Static Website Deployment on AWS S3 – Capstone Project Documentation

1. Introduction

This project demonstrates the deployment of a static website on Amazon Web Services (AWS) S3 with automated updates using GitHub Actions.
The goal was to understand how to host a static website in the cloud and automate deployment whenever new changes are pushed to GitHub.
The website used for this deployment was built using the HTML5 UP – Dimension template, with personalized text content to make it unique.

2. Architecture Overview

The system architecture is a simple static hosting setup using Amazon S3, supported by a CI/CD workflow from GitHub.
The Bash script handles manual deployment, while GitHub Actions automates the process after every code update.
Architecture Flow:
Local Files → GitHub Repository → GitHub Actions Workflow → AWS S3 → Live Website

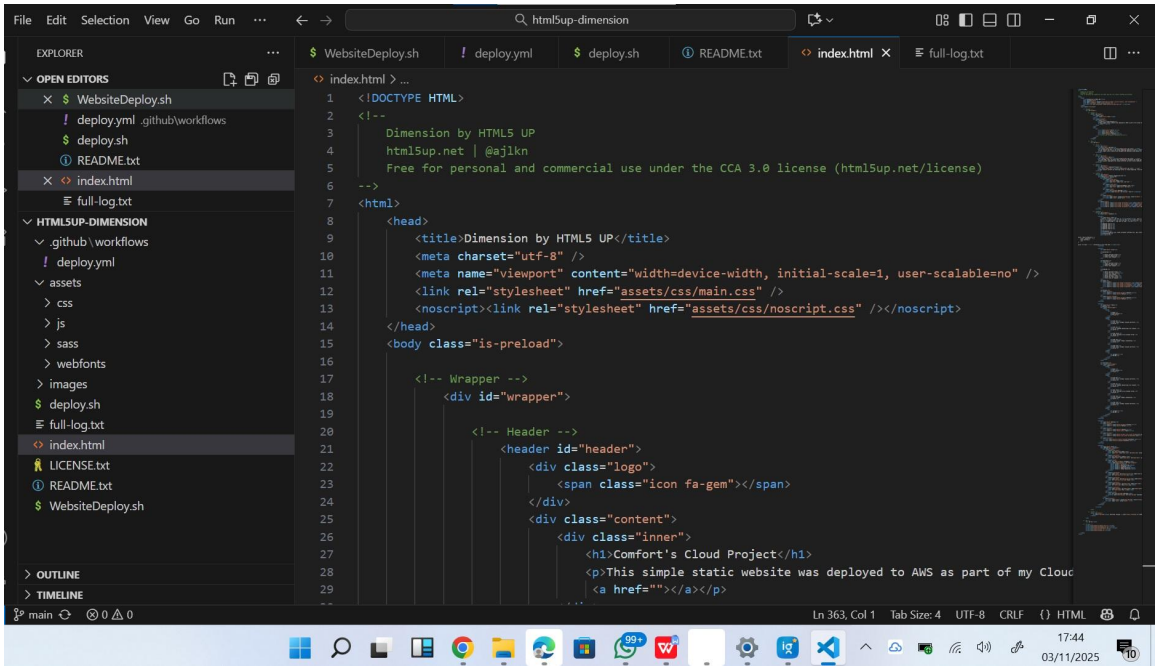
3. Tools and Technologies Used

- Amazon S3 – for hosting and storage
- AWS CLI – for command-line configuration and deployment
- Bash Script – to automate local deployment
- GitHub Actions – for automated deployment on push
- HTML5 UP (Dimension Template) – for the static website
- GitHub – for version control and automation

4. Step-by-Step Deployment Process

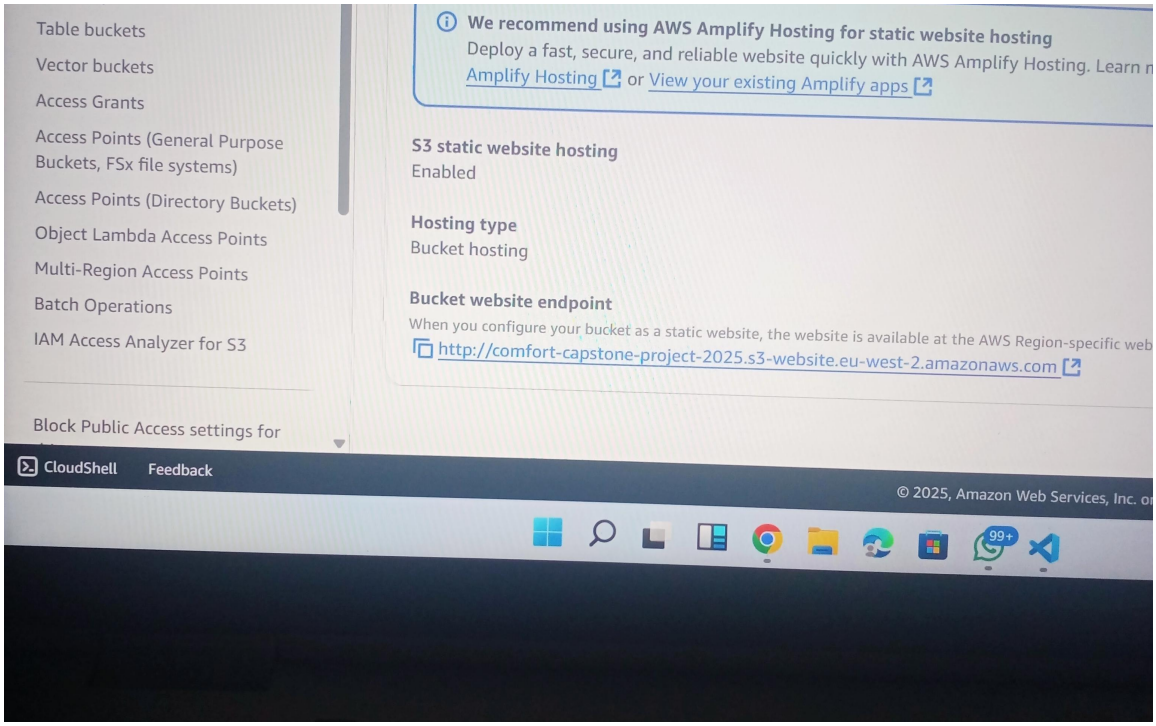
Step 1: Prepare Website Files

Downloaded the HTML5 UP – Dimension template and customized the text to include my own details.
Files were stored in the website/ folder for deployment.
[Insert Screenshot Here]



Step 2: Create an S3 Bucket

Created an S3 bucket named comfort-capstone-project-2025 in the eu-west-2 (London) region.
Enabled static website hosting and disabled “Block all public access”.
Configured the index and error documents.
[Insert Screenshot



Step 3: Configure Bucket Policy

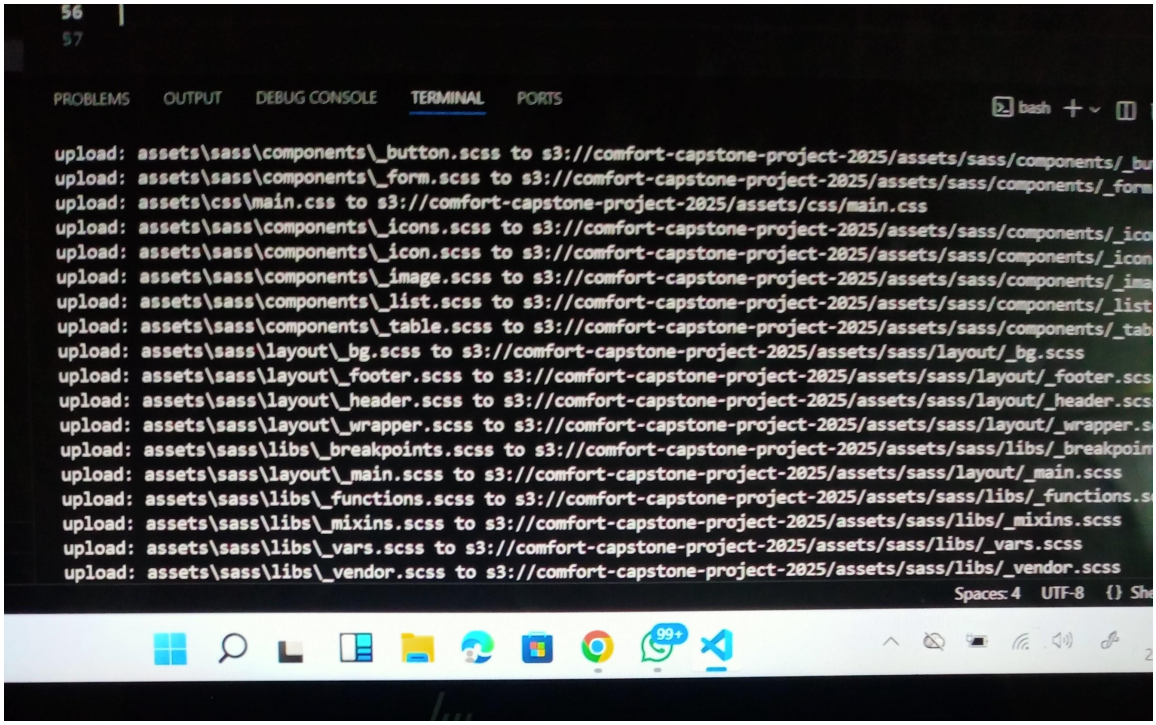
Configured a public read access policy to make the website publicly viewable.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "PublicReadGetObject",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "s3:GetObject",
   "Resource": "arn:aws:s3:::comfort-capstone-project-2025/*"
  }
 ]
}
```

Step 4: Deploy Using Bash Script

Created a Bash script to automate deployment from the local machine using AWS CLI.

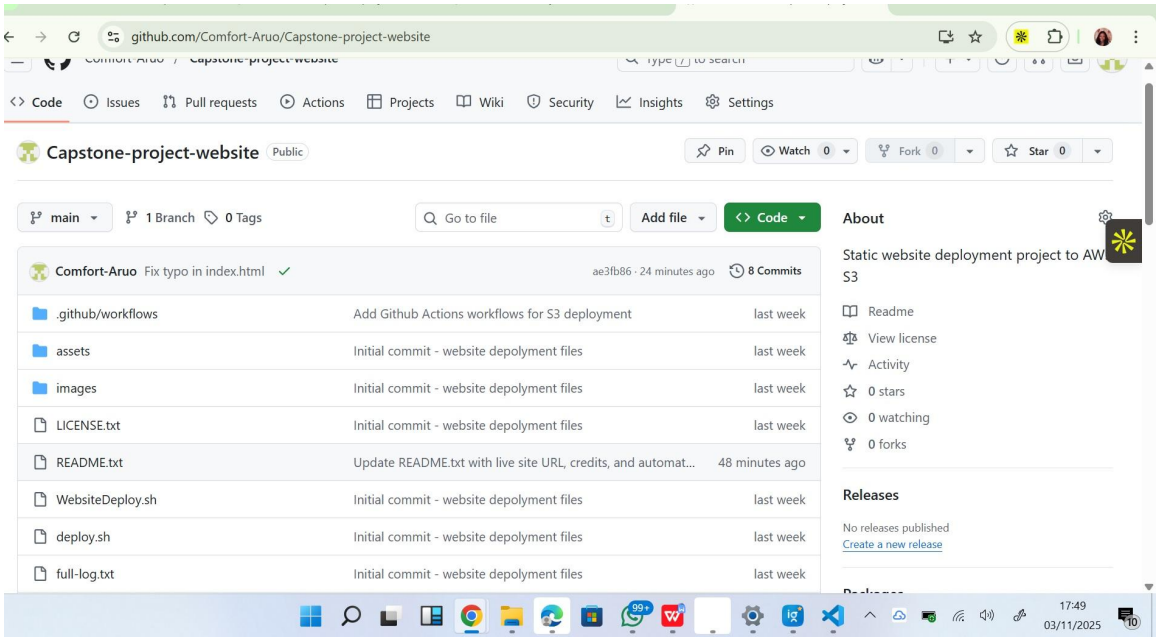
```
#!/bin/bash
# Comfort's Capstone Deployment Script

BUCKET_NAME="comfort-capstone-project-2025"
REGION="eu-west-2"

echo "Starting deployment to S3 bucket: $BUCKET_NAME..."

aws s3 website s3://$BUCKET_NAME/ --index-document index.html --error-document
index.html
echo "✅ Static website hosting enabled."

aws s3 sync . s3://$BUCKET_NAME --delete
echo "✅ Files uploaded successfully!"

echo "Your static website is live at:"
echo "    http://$BUCKET_NAME.s3-website-$REGION.amazonaws.com"
echo "    Deployment complete!"
```



Step 5: Automate Deployment with GitHub Actions (Bonus Task)

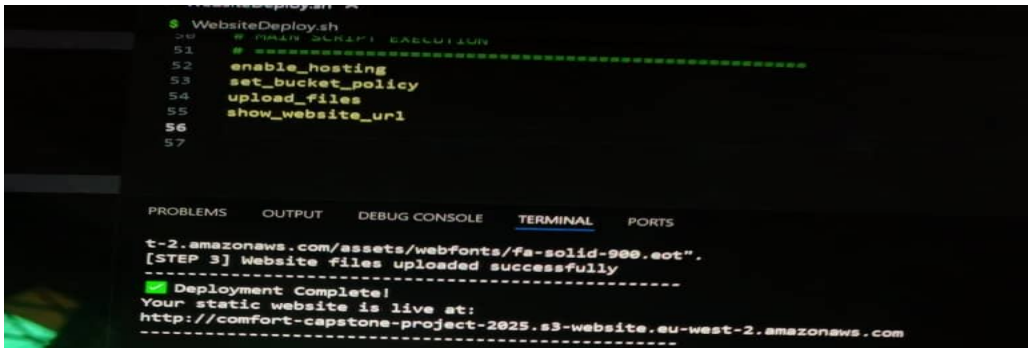
Configured GitHub Actions to automatically deploy new updates to S3 whenever changes are pushed to the main branch.

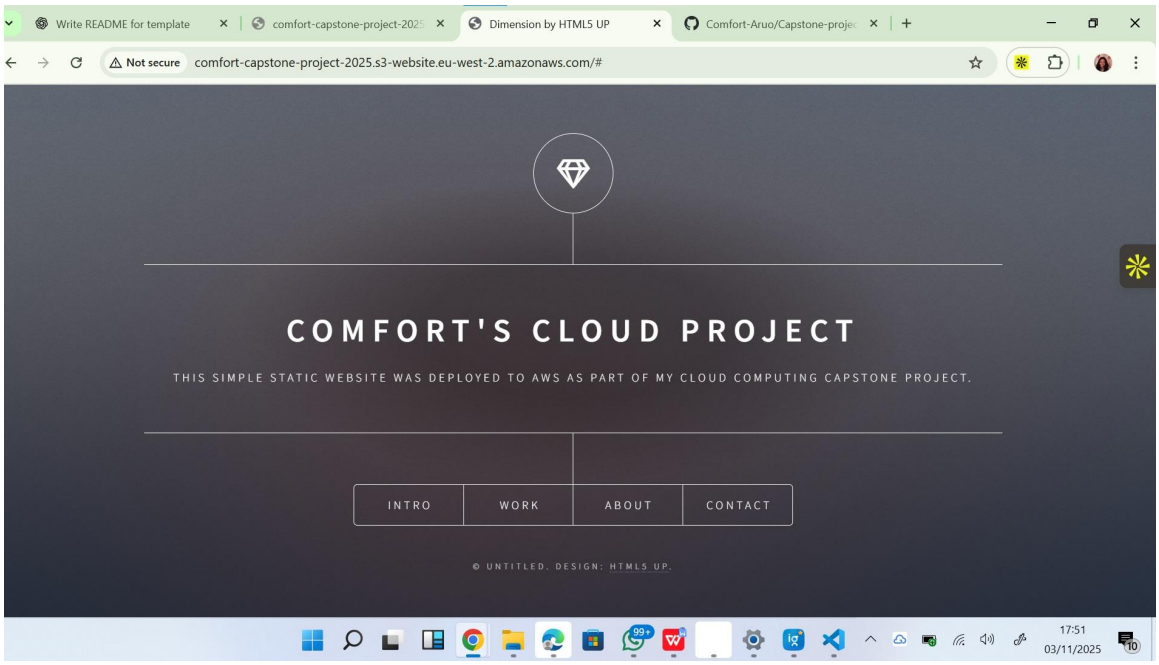
```
name: Deploy to S3
on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v3
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: eu-west-2

      - name: Sync files to S3
        run: aws s3 sync . s3://comfort-capstone-project-2025 --delete
```



Step 6: Verify Deployment

Website was accessible publicly at:
http://comfort-capstone-project-2025.s3-website-eu-west-2.amazonaws.com
[Insert Screenshot Here]





5. Challenges and Solutions

Challenge 1: After running my script I discovered that my website couldn't go live as I got an error code of 403. I had to go back to my AWS Console to change the settings of my bucket from 'Block Public Access' by unchecking it to make it public and allow public read access. After doing this, I still got an error code 404 (Not Found). After researching, I discovered that the bucket was empty — meaning the hosting setup was correct, but no files were uploaded yet. I fixed this issue by returning to my terminal and running the AWS CLI command 'aws s3 sync' to upload my files. Afterward, I verified on the AWS S3 Console that the files had been successfully uploaded.

Challenge 2: I encountered multiple trial-and-error attempts trying to automate the GitHub Actions. The issue was due to syntax errors in my YAML script. After correcting the syntax, I was able to successfully deploy the GitHub Actions for my static website.

6. Learning Outcomes

- Learned manual and automated deployment using AWS S3 and GitHub Actions
- Understood CI/CD concepts in real-world cloud environments
- Improved understanding of AWS CLI commands and permissions
- Gained hands-on experience using automation tools for cloud workflows

7. Conclusion

This project enhanced my understanding of cloud automation.
I successfully deployed and automated updates to my static website hosted on AWS S3.
The bonus task of integrating GitHub Actions allowed me to achieve a full CI/CD pipeline that updates my site automatically after each push.

8. References

- AWS S3 Documentation – https://docs.aws.amazon.com/s3/
- GitHub Actions for AWS – https://github.com/aws-actions
- HTML5 UP Templates – https://html5up.net
- AWS CLI Reference – https://docs.aws.amazon.com/cli/latest/index.html