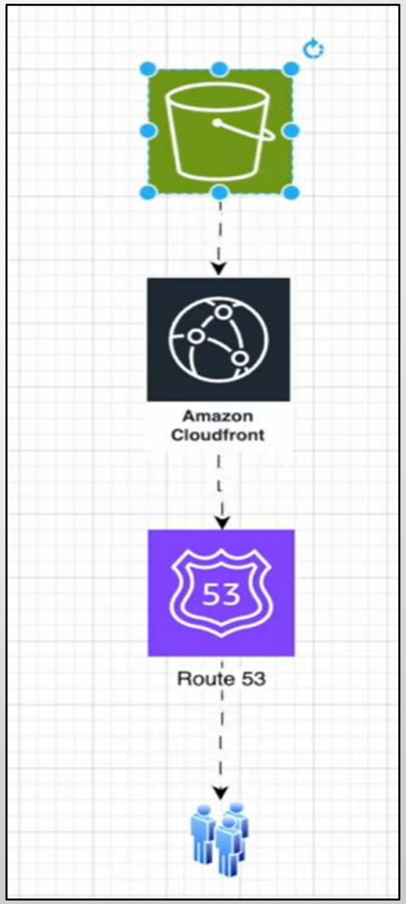
Static Website Hosting Using AWS with Custom Domain Integration

GOAL

The primary goal of this project was to design and deploy a fully functional static website using AWS cloud services.

The aim was to:
- Build a cost-effective, highly available, and secure static website.
- Integrate a custom domain (travelaws.xyz) purchased from GoDaddy.
- Use modern cloud technologies to serve content globally with minimal latency.



Tools and Technologies Used

- Amazon S3 (Simple Storage Service): For storing and hosting the static website content (HTML, CSS, JS).
  - Amazon CloudFront: For content delivery through a global CDN (Content Delivery Network) to reduce latency.
  - Amazon Route 53: For DNS management and domain routing.
  - AWS Certificate Manager (ACM): For provisioning and managing the SSL certificate to enable HTTPS.
  - GoDaddy: Domain registrar from where the custom domain travelaws.xyz was purchased.
  - IAM (Identity and Access Management): To manage access policies and secure the S3 bucket.
  - HTML, CSS, JavaScript: For building the static website content.

Methodology

Website Creation

- Designed a static website using HTML, CSS, and JavaScript.
- Ensured the site was responsive and lightweight.

S3 Bucket Configuration

- Created an S3 bucket with the same name as the domain (travelaws.xyz).
- Uploaded website files.
- Enabled static website hosting on the bucket.
- Set bucket policy to make content publicly readable.

CloudFront Setup

- Created a CloudFront distribution pointing to the S3 bucket.
- Enabled caching and selected the appropriate settings for performance.
- Requested an SSL certificate via ACM for travelaws.xyz and www.travelaws.xyz.

Route 53 Configuration

- Hosted zone was created in Route 53 for travelaws.xyz.
- Added A and CNAME records pointing to CloudFront distribution.
- Connected the GoDaddy domain to Route 53 by updating GoDaddy's nameservers.

PROBLEM STATEMENT

Many startups or individuals looking to publish a simple website face:
- High costs associated with traditional web hosting.
- Complex setup processes for domain integration.
- Lack of scalability and performance in traditional hosting models.

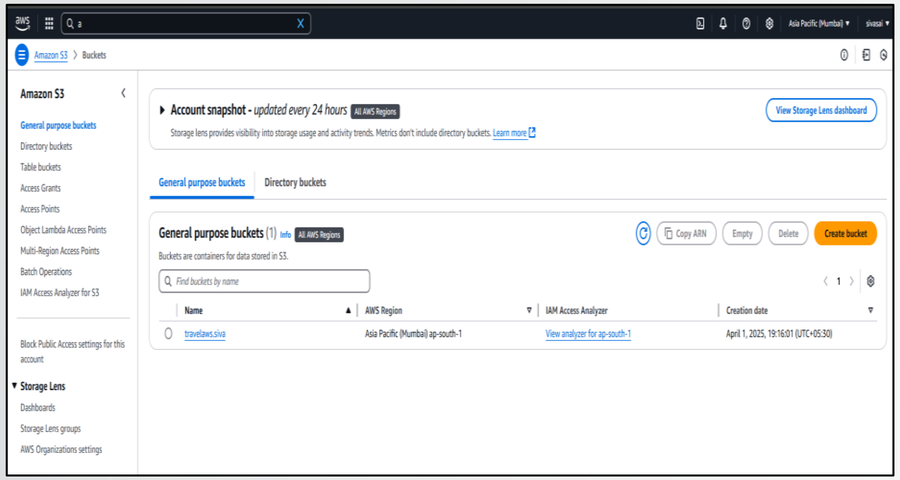
This project solves these issues by leveraging AWS's serverless and scalable architecture to deploy a static website efficiently, securely, and at minimal cost.

SOLUTIONS

By integrating various AWS services and a third-party domain provider, the project successfully:

- Hosted a static website with global availability.
- Ensured secure communication through HTTPS.
- Leveraged CloudFront caching to improve load speed and reduce latency.
- Created a low-cost, highly scalable, and maintenance-free solution suitable for blogs, portfolios, or business landing pages.

Screenshots & Output Pages



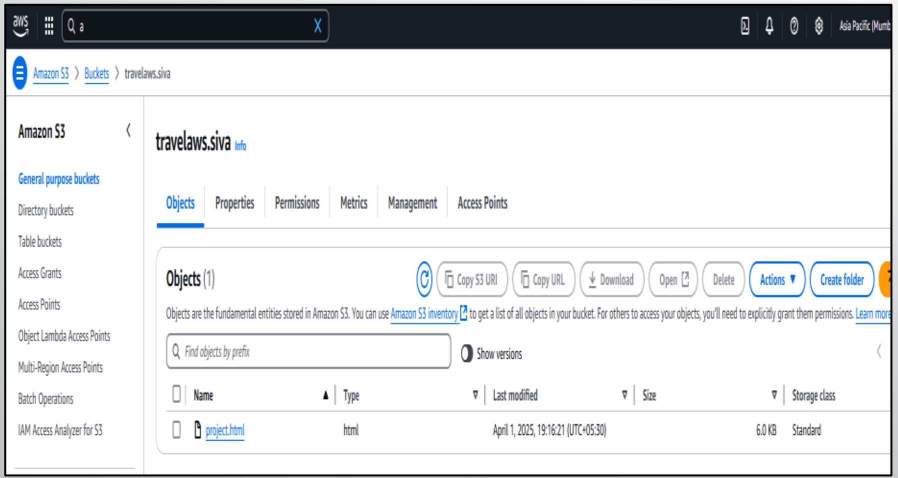
Account snapshot - *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. Learn more

General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
travelaws.siva	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	April 1, 2025, 19:16:01 (UTC+05:30)



travelaws.siva Info

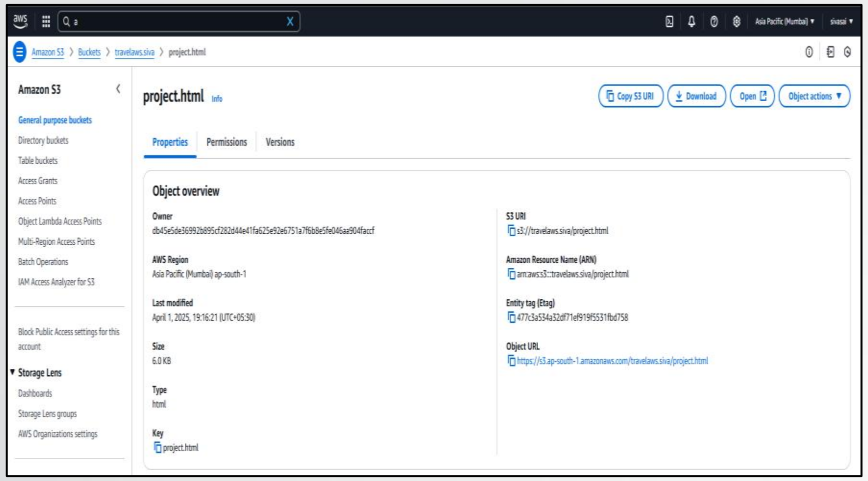
Objects (1)

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

Name	Type	Last modified	Size	Storage class
project.html	html	April 1, 2025, 19:16:21 (UTC+05:30)	6.0 KB	Standard

I began by creating an Amazon S3 bucket named travelaws.siva to serve as the foundation for hosting the static website. In this bucket, I uploaded all the necessary frontend files, including HTML, CSS, and JavaScript. I then enabled the static website hosting feature on the bucket and configured the appropriate permissions and policies to allow public access to the content.

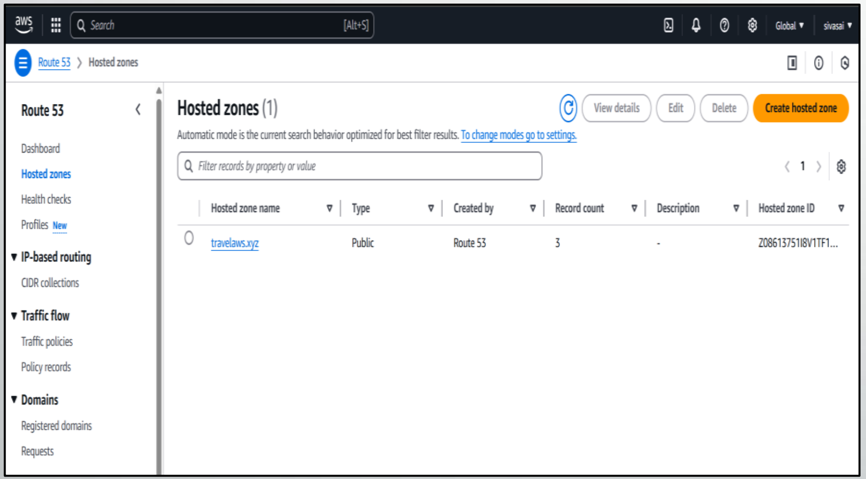
This step allowed the website to be served directly from the S3 bucket over the internet.



project.html Info

Object overview

Owner	db45e5de36992b895cf282d44e41fa625e92e6751a7f6b8e5fe046aa904faccf
AWS Region	Asia Pacific (Mumbai) ap-south-1
Last modified	April 1, 2025, 19:16:21 (UTC+05:30)
Size	6.0 KB
Type	html
Key	project.html
S3 URI	s3://travelaws.siva/project.html
Amazon Resource Name (ARN)	arn:aws:s3:::travelaws.siva/project.html
Entity tag (Etag)	477c3a534a32df71ef919f5531fbd758
Object URL	https://s3.ap-south-1.amazonaws.com/travelaws.siva/project.html



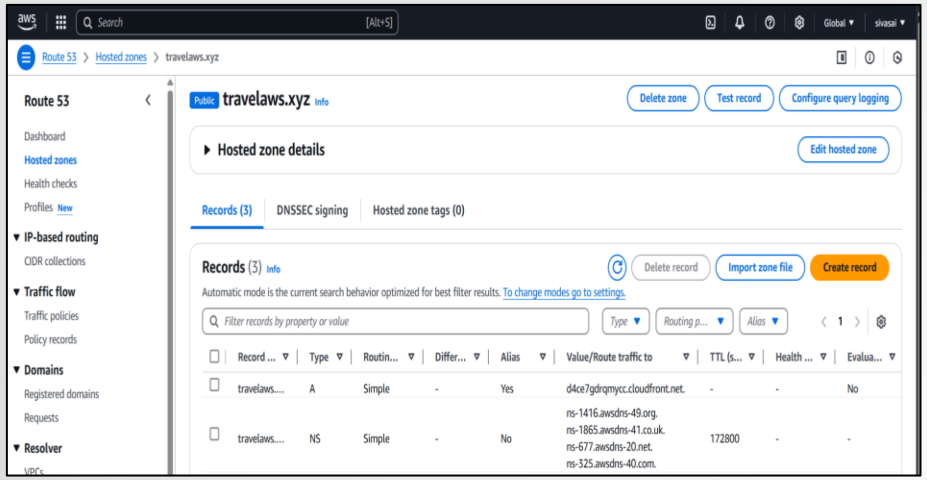
Hosted zones (1)

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Hosted zone name	Type	Created by	Record count	Description	Hosted zone ID
travelaws.xyz	Public	Route 53	3	-	Z08613751I8V1TF1...

Following this, I acquired a custom domain, travelaws.xyz, from GoDaddy, a domain registrar. To route traffic correctly to my website through AWS services, I set up a hosted zone in Amazon Route 53 for the domain. After creating the hosted zone,

I updated the nameserver (NS) records in the GoDaddy domain settings, replacing them with the nameservers provided by Route 53. This configuration ensured that Route 53 would handle all DNS queries for the domain, establishing a connection between the domain and AWS infrastructure.

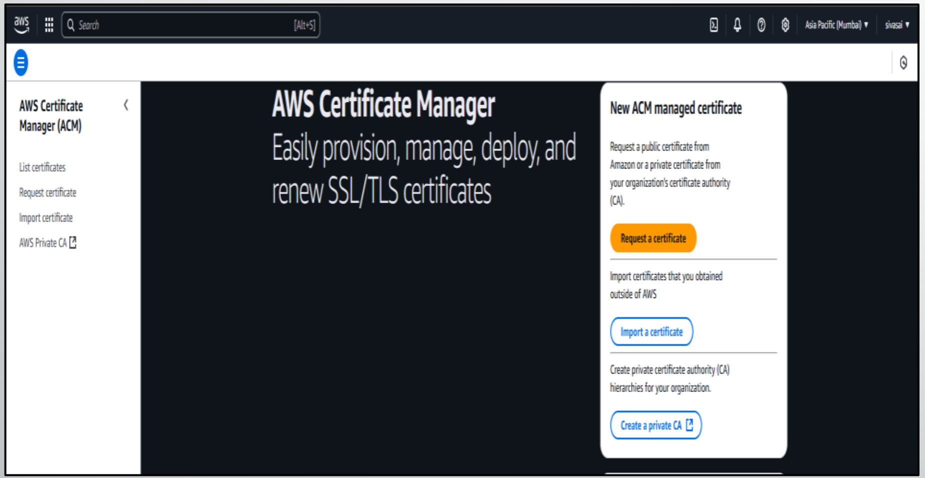


travelaws.xyz

Records (3)

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Record ...	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...	Health ...	Evalua...
travelaws....	A	Simple	-	Yes	d4ce7gdrqmycc.cloudfront.net.	-	-	No
travelaws....	NS	Simple	-	No	ns-1416.awsdns-49.org. ns-1865.awsdns-41.co.uk. ns-677.awsdns-20.net. ns-325.awsdns-40.com.	172800	-	-



AWS Certificate Manager

Easily provision, manage, deploy, and renew SSL/TLS certificates

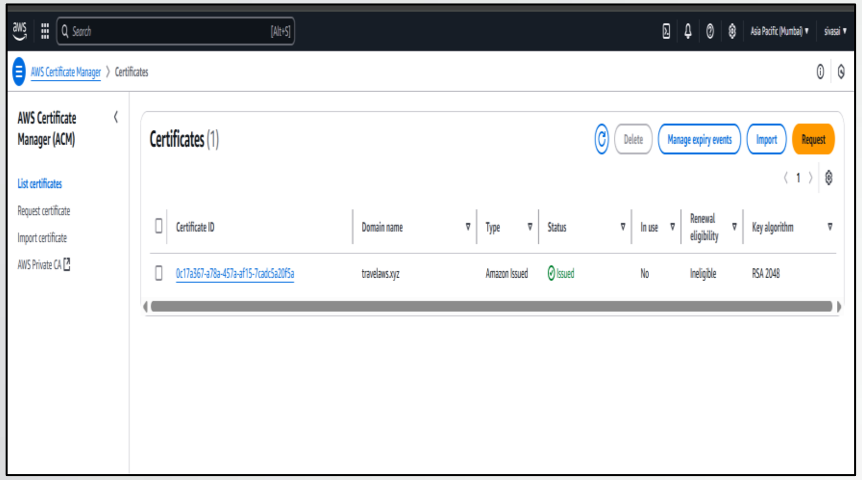
New ACM managed certificate

Request a public certificate from Amazon or a private certificate from your organization's certificate authority (CA).

Import certificates that you obtained outside of AWS

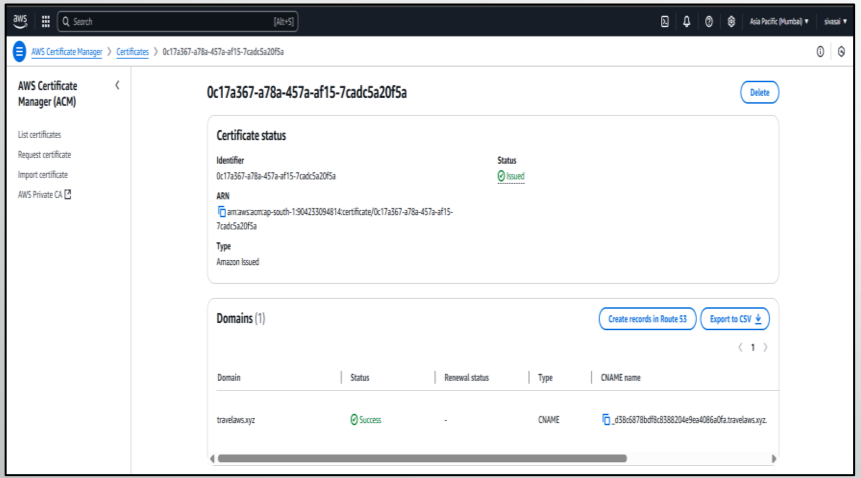
Create private certificate authority (CA) hierarchies for your organization.

After the DNS was configured, I moved on to securing the website by enabling HTTPS. For this, I used AWS Certificate Manager (ACM)
to request an SSL certificate for both travelaws.xyz and www.travelaws.xyz. Once the request was validated and the certificate was successfully issued, I was able to use it to encrypt data transferred between users and the website, ensuring a secure browsing experience.



Certificates (1)

Certificate ID	Domain name	Type	Status	In use	Renewal eligibility	Key algorithm
0c17a367-a78a-457a-af15-7cadc5a20f5a	travelaws.xyz	Amazon Issued	Issued	No	Ineligible	RSA 2048


Search [Alt+S]
Asia Pacific (Mumbai)
sivasai
AWS Certificate Manager > Certificates > 0c17a367-a78a-457a-af15-7cadc5a20f5a
AWS Certificate Manager (ACM)
List certificates
Request certificate
Import certificate
AWS Private CA
Delete
Create records in Route 53
Export to CSV

0c17a367-a78a-457a-af15-7cadc5a20f5a

Certificate status

Identifier: 0c17a367-a78a-457a-af15-7cadc5a20f5a

Status: Issued

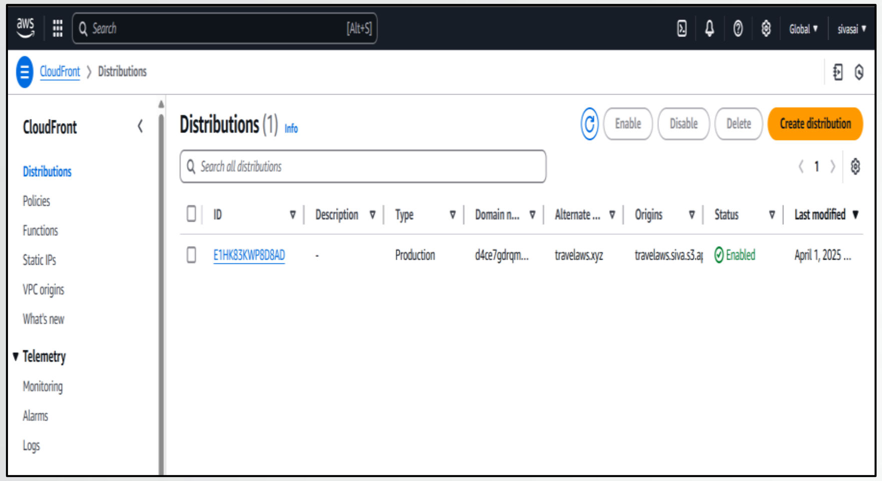
ARN: arn:aws:acm:ap-south-1:904233094814:certificate/0c17a367-a78a-457a-af15-7cadc5a20f5a

Type: Amazon Issued

Domains (1)

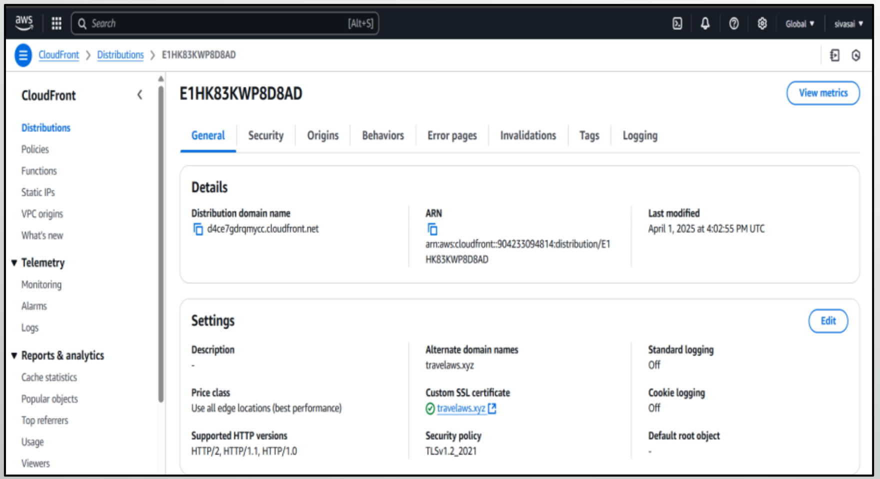
Domain	Status	Renewal status	Type	CNAME name
travelaws.xyz	Success	-	CNAME	_d38c6878bdf8c8388204e9ea4086a0fa.travelaws.xyz.

With the certificate in place, I proceeded to set up Amazon CloudFront, AWS's Content Delivery Network (CDN). I created a new CloudFront distribution and set the S3 bucket (travelaws.siva) as the origin. The CloudFront distribution was configured to serve content over HTTPS using the SSL certificate from ACM. It also allowed the website's static assets to be cached and delivered through AWS's global network of edge locations, significantly reducing latency and improving load times for users around the world.



Distributions (1) Info

ID	Description	Type	Domain n...	Alternate ...	Origins	Status	Last modified
E1HK83KWP8D8AD	-	Production	d4ce7gdrqm...	travelaws.xyz	travelaws.siva.s3.a	✓ Enabled	April 1, 2025 ...



E1HK83KWP8D8AD

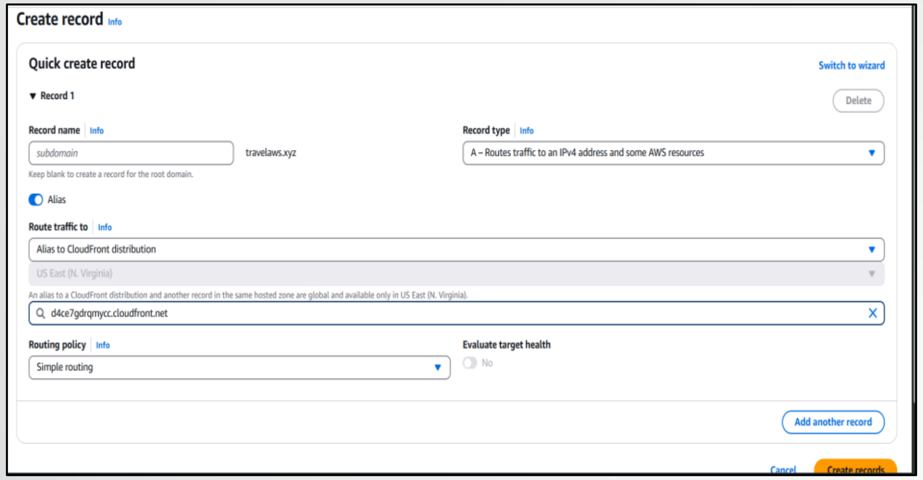
Details

Distribution domain name: d4ce7gdrqmycc.cloudfront.net
ARN: arn:aws:cloudfront::904233094814:distribution/E1HK83KWP8D8AD
Last modified: April 1, 2025 at 4:02:55 PM UTC

Settings

Description: -
Alternate domain names: travelaws.xyz
Standard logging: Off
Price class: Use all edge locations (best performance)
Custom SSL certificate: ✓ travelaws.xyz
Cookie logging: Off
Supported HTTP versions: HTTP/2, HTTP/1.1, HTTP/1.0
Security policy: TLSv1.2_2021
Default root object: -

To finalize the setup, I returned to Route 53 and created a new DNS record that pointed the domain travelaws.xyz to the CloudFront distribution. This record acted as the final link in the chain, ensuring that when users typed https://travelaws.xyz into their browsers, their requests would be routed to CloudFront, which would then serve the content securely and efficiently from the S3 bucket.



Create record Info

Quick create record

Record 1

Record name: subdomain travelaws.xyz

Keep blank to create a record for the root domain.

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

Alias

Route traffic to: Alias to CloudFront distribution

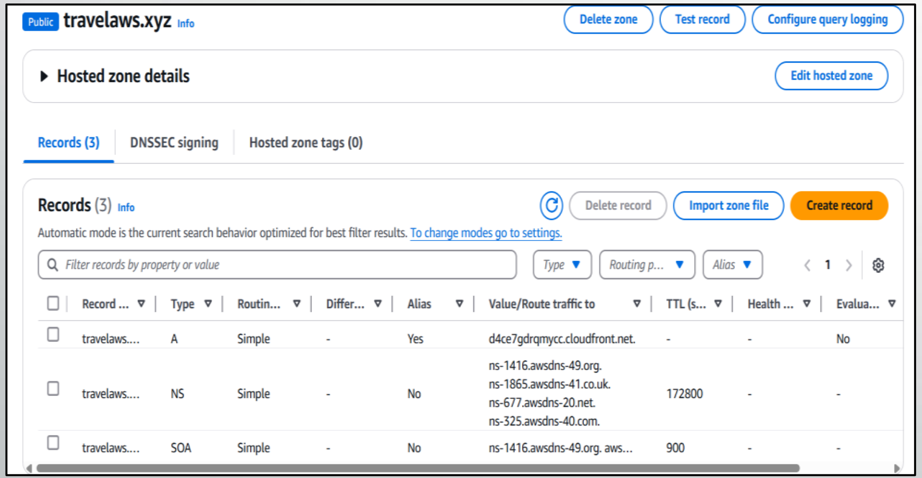
US East (N. Virginia)

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

d4ce7gdrqmycc.cloudfront.net

Routing policy: Simple routing

Evaluate target health: No



Public travelaws.xyz Info

Hosted zone details

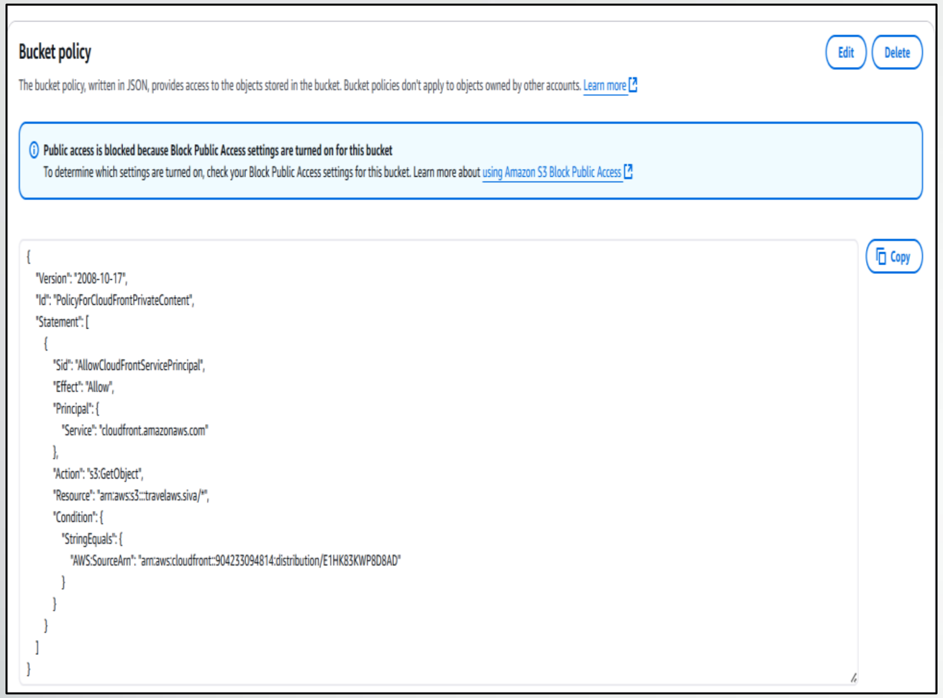
Records (3) | DNSSEC signing | Hosted zone tags (0)

Records (3) Info

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Record ...	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...	Health ...	Evalua...
travelaws....	A	Simple	-	Yes	d4ce7gdrqmycc.cloudfront.net.	-	-	No
travelaws....	NS	Simple	-	No	ns-1416.awsdns-49.org. ns-1865.awsdns-41.co.uk. ns-677.awsdns-20.net. ns-325.awsdns-40.com.	172800	-	-
travelaws....	SOA	Simple	-	No	ns-1416.awsdns-49.org. aws...	900	-	-

To finalize the setup, I returned to Route 53 and created a new DNS record that pointed the domain travelaws.xyz to the CloudFront distribution. This record acted as the final link in the chain, ensuring that when users typed https://travelaws.xyz into their browsers, their requests would be routed to CloudFront, which would then serve the content securely and efficiently from the S3 bucket.



Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

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

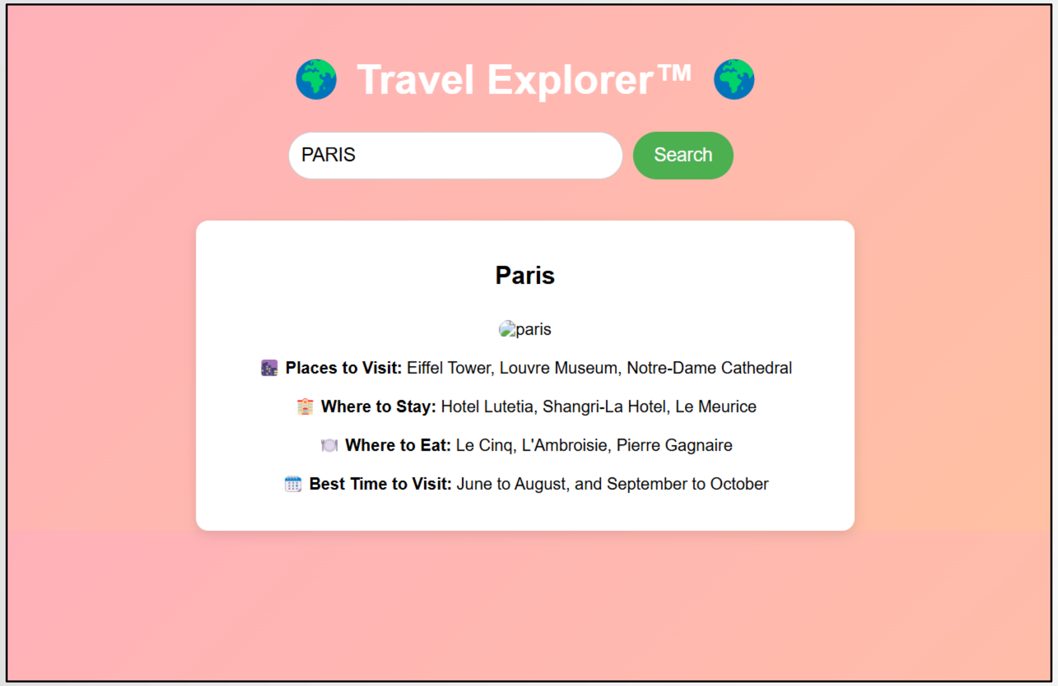
```
{
  "Version": "2008-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::travelaws.siva/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::904233094814:distribution/E1HK83KWP8D8AD"
        }
      }
    }
  ]
}
```

After completing the core setup, I focused on optimizing the **security and access control** of the S3 bucket. Using **S3 Control**, I reviewed and updated the **bucket policy** to follow best practices for public access. This included:

•**Restricting access only to CloudFront** by allowing traffic from the CloudFront origin access identity (OAI) or origin access control (OAC), instead of leaving the bucket open to
 the public.

•**Removing unnecessary public permissions** to reduce the attack surface.

•Ensuring that only the CloudFront distribution could retrieve content from the S3 bucket, thus enforcing secure and restricted content delivery.



Paris

paris

Places to Visit: Eiffel Tower, Louvre Museum, Notre-Dame Cathedral

Where to Stay: Hotel Lutetia, Shangri-La Hotel, Le Meurice

Where to Eat: Le Cinq, L'Ambroisie, Pierre Gagnaire

Best Time to Visit: June to August, and September to October

The figure shown below is the final output of the project – the live static website successfully hosted using various AWS services. It demonstrates the fully functional deployment of the site via Amazon S3, CloudFront, Route 53, and ACM, accessible through the custom domain https://travelaws.xyz."

CONCLUSION

This project demonstrates practical cloud deployment skills using Amazon Web Services (AWS). It showcases the ability to:

- Build secure and scalable websites with minimal infrastructure.
- Integrate third-party services, such as domain registrars (e.g., GoDaddy).
- Understand DNS, content delivery, and security configurations in real-time scenarios.

By completing this project, I have gained hands-on experience in cloud architecture, static website hosting, domain linking, and content delivery optimization using Amazon Web Services (AWS).