

# 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.

### Key Feature and Application

The core feature of this project is integrating AWS Cloud9, S3, and EC2 to showcase a cost-effective and scalable static website hosting solution. The inclusion of a backup EC2 server ensures high availability, allowing uninterrupted access even in case of server failure.

---

## 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**.

The screenshot shows the 'Create environment' page in the AWS Cloud9 console. The 'Name' field is filled with 'Myenvironment'. The 'Description - optional' field is empty. Under 'Environment type', the 'New EC2 instance' option is selected. Below this, under 'New EC2 instance', the 't2.micro (1 GiB RAM + 1 vCPU)' instance type is selected. The 'Additional instance types' option is also visible.

- 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**.

The screenshot shows the 'Environments' page in the AWS Cloud9 console. A green banner at the top indicates 'Successfully created Myenvironment'. Below this, a table lists the environments. The table has columns: Name, Cloud9 IDE, Environment type, Connection, Permission, and Owner ARN. One environment, 'Myenvironment', is listed with the type 'EC2 instance' and connection 'Secure Shell (SSH)'. The 'Create environment' button is visible in the top right corner.

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

## Step 2: Create the HTML Page

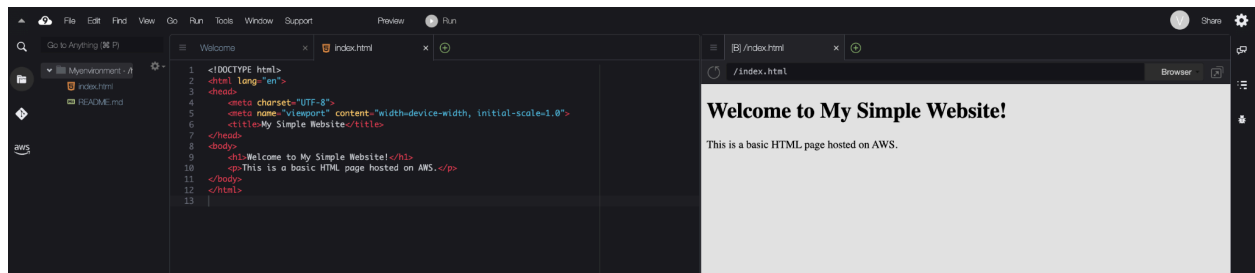
- In Cloud9, create a new file named `index.html`.

- 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**.

The screenshot shows the 'Create bucket' page in the AWS Management Console. The breadcrumb navigation is 'Amazon S3 > Buckets > Create bucket'. The page title is 'Create bucket' with an 'info' link. A sub-header states 'Buckets are containers for data stored in S3.' The 'General configuration' section includes the 'AWS Region' set to 'US East (N. Virginia) us-east-1' and the 'Bucket type' set to 'General purpose' (selected with a radio button). A description for 'General purpose' states it is recommended for most use cases and allows a mix of storage classes. The 'Directory' option is also visible. The 'Bucket name' field contains 'my-website'. Below this, there is a section for 'Copy settings from existing bucket - optional' with a 'Choose bucket' button. The page footer indicates the format 'Format: s3://bucket/prefix'.

The screenshot shows the 'Buckets' page in the AWS Management Console. A green banner at the top indicates 'Successfully created bucket "my-website-practical"'. Below this, there is a section for 'Account snapshot - updated every 24 hours'. The 'General purpose buckets' tab is selected, showing a list of buckets. The table has columns for Name, AWS Region, IAM Access Analyzer, and Creation date.

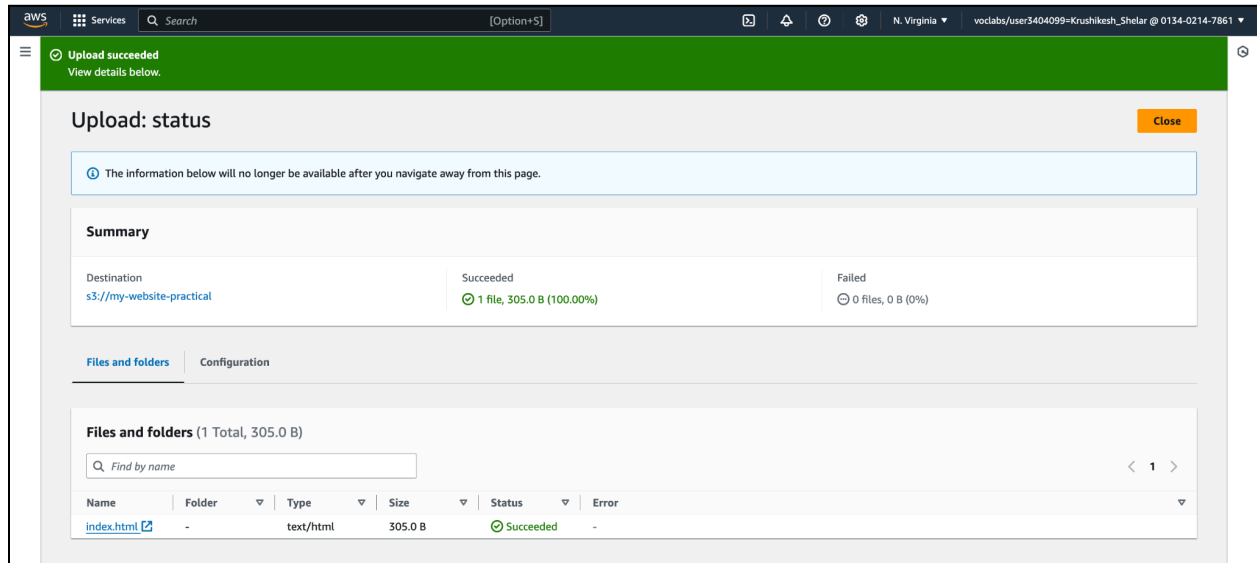
Name	AWS Region	IAM Access Analyzer	Creation date
<a href="#">elasticbeanstalk-us-east-1-013402147861</a>	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	August 9, 2024, 09:04:45 (UTC+05:30)
<a href="#">my-website-practical</a>	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	October 17, 2024, 09:13:52 (UTC+05:30)
<a href="#">rushibucket51</a>	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	October 1, 2024, 14:22:33 (UTC+05:30)

## 2. Upload the HTML File:

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

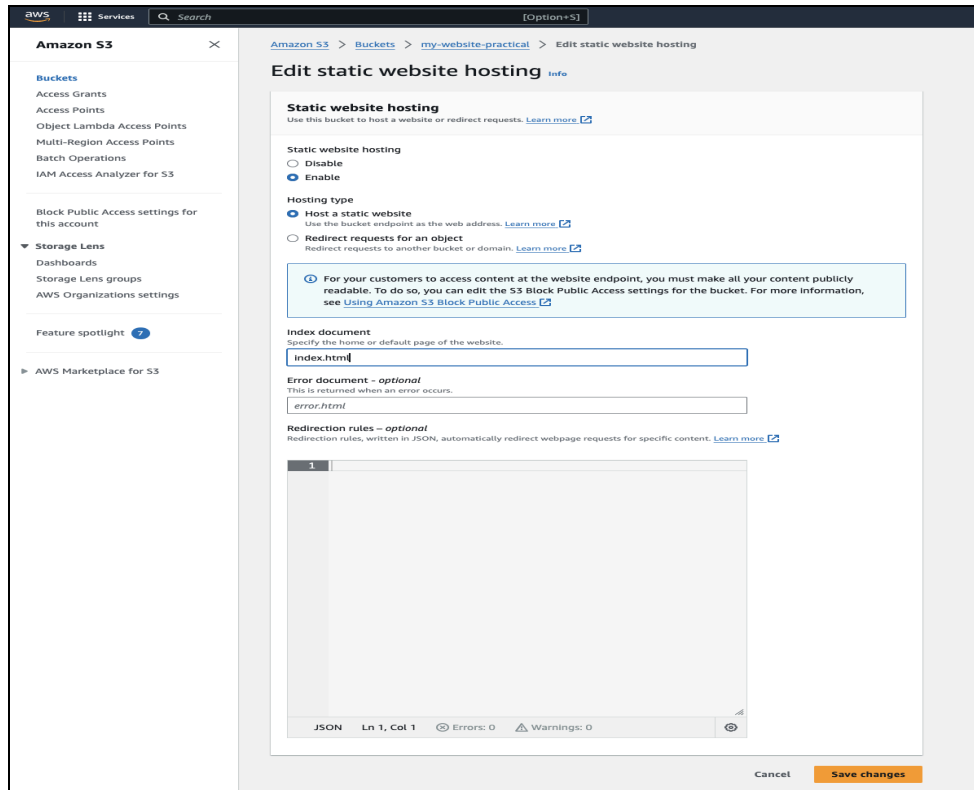
The screenshot shows the 'my-website-practical' bucket page in the AWS Management Console. The breadcrumb navigation is 'Amazon S3 > Buckets > my-website-practical'. The page title is 'my-website-practical' with an 'info' link. The 'Objects' tab is selected, showing a list of objects. The table has columns for Name, Type, Last modified, Size, and Storage class. The message 'No objects' is displayed, indicating that the bucket is currently empty.

- Click **Upload**.

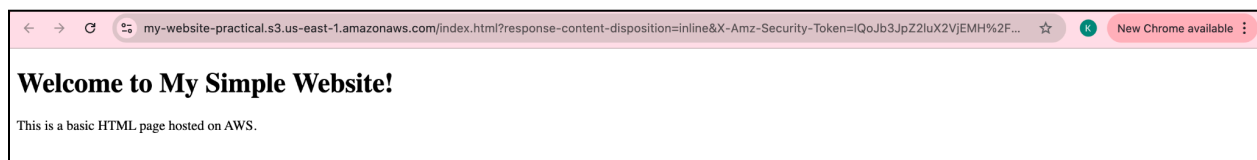
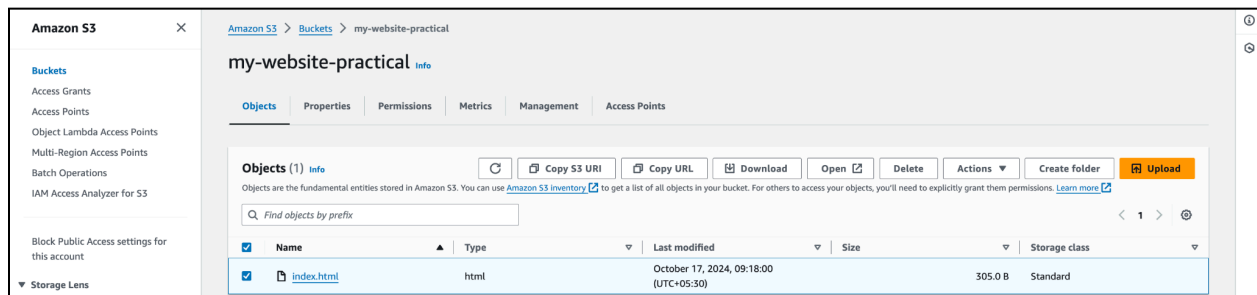


### 3. Enable Static Website Hosting:

- 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
- (e.g., <http://my-website.s3-website-us-east-1.amazonaws.com>).



On AWS ACADEMY Account you may not be directly able to access the endpoints because of the restrictions. You can still manually open the website by clicking on “Open” from the S3 bucket.



## Step 4: Launch an EC2 Instance

The screenshot shows the 'Launch an instance' page in the AWS Management Console. At the top, there's a navigation bar with the AWS logo, 'Services', a search bar, and a '[Option+S]' button. The main heading is 'Launch an instance' with an 'Info' link. Below it, a paragraph explains that Amazon EC2 allows creating virtual machines (instances) on the AWS Cloud. The page is divided into sections: 'Name and tags' with a text input for 'my-website-ec2' and an 'Add additional tags' link; 'Application and OS Images (Amazon Machine Image)' with an 'Info' link. This section includes a description of an AMI, a search bar, and tabs for 'Recents' and 'Quick Start'. Under 'Quick Start', there are tiles for various operating systems: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. To the right is a 'Browse more AMIs' section. Below the tiles, the 'Amazon Machine Image (AMI)' section shows 'Amazon Linux 2023 AMI' with its ID, architecture, and virtualization details. A 'Free tier eligible' badge is also present. At the bottom, a 'Description' section is partially visible.

**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

Search our full catalog including 1000s of application and OS images

Recents | **Quick Start**

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux

**Amazon Machine Image (AMI)**

Amazon Linux 2023 AMI [Free tier eligible](#)

ami-06b21ccea88cd686 (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 [Search] [Option+S] N. Virginia voclabs/user3404099=Krushikesh_Shelar @ 0134-0214-7861
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-46-179 ~]$ 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
libbrotli                             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.
  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. 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.