# R53 , CDN TASKS

|  |  |
| --- | --- |
| **Task Title** | **Deploy and Secure a Static Website (mujju.store) using Amazon S3, CloudFront & Route 53** |
| **Objective** | To host a static website with HTTPS encryption and global content delivery using AWS services mapped to GoDaddy-registered domain. |

**Prerequisites (Exact Requirements)**

| **Prerequisite** | | **Description** |
| --- | --- | --- |
| AWS Account | Access to AWS Management Console | |
| Registered Domain | **mujju.store** from GoDaddy | |
| IAM Permissions | Required: S3, CloudFront, Route 53, ACM | |
| Website Files | index.html uploaded to S3 | |
| S3 Bucket Name | Must match domain name: mujju.store | |
| DNS Control | Able to update nameservers on GoDaddy | |
| SSL Region Requirement | ACM certificate must be requested in **us-east-1** | |
| Public Access Config | S3 bucket policy + static hosting enabled | |
| Network | Stable internet to upload & verify | |

**Tools / Services Used**

| **Service / Tool** | **Purpose** | **Scope / Region** |
| --- | --- | --- |
| Amazon S3 | Static website Web Hosting | eu-north-1 |
| CloudFront | CDN + HTTPS delivery | Global |
| ACM (Certificate Manager) | SSL certificate (HTTPS) | us-east-1 |
| Route 53 | DNS & Nameserver hosting | Global |
| GoDaddy | Domain registrar | Registrar Only |
| Chrome Browser | Website functionality validation | Local |

**Steps Performed (High-Level Summary)**

| **Step No.** | **Action Completed** |
| --- | --- |
|  | Created S3 bucket mujju.store + enabled static hosting |
|  | Uploaded index.html |
|  | Configured public bucket access + policy |
|  | Generated SSL certificate using ACM |
|  | Completed DNS validation via GoDaddy |
|  | Created Route 53 public hosted zone |
|  | Updated GoDaddy nameservers → AWS |
|  | Created CloudFront distribution with HTTPS |
|  | Added A-Record (Alias→CloudFront) in Route 53 |
|  | Created cache invalidation for latest deployment |

**Validation Steps (Mandatory)**

| **Validation Type** | **Expected Result** |
| --- | --- |
| Resource Health | S3 website endpoint loads successfully |
| SSL Status | Certificate status = **Issued** |
| CloudFront Deployment | Status = **Deployed** |
| DNS Propagation | mujju.store resolves globally |
| HTTPS Enforcement | Secure padlock visible in browser |
| CDN Effectiveness | Invalidation success + fast load |

**Issues Faced**

| **Issue** | **Cause** | **Fix** |
| --- | --- | --- |
| CloudFront didn’t show updated site | Old cache | Invalidated /\* |

**Conclusion**

| **Summary** |
| --- |
| Deployment successfully completed — Website now loads securely via **https://mujju.store** using CloudFront CDN, with DNS fully managed by Route 53 and validated SSL certificate. |

1. Enable VPC peering for cross-region.

**A. Launch pub-server in Stockholm (eu-north-1)**

1. In the AWS console, region is set to **Europe (Stockholm)**.
2. Go to **EC2 → Launch instance**.
3. Under **Name and tags**, set:
   * **Name** = pub-server

**B. Launch pub-two-hyd in Hyderabad (ap-south-2)**

1. Change region in the top-right to **Asia Pacific (Hyderabad)**.
2. Go to **EC2 → Launch instance** again.
3. Under **Name and tags**, set:
   * **Name** = pub-two-hyd

**C. Create cross-region VPC peering (Stockholm → Hyderabad)**

1. Switch back to **Europe (Stockholm)** region.
2. Go to **VPC → Peering connections**.
3. Click **Create peering connection**.
4. Fill **Peering connection settings**:
   * **Name** = peering-01
5. In **Select a local VPC to peer with**:
   * **VPC ID (Requester)** = vpc-068d558547ab00f25 (my-vpc-01)
   * This VPC has CIDR 172.168.0.0/24.
6. In **Select another VPC to peer with**:
   * **Account** = *My account*
   * **Region** = **Another Region** → choose **Asia Pacific (Hyderabad) (ap-south-2)**
   * **VPC ID (Accepter)** = vpc-09197ad95e1cc0cce (the default VPC in Hyderabad).
7. Click **Create peering connection**.
   * A new peering pcx-0f3aff59a02e7a4a2 is created with status **Pending acceptance**.

**D. Accept the peering request in Hyderabad**

1. Switch region (top-right) to **Asia Pacific (Hyderabad)**.
2. Go to **VPC → Your VPCs** and confirm you have VPC:
   * vpc-09197ad95e1cc0cce with CIDR 172.31.0.0/16.
3. In the left menu, click **Peering connections**.
4. You see the peering connection from Stockholm with status **Pending acceptance**.
5. Select that peering connection row.
6. Click **Actions → Accept request**.
7. A confirmation popup **“Accept VPC peering connection request”** shows:
   * **Requester VPC** = vpc-068d558547ab00f25, Region = Stockholm, CIDR 172.168.0.0/24
   * **Accepter VPC** = vpc-09197ad95e1cc0cce, Region = Hyderabad, CIDR 172.31.0.0/16
8. Click **Accept request**.
9. The peering connection status changes to **Active**.

* **Update route tables** in both VPCs:
  + In **Stockholm VPC (172.168.0.0/24)**: add a route to 172.31.0.0/16 with target = the **peering connection**.
  + In **Hyderabad VPC (172.31.0.0/16)**: add a route to 172.168.0.0/24 with target = the **same peering connection**.

**Check CIDR Block of Hyderabad VPC**

Region: **Asia Pacific (Hyderabad)**

1. Go to **VPC → Your VPCs**
2. Verify CIDR: 172.31.0.0/16

**Edit Route Table in Hyderabad Region (Accepter VPC)**

Region: **Asia Pacific (Hyderabad)**  
 VPC: vpc-09197ad95e1cc0cce → CIDR: 172.31.0.0/16

1. Go to **VPC → Route tables**
2. Select **Main route table**
3. Click **Routes → Edit routes**
4. Add new route:
   * **Destination**: 172.168.0.0/24
   * **Target**: Select **Peering Connection** (pcx-0f3aff59a02e7a4a2)
5. Click **Save changes**

This allows return traffic from Hyderabad to Stockholm.

**Check Private IP of Hyderabad EC2 instance**

Region: **Hyderabad**

1. EC2 Dashboard → Instances
2. Select instance **pub-server-hyd**
3. Note the **Private IP**:
   * 172.31.35.127

**Test Connectivity from Stockholm EC2 → Hyderabad EC2**

Connected via SSH to **Stockholm EC2** (172.168.0.6)  
 Run ping command:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen with a white background

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen with a red arrow pointing to a white box

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

2.Purchase one domain from GoDaddy.

A screenshot of a phone

AI-generated content may be incorrect.

3.Deploy static website in S3.

* **Upload Website Files to S3**
* Open the S3 bucket → **Objects** tab
* Click **Upload**
* Click **Add files**
* Select your website files → index.html and error.html
* Confirm destination is your bucket → **techieeee-s3**
* Click **Upload**
* **Make Files Public**
* Select both index.html and error.html
* Click **Actions** → **Make public using ACL**  
   This allows the public to view your website pages.
* **Enable Static Website Hosting**
* Go to bucket → **Properties** tab
* Scroll to the **Static website hosting** section
* Choose:
  + **Enable**
  + **Hosting type: Bucket hosting**
  + **Index document: index.html**
  + **Error document: error.html**
* Click **Save changes**
* **Access Your Website**
* Copy the **Bucket website endpoint** URL  
  Example:
* http://techieeee-s3.s3-website.eu-north-1.amazonaws.com
* Paste in browser
* **Website Error Page Works**
* If a user opens a wrong URL  
  → It redirects to **error.html**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Index.html

A screenshot of a computer

AI-generated content may be incorrect.

1. Create a CDN and attach one SSL certificate.
2. **Created an S3 bucket**

* Navigated to **Amazon S3 → Create Bucket**
* Entered bucket name: **mujju.store**
* Selected Region: **eu-north-1 (Stockholm)**
* Left default settings (ACLs disabled, versioning disabled)
* Clicked **Create Bucket**

1. **Uploaded website content**

* Opened the bucket **mujju.store**
* Uploaded the file **index.html** into the Objects section

1. **Enabled Static Website Hosting**

* Went to **Properties tab → Static website hosting**
* Selected **Enable**
* Chose **Host a static website**
* Entered **index.html** as the Index document
* Saved changes
* Verified website works using:
* http://mujju.store.s3-website.eu-north-1.amazonaws.com

(Not secure → HTTP only)

1. **Requested an SSL Certificate**

* Navigated to **AWS Certificate Manager (ACM)**
* Selected **Region: us-east-1 (N. Virginia)**
* Chose **Request a public certificate**
* Entered domain:
* mujju.store
* DNS validation selected
* Submitted request (status: Pending validation)

1. **Added DNS Records in GoDaddy for Certificate Validation**

* Logged in to GoDaddy Domain DNS settings
* Added the required **CNAME** record(s) provided by ACM  
  → To verify domain ownership

A screenshot of a chat

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

|  |
| --- |
|  |

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

* Opened **Route 53 → Get Started**  
   Selected **Create Hosted Zone**  
   Created a **Public Hosted Zone** for domain:
* mujju.store
* Hosted zone was created with default **NS** and **SOA** records

1. **GoDaddy Domain Configuration**

* Went to GoDaddy → DNS → Nameservers tab  
   Selected **I’ll use my own nameservers**  
   Replaced GoDaddy nameservers with AWS Route 53 nameservers:
* ns-856.awsdns-43.net
* ns-482.awsdns-60.com
* ns-1906.awsdns-46.co.uk
* ns-1121.awsdns-12.org
* Saved → Domain DNS now fully managed by **Route 53**

1. **CloudFront Setup**

* Opened **CloudFront → Create Distribution**  
   Entered distribution name:
* mujju.store
* Configured domain via **Route 53 integration**  
   Set **S3 static website endpoint** as origin  
   Added **Alternate domain name (CNAME)**:
* mujju.store
* Selected **ACM SSL certificate** for HTTPS

1. **Route 53 DNS Record for CloudFront**

* Returned to Route 53 hosted zone  
   Created an **A record → Alias → CloudFront distribution**  
   Saved record

1. **CloudFront Cache Management**

* Created **Invalidation**
* /\*

→ To refresh website content globally

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

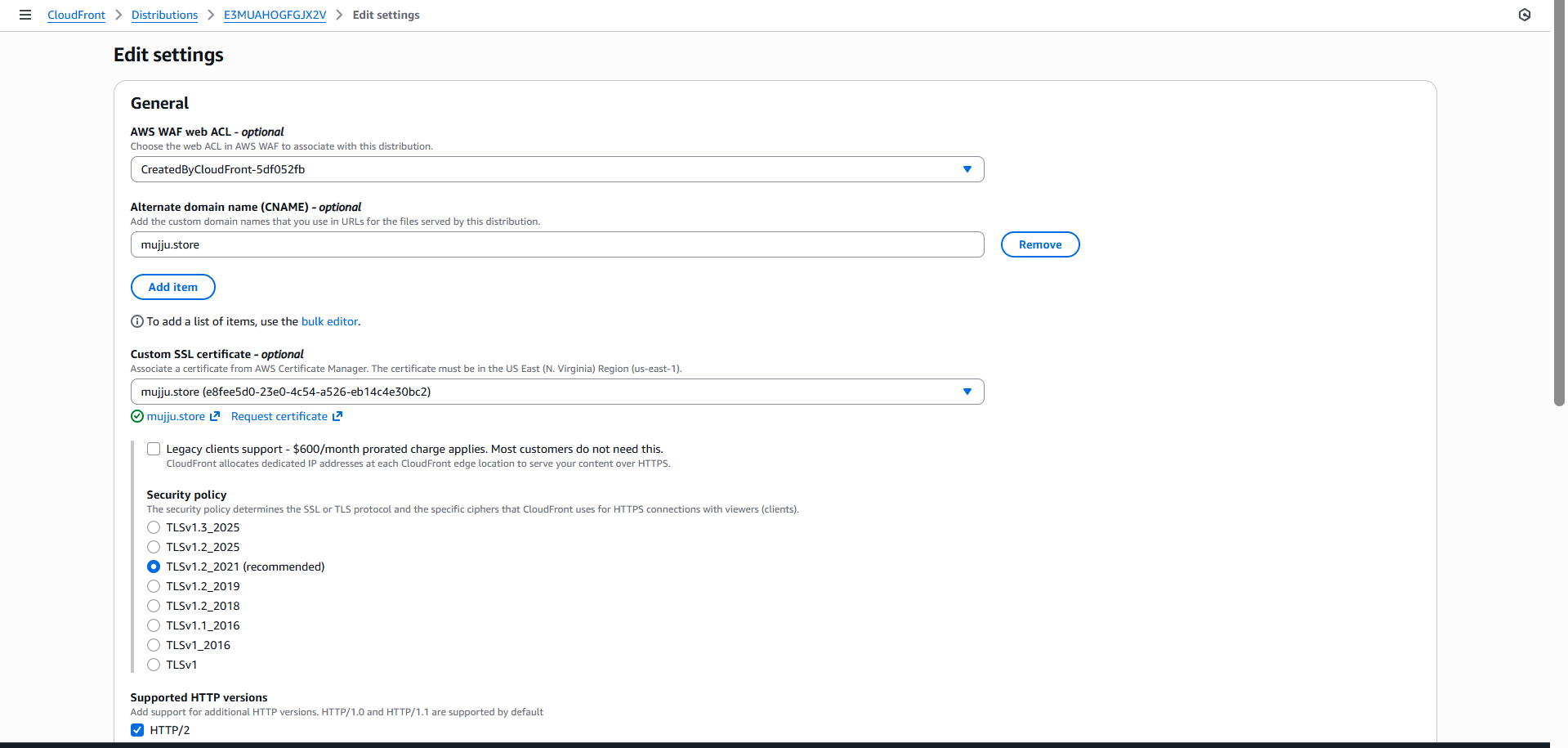
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.