

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**.

Create environment

Details

Name:

Description - optional:

Environment type

☒ **New EC2 instance**
Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.

☐ **Existing compute**
You have an existing instance or server that you'd like to use.

New EC2 instance

Instance type

☒ **t2.micro** (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.

☐ **t3.small** (2 GiB RAM + 2 vCPU)
Recommended for small web projects.

☐ **m5.large** (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose development.

☐ **Additional instance types**
Explore additional instances to fit your needs.

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**.

Environments (1)

Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
Myenvironment	Open	EC2 instance	Secure Shell (SSH)	Owner	arn:aws:sts:013402147861:assumed-role/voclabs/user3404099=Krushikesh_Shelar

Create environment

Step 2: Create the HTML Page

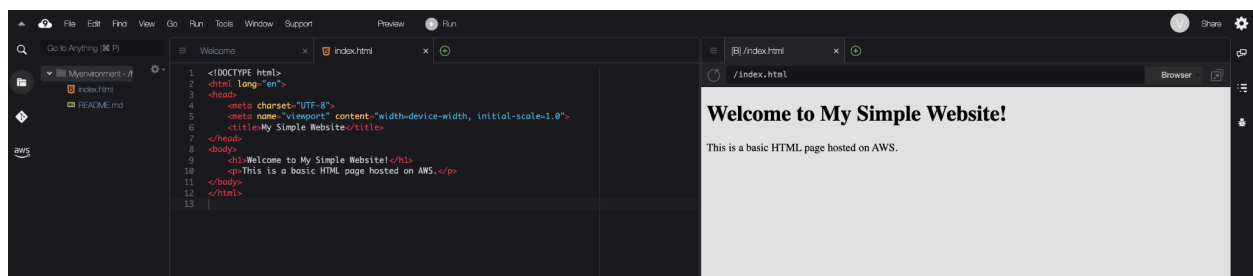
- In Cloud9, create a new file named `index.html`.
- Add the following code:

html

Copy code

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

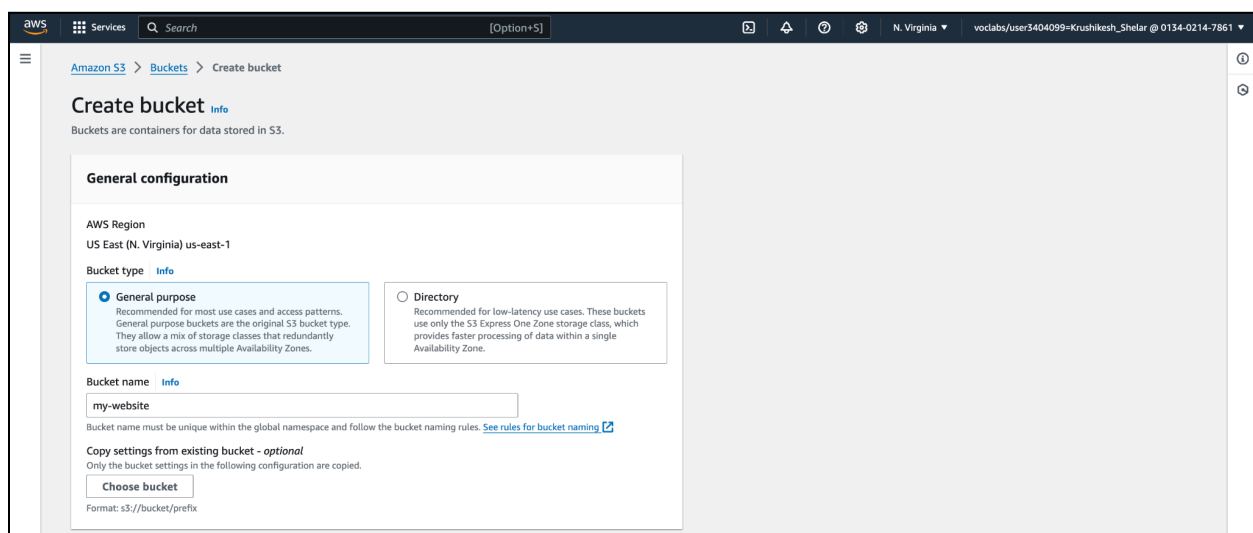
```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>My Simple Website</title>
</head>
<body>
  <h1>Welcome to My Simple Website!</h1>
  <p>This is a basic HTML page hosted on AWS.</p>
</body>
</html>
```

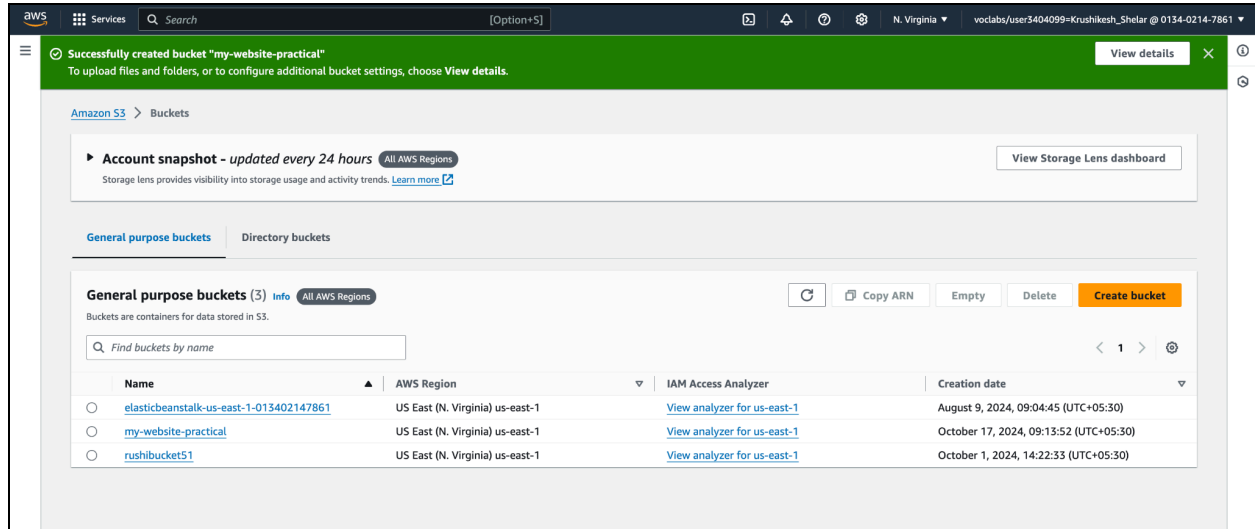


Step 3: Deploy the HTML Page to S3

1. Create an S3 Bucket:

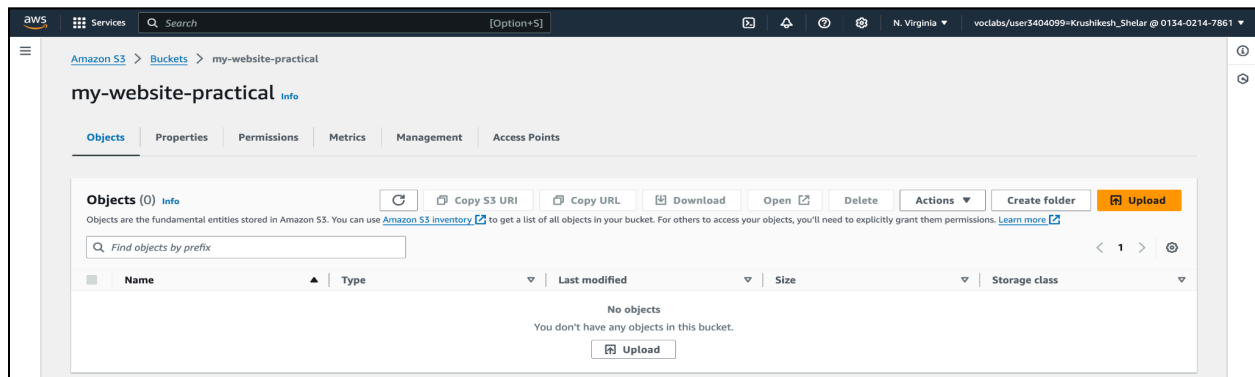
- Navigate to the **S3** service.
- Click **Create bucket** and name it (e.g., **my-simple-website-bucket**).
- Uncheck **Block all public access** and acknowledge the warning.
- Click **Create bucket**.



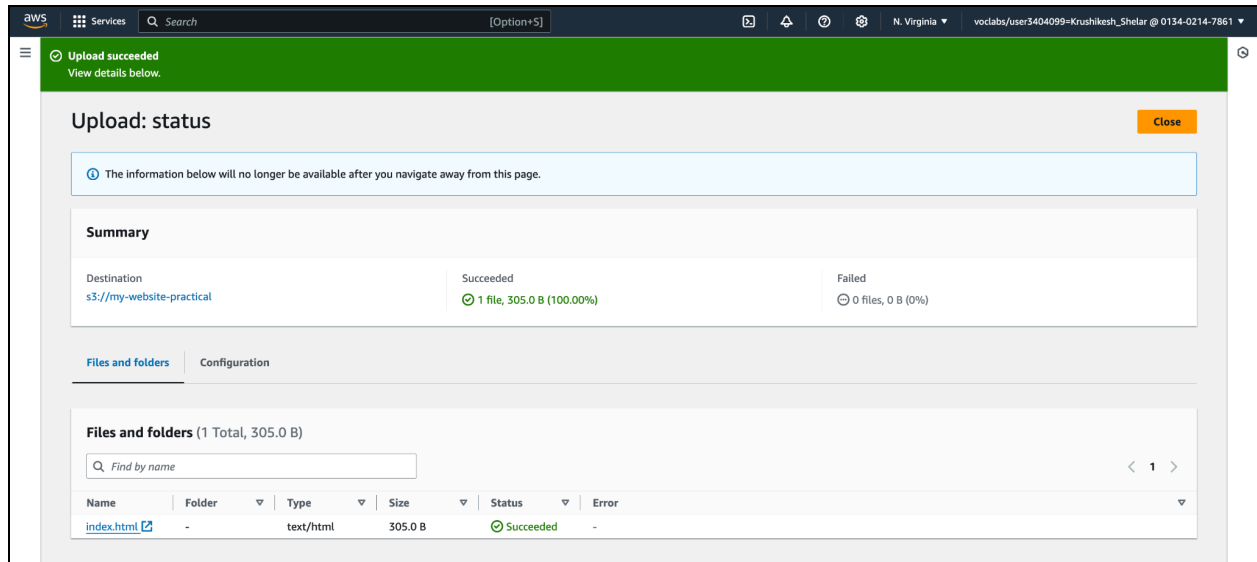


2. Upload the HTML File:

- Select your bucket.
- Click **Upload > Add files** and select **index.html**.



- Click **Upload**.

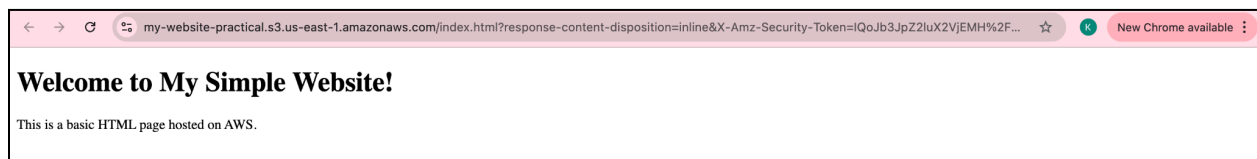
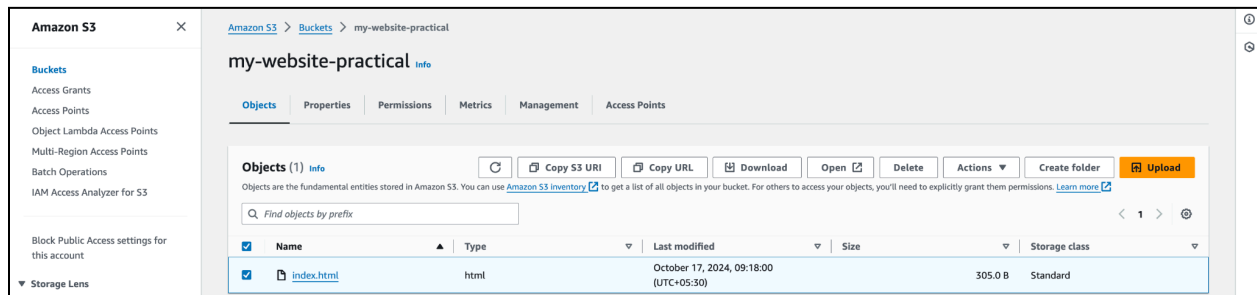
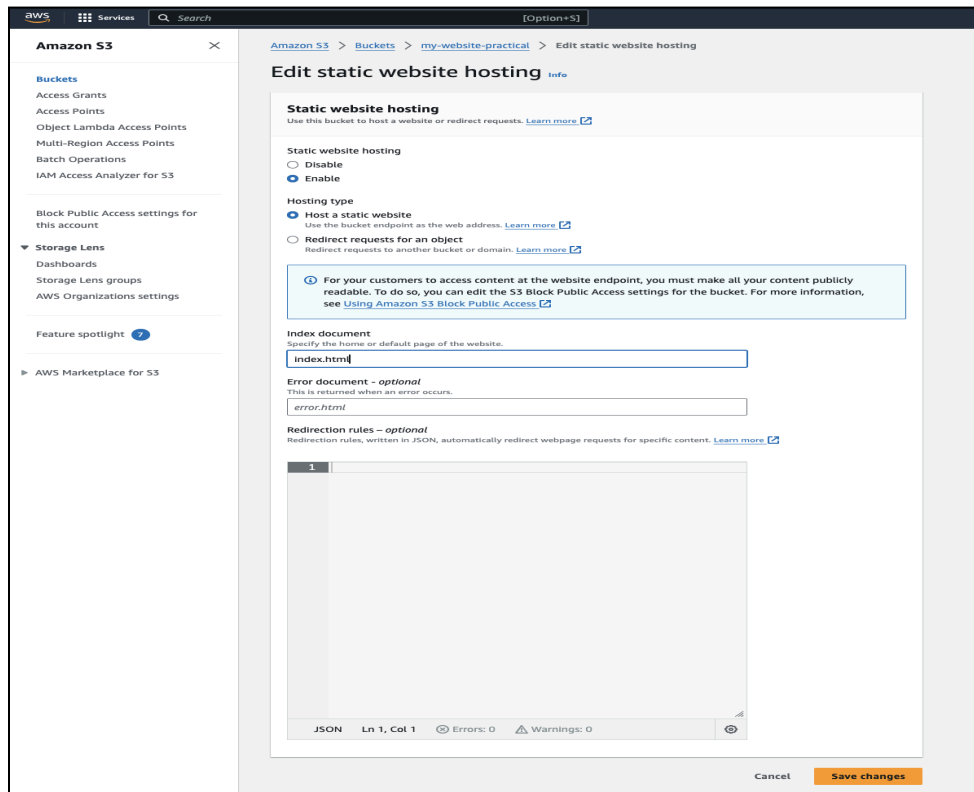


3. Enable Static Website Hosting:

- 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

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name
my-website-ec2 [Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Recents | **Quick Start**

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI Free tier eligible

ami-06b21c9aeff8cd686 (64-bit (x86), uefi-preferred) / ami-02801556a781a4499 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

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
```

```

aws Services Q Search [Option+S] N. Virginia voclabs/user3404099@Krushikesh_Shelar @ 0154-0214-7861
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-40-179 ~]$ sudo yum update -y
sudo yum install httpd -y
sudo systemctl start httpd
sudo systemctl enable httpd
Last metadata expiration check: 0:03:01 ago on Thu Oct 17 03:57:39 2024.
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:03:02 ago on Thu Oct 17 03:57:39 2024.
Dependencies resolved.
=====
Package                                Architecture      Version            Repository          Size
-----
Installing:
httpd                                  x86_64            2.4.62-1.amzn2023  amazonlinux         48 k
Installing dependencies:
apr                                   x86_64            1.7.2-2.amzn2023.0.2  amazonlinux        129 k
apr-util                             x86_64            1.6.3-1.amzn2023.0.1  amazonlinux         98 k
generic-logos-httpd                  noarch            18.0.0-12.amzn2023.0.3  amazonlinux         19 k
httpd-core                           x86_64            2.4.62-1.amzn2023    amazonlinux        1.4 M
httpd-filesystem                     noarch            2.4.62-1.amzn2023    amazonlinux         14 k
httpd-tools                          x86_64            2.4.62-1.amzn2023    amazonlinux         81 k
libbrotili                           x86_64            1.0.9-4.amzn2023.0.2  amazonlinux        315 k
mailcap                              noarch            2.1.49-3.amzn2023.0.3  amazonlinux         33 k
Installing weak dependencies:
apr-util-openssl                     x86_64            1.6.3-1.amzn2023.0.1  amazonlinux         17 k
mod_http2                            x86_64            2.0.27-1.amzn2023.0.3  amazonlinux        166 k
mod_lua                              x86_64            2.4.62-1.amzn2023    amazonlinux         61 k
Transaction Summary
-----
Install 12 Packages

Total download size: 2.3 M
Installed size: 6.9 M
Downloading Packages:

```

3. Copy the HTML Page to EC2:

Run the following commands:

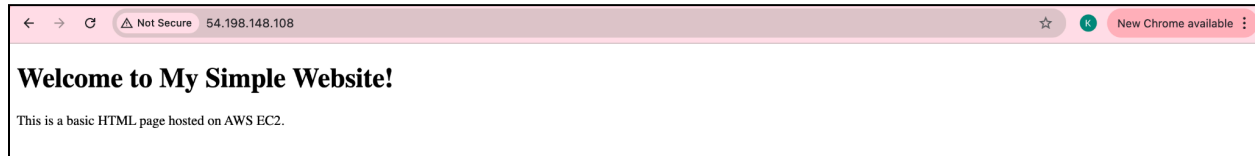
bash

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

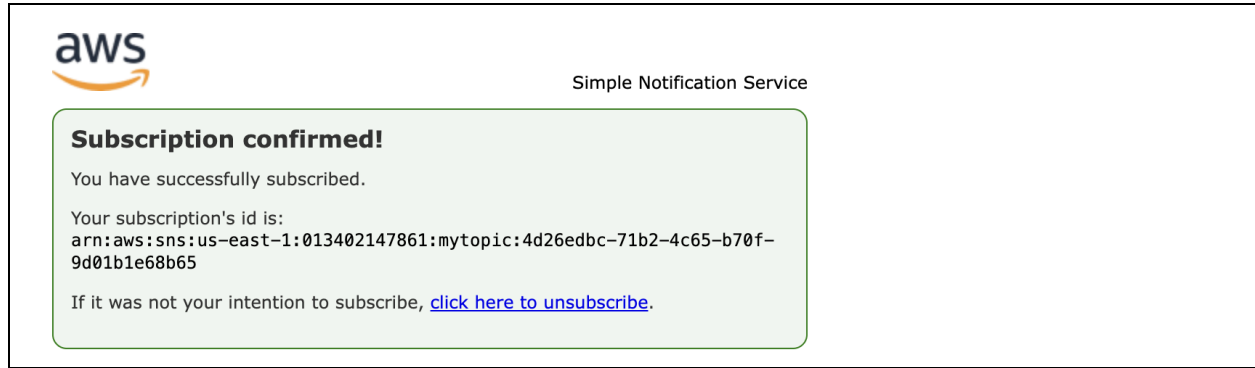
- Ensure the S3 bucket permissions allow public access to the static website.
- Check if your AWS Academy account allows direct access to the endpoint; if not, try opening it manually through the S3 bucket options.

4. Challenges Faced

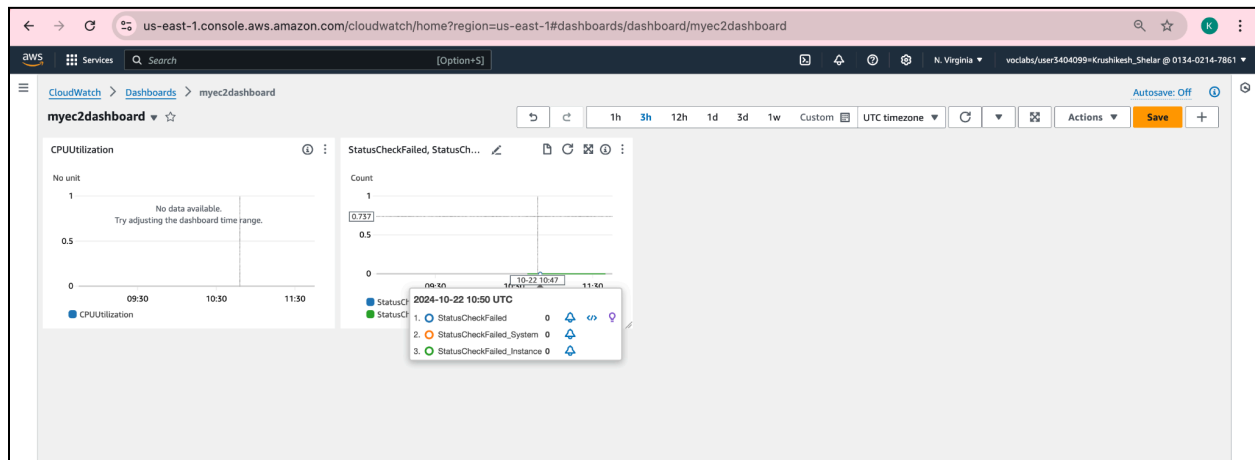
1. **Restricted Access to Endpoints:**
 - The AWS Academy environment limits access to public S3 bucket endpoints, preventing direct access to the hosted static website.
 - **Workaround:** The website had to be accessed through the "Open" option within the S3 console and also set the right bucket policies.
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

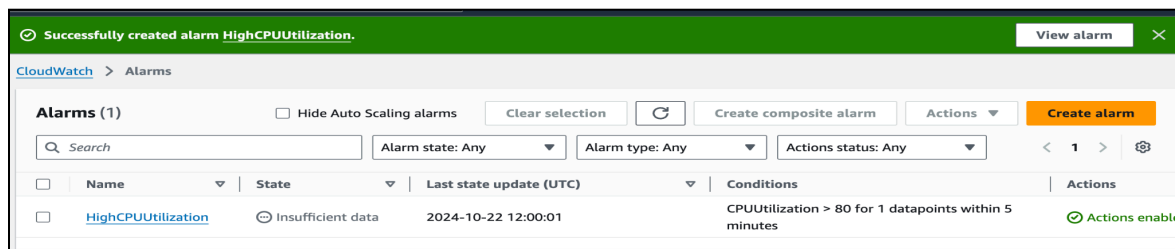
- SNS Alerting System - Configured Simple Notification Service (SNS) for automated email alerts - Integrated with CloudWatch for incident notifications



- 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.