1. Configure VPC peering in cross regions.
2. Task Title

**Configure Cross-Region VPC Peering in AWS Console**

2. Objective

**To establish network connectivity between two VPCs located in different AWS regions, allowing private communication using their private IP addresses.**

3. Prerequisites

**\* Two AWS VPCs created in \*\*different regions\*\*.**

**\* Each VPC must have \*\*non-overlapping CIDR blocks\*\*.**

**\* IAM user/role with permissions for \*\*EC2, VPC, Peering Connections\*\*.**

**\* Knowledge of the \*\*VPC IDs\*\* and \*\*CIDR ranges\*\* for both regions.**

4. Step-by-Step Implementation

Step 1: Create a VPC Peering Request

**1. Open \*\*AWS Management Console → VPC\*\*.**

**2. In the left menu, select \*\*Peering Connections\*\*.**

**3. Click \*\*Create Peering Connection\*\*.**

**4. Enter the following:**

**\* \*\*Peering connection name\*\***

**\* \*\*Requester VPC\*\* (VPC in Region A)**

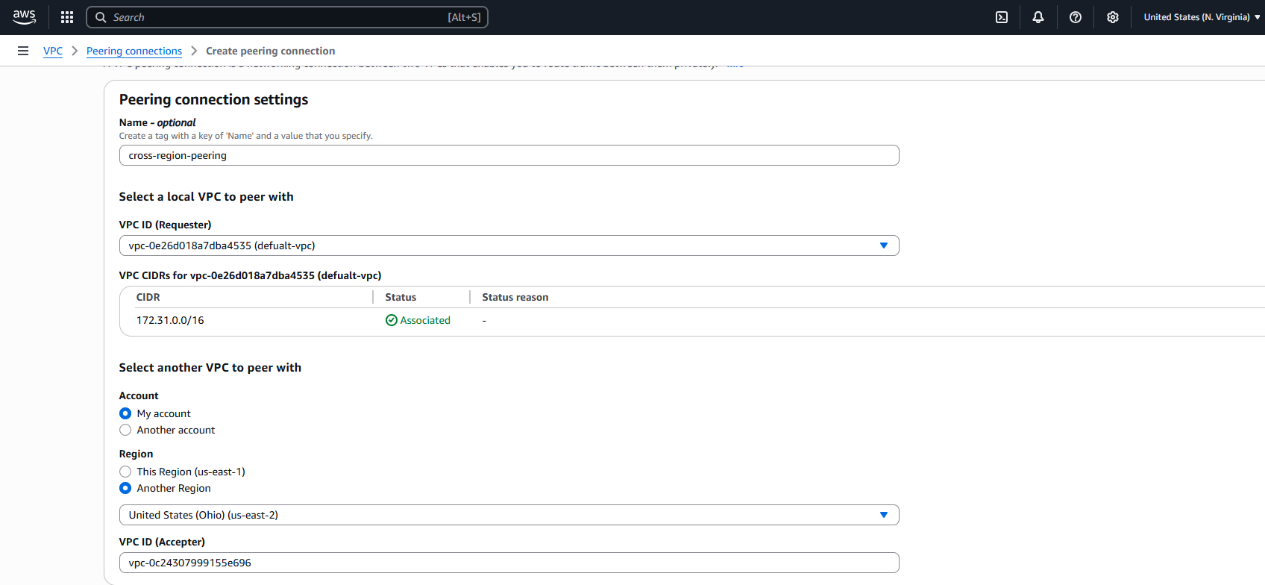
**\* \*\*Accepter VPC\*\***

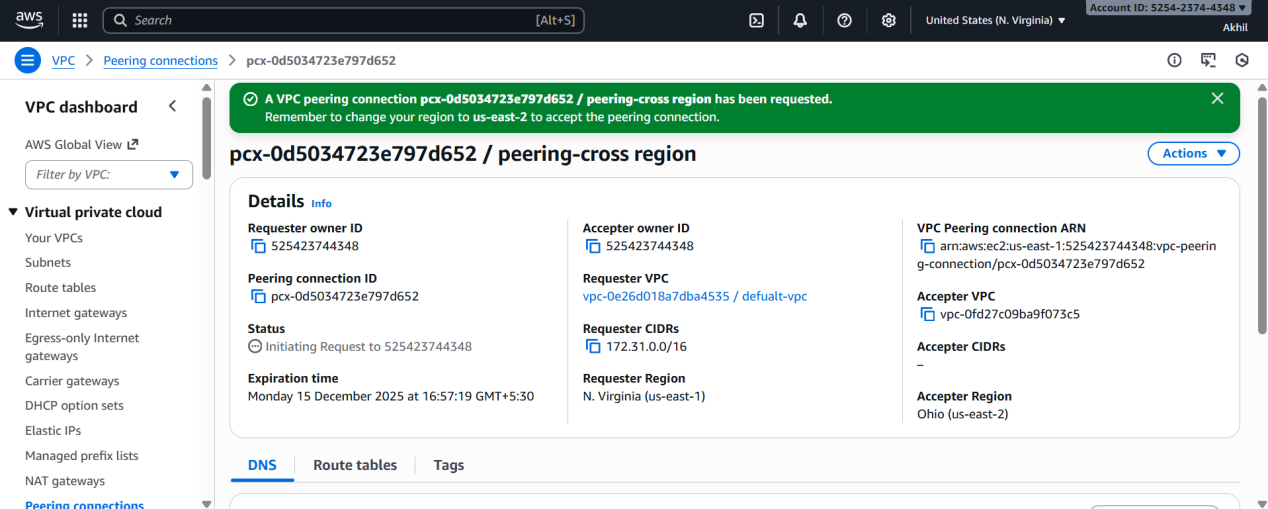
**\* Select \*\*Another Region\*\***

**\* Choose the target \*\*Region B\*\***

**\* Select the \*\*Accepter VPC ID\*\***

1. **Click \*\*Create Peering Connection**





Step 2: Accept the Peering Request in the Other Region\*\*

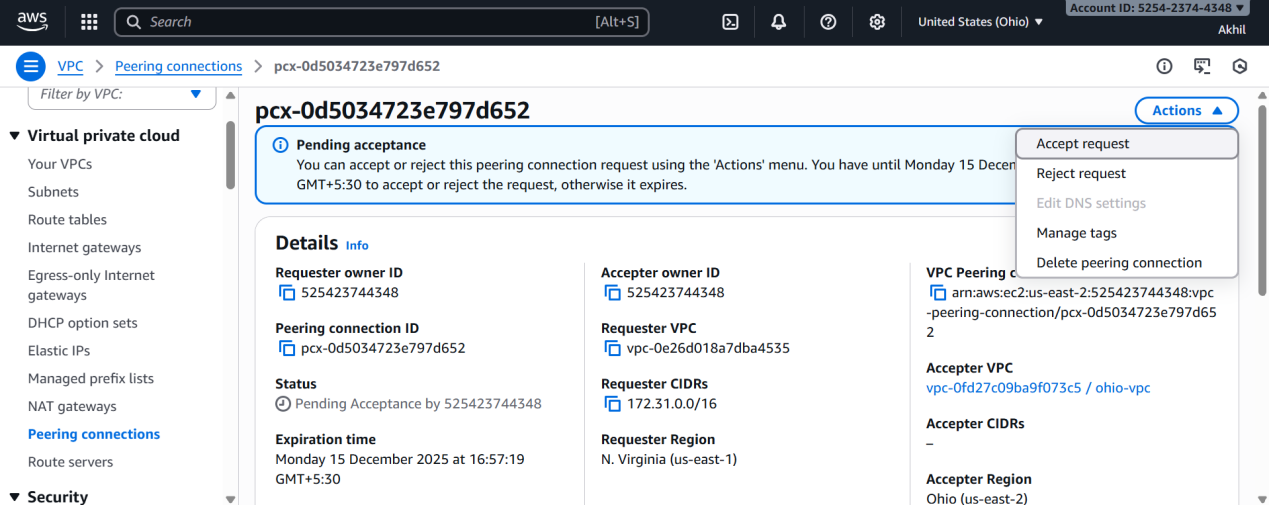
**1. Switch to \*\*Region B\*\* in the AWS Console.**

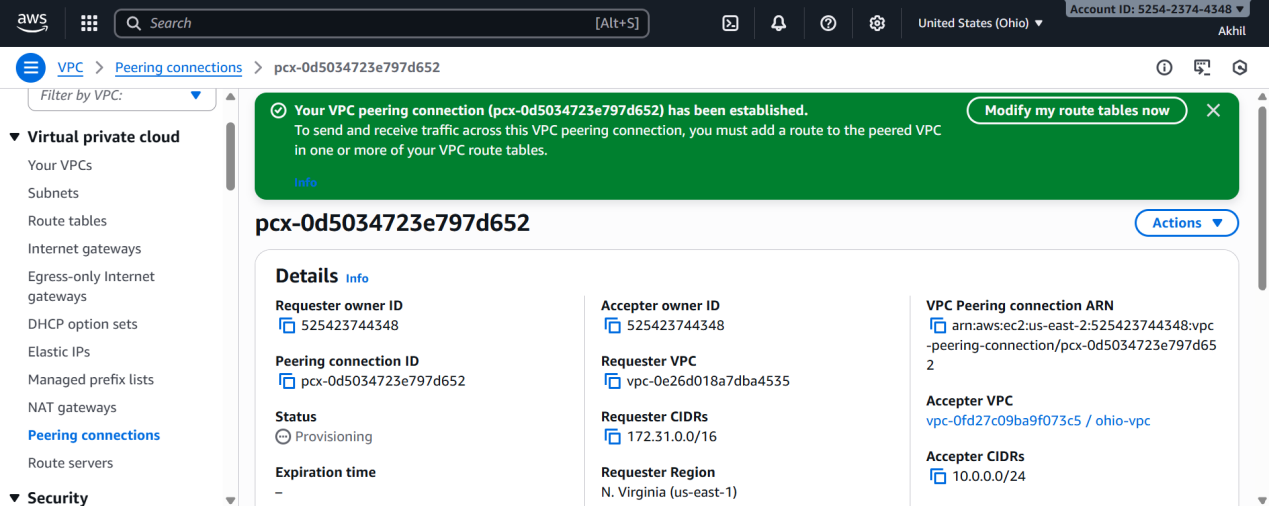
**2. Go to \*\*VPC → Peering Connections\*\*.**

**3. Select the pending peering request.**

**4. Click \*\*Actions → Accept Request\*\*.**

**5. Confirm acceptance.**





Step 3: Update Route Tables in Both VPCs

**You must add routes so both VPCs know how to reach each other.**

**#### \*\*In VPC A:\*\***

**1. Go to \*\*Route Tables\*\*.**

**2. Select the route table associated with subnets that should communicate.**

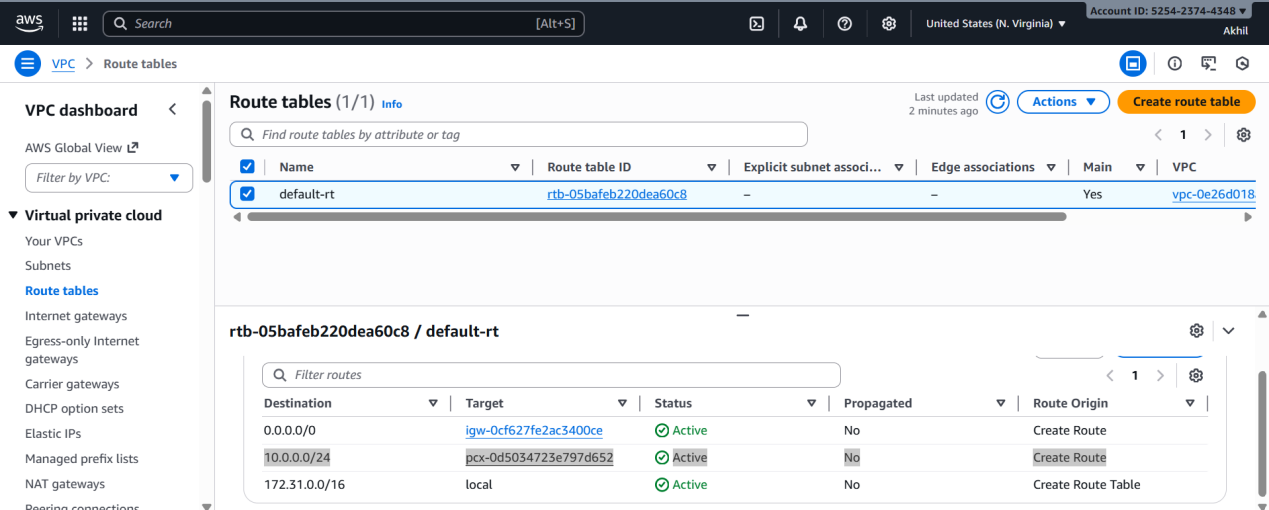
**3. Click \*\*Edit Routes → Add Route\*\*.**

**4. Enter:**

**\* \*\*Destination:\*\* CIDR of VPC B**

**\* \*\*Target:\*\* The Peering Connection**

1. **Save the routes.**

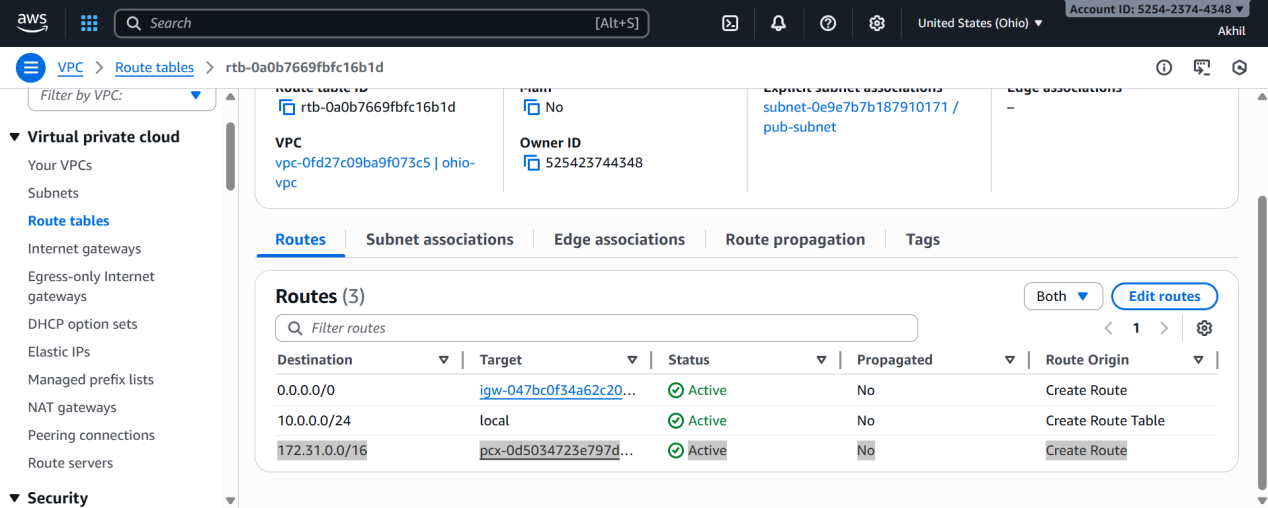


**#### \*\*In VPC B (Region B):\*\***

**Repeat the same steps but:**

**\* \*\*Destination:\*\* CIDR of VPC A**

**\* \*\*Target:\*\* Same peering connection**



Step 4: Update Security Groups (Optional but Required for Traffic)\*\*

**1. In both regions, go to \*\*EC2 → Security Groups\*\*.**

**2. Modify inbound rules:**

**\* Add a rule allowing required traffic (e.g., SSH, HTTP, ICMP).**

**\* \*\*Source:\*\* CIDR of the other VPC.**

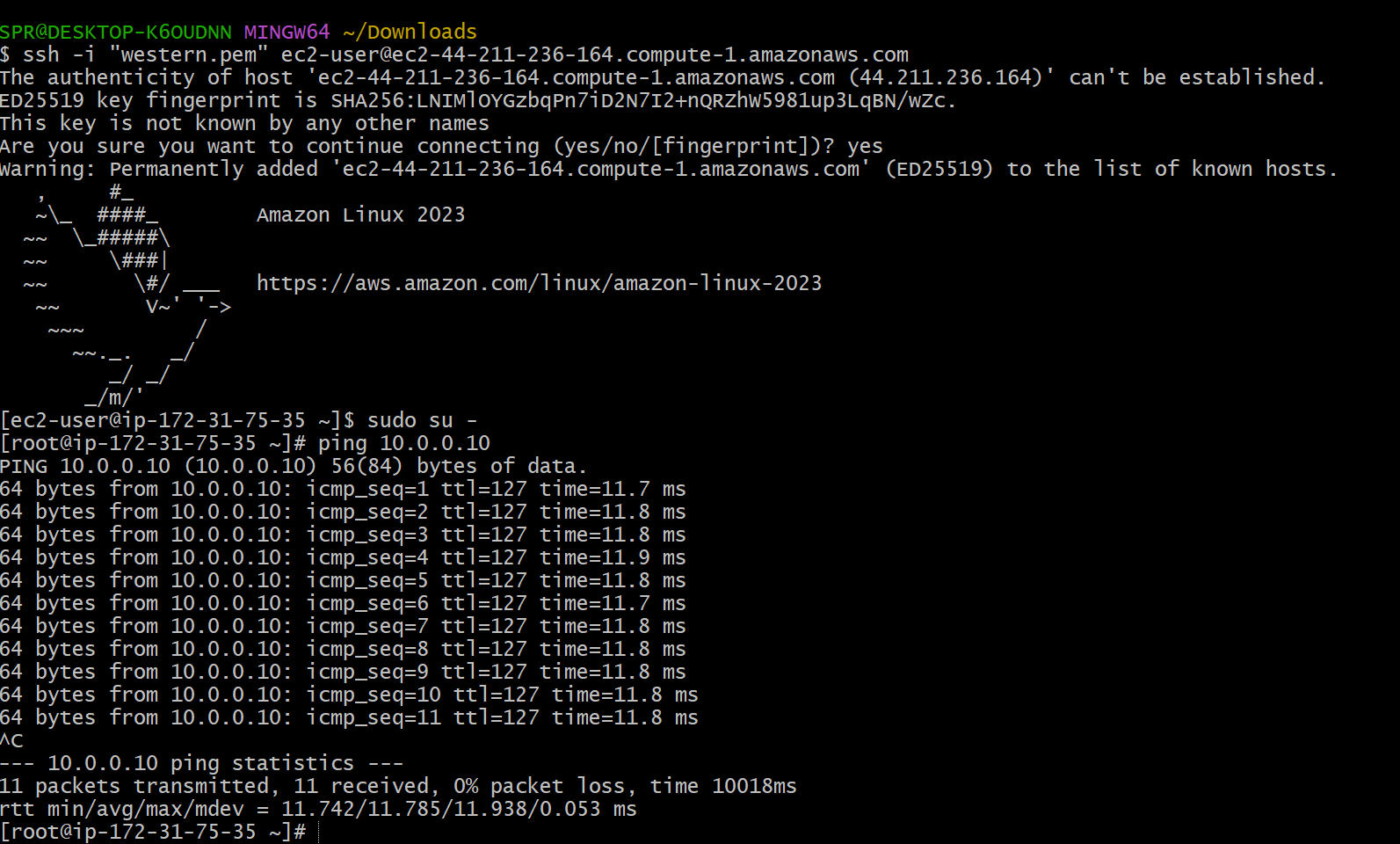
5. Validation Steps

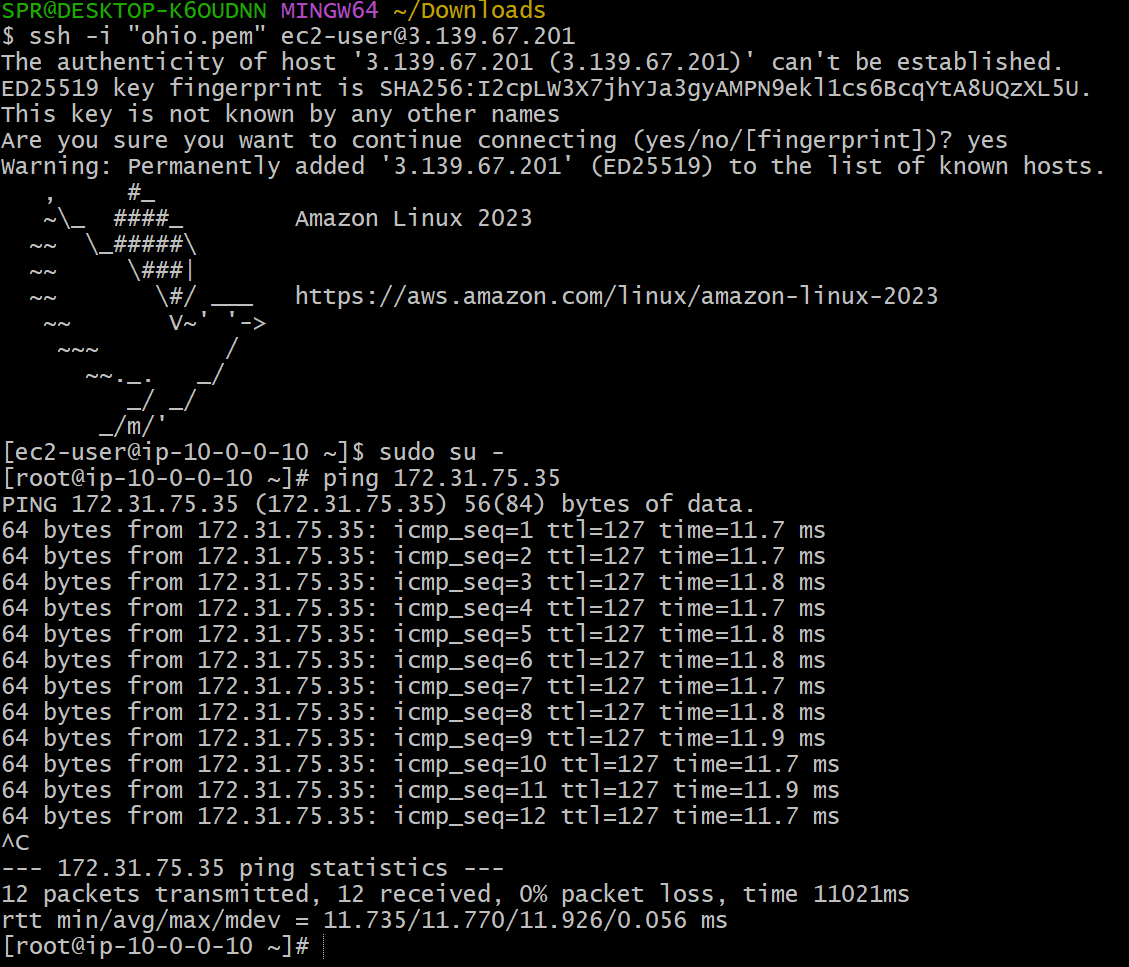
**\* Launch EC2 instances in both VPCs.**

**\* Ping (if ICMP allowed) or SSH using \*\*private IP addresses\*\* between instances.**

**\* Ensure the traffic flows through VPC peering.**

**\* Confirm that there is \*\*no NAT Gateway or Internet\*\* involvement.**





6. Conclusion

**Cross-region VPC peering was successfully configured, enabling secure private communication between VPCs in two different regions. The route tables and security groups confirm end-to-end connectivity.**

1. Purchase one domain from GoDaddy.
2. Deploy static website in S3.

**Step 1: Create an S3 Bucket\*\***

1. Go to \*\*AWS Console → S3\*\*

2. Click \*\*Create bucket\*\*

3. Enter a unique bucket name (e.g., \*my-static-site-123\*)

4. \*\*Region:\*\* Choose one near you

5. \*\*Uncheck\*\*: "Block all public access"

6. Confirm that the bucket will become public

7. Click \*\*Create bucket\*\*



**Step 2: Enable Static Website Hosting\*\***

1. Open your newly created bucket

2. Go to \*\*Properties\*\* tab

3. Scroll to \*\*Static website hosting\*\*

4. Click \*\*Edit\*\* → Enable

5. Choose \*\*Host a static website\*\*

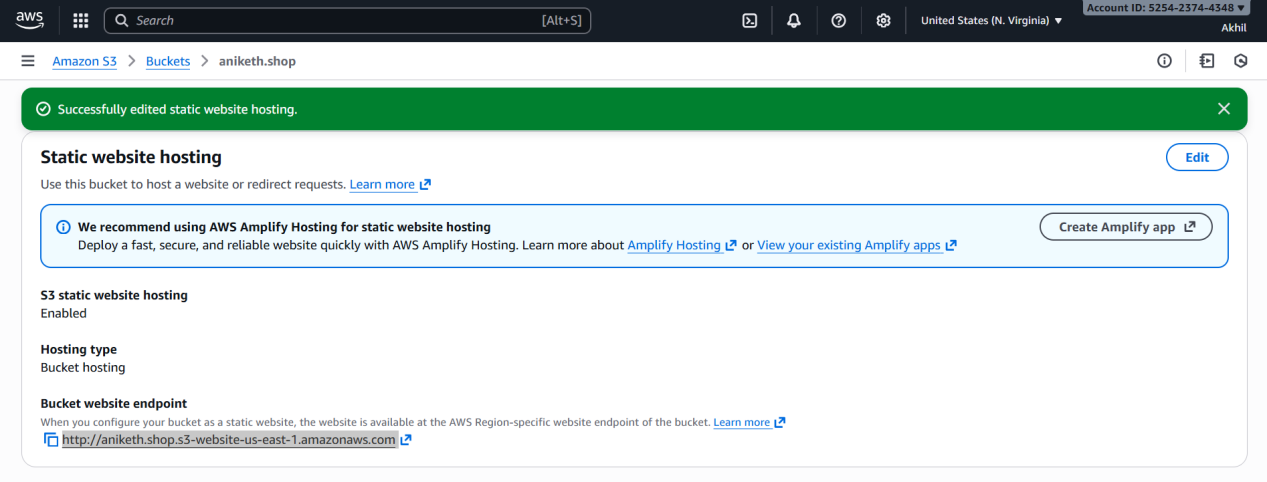
6. Specify:

\* Index document: `index.html`

\* Error document: `error.html` (optional)

7. Save changes

8. Note down the \*\*Endpoint URL\*\* — this is your website link.



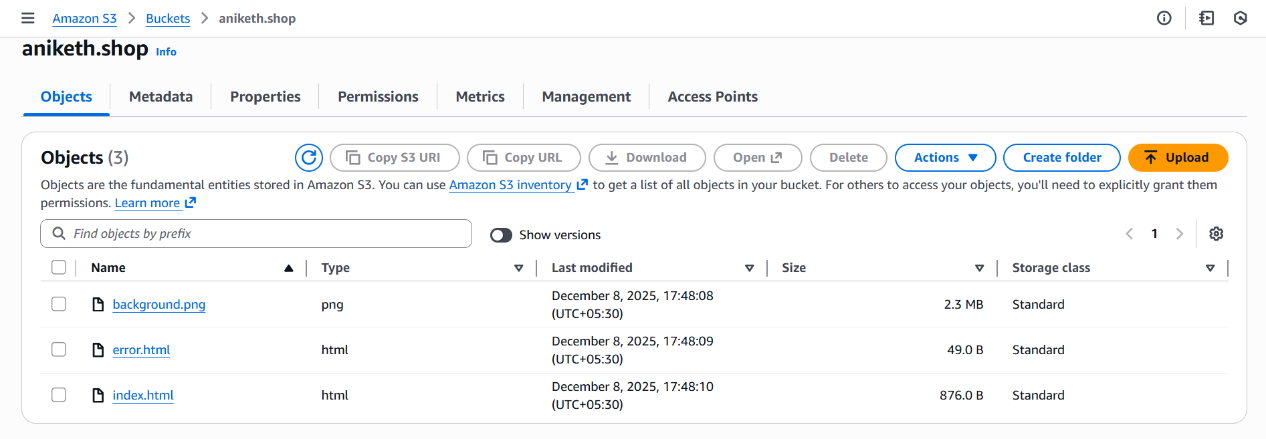
**Step 3: Upload Your Website Files\*\***

1. Go to \*\*Objects\*\* tab

2. Click \*\*Upload\*\*

3. Upload all your static files (HTML/CSS/JS/images)

4. Click \*\*Upload\*\*



**Step 4: Make Files Public\*\***

To allow public access, you need to update permissions.

### \*\*Option A — Bucket Policy (recommended)\*\*

Go to \*\*Permissions → Bucket policy\*\* and paste this:

```json

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::YOUR\_BUCKET\_NAME/\*"

}

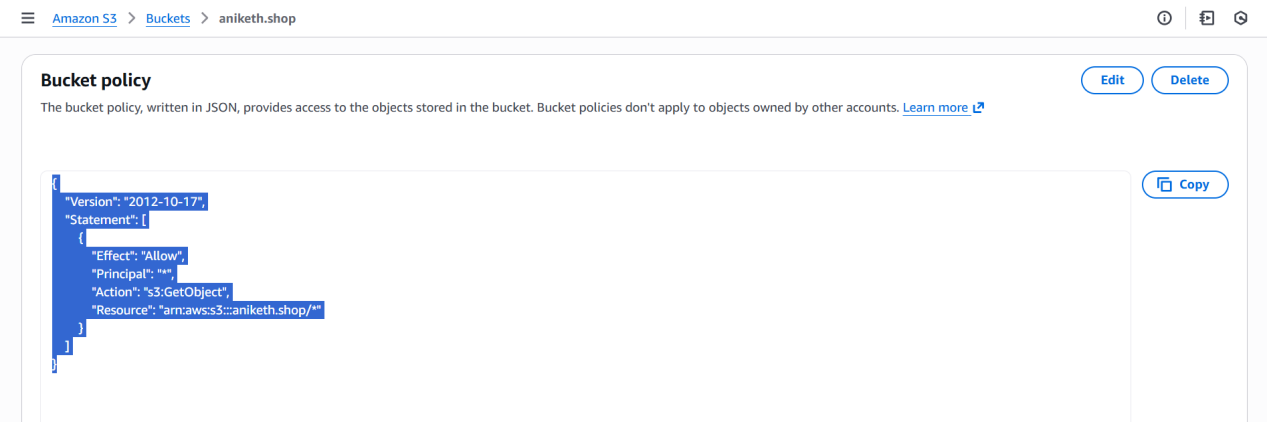
]

}

```

Replace `YOUR\_BUCKET\_NAME` with your actual bucket name.

Save the policy.



**Step 5: Test Your Website\*\***

Open the \*\*S3 website endpoint\*\*, which looks like:

http://YOUR\_BUCKET\_NAME.s3-website-YOUR-REGION.amazonaws.com

Your site should now load publicly.

1. Create a CDN and attach one SSL certificate.

**1. Task Title\*\***

\*\*Create a CDN (CloudFront Distribution) and Attach an SSL Certificate\*\*

**2. Objective\*\***

To create a CloudFront CDN for content delivery and secure the distribution using an SSL certificate for HTTPS access.

**3. Prerequisites\*\***

\* An existing \*\*S3 bucket\*\*, \*\*ALB\*\*, or \*\*Custom Origin\*\* to serve as the CDN’s origin.

\* A valid \*\*domain name\*\* managed in \*\*Route 53\*\* or another DNS provider.

\* An \*\*SSL certificate issued in AWS Certificate Manager (ACM – us-east-1 region)\*\*.

\* IAM permissions for \*\*CloudFront\*\*, \*\*ACM\*\*, and \*\*Route 53\*\* (if applicable).

**4. Step-by-Step Implementation\*\***

### \*\*Step 1: Request or Import an SSL Certificate in ACM\*\*

1. Open \*\*AWS Console → Certificate Manager (ACM)\*\*.

2. Ensure the region is \*\*us-east-1\*\* (mandatory for CloudFront).

3. Click \*\*Request a Certificate → Request a Public Certificate\*\*.

4. Add your domain name (e.g., `cdn.example.com`).

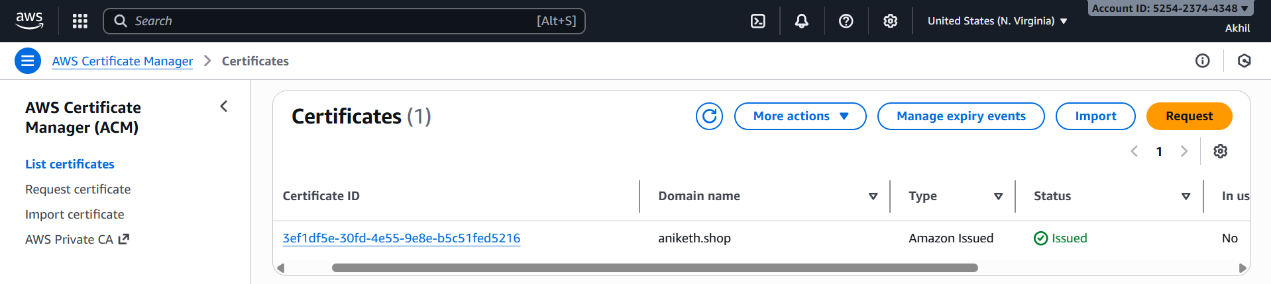
5. Validate using:

\* \*\*DNS validation\*\* (recommended), or

\* Email validation

6 .After this we need to create records in route53 - hosted zones with (cname) provided in certificate.

1. Wait for certificate status to become \*\*Issued\*\*.



####Step 2: Create a CloudFront Distribution

1. Go to \*\*AWS Console → CloudFront\*\*.

2. Click \*\*Create Distribution\*\*.

3. Under \*\*Origin Settings\*\*:

\* \*\*Origin Domain\*\*: Choose your S3 bucket, ALB, or custom origin.

\* \*\*Origin Protocol Policy\*\*: Select \*\*HTTPS Only\*\* (recommended).

### \*\*Step 3: Configure Default Cache Behavior\*\*

1. Set \*\*Viewer Protocol Policy\*\* to:

\* \*\*Redirect HTTP to HTTPS\*\* (recommended).

2. Leave caching settings as default or adjust based on your needs.

3. Under \*\*Allowed HTTP Methods\*\*, choose appropriate options (e.g., GET/HEAD).

### \*\*Step 4: Configure SSL Certificate\*\*

1. Under \*\*Settings → Custom SSL Certificate\*\*:

\* Choose \*\*"Custom SSL Certificate (example.com)"\*\*.

2. Select the ACM certificate you issued earlier.

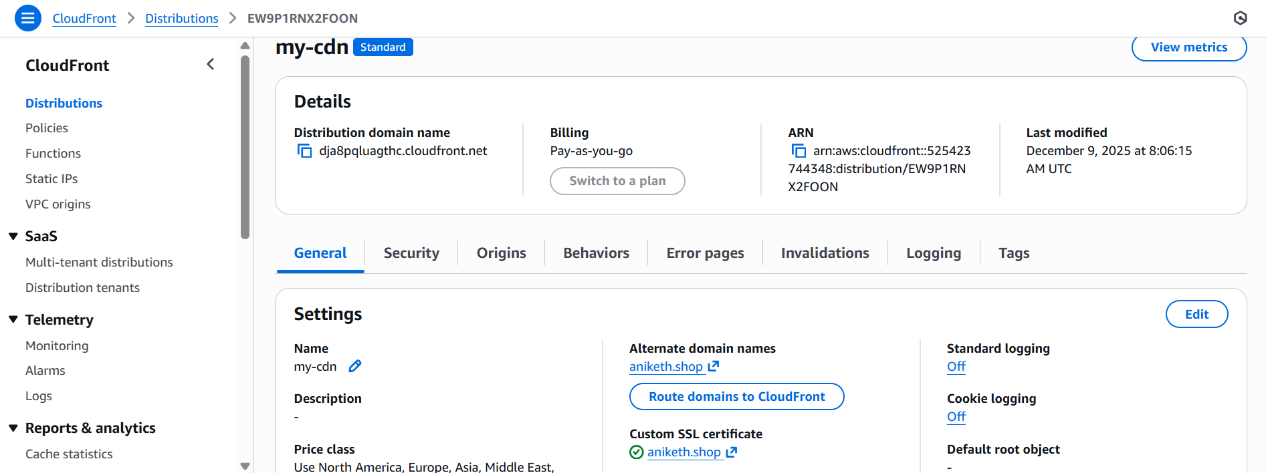
3. Enter a \*\*CNAME\*\* matching your domain (e.g., `cdn.example.com`).

4. Set supported security protocols and minimum TLS version as required.

### \*\*Step 5: Create the Distribution\*\*

1. Scroll to the bottom and click \*\*Create Distribution\*\*.

2. Wait 10–20 minutes for CloudFront to deploy globally.



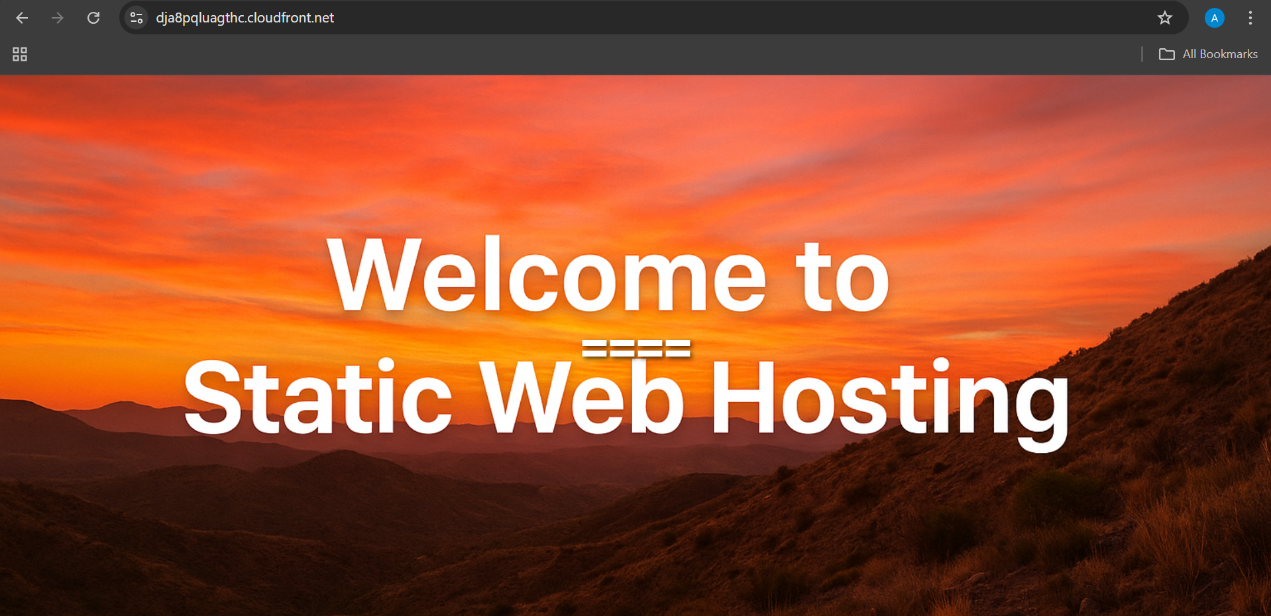
**5. Validation Steps\*\***

\* Open your CloudFront domain or custom domain (`cdn.example.com`).

\* Ensure it loads over \*\*HTTPS\*\* using the SSL certificate.

\* Check CloudFront distribution status shows \*\*Deployed\*\*.

\* Review headers in the browser (Developer Tools → Network) to verify content is served via CloudFront.



**6. Conclusion\*\***

A CloudFront CDN has been successfully created and secured with an SSL certificate. Your content is now delivered globally with low latency and HTTPS encryption.

1. Create a Route 53 hosted zone and map the domain with the CDN.

**1. Task Title\*\***

\*\*Create a Route 53 Hosted Zone and Map the Domain to a CloudFront CDN\*\*

**2. Objective\*\***

To create a public hosted zone in Route 53 for DNS management and configure a domain or subdomain to point to a CloudFront distribution.

**3. Prerequisites\*\***

\* A registered domain name (e.g., from GoDaddy, Namecheap, etc.).

\* An active \*\*CloudFront Distribution\*\*.

\* SSL certificate issued in \*\*ACM (us-east-1)\*\* if using HTTPS.

\* Access to your domain provider's DNS panel (if domain is external).

\* Permissions for \*\*Route 53\*\* and \*\*CloudFront\*\*.

**4. Step-by-Step Implementation\*\***

### \*\*Step 1: Create a Public Hosted Zone\*\*

1. Go to \*\*AWS Console → Route 53\*\*.

2. Click \*\*Hosted Zones → Create Hosted Zone\*\*.

3. Enter your domain name (e.g., `example.com`).

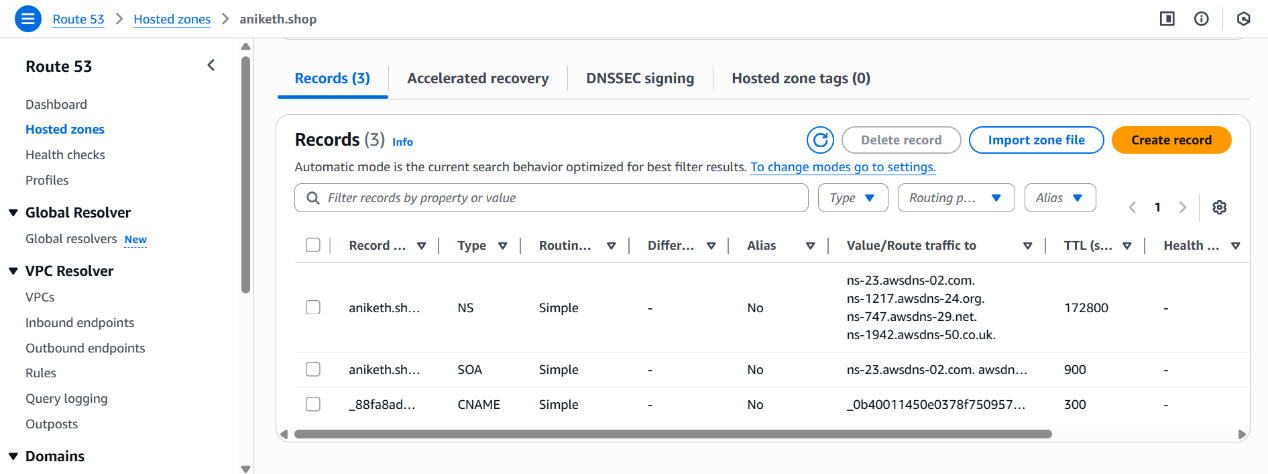
4. Select \*\*Public Hosted Zone\*\*.

5. Click \*\*Create Hosted Zone\*\*.

This creates:

\* NS (Name Server) records

\* SOA record



### \*\*Step 2: Update Name Servers in Your Domain Provider\*\*

If your domain is NOT in Route 53 (e.g., GoDaddy):

1. Open the hosted zone you created.

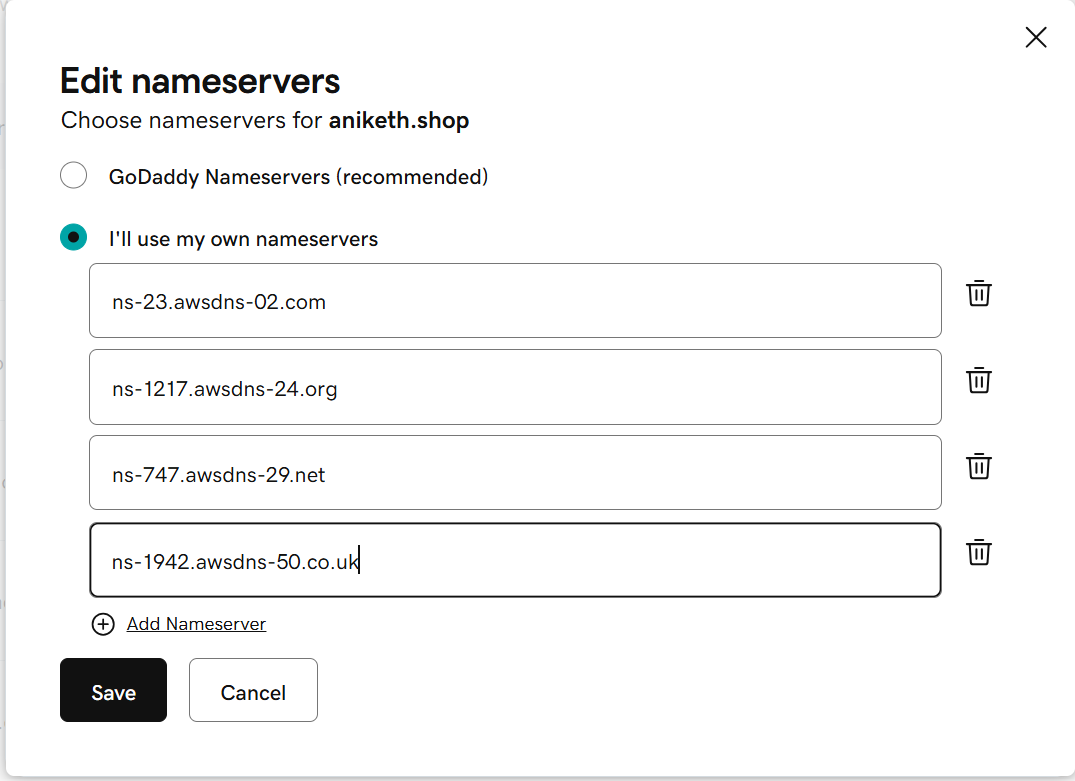
2. Copy the \*\*4 NS records\*\*.

3. Go to GoDaddy → DNS Settings.

4. Replace existing NS with the 4 Route 53 NS entries.

5. Save changes.

DNS propagation may take \*\*10–30 minutes\*\* (sometimes 24 hrs).



### \*\*Step 3: Create an Alias Record to CloudFront\*\*

Once the hosted zone is active:

1. Go to \*\*Route 53 → Hosted Zone → Create record\*\*.

2. Choose \*\*Simple Routing\*\*.

3. Click \*\*A Record\*\*.

4. Enable \*\*Alias → Yes\*\*.

5. For alias target, choose \*\*CloudFront Distribution\*\* from the dropdown.

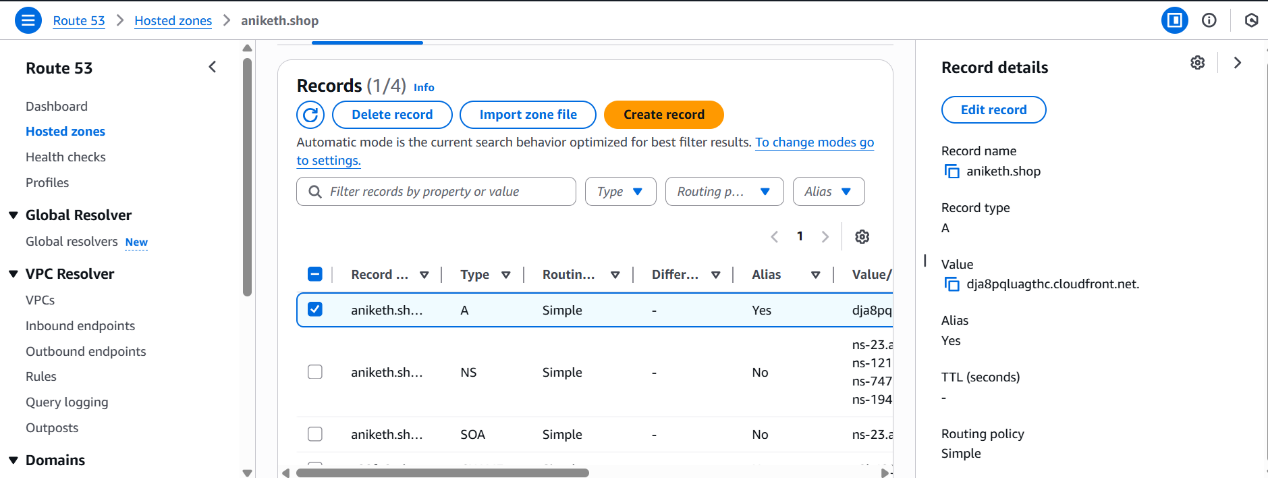
6. Enter your custom domain:

Examples:

\* ‘aniketh.shop’

7. Click \*\*Create Record\*\*.

This maps your domain to CloudFront.



### \*\*Step 4: Update CloudFront Distribution (If not done already)\*\*

1. Go to \*\*CloudFront → Your Distribution → Settings\*\*.

2. Under \*\*Alternate Domain Names (CNAMEs)\*\*, add:

aniketh.shop

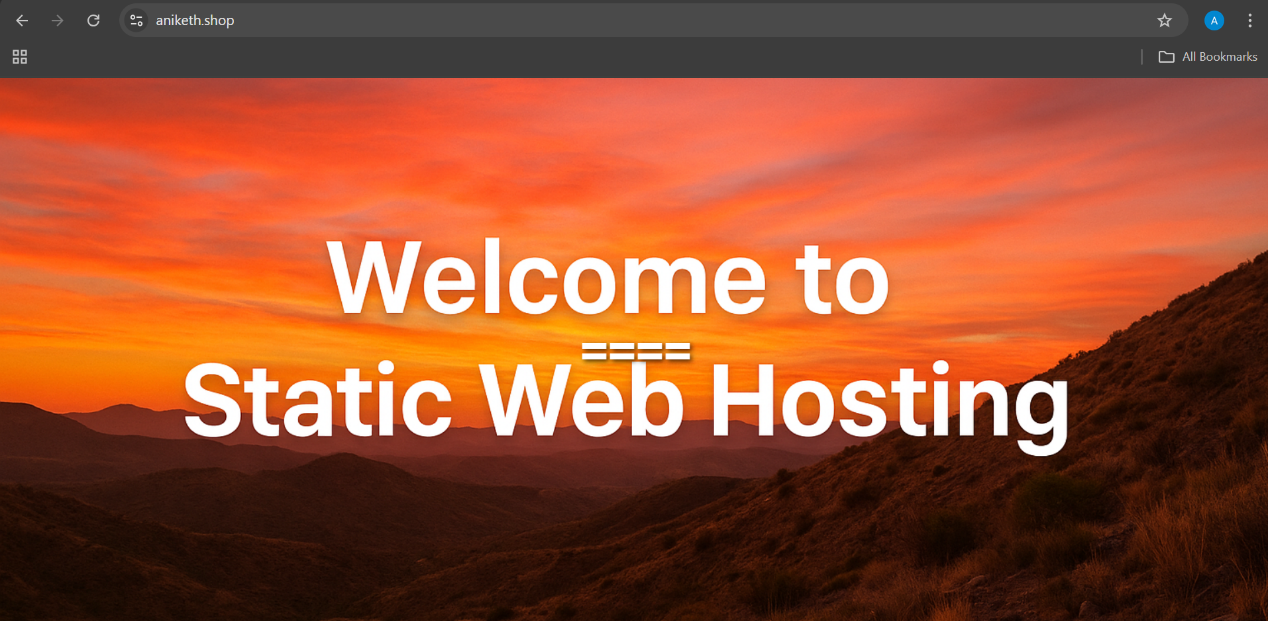
3. Attach your \*\*ACM SSL certificate\*\*.

4. Save and let CloudFront deploy.

**5. Validation Steps\*\***

\* Open your domain:

https://aniketh.shop



**6. Conclusion\*\***

A Route 53 hosted zone was successfully created and your domain was mapped to the CloudFront CDN using an Alias A record. Your domain now delivers content securely through CloudFront with global performance optimization.

1. Update the index.html in the S3 bucket and ensure the updated file is accessible using the domain name.

**1. Task Title\*\***

Update `index.html` in S3 and make it accessible via the domain name.

**2. Objective\*\***

To upload the updated `index.html` file to the S3 bucket and ensure it is publicly accessible through the domain (e.g., \*\*aniketh.shop\*\*).

**3. Prerequisites\*\***

\* An S3 bucket with \*\*Static Website Hosting enabled\*\*.

\* Updated `index.html` file ready for upload.

\* Bucket policy allowing \*\*public read\*\* access.

\* Domain (aniketh.shop) already mapped to \*\*CloudFront\*\* or directly to the S3 website endpoint.

\* AWS Console access.

**4. Step-by-Step Implementation\*\***

### Step 1: Open the S3 Bucket\*\*

\* Go to \*\*AWS Console → S3\*\*.

\* Open the bucket hosting your static website.

### Step 2: Upload Updated `index.html`\*\*

\* Open the \*\*Objects\*\* tab.

\* Click \*\*Upload\*\* → select the updated `index.html`.

\* Click \*\*Upload\*\* to overwrite the existing file.

### Step 3: Ensure the File is Public\*\*

\* If your bucket is already public, continue.

\* If not, apply this bucket policy (update bucket name if needed):

```json

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::aniketh.shop/\*"

}

]

}

```

### Step 4: Clear CloudFront Cache (If Applicable)\*\*

If your domain uses CloudFront:

\* Open \*\*CloudFront → Distributions\*\*.

\* Select your distribution → go to \*\*Invalidations\*\*.

\* Create an invalidation for:

```

/\*

```

This forces CloudFront to fetch the new `index.html`.

**5. Validation Steps\*\***

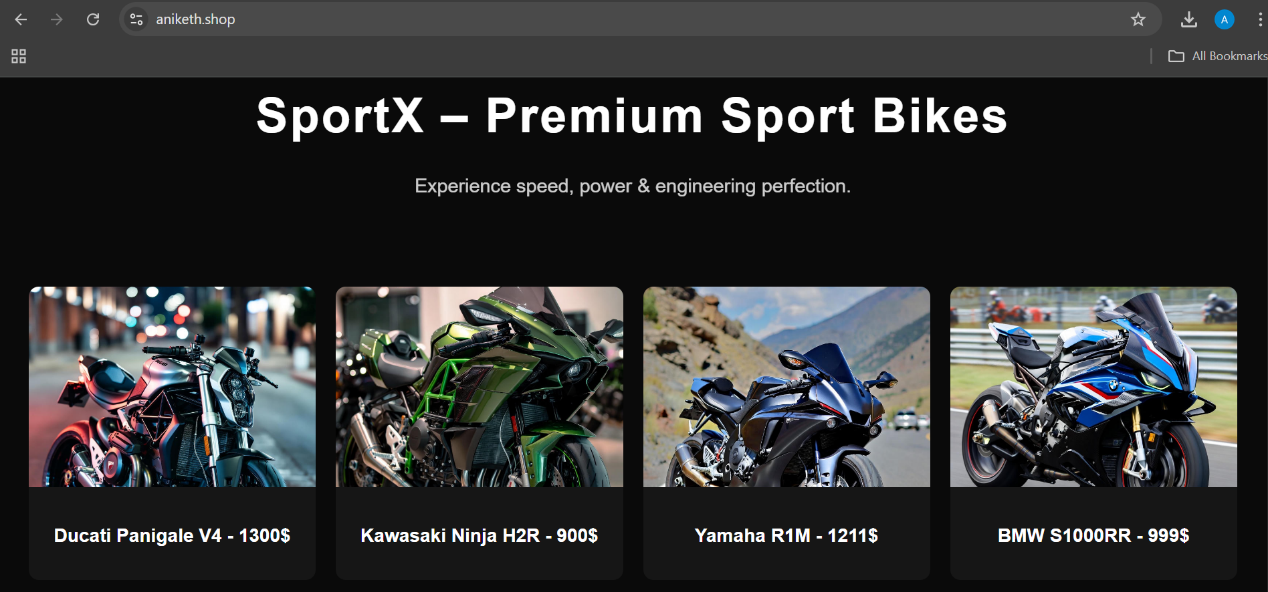
\* Open your domain in a browser:

\*\*[https://aniketh.shop](https://aniketh.shop)\*\*

\* Perform a hard refresh (Ctrl + F5 or Cmd + Shift + R).

\* Confirm the updated `index.html` content appears.

\* Optionally, test the S3 website endpoint directly to verify file update.



**6. Conclusion\*\***

The updated `index.html` file is uploaded successfully, and the domain now serves the latest version of your website page.

1. Share the domain name in Slack to test the connectivity

**===== Aniketh.shop =====**

R53