

Host a Website on Amazon S3

Project Overview

This project demonstrates how to host a static website using Amazon S3 (Simple Storage Service). I successfully configured an S3 bucket to serve web content publicly, making it accessible via the internet.

Project Duration: Approximately 20 minutes

Difficulty Level: Easy

AWS Region Used: eu-west-3 (Paris)

Table of Contents

- [What I Built](#)
 - [Technologies & Concepts](#)
 - [Step-by-Step Implementation](#)
 - [Conclusion](#)
-

What I Built

A fully functional static website hosted on AWS S3, accessible to anyone on the internet through a public URL. The website includes HTML content and image assets, all served directly from an S3 bucket.

Live Website Endpoint: [http://nextwork-website-project-\[yourname\].s3-website.eu-west-3.amazonaws.com](http://nextwork-website-project-[yourname].s3-website.eu-west-3.amazonaws.com)

Technologies & Concepts

AWS Services Used

- **Amazon S3 (Simple Storage Service)** - Object storage service for hosting website files

Key Concepts Learned

1. **S3 Buckets** - Storage containers for organizing and storing objects in the cloud
 2. **Static Website Hosting** - Serving HTML, CSS, and JavaScript files directly from S3
 3. **Access Control Lists (ACLs)** - Managing permissions at the object level
 4. **Bucket Policies** - Controlling access to bucket resources
 5. **Public vs Private Access** - Understanding security configurations for web hosting
 6. **Bucket Versioning** - Tracking changes to objects over time
-

Step-by-Step Implementation

Step 1: Create an S3 Bucket

What I did:

- Opened the AWS Management Console and navigated to S3
- Selected the AWS Region closest to my location (eu-west-3 - Paris)
- Created a new bucket with a globally unique name: `nextwork-website-project-youhad`

Configuration Settings:

- **Object Ownership:** ACLs enabled
- **Block Public Access:** Disabled (required for public website)
- **Bucket Versioning:** Enabled
- **Bucket Owner:** Preferred

Why this matters:

- S3 bucket names must be globally unique across all AWS accounts
- Choosing a nearby region reduces latency for website visitors
- ACLs allow granular control over individual object permissions

General purpose buckets (3) <small>Info</small>						
Buckets are containers for data stored in S3.						
<input type="text"/> Find buckets by name						
Name	AWS Region	Creation date	<	1	>	
nextwork-website-project-youhad	Europe (Paris) eu-west-3	December 16, 2025, 23:15:07 (UTC+01:00)				
[REDACTED]	Europe (Paris) eu-west-3	July 3, 2025, 12:10:53 (UTC+01:00)				
[REDACTED]	US East (N. Virginia) us-east-1	July 3, 2025, 11:23:17 (UTC+01:00)				

Time taken: ~1 minute

Step 2: Upload Website Content

What I did:

- Downloaded the provided HTML file (`index.html`)
- Downloaded and unzipped the images folder (`NextWork - Everyone should be in a job they love_files`)
- Uploaded both the HTML file and the unzipped folder to my S3 bucket

Files Uploaded:

1. `index.html` - The main webpage file
2. `NextWork - Everyone should be in a job they love_files/` - Folder containing all image assets

How they work together: The `index.html` file contains references to images stored in the folder. When the webpage loads, it retrieves these images from the S3 bucket to display the complete website.

nextwork-website-project-youhad [Info](#)

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Find objects by prefix](#) Show versions

Name	Type	Last modified	Size	Storage class
index.html	html	December 17, 2025, 20:27:50 (UTC+01:00)	58.8 KB	Standard
NextWork - Everyone should be in a job they love_files/	Folder	-	-	-

Key Understanding:

- HTML files define the structure and content of web pages
- S3 stores these files as objects within the bucket
- All website assets must be uploaded for the site to display correctly

Step 3: Configure Static Website Hosting

What I did:

- Navigated to the bucket's **Properties** tab
- Scrolled to the **Static website hosting** panel
- Enabled static website hosting with the following settings:
 - Static web hosting:** Enable
 - Hosting type:** Host a static website
 - Index document:** [index.html](#)

What is Website Hosting? Website hosting makes files stored on a server accessible via the internet. By enabling static website hosting on S3, AWS generates a public URL (bucket website endpoint) that allows anyone to view the website.

Edit static website hosting [Info](#)

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

Disable Enable

Hosting type

Host a static website
Use the bucket endpoint as the web address. [Learn more](#)

Redirect requests for an object
Redirect requests to another bucket or domain. [Learn more](#)

Index document

Specify the home or default page of the website.

index.html

Understanding Bucket Website Endpoint: A bucket website endpoint is a URL like:

`http://nextwork-website-project-youhad.s3-website.eu-west-3.amazonaws.com`

This endpoint allows browsers to access and display the website content stored in the S3 bucket.

Step 4: Initial Endpoint Test (403 Error)

What happened: When I first clicked on the bucket website endpoint, I encountered a **403 Forbidden** error.



403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: 8HPTACNB2TJB6T0
- HostId: DXq++dLqn+mEDfjehwJHdy98nG7La079D02qd/SMGOv3jvKWLWulteo8Tw5wiB5koYt+dSL80e4=

Why this error occurred:

- By default, all objects in S3 are **private**
- Even though static website hosting was enabled, the actual files (index.html and images) were still private
- The bucket was visible, but its contents were not accessible to the public

This is actually good! AWS keeps data secure by default. This error is a security feature protecting your content until you explicitly grant public access.

Step 5: Make Objects Public Using ACLs

What I did:

1. Returned to the S3 console and navigated to the **Objects** tab
2. Selected all uploaded objects (index.html and the images folder)
3. Clicked the **Actions** button
4. Selected **Make public using ACL**
5. Confirmed the action

What are ACLs? Access Control Lists (ACLs) are sets of rules that determine who can access specific objects in S3. They provide object-level permission management, allowing different AWS accounts to own and control different files within the same bucket.

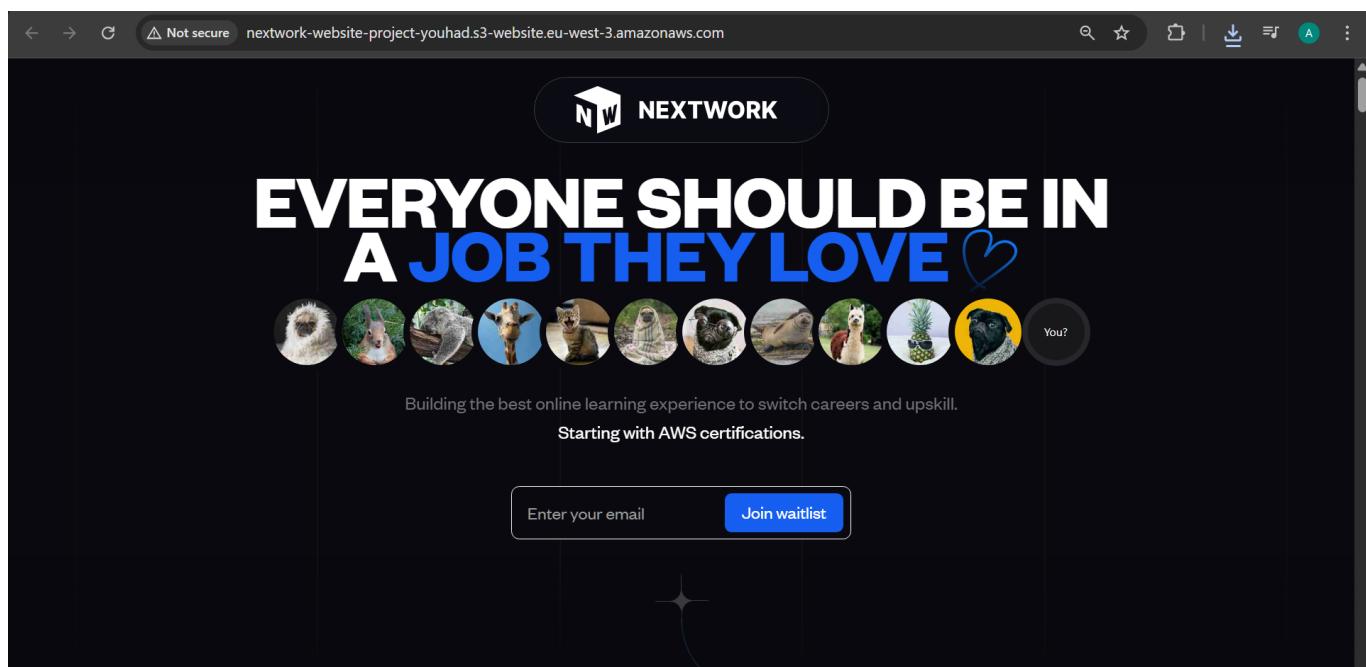
The screenshot shows the AWS S3 console interface. At the top, there's a toolbar with buttons for Copy S3 URI, Copy URL, Download, Open, and Delete. Below the toolbar, there are buttons for Actions (with arrows), Create folder, and Upload. A dropdown menu under Actions includes options like Download as, Share with a presigned URL, Calculate total size, Copy, Move, Initiate restore, Query with S3 Select, Edit actions, Rename object, Edit storage class, Edit server-side encryption, Edit metadata, Edit tags, and Make public using ACL. The main area displays a single object named "index.html". The object details are as follows:

Type	Last modified	Size	Storage class
html	March 27, 2025, 14:50:30 (UTC+13:00)	58.8 KB	Standard
Folder	-	-	-

Result: After making the objects public, I refreshed the bucket website endpoint URL and...

Step 6: Success! Website is Live

The website is now publicly accessible!



The website loaded perfectly with all images displaying correctly. Anyone with the bucket website endpoint URL can now view this website from anywhere in the world!

What I achieved so far:

- Created an S3 bucket
 - Uploaded website files (HTML and images)
 - Configured static website hosting
 - Made objects publicly accessible
 - Successfully hosted a website on AWS!
-

Step 7: Creating a Bucket Policy

After successfully hosting my website, I took on the **Secret Mission** challenge to use bucket policies for advanced access control!

What I did:

- Navigated to the **Permissions** tab in my S3 bucket
- Clicked **Edit** on the **Bucket policy** section
- Created a JSON policy to **deny deletion** of the `index.html` file

The Bucket Policy I Created:

```
{
  "Version": "2012-10-17",
  "Id": "MyBucketPolicy",
  "Statement": [
    {
      "Sid": "BucketPutDelete",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:DeleteObject",
      "Resource": "arn:aws:s3:::nextwork-website-project-youhad/index.html"
    }
  ]
}
```

The screenshot shows the 'Edit bucket policy' page in the AWS Management Console. At the top, there's a breadcrumb navigation: Amazon S3 > Buckets > nextwork-website-project-youhad > Edit bucket policy. To the right of the breadcrumb are three small icons: a magnifying glass, a copy symbol, and a refresh symbol.

The main area is titled 'Bucket policy' and contains the following JSON code:

```
1 Version: "2012-10-17",
2   "Id": "MyBucketPolicy",
3   "Statement": [
4     {
5       "Sid": "BucketPutDelete",
6       "Effect": "Deny",
7       "Principal": "*",
8       "Action": "s3:DeleteObject",
9       "Resource": "arn:aws:s3:::nextwork-website-project-youhad/index.html"
10      }
11    ]
12  ]
```

To the right of the code editor, there's a sidebar with the following sections:

- Edit statement**: A button to edit an existing statement or add a new one.
- Select a statement**: A placeholder text: "Select an existing statement in the policy or add a new statement."
- Add new statement**: A button to add a new statement.
- Activate Windows**: A note: "Go to Settings to activate Windows."

What this policy does:

- Effect: "Deny"** - Blocks the specified action
- Action: "s3:DeleteObject"** - Prevents object deletion
- Resource** - Targets specifically the `index.html` file
- Principal: "*" - Applies to all users (including the bucket owner!)**

Key Learning: Bucket policies use JSON format and can control access at the bucket level or for specific objects. Unlike ACLs which control individual object permissions, bucket policies can enforce rules across multiple objects or the entire bucket.

Step 8: Testing the Bucket Policy

Time to test if my bucket policy actually works!

What I did:

- Went to the **Objects** tab
- Selected the `index.html` file
- Clicked **Delete** to try deleting it
- Confirmed the deletion

Result: Access Denied! 🔒

The screenshot shows the AWS S3 console with a red header bar indicating a failure to delete objects. The main content area displays a summary of successful and failed deletions, followed by a detailed table of the failed deletion. The table includes columns for Name, Type, Last modified, Size, and Error. The error column for the single object shows 'Access denied'.

Name	Type	Last modified	Size	Error
index.html	html	December 17, 2025, 22:06:04 (UTC+01:00)	58.8 KB	Access denied

The error message showed:

- **Failed to delete:** 1 object, 58.8 KB
- **Error:** Access denied

This proves the bucket policy is working! Even though I'm the bucket owner, the policy successfully prevented me from deleting the `index.html` file. This demonstrates how bucket policies can enforce strict access controls to protect critical files.

Why this matters: In production environments, bucket policies can prevent accidental deletion of important files, enforce compliance requirements, and add an extra layer of security to your S3 buckets.

Resources Cleanup

Now that I've completed the project and tested the bucket policy, it's time to clean up all resources to avoid any charges.

Step 9: Deleting the Bucket Policy

Before I can delete objects, I need to remove the bucket policy that's preventing deletion!

What I did:

1. Navigated back to the **Permissions** tab
2. Found the **Bucket policy** section
3. Clicked **Delete** to remove the policy
4. Confirmed deletion by typing "delete"

The screenshot shows the AWS S3 Bucket Policy page for a bucket named "nextwork-website-project-youhad". A modal window titled "Delete bucket policy?" is open. It contains a warning message: "⚠ Deleting a bucket policy can't be undone. We recommend you back up a copy of this policy before deleting it." Below this is a text input field with the word "delete" typed into it. At the bottom of the modal are "Cancel" and "Delete" buttons. The background of the main page shows the JSON code of the bucket policy.

Important: Bucket policies cannot be undone once deleted. AWS recommends backing up your policy before deletion - good thing this was just a learning project!

Step 10: Successfully Deleting All Objects

With the bucket policy removed, I can now delete all objects!

What I did:

1. Returned to the **Objects** tab
2. Selected **all objects** in the bucket (index.html and the images folder)
3. Clicked **Delete**
4. Confirmed by typing the required text
5. Clicked **Delete objects**

Result: Success! ✓

The screenshot shows the "Delete objects: status" page. At the top, a green bar indicates "Successfully deleted objects" with a count of "45 objects, 2.7 MB". Below this, a message says "After you navigate away from this page, the following information is no longer available." The "Summary" section shows the source as "S3://nextwork-website-project-youhad" and the results as "Successfully deleted: 45 objects, 2.7 MB" and "Failed to delete: 0 objects". The "Failed to delete" tab is selected, showing a table with 0 rows. The table has columns for Name, Type, Last modified, Size, and Error. A note at the bottom right says "Activate Windows Go to Settings to activate Windows".

The deletion status showed:

- **Successfully deleted:** 45 objects, 2.7 MB
- **Failed to delete:** 0 objects

All website files have been removed from the S3 bucket!

Step 11: Deleting the S3 Bucket

With all objects deleted, it's time to remove the bucket itself!

What I did:

1. Navigated to the **Buckets** list page
2. Selected my bucket (`nextwork-website-project-youhad`)
3. Clicked **Delete**
4. Confirmed deletion by typing the bucket name: `nextwork-website-project-youhad`
5. Clicked **Delete bucket**

The screenshot shows the 'Delete bucket' confirmation dialog. At the top, there is a breadcrumb navigation: 'Amazon S3 > Buckets > nextwork-website-project-youhad > Delete bucket'. Below the title 'Delete bucket `nextwork-website-project-youhad`' is a warning message: '⚠ Deleting a bucket cannot be undone.' followed by four bullet points about bucket uniqueness and access points. A 'Learn more' link is also present. The main input field contains the bucket name 'nextwork-website-project-youhad'. At the bottom right are 'Cancel' and 'Delete bucket' buttons.

Result: Bucket successfully deleted!

The bucket and all its configurations (static website hosting, ACLs, versioning) have been completely removed from AWS.

Why cleanup is critical:

- S3 charges for storage, even small amounts add up over time
 - Empty buckets don't incur storage charges, but it's best practice to delete unused resources
 - Cleaning up prevents clutter and keeps your AWS account organized
 - Once a bucket is deleted, the name becomes available for others to use
-

Conclusion

This project provided hands-on experience with one of AWS's most fundamental services - Amazon S3. I successfully:

- Created and configured an S3 bucket for web hosting
- Managed object permissions using ACLs
- Hosted a publicly accessible static website
- **Implemented bucket policies to prevent file deletion (Secret Mission!)**
- **Tested security policies to verify they work correctly**
- Properly cleaned up all resources following AWS best practices
- Gained practical understanding of cloud storage, web hosting, and access control concepts

Additional Resources

- [AWS S3 Documentation](#)
 - [Static Website Hosting Guide](#)
 - [S3 Pricing Information](#)
-

Project Completed: December 2025

Author: YOUHAD AYOUB

Region: eu-west-3 (Paris)

NextWork Challenge: AWS Beginners Challenge - Project 1

This project was completed as part of the NextWork AWS Beginners Challenge. Special thanks to the NextWork community for their support and guidance!