AWS: Static Website Host

A study into cloud solutions

Henry Lam

Objective

Demonstrate cloud administration expertise by hosting a static HTML website on Amazon Web Services (AWS).

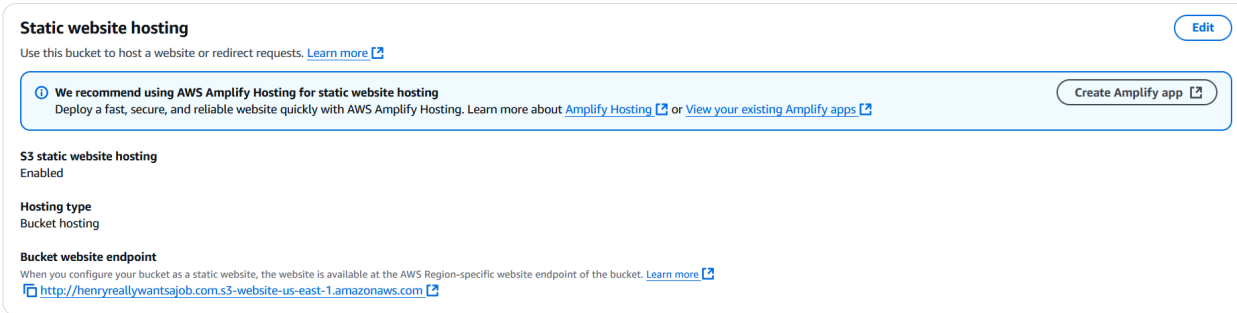
Introduction

Hosting a static website provides an excellent foundation for showcasing cloud administration skills. This report details the process of deploying a static HTML website using AWS. By utilizing AWS tools like S3 Buckets, Route 53, and CloudFront, I successfully configured a secure, scalable, and functional website. This project highlights my ability to design and manage efficient cloud-based solutions.

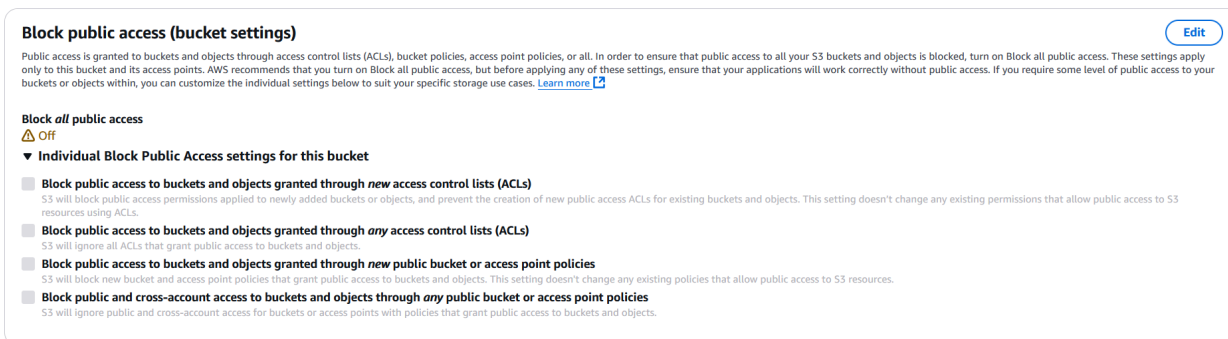
S3 Buckets

To host the front-end components of the static website, I created an S3 bucket named henryreallywantsajob.com. This bucket houses the index.html file, which is available in my GitHub repository. The following configurations ensured the website's public accessibility:

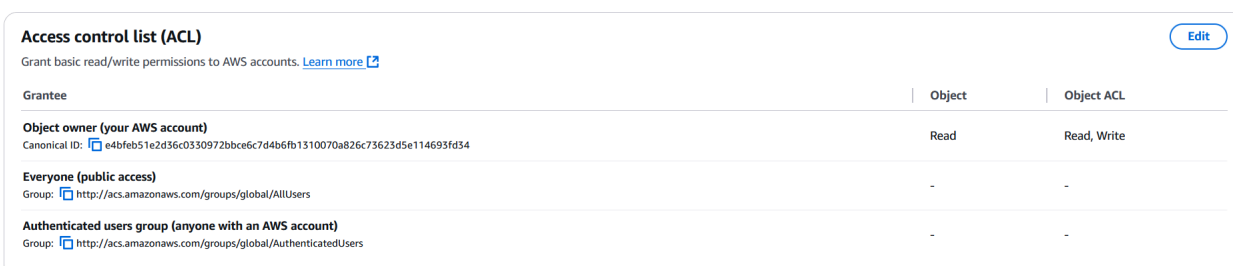
- Enable static website hosting.
    - I activated static website hosting for the bucket, generating the endpoint: http://henryreallywantsajob.com.s3-website-us-east-1.amazonaws.com



- Allow public access.
    - I modified the bucket's public access settings to enable access, ensuring the contents became reachable to users.



- Enable Access Control List (ACL)
    - By adjusting the ACL settings, I made individual objects (e.g., index.html) publicly accessible, resolving any access errors.



- Configuring the Bucket Policy
    - I implemented a bucket policy that granted public access to only the necessary objects, balancing accessibility with security.

Bucket policy

Edit Delete

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

Copy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::henryreallywantsajob.com/*"
    }
  ]
}
```

Route 53

To link a custom domain to the S3 bucket, I utilized AWS Route 53 alongside CloudFront. Key steps included:

- Creating a public hosted zone.
    - I set up a public hosted zone for henryreallywantsajob.com, establishing a container for DNS records to manage domain routing.
- Allow public access
    - I created a CNAME record pointing henryreallywantsajob.com to the S3 bucket's static website endpoint. This made the domain accessible and routed traffic to the bucket efficiently.

Record details

Edit record

Record name

_332563a62921d5c086d18b1f215d432b.henryreally

Record type

CNAME

Value

_ad23e5f0e67aae9d2745e1bad29ca958.xlfgrmvvlj.acm-validations.aws.

Alias

No

TTL (seconds)

300

Routing policy

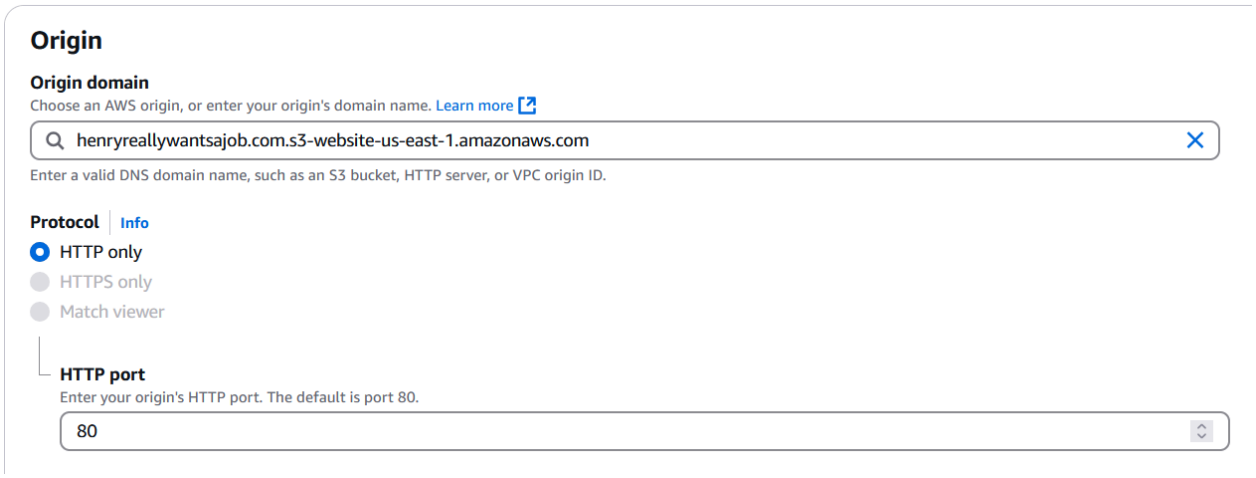
Simple

Cloudfront

To enhance security and performance, I configured AWS CloudFront as a content delivery network (CDN) with HTTPS support:

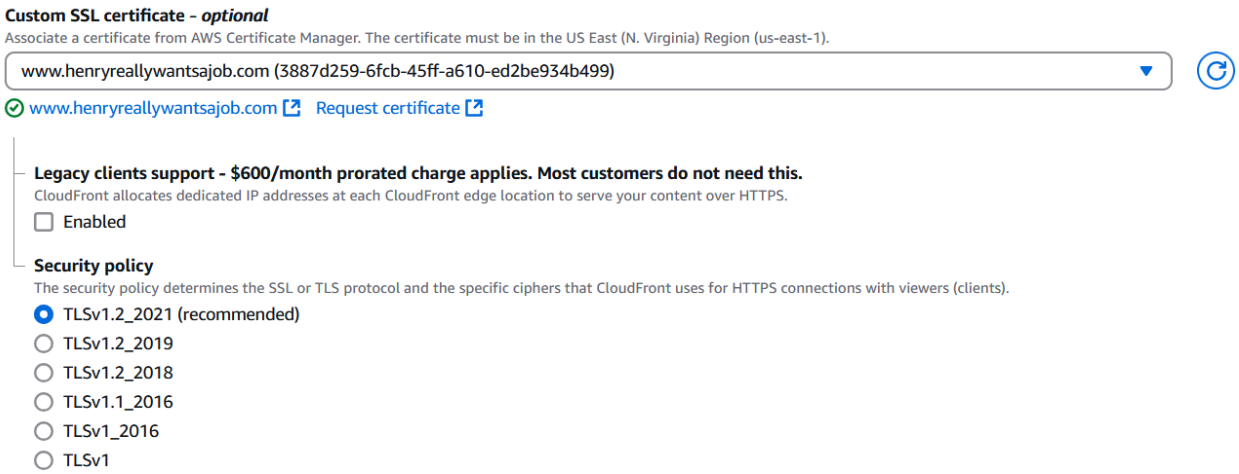
- Setting up a Cloudfront distribution

- I configured a CloudFront distribution with the S3 bucket as the origin, enabling secure and efficient content delivery.



- I enforced HTTPS for all connections to ensure secure communication by redirecting all HTTP connections to HTTPS.

- Requesting an SSL certificate.
  - Using AWS Certificate Manager (ACM), I obtained a public SSL certificate and verified domain ownership through Route 53 DNS validation records.
- Connecting the Certificate to CloudFront:
  - Once validated, I linked the SSL certificate to the CloudFront distribution, enabling secure HTTPS connections. I also added an alternate domain name (CNAME) for custom domain support.



- Updating Route 53 records
    - I updated the Route 53 hosted zone to route traffic to the CloudFront distribution. A new CNAME record was added to direct www.henryreallywantsajob.com requests to the CloudFront endpoint.

Record name

www.henryreallywantsajob.com

Record type

CNAME

Value

d3ap9l1bbe5iy1.cloudfront.net

Alias

No

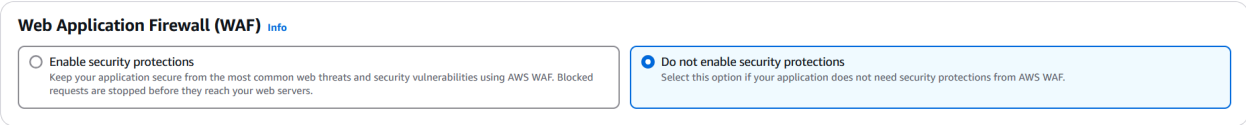
TTL (seconds)

300

Routing policy

Simple

- Firewall Considerations
    - Given the static nature of the website, I chose not to enable a web application firewall (WAF), as the site lacks dynamic content or application logic.



Result

The final outcome is a secure, scalable static website hosted on AWS with the custom domain henryreallywantsajob.com. Accessible via HTTPS, the website guarantees trust and reliability for users while leveraging cost-effective and efficient cloud solutions.