

STATIC WEBSITE HOSTING IN AWS (AMAZON WEB SERVICE) BY USING S3 AND EC2.

NAME – PRANAY ADSUL

BATCH – DEVOPS BATCH 9 (B9)

DATE – 26/02/2025

ABSTRACT

The abstract or executive summary provides a brief and concise overview of the entire project, allowing readers to quickly grasp the essence of what the project is about. This section should include the following:

Introduction to the Project:

This project focuses on demonstrating how to effectively host a static website using AWS services such as Amazon S3 and EC2. Amazon S3 provides an affordable, scalable solution for static website hosting, while EC2 offers more control for custom backends or additional features that may be required. The project outlines the process of setting up both services, optimizing for performance, and securing the website.

Objective of the Project:

The objective of this project was to first demonstrate the ease of hosting a static website on **Amazon S3** and then extend the website's functionality by migrating to **Amazon EC2**. This allowed for the exploration of both static and dynamic hosting solutions on AWS, showcasing the flexibility and scalability of AWS services.

Key Technologies Used:

The key technologies used in this project include:

- **Amazon S3 (Simple Storage Service):** Used initially for hosting static content such as HTML, CSS, and JavaScript files. S3 offers scalability, cost-effectiveness, and simple configuration for static websites.
- **Amazon EC2 (Elastic Compute Cloud):** Used after the initial S3 deployment to host the website with more control, allowing for dynamic content generation or backend processing.

Project Scope and Significance:

The project initially focused on hosting a simple static website using **Amazon S3**, which is ideal for small, lightweight websites that do not require server-side processing. As the project progressed and the need for more dynamic features or backend processing arose, **Amazon EC2** was introduced to provide a full-fledged server environment capable of running web applications, databases, or custom server-side code. This two-phase approach demonstrates the flexibility of AWS, allowing websites to scale from simple static content to more complex dynamic applications as requirements grow.

Summary of the Outcome:

The project successfully demonstrated two different hosting solutions on AWS. In the first phase, the website was hosted using **Amazon S3**, providing a cost-effective, reliable, and scalable solution for serving static content. In the second phase, the website was migrated to **Amazon EC2**, allowing for more control over the server environment and enabling dynamic functionality. The project showcases the benefits of both approaches and how AWS can be leveraged to meet the evolving needs of a website, from simple static hosting to more complex, dynamic hosting solutions.

CONTENT

1. Title Page
2. Abstract/Executive Summary
3. Table of Contents
4. Introduction
 - 4.1. Overview of Static Website Hosting
 - 4.2. Project Goal
5. Technologies Used
 - 5.1. Amazon S3
 - 5.2. Amazon EC2
6. Architecture Diagram
7. Step-by-Step Implementation
 - 7.1. Hosting Static Website on S3
 - 7.2. Setting Up EC2 (Linux)
8. Testing and Validation
 - 8.1. Testing Static Website on S3
 - 8.2. Testing EC2 Hosting
9. Challenges and Solutions
10. Results and Benefits
 - 10.1. Benefits of Using AWS S3
 - 10.2. Why Use EC2
11. Conclusion

INTRODUCTION

Overview of Static Website Hosting:

Static website hosting refers to serving content that does not change in real-time. The files (HTML, CSS, JavaScript) are stored and served as they are without needing server-side processing. AWS offers two powerful services for hosting websites: S3 for static files and EC2 for more customizable server-based needs.

Difference Between Static and Dynamic Websites:

- **Static Websites:** Content is fixed and does not change unless manually updated.
- **Dynamic Websites:** Content is generated on-demand using server-side technologies (e.g., PHP, Node.js).

Why AWS S3 and EC2?

- **Amazon S3:** Ideal for static website hosting, as it is highly scalable, cost-effective, and easy to set up.
- **Amazon EC2:** Offers full control over the server and is used here for potential dynamic content handling or custom backend needs, such as running a Linux web server (e.g., Apache or Nginx).

Project Goal:

The goal of this project is to set up a static website using S3 and optionally use EC2 for additional server-side functionality if required. This setup will help demonstrate the flexibility and scalability of AWS services.

TECHNOLOGIES USED

Amazon S3 (Simple Storage Service)

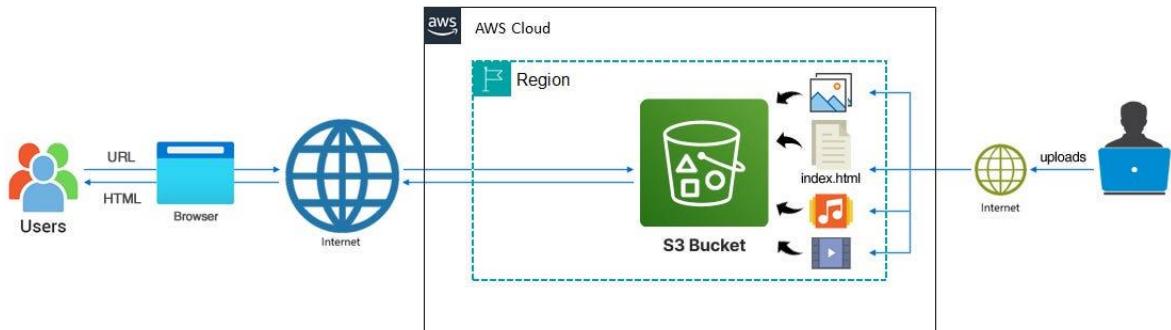
Amazon S3 provides scalable object storage for hosting static content. For this project, S3 is used to host HTML, CSS, and JavaScript files in a cost-effective and scalable manner.

Amazon EC2 (Elastic Compute Cloud)

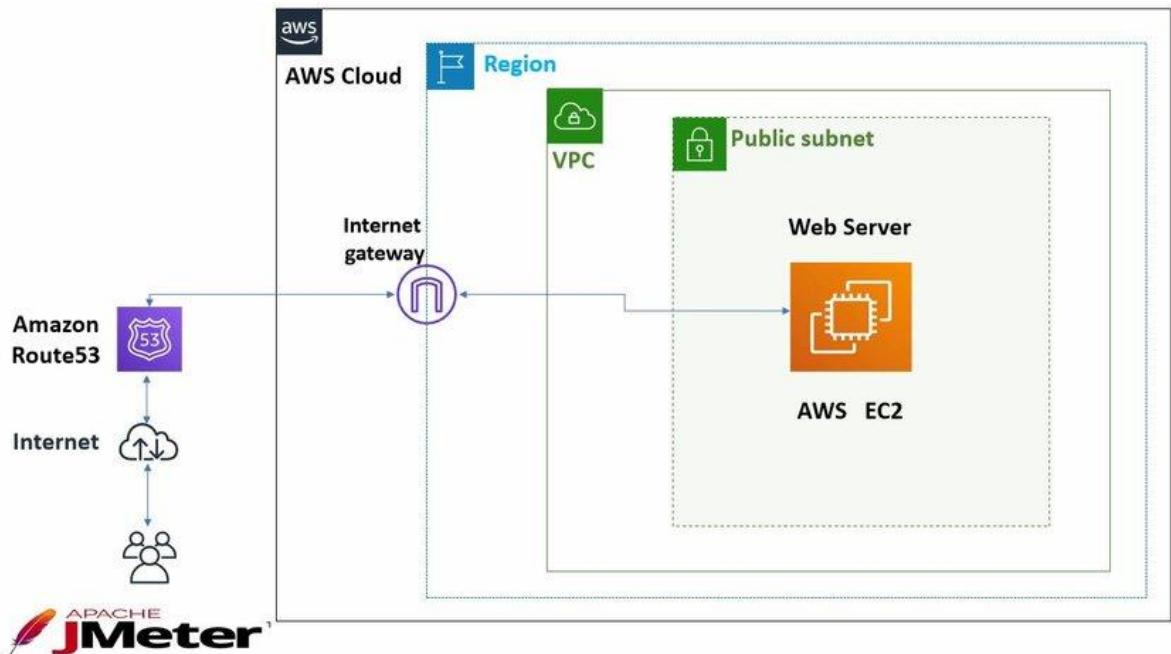
EC2 allows you to provision and manage Linux-based virtual machines (VMs). In this project, an EC2 Linux instance is used for additional flexibility, such as running a web server to handle requests that go beyond the basic static content, or for logging and monitoring purposes.

ARCHITECTURE DIAGRAM

S3 bucket hosting the static website (HTML, CSS, JavaScript):



EC2 Linux instance running a web server (httpd) if required for additional functionality:



APACHE JMeter¹

STEP-BY-STEP IMPLEMENTATION

Hosting Static Website on S3

- 1. Create an S3 Bucket:**
 - Log in to AWS and create an S3 bucket with a globally unique name.
- 2. Configure Bucket for Website Hosting:**
 - Enable static website hosting in the "Properties" section of the bucket.
 - Set the index and error pages (typically index.html and error.html).
- 3. Upload Website Files:**
 - Upload all static files (HTML, CSS, JS) to the S3 bucket.
- 4. Set Permissions:**
 - Modify the bucket policy to allow public access to the files, ensuring users can view the website.
 - Example of a basic public read bucket policy for S3.

Setting Up EC2 with Apache HTTP Server (httpd)

- 1. Launch an EC2 Instance:**
 - Choose an appropriate Linux AMI (e.g., Ubuntu, Amazon Linux).
 - Select instance type (e.g., t2.micro for small-scale use).
- 2. Configure Security Groups:**
 - Allow HTTP (port 80) and SSH (port 22) access for management.
- 3. Install HTTP Server:**
 - First install httpd in instance. By using command – `yum install httpd -y`.
 - Start the httpd. `Systemctl start httpd`.
 - Enable the http. `Systemctl enable httpd`.
- 4. Configure website:**
 - By default, http will serve files from /var/www/html. You can copy the static website files (if using EC2 for dynamic content) or serve them from another directory depending on your needs.

Website hosting in S3:

Step 1: Create the Bucket

The screenshot shows the AWS S3 console with the following details:

- Region:** Asia Pacific (Mumbai)
- Buckets:** One bucket named "pranay-adsl" was successfully created.
- Account snapshot:** Updated every 24 hours.
- Storage lens:** Provides visibility into storage usage and activity trends.
- General purpose buckets:** One bucket named "pranay-adsl" is listed.
- Details:** Bucket ARN: arn:aws:s3:::pranay-adsl, IAM Access Analyzer: View analyzer for ap-south-1, Creation date: February 12, 2025, 18:05:42 (UTC+05:30).

Step 2: Enable the Static Website Hosting:

The screenshot shows the AWS S3 console with the following details:

- Region:** Asia Pacific (Mumbai)
- Buckets:** One bucket named "adsul-612001" has static website hosting enabled.
- Requester pays:** Disabled.
- Static website hosting:** Enabled.
- Hosting type:** Bucket hosting.
- Bucket website endpoint:** http://adsul-612001.s3-website.ap-south-1.amazonaws.com

Step 3: Edit Public Access:

The screenshot shows the 'Edit Block public access (bucket settings)' page for the bucket 'adsul-612001'. At the top, there's a navigation bar with the AWS logo, search bar, and account information ('Asia Pacific (Mumbai) Adsu_Prany'). Below the navigation is a breadcrumb trail: 'Amazon S3 > Buckets > adsul-612001 > Edit Block public access (bucket settings)'. The main content area has a title 'Edit Block public access (bucket settings)' with an 'Info' link. A note below the title states: 'Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases.' It includes a 'Learn more' link. There are five checkboxes under the heading 'Block public access (bucket settings)':

- Block all public access**: Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
- Block public access to buckets and objects granted through new access control lists (ACLs)**: S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**: S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**: S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**: S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

At the bottom right are 'Cancel' and 'Save changes' buttons.

Step 4: Enable the ACL:

The screenshot shows the 'Edit Object Ownership' page for the bucket 'adsul-612001'. At the top, there's a navigation bar with the AWS logo, search bar, and account information ('Asia Pacific (Mumbai) Adsu_Prany'). Below the navigation is a breadcrumb trail: 'Amazon S3 > Buckets > adsul-612001 > Edit Object Ownership'. The main content area has a title 'Object Ownership' with a note: 'Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.' There are two radio buttons:

- ACLs disabled (recommended)**: All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ACLs enabled**: Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Below the radio buttons are two callout boxes:

- We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.**
- Enabling ACLs turns off the bucket owner enforced setting for Object Ownership**: Once the bucket owner enforced setting is turned off, access control lists (ACLs) and their associated permissions are restored. Access to objects that you do not own will be based on ACLs and not the bucket policy.
 I acknowledge that ACLs will be restored.

At the bottom right are 'Cancel' and 'Save changes' buttons.

Step 5: Add the website the bucket and make public using ACL:

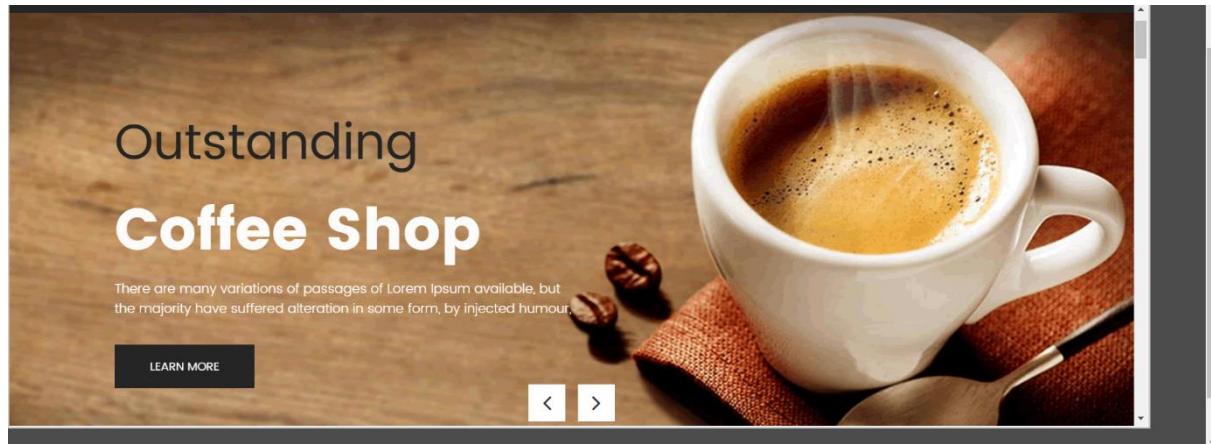
The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with 'Search' and 'Alt+S'. Below it, the path 'Amazon S3 > Buckets > adsul-612001'. The main area is titled 'adsul-612001' with a 'Info' link. Below that is a table titled 'Objects (10/10)'. The table lists the following files:

Name	Type	Last modified	Size
about.html	html	February 14, 2025, 14:43:22 (UTC+05:30)	
cake-shop-website-template.jpg	jpg	February 14, 2025, 14:43:23 (UTC+05:30)	
contact.html	html	February 14, 2025, 14:43:23 (UTC+05:30)	
index.html	html	February 14, 2025, 14:43:20 (UTC+05:30)	
LICENSE.txt	txt	February 14, 2025, 14:43:21 (UTC+05:30)	
menu.html	html	February 14, 2025, 14:43:21 (UTC+05:30)	

A context menu is open over the selected objects, with 'Actions' expanded. The visible options include:

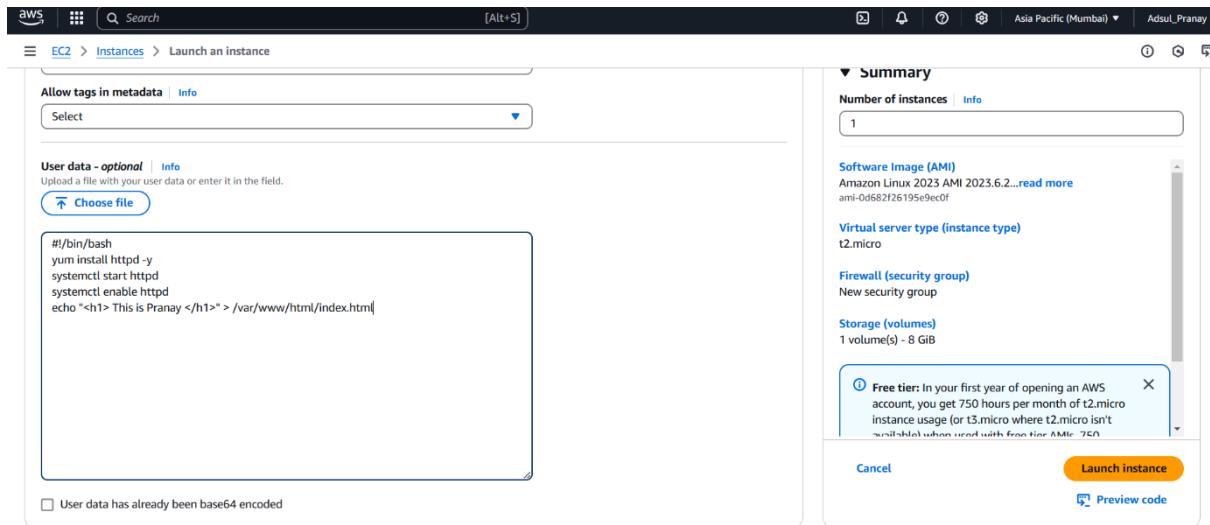
- Download as
- Share with a presigned URL
- Calculate total size
- Copy
- Move
- Initiate restore
- Query with S3 Select
- Edit actions
- Rename object
- Edit storage class
- Edit server-side encryption
- Edit metadata
- Edit tags
- Make public using ACL

Step 6: Copy URL and see the result:

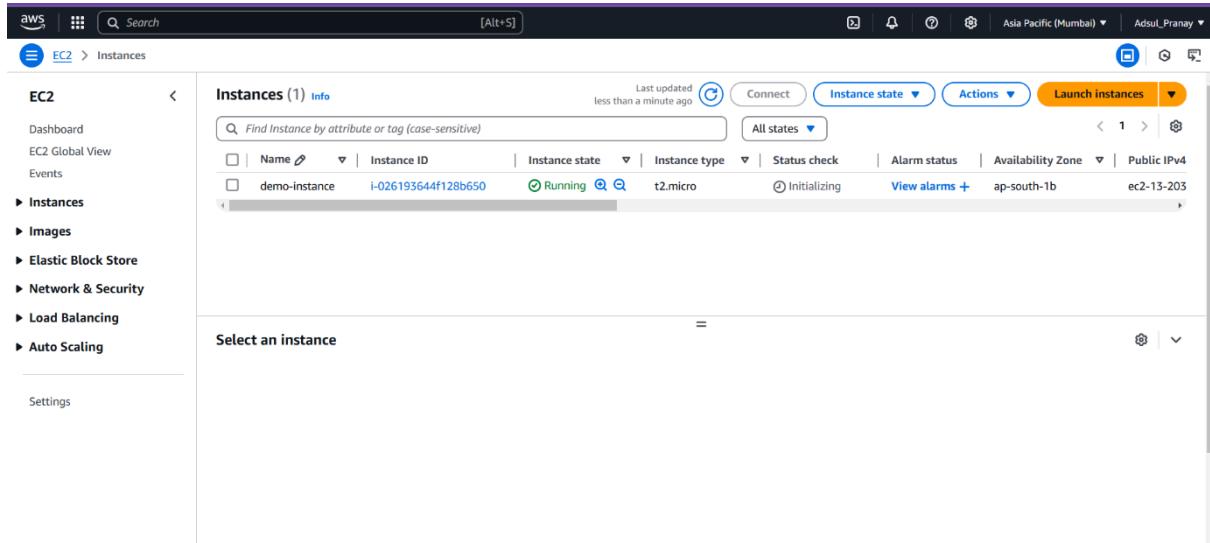


Website Hosting In EC2:

Step 1 : Create the instance and add http server and script in it:



Step 2: Launch Instance:

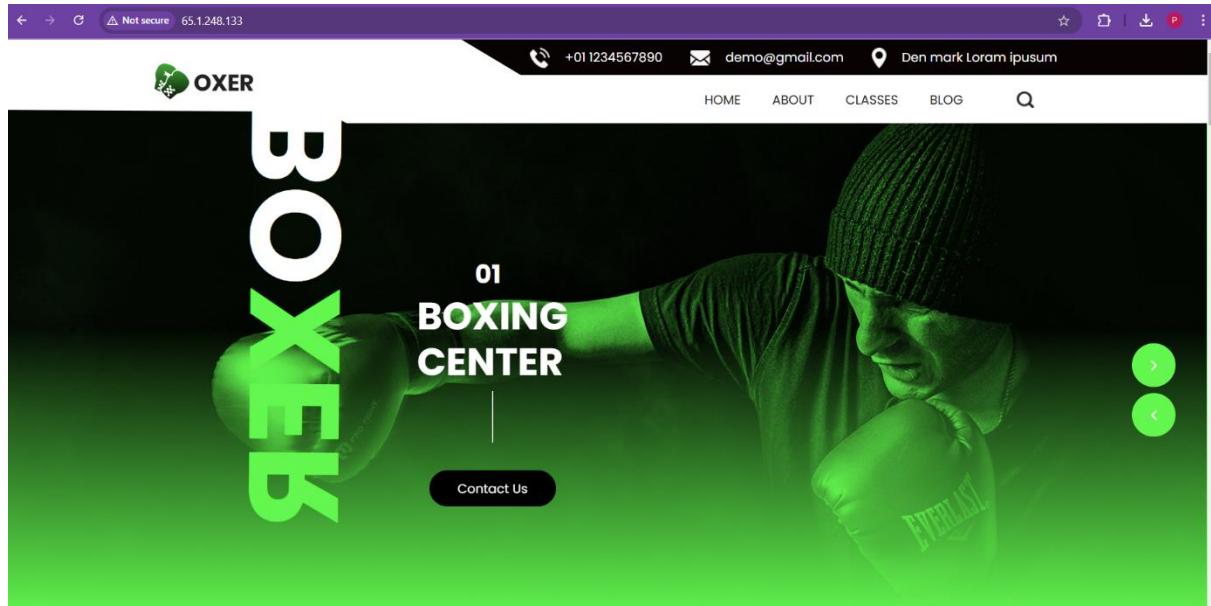


Step 2: Connect the Instance and add website in it:

```
inflating: oxer-html/images/prev-arrow.png
inflating: oxer-html/images/prev-grey.png
inflating: oxer-html/images/prev.png
inflating: oxer-html/images/quote-angle
inflating: oxer-html/images/right-angle.png
inflating: oxer-html/images/right-black-arrow.png
inflating: oxer-html/images/search-icon.png
inflating: oxer-html/images/telephone.png
inflating: oxer-html/images/twitter.png
inflating: oxer-html/images/youtube.png
inflating: oxer-html/index.html
creating: oxer-html/js/
inflating: oxer-html/js/bootstrap.js
inflating: oxer-html/js/jquery-3.4.1.min.js
[root@ip-172-31-2-90 ~]# cd oxer-html/
[root@ip-172-31-2-90 oxer-html]# cd
[root@ip-172-31-2-90 ~]# ls
oxer-html  oxer.zip
[root@ip-172-31-2-90 ~]# cd oxer-html/
[root@ip-172-31-2-90 oxer-html]# ls
about.html  blog.html  class.html  css  images  index.html  js
[root@ip-172-31-2-90 oxer-html]# cd
[root@ip-172-31-2-90 ~]# mv oxer-html/* /var/www/html
[root@ip-172-31-2-90 ~]# cd /var/www/html
[root@ip-172-31-2-90 html]# ls
about.html  blog.html  class.html  css  images  index.html  js
[root@ip-172-31-2-90 html]# cd
[root@ip-172-31-2-90 ~]#
```

i-0d1c6390277b6de60 (instance-pran)
Public IPs: 65.1.248.133 Private IPs: 172.31.2.90

Step 3: Copy Public IP and see the result:



TESTING AND VALIDATION

Testing Static Website on S3

- Access the website via the S3 bucket URL.
- Check if HTML, CSS, and JS files are loading correctly.
- Verify the website's responsiveness and static content.

Testing EC2 Hosting

- Access the EC2 instance via its public IP or domain name.
- Verify that the Apache web server is serving content correctly from the instance.
- Test any dynamic functionality if EC2 is used for such purposes.

CHALLENGES AND SOLUTIONS

Challenges

- Permissions Issues: Incorrect S3 bucket policies preventing public access to the website.
- EC2 Security Group Configuration: Initial firewall settings preventing HTTP access.
- Apache Configuration: Issues with starting Apache or configuring the web server to serve content correctly.

Solutions

- Adjusted S3 bucket policy to ensure proper access permissions.
- Configured EC2 security groups to allow traffic on port 80 (HTTP).
- Ensured Apache was correctly installed and started, and website files were placed in the right directory (/var/www/html).

RESULTS AND BENEFITS

Benefits of Using AWS S3

- **Cost-effective:** No need for server management. Only pay for storage and bandwidth.
- **Scalable:** Automatically scales to handle large amounts of traffic.
- **No Server Management:** No need to manage traditional web servers for static content.

Why Use EC2 with httpd:

EC2 with Apache provides more flexibility for handling dynamic server-side requirements, custom backend services, or running additional applications that require processing

CONCLUSION

The project successfully demonstrated hosting a static website using Amazon S3, with an optional EC2 Linux instance running HTTP Server (httpd) for additional functionality. This setup is highly scalable, cost-effective, and easy to manage. AWS provides an excellent environment for deploying websites with minimal overhead and high flexibility.