Cloud & DevOps Integration Project

Deploying a Static Website using S3

NAME: S. JAPESH

REG NO: 23BCE8857

REPORT

Introduction and Cloud Service Models

Introduction

In today's fast-paced digital world, cloud computing and DevOps have become essential for organizations to deliver products and services efficiently. This project demonstrates the deployment of a static website using Amazon Web Services (AWS) EC2 and S3 services, combined with basic DevOps practices like automation, version control, and continuous integration.
The aim is to understand cloud service models and how DevOps integrates seamlessly with cloud platforms to create scalable, reliable, and efficient systems.

Difference between IaaS, PaaS, and SaaS

Model	Description	Example
IaaS	Infrastructure as a Service provides virtualized computing resources over the internet. Users manage the operating systems, storage, and deployed applications.	**AWS EC2**: A virtual server that can be customized based on the need (CPU, RAM, OS, etc.).
PaaS	Platform as a Service offers hardware and software tools over the internet, typically for application development. No need to manage infrastructure.	**Google App Engine**: Developers can build and deploy apps without worrying about servers.
SaaS	Software as a Service delivers software applications over the internet on a subscription basis. No installation or maintenance needed.	**Google Docs, Gmail**: Accessed through a web browser without any backend management.

Summary:
IaaS gives you raw resources (like machines), PaaS gives you an environment (ready to code), and SaaS gives you a complete software application ready to use.

DevOps and Cloud Platform Integration

How DevOps Aligns with Cloud Platforms (AWS, Azure, GCP)

DevOps and Cloud are deeply interconnected.
Cloud platforms like AWS, Azure, and GCP provide the flexibility and tools needed to implement core DevOps practices effectively.

Key Points:

- **Automation:** Cloud platforms allow automatic provisioning of servers, databases, and networks (Infrastructure as Code) using tools like AWS CloudFormation or Terraform.
- **Continuous Integration/Continuous Delivery (CI/CD):** Cloud services integrate easily with CI
- **Scalability:** Applications can scale automatically based on traffic using services like AWS Auto Scaling Groups or Azure Scale Sets.
- **Monitoring and Logging:** Tools like AWS CloudWatch, Azure Monitor, and GCP Stackdriver help in real-time monitoring, alerting, and log analysis.
- **Cost Optimization:** Pay-as-you-go models in the cloud support dynamic scaling, reducing infrastructure costs.
- **Global Deployment:** Cloud providers offer multi-region support, making it easy to deploy applications across the globe with minimal latency.

Conclusion:
By combining DevOps with cloud platforms, businesses can achieve faster releases, more reliable deployments, and significant operational efficiency.

Hands-on Activity – Static Website Deployment

Step-by-Step Activity

1. **AWS Free Tier Account Creation:**
    - Created a free AWS account with root access.
    - Set up security credentials (access key ID and secret).
2. **Launching EC2 Instance:**
    - Launched a Linux-based (Ubuntu) EC2 instance.
    - Selected free-tier eligible t2.micro instance.
    - Created a security group to allow HTTP (port 80) and SSH (port 22) access.
3. **Uploading HTML Page:**
    - Connected to the EC2 instance via SSH using .pem key.
    - Installed Apache web server:

bash

CopyEdit

sudo apt update

sudo apt install apache2 -y

- Uploaded index.html and style.css to /var/www/html/.
- Verified the web server by accessing the public IP.

4. **Creating S3 Bucket:**
   - Created an S3 bucket with public access enabled.
   - Uploaded static files (images, CSS files) that were referenced in the HTML page.
   - Configured bucket policy to allow public read access to assets.
5. **VPC and Subnet Settings:**
   - Ensured EC2 instance was launched in a public subnet with internet gateway attached.
   - Confirmed correct route tables to allow outgoing internet access.
6. **Connecting EC2 and S3:**
   - HTML files hosted on EC2 referenced static assets hosted on S3 (e.g., background images, logos).
   - Tested cross-access and corrected CORS policies if required.
7. **Website Deployment Complete:**
   - Accessed the static website using the EC2 instance public IP.
   - Example: http://13.126.176.93/

Screenshots & Conclusion

Screenshots Captured:

- AWS Management Console showing EC2 instance running.
- SSH session showing Apache installation.
- File upload to /var/www/html/ directory.
- S3 bucket setup with static files.
- Final website opened through public IP.

Example Screenshots:

- EC2 Dashboard → Running instance
- SSH terminal → Apache installed
- S3 Dashboard → Bucket objects
- Website preview → in browser

Conclusion

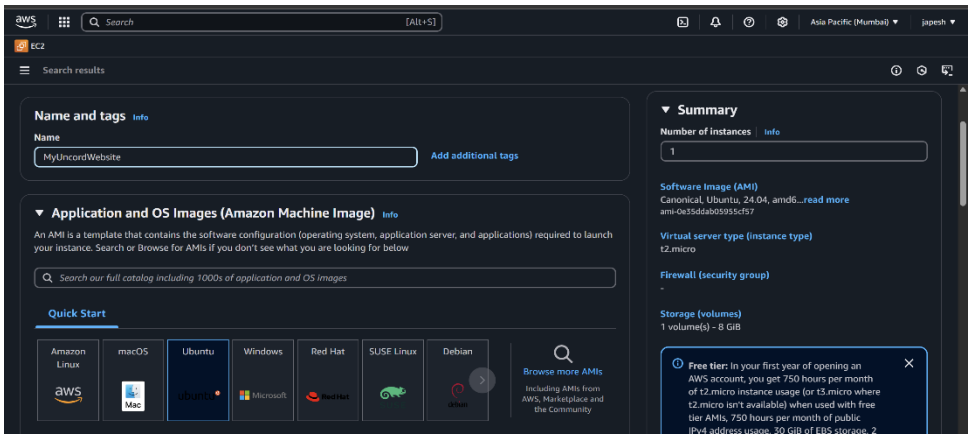
Through this project, a static website was successfully deployed using AWS EC2 and S3 services following DevOps principles.
The exercise showcased the simplicity, scalability, and effectiveness of cloud-native approaches combined with DevOps automation practices.
Understanding IaaS, PaaS, and SaaS helps build better cloud architectures, while leveraging DevOps enables faster, high-quality, and reliable deployments — essential for modern software delivery.

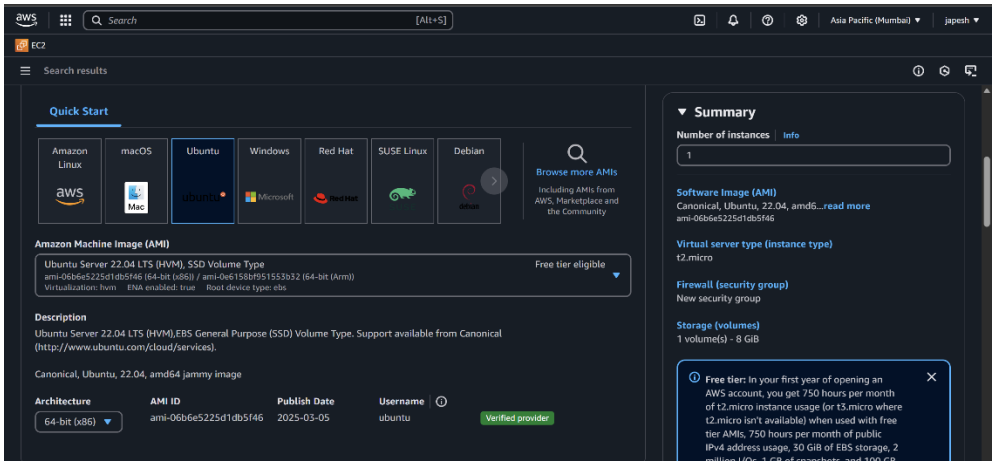
Step-by-Step Instructions with Screenshot Suggestions

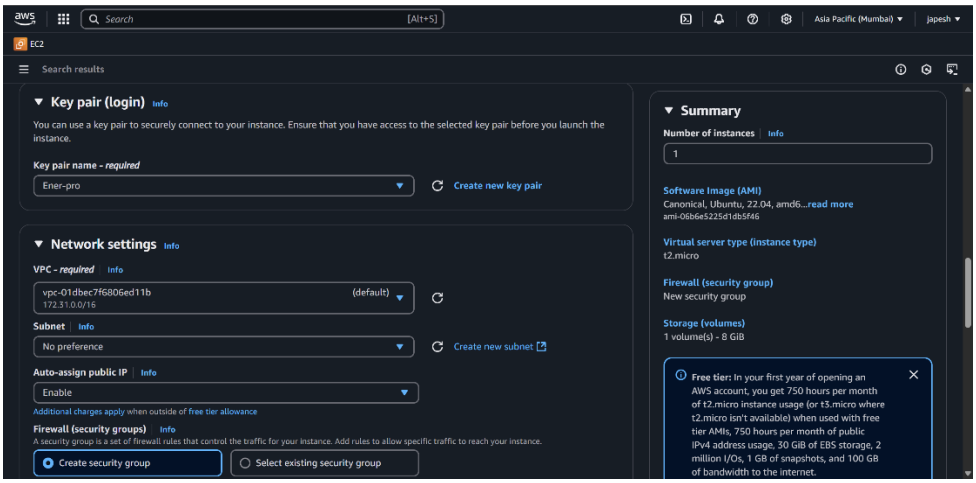
1. **Create AWS Free Tier Account**
    - Go to https://aws.amazon.com/free
    - Sign up and log into the AWS Management Console.
2. **Launch EC2 Instance (Ubuntu)**
    - Navigate to **EC2 > Launch Instance**
    - Choose **Ubuntu AMI(22.04 LTS)**
    - Select **t2.micro** (Free Tier eligible)
    - Configure instance details, add a key pair, and launch.

Public IPV4 adress

13.126.176.93

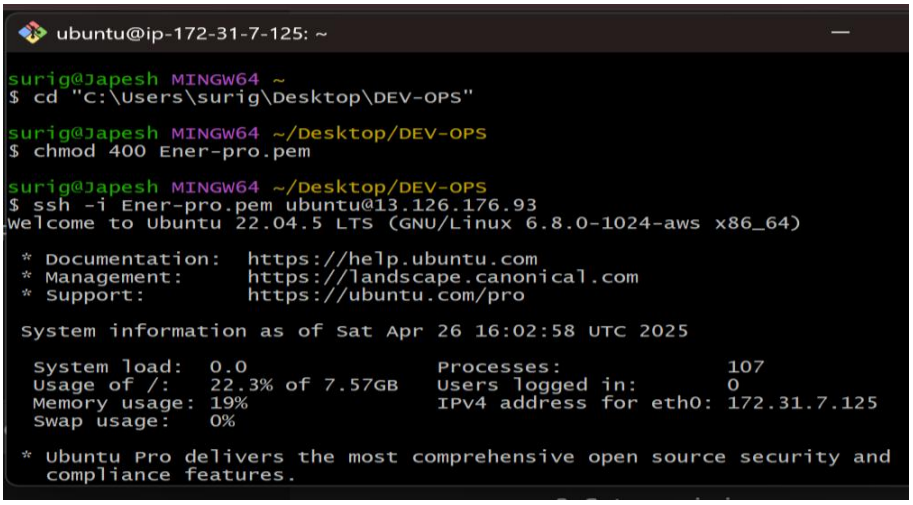






3. Connect to EC2 Instance

Uses Git Bash to SSH into the instance:



ssh -i your-key.pem ubuntu@your-ec2-public-ip

- Install Apache:

bash

sudo apt update

sudo apt install apache2 -y

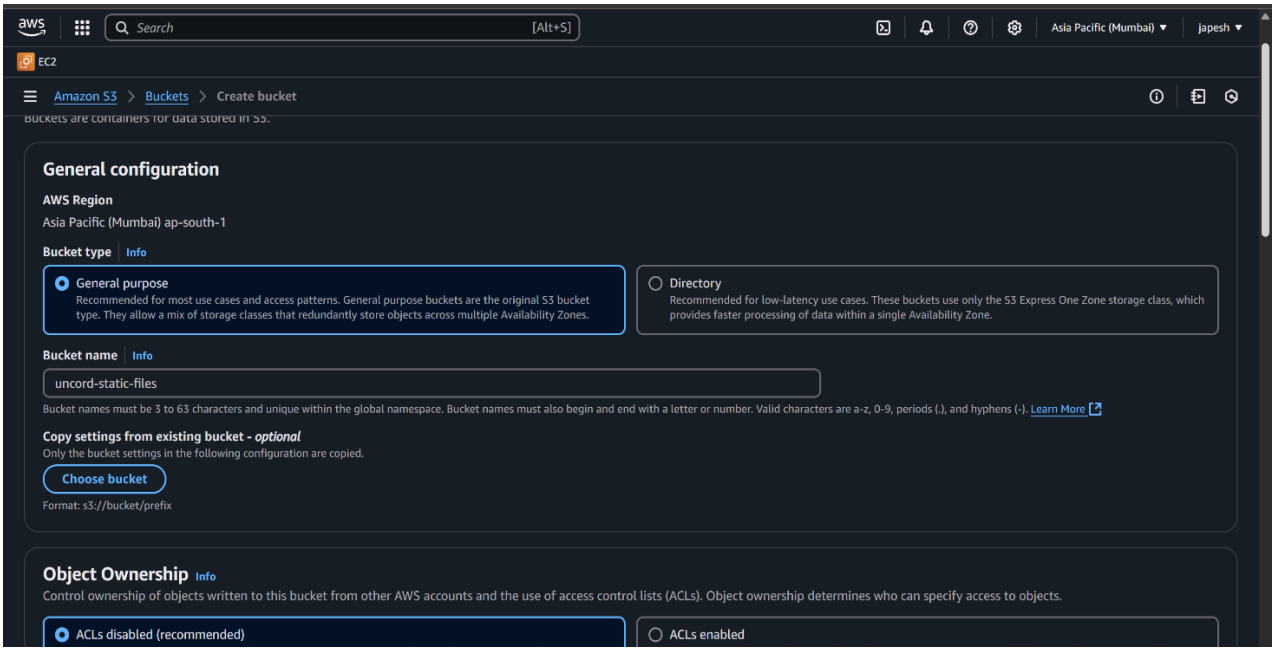
- Upload your HTML file to /var/www/html/index.html

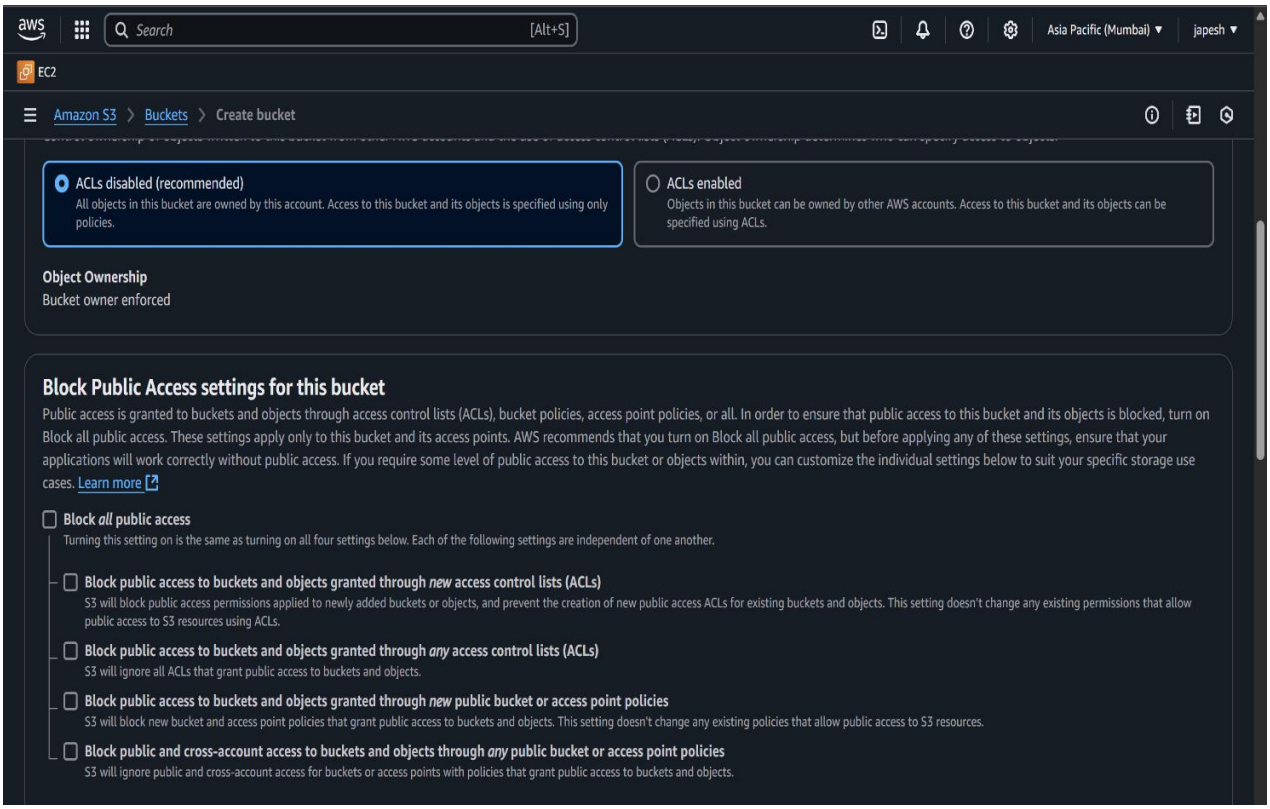
Step 4: Install Apache Web Server

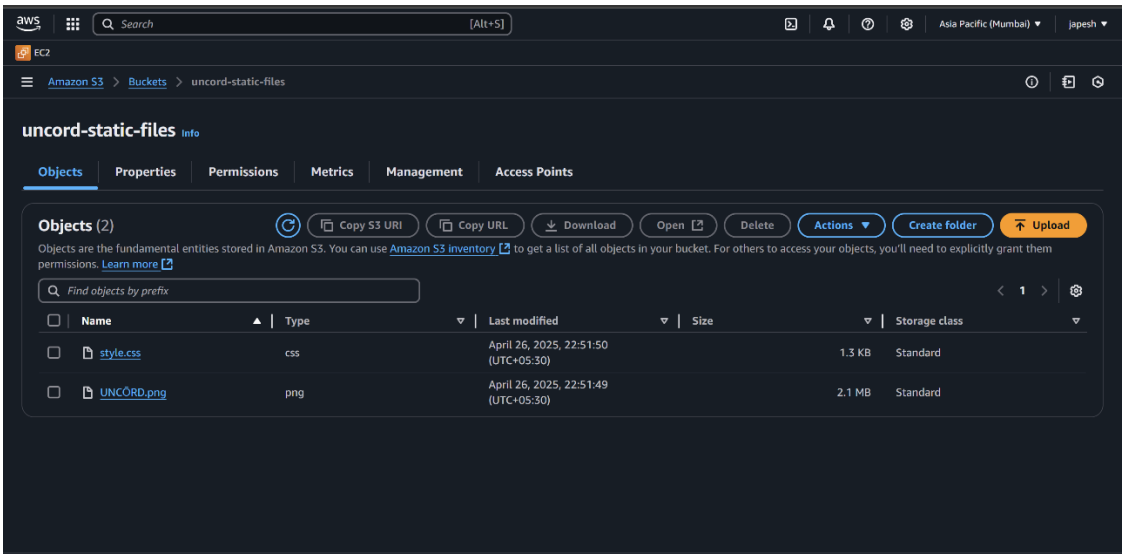
- On EC2 instance, installed and started Apache:

Step 5: Create S3 Bucket for Static Files

- Created an S3 bucket named uncord-static-files.
- Uploaded:
  - A CSS file (style.css)
  - An Image file (image.jpg or .png)
- Made both files public by adjusting permissions and bucket policy.


Search
[Alt+S]
Asia Pacific (Mumbai)
japesh
EC2
Amazon S3 > Buckets > Create bucket
Buckets are containers for data stored in S3.
General configuration
AWS Region
Asia Pacific (Mumbai) ap-south-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
uncord-static-files
Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). Learn More
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix
Object Ownership Info
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.
ACLs disabled (recommended)
ACLs enabled


Search
[Alt+S]
Asia Pacific (Mumbai)
japesh
EC2
Amazon S3 > Buckets > Create bucket
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.
Object Ownership
Bucket owner enforced
Block Public Access settings for this bucket
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 this bucket and its 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 this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
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.



Bucket Policy:

Step 6: Update HTML File

- Connected to EC2 via SSH again.

Edited the default Apache HTML page:

bash:

sudo nano /var/www/html/index.html

- Updated HTML file to:
    - Link to the external CSS hosted in S3.
    - Display the Image from S3.
- Saved and closed the

CSS File URL:

https://uncord-static-files.s3.ap-south-1.amazonaws.com/style.css

IMAGE URL:

https://uncord-static-files.s3.ap-south-1.amazonaws.com/UNC%C5%8CRD.png

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>UNCŌRD - Professional Clothing Brand</title>
    <link rel="stylesheet" href="https://uncord-static-files.s3.ap-south-1.amazonaws.com/style.css">
</head>
<body>
    <header>
        <div class="logo">
            <img src="https://uncord-static-files.s3.ap-south-1.amazonaws.com/UNC%C5%8CRD.png" alt="UNCŌRD Logo" style="height: 60px;">
        </div>
        <nav>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">Collections</a></li>
                <li><a href="#">About Us</a></li>
                <li><a href="#">Contact</a></li>
                <li><a href="#">Shop Now</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section class="hero">
            <h1>Step into Style with UNCŌRD</h1>
            <p>Discover premium fashion crafted for champions. Stay bold, stay limitless.</p>
            <a href="#" class="btn">Explore Collection</a>
        </section>

        <section class="features">
            <div class="feature">
                <h2>Performance Wear</h2>
                <p>Engineered for comfort, built for excellence.</p>
            </div>
            <div class="feature">
                <h2>Street Style</h2>
                <p>Blend of modern design and everyday style.</p>
            </div>
            <div class="feature">
                <h2>Limited Editions</h2>
                <p>Stand out with exclusive UNCŌRD collections.</p>
            </div>
        </section>
    </main>

    <footer>
        <p>&copy; 2025 UNCŌRD. All rights reserved.</p>
    </footer>
</body>
</html>
```

Configure VPC and Subnet for Internet Access

Default VPC Usage

- Used AWS's **default VPC**, which is automatically created in each region.
- The default VPC comes with:
    - A default public **subnet**.
    - An attached **Internet Gateway (IGW)**.
    - Preconfigured route tables allowing internet access.

Security Group Settings

- Modified the **EC2 instance's security group** to allow inbound traffic:
    - **SSH** (port 22) — from **My IP** (for secure instance management).
    - **HTTP** (port 80) — from **Anywhere** (0.0.0.0/0) to allow public website access.

Public IP Assignment

- Ensured that the EC2 instance was launched with a **Public IPv4 address** assigned.
- This allowed the website to be accessible over the internet using the instance's public IP.

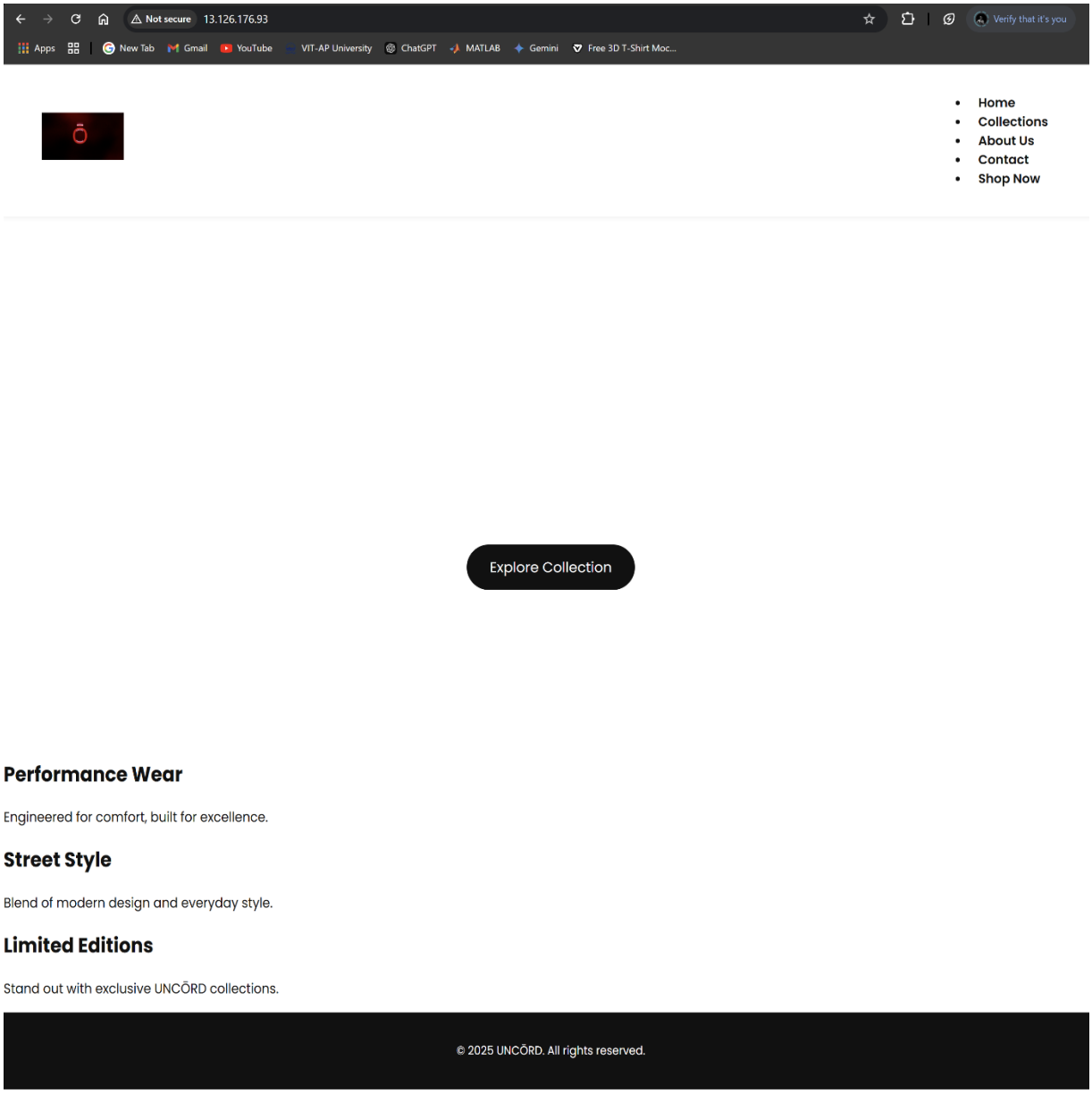
Step 7: Verify Website

- Opened browser and visited http://13.126.176.93/.
- Verified that:
    - Custom styling (CSS) was applied.
    - Image from S3 bucket was displayed

Public IPv4 address

13.126.176.93

WEBSITE LOOK:



WEBSITE LINK:

http://13.126.17693/\

🛠️

Technologies

Used

- **Amazon S3** – Static file hosting
- **AWS IAM** – Access and permissions
- **Amazon Route 53** *(optional)* – Domain name system management
- **AWS CloudFront** *(optional)* – CDN and HTTPS support
- **HTML/CSS/JS** – Static website content

📌 Use Cases

- Personal

- portfolio websites
- Company landing pages
- Documentation portals
- Marketing or product launch pages

WEBSITE LINK:

http://13.126.17693/\