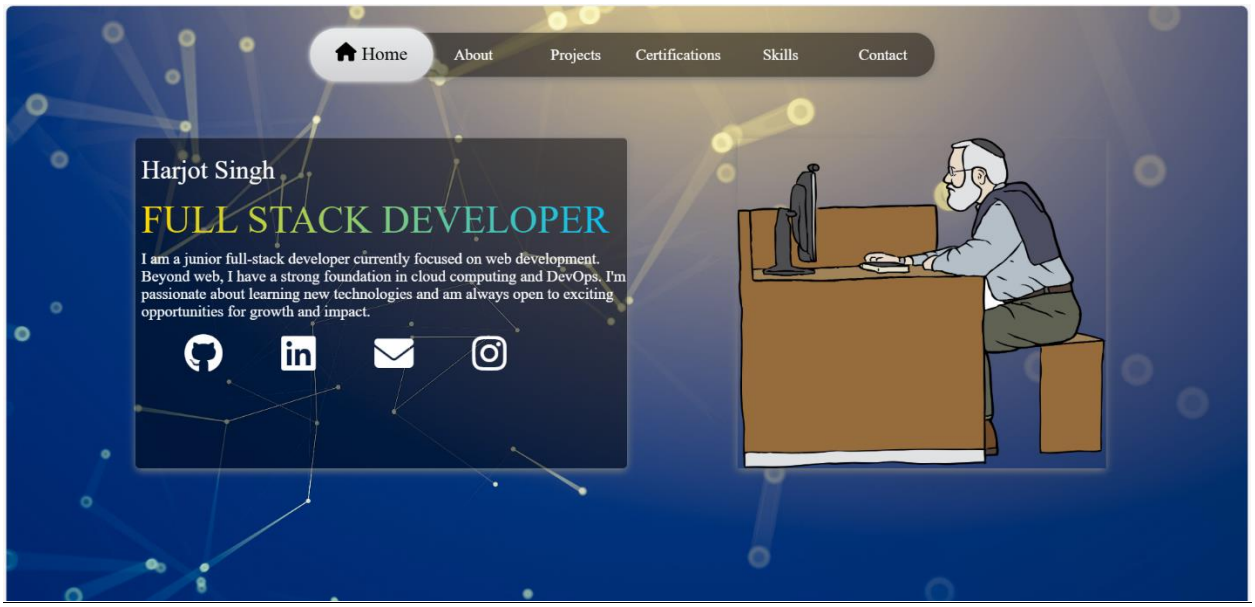
Live: https://www.harjotbasota.com

Repo: https://github.com/harjotbasota/portfolioNew.git

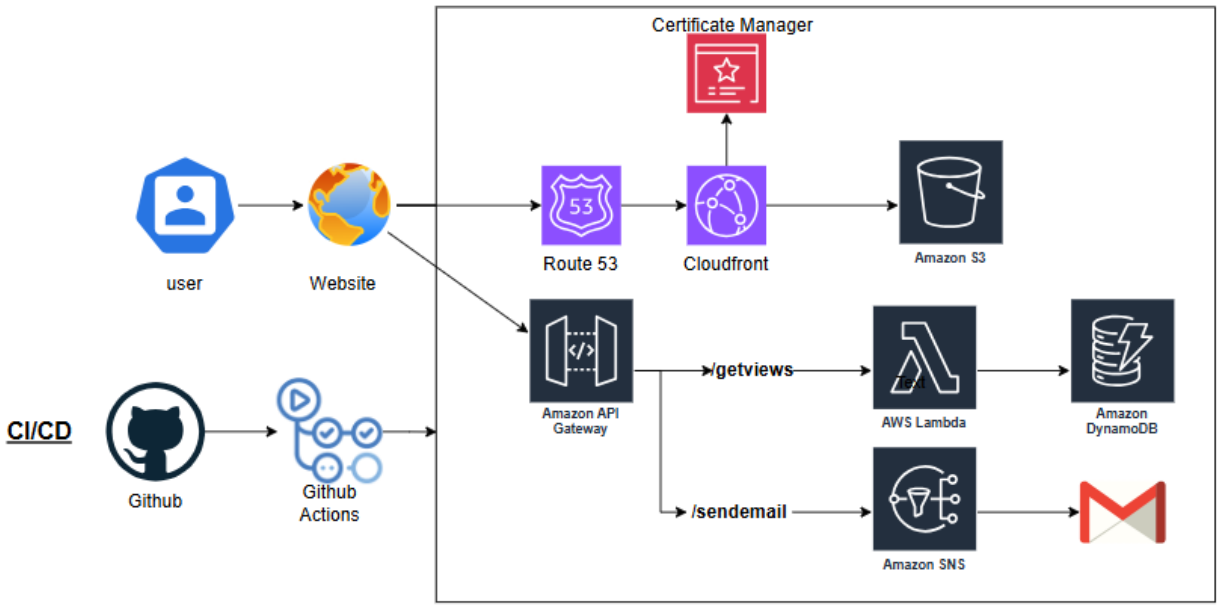
Tech Stack

- HTML
- CSS
- JavaScript
- Python
- APIs
- Git/GitHub
- GitHub Actions
- AWS Route 53
- AWS CloudFront
- AWS S3
- AWS Certificate Manager
- AWS API Gateway
- AWS Lambda Function
- AWS CloudFront
- AWS SNS
- AWS DynamoDB
- CI/CD

Website



Arch Diagram



Application Description – Frontend

- **Built with JavaScript and CSS**: The portfolio site is developed using pure JavaScript and CSS for a lightweight, efficient experience.
- **Key Features**:

- Displays total page visits.
- Includes a contact form to allow users to email the developer.
- Animations for enhanced user interactivity.
- **Fully Responsive**: The design adapts seamlessly to various screen sizes, ensuring optimal performance on mobile, tablet, and desktop devices.
- **Styling**: The site uses vanilla CSS for styling, with no external frameworks or libraries.
- **API Integration**: The frontend fetches data through APIs using the Fetch API.
- **Deployment**: Hosted on AWS S3 and distributed via CloudFront, providing fast load times and high availability

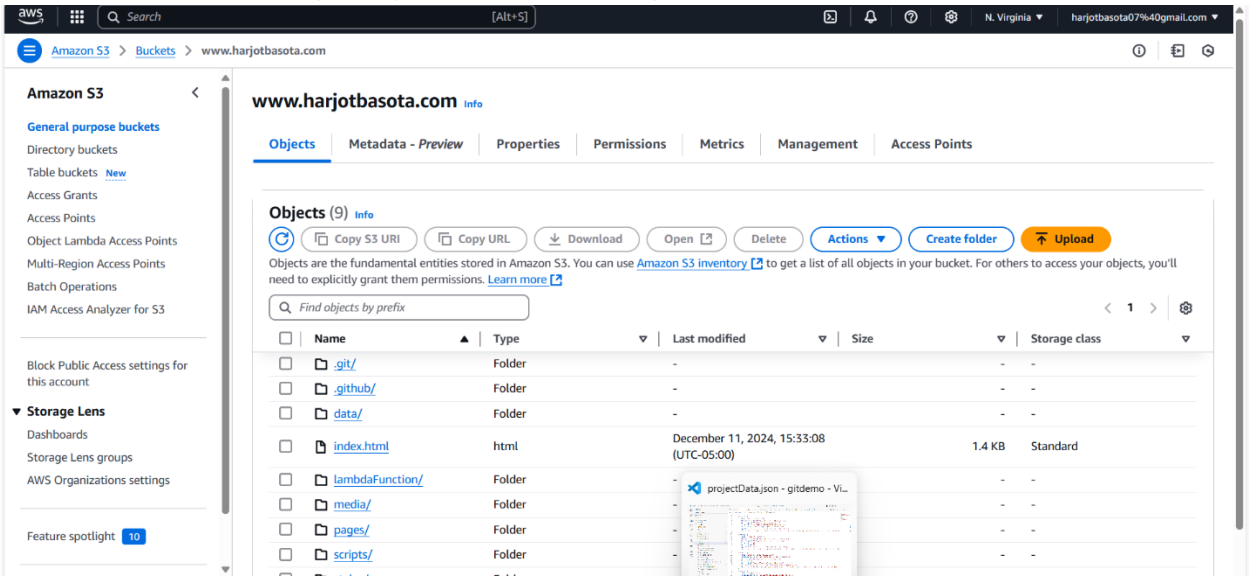
Application Description – Backend

- **Built with Python**: The backend of the portfolio site is powered by Python, utilizing AWS services like SNS, Lambda, and DynamoDB.
- **Core Functionality**:

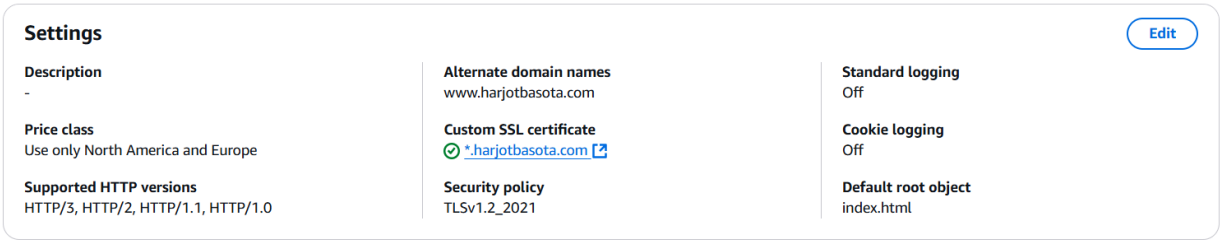
- **APIs**: Exposes REST APIs, including:
    - /sendemail to send emails to developer.
    - /getviews to fetch view counts from DynamoDB.
- **Database**: Interacts with DynamoDB for data storage and management.
- **Security**: CORS policy is enforced to secure API access.
- **Deployment**: Deployed on AWS using API Gateway to serve REST APIs, SNS for notifications, Lambda for serverless execution, and DynamoDB for database management

Application Deployment

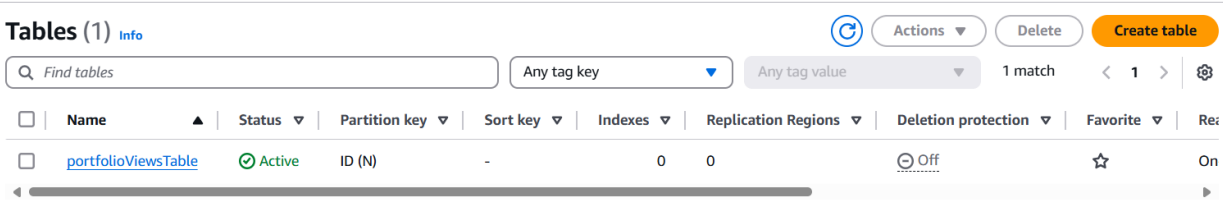
1. Create a S3 bucket and upload your files, ensure all public access is blocked



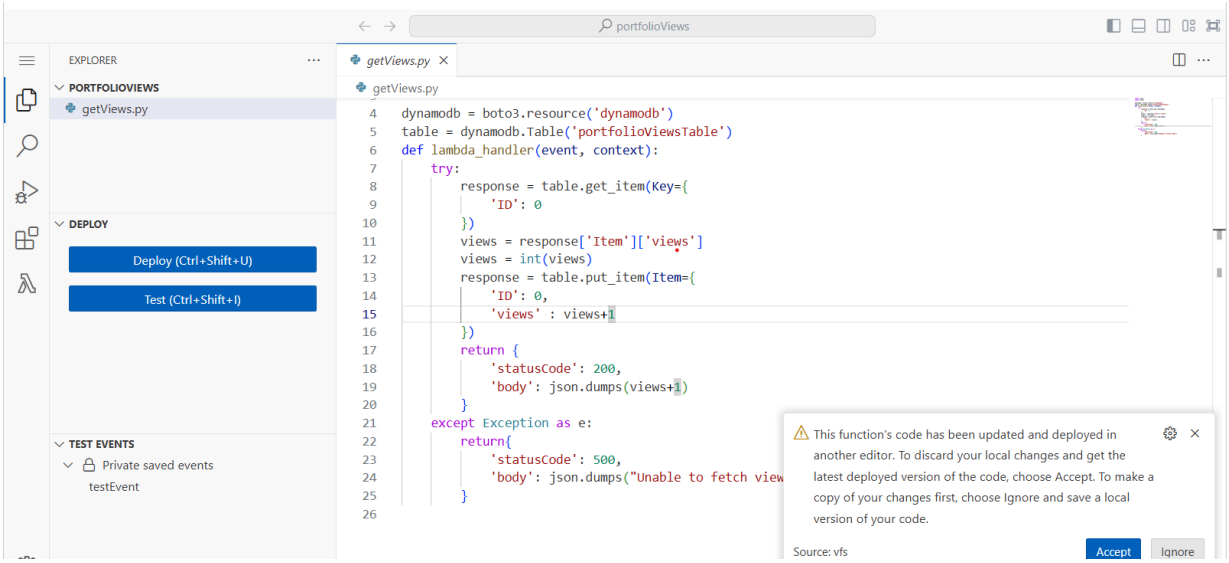
2. Create a cloudfront distribution with your s3 bucket as origin. Set up the alternate domain name, ssl certificate, default root object and attach the required policy to s3



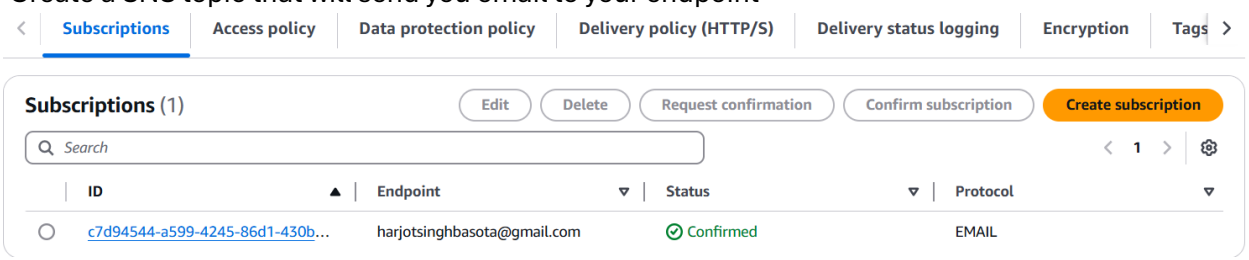
3. Create route 53 record that points your domain to cloud front distribution
4. Setup the dynamo DB table with item ID – 0 and views -0



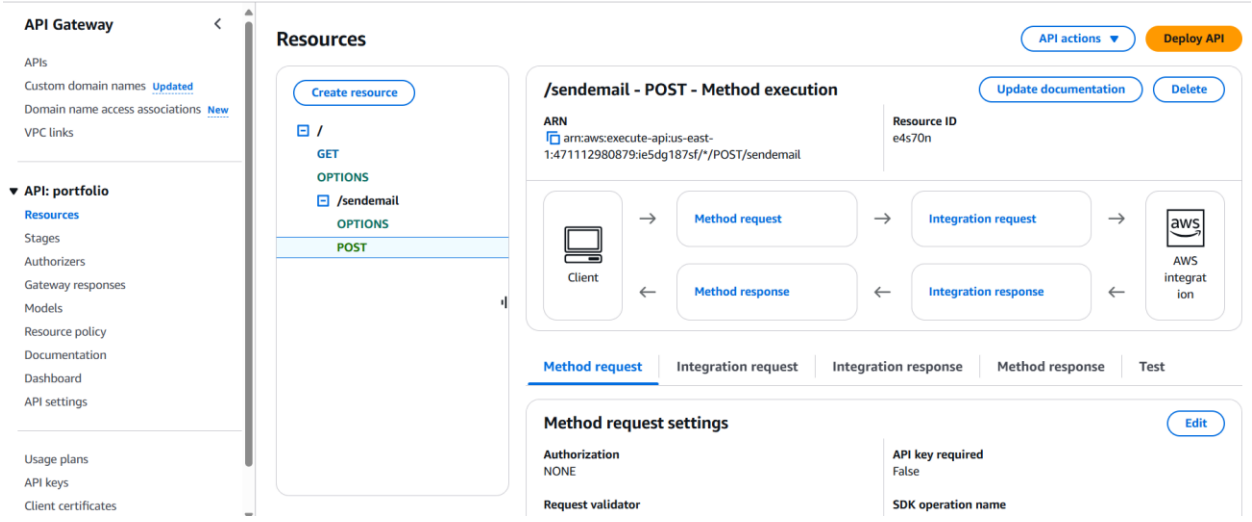
5. Create a Lambda function to access your table



6. Create a SNS topic that will send you email to your endpoint



7. Configure API gateway to both these services and enable the cors to allow your domain as origin



8. Configure github repo with the following secrets

AWS_ACCESS_KEY_ID

AWS_REGION

AWS_SECRET_ACCESS_KEY

CLOUDFRONT_DIST_ID

LAMBDA_FUNCTION_NAME

S3_BUCKET_NAME

9. Push changes to repo and test the pipeline