Simple Cloud-Based Application Deployment

Concepts Used:

AWS Cloud9, S3, and EC2.

Problem Statement:

Develop a simple HTML page using AWS Cloud9 and deploy it to an S3 bucket for static website hosting. Set up an EC2 instance as a backup server for the website.

Tasks:

- **Create a basic HTML page** using AWS Cloud9.
- **Deploy the HTML page** to an S3 bucket and enable static website hosting.
- **Configure an EC2 instance** to serve as a backup server for the website.

---

1. Introduction

Case Study Overview

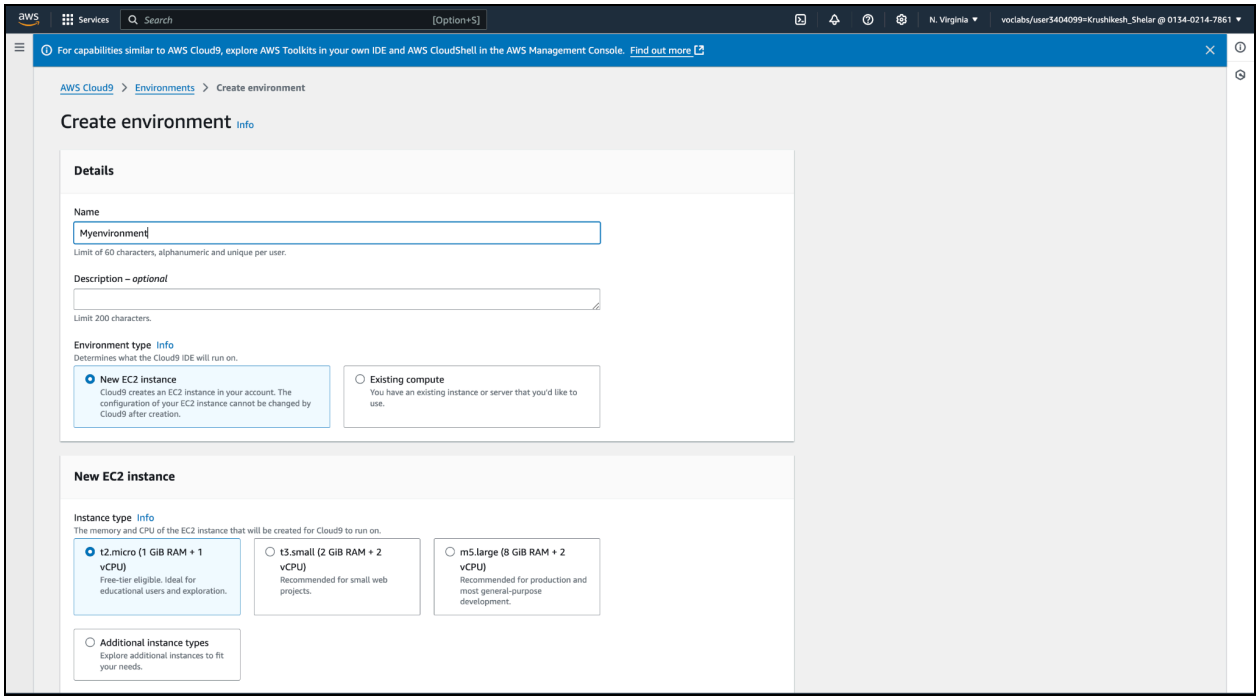
This case study focuses on deploying a simple cloud-based application using AWS services. It involves creating a basic HTML page, hosting it on an S3 bucket, and setting up an EC2 instance as a backup server. This deployment demonstrates the practical use of cloud infrastructure for web hosting and disaster recovery.

---

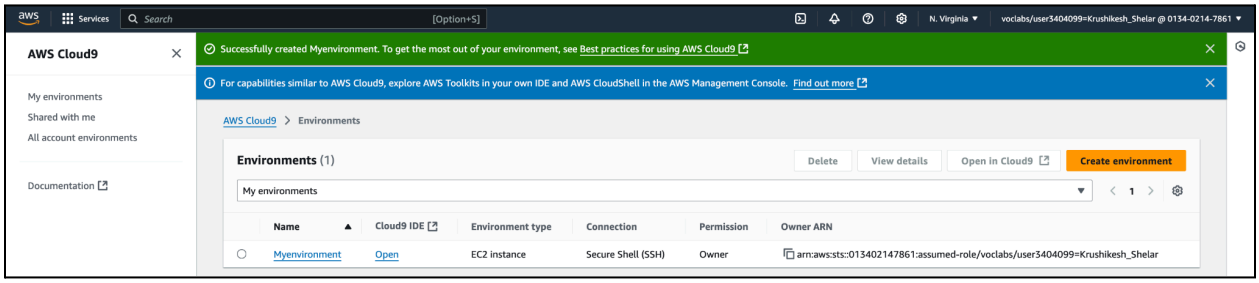
2. Step-by-Step Explanation

Step 1: Initial Setup

- **Create an AWS Account:** Sign up for AWS if you don't already have an account.
- **Access AWS Cloud9:**
    - Navigate to the AWS Management Console.
    - Search for "Cloud9" and click **Create environment**.



- Provide a name and description.
- Choose **Create a new EC2 instance for environment** (recommended).
- Select the instance type (e.g., t2.micro).
- Click **Create environment**.



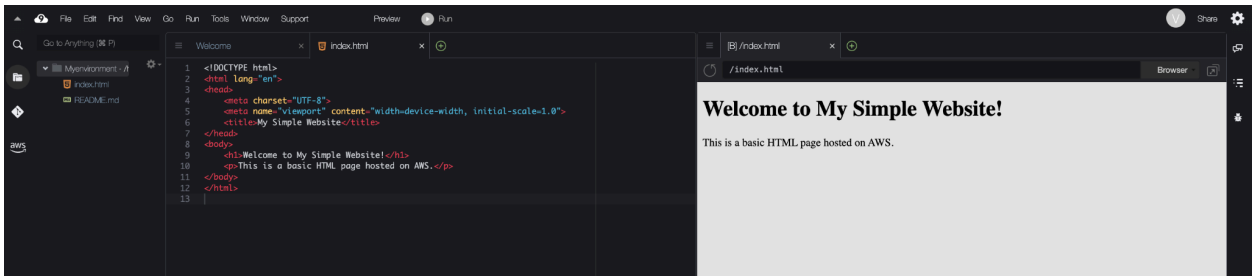
Step 2: Create the HTML Page

- In Cloud9, create a new file named `index.html`.
- Add the following code:

html
Copy code

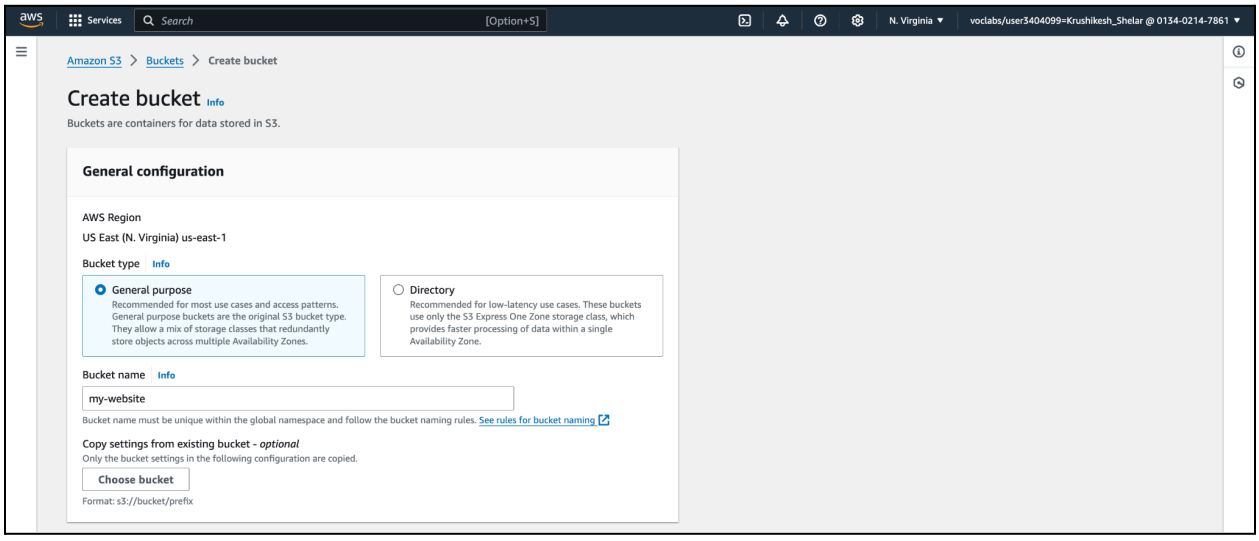
```
<!DOCTYPE html>
<html lang="en">
<head>
```

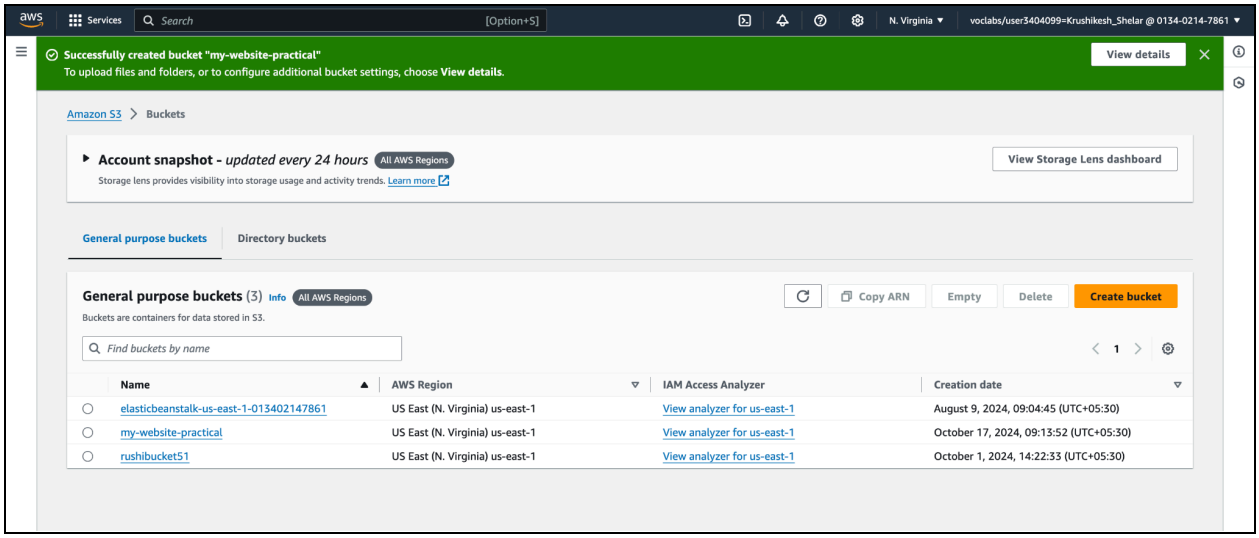
```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>My Simple Website</title>
</head>
<body>
    <h1>Welcome to My Simple Website!</h1>
    <p>This is a basic HTML page hosted on AWS.</p>
</body>
</html>
```



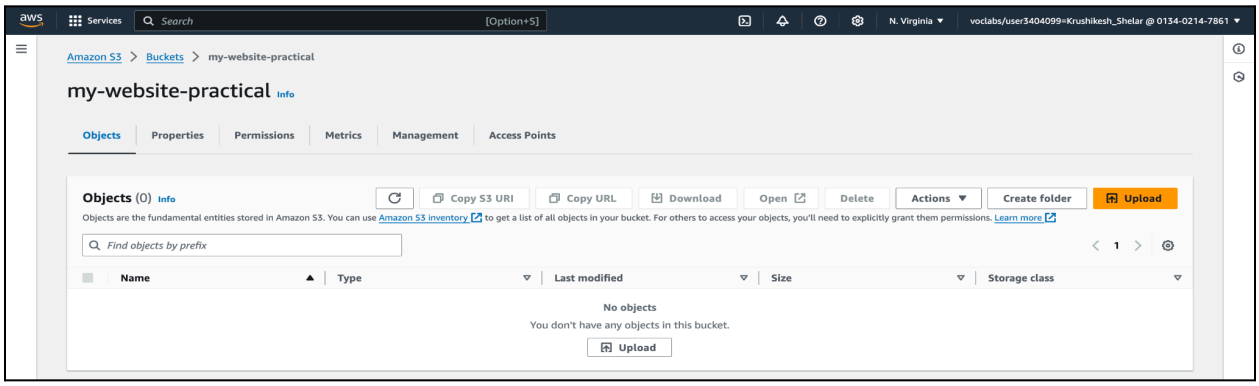
Step 3: Deploy the HTML Page to S3

1. **Create an S3 Bucket:**
    - Navigate to the **S3** service.
    - Click **Create bucket** and name it (e.g., `my-simple-website-bucket`).
    - Uncheck **Block all public access** and acknowledge the warning.
    - Click **Create bucket**.

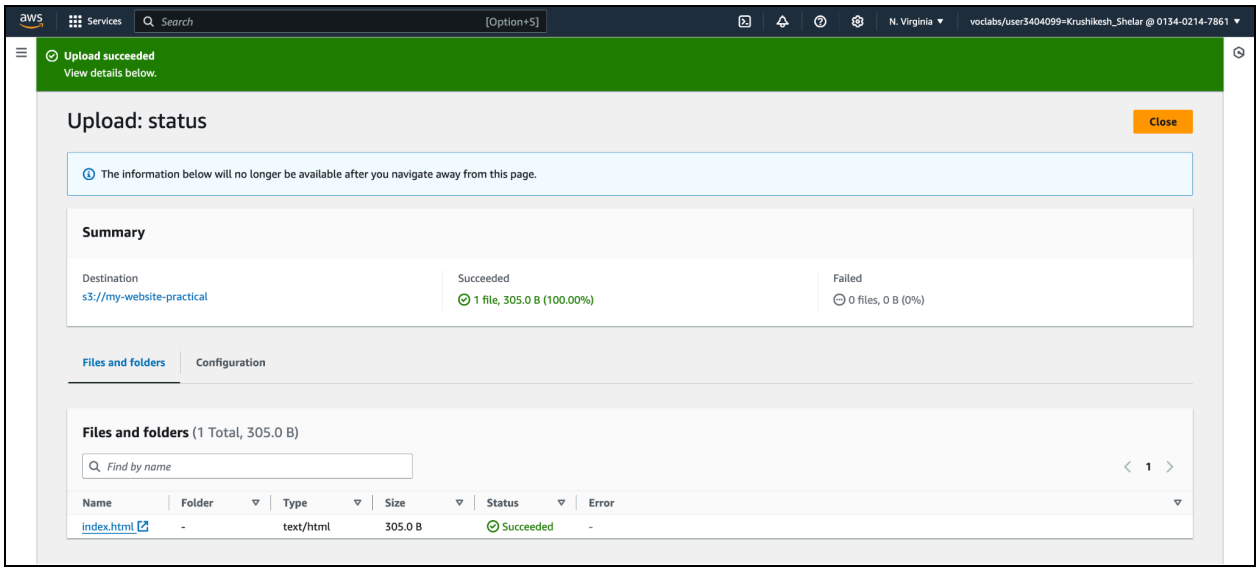




2. **Upload the HTML File:**
   - Select your bucket.
   - Click **Upload > Add files** and select `index.html`.



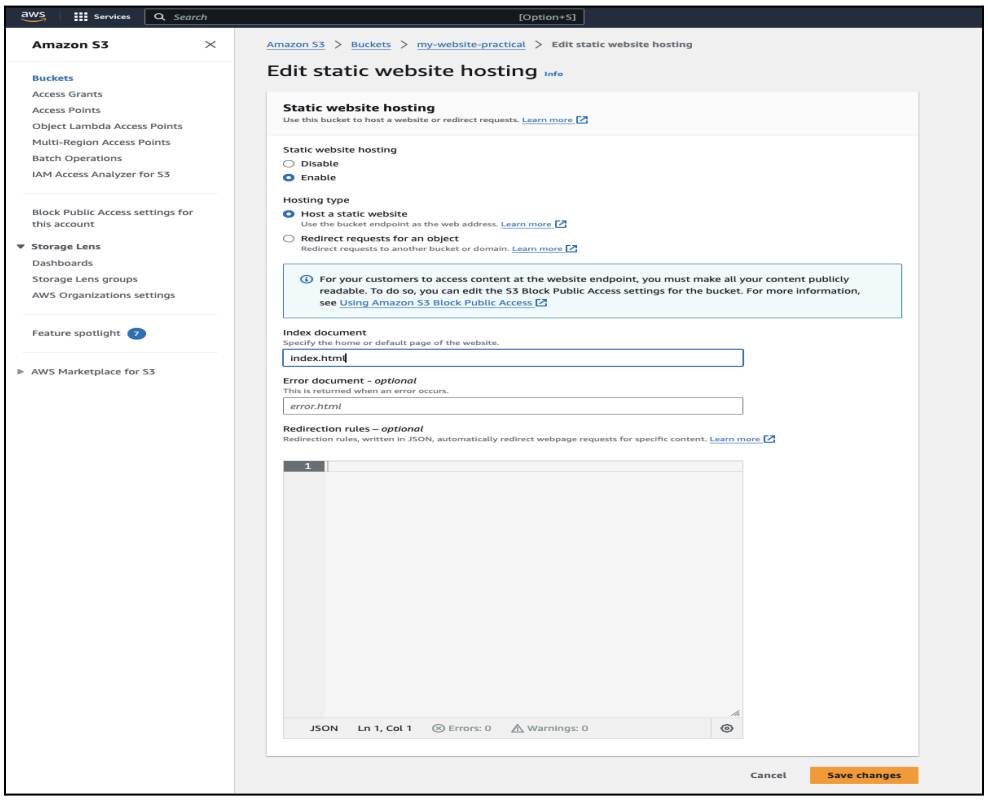
   - Click **Upload**.

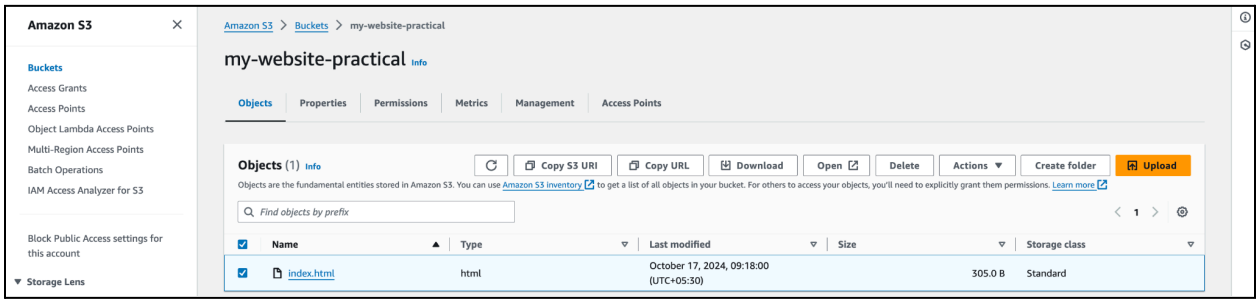


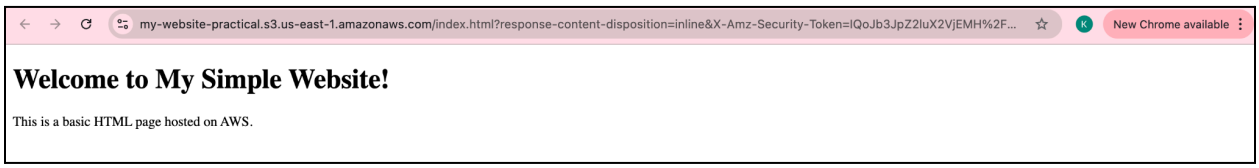
3. **Enable Static Website Hosting:**
   - In **Properties**, click **Static website hosting**.
   - Select **Use this bucket to host a website**.
   - Set **index document** to `index.html`.
   - Save changes and note the endpoint URL (also set the policies)

http://my-website-practical.s3-website-us-east-1.amazonaws.com/).

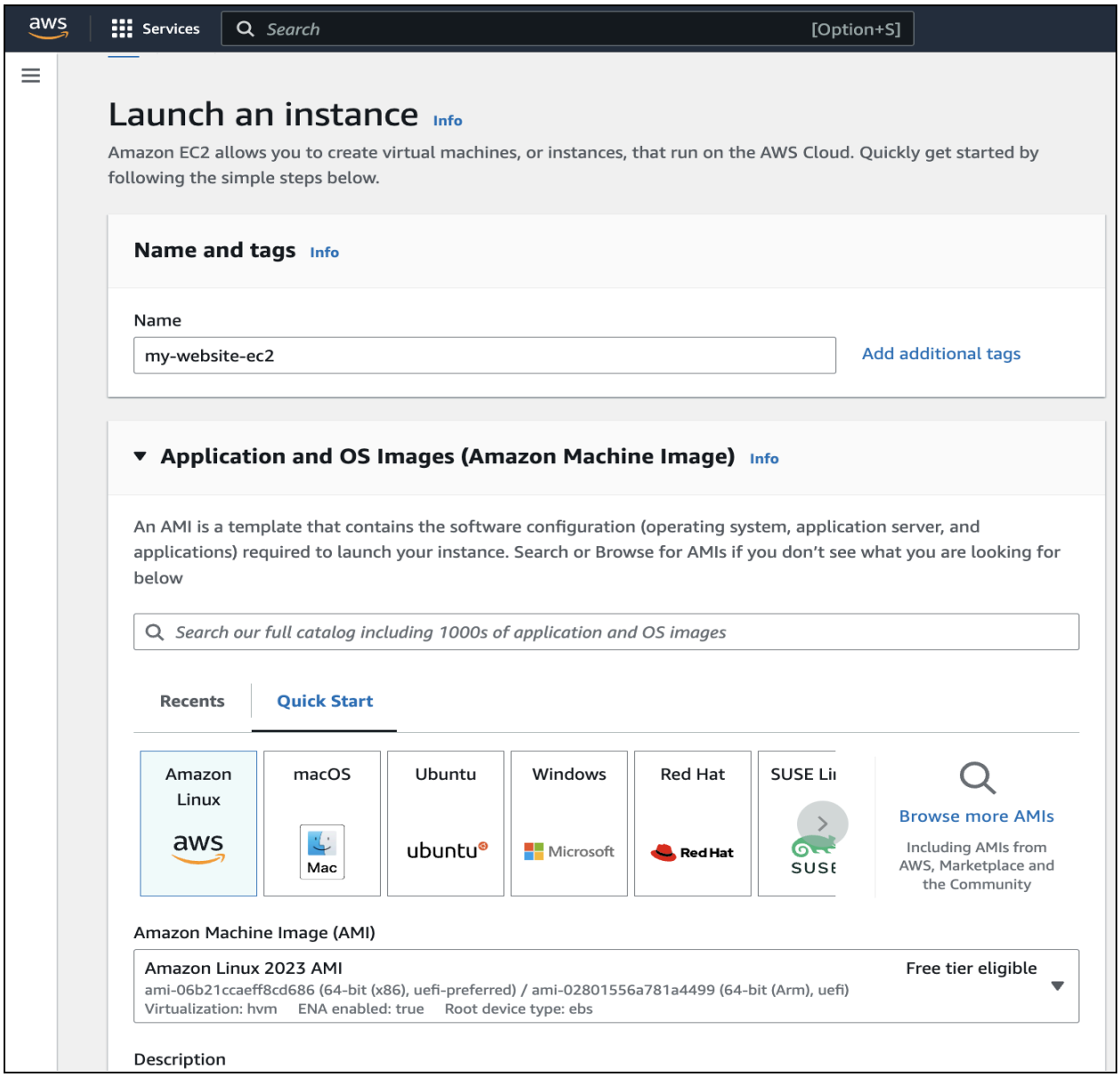
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::your-bucket-name/*"
    }
  ]
}
```







Step 4: Launch an EC2 Instance

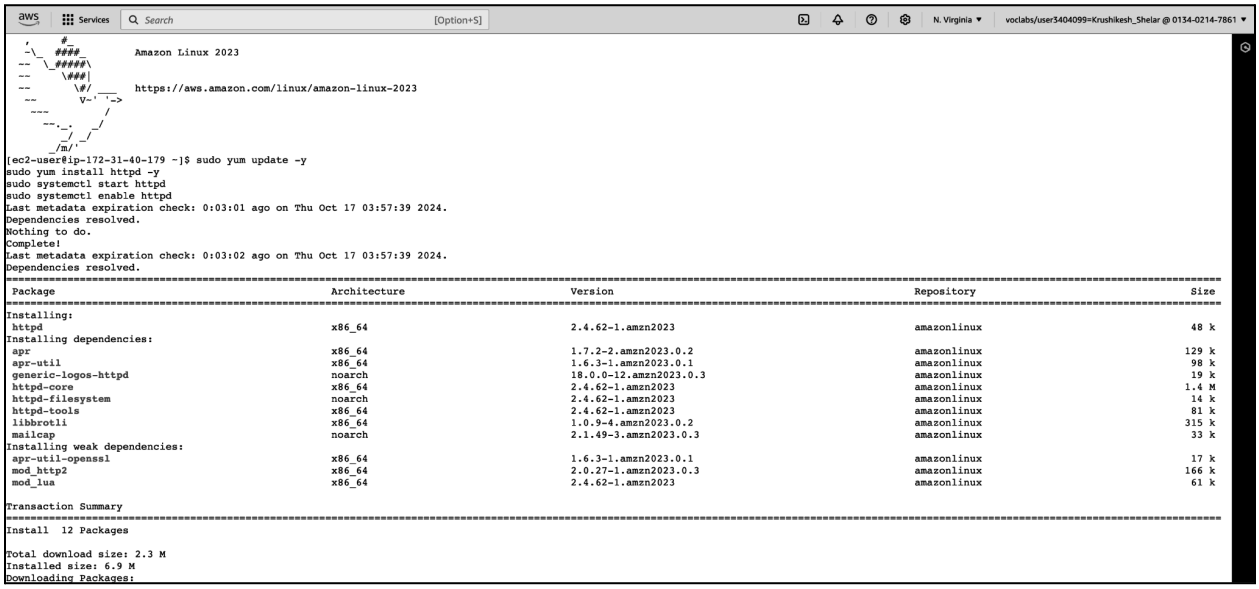


1. **Create an EC2 Instance:**
   - Navigate to the **EC2** service.
   - Click **Launch Instance** and select an Amazon Linux 2 AMI.
   - Choose the instance type (e.g., t2.micro).
   - Configure **security groups** to allow HTTP (port 80) and SSH (port 22).
   - Launch the instance.
2. **Install a Web Server on EC2:**
   - SSH into the EC2 instance using the key pair.

Run the following commands to install Apache:
bash

Copy code

```
sudo yum update -y
sudo yum install httpd -y
sudo systemctl start httpd
sudo systemctl enable httpd
```



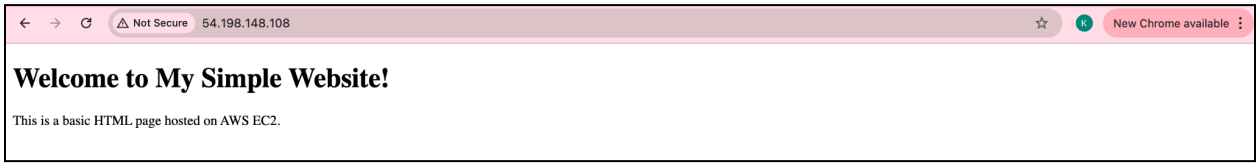
3. **Copy the HTML Page to EC2:**

Run the following commands:
bash
Copy code

```
sudo su
echo '<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>My Simple Website</title>
</head>
<body>
    <h1>Welcome to My Simple Website!</h1>
    <p>This is a basic HTML page hosted on AWS EC2.</p>
</body>
</html>' > /var/www/html/index.html
```

Step 5: Verify EC2 Website

1. Go to the EC2 instance in the AWS Console.
2. Copy the **Public IPv4 address**.
3. Open a browser and paste the IP address to verify the web server is running.
4. Note:- Make sure that you visit the URL with HTTP protocol and not HTTPS.



---

3. Guidelines and Tips

- Ensure the S3 bucket permissions allow public access to the static website.
- Check if your AWS Academy account allows direct access to the endpoint; if not, try opening it manually through the S3 bucket options.

---

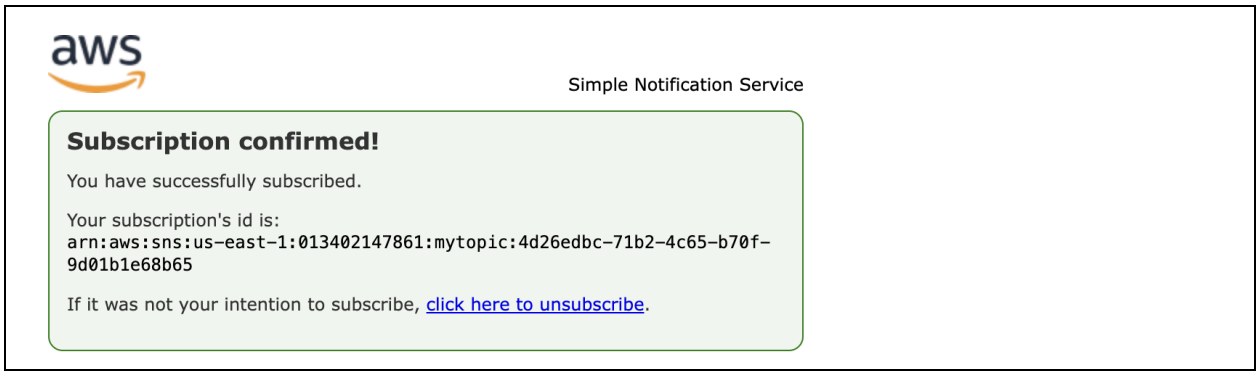
4. Challenges Faced

1. **Restricted Access to Endpoints:**
   - The AWS Academy environment limits access to public S3 bucket endpoints, preventing direct access to the hosted static website.
   - **Workaround:** The website had to be accessed through the "Open" option within the S3 console and also set the right bucket policies.
2. **Public Access Configurations:**
   - Enabling public access for S3 buckets involved warnings about potential security risks.
   - **Workaround:** We carefully configured the bucket permissions to allow public access while adhering to security best practices.

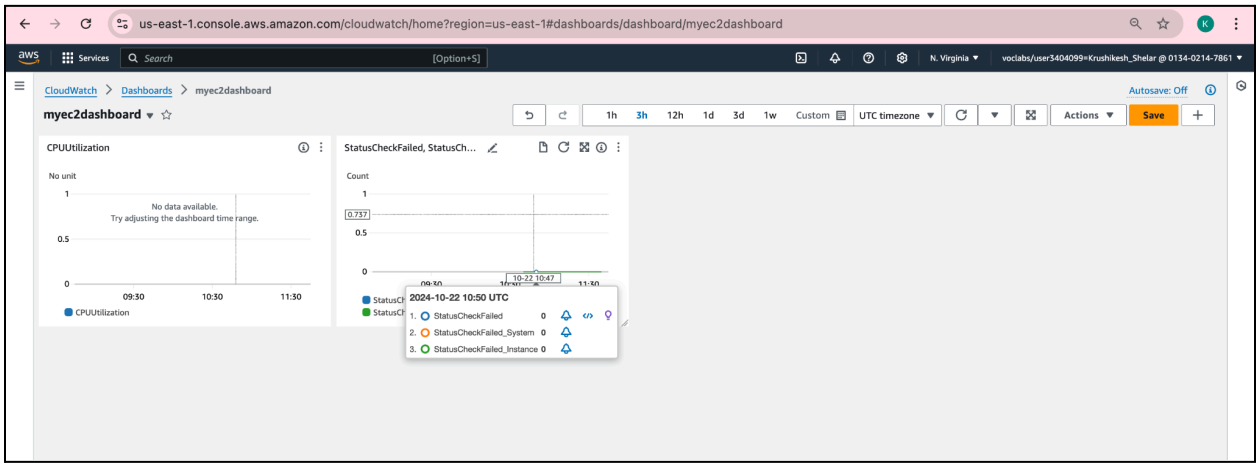
---

5. Enhanced Monitoring and Disaster Recovery

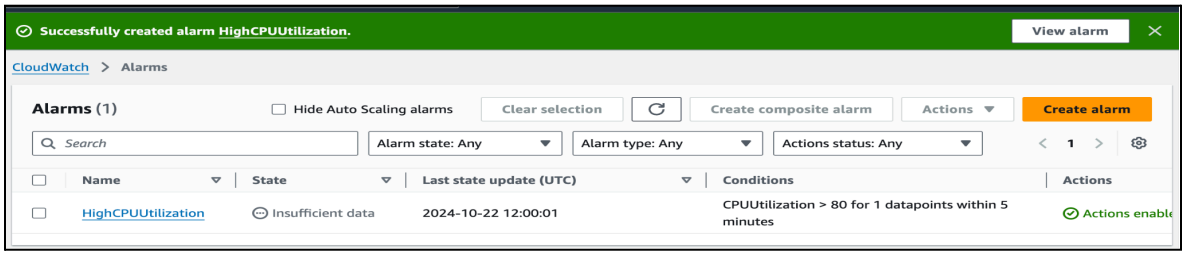
- SNS Alerting System - Configured Simple Notification Service (SNS) for automated email alerts - Integrated with CloudWatch for incident notifications



- CloudWatch Monitoring - Implemented critical metrics monitoring: EC2 CPU Utilization (80% threshold) , Instance Status Checks , HTTP Response Status



- **Implemented EC2 CPU Utilization Alarm**:
  - **Threshold**: Alerts when CPU utilization exceeds **80%**.
  - Triggers SNS notifications for timely incident response.



6.Conclusion:

This case study demonstrates a simple and practical cloud-based application deployment. Hosting the primary website on S3 ensures cost-effective and highly available static hosting, while the backup EC2 server guarantees continuity in case of issues. This architecture highlights the importance of redundancy and efficient resource utilization in modern cloud environments.