Scalable Static Website with S3 + Cloudflare + GitHub Actions

Introduction

In today's digital landscape, continuous integration and continuous deployment (CI/CD) pipelines are widely used to ensure fast, reliable, and automated delivery of applications. Static websites are a common use case where automation can eliminate manual effort and guarantee that every change made in code is immediately reflected on the live site.

This project demonstrates how to build a fully automated pipeline to host and deploy a static website using **AWS S3 (Free Tier)** for storage, **Cloudflare (Free)** for global CDN and HTTPS, and **GitHub Actions** for automation.

Abstract

The objective of this project was to host and auto-deploy a static website with free-tier cloud resources. The static site was built with simple **HTML and CSS**, and deployment was automated using **GitHub Actions**.

Initially, the website content was pushed to an **S3 bucket** configured for static hosting. A GitHub Actions workflow was created to automatically sync code changes to the S3 bucket. To ensure the site is served with **global performance and HTTPS**, the setup was integrated with **Cloudflare**. Cloudflare Pages provided a free *.pages.dev domain with automatic SSL certificates, ensuring secure access.

This approach demonstrates a **zero-cost, cloud-native pipeline** that is scalable, automated, and secure.

Tools Used

- **AWS S3** → Object storage used for hosting the website content.
- **Cloudflare** → Provided CDN, DNS, and HTTPS with free SSL.
- **GitHub Actions** → Automated deployment pipeline triggered on every commit.
- **HTML/CSS** → Used to build the static website.
- **Bash & AWS CLI** → For syncing website files to S3 in the workflow.

Steps Involved in Building the Project

1. Static Website Creation

- A simple index.html and supporting CSS were created.
- The site was tested locally by opening the HTML file in a browser.

2. GitHub Repository Setup

- A new GitHub repository was created.

- The static website files were committed and pushed to the repository.

3. AWS S3 Bucket Configuration

- An S3 bucket was created with **static website hosting enabled**.
- Public access was configured via bucket policy.
- An IAM user with programmatic access was created and granted AmazonS3FullAccess.

4. GitHub Actions Workflow

- A workflow file (deploy.yml) was created under .github/workflows/.
- It was configured to run on every push to the main branch.
- The workflow used AWS credentials (stored as GitHub secrets) to **sync files to S3** automatically.

5. Cloudflare Integration

- The GitHub repository was connected to **Cloudflare Pages**.
- The root / was set as the **build output directory**.
- Cloudflare provided a free domain in the format *.pages.dev with automatic **HTTPS** enabled.

6. Auto-Deployment Testing

- A change was made in index.html and pushed to GitHub.
- GitHub Actions ran successfully, and Cloudflare deployed the updated site with HTTPS.

Conclusion

The project successfully demonstrates a **CI/CD pipeline** for hosting a static website using free-tier cloud services. The integration of **AWS S3, Cloudflare, and GitHub Actions** ensures that the website is globally accessible, secure (via HTTPS), and automatically updated with every commit.

This solution highlights how cloud-native tools can be leveraged to build an efficient, cost-effective, and automated deployment workflow for static websites.

Deliverables

- **Live Website Link with HTTPS:** https://s3-static-site.pages.dev/
- **Screenshots**:
  - GitHub Actions workflow run (successful deployment)
  - S3 bucket showing files
  - Cloudflare Pages dashboard
  - Live website with HTTPS in browser