Project 10: Uploading the EpicReads Website to an Amazon S3 Bucket

Project Objectives:

- To successfully upload the website's files to the S3 bucket using various methods.
- To ensure that the upload process is secure, efficient, and aligns with best practices.
- To lay the groundwork for future updates and maintenance through automated CI/CD processes.

ICON KEY

Throughout this Project, we use various icons to highlight different types of instructions and notes. Here's what each icon signifies:

- 📓 **Note:** Offers a helpful hint, tip, or piece of crucial guidance.
- ℹ️ **Learn More:** Directs you to additional sources for more detailed information.
- ⚠️ **Caution:** Points out information that, while not critical, is important and missing it might necessitate redoing steps.
- 🔥 **WARNING:** Indicates an action that's permanent with potential consequences that could cause process failure or irreversible changes.
- ✅ **Expected Output:** Shows a sample of the expected result to help confirm correct execution of a command or file edit.

- 💻 **Command:** Denotes a specific command that you're required to execute.
- 💡 **Consider:** Encourages you to reflect on how to apply a concept in your scenario or to spark a discussion related to the topic at hand.

COMMON ERRORS

- Command not found errors
- The "aws --version" command returns a different version than you installed
- The "aws --version" command returns a version after uninstalling the AWS CLI
- The AWS CLI processed a command with an incomplete parameter name
- Access denied errors
- Invalid credentials and key errors
- Signature does not match errors
- SSL certificate errors
- I'm using the Amazon S3 static website feature but getting an Access Denied error. Why is this happening?

Refer to *COMMON ERRORS - EXPLANATION SECTION* for the solution.

Start Project:

Welcome back to Project 10 Solution page!

In Project 9 you were able to deploy a Static website in Amazon S3.

In this Project 10. As a Cloud Solutions Architect at EpicReads, your task is to deploy the company's main website with *a customised* website template to Amazon S3 bucket.

You will involve the task using different methods such as the

- AWS Management Console,
- AWS CLI, and
- Automated CI/CD pipeline, ensuring a smooth and efficient upload process.

Let’s start right now!

Task 1.Upload the website's files using AWS Management Console

STEP 1: Create Amazon S3 Bucket

Navigate to *S3 bucket* by just typing the service on to the *Search box* in the *Navigation bar (Figure 10.1: Navigate to S3 bucket)*

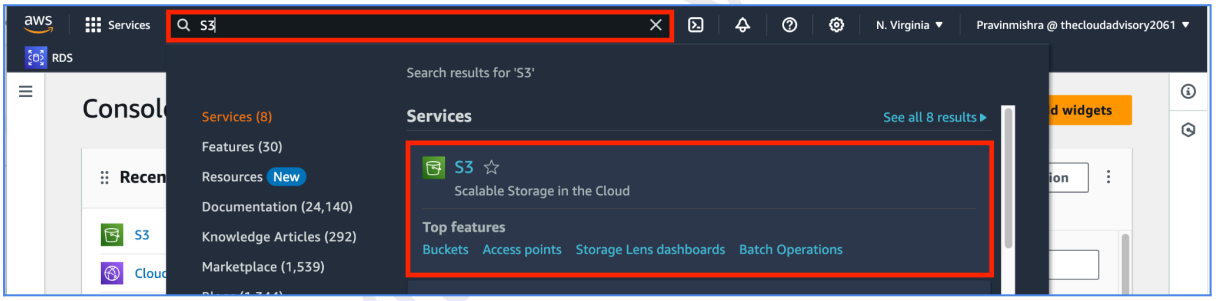


Figure 10.1: Navigate to S3 bucket

Click on *Create bucket (Figure 10.2: Create bucket)*

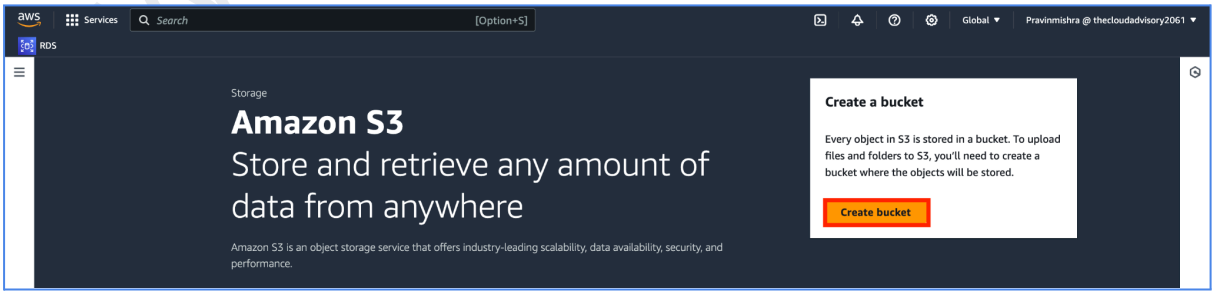


Figure 10.2: Create bucket

Under general configuration *Choose AWS region*, give the *Bucket name (Figure 10.3: General configuration)*

ℹ️ **Learn More:** To check the guidelines for Bucket naming rules you may check *here*

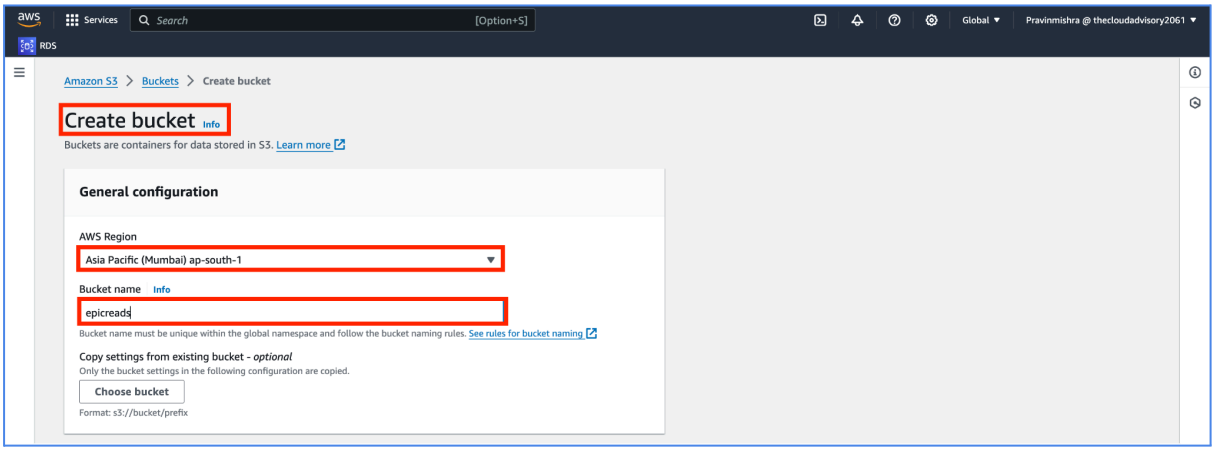


Figure 10.3: General configuration

Scrolling down *uncheck* the checkbox *Block Public Access settings for this bucket.* So that the files in the S3 bucket will be *Publicly accessible.*

Click on the checkbox that you acknowledge the current settings *(Figure 10.4: Uncheck Block all public access)*

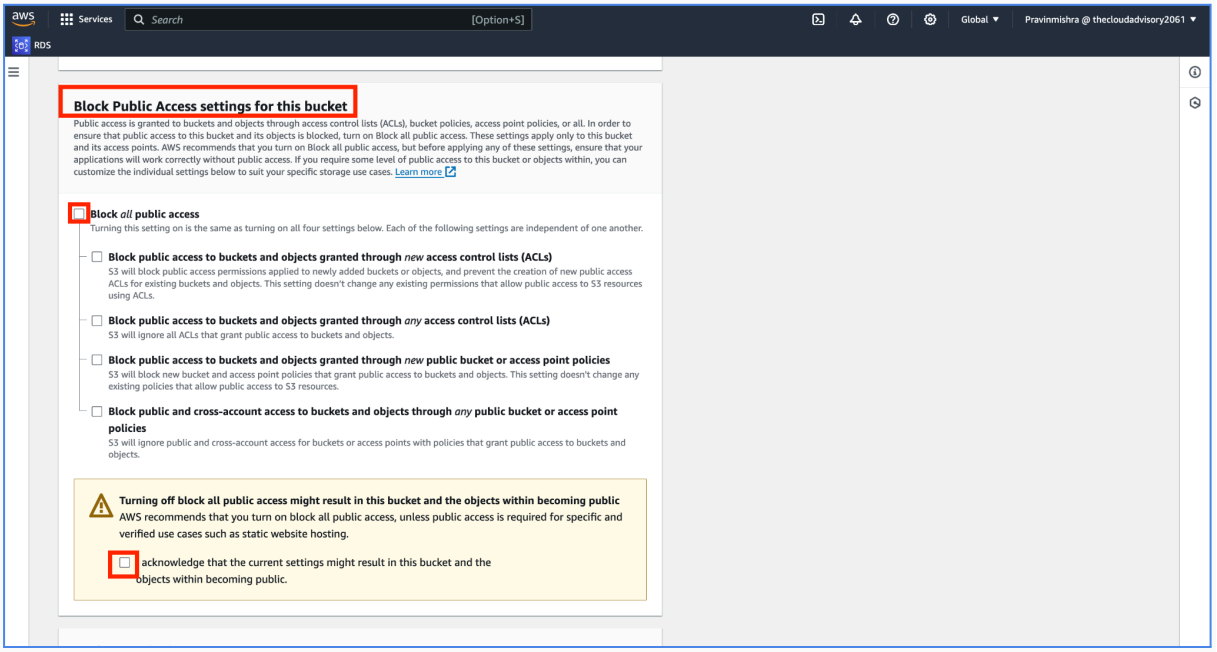


Figure 10.4: Uncheck Block all public access

Click on the button *Create bucket (Figure 10.5: Create bucket)*

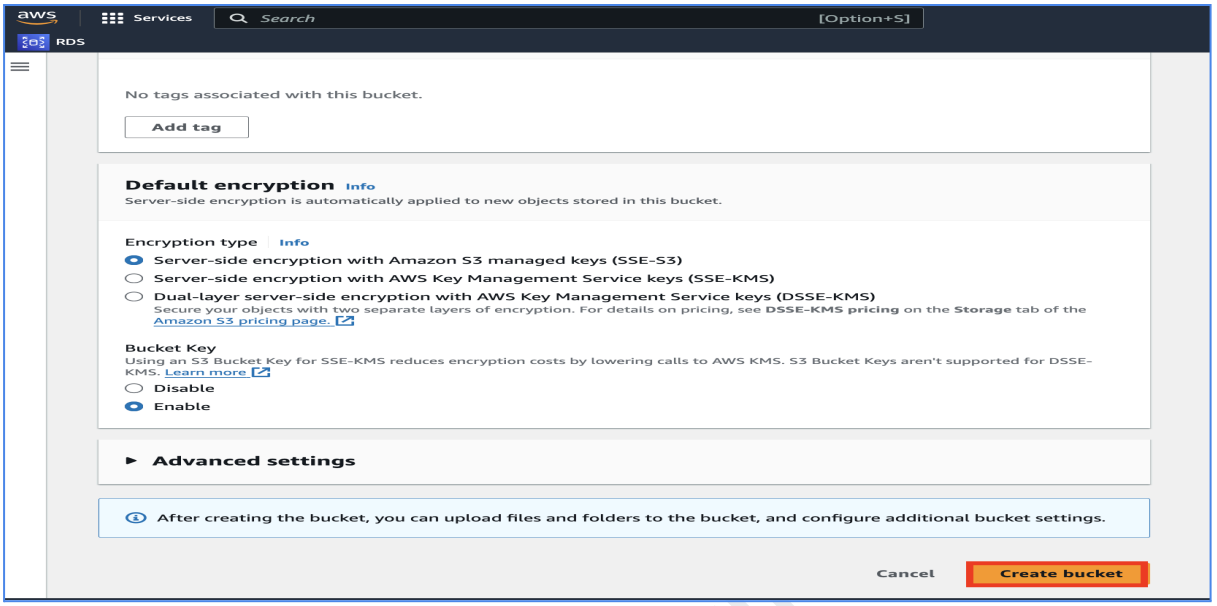


Figure 10.5: Create bucket

You will find a success message on top, and thus you have created a *S3 bucket for EpicReads*. *(Figure 10.6: EpicReads S3 bucket)*

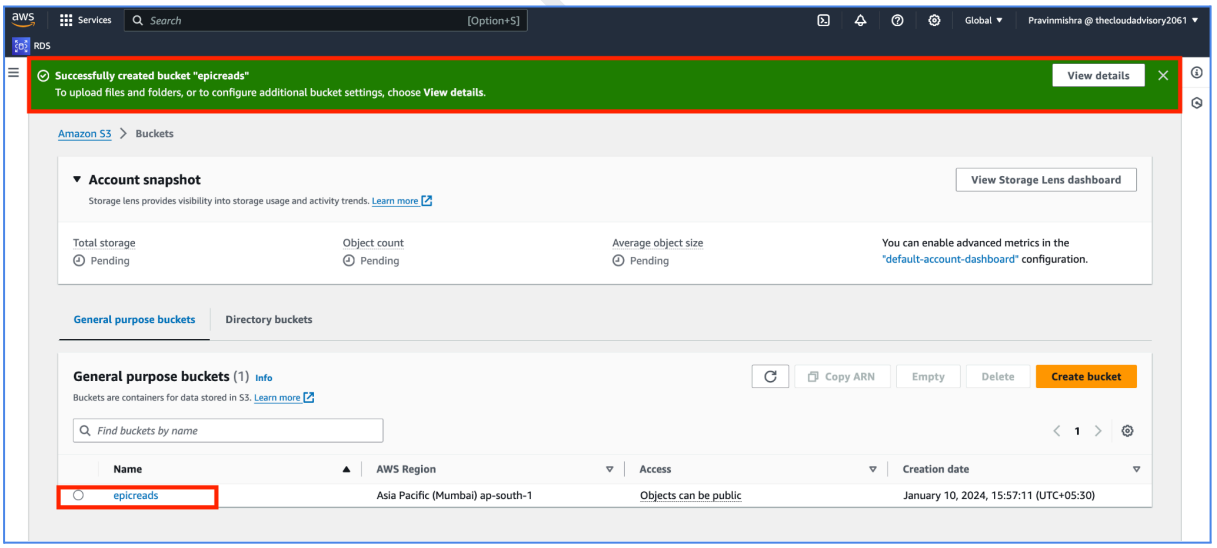


Figure 10.6: EpicReads S3 bucket

STEP 2: Upload Object on S3 bucket

The bucket is all set, as you are going to deploy a static website, upload the website's files - including HTML, CSS, JavaScript, and media files - to the designated *epicreads* S3 bucket configured for website hosting.

Click on the S3 bucket name *'epicread' (Figure 10.7: Click on 'epicreads')*

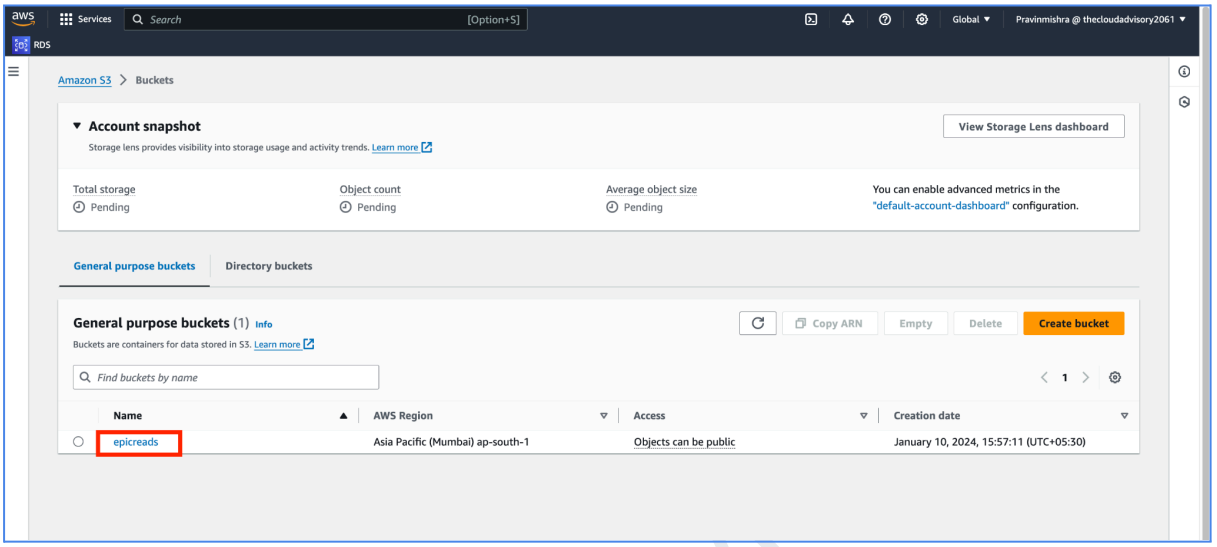


Figure 10.7: Click on 'epicreads'

Choose the *'Upload'* option *(Figure 10.8: Choose Upload)*

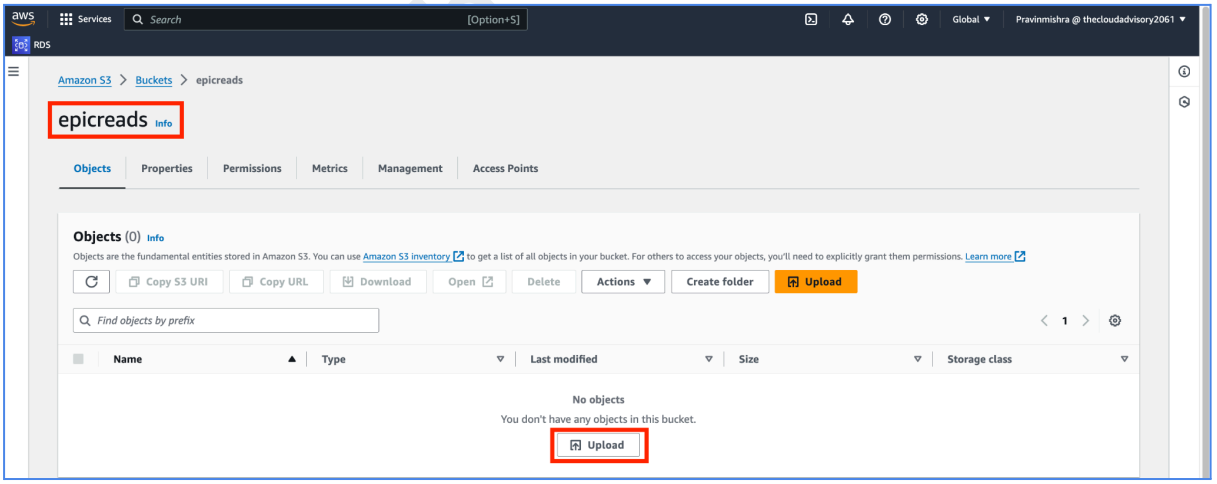


Figure 10.8: Choose Upload

Click on Add Folder and locate your files that you have to upload to S3 bucket.*(Figure 10.9: Add Folder)*

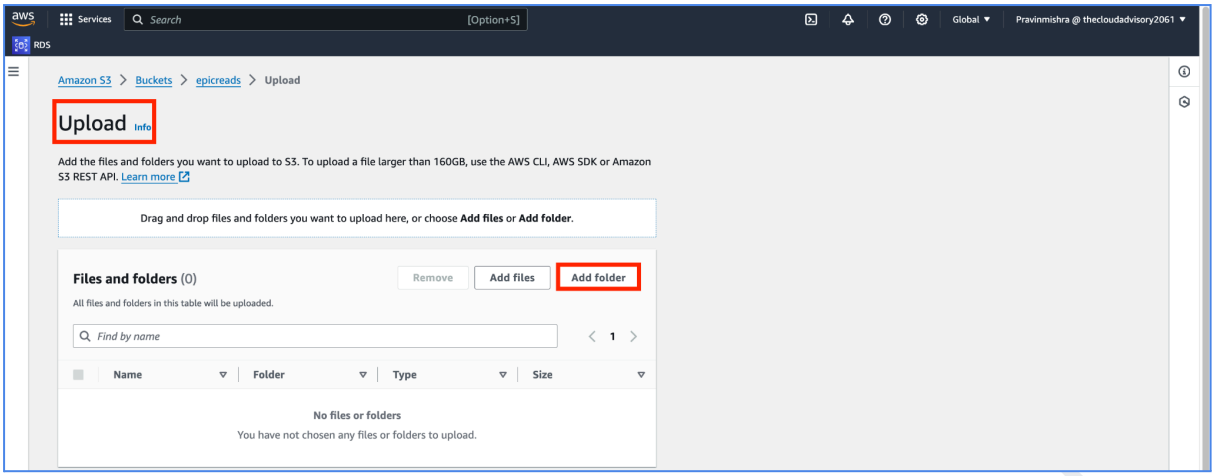


Figure 10.9: Add Folder

Click on the folder and click on the *Upload* button. Once again click on *Upload (Figure 10.10: Upload epicreads Images)*

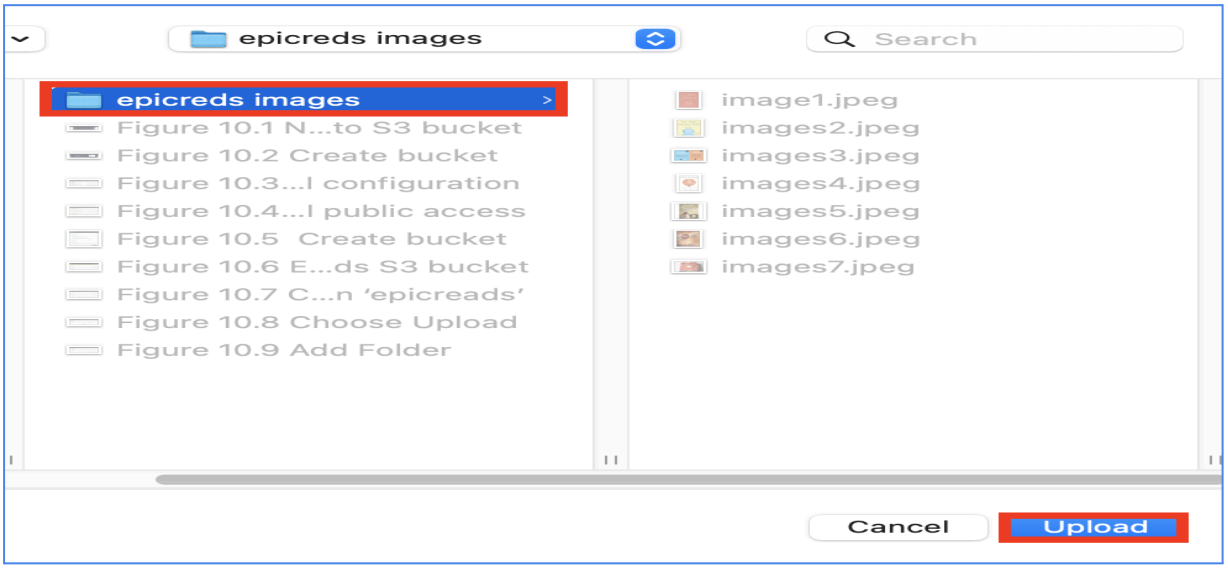


Figure 10.10: Upload epicreads Images

You can see that there are files to get uploaded. Click on *Upload* button now *(Figure 10.11: Upload objects)*

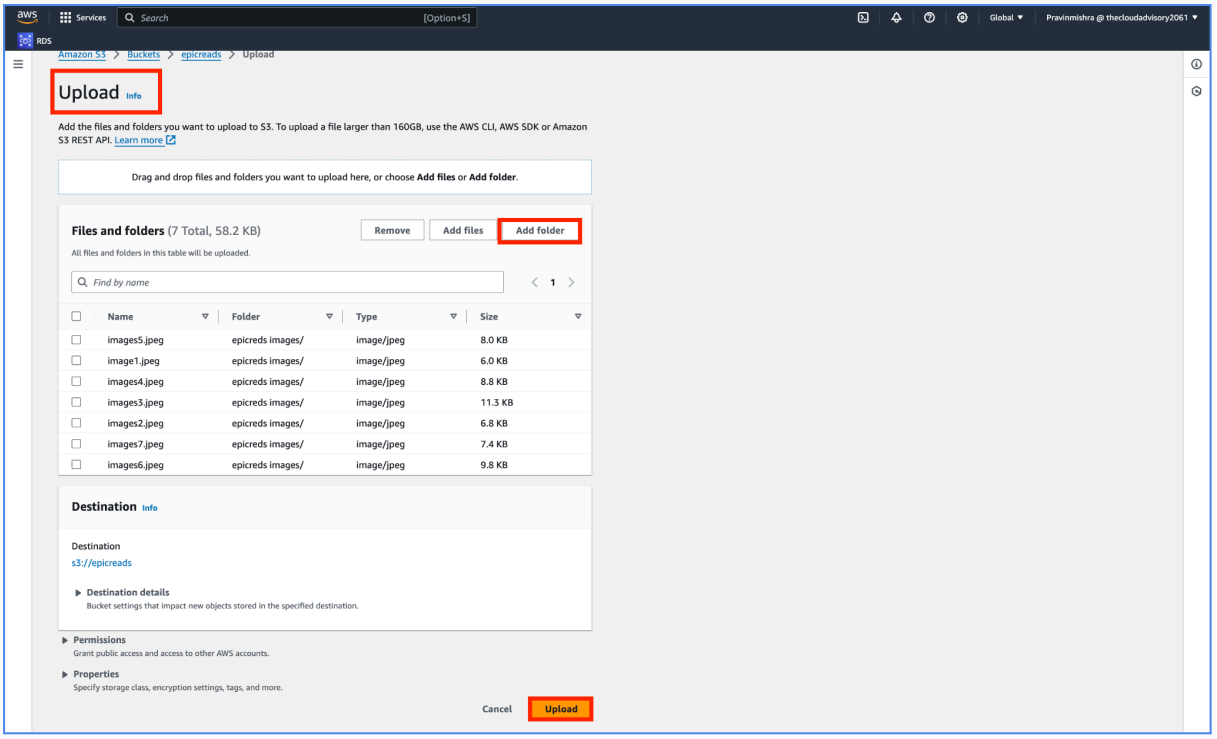


Figure 10.11: Upload objects

You will find the Success *message* on top and click on *Close* button *(Figure 10.12: Uploaded successfully)*

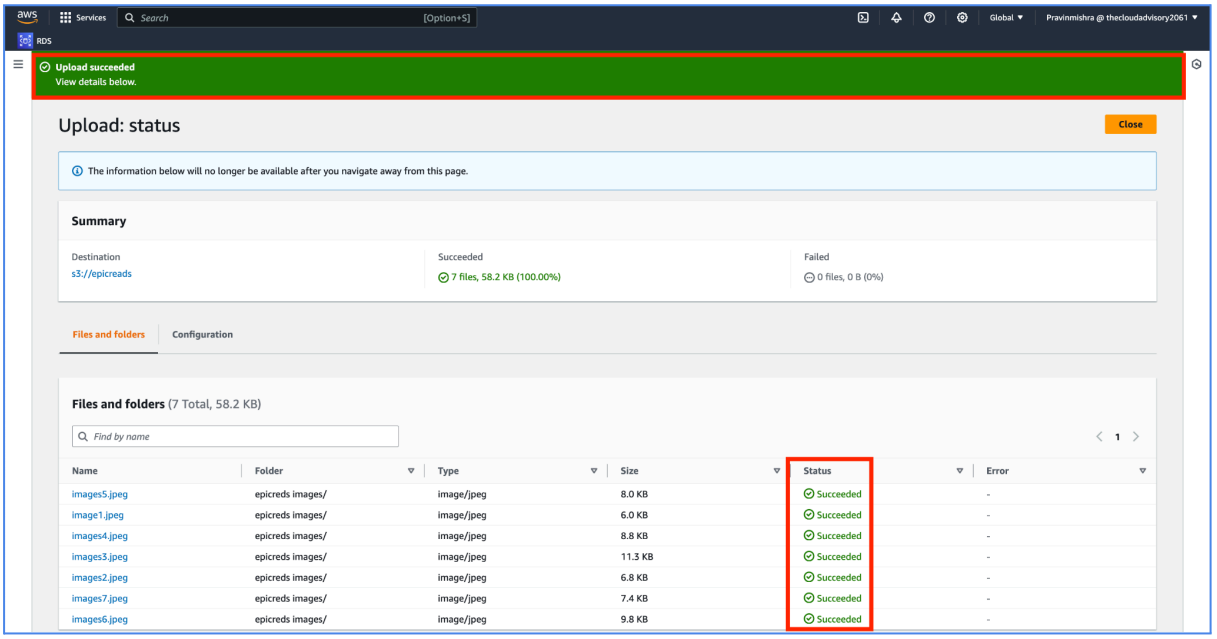


Figure 10.12: Uploaded successfully

STEP 3: Access the Object URL on S3 Bucket

When you save your files into S3 bucket, you will find list of objects that you have uploaded a while ago.

Click on any of the image files *(Figure 10.13: Image1)*

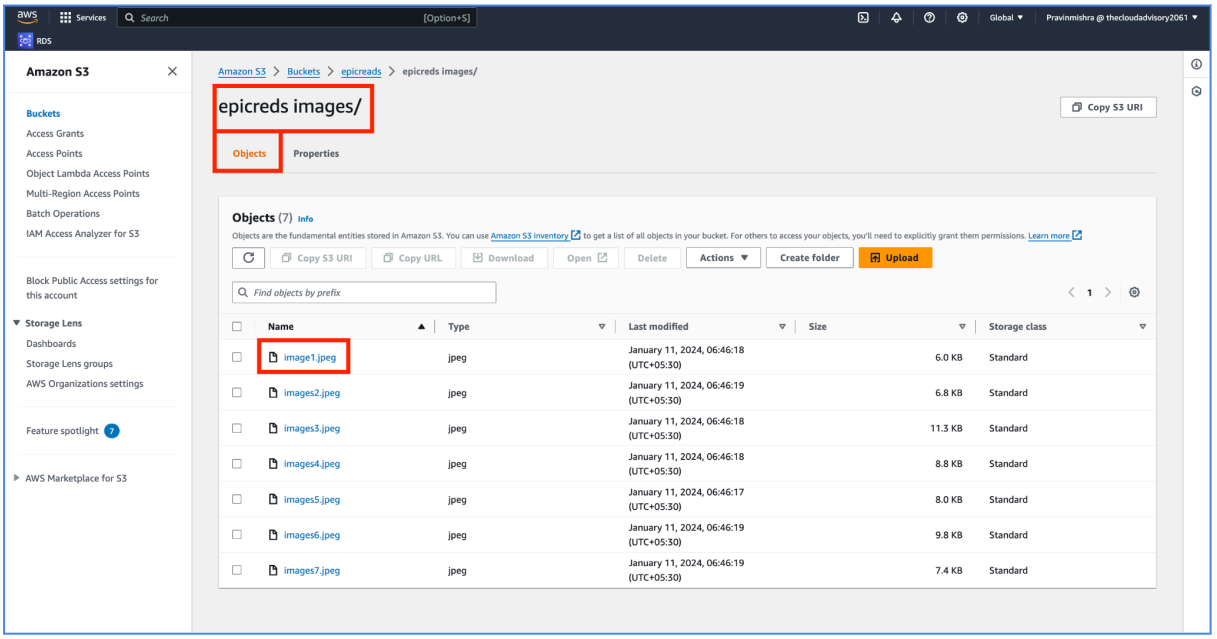


Figure 10.13: Image1

There is a unique *URI* created for each object, which includes the bucket name as the domain.

Now click on the *Object URL* to access the Website on the web browser *(Figure 10.14:Object URL)*

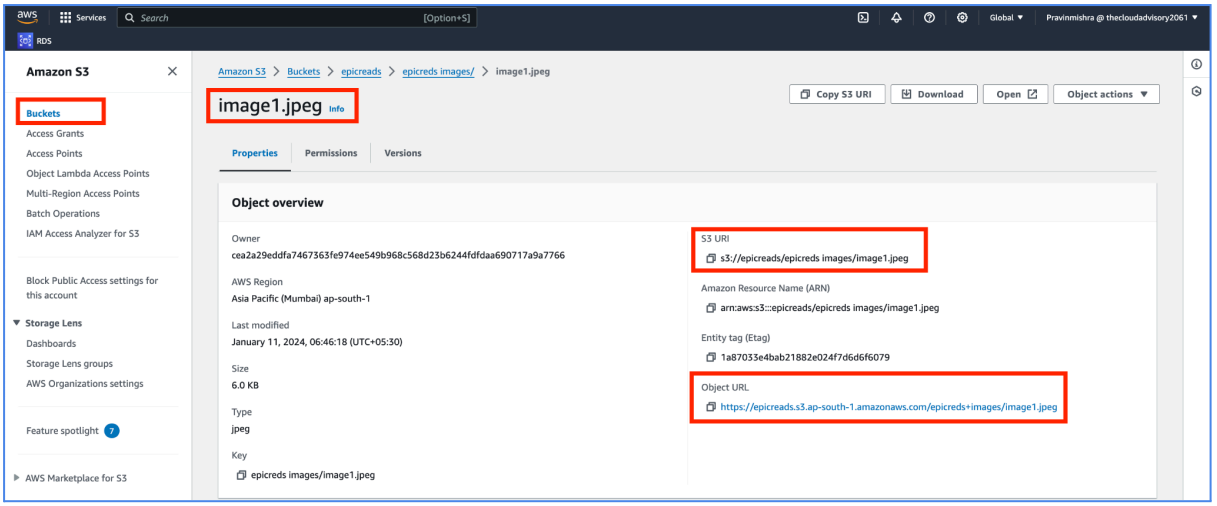


Figure 10.14:Object URL

By default, the permissions on an object are private *(Figure 10.15: Error page)*

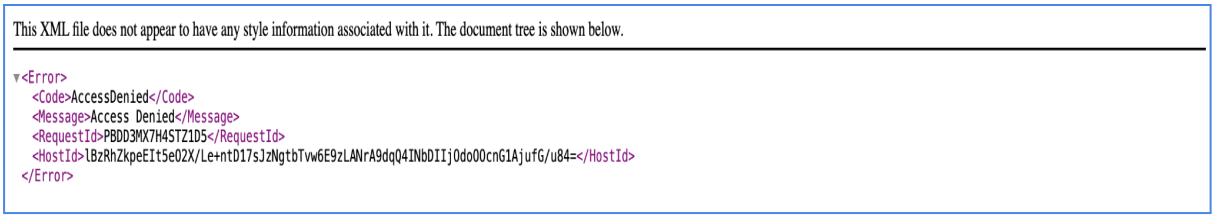


Figure 10.15: Error page

Note: You can set up access control policies to grant permissions to others. You can create a public object using individual *Actions*.

But this is not often necessary compared to enabling the S3 bucket. When an object is uploaded to the bucket your object is automatically accessible publicly.

STEP 4: Enable S3 Bucket on Public Access

Get back to *Buckets*, click on *'epicreads'* bucket name.

Select *Bucket Policy* under *Permissions* tab

Use the following S3 policy below on the editor area and then enter your bucket name '*epicreads'* for Resource key and click on Save changes *(Figure 10.16: epicreads Permission)*

💻 **Command:**

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Public S3 Bucket",
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::epicreads/*"
      }
   ]
}
```

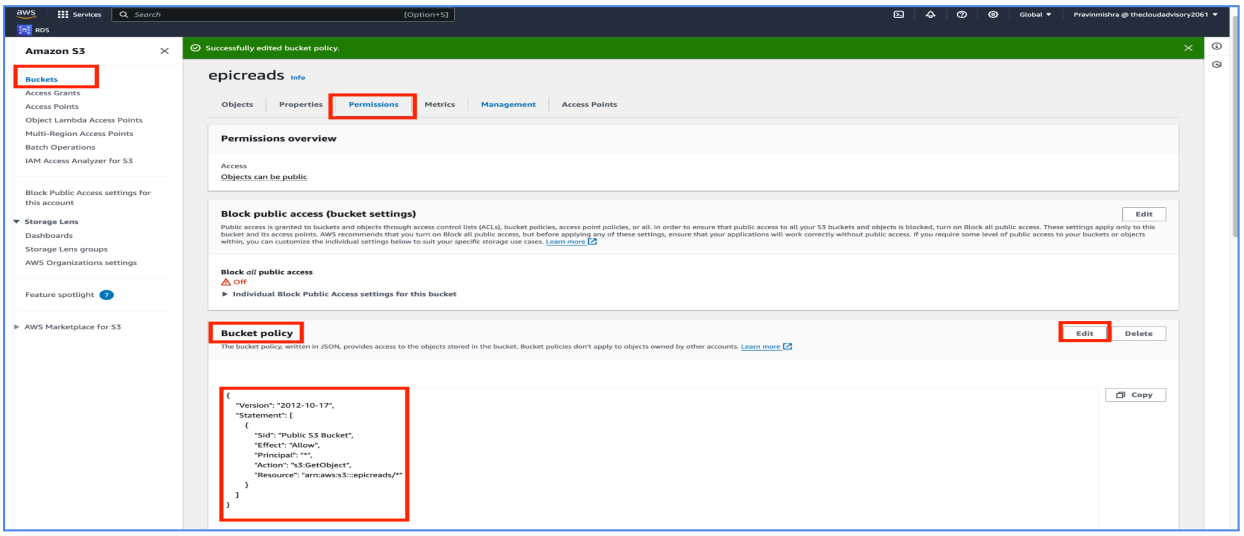


Figure 10.16: epicreads Permission

STEP 5: Verify the Object URL for Public Access

Now click on the *Objects*, click on *epicread images/*, Select any of the images *(Figure 10.17: Objects of epicreads)*

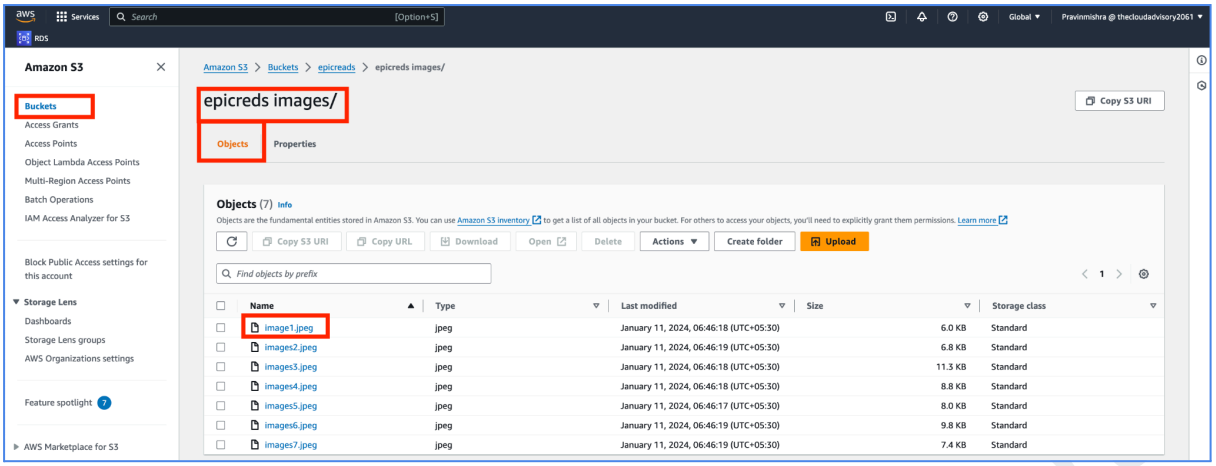


Figure 10.17: Objects of epicreads

You will find the *Object URL* link. Now click on the *Object URL* to access the Website on the web browser *(Figure 10.18: image1 URL)*

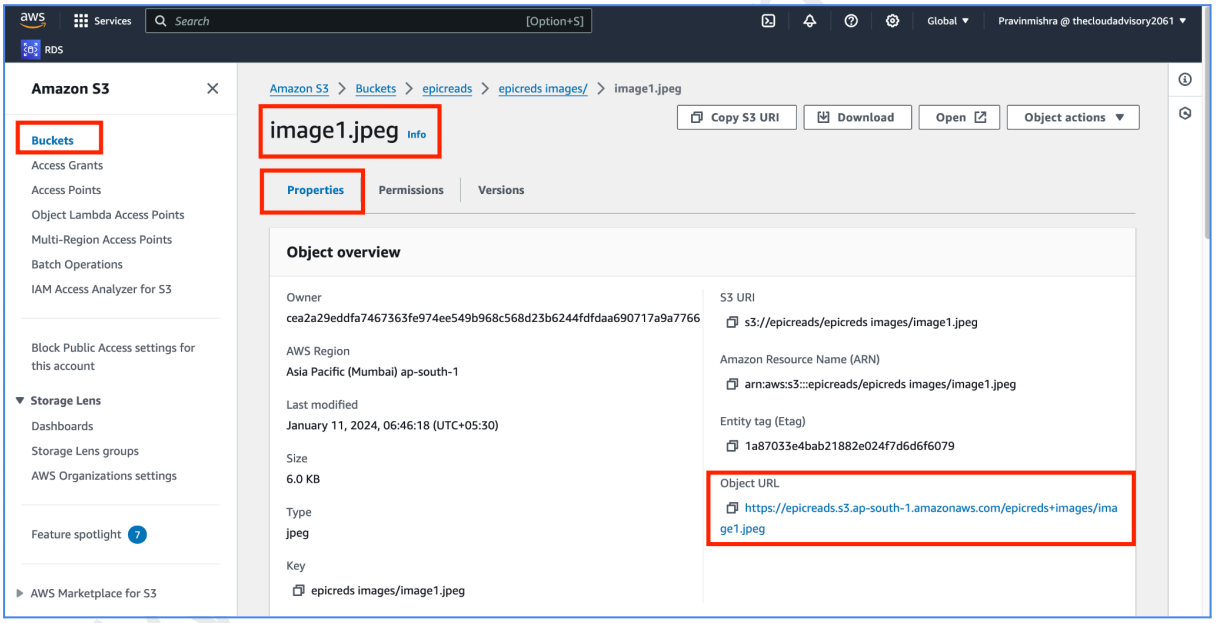


Figure 10.18: image1 URL

Well you will be able to see the image1 being publicly accessible *(Figure 10.19: image1)*

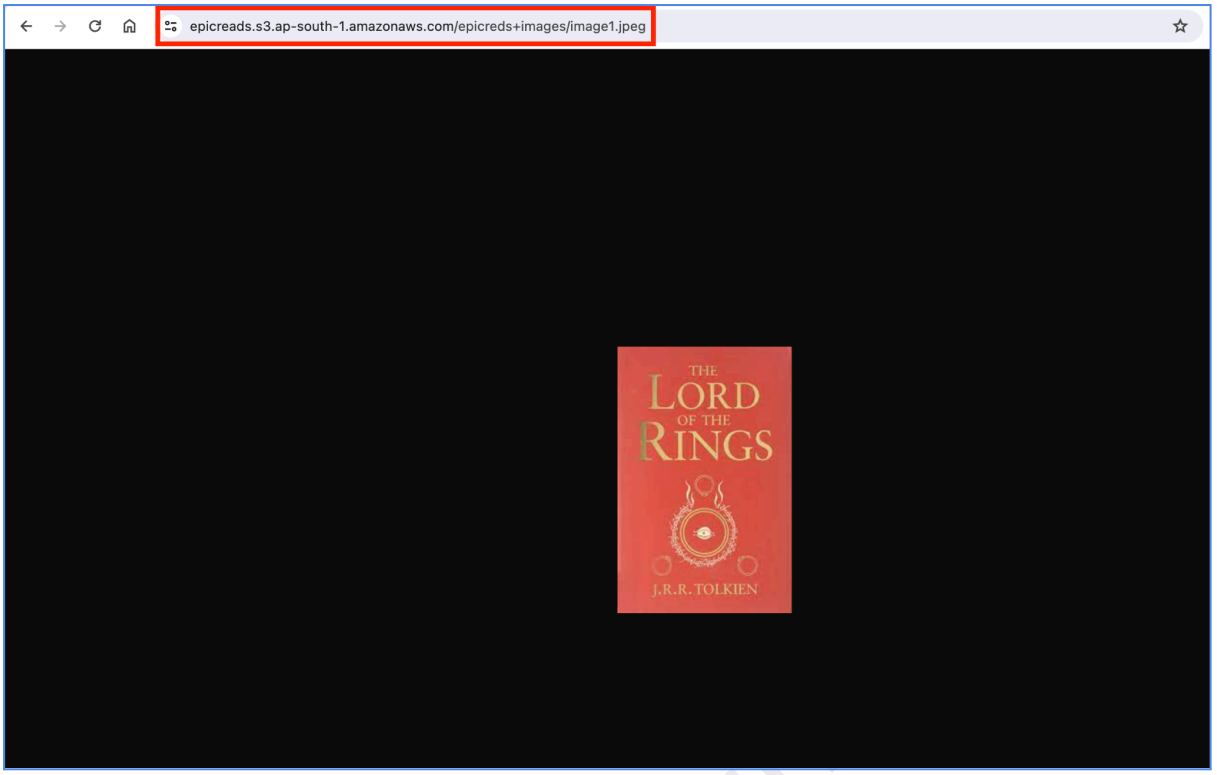


Figure 10.19: image1

So here ends *TASK 1*. Let's see how *AWS CLI* can be used to *HOST* static website content

Task 2. Upload and verify that the files are correctly uploaded using AWS CLI

STEP 6: Install and configure the AWS CLI

ℹ️ **Learn More:** Well explained video to install and configure AWS CLI is already there in the

Module 2. Getting Started with AWS -

Project 4: Installing and Verifying AWS CLI for EpicReads

Following the above project you are all set to do *TASK 2.*

STEP 7: Download website template for EpicReads

Click on the link *TemplateMo* to download free website templates.

Scrolling down, choose any one of the templates for EpicReads and click on it. *(Figure 10.20: Website template)*

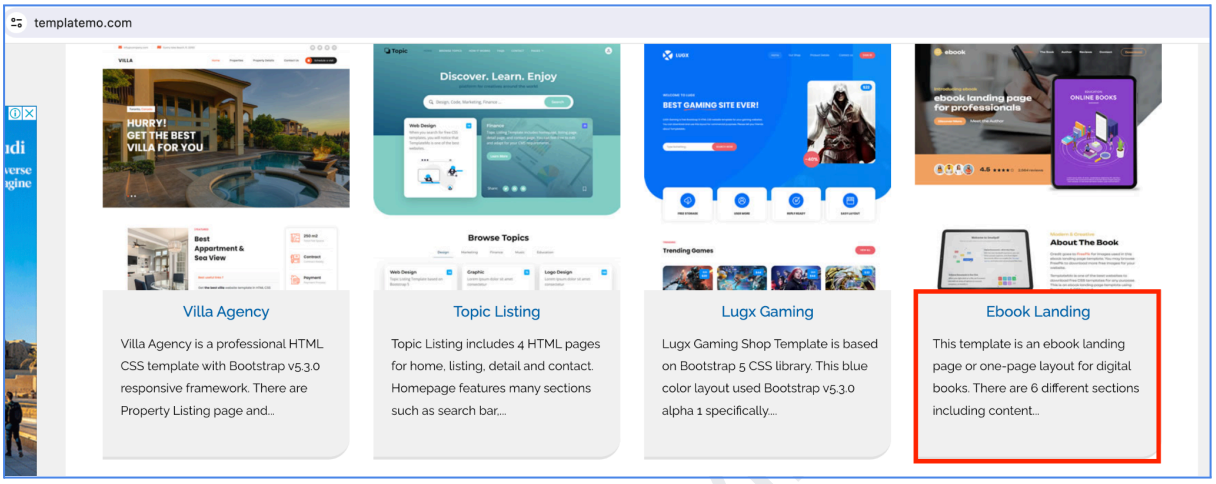


Figure 10.20: Website template

Scroll down to the bottom of the page and click on the Download button *(Figure 10.21: Website templateTemplate Download)*

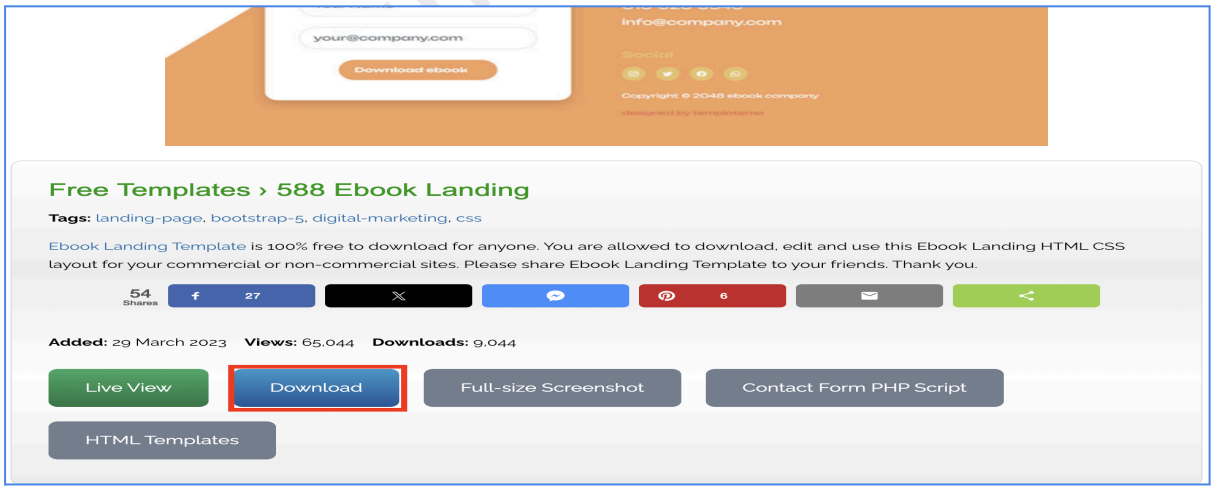


Figure 10.21: Website templateTemplate Download

STEP 8: Locate the downloaded file in your terminal

Open your terminal, check for the downloaded file with command given below *(Figure 10.22: Locate the downloaded file)*

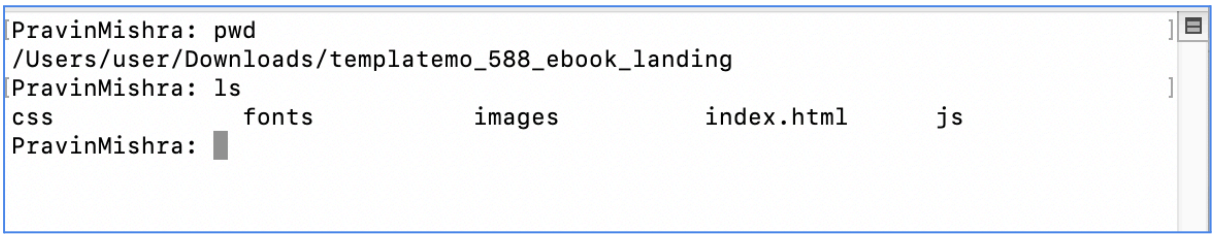


Figure 10.22: Locate the downloaded file

STEP 9: Create Amazon S3 bucket

In your terminal use the command below to see the list of buckets you have created in S3

💻 **Command:** *aws s3 ls*

✅ **Expected Output:** It would give list of buckets created else it would go to the next command line.

Now use the below command to create the command

💻 **Command:** *aws s3 mb s3://epicreads*

📓 **Note:** Refer the *link* to know the guidelines for naming the bucket

✅ **Expected Output:** *make_bucket: epicreads (Figure 10.23: Create bucket)*

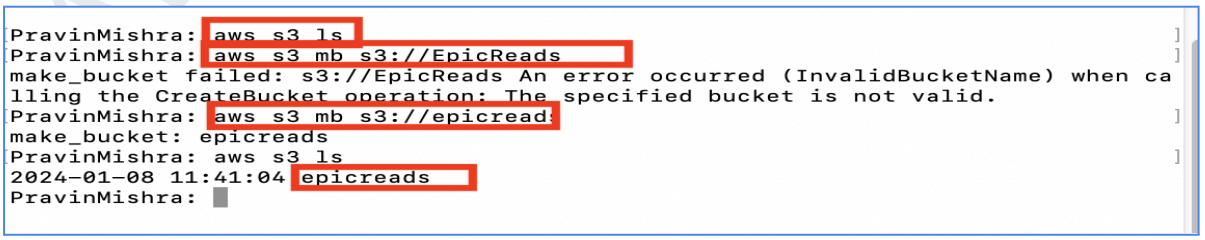


Figure 10.23: Create bucket

STEP 10: Edit static website hosting

Navigate back to S3 bucket from your console *(Figure 10.24: Navigate to S3 bucket)*

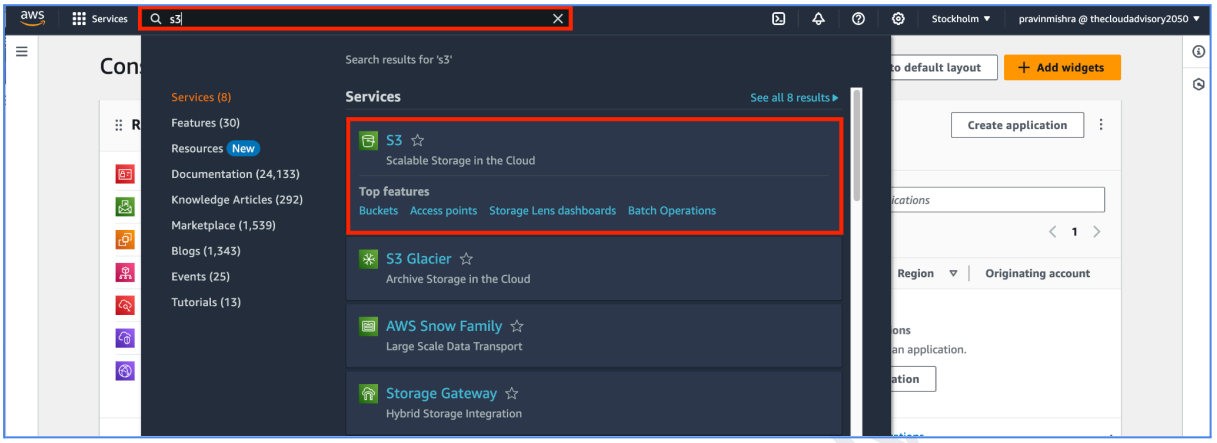


Figure 10.24: Navigate to S3 bucket

You will find the bucket that has been created, click on the bucket name *epicread (Figure 10.25: epicreads bucket)*

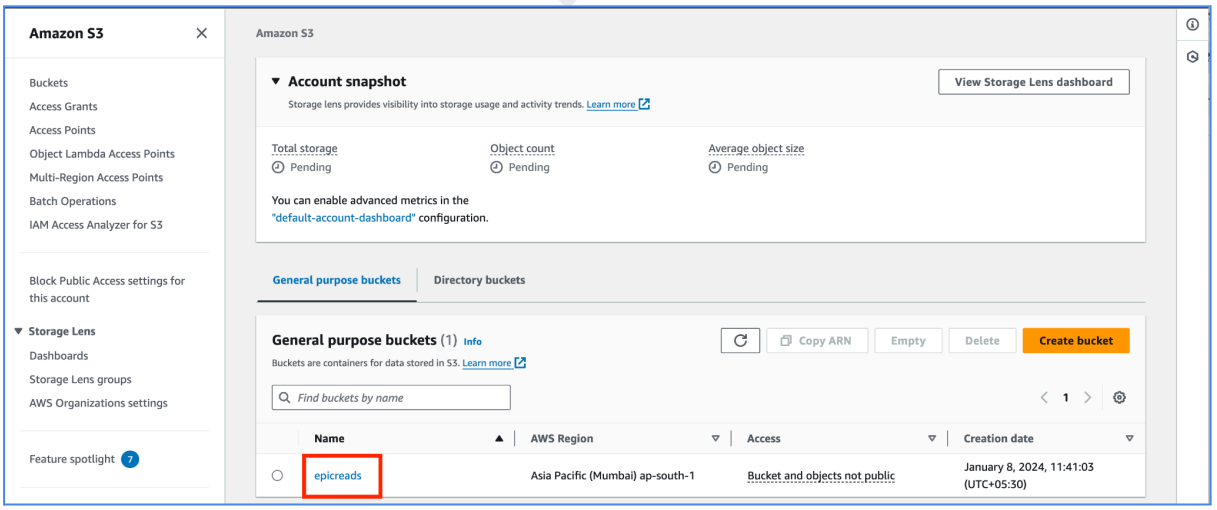


Figure 10.25: epicreads bucket

As you click you will find there is nothing in the *Objects'* section *(Figure 10.26: EpicReads Objects)*.

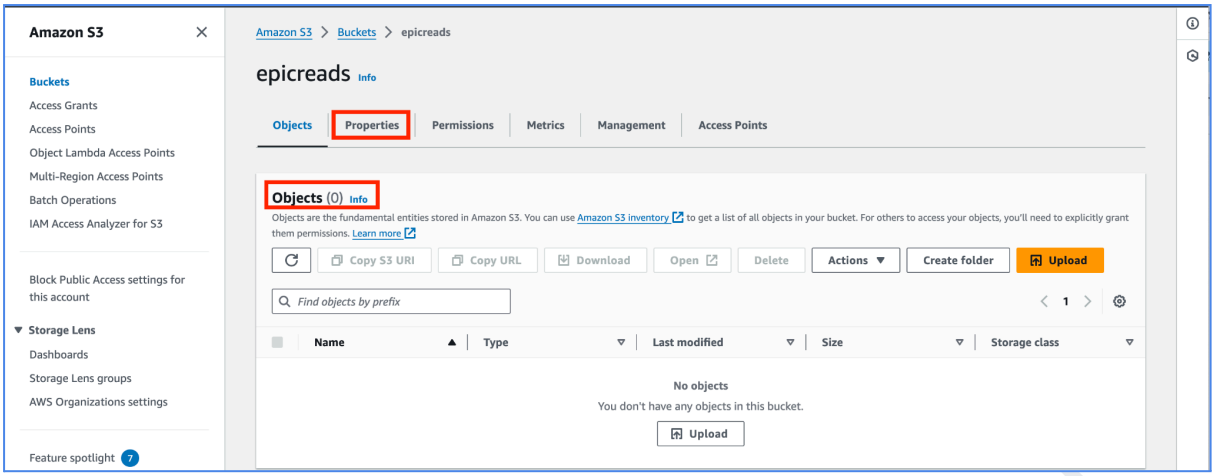


Figure 10.26: EpicReads Objects

Choose *Properties* and scroll down to the option *Static website hosting* and click on *Edit* button. *(Figure 10.27: Static website hosting)*

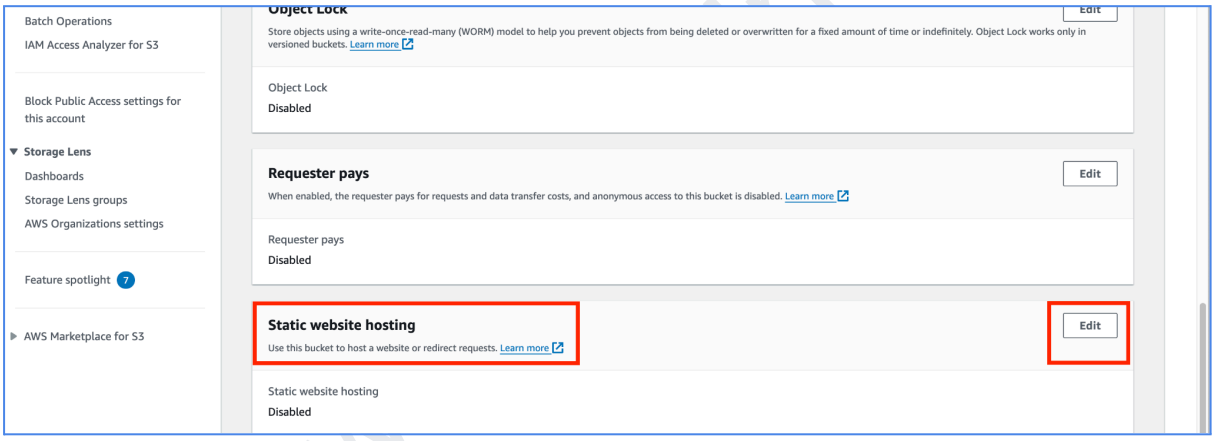


Figure 10.27: Static website hosting

Click on *Enable* radio button, choose *Host a static website* for the option *Hosting Type.*

To specify the home or default page of the website, copy paste '*index.html*' in the download folder from the terminal to the index document option in *Properties (Figure 10.28: index.html)*

```
/Users/user/Downloads/templatemo_588_ebook_landing
[PravinMishra: aws s3 ls
2024-01-08 11:41:04 epicreads
[PravinMishra: ls
aws             css             images          js
awscliv2.zip    fonts           index.html
PravinMishra:
```

Figure 10.28: index.html

When the webpage faces an error, give *404.html* to display the error page and click on *Save changes* button *(Figure 10.29: Edit static web hosting)*

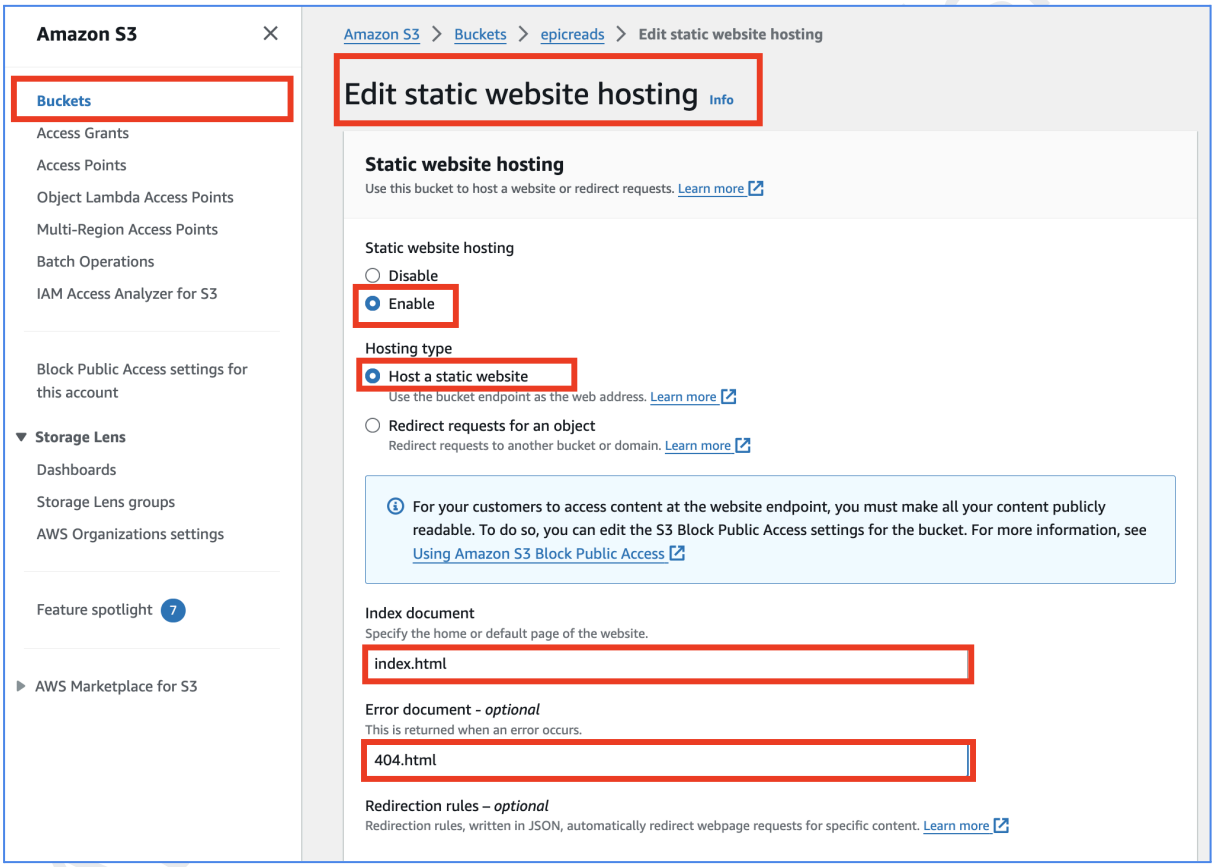


Figure 10.29: Edit static web hosting

You will find the success message on top. Scrolling down to the *Static website hosting* section you will find the *link* of you bucket, click on it to open the *link (Figure 10.30: Static website link)*

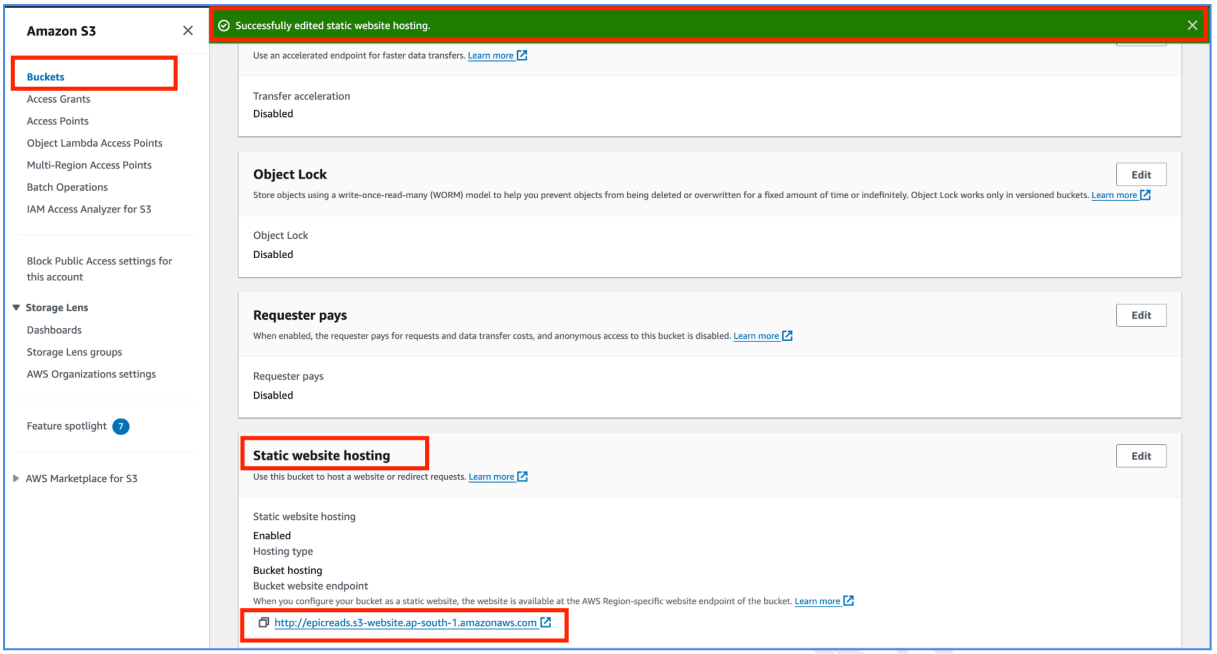


Figure 10.30: Static website link

STEP 11: Edit bucket settings

Now you will find webpage *403 Forbidden (Figure 10.31: 403 Forbidden)*

Figure 10.31: 403 Forbidden

📓 **Note:** By default you create any bucket, it is not publicly accessible.

To make the bucket accessible, click on *Permissions* and you will find that *Block all public access* is *ON.* Click on *Edit* button *(Figure 10.32: Bucket settings)*

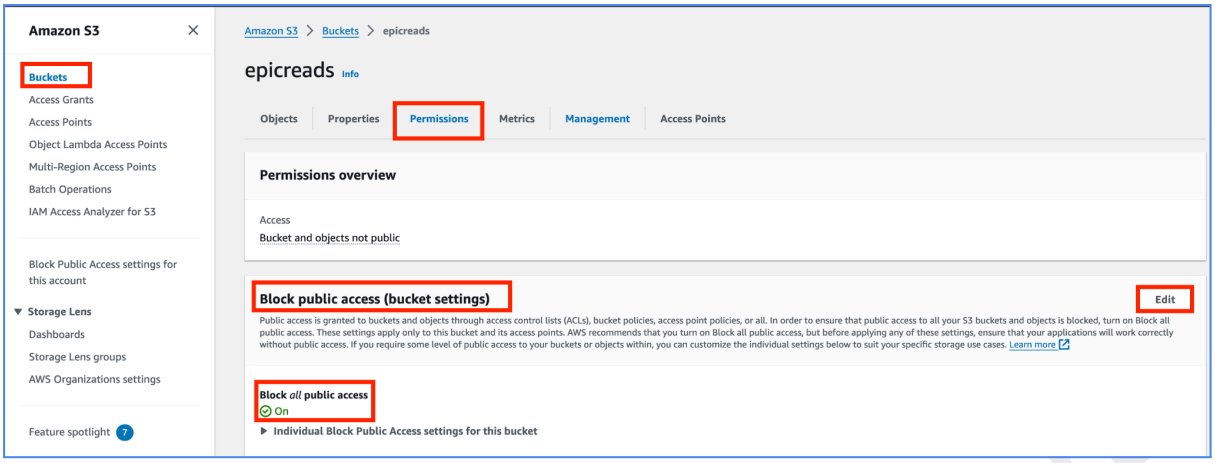


Figure 10.32: Bucket settings

Uncheck the checkbox *Block all public access,* click on *Save changes*,type *confirm* and click on *Confirm* button *(Figure 10.33: Unblock public access)*

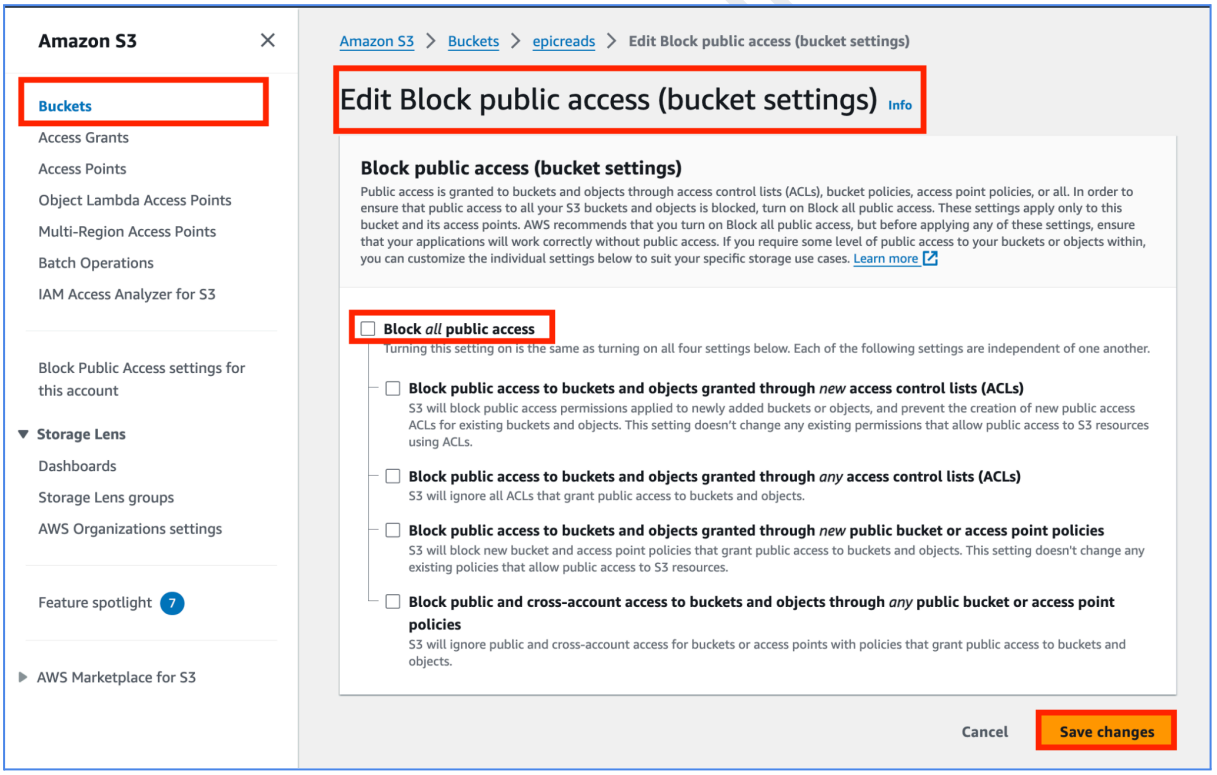


Figure 9.19: Unblock public access

Clicking once again the bucket link you will the same 403 Forbidden page *(Figure 10.33: Bucket link)*

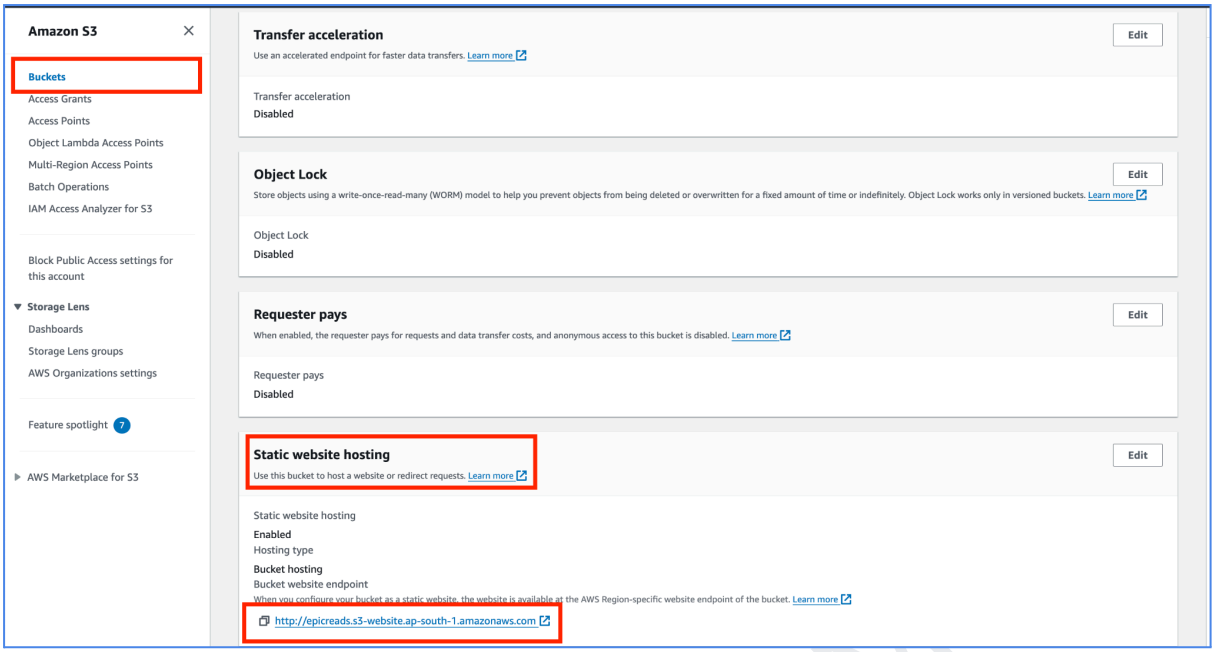


Figure 10.33: Bucket link

Once again you will find the same *403 Forbidden* page *(Figure 10.34: Forbidden page)*

Figure 10.34: Forbidden page

📓 **Note:** The best part of S3 is that it gives multiple level of blocking to make your bucket public both bucket level and account level.

STEP 12: Add Bucket policy

Here let's edit the bucket policy so that the bucket is accessible. Under *Permissions*, scroll down to the *Bucket policy* and click on the *Edit* button *(Figure 10.35: Edit Bucket policy)*

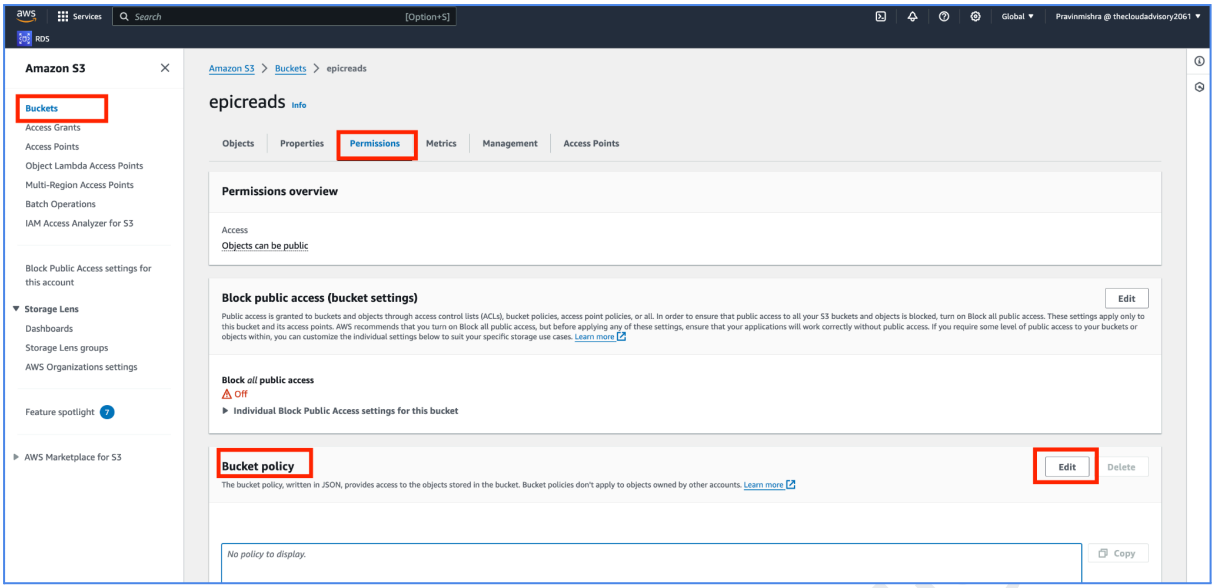


Figure 10.35: Edit Bucket policy

Copy paste the below script and replace the *Bucket-Name* with '*epicreads*' and click on *Save changes (Figure 10.36: Bucket Policy)*

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PublicReadGetObject",
         "Effect": "Allow",
         "Principal": "*",
         "Action": [
            "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::Bucket-Name/*"
         ]
      }
   ]
}
```

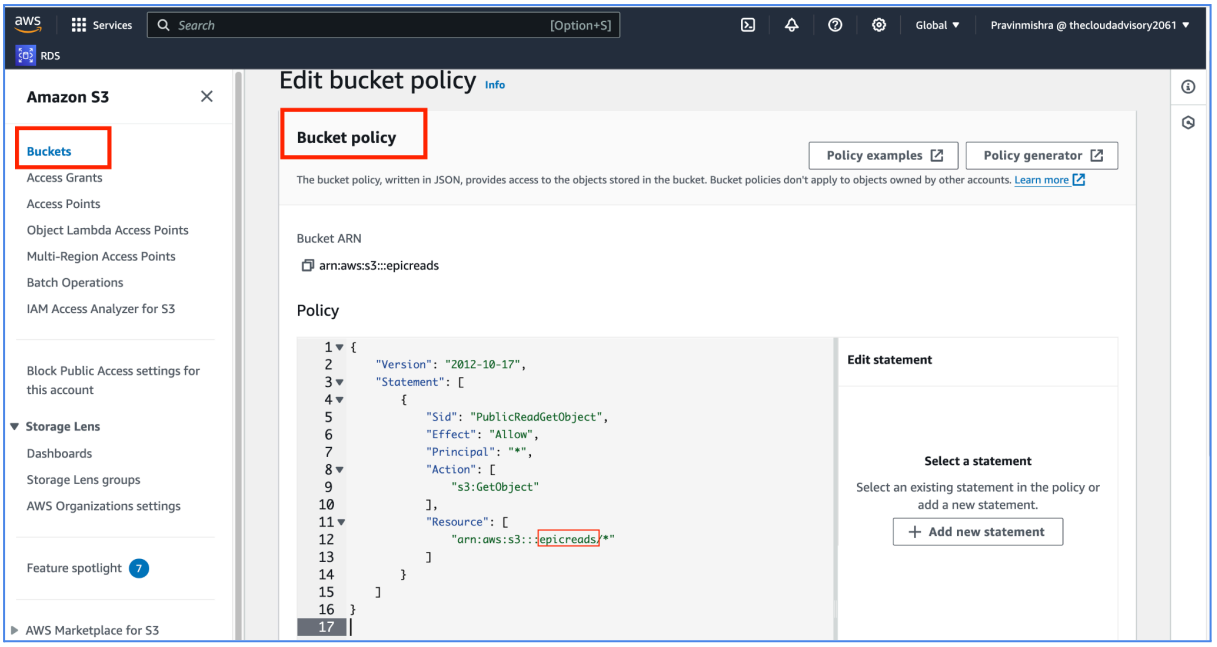


Figure 10.36: Bucket Policy

💡 **Consider:** If you notice the *Action: "s3:GetObject"* allows the action of retrieving (reading) objects from the bucket, means the users can only see, read the data and can never ever attempt to delete the data.

Now you will find that your S3 bucket is *Publicly accessible (Figure 10.37: Publicly accessible)*

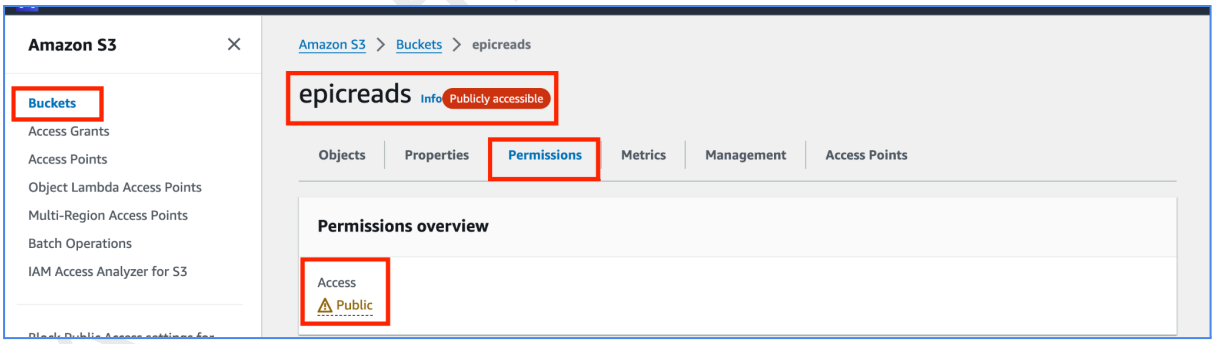


Figure 10.37: Publicly accessible

Now Click on Properties, scroll down and click on the S3 bucket *link* that is in *Static website hosting* section *(Figure 10.38: S3 bucket link)*

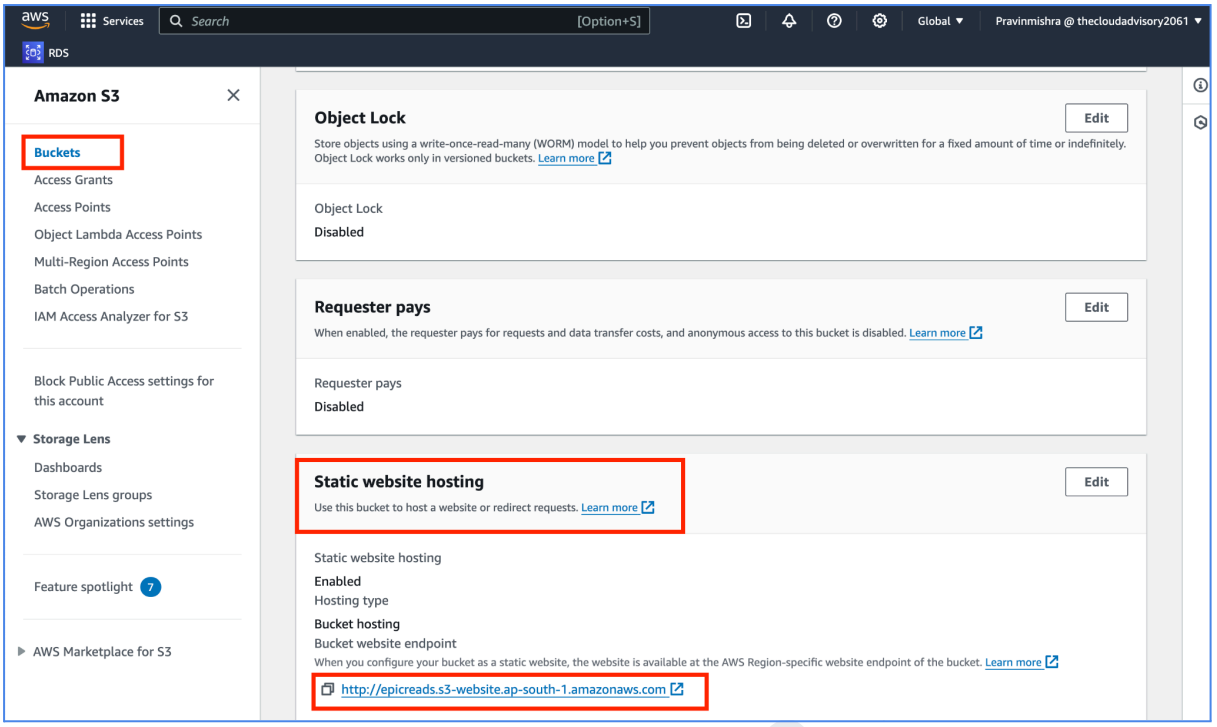


Figure 10.38: S3 bucket link

The link displays **404 Not Found**, with message *Code: NoSuchKey* and the key mentioned there was *Key: index.html*

💡 **Consider:** Navigating back to the *epicreads* S3 bucket you will find there is no object that has to be uploaded. Well that is the reason it displays *404 Not Found (Figure 10.39: 404 Not Found)*

Figure 10.39: 404 Not Found

Editor will be opened where you can customise your page with the message as given below and type *:wq* to save the editor and enter. *(Figure 10.40: vi editor)*

Hi I am an error page!

Sorry, I will be back soon!

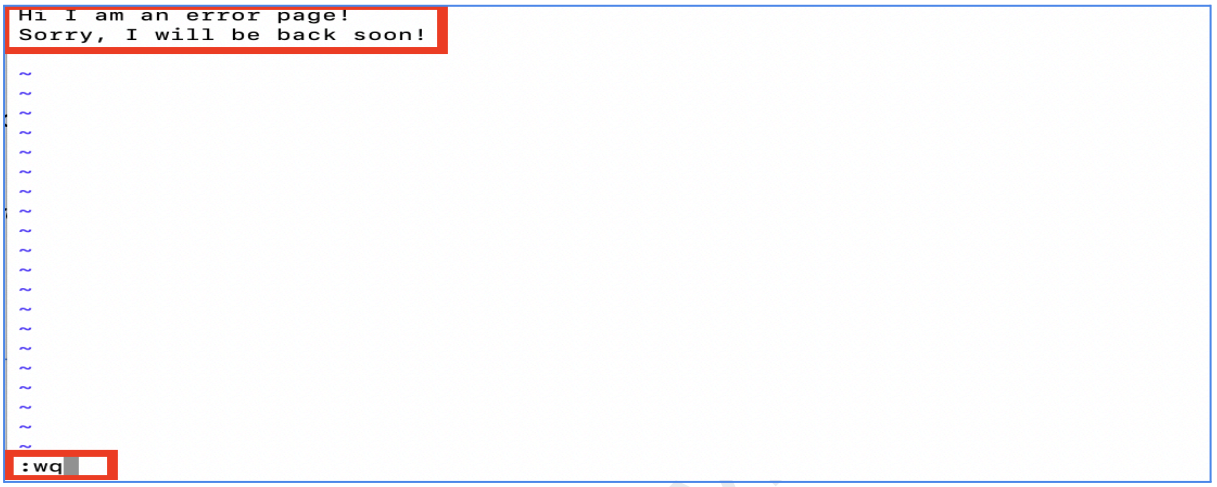


Figure 10.40: vi editor

Use the command given below to list out the files

💻 **Command:** *ls*

You will find out 404.html that we created. Now copy this *404.html file* to the *epicreads* S3 bucket using the command given below *(Figure 10.41: Copy error file to S3)*

💻 **Command:** *aws s3 cp 404.html s3://epicreads*

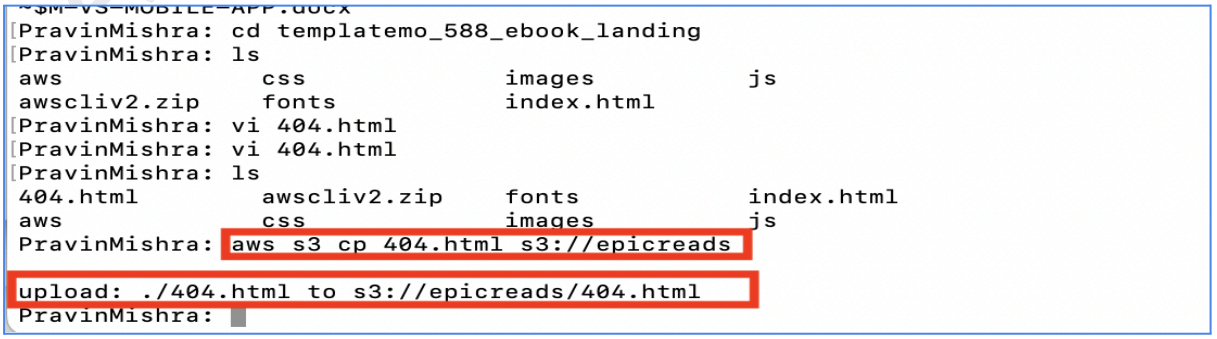


Figure 10.41: Copy error file to S3

Refreshing the browser you will find the customised message instead of 404 Not found *(Figure 10.42: error page)*

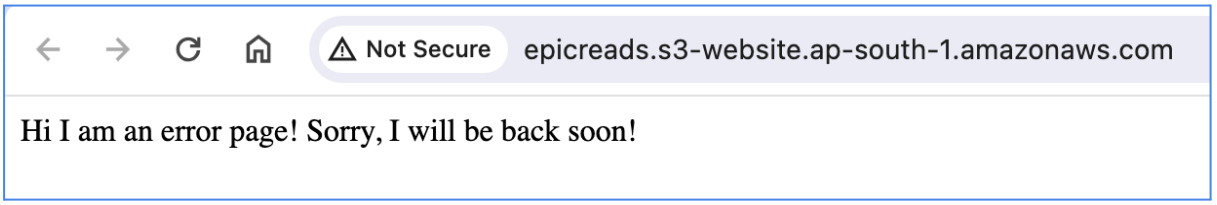


Figure 10.42: error page

💡 **Consider:** In the other way to understand, this link displays error page as there is no other files available in the bucket for the user to access.

STEP 13: Sync the files to S3 bucket

With no delay let's upload the files to the S3 Bucket by using the command that is given below
💻 **Command:** *aws s3 sync . s3://epicreads*

📓 **Note:**

sync: This is the command used for synchronising directories and objects to Amazon S3.

. : Represents the source directory on your local machine. In this case, it's the current directory (where you are running the command).

s3://epicreads: Specifies the destination S3 bucket where you want to sync the files.

✅ **Expected Output:** Navigate back to the *epicreads* S3 bucket in the AWS Management Console and you can see all the files have been uploaded *(Figure 10.43: Objects in S3)*

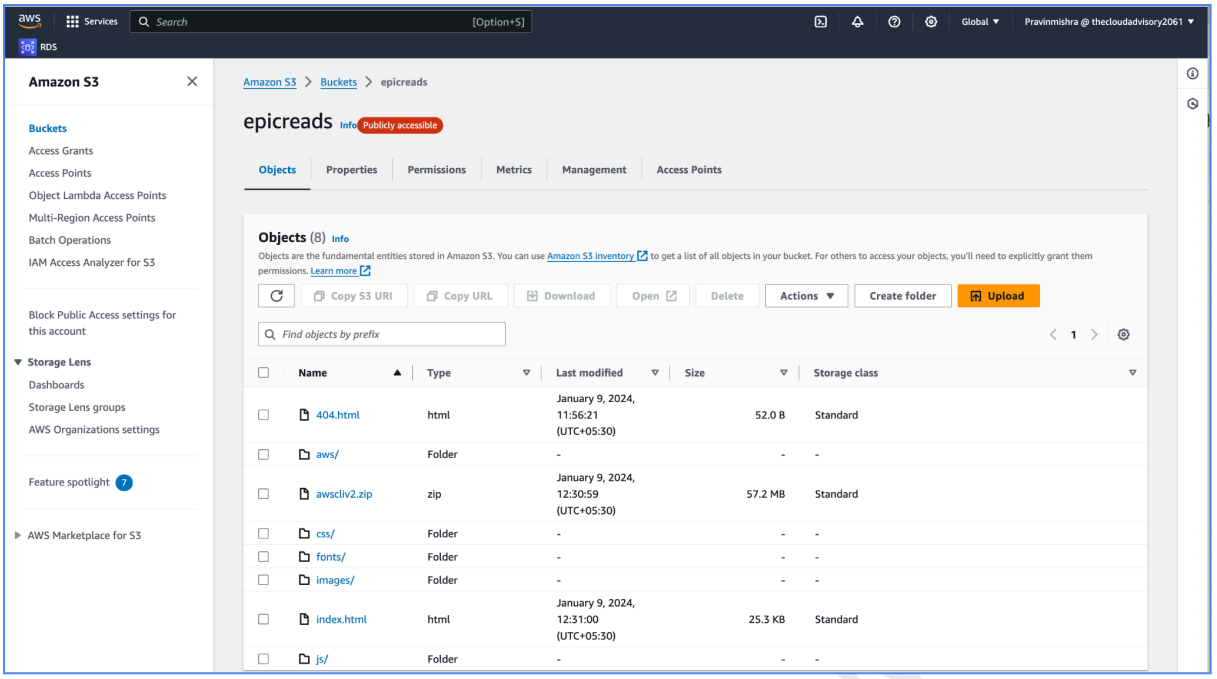


Figure 10.43: Objects in S3

As the files have been uploaded, all set to give a test run now!

STEP 14: Refresh the browser

Now click refresh in your browser and you should see the Static website is ready *(Figure 10.44: EpicReads Static website)*

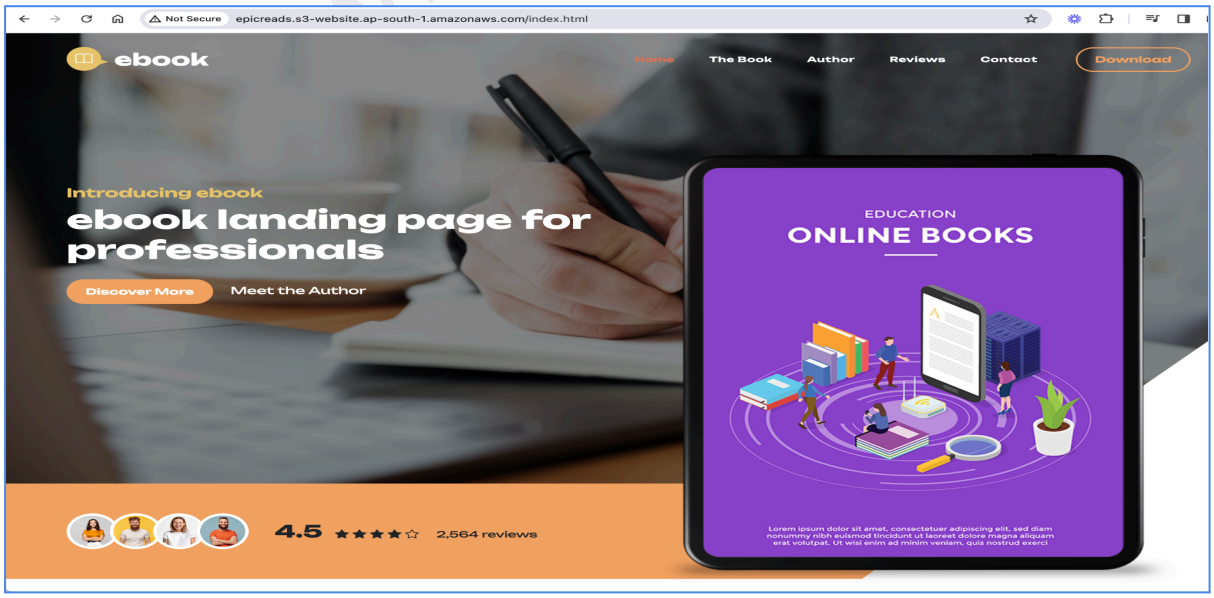


Figure 10.44: EpicReads Static website

Thus ***Task 2*** ends here!!!

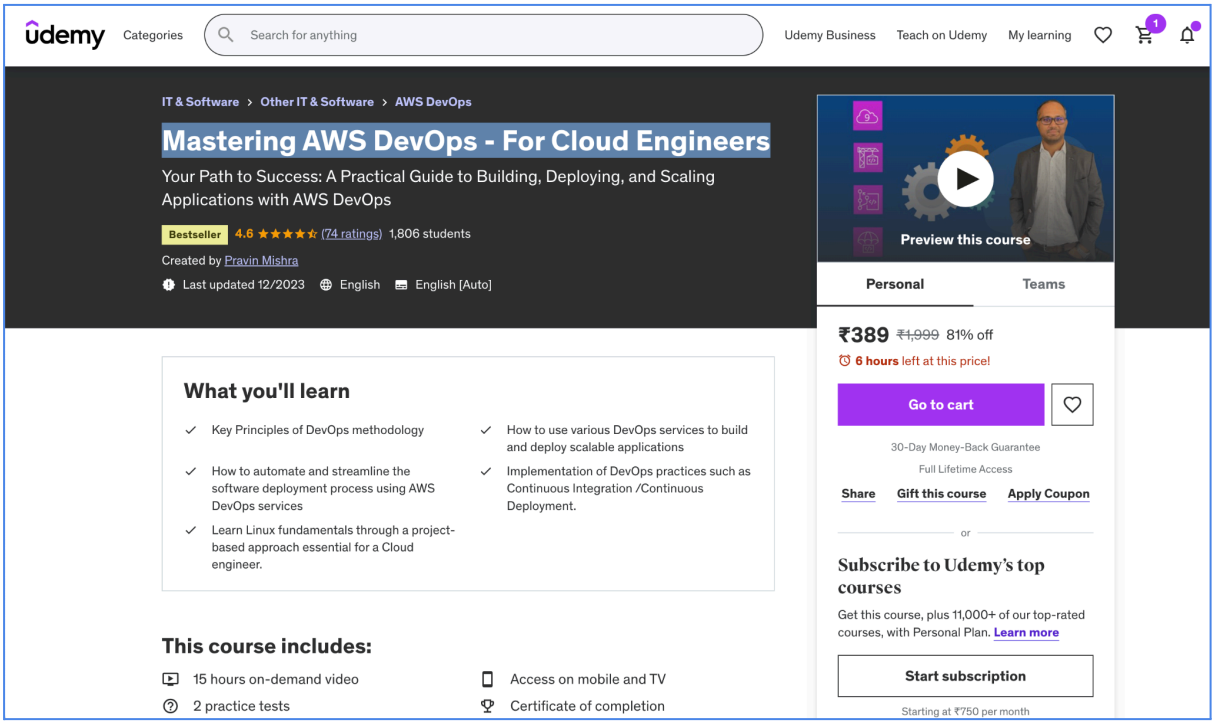
Task 3. Setting Up Automated CI/CD

Continuous Integration (CI) and Continuous Deployment (CD) pipelines have become indispensable tools for automating the process, ensuring code quality, and accelerating development cycles.

ℹ️ **Learn More:** Setting up an automated CI/CD pipeline to create an AWS S3 bucket involves using various AWS services such as AWS CodePipeline, AWS CodeBuild, and AWS CloudFormation.

DevOps is not merely a set of tools; it's a cultural shift that advances the collaboration between development and operations teams. As this topic is pretty deep to dive-in I have a suggestion that you can definitely have a look on the course by clicking the link below.

Mastering AWS DevOps - For Cloud Engineers



Hands-On with AWS CodeDeploy: Setup,Demonstration and Security Module 14 topic explains the way to achieve this task.

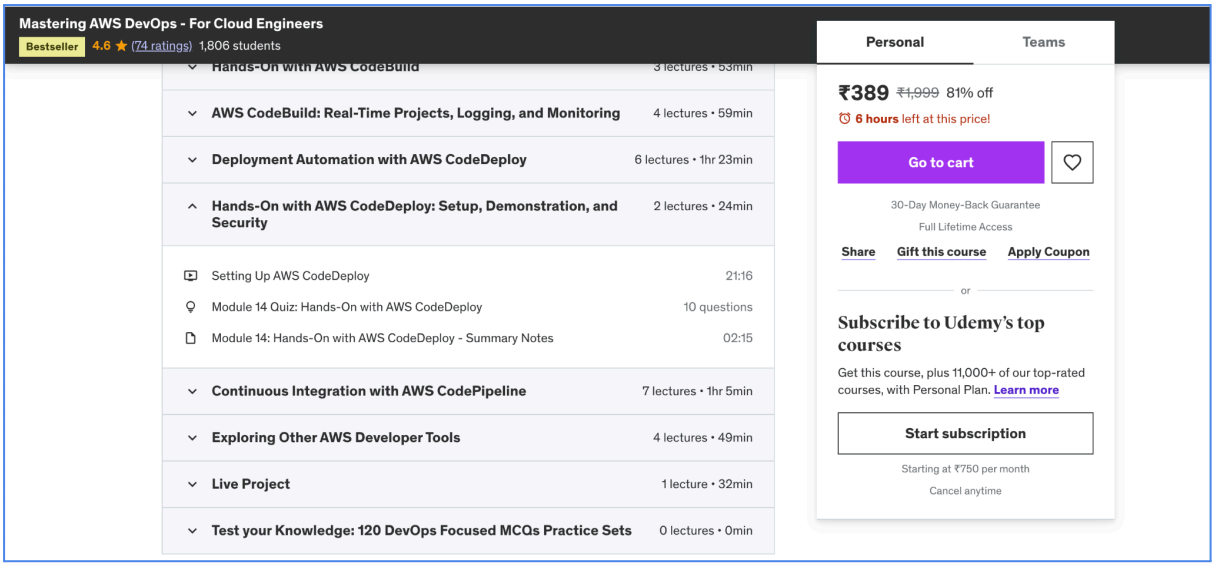


Figure 10.44: AWS CodeDeploy

COMMON ERRORS - Explanation

Before starting to explain the common errors,I would like to suggest you try the General troubleshooting.

ℹ️ **Learn More:** Refer *here* for troubleshooting the common errors.

1) Check your AWS CLI command formatting

- Check your command for spelling and formatting errors.
- Confirm all quotes and escaping appropriate for your terminal is correct in your command.

- Generate an AWS CLI skeleton to confirm your command structure.
  ℹ️ **Learn More:** Check the link to know the details on AWS CLI skeleton
- For JSON, see the additional troubleshooting for JSON values. If you're having issues with your terminal processing JSON formatting, we suggest skipping past the terminal's quoting rules by using Blobs to pass JSON data directly to the AWS CLI.
  ℹ️ **Learn More:** To know details on parameter-type-blob, check here

2) Confirm that you're running a recent version of the AWS CLI

Updated versions of the AWS CLI are released almost every business day. New AWS services, features, and parameters are introduced in those new versions of the AWS CLI. The only way to get access to those new services, features, or parameters is to upgrade to a version that was released after that element was first introduced.

ℹ️ **Learn More:** To install CLI recent version click *here*

3) Enable and review the AWS CLI command history logs

- You can enable the AWS CLI command history logs using the cli_history file setting. After enabling this setting, the AWS CLI records the history of aws commands.
- You can list your history using the aws history list command, and use the resulting command_ids in the aws history show command for details. For more information see aws history in the *AWS CLI reference guide*.

ℹ️ **Learn More:** To know more about *CLI history* and AWS history click on respective links.

4) Confirm that your AWS CLI is configured

Various errors can occur if your config and credentials files or your IAM user or role is not configured correctly. For more information on resolving errors with config and credentials files or your IAM user or roles, see Access denied errors and Invalid credentials and key errors.

ℹ️ **Learn More:** Check here for *Access denied errors* and Invalid credentials

5) I'm using the Amazon S3 static website feature but getting an Access Denied error

If you're trying to host a static website using Amazon S3, but you're getting an Access Denied error, check the following requirements:

- **Objects in the bucket must be publicly accessible**

S3 static website endpoint supports only publicly accessible content. To verify whether an object in your S3 bucket is publicly accessible, open the object's URL in a web browser. Or, you can run a cURL command on the URL.

💻 **Command:**
http://doc-example-bucket.s3-website-us-east-1.amazonaws.com/index.html

If an Access Denied error is returned by the web browser or cURL command, then the object isn't publicly accessible. To allow public read access to your S3 object, create a bucket policy that allows public read access for all objects in the bucket.

- **S3 bucket policy must allow access to the s3:GetObject action**

Review your bucket policy, and make sure that there aren't any deny statements that block public read access to the *s3:GetObject* action.

- **The AWS account that owns the bucket must also own the object**

To allow public read access to objects, the AWS account that owns the bucket must also own the objects. A bucket or object is owned by the account of the AWS Identity and Access Management (IAM) identity that created the bucket or object.

Note: The object-ownership requirement applies to public read access granted by a bucket policy. It doesn't apply to public read access granted by the object's access control list (ACL).

ℹ️ **Learn More:** To know more about this click here

- **Objects that are requested must exist in the S3 bucket**

If the user performing the request doesn’t have s3:ListBucket permissions, then the user gets an Access Denied error for missing objects.

You can run the head-object AWS CLI command to check if an object exists in the bucket.

Note: S3 object names are case-sensitive. If the request doesn't have a valid object name, then Amazon S3 will report that the object is missing.

If the object exists in the bucket, then the Access Denied error isn't masking a 404 Not Found error. Verify other configuration requirements to resolve the Access Denied error.

If the object doesn't exist in the bucket, then the Access Denied error is masking a 404 Not Found error. Resolve the issue related to the missing object.

- **Amazon S3 Block Public Access must be disabled on the bucket and account level**

Amazon S3 Block Public Access settings can apply to individual buckets or AWS accounts. Confirm that there aren't any *Amazon S3 Block Public Access settings* applied to either your S3 bucket or AWS account. These settings can override permissions that allow public read access.

ℹ️ **Learn More:** To know more click here

CONCLUSION:

In conclusion, mastering the art of uploading website files to an S3 bucket is a fundamental skill for any web developer or cloud engineer leveraging Amazon Web Services (AWS). Throughout this *Project 10*, we explored various methods to achieve this task, each offering its unique advantages

From the simplicity of the AWS Management Console to the flexibility of the AWS Command Line Interface (CLI) and the use of AWS CodePipeline and CodeBuild provides a more advanced and automated approach, ideal for continuous integration and deployment scenarios.

By understanding these methods as a learner I make sure that you can approach any of your projects based on the requirements, collaboration needs, and the level of automation desired

Happy Learning!! Cheers that you have completed Project 10 with great understanding.

See you in the next Project solving page!!