Project Title :- Hosting a Carvilla Website Using AWS S3 Services

- **Hosting a Carvilla Website Using AWS S3 Services:**
  This document provides a step-by-step guide to host a Carvilla static website using Amazon S3, which is an object storage service offered by AWS. The guide demonstrates how to configure your S3 bucket for public static website hosting and how to manage permissions and logging.

- **Key Features & Implementations:**

1. **Static Frontend Hosting:**
   HTML/CSS/JavaScript-based UI mimicking the Car shopping experience.

2. **Fast and Scalable Hosting via AWS S3:**
   Hosted on an Amazon S3 bucket configured for static website hosting.
   Auto-scales globally with S3's performance and reliability.

3. **Public Accessibility:**
   Static website is publicly accessible using a clean S3 website URL.
   Bucket policies and ACLs allow public read access to site files.

4. **Versioning and Backup:**
   S3 Versioning enabled to keep track of all versions of the files.
   Allows easy rollback and backups in case of issues.

5. **Access Control:**
   Controlled via AWS IAM policies, ensuring only authorized users can update or manage the bucket. Fine-grained permissions allow for dev/admin role separation.

6. **Logging and Monitoring:**
   Server access logging enabled to track visits and activity on the site.
   Logs stored in a separate S3 bucket (e.g., carvilla-log) for analytics and auditing.

7. **(Optional) CI/CD Deployment:**
   Integration with GitHub Actions or AWS CodePipeline for automated file uploads on commit.

- **Technology Used:-**
  The Carvilla static website hosted on AWS S3 primarily uses the following technologies:

1. **Frontend Technologies:**

   - HTML – For structuring the content
   - CSS – For styling (can include frameworks like Bootstrap or Tailwind)
   - JavaScript – For client-side interactivity (can include jQuery or vanilla JS)
   - Images & Assets – Product images, logos, etc.

2. **Hosting & Cloud Infrastructure:**
   These AWS technologies are used for hosting:

   - Amazon S3 (Simple Storage Service) – Main hosting service for static content
   - S3 Static Website Hosting – Special configuration to serve files as a website
   - Access Control Lists (ACLs) – For managing public access to files
   - Bucket Policies – (Optional) For fine-grained access control
   - Versioning – To keep backup versions of your files
   - Server Access Logging – For tracking usage and access.

- ## Outcome :-

1. **Live Static Website:**
   The Carvilla Static Website is publicly accessible through a custom S3 website URL, simulating a professional e-commerce front end.

2. **Secure & Scalable Hosting:**
   AWS S3 provides high availability and durability for static content, making your site reliable without server management.

3. **Cost-Effective Deployment:**
   Since S3 is a pay-as-you-go service, hosting static sites is extremely cheap—ideal for prototypes, portfolios, and frontend projects.

- ## Impact:-

1. **Real-World Cloud Experience:**
   Demonstrates practical knowledge of AWS infrastructure, particularly for web developers and cloud beginners.

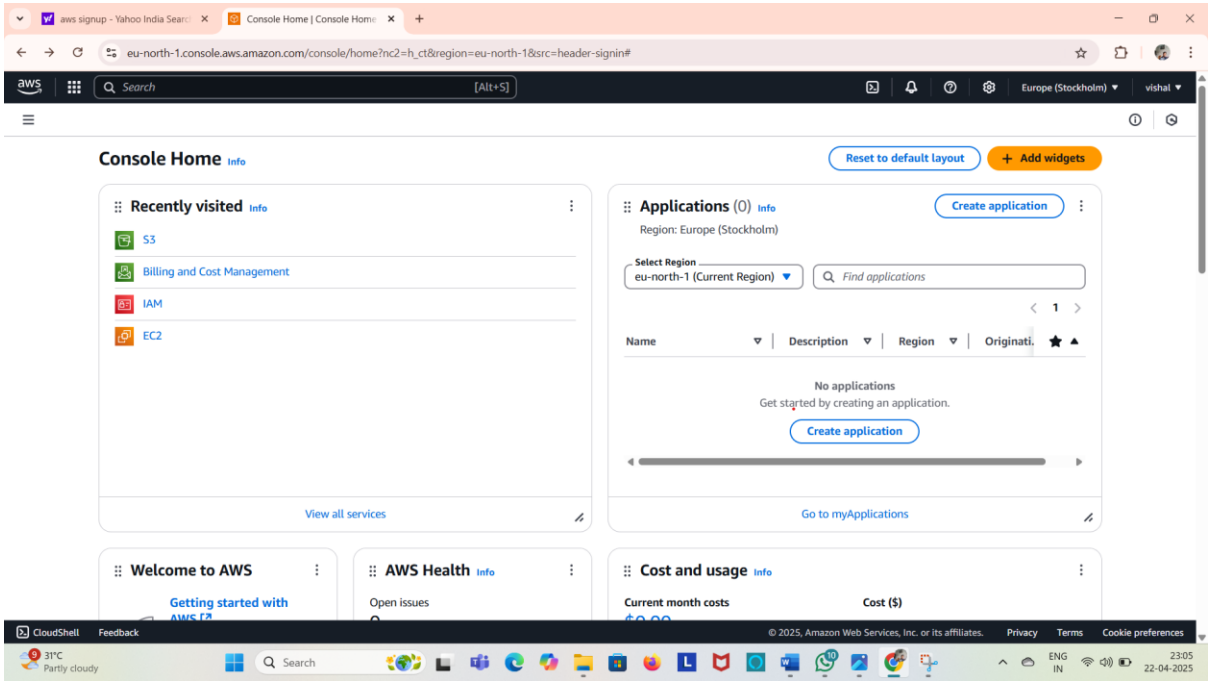
2. **Portfolio Enhancement:**
   Hosting a live clone of a known platform (like Carvilla) showcases your frontend and cloud skills to potential employers or clients.

3. **Foundation for Future Projects:**
   Sets the stage for enhancements like integrating a backend (with Lambda, API Gateway, or Firebase), adding a domain name, enabling HTTPS with CloudFront, or CI/CD pipelines.

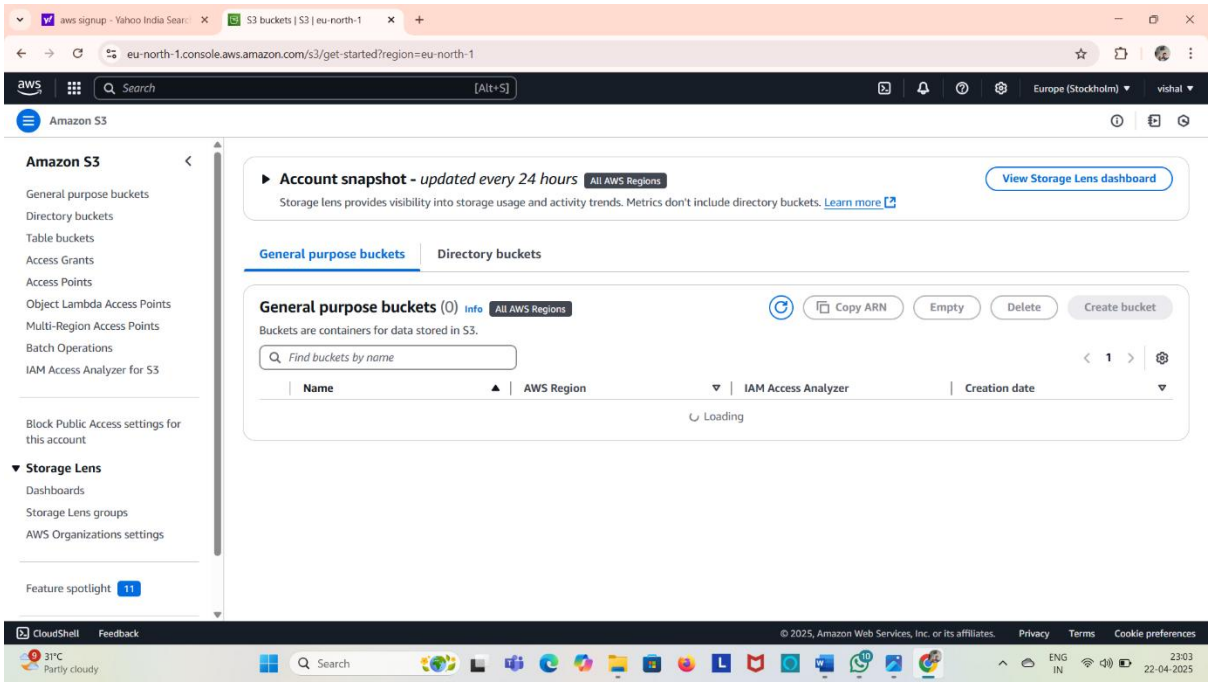
- **Step-by-Step Guide:-**

Step 1: Create S3 Bucket:

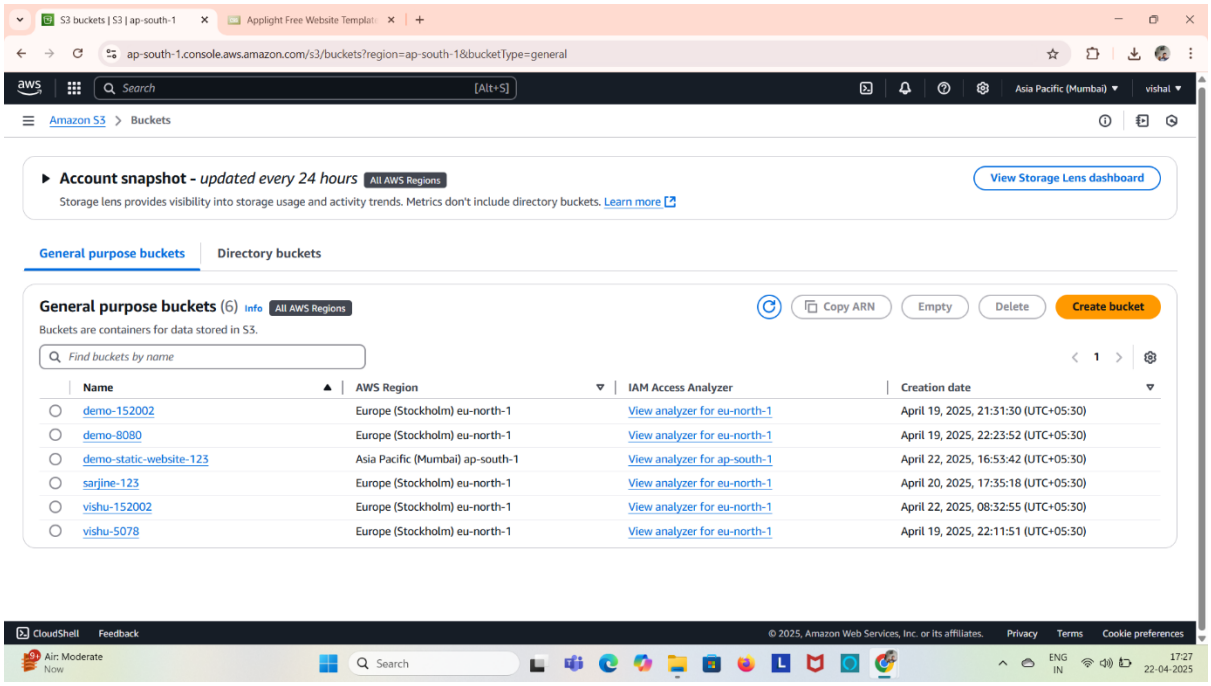
1. Open the AWS Management Console



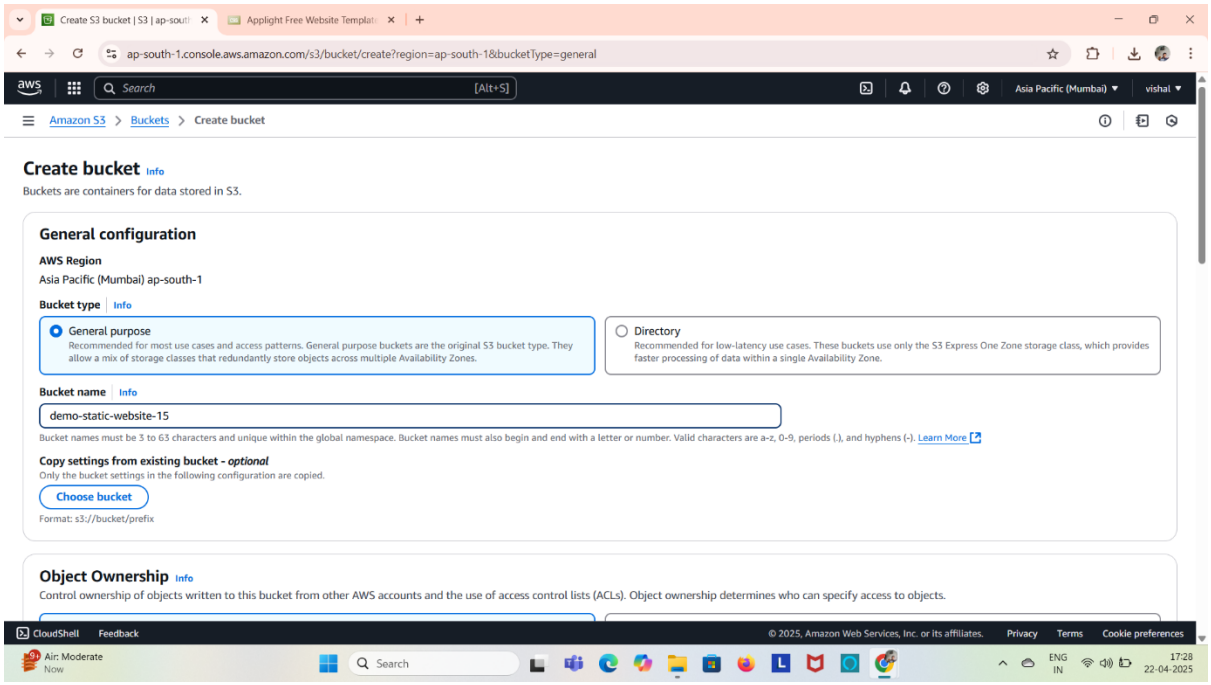
2. Navigate to S3 under the Services section



3. Click on “Create bucket”

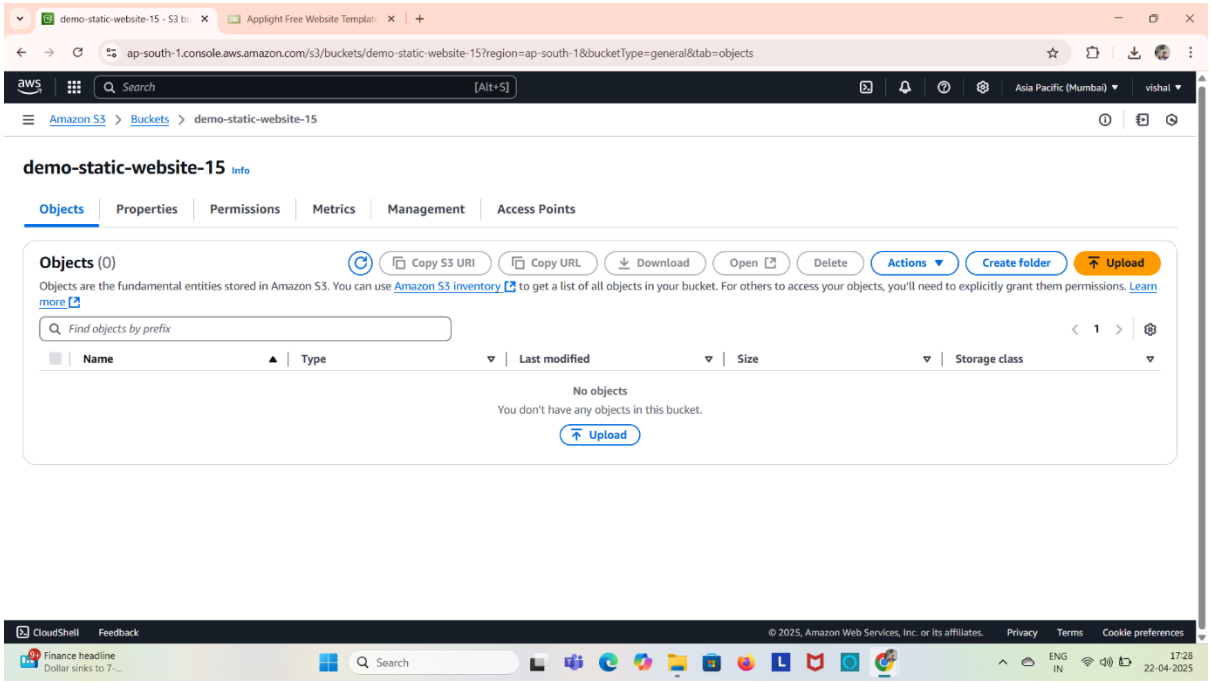


4. Name the bucket `demo-static-website-15` (Bucket names must be globally unique).

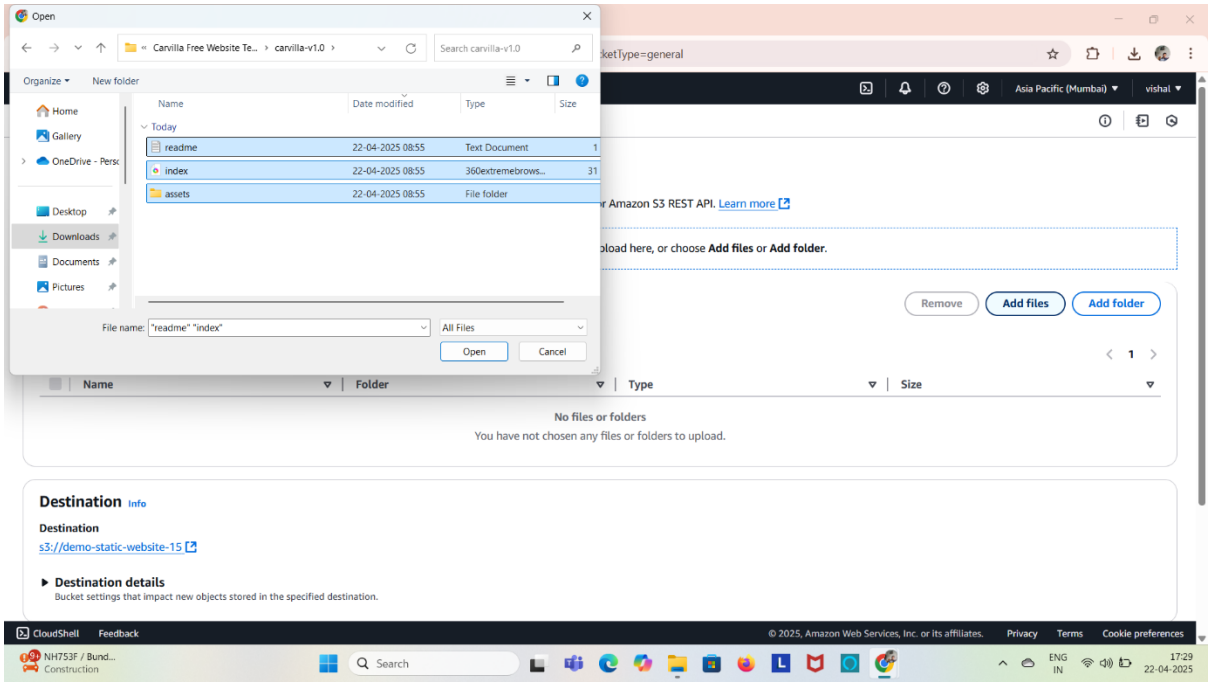


Step 2: Upload Website Files:

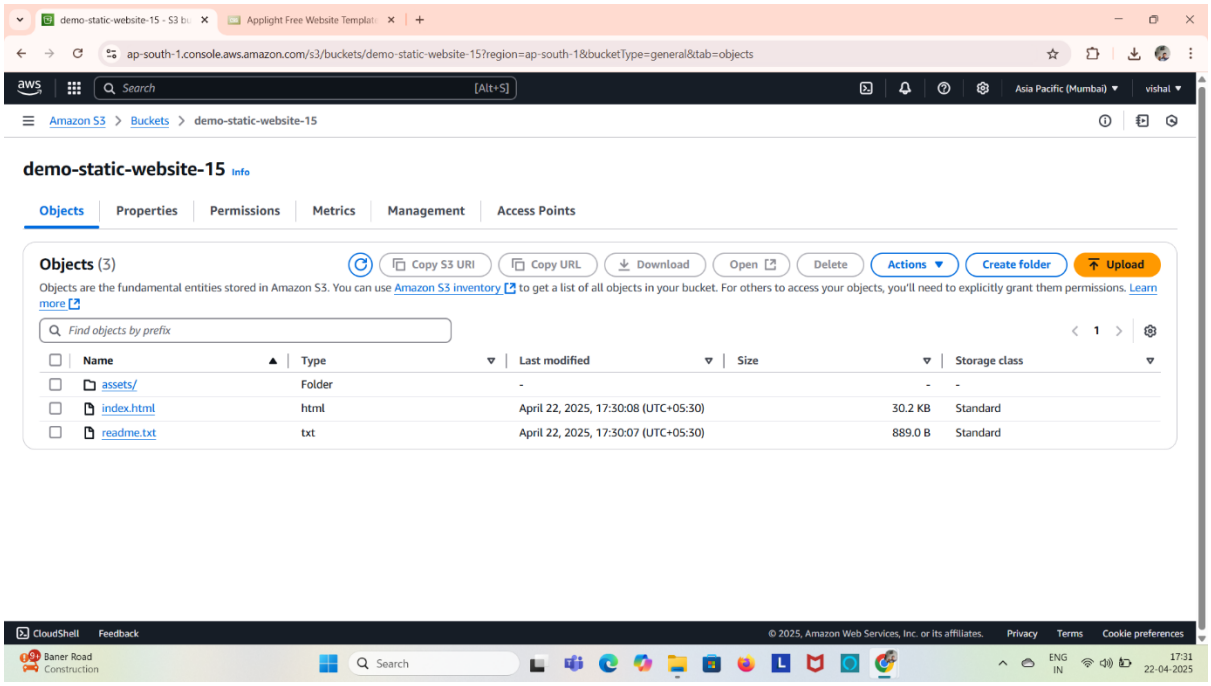
1. After creating the bucket, go inside it.



2. Click on the Upload button.

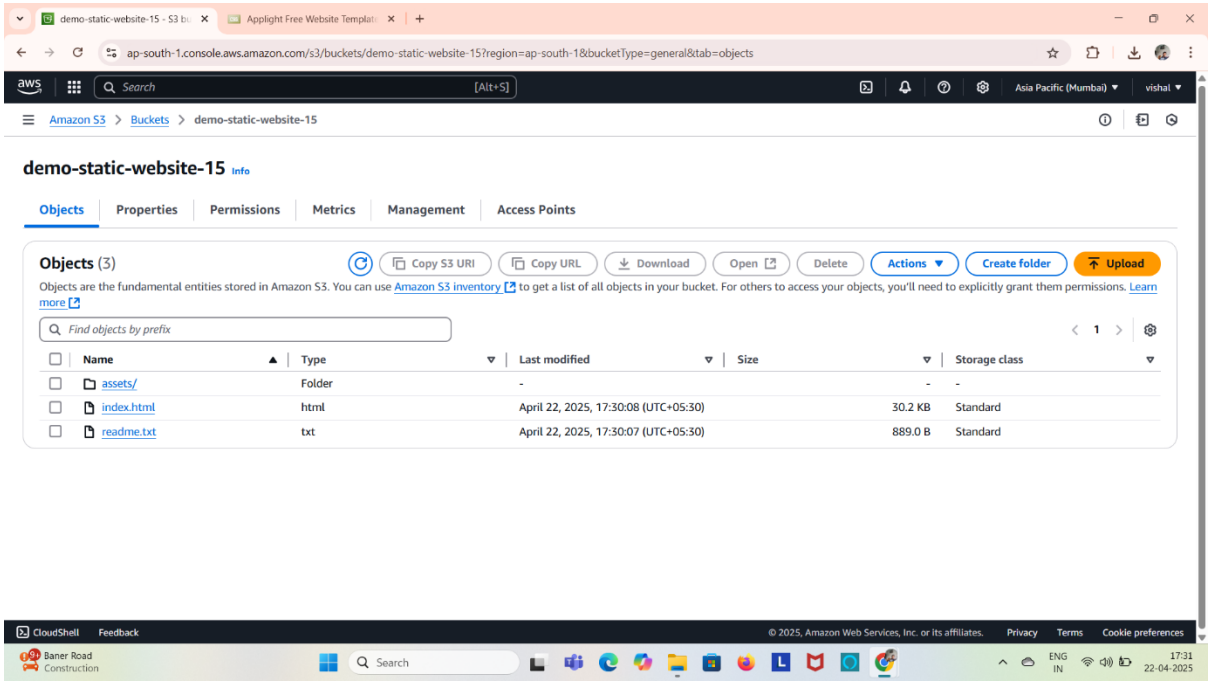


3. Upload all your website's static files — like `index.html`, `styles.css`, `script.js`, images, etc.

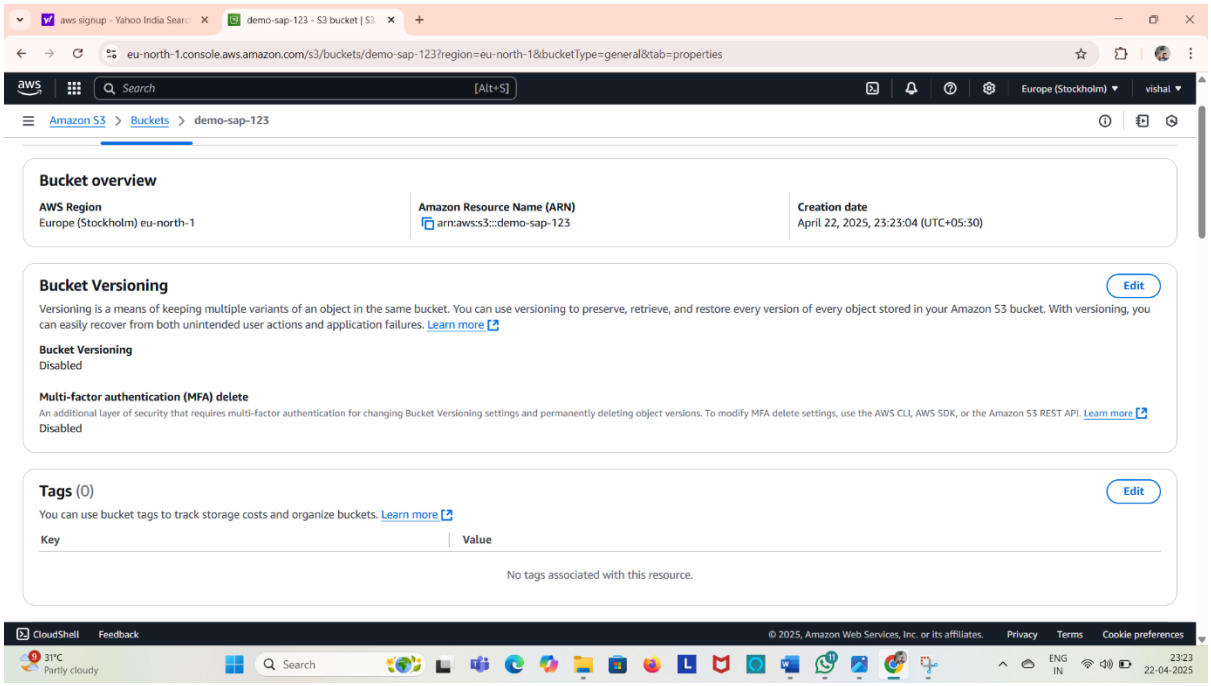


Step 3: Enable Bucket Versioning:

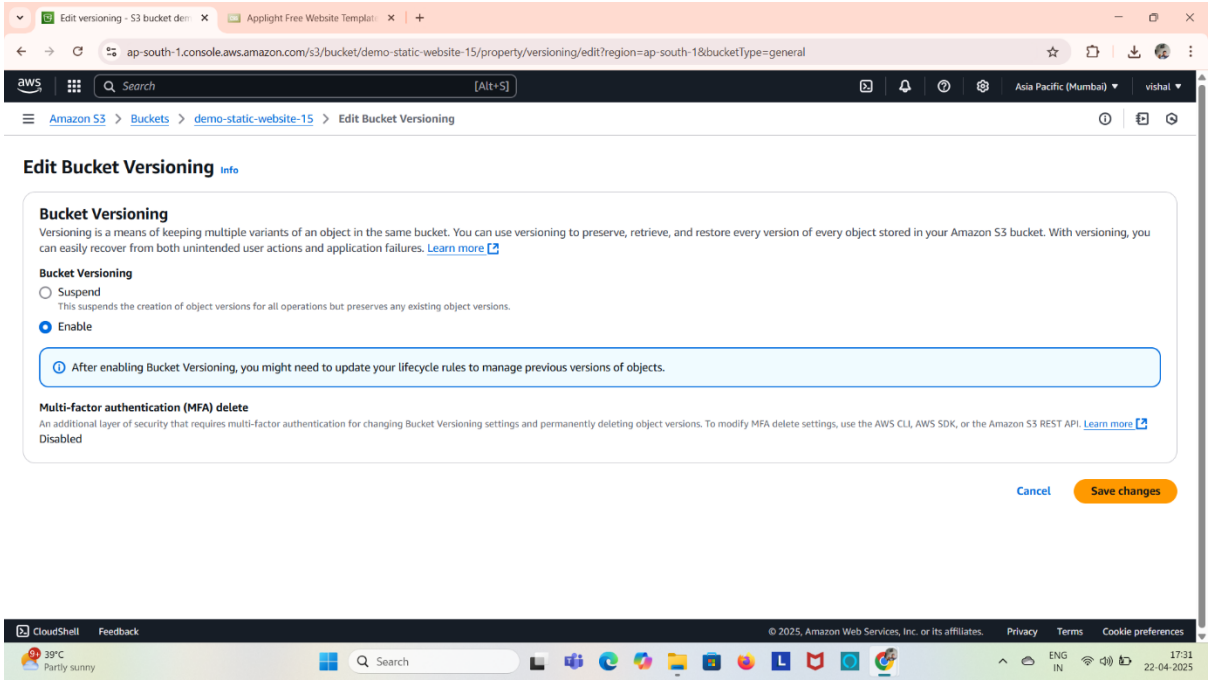
1. Go to the Properties tab of the bucket.



2. Scroll to the Bucket Versioning section.

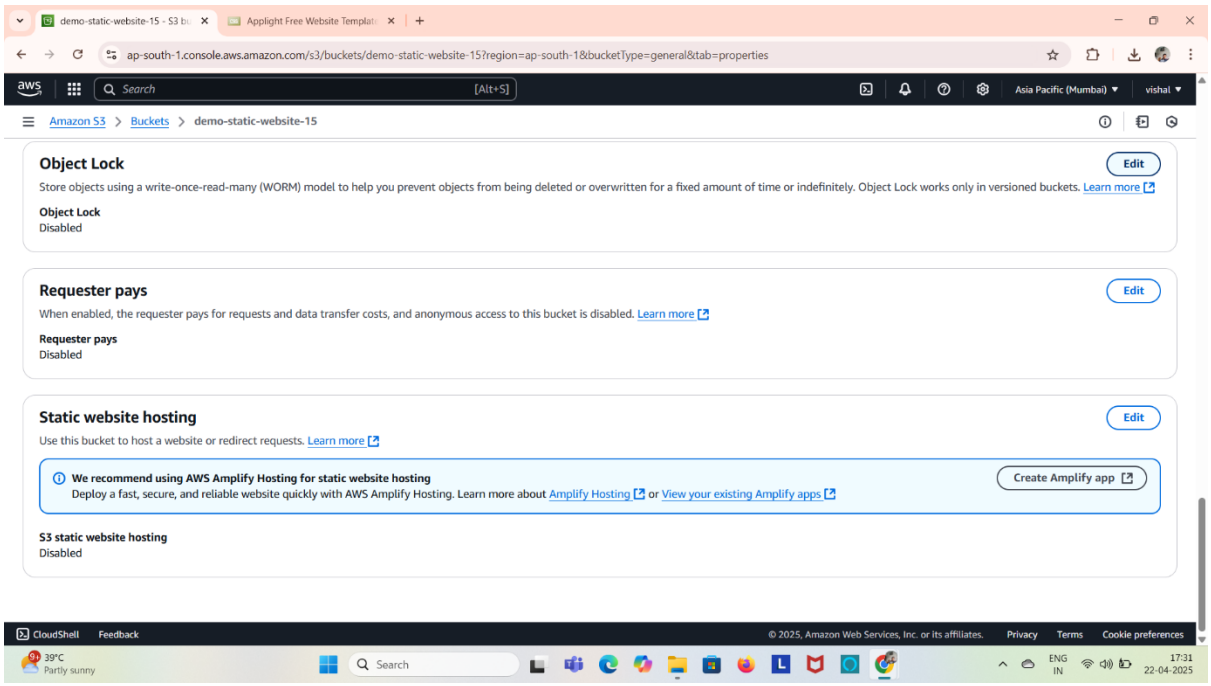


3. Click Edit and enable versioning.

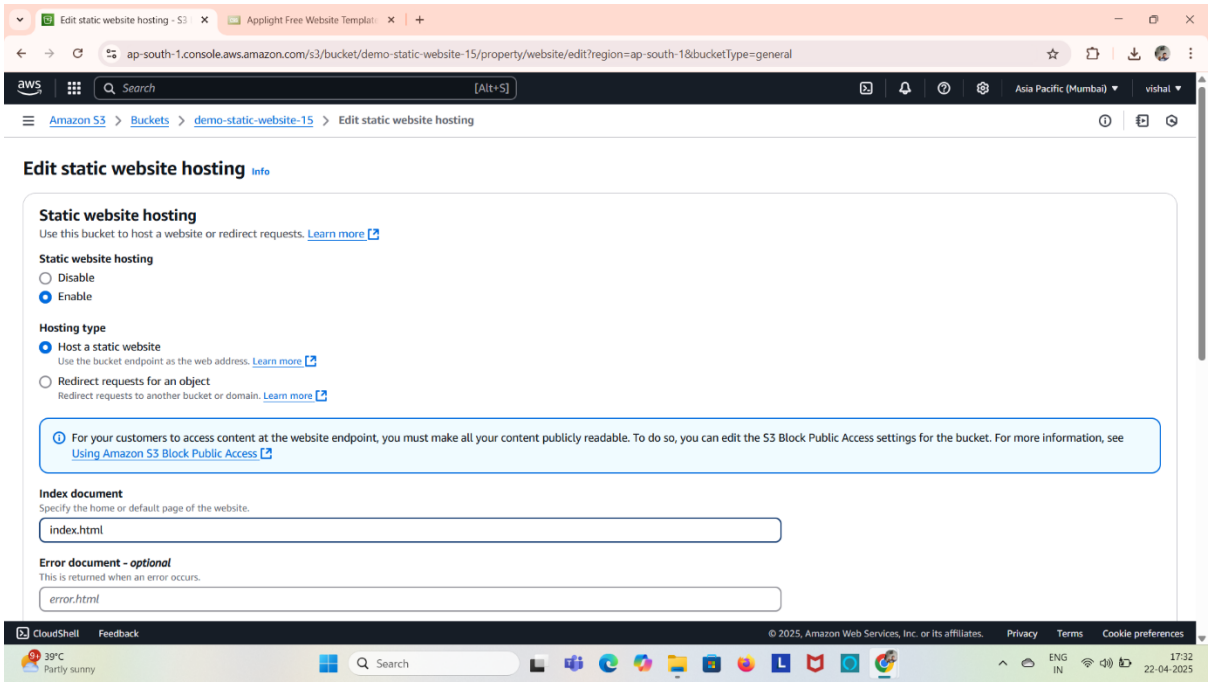


Step 4: Enable Static Web Hosting:

1. In the Properties tab, scroll to Static website hosting.

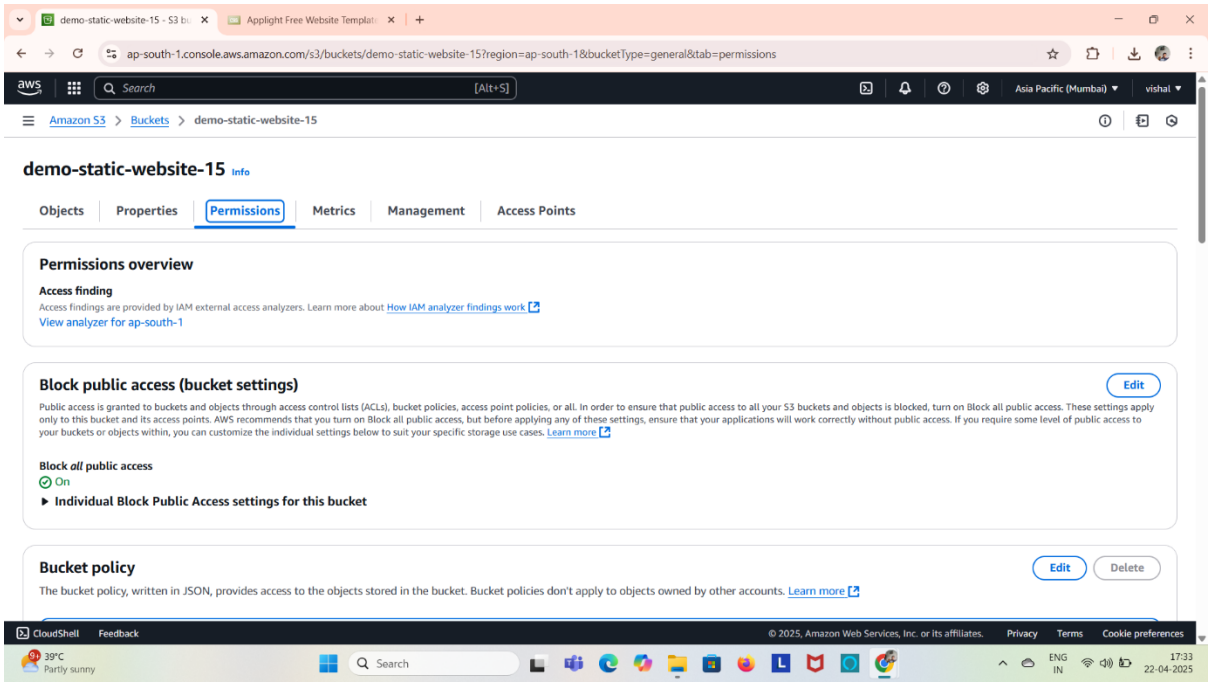


2. Click Edit, then:
   - Select Enable.
   - Set the Index document to `index.html`.
   - (Optional) Set an error document like `error.html`.

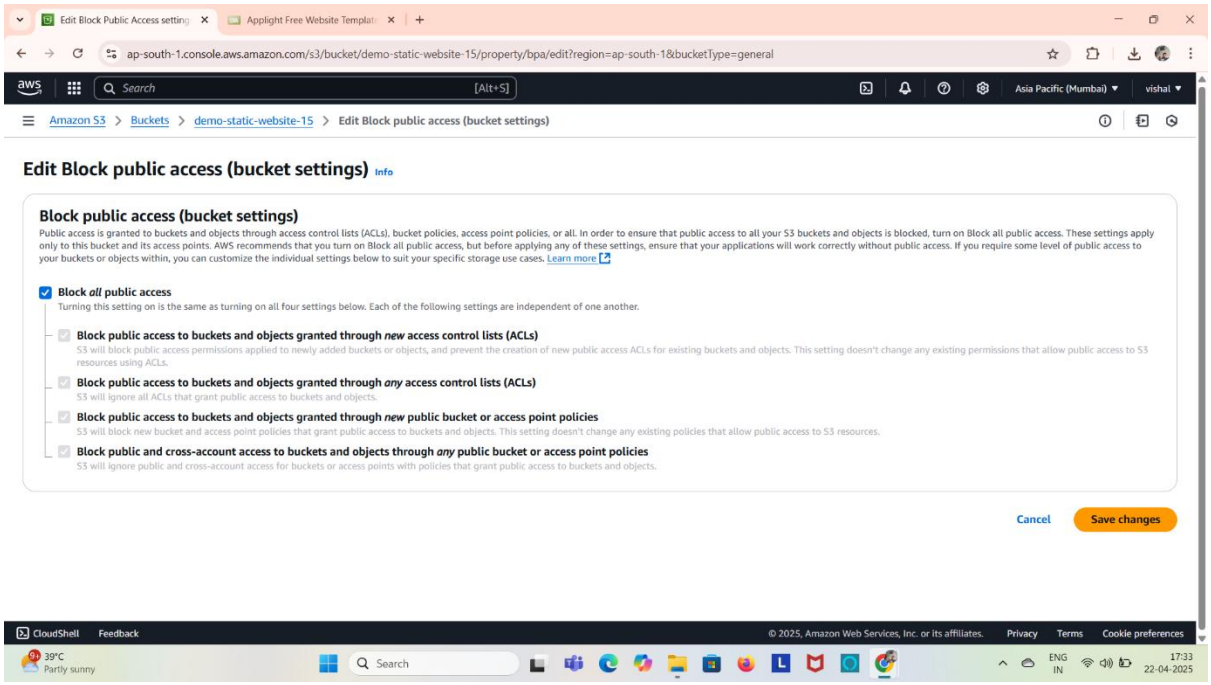


Step 5: Unblock Public Access:

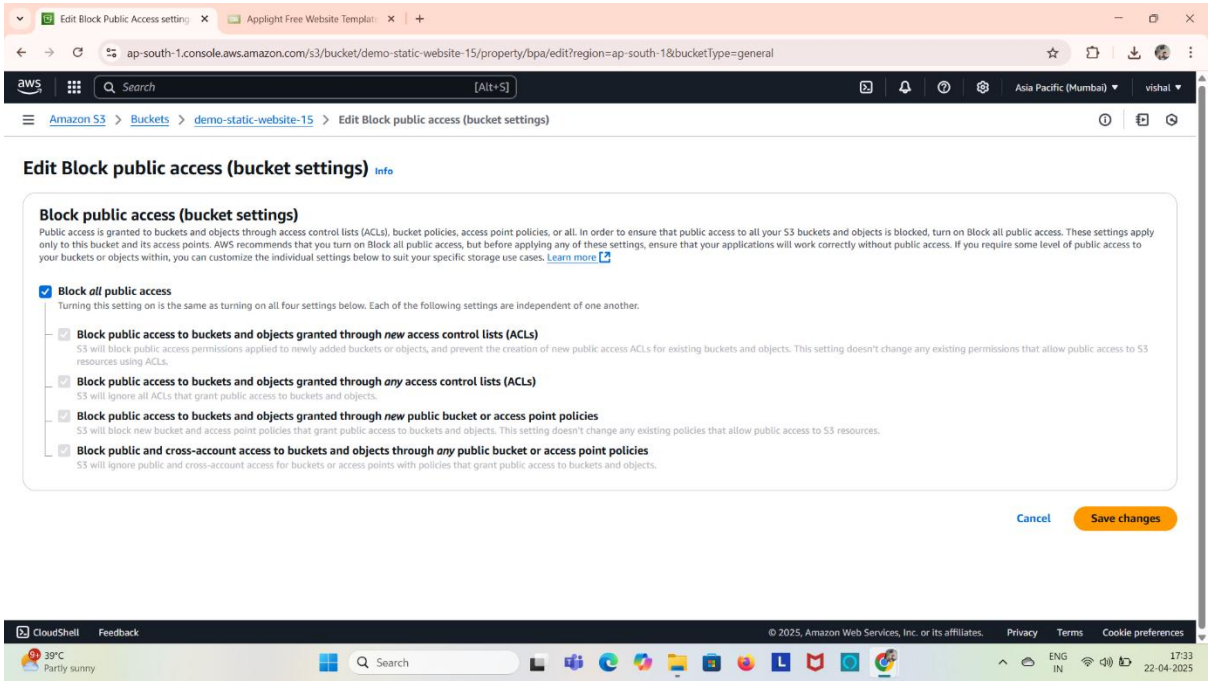
1. Go to the Permissions tab



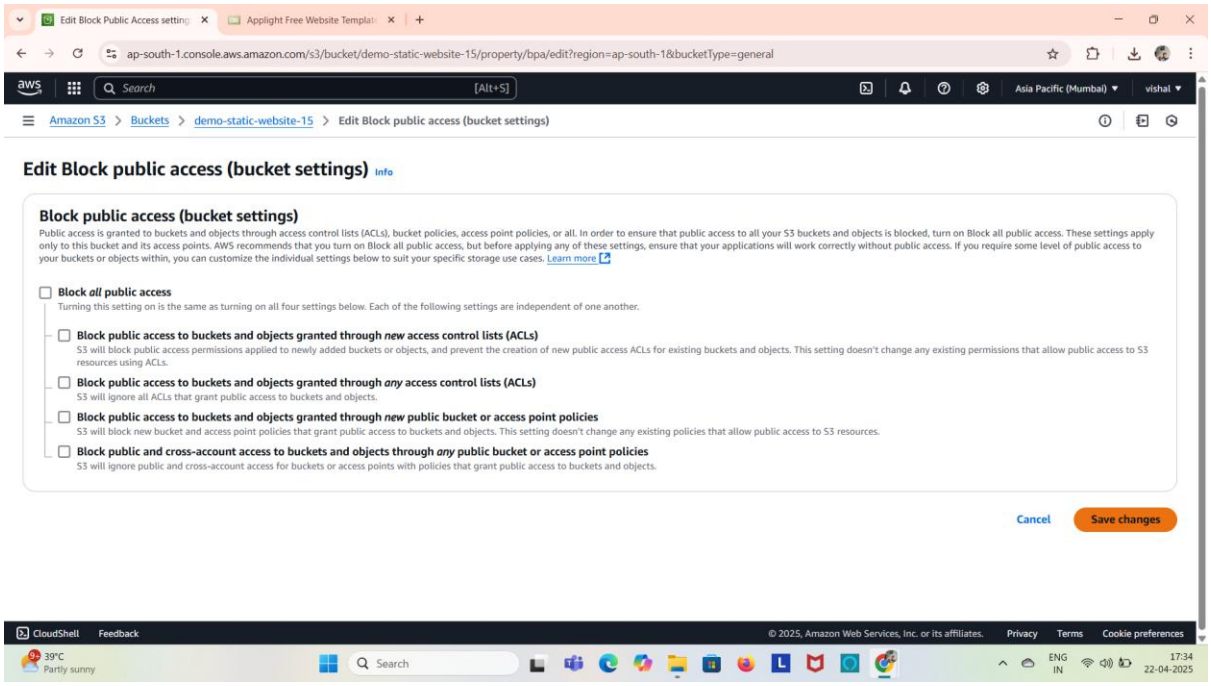
2. Click on Block public access (bucket settings).



3. Click Edit, uncheck all options to disable blocking public access.

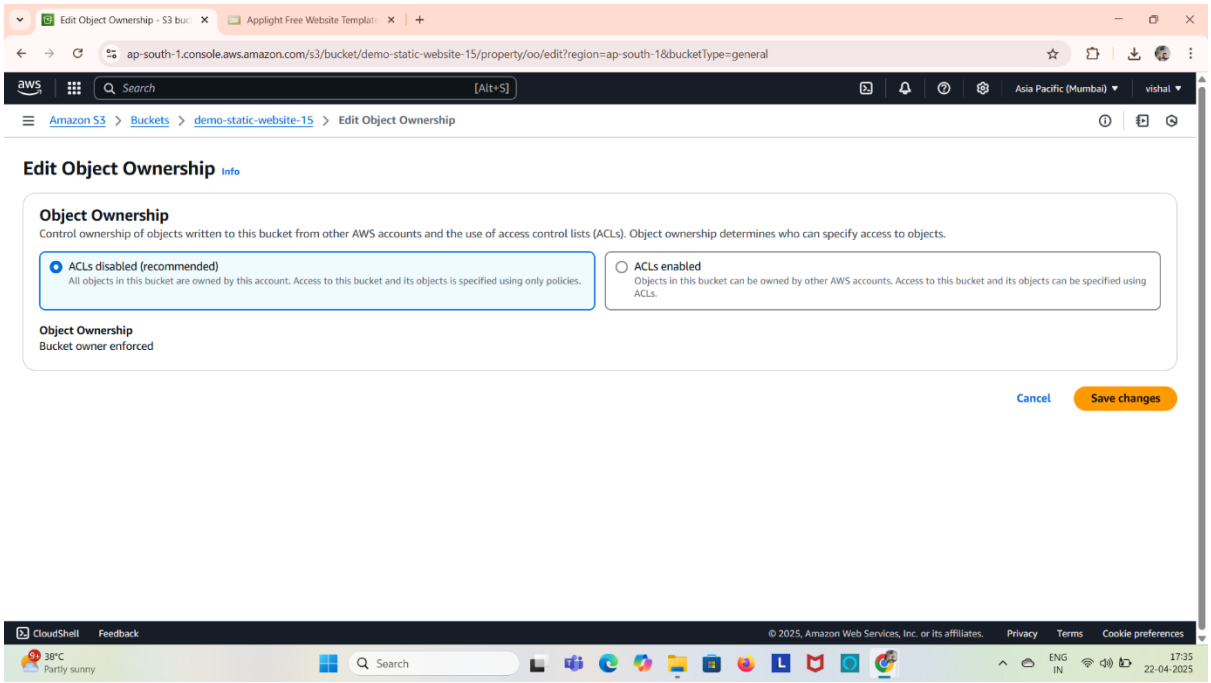


4. Confirm the warning and save.

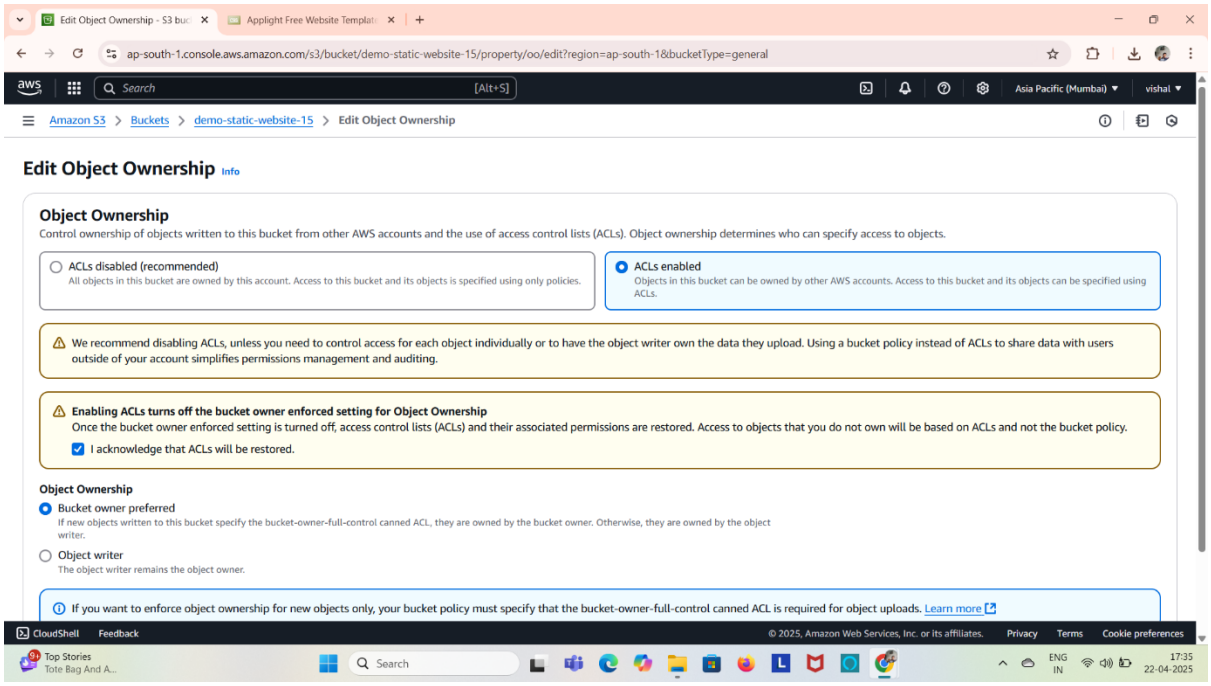


Step 6: Change Object Ownership & Enable Ownership:

1. Still under Permissions, go to Object Ownership.

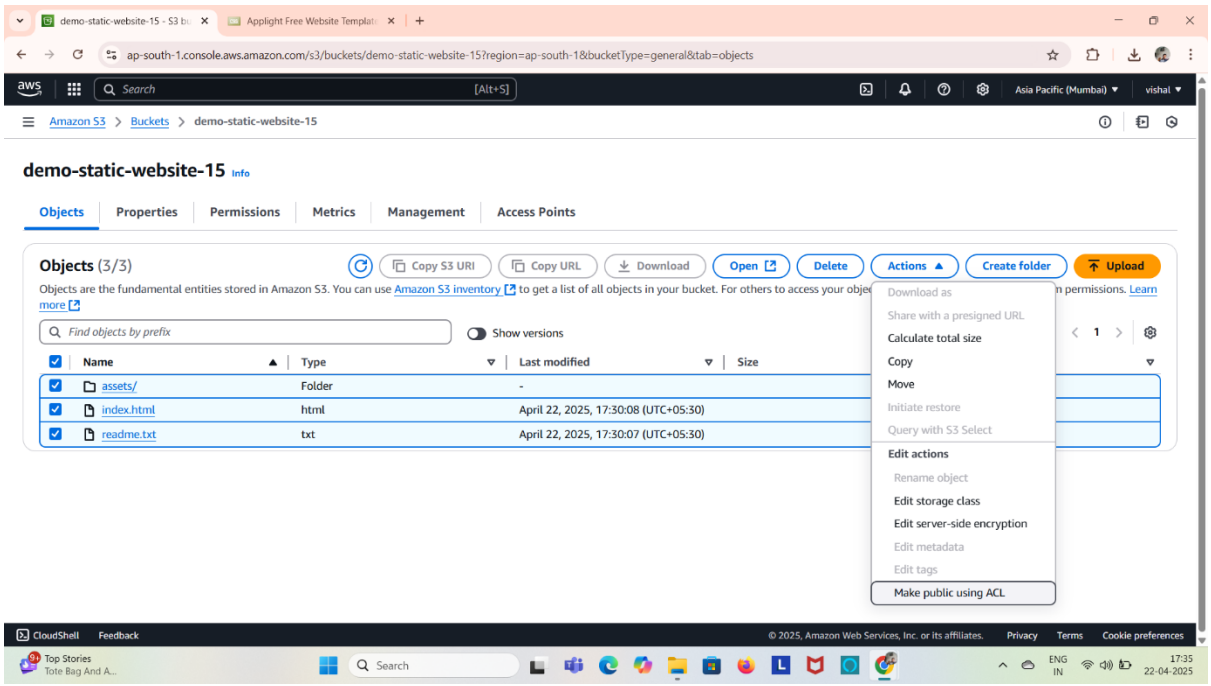


2. Click Edit and select ACLs enabled (bucket owner preferred or object writer preferred).

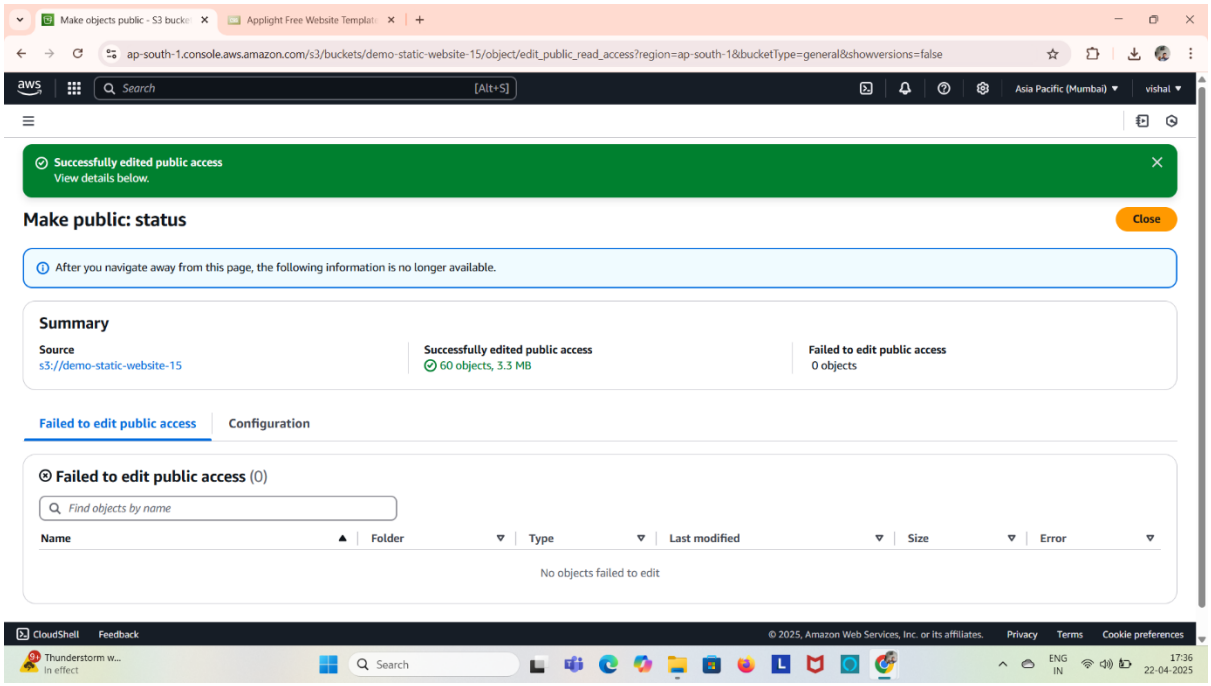


Step 7: Make Uploaded File Public Using ACL:

1. Go to the Objects,Select all files and click Actions → Make public.

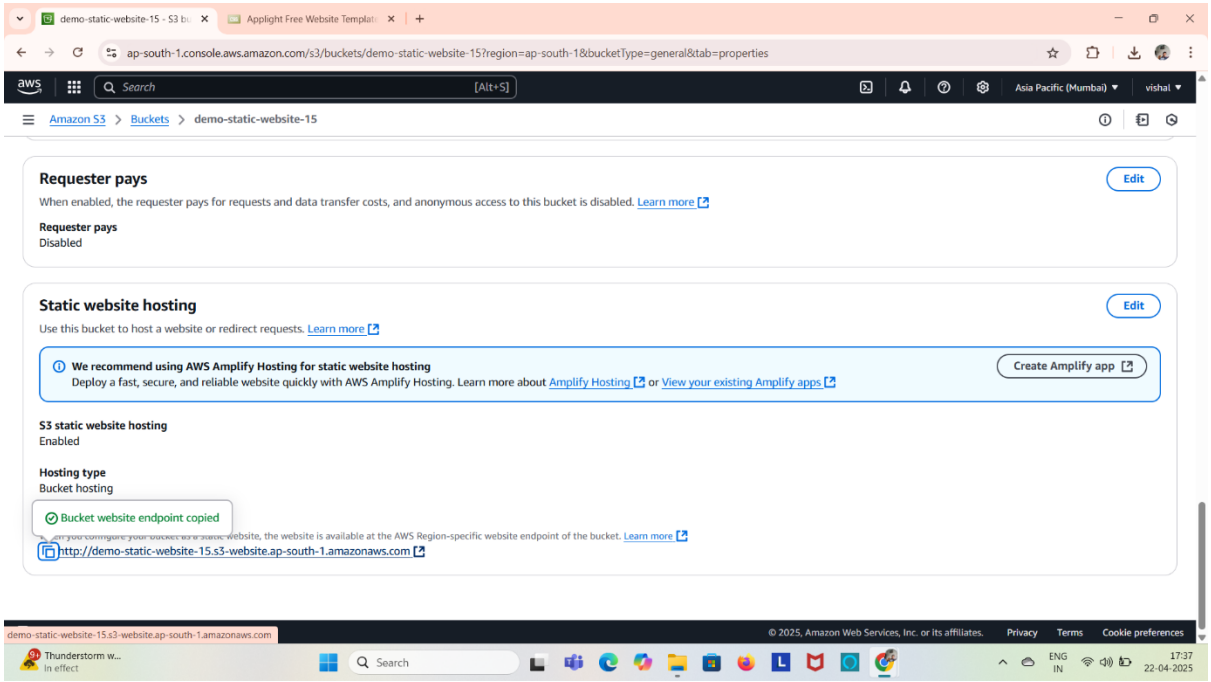


2. Confirm the action.
   Each object will now have a public URL accessible via browser.



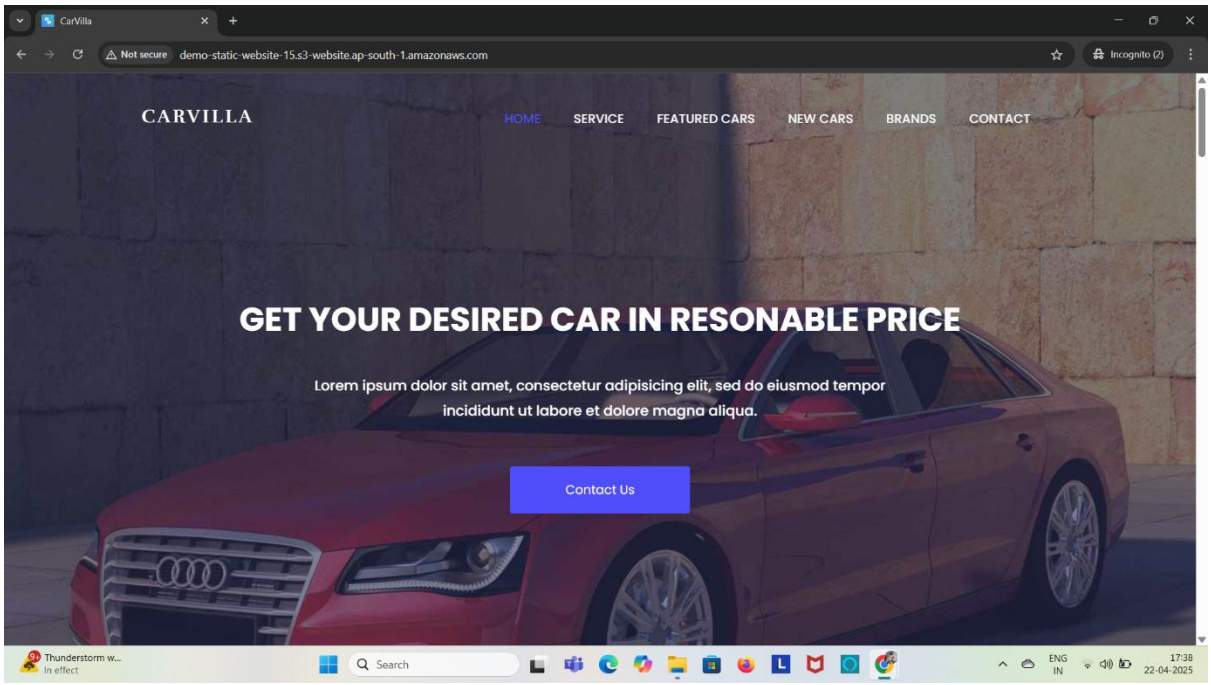
Step 8: Access the Static Website:

After enabling hosting and permissions, your site can be accessed at:
http://demo-static-website-15.s3-website.ap-south-1.amazonaws.com

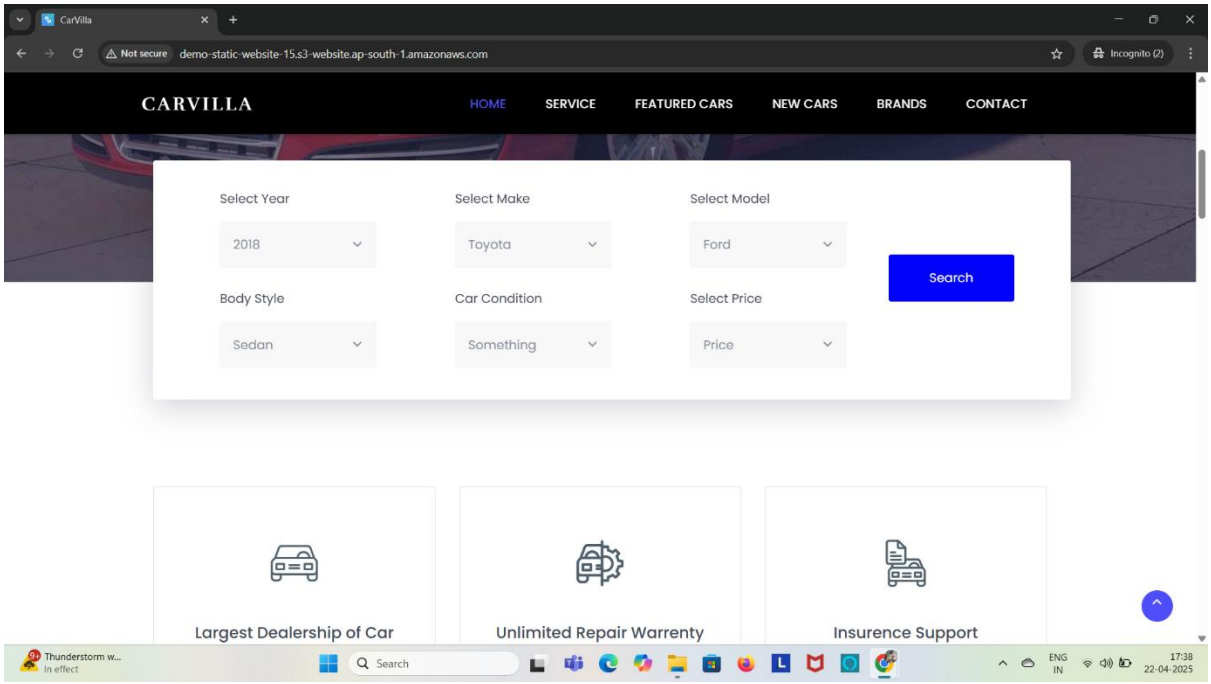


- **Web Pages:-**

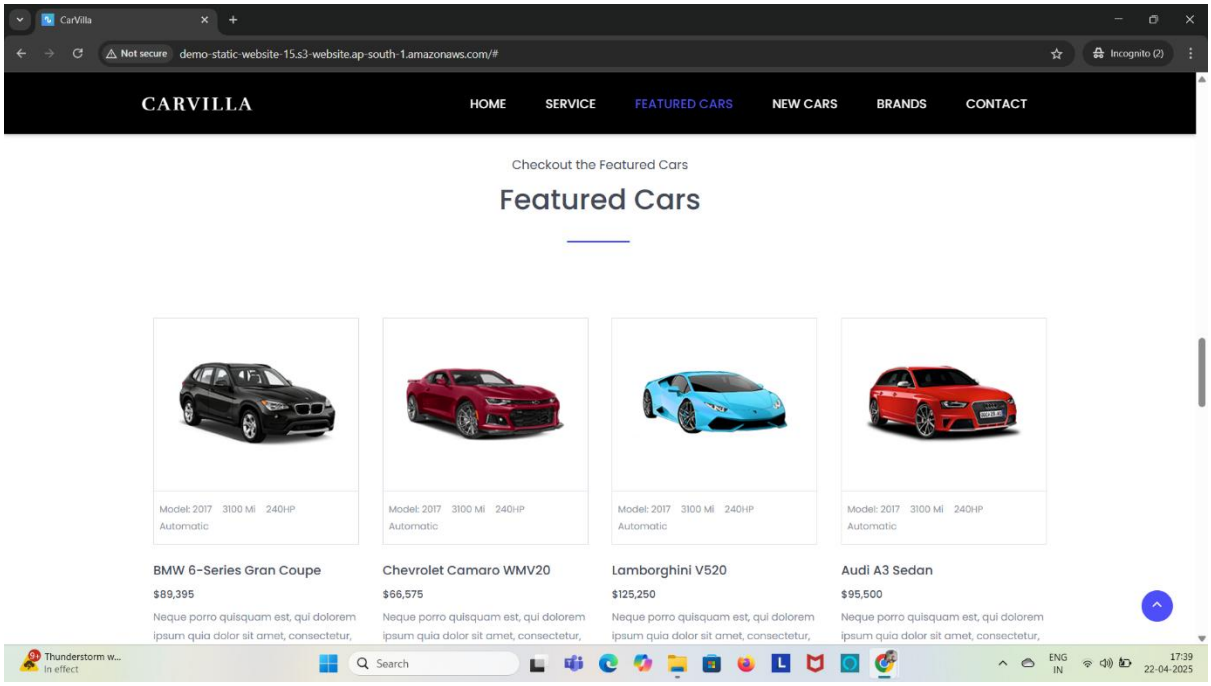
1. Home Page



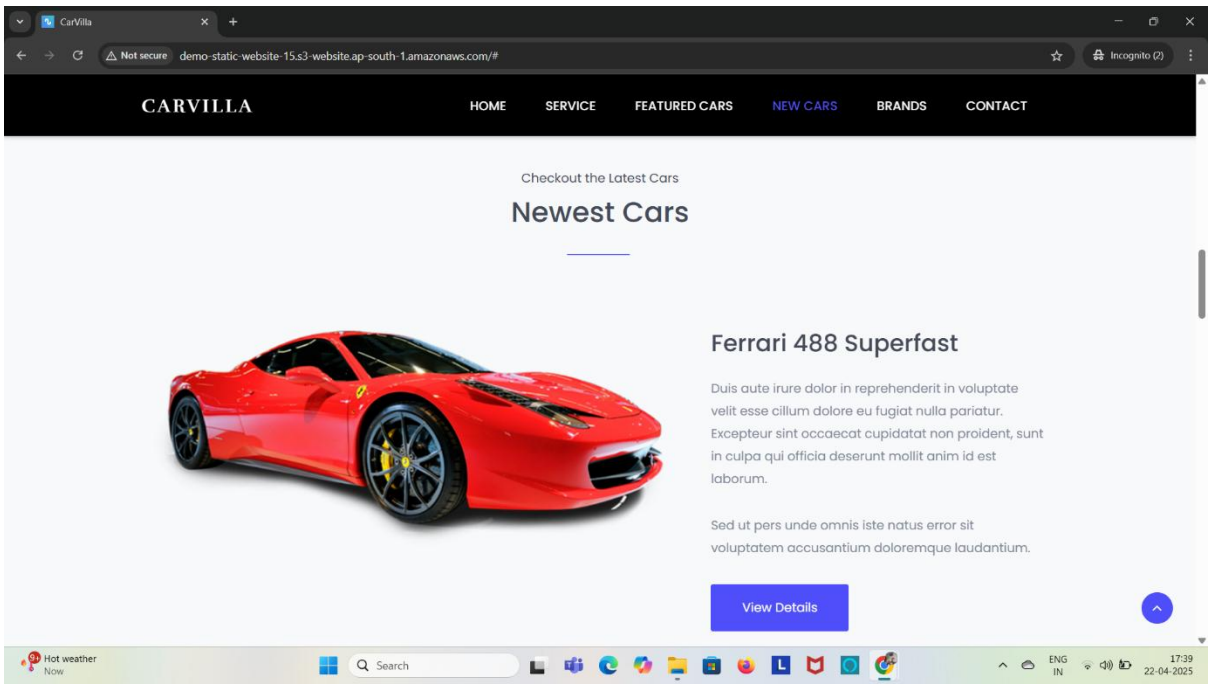
2. Service Page



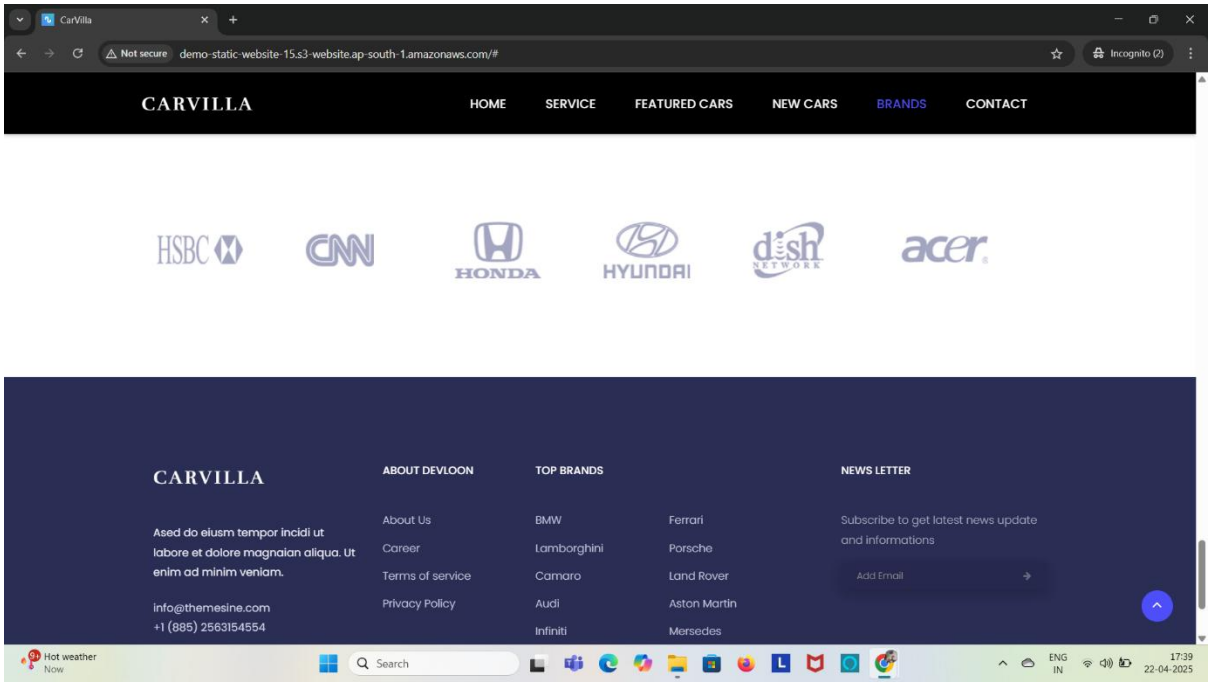
3. Featured Cars



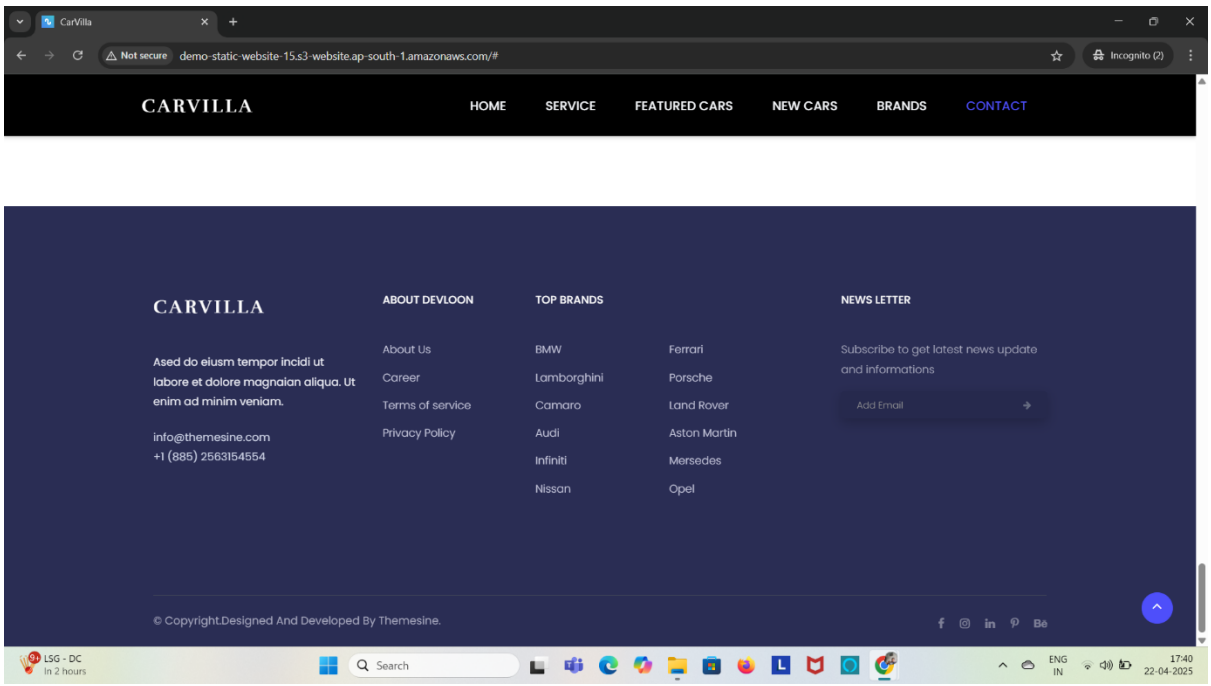
4. New Cars



5. Brands



6. Contact



- **Summary:-**

  This project demonstrates how to deploy a fully functional static Carvilla static website using Amazon S3. It covers the entire lifecycle of static website hosting—from bucket creation and file uploads to permission management and access logging. The process ensures that the frontend website is live, publicly accessible, version-controlled, and monitored.

- **Optical Enhancement:-**

  To enhance the Carvilla Static Website further, several advanced features can be integrated. Adding a custom domain using Amazon Route 53, along with CloudFront for HTTPS and global content delivery, gives the project a professional edge. Converting the site into a Progressive Web App (PWA) with offline support, service workers, and an installable interface improves usability across devices. Incorporating a frontend framework like React.js or Vue.js can add dynamic product interactions, while CI/CD pipelines using GitHub Actions or AWS CodePipeline can automate deployments.