# Project Report: Scalable Static Website

Project: Project: Scalable Static Website with S3 + Cloudflare + GitHub Actions

Author: Mazin

**1. Introduction**

The objective of this project was to implement a modern, serverless, and automated hosting solution for a static website. Traditional web hosting is often manual, requires server management, and is difficult to scale. This project's goal was to build a hands-off CI/CD pipeline using cloud services. This pipeline automatically deploys website updates from a code repository, creating a highly reliable and scalable solution.

**2. Abstract**

This project successfully implemented a Continuous Integration and Continuous Deployment (CI/CD) pipeline for a static website using AWS S3 and GitHub Actions. A static website (HTML/CSS) was stored in a GitHub repository, and an AWS S3 bucket was configured for public website hosting. A GitHub Actions workflow was then developed to automatically detect a git push to the main branch, sync the files from the correct project subfolder to the S3 bucket, and publish the changes live to the internet.

**3. Tools Used**

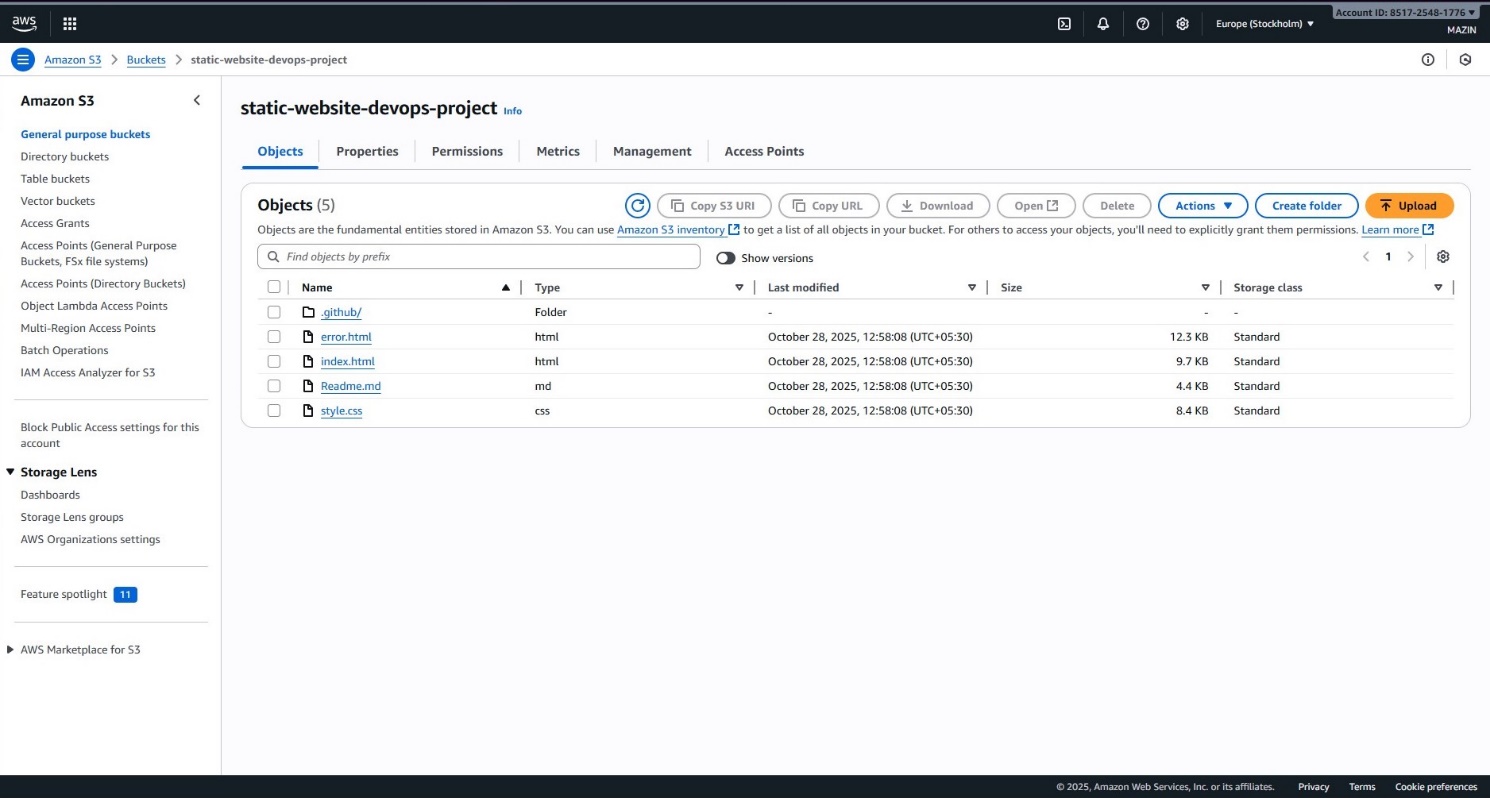
* **AWS S3 (Simple Storage Service):** Used as the serverless hosting service for the static website files.
* **GitHub Actions:** Used as the CI/CD platform to automate the build and deployment process.
* **AWS IAM (Identity and Access Management):** Used to create a secure user with programmatic access, allowing GitHub Actions to communicate with the AWS API.
* **HTML/CSS:** Used for the content of the static website.

**4. Steps Involved in Building the Project**

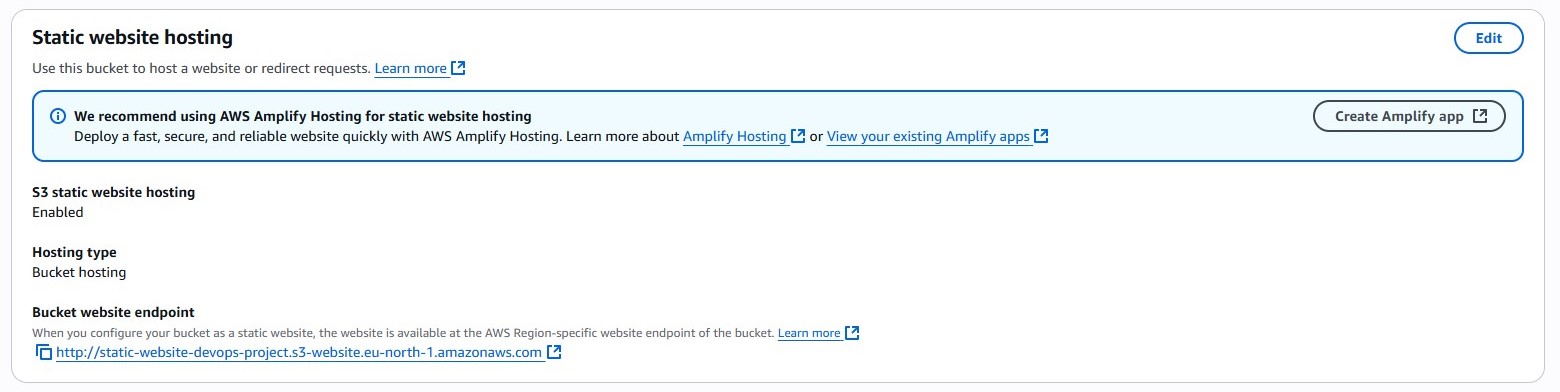
The project was built and validated in the following steps:

**Step 1: S3 Bucket and Hosting Configuration**

An S3 bucket (static-website-devops-project) was created. Static website hosting was enabled, with index.html as the index document.

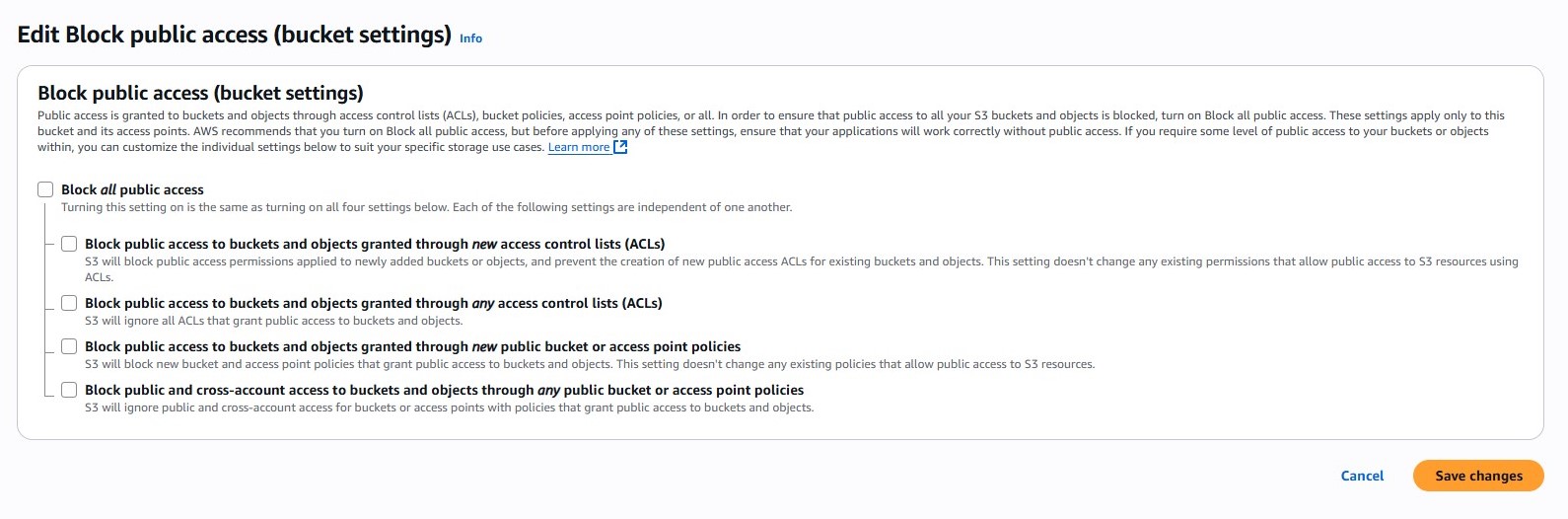
****

**S3 Bucket Files**



**Static Website Hosting Enabled**

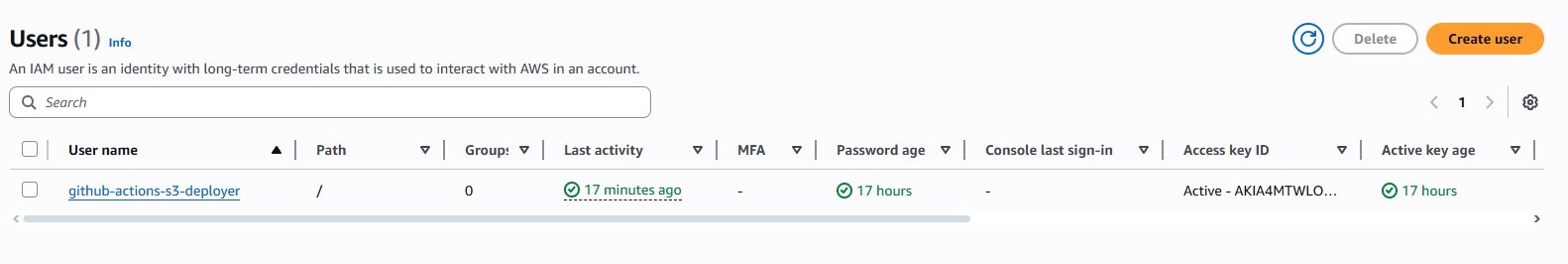
To make the site public, "Block all public access" was disabled, and a specific bucket policy was applied to grant s3:GetObject permissions to all users.



**Public Access Policy**

**Step 2: IAM Security Configuration**

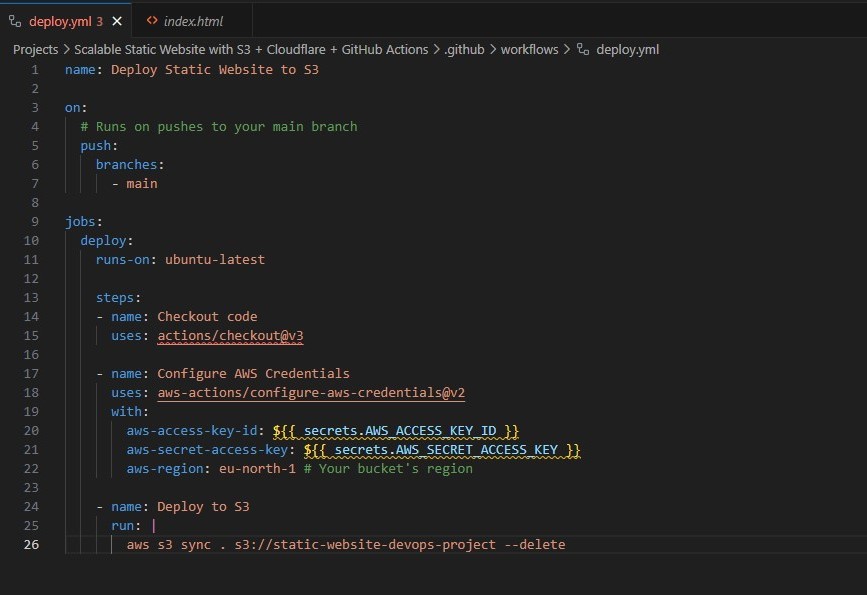
A dedicated IAM user (github-actions-s3-deployer) was created for the GitHub Actions workflow. This user was granted programmatic access and the permissions necessary to sync files to the S3 bucket.

****

**IAM User for GitHub Actions**

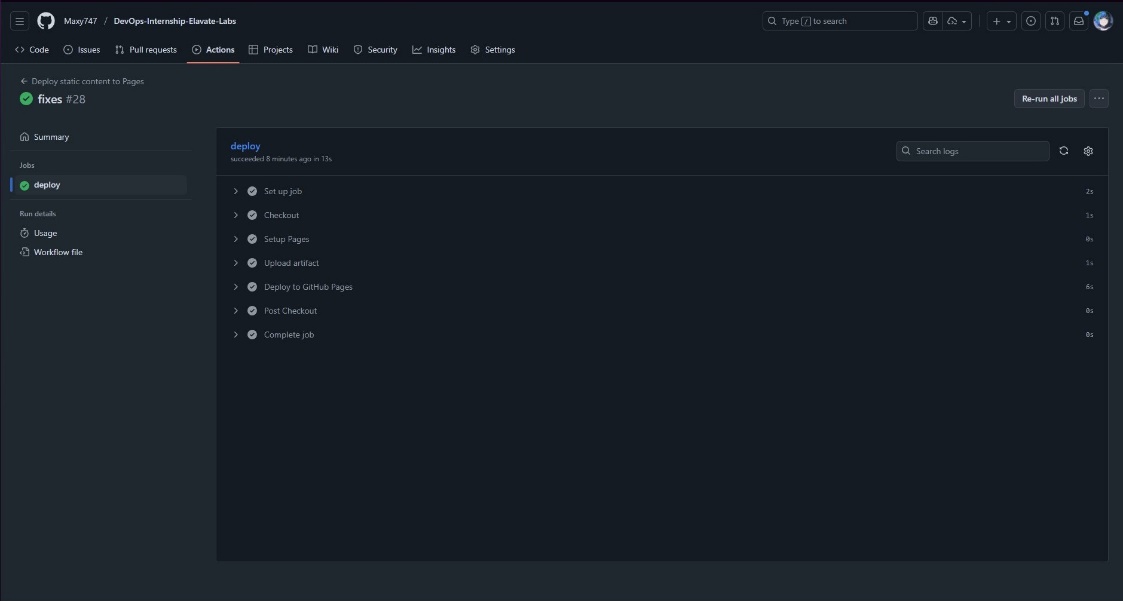
**Step 3: GitHub Actions CI/CD Workflow**

A workflow file (.github/workflows/deployS3.yml) was created in the repository. This file defines the CI/CD pipeline.



**Workflow .yml File**

The workflow was configured to trigger on a push to the main branch. It checks out the code, configures AWS credentials (via stored secrets), and runs the aws s3 sync command. This command was precisely configured to sync *only* the project subfolder ("Projects/Scalable Static Website...") to the root of the S3 bucket.



**Workflow Run in Progress:**

**Step 4: Deployment and Validation**

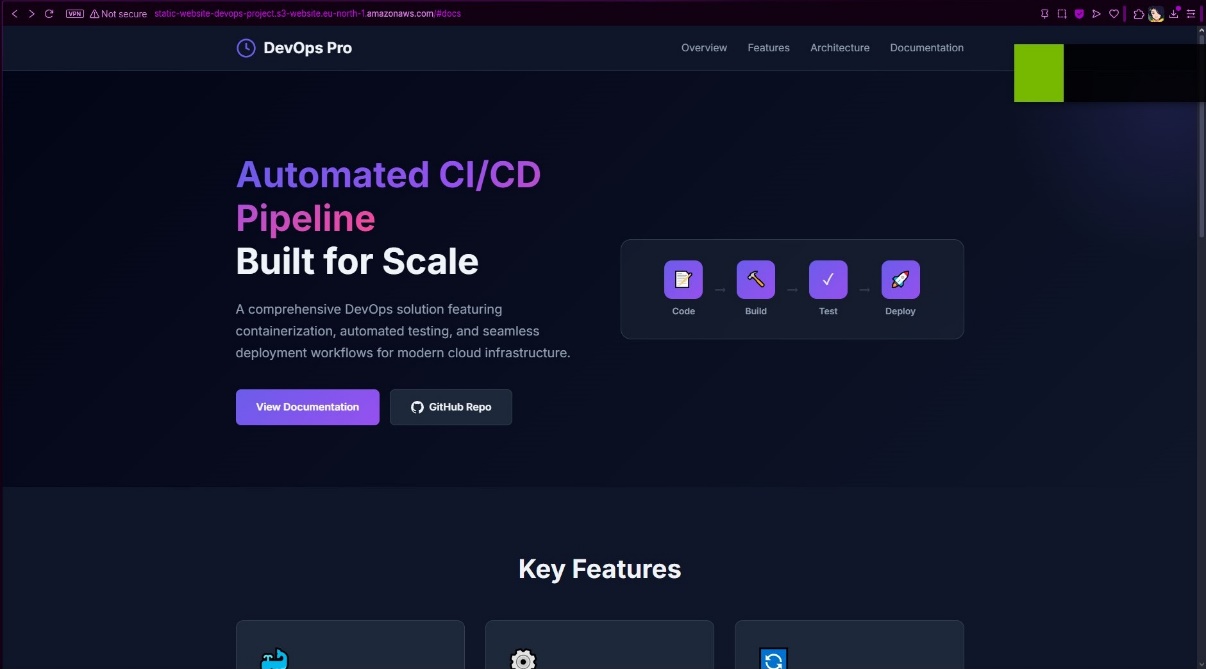
A commit was pushed to the main branch, triggering the workflow. The workflow ran successfully, as shown by the green checkmark.



**Workflow Succeeded**

After the workflow completed, the S3 bucket was correctly populated with the website files.

The successful deployment was validated by accessing the public S3 website endpoint, which correctly displayed the live website.



**Website successfully deployed and live at the S3 endpoint**

**Live Website:** <http://static-website-devops-project.s3-website.eu-north-1.amazonaws.com>

**5. Conclusion**

This project successfully achieved its core objective of creating an automated deployment pipeline for a static website. By integrating GitHub Actions with AWS S3, the manual process of uploading files is eliminated, improving efficiency and reliability. The infrastructure is now fully automated: any git push to the main branch results in a live-staged deployment.

The original project scope also included Cloudflare integration for a global CDN and HTTPS. This step was skipped to focus on the foundational CI/CD pipeline. The next logical step would be to register a free domain and add it to Cloudflare, completing the project's scalability and security goals.