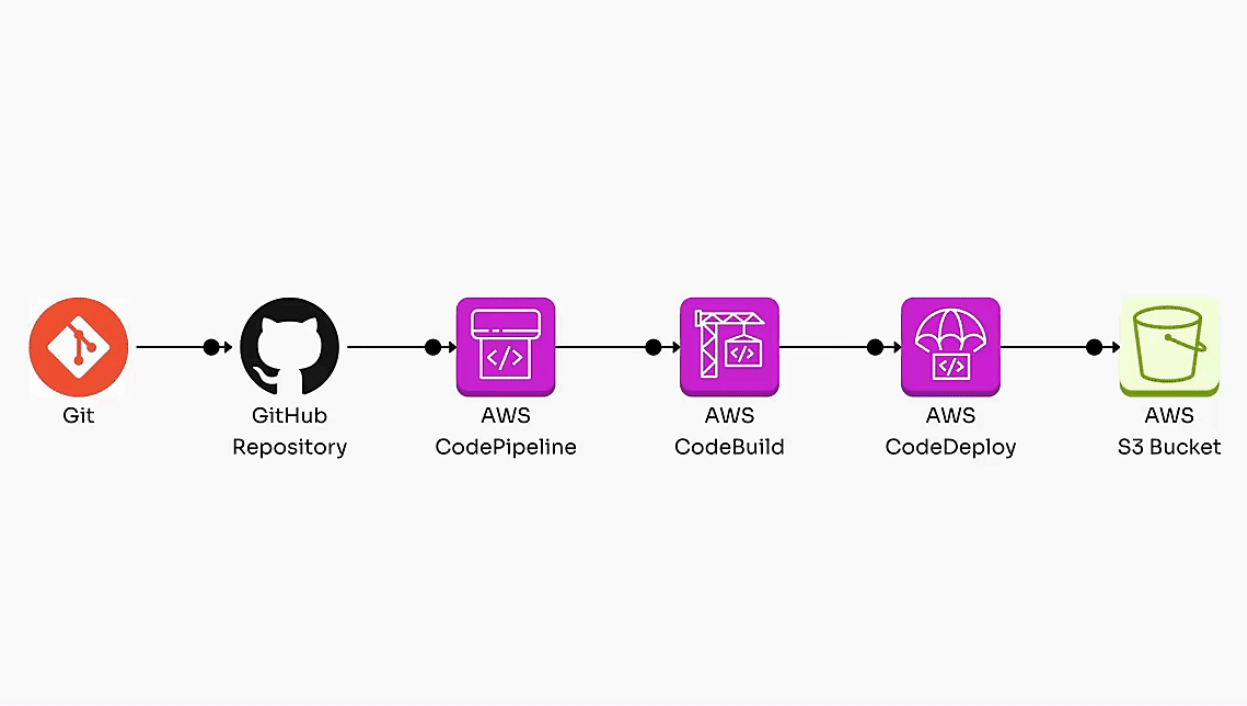
Automated CI/CD Pipeline for Static Website Hosting on AWS

Project Overview

This project demonstrates the creation of a fully automated **CI/CD pipeline** to deploy a static website using **Amazon Web Services (AWS)**. The website is hosted on **Amazon S3** with automated builds and deployments triggered by code changes in a **GitHub repository**, managed through **AWS CodePipeline** and **AWS CodeBuild**.

Architecture Diagram



Technologies Used

1. **Amazon S3**: Hosts the static website and serves it over the internet.
2. **AWS CodePipeline**: Automates the build, test, and deploy process.
3. **AWS CodeBuild**: Executes the build process and packages the website files.
4. **AWS CodeDeploy**: Deploys the website files to S3 bucket.
5. **GitHub**: Stores the website's source code and triggers pipeline executions on new commits.

Workflow

1. Clone a repository and push files into GitHub.

- Installed git and cloned a repository named "**Serverless-CICD**".

```
hardikrathod@DA:F8:CA:C3:DD:3C ~ % git --version
git version 2.47.0
hardikrathod@DA:F8:CA:C3:DD:3C ~ % git clone https://github.com/Hardik9791/Serve
rless-CICD.git
Cloning into 'Serverless-CICD'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
hardikrathod@DA:F8:CA:C3:DD:3C ~ %
```

- Added static website files to cloned repository and verified it.

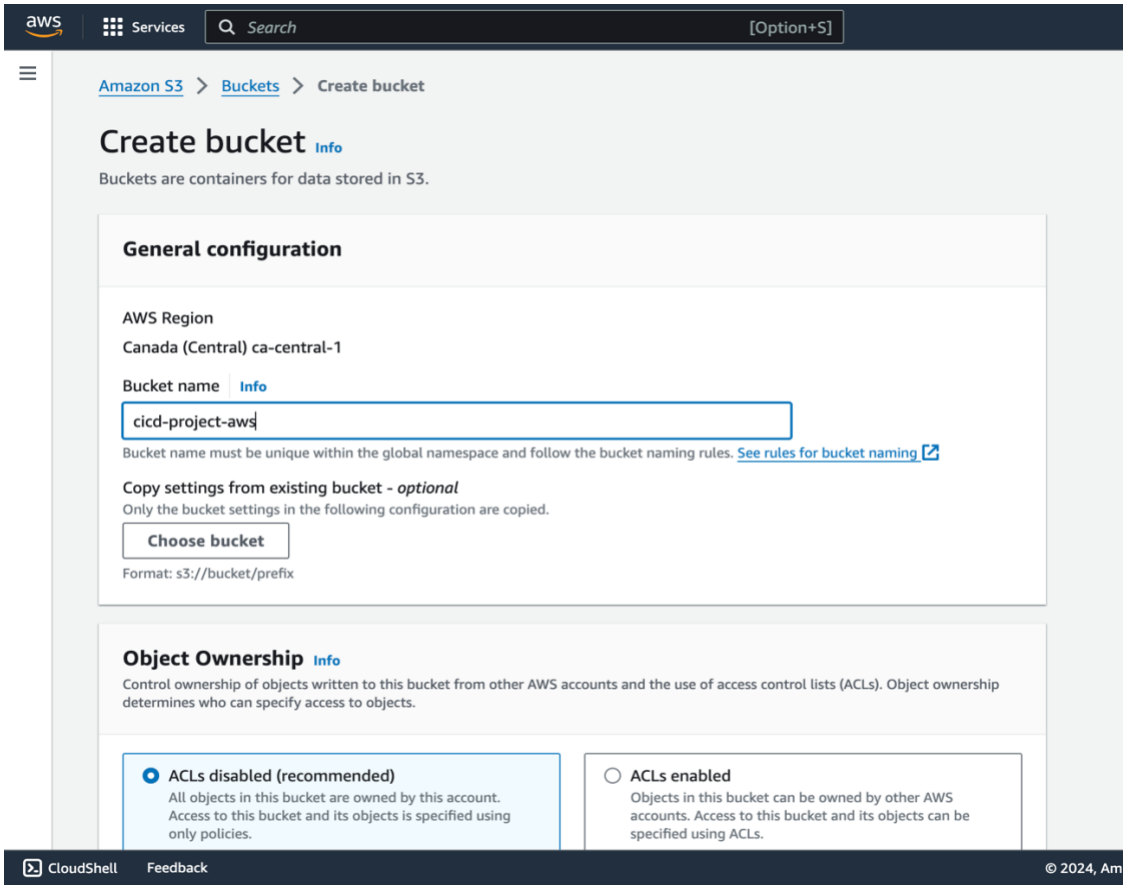
```
hardikrathod@DA:F8:CA:C3:DD:3C Serverless-CICD % git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        app.js
        buildspec.yaml
        index.html
        styles.css

nothing added to commit but untracked files present (use "git add" to track)
hardikrathod@DA:F8:CA:C3:DD:3C Serverless-CICD %
```

- Pushed all the files to **GitHub** using **"git"** commands.

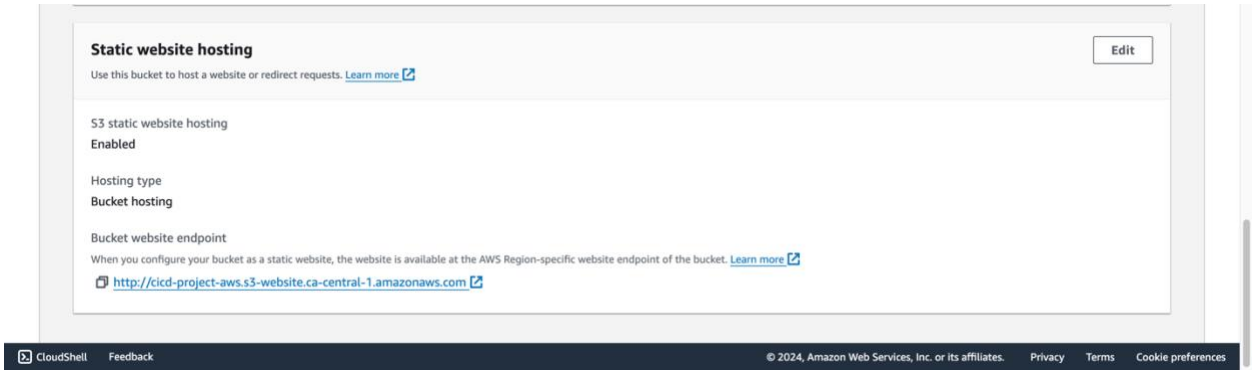
```
hardikrathod@DA:F8:CA:C3:DD:3C Serverless-CICD % git push origin main
Username for 'https://github.com/Hardik9791/Serverless-CICD.git': hardik9791
Password for 'https://hardik9791@github.com/Hardik9791/Serverless-CICD.git':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.08 KiB | 1.08 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Hardik9791/Serverless-CICD.git
   db9cf8f..d47e4b2  main -> main
hardikrathod@DA:F8:CA:C3:DD:3C Serverless-CICD %
```

2. Set Up S3 for Static Website Hosting

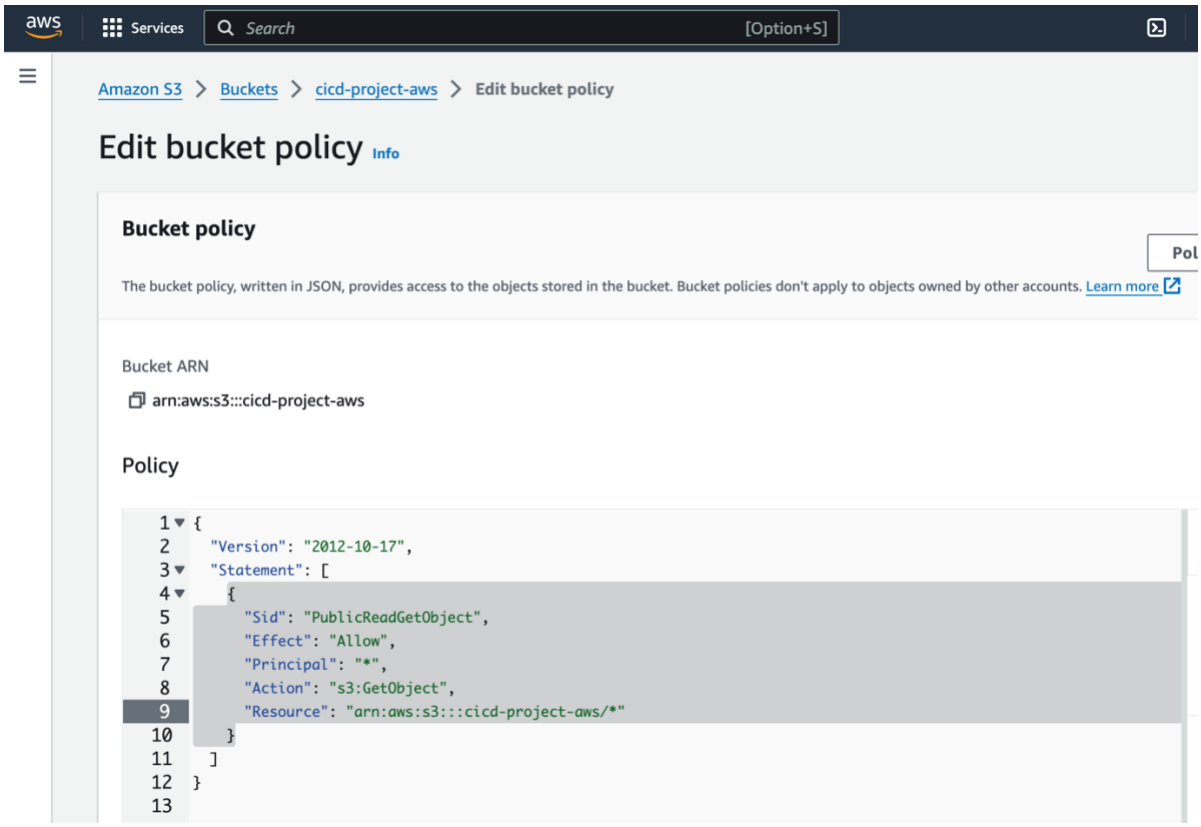
- I created an **S3 bucket** named cidcd-project-aws and ensured it was publicly accessible by disabling the "Block all public access" setting.



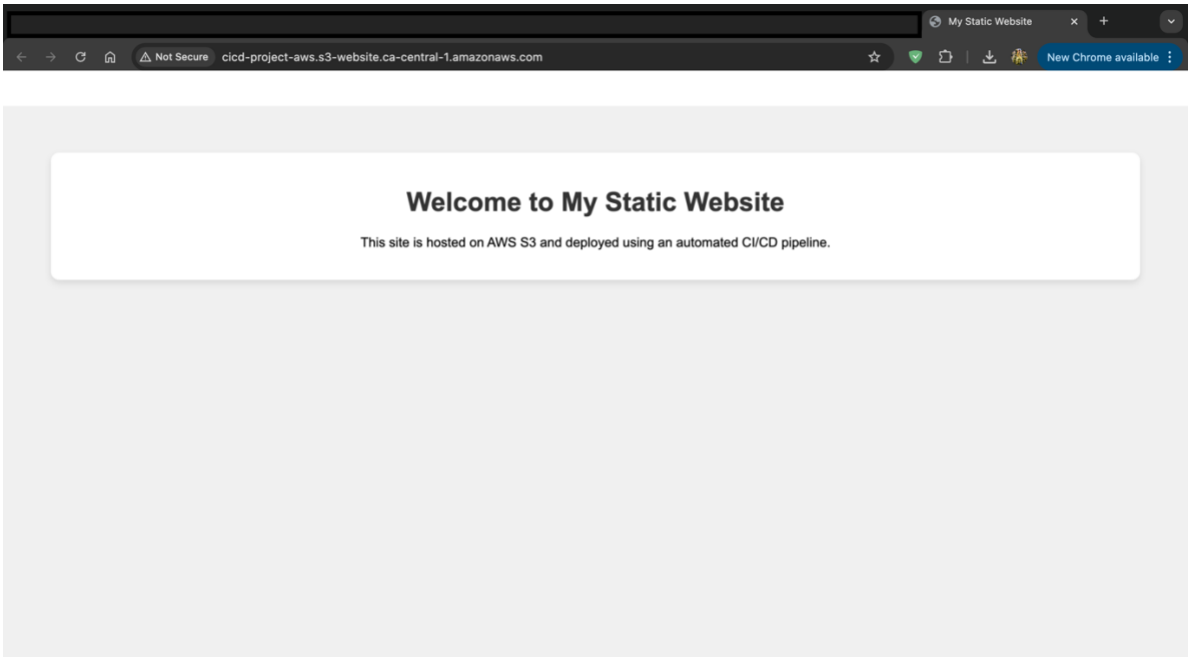
- I enabled **Static Website Hosting** for the bucket and specified index.html as the **Index document**.



- To ensure public access, I added a **Bucket Policy** that grants read permissions to all users, allowing my static files to be accessed over the web.

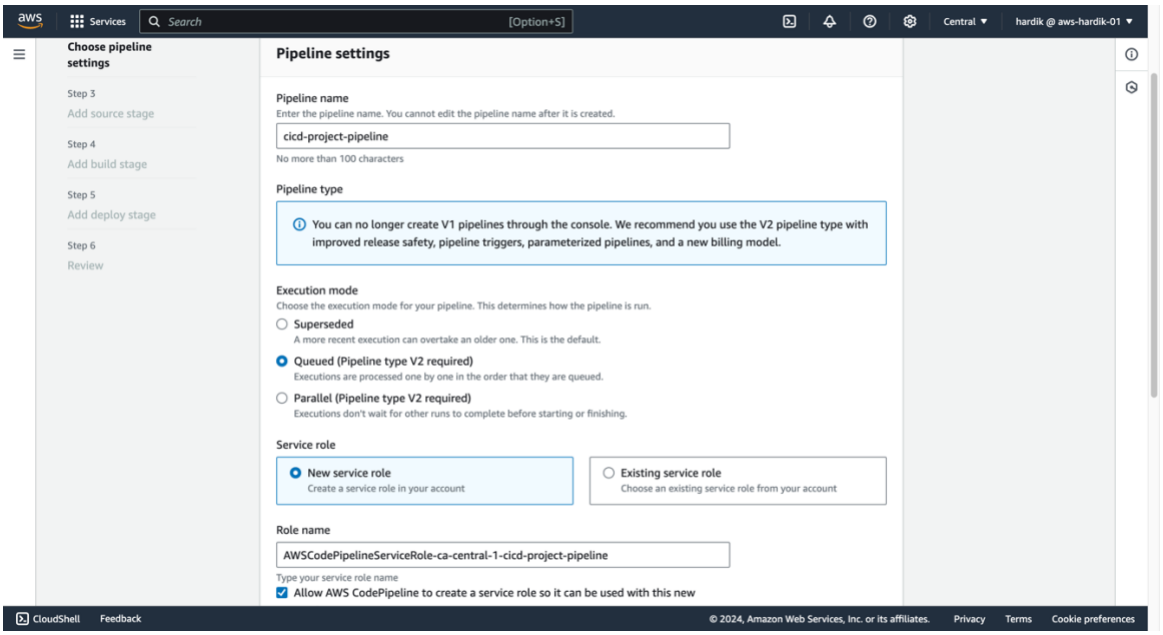


- Accessed a website using static website URL.

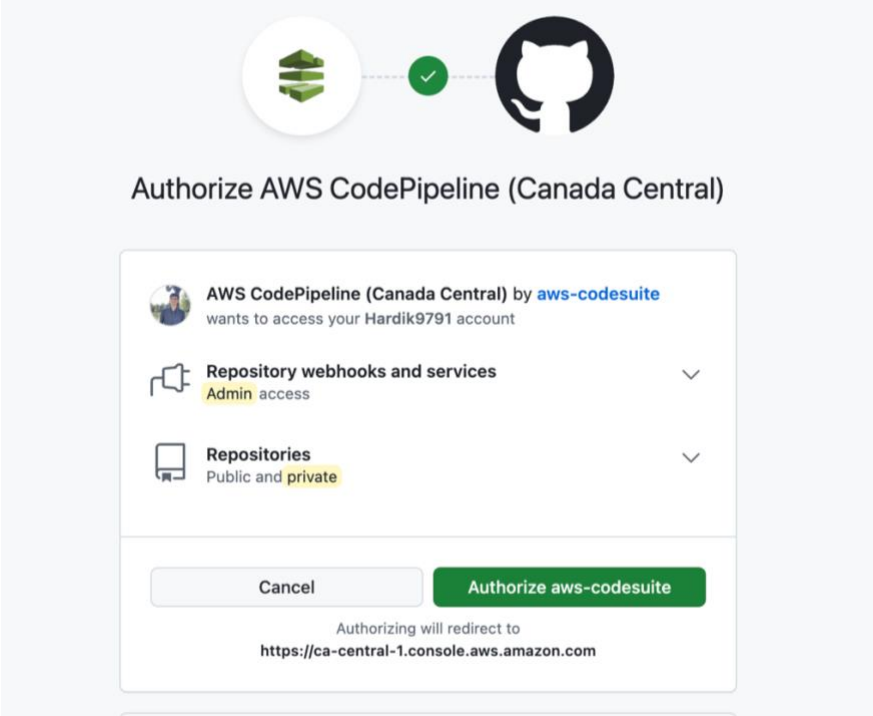


3. Set Up AWS CodePipeline

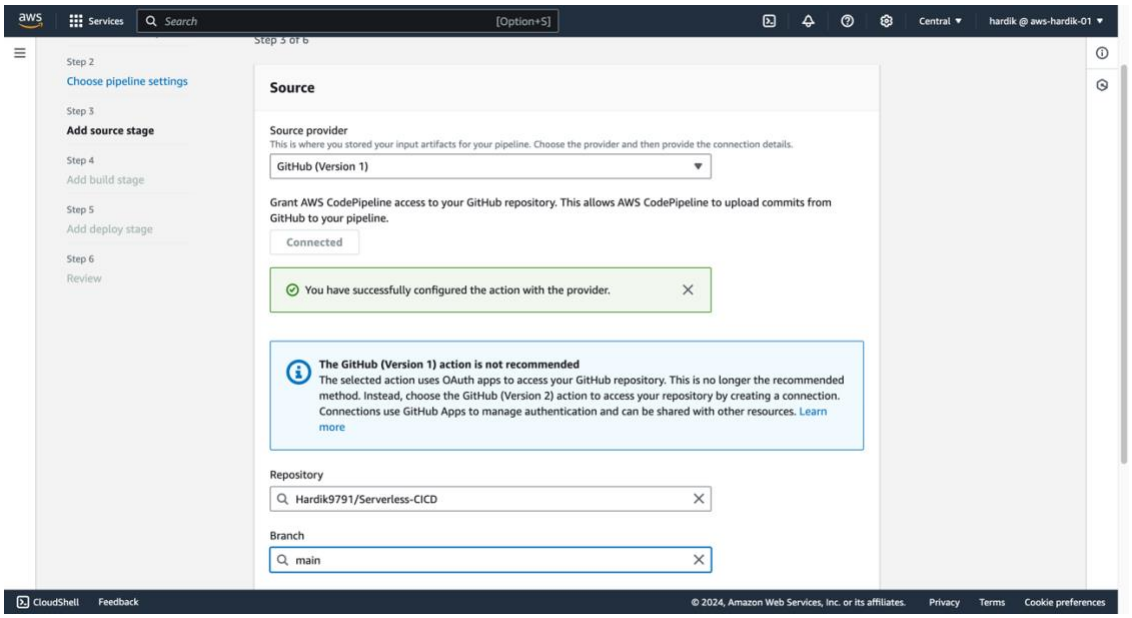
- I created a new **AWS CodePipeline** to automate the deployment process.



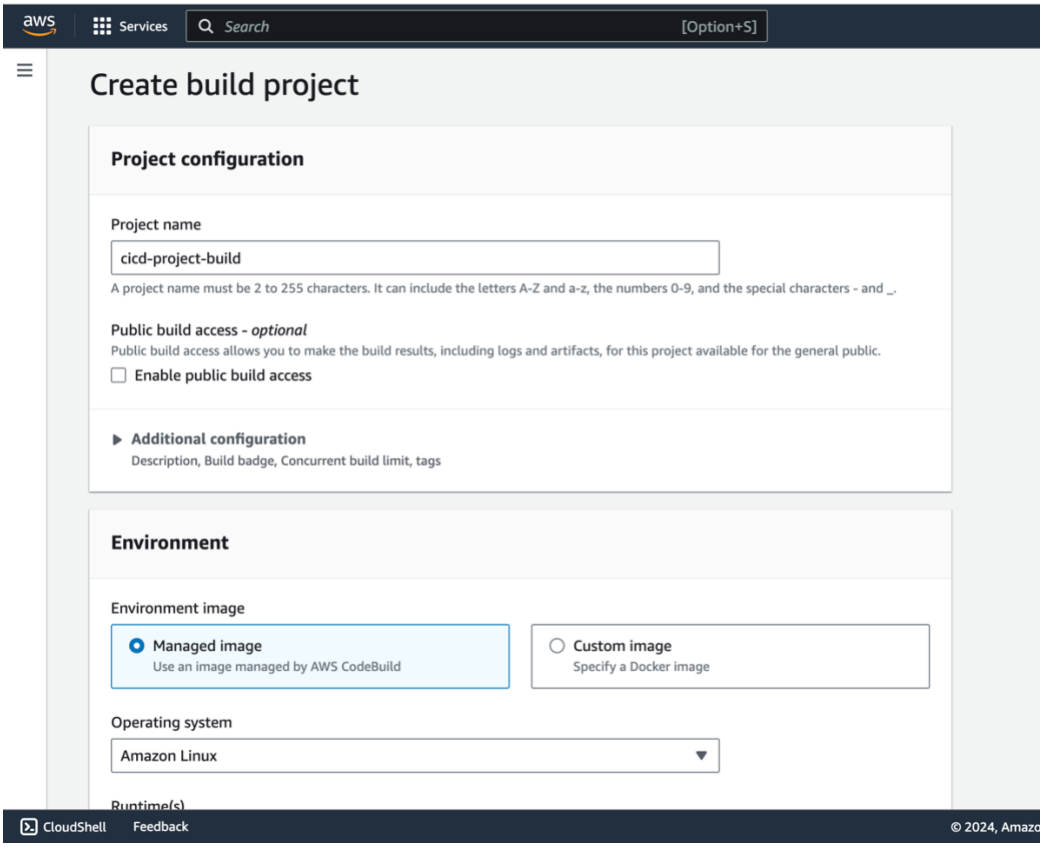
- **Source Stage**: I connected my GitHub repository to the pipeline, so any new commits to the main branch would trigger the pipeline.

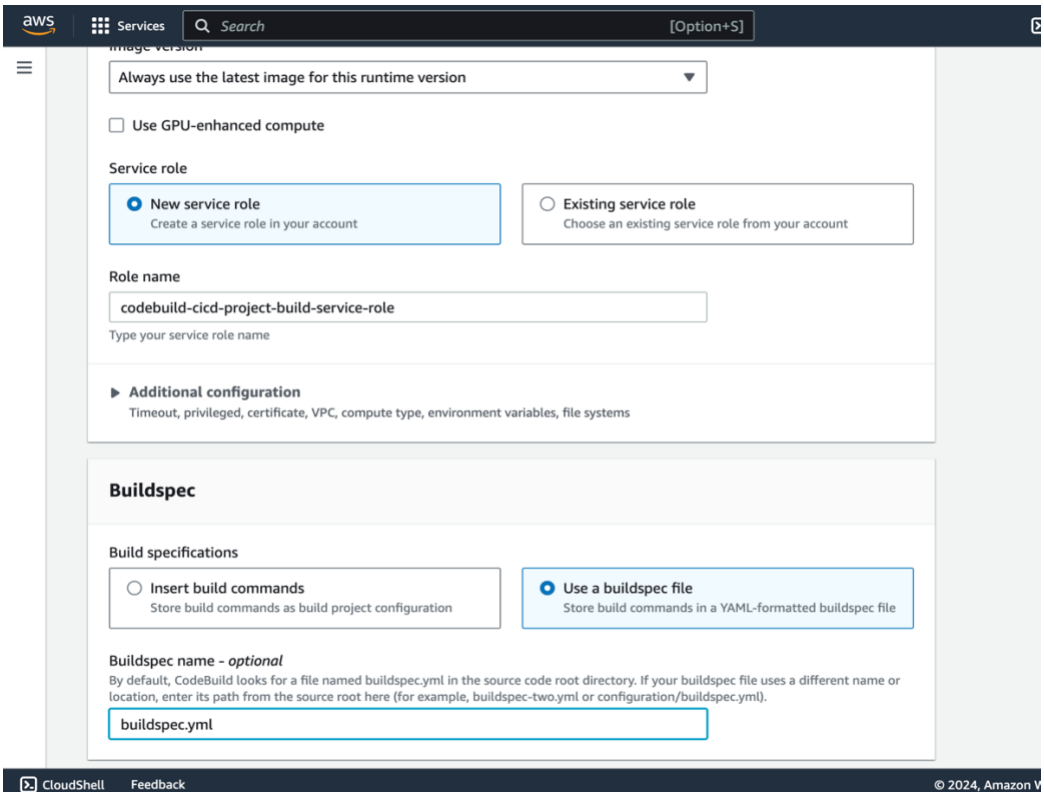


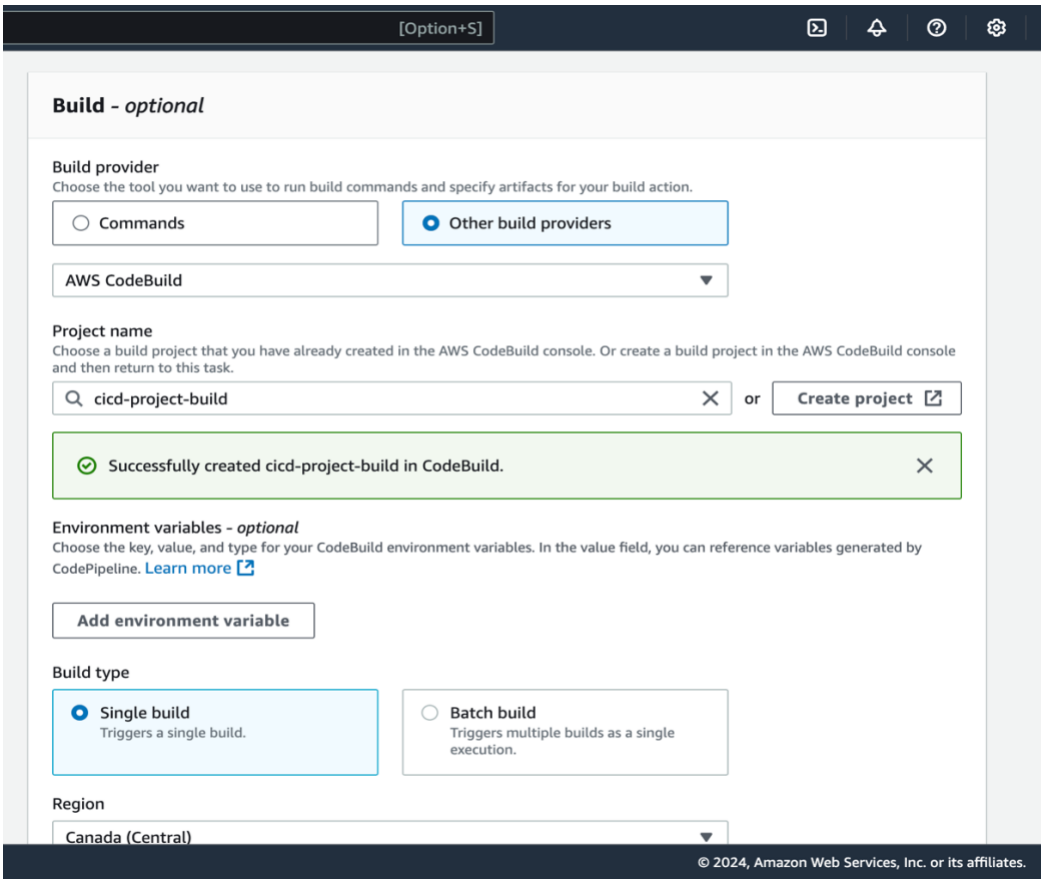
- After connecting, selected GitHub as a source provider.



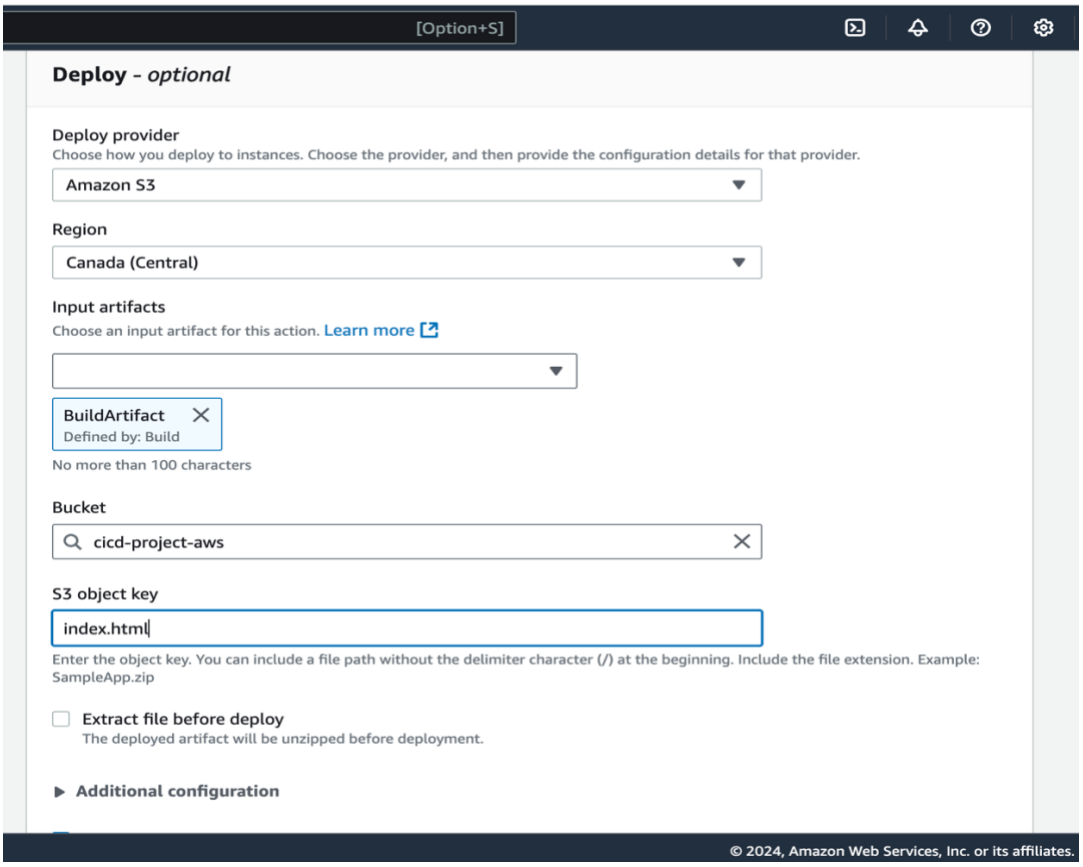
- Build Stage**: I set up** AWS CodeBuild **to package my static website files. In my GitHub repository, I added a "buildspec.yml" file to define how the files are built and prepared for deployment.**

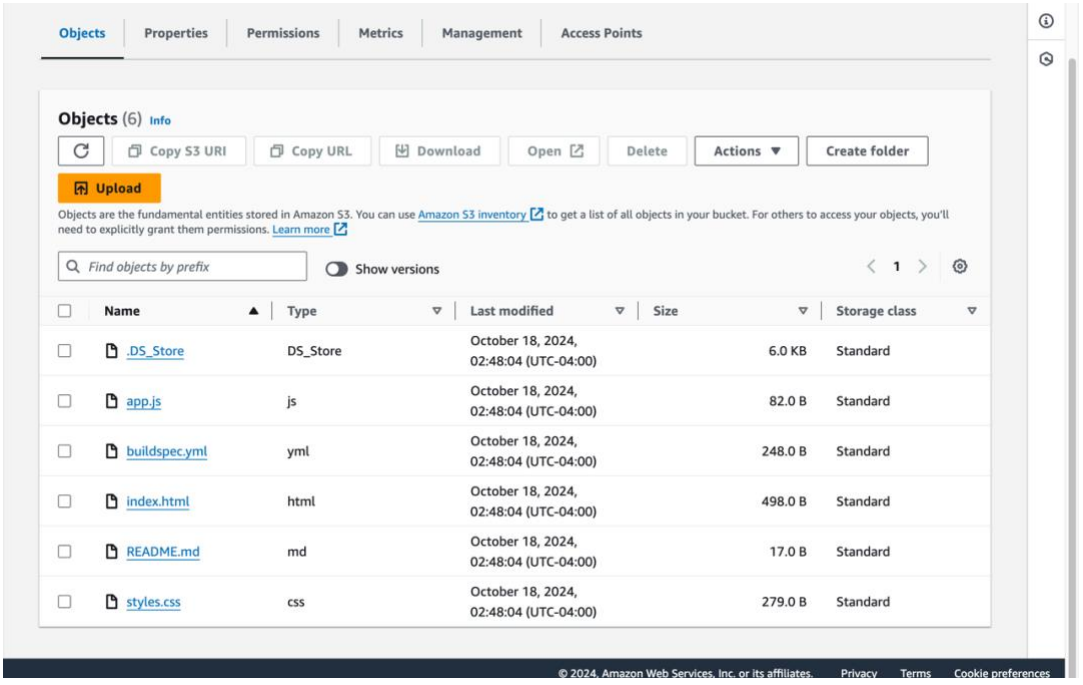



aws
Services
Search
[Option+S]
Always use the latest image for this runtime version
Use GPU-enhanced compute
Service role
New service role
Create a service role in your account
Existing service role
Choose an existing service role from your account
Role name
codebuild-cicd-project-build-service-role
Type your service role name
Additional configuration
Timeout, privileged, certificate, VPC, compute type, environment variables, file systems
Buildspec
Build specifications
Insert build commands
Store build commands as build project configuration
Use a buildspec file
Store build commands in a YAML-formatted buildspec file
Buildspec name - optional
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).
buildspec.yml
CloudShell
Feedback
© 2024, Amazon W


[Option+S]
Build - optional
Build provider
Choose the tool you want to use to run build commands and specify artifacts for your build action.
Commands
Other build providers
AWS CodeBuild
Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.
cicd-project-build
or
Create project
Successfully created cicd-project-build in CodeBuild.
Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. Learn more
Add environment variable
Build type
Single build
Triggers a single build.
Batch build
Triggers multiple builds as a single execution.
Region
Canada (Central)
© 2024, Amazon Web Services, Inc. or its affiliates.

- **Deploy Stage**: I configured the pipeline to deploy the built files to my S3 bucket using the **S3 deploy action**, ensuring that the latest version of the website would be served.





4. Tested the CI/CD Pipeline

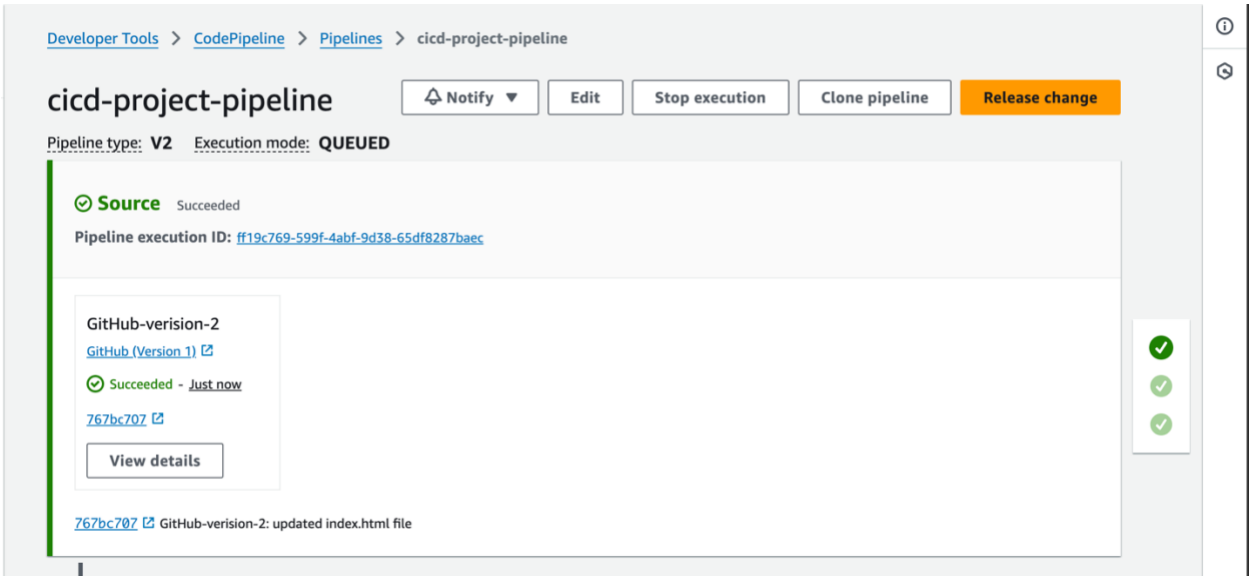
- I made changes to my website's source code in the GitHub repository, including updates to index.html.

```
<div class="container">
    <h1>Welcome to My Static Website</h1>
    <p>This site is hosted on AWS S3 and deployed using an automated CI/CD pipeline.<
    <p>CICD Working!</p>
```

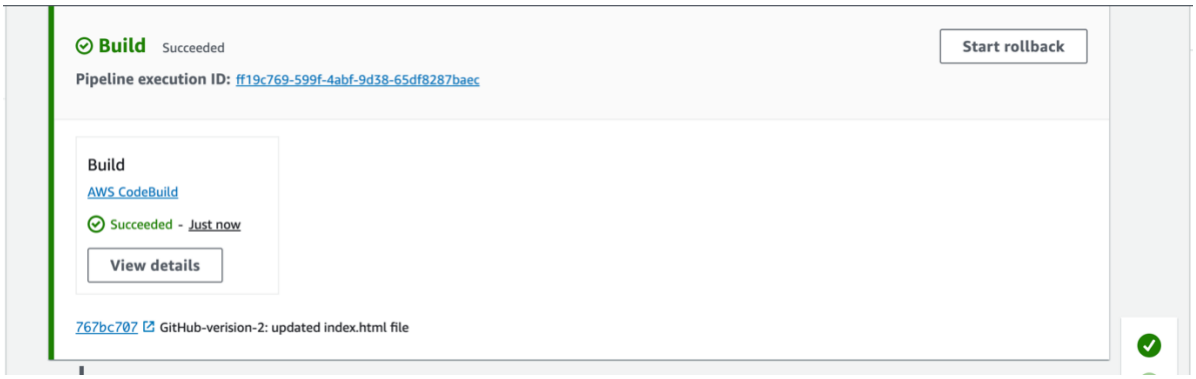
- After pushing the changes to GitHub, the **CodePipeline** automatically triggered the process:

```
[hardikrathod@DA:F8:CA:C3:DD:3C Serverless-CICD % git add .                    ]
[hardikrathod@DA:F8:CA:C3:DD:3C Serverless-CICD % git commit -m "updated index.ht]
ml file"
[main 100d4e0] updated index.html file
 1 file changed, 1 insertion(+), 1 deletion(-)
[hardikrathod@DA:F8:CA:C3:DD:3C Serverless-CICD % git push origin main        ]
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Hardik9791/Serverless-CICD.git
   767bc70..100d4e0  main -> main
hardikrathod@DA:F8:CA:C3:DD:3C Serverless-CICD %
```

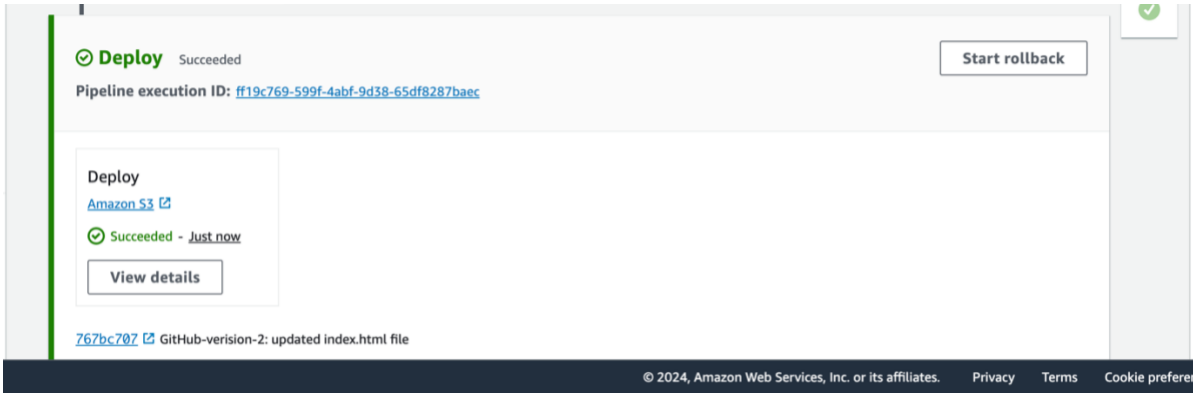
- **The pipeline fetched the new code from GitHub.**



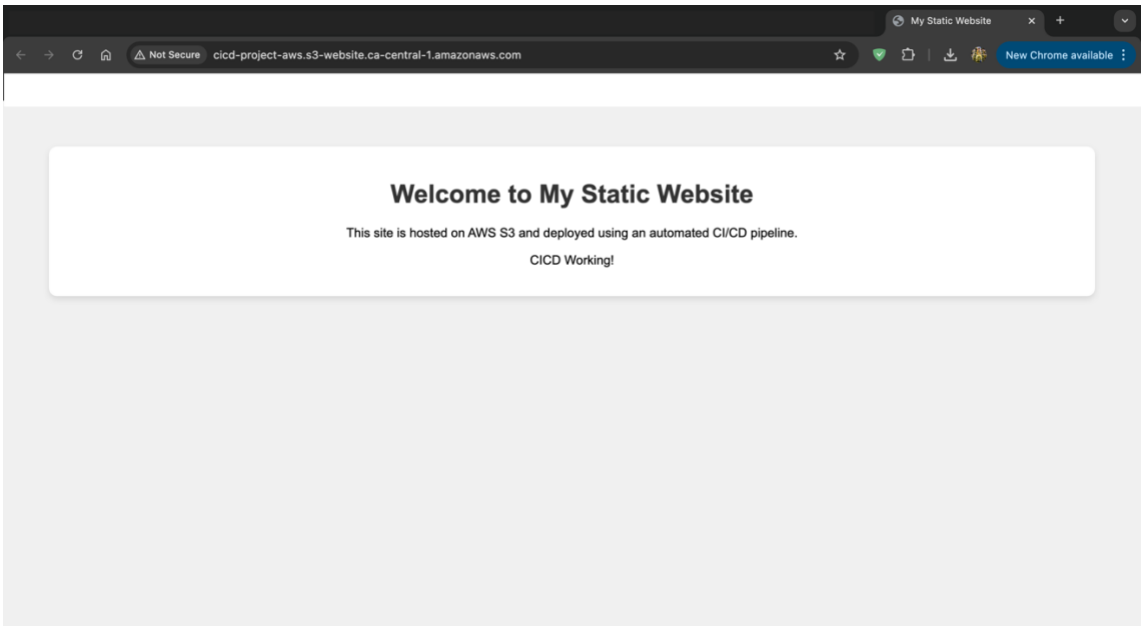
- CodeBuild **packaged the updated files according to my buildspec.yml configuration.**



- The new version of the site was deployed to my **S3 bucket**.



- Finally, I accessed my static website through the **S3 website endpoint** to verify that the changes were deployed successfully.



Project Features

- **Automated Deployments**: The pipeline automatically builds and deploys the website whenever there is a new commit to the GitHub repository.
- **No Servers to Manage**: Since the website is hosted on S3 and deployed using CodePipeline, you don't need to manage any infrastructure.
- **Scalable and Cost-Effective**: S3 provides virtually unlimited scalability, and you only pay for the storage and bandwidth you use.

How the Project Works

1. **Source Code**: The static website's source code (HTML, CSS, JavaScript) is stored in a **GitHub repository**.
2. **Pipeline Trigger**: When changes are pushed to the GitHub repository, the **CodePipeline** is automatically triggered to start the deployment process.
3. **Build Process**:
    - **AWS CodeBuild** packages the static website files based on the instructions defined in the `buildspec.yml` file.
    - The files are prepared for deployment, ensuring that the static assets (HTML, CSS, JS) are correctly bundled.
4. **Deployment**:
    - The packaged files are deployed to **Amazon S3**, where they are made publicly accessible.
    - Proper deployment configuration ensures that all files are extracted and placed in the root of the S3 bucket, making the website accessible.
5. **Accessing the Website**: Users can access the website via the **S3 website endpoint**, which serves the static files directly over the web.

Challenges Faced

- **Deployment Configuration Issue**: I initially set the S3 key object as `index.html` in the deploy stage, which caused all files to be bundled into a single `index.html` file and compressed into a zip. This made the static website inaccessible.
- **Solution**: I corrected the issue by removing the `index.html` from the S3 key object and choosing the option to **extract all files before deployment**. This allowed the files to be deployed properly, and I was able to access my static website successfully.

Skills Highlighted

- **AWS Services Expertise**: Demonstrated proficiency in AWS S3, CodePipeline, and CodeBuild, effectively automating the deployment process for a static website.
- **Problem Solving**: Identified and resolved issues with file extraction and deployment paths, ensuring the static website was accessible post-deployment.**CI/CD Pipeline**
- **Automation**: Set up a fully automated CI/CD pipeline, showcasing knowledge of continuous integration and deployment processes using AWS services.
- **GitHub Integration**: Successfully integrated GitHub with CodePipeline to trigger automatic deployments based on new commits, highlighting proficiency in version control and source management.

Conclusion

This project provides a fully automated, serverless CI/CD pipeline for deploying static websites using AWS services. By leveraging **S3**, **CodePipeline**, and **CodeBuild**, we achieved continuous deployment with minimal setup and no manual intervention required after code changes.