***File-Sharing Web Application on AWS***

PROJECT REPORT

subervised BY:

eng.sAlma

Team Members and Roles

|  |  |
| --- | --- |
|  | **AWS Account Setup, IAM Setup** |
|  | **S3 & File Handling** |
|  | **VPC & Testing** |
|  | **EC2 & connections Setup** |
|  | **Monitoring & documentation**  **Bonus points** |

1. Introduction

The goal of this project was to develop a simplified web application for file sharing, hosted on AWS. The application allows users to upload files, which are then stored securely in Amazon S3 (Simple Storage Service). Users can then download the files using a link generated after a successful upload or delete it. The backend of the application is hosted on an EC2 instance, with proper IAM roles and permissions set up for secure access to S3 storage.

2. Architecture Overview

## High-Level Architecture

The architecture for the file-sharing web application is broken down into the following AWS components:

1. Amazon EC2 (Elastic Compute Cloud): The web server hosting the application logic, which handles the file upload and retrieval process. It also serves the HTML, CSS, and JavaScript files that make up the frontend.
2. Amazon S3 (Simple Storage Service): A highly scalable, secure, and durable storage service where uploaded files are stored. S3 acts as the central storage for all user files.
3. IAM (Identity and Access Management): Manages permissions securely, allowing the EC2 instance to interact with the S3 bucket while preventing unauthorized access.
4. Amazon VPC (Virtual Private Cloud): Ensures the EC2 instance is deployed in a secure and isolated network, providing controlled access to the application.

3. Setup and Configuration

## 3.1 AWS Account Setup

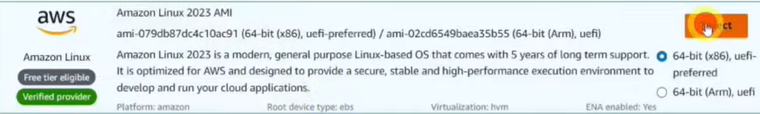
To begin the project, we created an AWS Free Tier account. AWS Free Tier offers access to many AWS services at no charge, within certain usage limits. We created an account specifically for this project to avoid any charges by staying within the Free Tier limits.

After creating the AWS account, Billing Alert Created: A billing alert was configured using AWS Budgets, with a threshold of $1 To ensure usage does not exceed $1 and still in free tier.

## 3.2 EC2 Instance Setup

**3.2.1 Launching the EC2 Instance**

1. Selecting an EC2 Instance: We launched a t2.micro instance (under the AWS Free Tier) with an Amazon Linux 2 AMI. The Amazon Linux AMI is a lightweight and secure operating system optimized for AWS, ensuring better performance and stability.



1. Security Group Configuration: We created a security group for the EC2 instance, which controls the inbound and outbound traffic. Specifically, we allowed:
   * HTTP (Port 80): For the web application to be accessible from the internet.

**3.2.2 IAM Users, Roles, and Permissions**

* Users Created per Team Member: Each team member was given an IAM user with access and AWS Console login.
* All users and roles were assigned policies according to their specific role .

## 3.3 Amazon S3 Setup

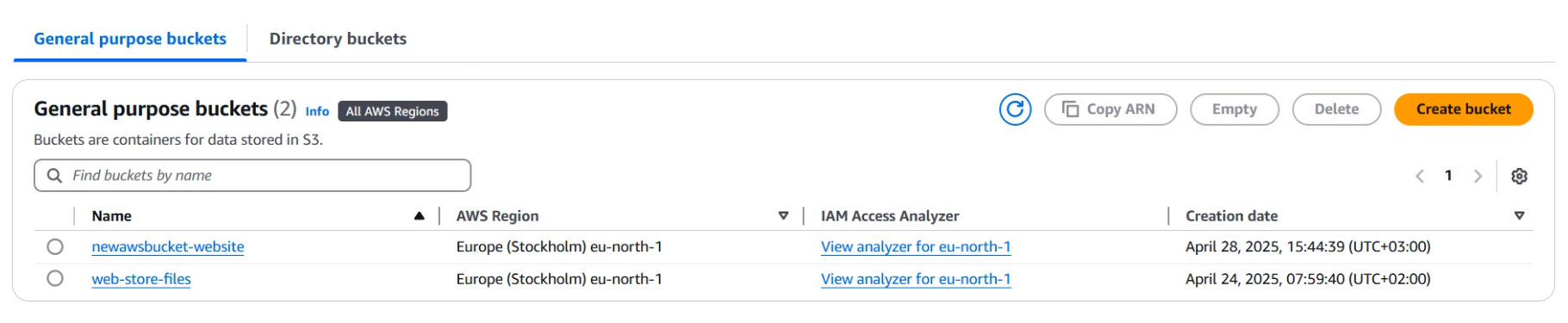
**3.3.1 Creating an S3 Bucket**

We created a new S3 bucket named “my-file-sharing-bucket”, where the files will be stored.

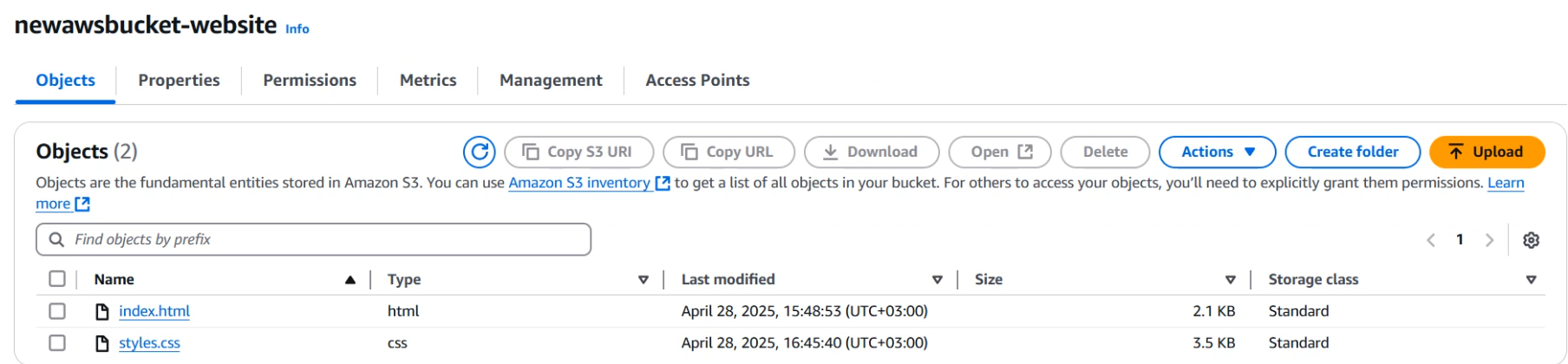
**3.3.2 S3 Bucket Policy and Permissions**

We configured the S3 bucket policy to ensure secure file uploads and downloads:

* EC2 Role Access: We allowed the EC2 instance (via the IAM role) to upload files to the bucket, ensuring secure interactions between EC2 and S3.
* Public Read Access: To enable users to download their uploaded files, we configured the S3 bucket to allow public read access for object (**Block Public Access**: Disabled).
* 2Buckets:



* File Storage Bucket
  + Name: my-file-sharing-bucket
  + Block Public Access: Enabled
  + Permissions: EC2 role allowed to read/write objects.
* Frontend Static Site Bucket
  + Name: my-static-frontend
  + Block Public Access: Disabled
  + Static Website Hosting: Enabled
    - Index document: index.html
    - Error document: error.html
  + ECL (Endpoint Control List): Enabled for logging access.



## 3.4 VPC Setup

**3.4.1 VPC Creation**

We created a custom VPC to host the EC2 instance within an isolated and secure network. The VPC setup included:

* CIDR Block: We chose the CIDR block 10.0.0.0/16, allowing us to create multiple subnets in the future if needed.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Subnets: We created a public subnet associate with vpc where the EC2 instance was deployed, ensuring it had internet access via a public IP.

A screenshot of a computer

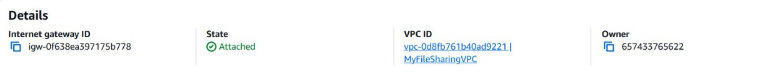
AI-generated content may be incorrect.

* create route table: then add root with target -> internet gateway .

A close-up of a computer screen

AI-generated content may be incorrect.

* Internet Gateway: We attached an internet gateway to the VPC, allowing the EC2 instance to communicate with the internet.



**3.4.2 Security Groups**

The EC2 instance was assigned to a security group that allowed :

Source: 0.0.0.0/0 (public access)

* **HTTP (Port 80)**: This port was open to allow web traffic from users.

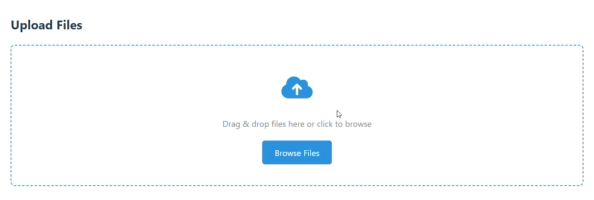
## 3.5 Web Application Development

The frontend of the web application was developed using basic HTML, CSS, and JavaScript:

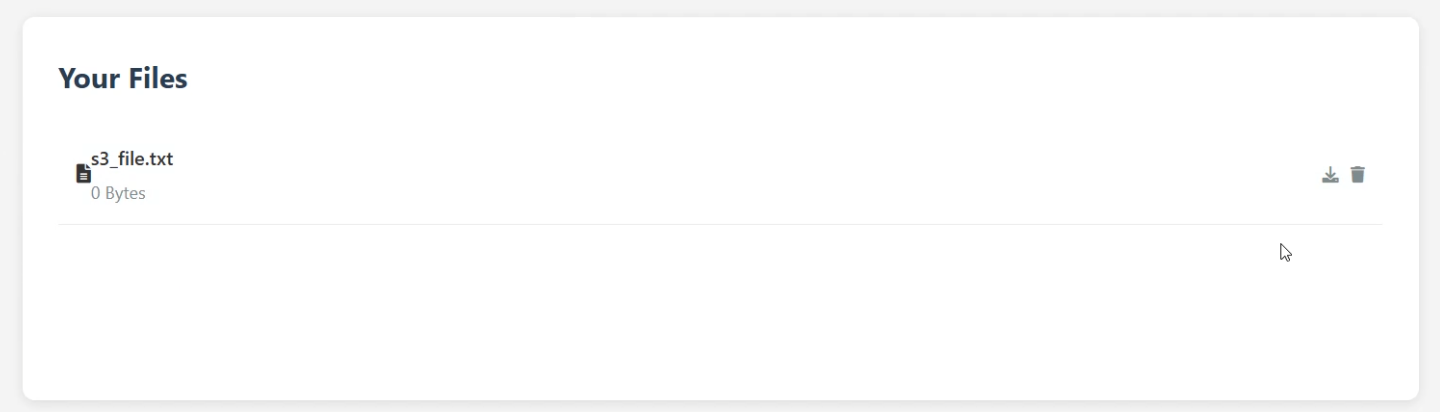
* HTML: The web page contains a simple file upload form that allows users to select and upload files.
* CSS: The interface is designed to be minimalist but functional, with clear instructions for the user.
* JavaScript: JavaScript is used to handle the form submission, send the file to the server via HTTP, and display the download link upon a successful upload and delete.

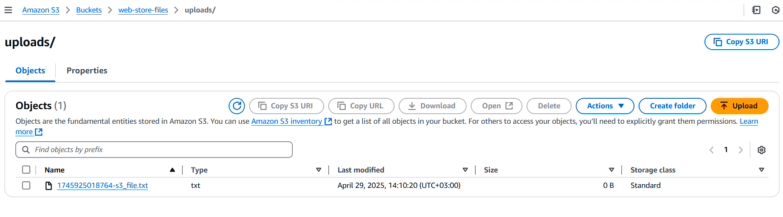
**Features of the app** :

* **File Upload**



* Files retrieval
* Download file
* Delete file





## 3.6 Testing

**3.6.1 Manual Testing**

We tested the application thoroughly by:

* Uploading various types of files (PDFs, text files) and ensuring they were correctly stored in the S3 bucket and can be retrieved .
* Validating that the download links generated for each file worked as expected.
* Validating delete option.

Monitoring

* CloudWatch Alarms: Configured for EC2 CPU usage.
* Billing Notification
* **Bonus Tasks Implemented**
  + ✅ Display a List of Uploaded Files
  + ✅ Styled Frontend UI (CSS-enhanced)
  + ✅ Static Frontend Hosted on S3

4 Conclusion

The project successfully demonstrated how to integrate multiple AWS services to build a scalable, secure file-sharing application. We gained practical experience with AWS EC2, S3, IAM, and VPC, and understood how to deploy, secure, and manage cloud-based applications. Despite facing challenges, we successfully implemented key features like file uploads, downloads, and CloudWatch monitoring, ensuring a functional and reliable solution.

***Appendix.***

Demo Link: https://drive.google.com/drive/my-drive?dmr=1&ec=wgc-drive-hero-goto