

Static Website Hosting & CI/CD Deployment on AWS

Prepared by
Vikram Kumar Matte

Project Overview

Project Summary:

This project demonstrates the end-to-end process of hosting a static website using AWS services, with a focus on security, performance, and automation. The website is hosted in an S3 bucket, distributed globally via CloudFront CDN, and secured using HTTPS with an SSL certificate from AWS Certificate Manager (ACM).

A custom domain is registered and configured using Amazon Route 53, and GitHub is integrated with AWS CodePipeline to automatically deploy updates to the website, streamlining the DevOps workflow.

This approach ensures:

- Low-cost and highly durable website hosting
- Fast and secure content delivery via CloudFront
- Automated deployment with GitHub integration
- Domain-level routing and HTTPS support

AWS Services and Tools Used:

- Amazon S3
- Amazon CloudFront
- AWS Certificate Manager (ACM)
- Amazon Route 53
- AWS CodePipeline
- Git & GitHub

Domain Registration

Step 1: Register a Domain Name Using Route 53

Before creating your static website, the first important step is to decide what domain name your website will use — for example, `www.static-website-hosting.org`.

A domain name is the web address that people type into their browser to visit your site. It's your website's identity on the internet.

Why Do You Need a Domain Name?

- It gives your website a professional, branded appearance.
- It makes your website easier to remember and access.
- Later, this domain name will help route internet traffic to the correct location where your site is hosted.

Using Route 53 to Register a Domain

Amazon Route 53 is a reliable service from AWS that lets you register and manage domain names directly within your AWS account.

Tip: You can also use a third-party registrar (like GoDaddy or Namecheap), but using Route 53 simplifies integration with other AWS services later in the setup.

Important Note

Domain registration is NOT included in the AWS Free Tier.

Charges will apply based on the top-level domain you choose (.com, .org, etc.).

Pricing usually starts around \$12/year.

Steps to Register Your Domain in Route 53

- Go to the AWS Console → Open Route 53
- You can search for “Route 53” in the AWS search bar.
- Click “Registered Domains” → “Register Domain”
- Search for Your Desired Domain
- Example: www.static-website-hosting.org
- Check Availability
- If it’s available, proceed to register it. If not, try another variation.
- Fill Out Contact Details & Confirm Purchase
- AWS will ask for registrant details (name, address, email).
- Make Payment for 1-Year Registration
- Wait for Confirmation
- Once complete, the domain will be listed under Registered Domains, and a Hosted Zone will be automatically created (we’ll use this later for routing web traffic).

The screenshot shows the AWS Route 53 'Register domains' interface. At the top, the breadcrumb navigation reads 'Route 53 > Registered domains > Register domains'. The main heading is 'Register domains' with an 'Info' link. Below this, a note states: 'Pricing for domain names varies by top-level domain (TLD). For more information, view [price with different TLDs](#).' The interface is divided into several sections:

- Search for domain:** Contains a text input field with 'static-website-hosting.org', a 'Search' button, and a 'Check availability for a domain' link.
- Standard pricing:** A section with a right-pointing triangle icon, stating 'Pricing for domain names varies by top-level domain (TLD), such as .com or .org.'
- Search result:** A table with three columns: 'Domain', 'Price/year', and 'Actions'. It lists 'static-website-hosting.org' with a green 'Exact match' tag, a price of '14.00 USD', and a 'Renews at 14.00 USD' note. A 'Select' button is next to the entry.
- Selected domains (0/5):** A box on the right with the text 'Search for domains and make a selection' and a 'Proceed to checkout' button at the bottom.

Domain	Price/year	Actions
static-website-hosting.org Exact match	14.00 USD Renews at 14.00 USD	Select

Creation of an S3 Bucket

Step 2: Create an S3 Bucket to Host Your Website Files

What is Amazon S3?

Amazon S3 (Simple Storage Service) is a service provided by AWS that allows you to store and retrieve any amount of data — such as HTML, CSS, JavaScript, images, and other static files — from anywhere on the web.

In this project, we use S3 to host our static website files, such as index.html, style.css, and more.

Create an S3 Bucket (Must Match Your Domain)

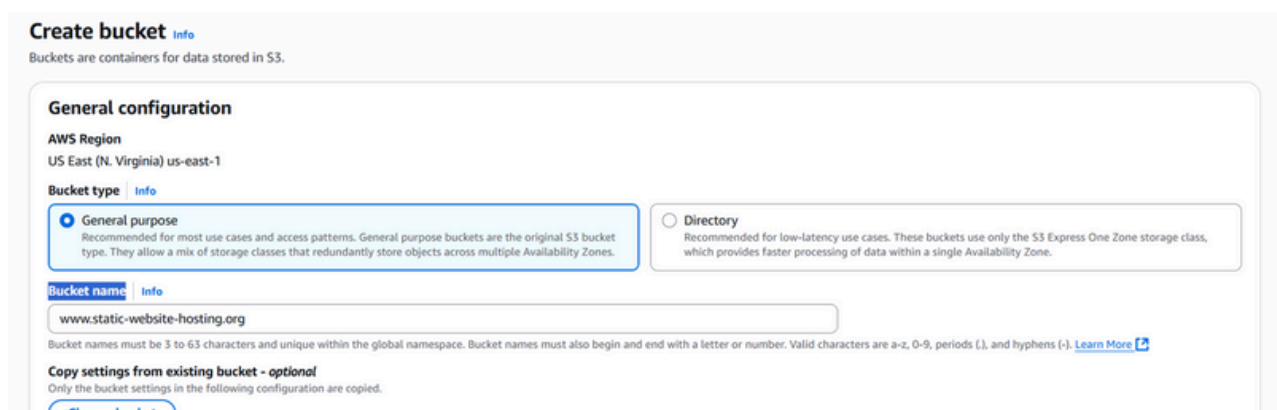
If your domain is `www.static-website-hosting.org`, then your S3 bucket name must exactly match the domain or subdomain you want to serve.

For example:

- Use `www.static-website-hosting.org` as the bucket name if you want to serve the website at `www.static-website-hosting.org`.

Steps to Create the S3 Bucket

1. Go to the S3 service in AWS Console
2. Click “Create Bucket”
3. Enter the bucket name → e.g., `www.static-website-hosting.org`
4. Leave rest of the settings as default



The screenshot shows the 'Create bucket' page in the AWS Management Console. At the top, it says 'Create bucket' with an 'Info' link. Below that, a note states 'Buckets are containers for data stored in S3.' The main section is titled 'General configuration'. Under 'AWS Region', it shows 'US East (N. Virginia) us-east-1'. For 'Bucket type', there are two options: 'General purpose' (selected with a radio button) and 'Directory'. The 'General purpose' option has a description: 'Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.' The 'Directory' option is described as 'Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.' Below this, the 'Bucket name' field is populated with 'www.static-website-hosting.org'. A note below the field states: 'Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn More](#)'. At the bottom, there is a section for 'Copy settings from existing bucket - optional' with a note: 'Only the bucket settings in the following configuration are copied.' and a 'Choose bucket' button.

Block All Public Access (Very Important)

On the same screen, you'll see a section to block public access to the bucket.

Leave it enabled (checked) to block all public access.

We are keeping this bucket private for security.

Later, we will allow access through a secure AWS service called CloudFront, which you will configure in the next steps.

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☒ Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- ☒ **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☒ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- ☒ **Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☒ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Upload Website Files into the S3 Bucket

Once the bucket is created:

1. Open the bucket
2. Go to the "Objects" tab
3. Click "Upload"
4. Upload all your website files — like index.html, style.css, etc.

Amazon S3 > Buckets > www.static-website-hosting.org

www.static-website-hosting.org

Objects (3)

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

Name	Type	Last modified	Size	Storage class
index.html	html	June 7, 2025, 18:10:26 (UTC-05:00)	810.0 B	Standard
steps.html	html	June 7, 2025, 18:10:27 (UTC-05:00)	1.2 KB	Standard
style.css	css	June 7, 2025, 18:10:27 (UTC-05:00)	439.0 B	Standard

Setting up S3 Website Hosting Endpoint

Step 3: Enable Static Website Hosting on the S3 Bucket

Once your website files are uploaded to the S3 bucket, the next step is to enable static website hosting for that bucket.

Enable Static Website Hosting

1. Go to the Properties tab of your S3 bucket
2. Scroll down to the section called "Static website hosting"
3. Click Edit
4. Select "Enable"
5. Enter the name of your main file in Index document (usually index.html)
6. Click Save Changes

www.static-website-hosting.org [Info](#)

Objects

Metadata

Properties

Permissions

Metrics

Management

Access Points

Static website hosting

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

Edit

We recommend using AWS Amplify Hosting for static website hosting

Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. [Learn more about Amplify Hosting](#) or [View your existing Amplify apps](#)

Create Amplify app

S3 static website hosting

Disabled

Amazon S3 > Buckets > www.static-website-hosting.org > Edit static website hosting

Info Help Refresh

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](#)

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

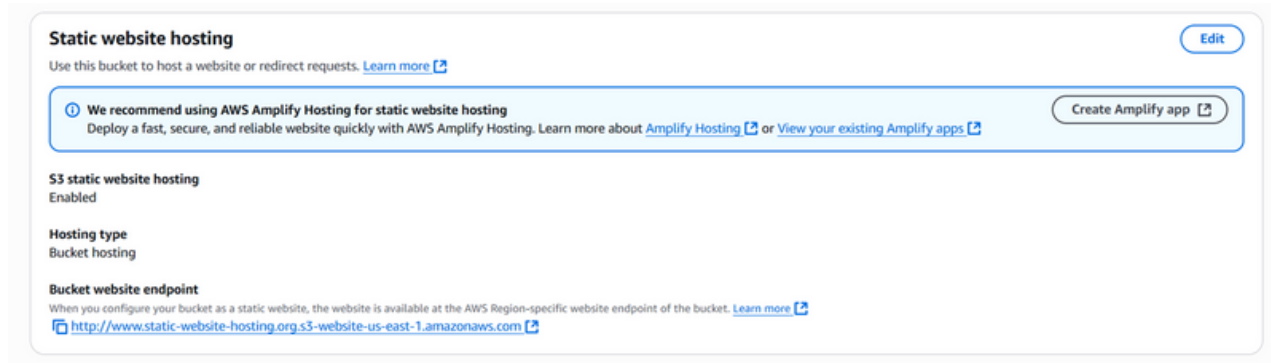
Index document

Specify the home or default page of the website.

index.html

Get the S3 Website Hosting Endpoint

After enabling website hosting, S3 will generate a website endpoint URL for your bucket.



Try Opening the Link – You'll See an Error

If you open this link in a browser, you'll likely see an Access Denied error.

403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: 2VBJC7Y7NKGAKW8
- HostId: Xa9R1X3GaZy9ZhlUzle+Cvi2xLSHDnXCmTPuqOJ6QpQemh3WT1ywxMu3rJOVMzCtLRo2YmDxDxK3bIXAGYk9yTSe/zbFuu

Why Enable Website Hosting if It Doesn't Work Yet?

Even though the website doesn't load right now, enabling static website hosting is a required step because:

- It tells S3 to behave like a web server, serving your files like a normal website.
- It defines which file (like index.html) should load by default when someone visits your site.
- This endpoint will be used later as the origin when we set up CloudFront (our secure delivery layer).

We'll fix the access issue later by using CloudFront + Origin Access Identity, which will securely serve your website to the public.

CloudFront Distribution

Step 4: Set Up CloudFront Distribution

What is Amazon CloudFront?

CloudFront is Amazon's Content Delivery Network (CDN). It delivers your website content securely and quickly to users around the world by caching it at multiple edge locations (near the users).

How Are We Using It?

In this project, we're using CloudFront to:

- Serve content from our private S3 bucket
- Add HTTPS (secure connection)
- Improve performance using caching
- Hide the raw S3 URL and serve content from our custom domain

Create a CloudFront Distribution

1. Go to the CloudFront service
2. Click Create Distribution
3. Under Origin domain, choose the S3 static website hosting endpoint. (This is the URL we got in the previous step after enabling website hosting)


Origin domain

Choose an AWS origin, or enter your origin's domain name. [Learn more](#) 





Enter a valid DNS domain name, such as an S3 bucket, HTTP server, or VPC origin ID.

 This S3 bucket has static web hosting enabled. If you plan to use this distribution as a website, we recommend using the S3 website endpoint rather than the bucket endpoint.

[Use website endpoint](#)

Add an OAI (Origin Access Identity)

When asked to "Restrict Bucket Access", enable it and create a new OAI.

What is OAI?

- OAI is a special CloudFront identity that can access your private S3 bucket.
- This ensures that only CloudFront can fetch files from your bucket.
- No one can directly access your S3 bucket using its link, which improves security.

Update the S3 Bucket Policy

After creating the OAI, CloudFront will generate a bucket policy for you. This policy gives read access only to CloudFront (via the OAI), not to the public. This makes your bucket private, yet still accessible securely via CloudFront.

Origin access | Info

☐ Public
Bucket must allow public access.

☐ Origin access control settings (recommended)
Bucket can restrict access to only CloudFront.

☒ Legacy access identities
Use a CloudFront origin access identity (OAI) to access the S3 bucket.

Origin access identity
Select an existing origin access identity (recommended) or create a new identity.

www.static-website-hosting.com.s3.us-east-1.amazonaws.com ▼

Create new OAI

Bucket policy
Update the S3 bucket policy to allow read access to the OAI.

☐ No, I will update the bucket policy

☒ Yes, update the bucket policy

Redirect HTTP to HTTPS

Set Viewer Protocol Policy to "Redirect HTTP to HTTPS"

Why?

- Ensures every visitor uses a secure connection
- Automatically redirects users who type http:// to https://
- Protects your website and users' data

Viewer

Viewer protocol policy

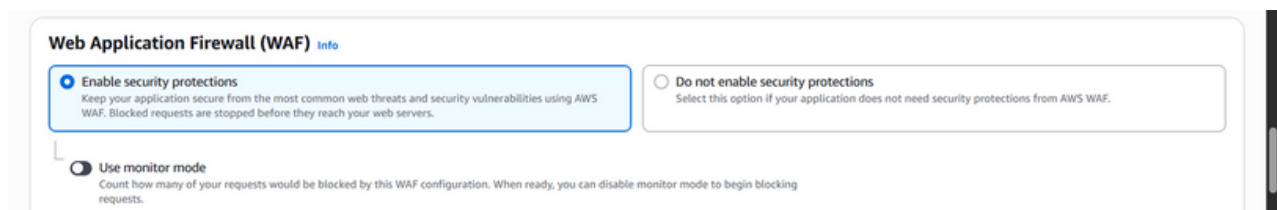
- ☐ HTTP and HTTPS
- ☒ Redirect HTTP to HTTPS
- ☐ HTTPS only

Enable AWS WAF (Optional)

You can attach an AWS Web Application Firewall (WAF) to your CloudFront distribution.

Why?

- WAF helps protect your site from common threats like SQL injection, bot traffic, etc.
- Adds another layer of security to your website



The screenshot shows the 'Web Application Firewall (WAF)' configuration section. It has a title 'Web Application Firewall (WAF)' with an 'Info' link. Below the title are two radio button options. The first option, 'Enable security protections', is selected and highlighted with a blue border. Its description reads: 'Keep your application secure from the most common web threats and security vulnerabilities using AWS WAF. Blocked requests are stopped before they reach your web servers.' The second option, 'Do not enable security protections', is unselected. Its description reads: 'Select this option if your application does not need security protections from AWS WAF.' Below these options is a checkbox labeled 'Use monitor mode', which is also unselected. Its description reads: 'Count how many of your requests would be blocked by this WAF configuration. When ready, you can disable monitor mode to begin blocking requests.'

Add Alternate Domain Names (CNAMEs)

In the “Alternate Domain Names (CNAMEs)” section, add:

- www.static-website-hosting.org
- static-website-hosting.org

Why?

- This tells CloudFront to accept requests on your custom domain
- Without this, CloudFront will only work via its default .cloudfront.net URL



The screenshot shows the 'Alternate domain name (CNAME) - optional' section. It has a title 'Alternate domain name (CNAME) - optional' and a subtitle 'Add the custom domain names that you use in URLs for the files served by this distribution.' Below the subtitle are two input fields. The first field contains 'www.static-website-hosting.org' and has a 'Remove' button to its right. The second field contains 'static-website-hosting.org' and also has a 'Remove' button to its right. At the bottom of the section is an 'Add item' button.

Certificate Manager (ACM)

Step 5: Set Up SSL/TLS Certificate Using AWS Certificate Manager (ACM)

Why Do We Need This?

To make your website secure (HTTPS) and trusted by users' browsers, you need an SSL/TLS certificate.

Because we're hosting the site using a custom domain (like www.static-website-hosting.org) and using CloudFront, we must:

- Prove ownership of the domain
- Provide a valid certificate
- Ensure all communication is encrypted (HTTPS)

AWS provides this certificate service via AWS Certificate Manager (ACM) — free for public certificates.

When Does ACM Come Into Play?

While creating the CloudFront distribution, you'll see a section to attach an SSL certificate.

Click on “Request or import a certificate with ACM” — this takes you to the ACM console where you can request one.

Request a Public Certificate

In ACM:

1. Choose Request a public certificate
2. In Fully Qualified Domain Name (FQDN), enter the following:

Why Two Domain Names?

- `www.static-website-hosting.org` → For users who visit the site with the "www" prefix.
- `static-website-hosting.org` → For users who visit the site directly without "www".

This ensures your site works with both versions of the domain and gives a seamless user experience.

What Is a Fully Qualified Domain Name (FQDN)?

A Fully Qualified Domain Name is the complete domain path that uniquely identifies a resource on the internet.

Example: `www.static-website-hosting.org` is a fully qualified domain name.

Certificate Status: “Pending Validation”

After submitting the certificate request, ACM shows the status as “Pending Validation”.

This means ACM needs to verify that you own the domain before issuing the certificate.

How to Validate Ownership (Using CNAME)

1. You'll see a section to validate using DNS (CNAME).
2. If you're using Route 53 (as in this project), just click “Create records in Route 53”.
 - This will automatically add CNAME records to your domain's DNS settings.
3. These CNAME records act as proof to AWS that you control the domain.

Final Step: Wait for Validation

Once CNAME records are created:

- ACM will automatically detect them.
- The status will change from “Pending Validation” to “Issued” (usually within minutes).

Domains (2) [Create records in Route 53](#) [Export to CSV](#)

Domain	Status	Renewal status	Type	CNAME name
*.static-website-hosting.org	Success	-	CNAME	static-website-hosting.org
static-website-hosting.org	Success	-	CNAME	static-website-hosting.org

Now your SSL/TLS certificate is ready!

Custom SSL certificate - optional

Step 6: Complete CloudFront Distribution Setup

Enter Default Root Object

In the CloudFront distribution settings, you'll find a field called: Default Root Object

Why is this needed?

When someone visits your site (like <https://www.static-website-hosting.org>), they aren't requesting any specific file — just the root URL.

By setting `index.html` as the default, CloudFront knows to serve your homepage automatically.

What if you leave it blank?

- Users might see an Access Denied error.
- CloudFront won't know which file to load.
- Your site may not work correctly.

So it's important to always set this to `index.html` (or your homepage filename).

Default root object - optional

The object (file name) to return when a viewer requests the root URL (/) instead of a specific object.

Observation: Auto-Generated S3 Bucket Policy

After completing the CloudFront distribution setup, navigate back to your S3 bucket.

Initially:

- We did not manually create any bucket policy.

But Now:

- You'll notice that a new bucket policy has been automatically added.

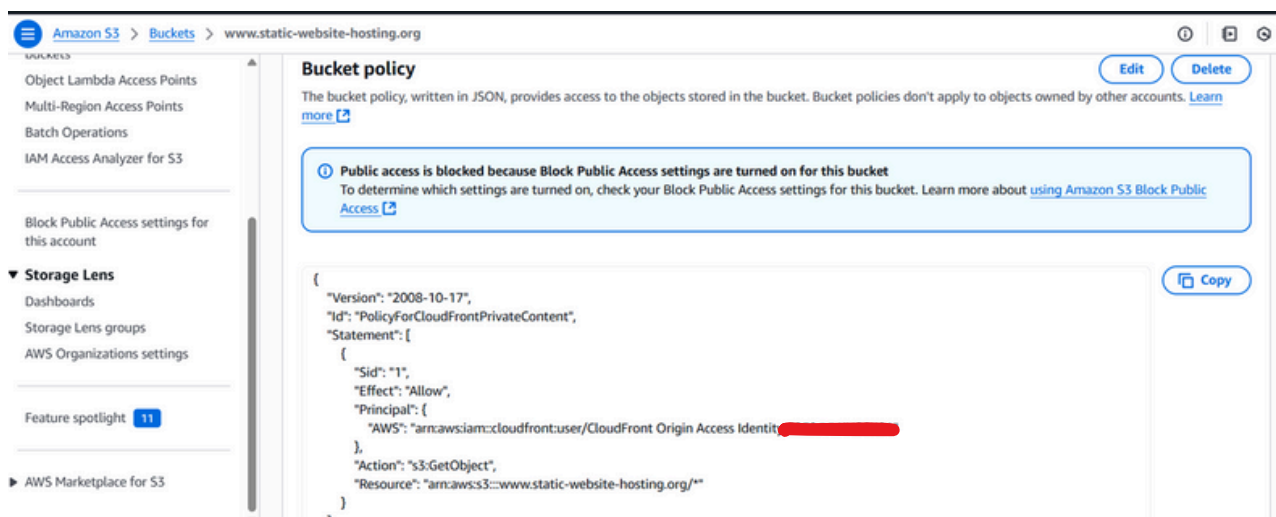
Why did this happen?

This was done when we:

Granted CloudFront permission to access the S3 bucket using an Origin Access Identity (OAI) during Step 4.

Purpose of this auto-generated policy:

- Allows only CloudFront (via OAI) to access your private S3 content.
- Ensures that the files remain secure and not publicly accessible by everyone.



The screenshot shows the Amazon S3 console interface. The left sidebar contains navigation links for 'BUCKETS', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', 'IAM Access Analyzer for S3', 'Block Public Access settings for this account', 'Storage Lens' (with sub-links for Dashboards, Storage Lens groups, and AWS Organizations settings), 'Feature spotlight', and 'AWS Marketplace for S3'. The main content area is titled 'Bucket policy' and shows the JSON policy for the bucket 'www.static-website-hosting.org'. A blue notification banner states: 'Public access is blocked because Block Public Access settings are turned on for this bucket. To determine which settings are turned on, check your Block Public Access settings for this bucket. Learn more about using Amazon S3 Block Public Access.' The JSON policy is as follows:

```
{
  "Version": "2008-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity [redacted]"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::www.static-website-hosting.org/*"
    }
  ]
}
```

Buttons for 'Edit', 'Delete', and 'Copy' are visible in the top right corner of the policy section.

DNS Routing Setup

Step 7: DNS Routing Using Route 53

What is DNS Routing?

DNS (Domain Name System) routing helps direct user traffic from your custom domain (like `www.static-website-hosting.org`) to the correct AWS service—in our case, the CloudFront distribution.

This makes sure that when someone visits your website, they're routed through the global CDN (CloudFront), which then pulls your content securely from S3.

Go to:

Route 53 > Hosted Zones > Select your domain > Create record

We will now create two A Records.

A Record 1: To route traffic from `www.static-website-hosting.org` to CloudFront.

- Record Name: `www`
- Record Type: A – IPv4 address
- Alias: Yes (ON)
- Alias Target: Select your CloudFront distribution

A Record 2: This handles users who visit just the root domain: `static-website-hosting.org`.

- Record Name: (Leave blank)
- Record Type: A – IPv4 address
- Alias: Yes (ON)
- Alias Target: Select the same CloudFront distribution

Route 53 > Hosted zones > static-website-hosting.org > Create record

Quick create record Switch to wizard

▼ Record 1 Delete

Record name Info .static-website-hosting.org

Record type Info A – Routes traffic to an IPv4 address and some AWS resources

Keep blank to create a record for the root domain.

☒ Alias

Route traffic to Info

Alias to CloudFront distribution

US East (N. Virginia)

An alias to a CloudFront distribution and another record in the same hosted zone are global and available only in US East (N. Virginia).

×

Routing policy Info Simple routing

Evaluate target health No

Route 53 > Hosted zones > static-website-hosting.org > Create record

Create record Info

Quick create record Switch to wizard

▼ Record 1 Delete

Record name Info .static-website-hosting.org

Record type Info A – Routes traffic to an IPv4 address and some AWS resources

Keep blank to create a record for the root domain.

☒ Alias

Route traffic to Info

Alias to CloudFront distribution

US East (N. Virginia)

An alias to a CloudFront distribution and another record in the same hosted zone are global and available only in US East (N. Virginia).

×

Routing policy Info Simple routing

Evaluate target health No

To make your custom domain fully functional and secure, you need both types of DNS records in Route 53:

CNAME Records

Used only once during SSL Certificate creation to validate domain ownership via AWS Certificate Manager (ACM). These are created automatically if you choose “Create records in Route 53” during certificate request.

A Records (Alias)

Used for actual traffic routing. These map your domain (like `www.static-website-hosting.org`) to your CloudFront distribution so users can access your static website.

Now your domain is fully connected to CloudFront, and your static website can be accessed securely using both versions:

<https://www.static-website-hosting.org/>

<https://static-website-hosting.org/>



GitHub Setup

Step 8: Setting Up GitHub for Deployment

Let's automate the website deployment using GitHub + AWS CodePipeline, so that any changes you push to GitHub are automatically deployed to your S3-hosted site.

1. Download Git (if not already installed)

2. Clone your GitHub repository locally

```
git clone https://github.com/your-username/your-repo-name.git
```

This creates a local folder with the same name as your repository.

3. Move into your project directory

```
cd your-repo-name
```

4. Copy your static website files into this directory

Replace the source path below with the actual location of your project files:

```
cp /path/to/your/local/website/files/* .
```

This places your website files (like index.html, style.css, etc.) directly in the root of the GitHub repository.

5. Confirm the files are present

```
ls
```

6. Add the files to Git for versioncontrol

```
git add .
```

7. Commit the changes with a message

```
git commit -m "Add initial static website files to root directory"
```

8. Push the changes to GitHub

```
git push origin main
```

This uploads your local changes to the GitHub server.

CI/CD PipeLine Setup

Step 9: Creating the CI/CD Pipeline with AWS CodePipeline

Now that your GitHub repository is ready and contains your website files, it's time to automate deployment using AWS CodePipeline. This will ensure that every time you push changes to GitHub, your static site is automatically updated on S3.

1. Go to AWS CodePipeline

Navigate to CodePipeline.

2. Click "Create pipeline" (Custom)

3. Pipeline settings

- Pipeline name: (Give your pipeline a meaningful name, e.g., static-site-deploy-pipeline)
- Default execution mode: Queued
- What it means: This mode allows pipeline executions to queue up if another execution is already in progress. It's useful when you don't want to interrupt ongoing deployments.

Choose pipeline settings

Step 3
Add source stage

Step 4
Add build stage

Step 5
Add test stage

Step 6
Add deploy stage

Step 7
Review

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
static-website-deploy-pipeline
No more than 100 characters

Execution mode [Info](#)
Choose the execution mode for your pipeline. This determines how the pipeline is run.
☐ Superseded
☒ Queued
☐ Parallel

Service role
☒ New service role
Create a service role in your account
☐ Existing service role
Choose an existing service role from your account

Role name
AWSCodePipelineServiceRole-us-east-1-static-website-deploy-pipe
Type your service role name
☒ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

4. Add source stage

- Source provider: GitHub
- Click "Connect to GitHub" to authenticate and select your repository.
- Choose the branch you want to deploy from (usually main).

The screenshot shows the 'Add source stage' step in the AWS CodePipeline console. The left sidebar lists steps from 1 to 7, with 'Add source stage' selected as Step 3. The main area is titled 'Add source stage' and 'Step 3 of 7'. Under the 'Source' section, 'GitHub (via OAuth app)' is selected as the source provider. A message prompts the user to grant AWS CodePipeline access to their GitHub repository. A 'Connect to GitHub' button is visible. A blue information box states that the OAuth app action is not recommended and suggests using the GitHub App action instead. At the bottom, there is a checkbox for 'Enable automatic retry on stage failure' which is checked.

The screenshot shows the 'Add source stage' step in the AWS CodePipeline console after successful configuration. The left sidebar shows 'Add source stage' as Step 3. The main area shows 'GitHub (via OAuth app)' as the selected source provider. The 'Connect to GitHub' button has been replaced by a 'Connected' button. A green success message box at the bottom states: 'You have successfully configured the action with the provider.' The breadcrumb navigation at the top reads: 'Developer Tools > CodePipeline > Pipelines > Create new pipeline'.

5. Skip Build & Test Stages

During the pipeline setup in AWS CodePipeline, you'll be asked to define Build and Test stages.

Since we are hosting a static website (HTML, CSS, JS — no backend code or server-side logic), we can safely skip these stages.

Why skip?

- No compilation required (like in React, Angular, or backend services).
- The website files (e.g., index.html, style.css) are already production-ready.
- Our goal is to copy files directly from GitHub to S3, not to build or test anything.

6. Add Deploy Stage

- Deploy provider: Amazon S3
- Select your target S3 bucket (the one that hosts your static site)

The screenshot shows the 'Add deploy stage' configuration interface in the AWS CodePipeline console. On the left, a sidebar lists the steps: 'Add test stage', 'Step 6: Add deploy stage' (selected), 'Step 7', and 'Review'. The main area is titled 'Deploy provider' and includes the following fields:

- Deploy provider:** A dropdown menu set to 'Amazon S3'.
- Region:** A dropdown menu set to 'United States (N. Virginia)'.
- Input artifacts:** A section with a dropdown menu and a 'SourceArtifact' tag. Below it, a text field contains 'www.static-website-hosting.org'.
- Bucket:** A text field with a search icon and a close icon, containing 'www.static-website-hosting.org'.
- Deployment path - optional:** An empty text field.
- Extract file before deploy:** A checked checkbox with the text 'The deployed artifact will be unzipped before deployment.'

7. Create the Pipeline

Click "Create pipeline" — and you're done!

The screenshot displays the AWS CodePipeline console interface for a pipeline named "static-website-deploy-pipeline". The breadcrumb navigation at the top reads "Developer Tools > CodePipeline > Pipelines > static-website-deploy-pipeline". The pipeline name is prominently displayed, followed by icons for a document and a gear. Action buttons include "Edit", "Stop execution", "Create trigger", "Clone pipeline", and a highlighted "Release change" button. Below these are tabs for "Pipeline", "Executions", "Triggers", "Settings", "Tags", and "Stage". The "Pipeline" tab is active, showing a visual representation of the pipeline stages. Two stages are visible: "Source" and "Deploy", connected by a right-pointing arrow. Each stage box shows a green checkmark icon, the stage name, and the text "All actions succeeded." The "Source" stage lists the action "Source" using the provider "GitHub (via OAuth app)", with a timestamp of "Just now". The "Deploy" stage lists the action "Deploy" using the provider "Amazon S3", with a timestamp of "1 minute ago". Both stage boxes include a redacted area and a "Source: Add initialia" link. On the left side of the pipeline view, there are controls for zooming in (+) and out (-), and a full-screen icon (⛶). Above the pipeline diagram, there are two green checkmark icons and an information icon (i).

Testing the CI/CD Pipeline

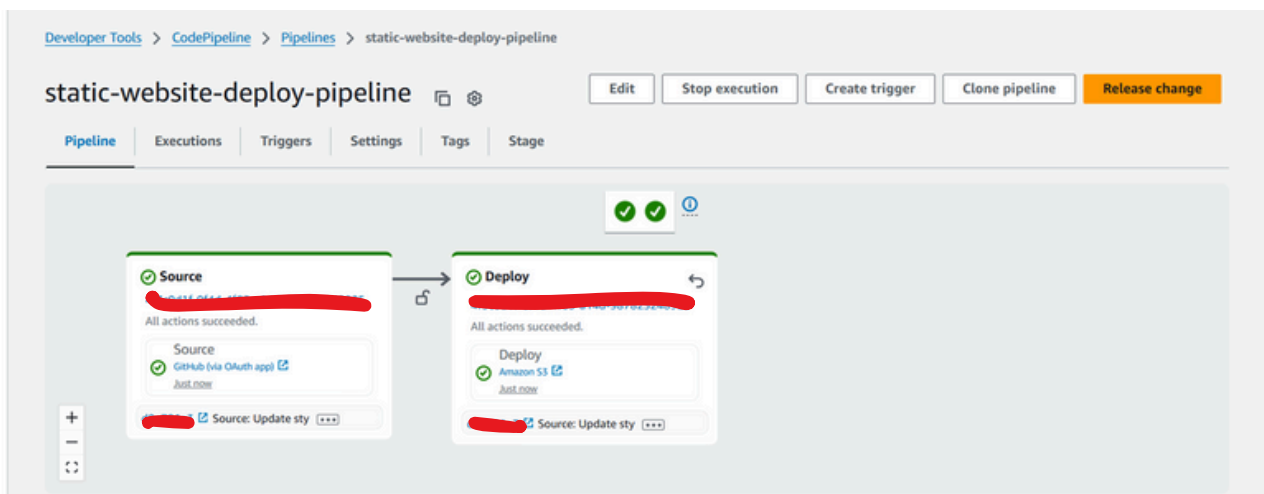
Now that your CI/CD pipeline is set up using GitHub + AWS CodePipeline, it's time to test if everything is working correctly!

Make a Code Change

Open your GitHub repository and make a small change — for example, edit index.html to update the heading or add a new line of text.

Watch AWS CodePipeline in Action

- Go to AWS Console → CodePipeline
- Open your pipeline
- You'll see the pipeline automatically detects the new GitHub commit
- It starts the deployment process and updates your S3 bucket



Visit Your Website

- Open your CloudFront domain (e.g., <https://www.static-website-hosting.org>) in the browser.
- You should see the updated content live — without any manual uploading!

How to Host a Static Website on AWS

[Home](#) [Steps](#)

1. **Register a Domain (Route 53):**
Buy a domain (e.g., `example.com`) using Route 53 or any registrar.
2. **Create S3 Buckets:**
 - Bucket 1: `example.com`
 - Bucket 2: `www.example.com`(Names must match your domain/subdomain.)
3. **Enable Static Website Hosting:**
 - In bucket properties, enable static website hosting
 - Set `index.html` as the root document
4. **Upload Website Files to S3:**
 - Upload `index.html`, `style.css`, etc.
5. **Set Bucket Access:**
 - Block public access (you'll allow access via CloudFront)
 - CloudFront will use an Origin Access Identity (OAI) for secure access
6. **Request SSL Certificate (ACM):**
 - Use ACM in `us-east-1`
 - Add DNS validation records via Route 53
 - Wait for certificate status to turn "Success"
7. **Create a CloudFront Distribution:**
 - Set S3 website endpoint as origin
 - Attach the SSL certificate from ACM
 - Enable "Redirect HTTP to HTTPS"
 - Set default root object to `index.html`
8. **Configure Route 53 DNS:**

Conclusion

This project successfully demonstrates how to host and deploy a secure, scalable, and highly available static website on AWS using services such as S3, CloudFront, Route 53, ACM, and CodePipeline.

It also integrates CI/CD from GitHub, enabling automation and version control.

This guide can serve as a reference for beginners or DevOps learners aiming to modernize website delivery using cloud-native solutions.

References

- <https://ankitjodhani.hashnode.dev/host-your-static-website-on-amazon-s3-services-cicd-pipeline-with-the-domain-name-and-ssl-certificate-10weeksofcloudops>
- <https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/>
- <https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>