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

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`

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:

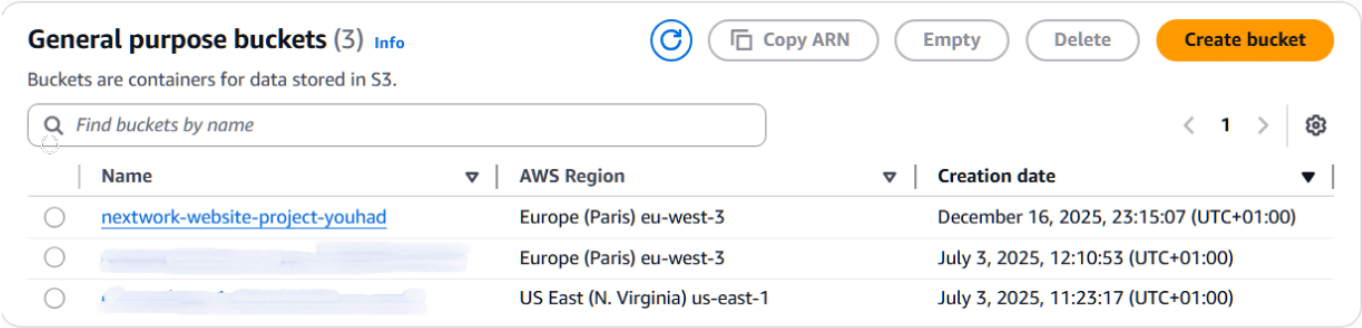
- Opened the AWS Management Console and navigated to S3
- Selected the AWS Region closest to my location (eu-west-3 - Paris)
- Created a new bucket with a globally unique name: `nextwork-website-project-youhad`

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



Time taken: ~1 minute

Step 2: Upload Website Content

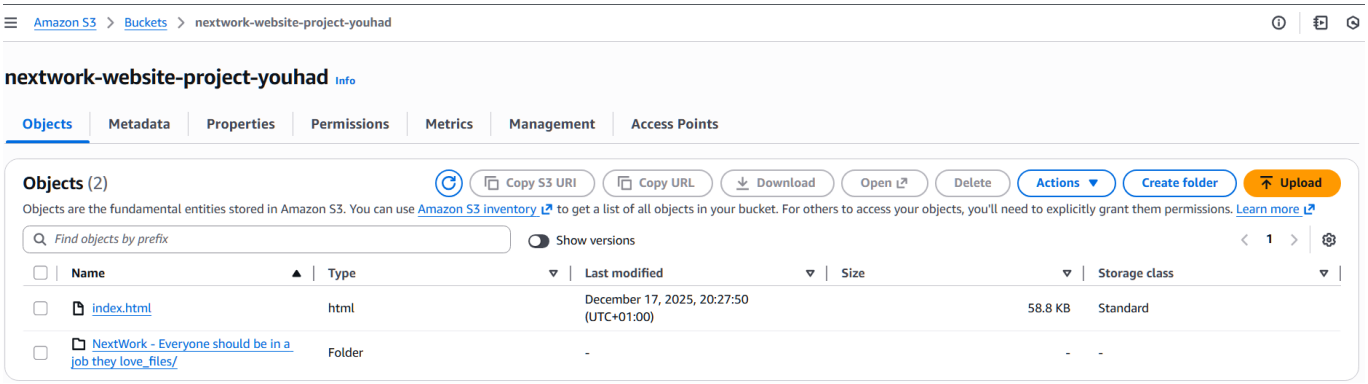
What I did:

- 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.



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

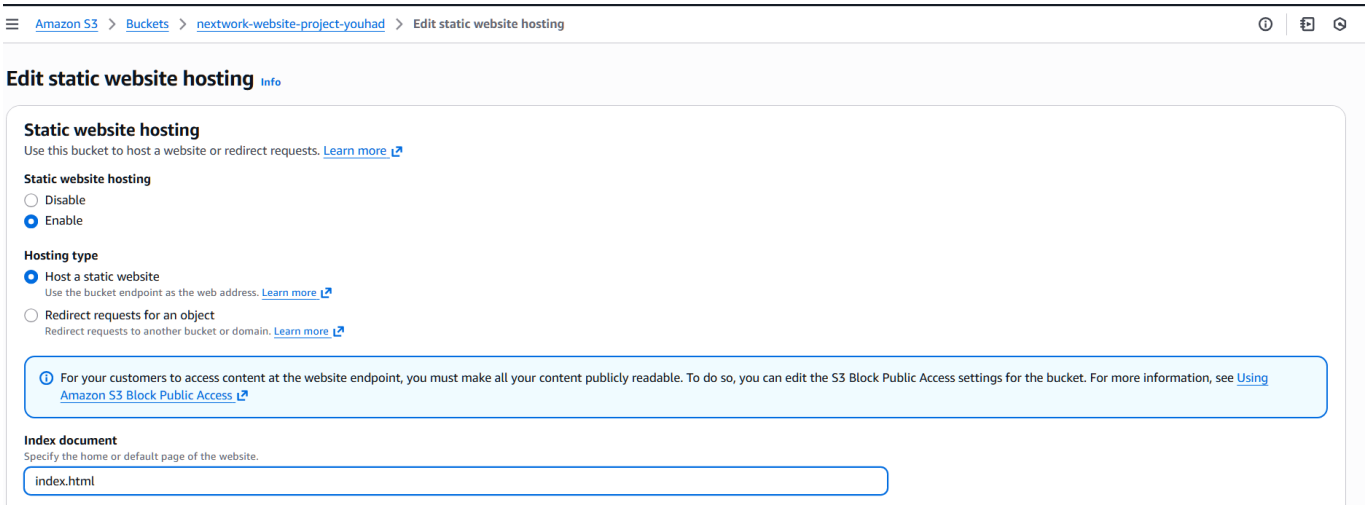
---

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.



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.

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

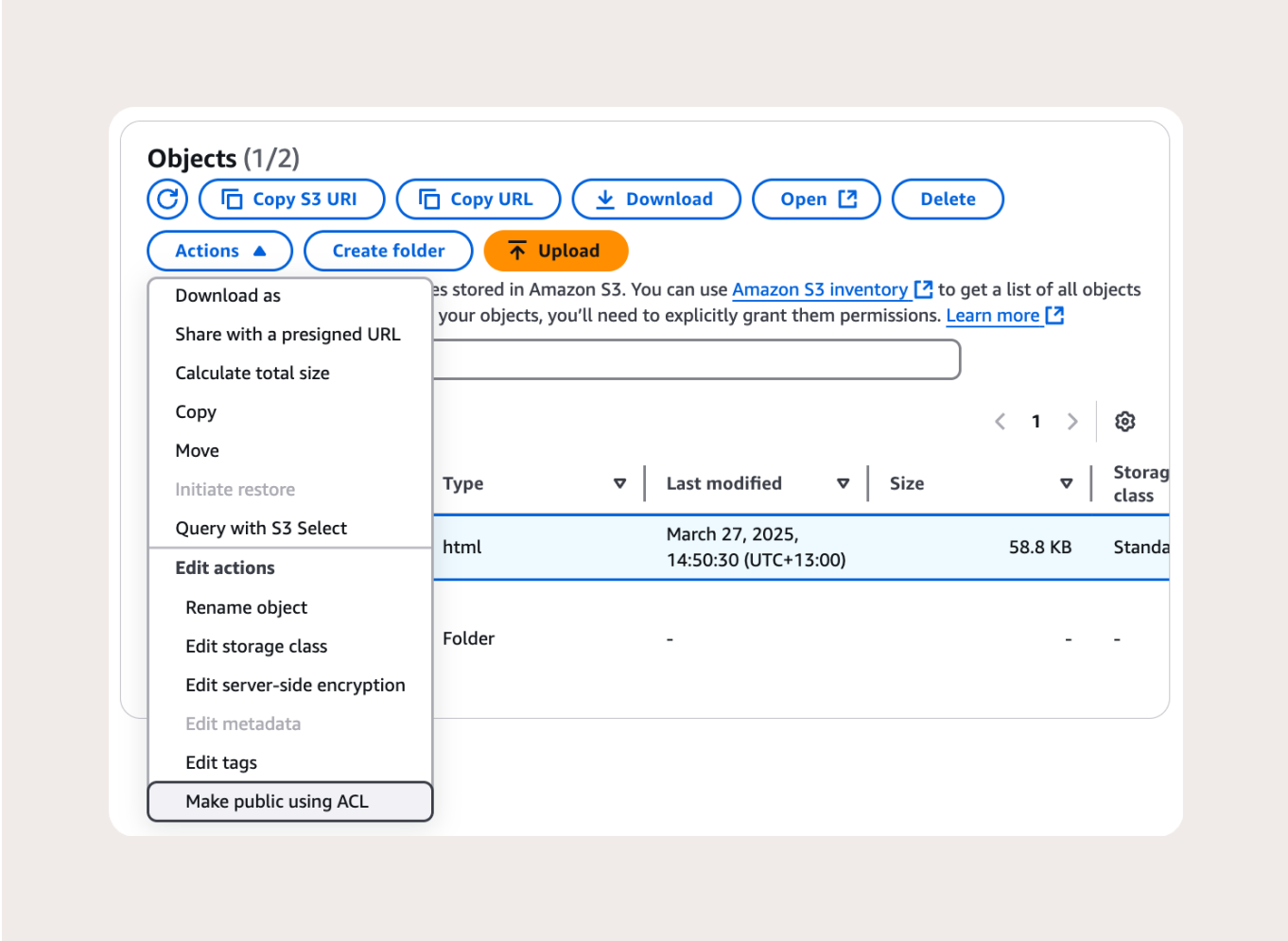
This is actually good! AWS keeps data secure by default. This error is a security feature protecting your content until you explicitly grant public access.

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

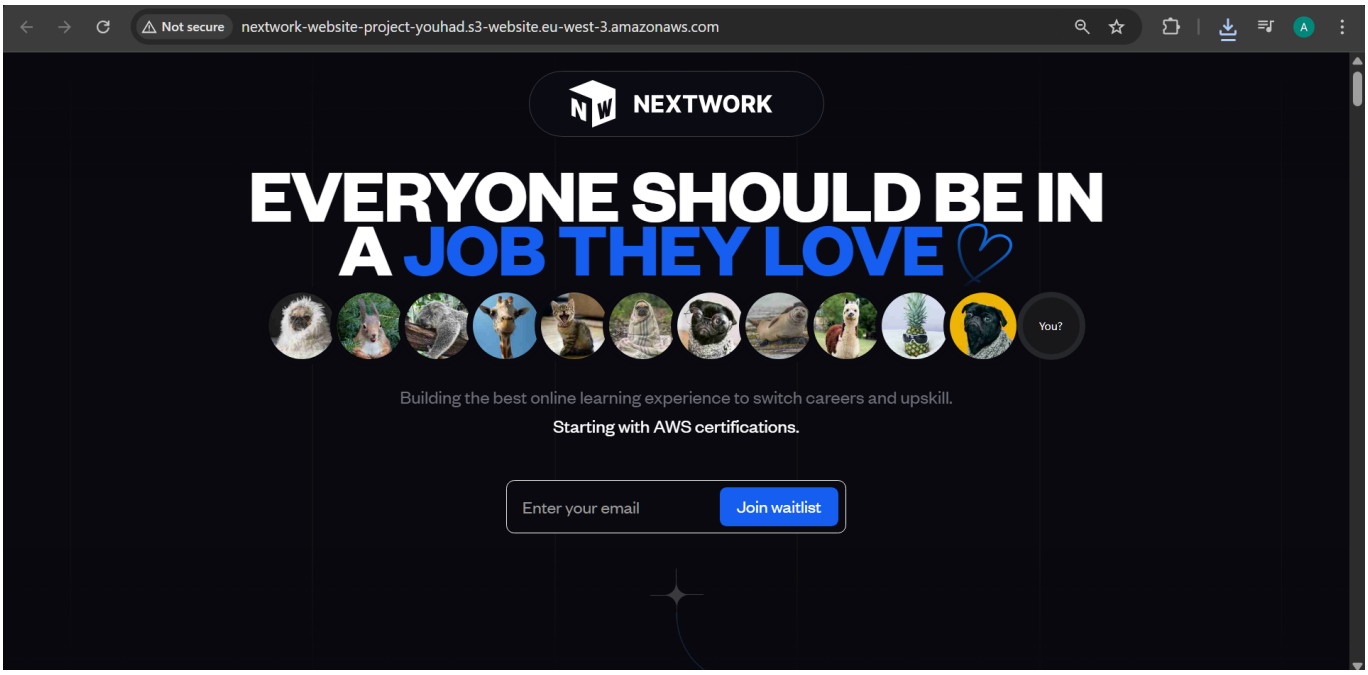
What are ACLs? Access Control Lists (ACLs) are sets of rules that determine who can access specific objects in S3. They provide object-level permission management, allowing different AWS accounts to own and control different files within the same bucket.



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

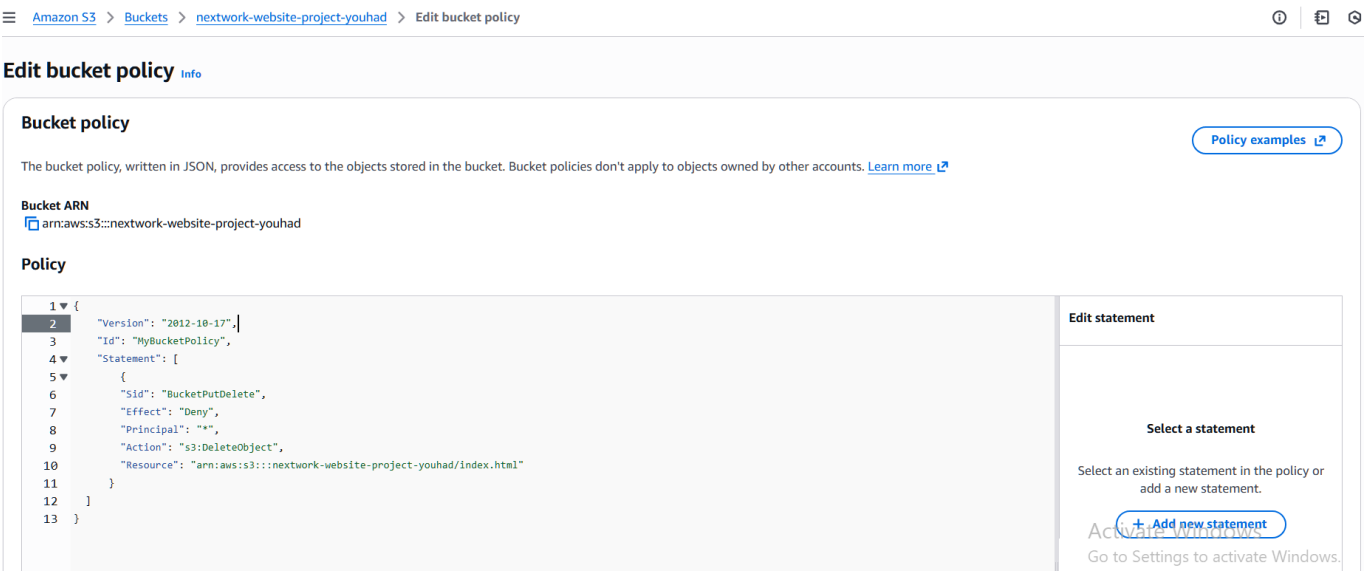
After successfully hosting my website, I took on the **Secret Mission** challenge to use bucket policies for advanced access control!

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"
    }
  ]
}
```



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.

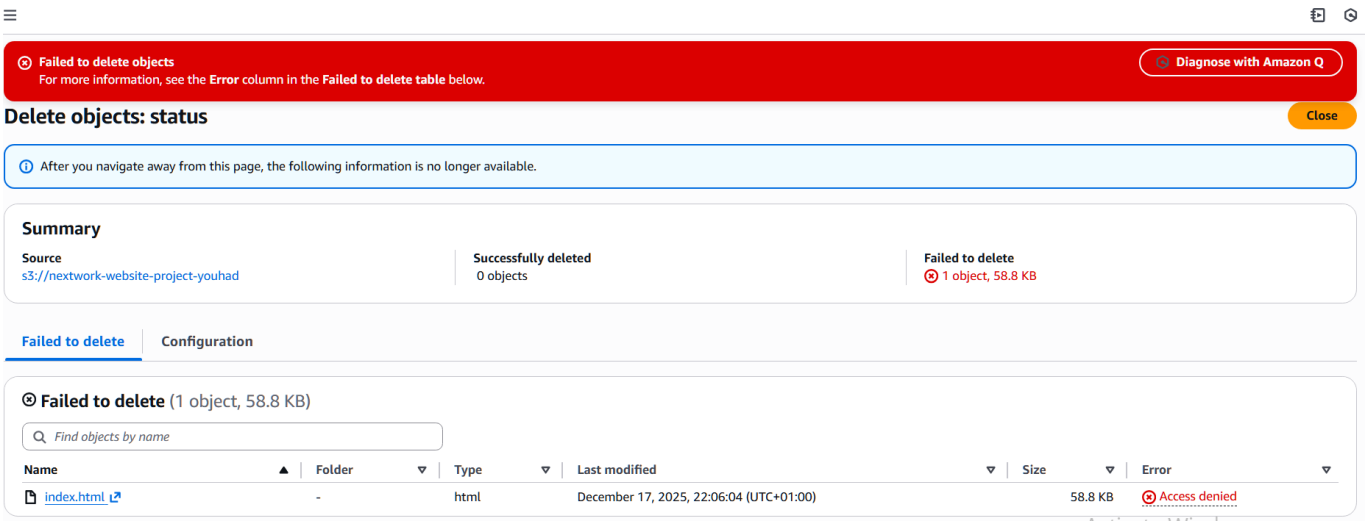
Step 8: Testing the Bucket Policy

Time to test if my bucket policy actually works!

What I did:

1. Went to the **Objects** tab
2. Selected the `index.html` file
3. Clicked **Delete** to try deleting it
4. Confirmed the deletion

Result: 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

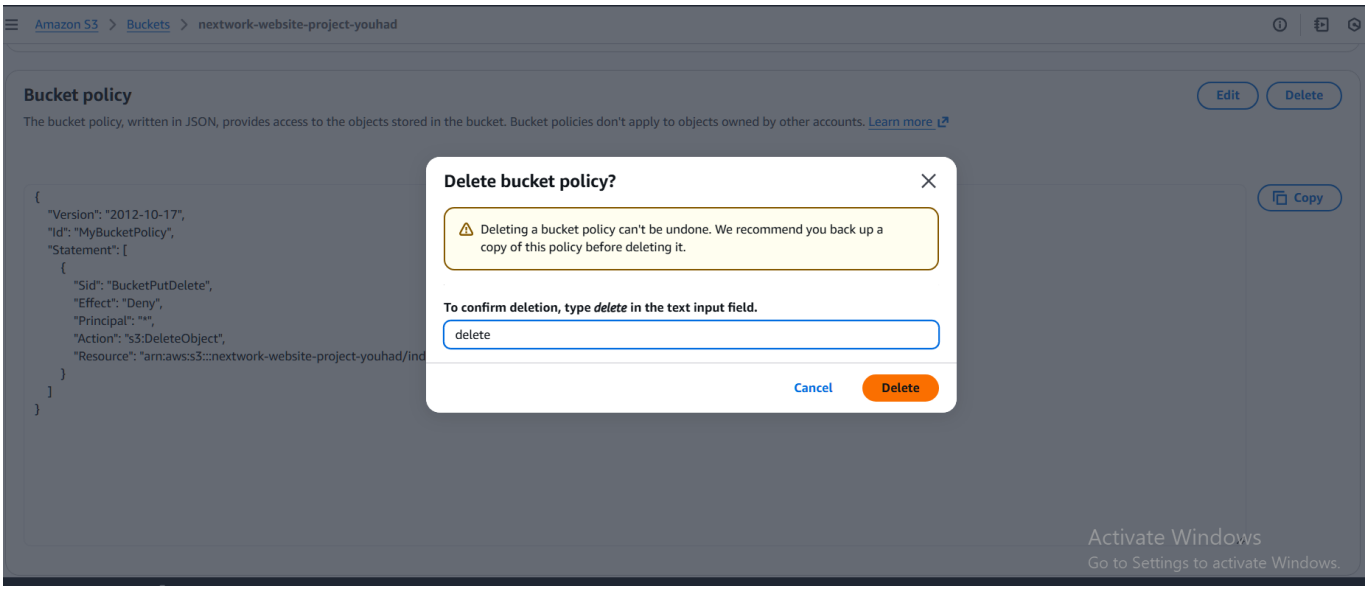
Now that I've completed the project and tested the bucket policy, it's time to clean up all resources to avoid any charges.

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"



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

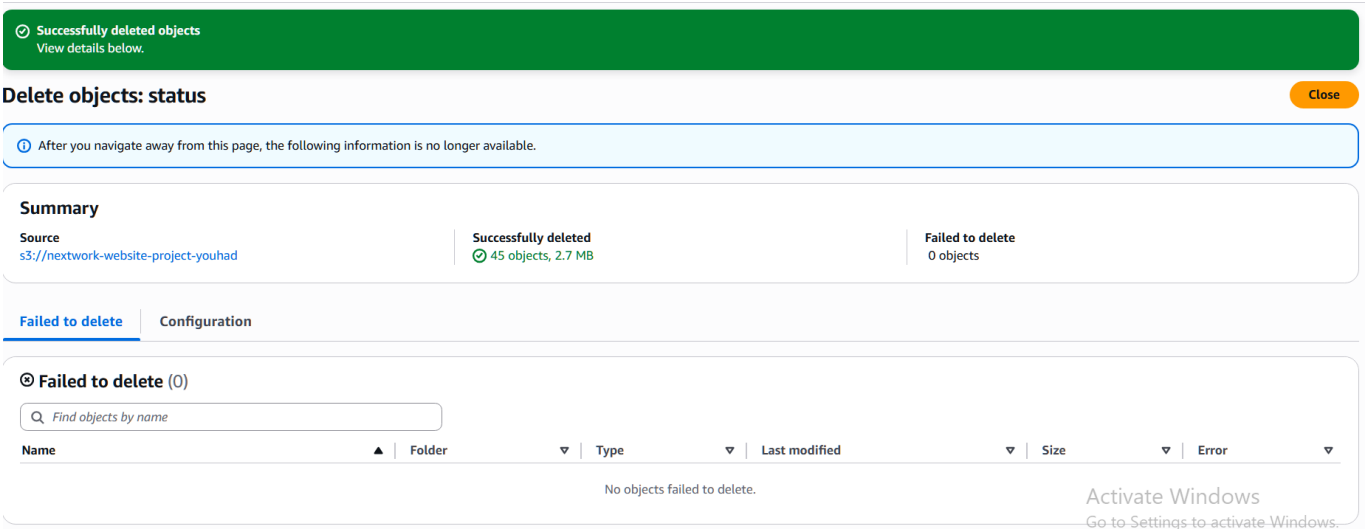
Step 10: Successfully Deleting All Objects

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 deletion status showed:

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

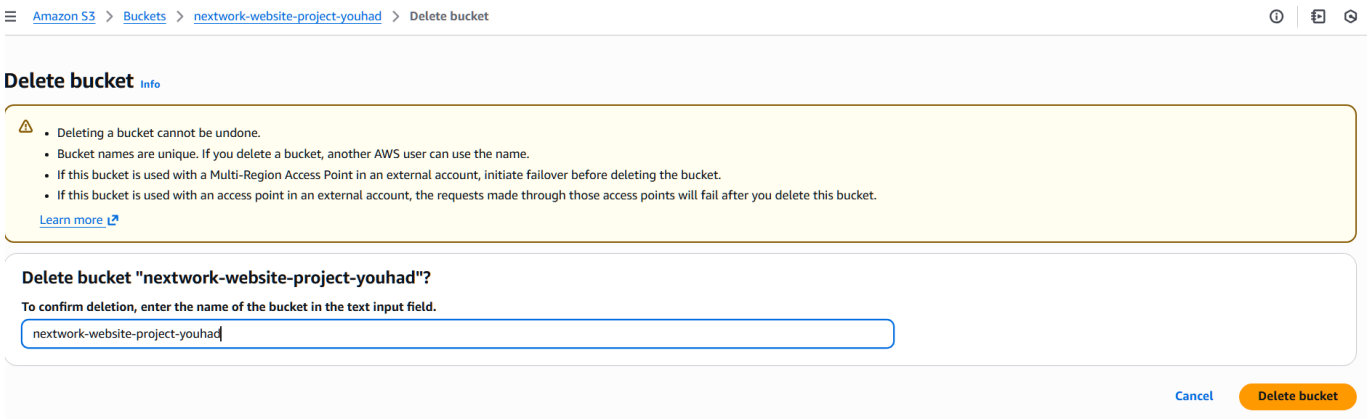
All website files have been removed from the S3 bucket!

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**



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!