

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 'Register domains' page in the AWS Route 53 console. The top navigation bar includes 'Route 53', 'Registered domains', and 'Register domains'. Below the navigation is a search bar with the placeholder 'static-website-hosting.org' and a 'Search' button. To the right is a 'Selected domains (0/5)' section with a 'Proceed to checkout' button. The main area contains sections for 'Search for domain', 'Standard pricing', and 'Search result'. The 'Search result' table has columns for 'Domain', 'Price/year', and 'Actions'. A single row is shown for 'static-website-hosting.org' with a price of '14.00 USD' and an action button labeled 'Select'.

Domain	Price/year	Actions
static-website-hosting.org	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' configuration page. At the top, it says 'Create bucket Info' and 'Buckets are containers for data stored in S3.' Below this is a 'General configuration' section. Under 'AWS Region', 'US East (N. Virginia) us-east-1' is selected. Under 'Bucket type', 'General purpose' is selected (indicated by a blue border). A tooltip for 'General purpose' states: '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.' There is also a 'Directory' option with a tooltip: '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.' At the bottom of the configuration section, there is a 'Bucket name' field containing 'www.static-website-hosting.org'. A note below it says: '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](#)'.

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.

Objects (3)

Copy S3 URI Copy URL Download Open Delete Actions Create folder

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

The screenshot shows the AWS S3 Properties tab for the bucket [www.static-website-hosting.org](#). The 'Properties' tab is selected. Under the 'Static website hosting' section, the 'Enabled' radio button is selected. The 'Hosting type' section shows 'Host a static website' selected. The 'Index document' field is set to 'index.html'. A note at the bottom states: '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](#)'.

Get the S3 Website Hosting Endpoint

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

The screenshot shows the AWS S3 console with the 'Static website hosting' tab selected. At the top, there's a note: '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](#)'. Below this, the 'S3 static website hosting' status is shown as 'Enabled'. The 'Hosting type' is 'Bucket hosting'. The 'Bucket website endpoint' is listed as <http://www.static-website-hosting.org.s3-website-us-east-1.amazonaws.com>.

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: 2VBJC7Y7NGGAKEW8
- HostId: Xa9R1X5GaZy9ZhlUszle+Cvi2xLSDnXCmTPuuqOJ6QpQemh3WI1ywxAu3rJOVMzCtLRo2YmDxDxK3blXAGYk9yTSe/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

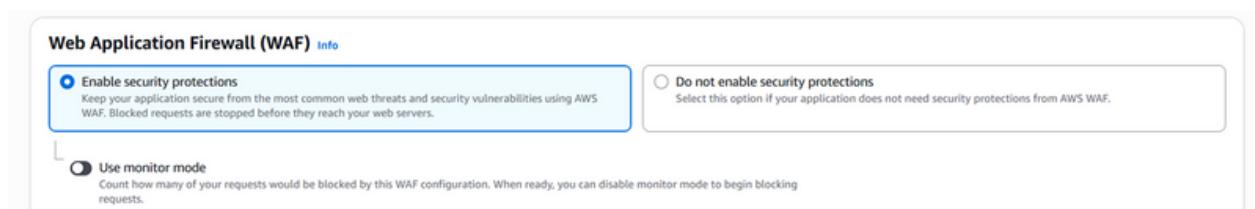
- HTTP and HTTPS**
- Redirect HTTP to HTTPS**
- HTTPS only**

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



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



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:

The screenshot shows the AWS Certificate Manager interface for requesting a public certificate. The top navigation bar includes 'AWS Certificate Manager', 'Certificates', 'Request certificate', and 'Request public certificate'. Below the navigation, a message says 'Provide one or more domain names for your certificate.' A 'Fully qualified domain name' field contains two entries: '*.static-website-hosting.org' and 'static-website-hosting.org', each with a 'Remove' button. At the bottom of the form, there is a link 'Add another name to this certificate' and a note: 'You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name.'

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)				
Domain	Status	Renewal status	Type	CNAME name
*.static-website-hosting.org	Success	-	CNAME	 [REDACTED] hosting.org.
static-website-hosting.org	Success	-	CNAME	 [REDACTED] hosting.org.

Now your SSL/TLS certificate is ready!

You can go back to CloudFront and attach this certificate to your distribution to enable secure HTTPS browsing for your users.

Custom SSL certificate - optional

Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

*.static-website-hosting.org  [REDACTED]

*.static-website-hosting.org  Request certificate 

Legacy clients support - \$600/month prorated charge applies. Most customers do not need this.
CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS.

Security policy

The security policy determines the SSL or TLS protocol and the specific ciphers that CloudFront uses for HTTPS connections with viewers (clients).

- TLSv1.2_2021 (recommended)
- TLSv1.2_2019
- TLSv1.2_2018
- TLSv1.1_2016
- TLSv1_2016
- TLSv1

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 AWS S3 Bucket policy page for the bucket 'www.static-website-hosting.org'. The left sidebar includes links for Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings for this account, Storage Lens (Dashboards, Storage Lens groups, AWS Organizations settings), Feature spotlight (11 notifications), and AWS Marketplace for S3. The main content area is titled 'Bucket policy' and contains the following JSON policy:

```
{
  "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/*"
    }
  ]
}
```

A blue 'Copy' button is located next to the policy text. At the top right of the main content area are 'Edit' and 'Delete' buttons. Below the policy text, a note 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](#)'.

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

Record 1

Record name Info	.static-website-hosting.org	Record type Info
www	A – Routes traffic to an IPv4 address and some AWS resources	
Keep blank to create a record for the root domain.		
<input checked="" type="radio"/> 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).		
<input type="text"/> Q [REDACTED] X		
Routing policy Info	Evaluate target health	
Simple routing	<input type="radio"/> No	

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

Create record [Info](#)

Quick create record

Record 1

Record name Info	static-website-hosting.org	Record type Info
subdomain	A – Routes traffic to an IPv4 address and some AWS resources	
Keep blank to create a record for the root domain.		
<input checked="" type="radio"/> 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).		
<input type="text"/> Q [REDACTED] X		
Routing policy Info	Evaluate target health	
Simple routing	<input type="radio"/> 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.

The screenshot shows the 'Pipeline settings' step of the AWS CodePipeline creation wizard. On the left, a sidebar lists steps from 'Step 3' to 'Review'. The main area has tabs for 'Pipeline settings' and 'Stages'. Under 'Pipeline settings', there are sections for 'Pipeline name' (set to 'static-website-deploy-pipeline'), 'Execution mode' (set to 'Queued'), and 'Service role'. The 'Service role' section shows two options: 'New service role' (selected) and 'Existing service role'. A 'Role name' field contains 'AWSCodePipelineServiceRole-us-east-1-static-website-deploy-pipe'. A checked checkbox at the bottom says '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).

Step 1 Choose creation option

Step 2 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

Add source stage Info

Step 3 of 7

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (via OAuth app)

Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.

Connect to GitHub

The GitHub (via OAuth app) action is not recommended
The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (via GitHub App) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. [Learn more](#)

Enable automatic retry on stage failure

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose creation option

Step 2 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

Add source stage Info

Step 3 of 7

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (via OAuth app)

Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.

Connected

Success You have successfully configured the action with the provider. X

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

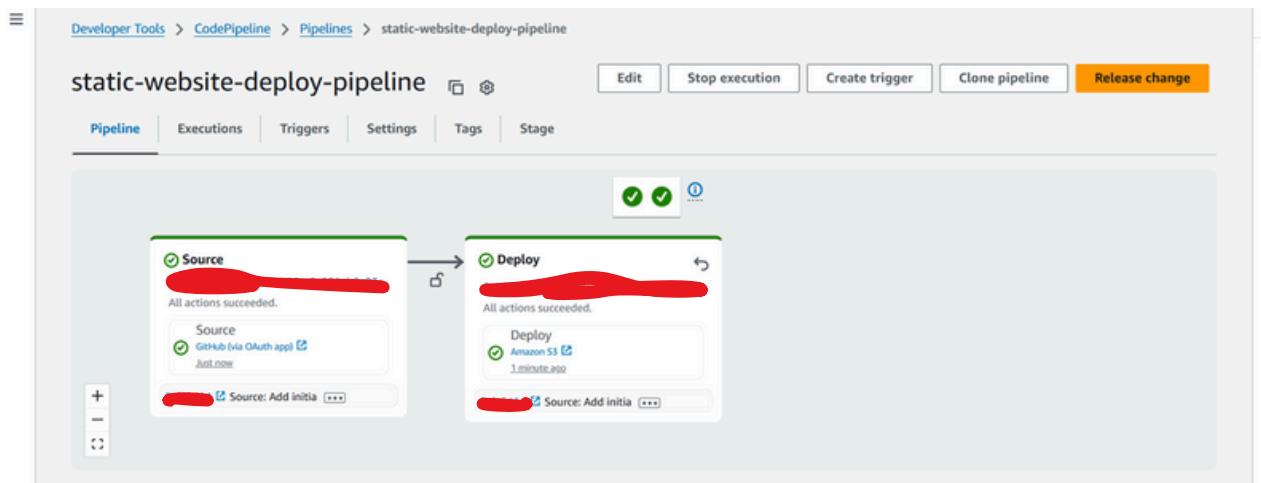
- Deploy provider: Amazon S3
- Select your target S3 bucket (the one that hosts your static site)

The screenshot shows the 'Add deploy stage' configuration screen in AWS CodePipeline. On the left, there's a sidebar with 'Step 6' (Add deploy stage) and 'Step 7' (Review). The main area has the following fields:

- Deploy provider:** Set to "Amazon S3".
- Region:** Set to "United States (N. Virginia)".
- Input artifacts:** A dropdown menu showing "SourceArtifact" (selected) and "Defined by: Source". Below it is a note: "No more than 100 characters".
- Bucket:** A search bar containing "www.static-website-hosting.org".
- Deployment path - optional:** An empty input field.
- Extract file before deploy:** A checked checkbox with the note: "The deployed artifact will be unzipped before deployment."

7. Create the Pipeline

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



Testing the CI/CD Pipeline

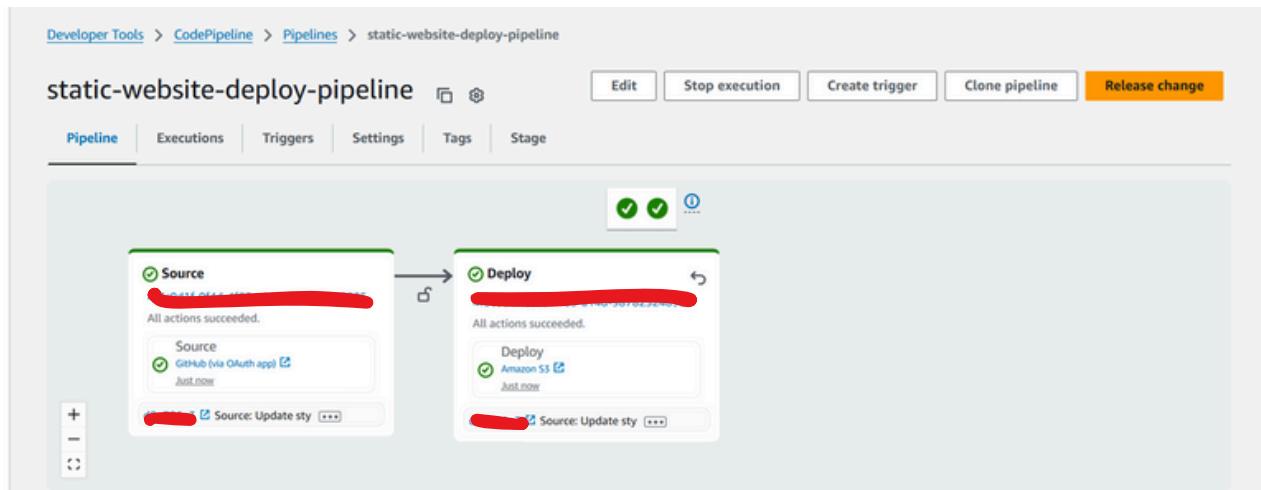
Now that your CI/CD pipeline is set up using GitHub + AWS CodePipeline, it's time to test if everything is working correctly!

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!

The screenshot shows a web page titled "How to Host a Static Website on AWS". At the top, there is a navigation bar with links for "Home" and "Steps". Below the title, the main content area lists eight steps with sub-points:

- 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>