ChatGPT

Deploying a Secure Static Website on Amazon S3 with CloudFront

This guide explains how to host a static personal portfolio (HTML, CSS, JS, images) on Amazon Web Services using an S3 bucket and a CloudFront distribution. The site will be delivered securely over HTTPS from a content delivery network (CDN). We cover creating the S3 bucket, configuring permissions, setting up CloudFront with SSL, customizing error pages, testing the deployment, and optional CI/CD and monitoring enhancements. Follow the steps below to replicate the setup.

Prerequisites

- **AWS Account and Permissions:** You need an AWS account with permission to create S3 buckets, CloudFront distributions, and IAM policies. Administrator privileges or equivalent permissions are required.
- **AWS CLI Installed:** Install and configure the AWS CLI on your local machine. Make sure your AWS credentials are set up (`aws configure`) or that you have a named profile with the needed permissions.
- **Domain Name (Optional):** If you want to use a custom domain (e.g. `www.example.com`), have the domain registered and DNS access (but we assume you're not using Route 53). You can also use the default CloudFront domain name.
- **Local Website Files:** Prepare your website files locally: an `index.html`, any CSS/JS/images, and a `404.html` (or similar) for the error page.

Step 1: Create and Configure the S3 Bucket

1. **Create the S3 bucket:** Choose a unique bucket name (often the same as your domain). For example, to create a bucket named `my-portfolio-bucket` in the `us-east-1` region, run:

```
aws s3api create-bucket --bucket my-portfolio-bucket --region us-east-1
```

If you choose a different region (e.g. `us-west-2`), include the `--create-bucket-configuration` option with the region. For example:

```
aws s3api create-bucket --bucket my-portfolio-bucket --region us-west-2 \
  --create-bucket-configuration LocationConstraint=us-west-2
```

1. **Enable static website hosting:** Configure the bucket as a static website. Specify the main page (`index.html`) and the error page (`404.html`). For example:

```
aws s3 website s3://my-portfolio-bucket/ \
  --index-document index.html --error-document 404.html
```

This step tells S3 which files to serve for the root and error conditions. Note the *website endpoint* that will be shown after this command (e.g. `http://my-portfolio-bucket.s3-website-us-east-1.amazonaws.com`), although you will use CloudFront for HTTPS delivery.

1. **Configure S3 bucket settings:**
2. **Block Public Access:** By default, new buckets block public access. If you plan to use CloudFront with Origin Access Identity (OAI) for security, **keep** public access blocked (we will grant CloudFront explicit access later). If you prefer to allow public read access directly to S3 (less secure), you could disable the public block settings or grant public read in the bucket policy. This guide uses the OAI method for maximum security.
3. **Optional: Encryption and Versioning:** For extra safety, you can enable bucket encryption (SSE-S3 or SSE-KMS) and versioning. This ensures your content is encrypted at rest and older versions are retained. In most cases, static site content is fine without versioning.

Step 2: Upload Your Website Content

With the bucket ready, upload your site files to it. Place all static files (HTML, CSS, JS, images) into a local folder (e.g. `./website-files/`) in the structure you want. Then use the AWS CLI to sync or copy them:

```
aws s3 sync ./website-files/ s3://my-portfolio-bucket/ --acl public-read
```

This command uploads all files in `website-files` to the S3 bucket. The `--acl public-read` flag makes objects readable by anyone. If you are using CloudFront with OAI (recommended), you can omit `--acl public-read` and manage access via the bucket policy (see next step). The file `index.html` should now be at the root of the bucket. You can verify by running:

```
aws s3 ls s3://my-portfolio-bucket/
```

Confirm that your files (e.g. `index.html`, `404.html`, `css/`, `js/`, `images/`) are present.

Step 3: Configure S3 Bucket Access with Origin Access Identity (OAI)

To serve content securely through CloudFront, we prevent direct public access to S3 and allow only CloudFront to fetch content. We do this by creating an Origin Access Identity (OAI) and updating the bucket policy.

1. **Create a CloudFront Origin Access Identity:** This is a special CloudFront user that we grant permission in S3. You can create it via AWS CLI or Console. For example, using the AWS CLI:

```
aws cloudfront create-cloud-front-origin-access-identity \
  --cloud-front-origin-access-identity-config CallerReference="$(date +
%s)",Comment="OAI for my portfolio"
```

The command will return a JSON containing the `Id` and the `S3CanonicalUserId`. Note the `Id` value (looks like `E2ABCDEF123XYZ`). For example:

```
{
  "CloudFrontOriginAccessIdentity": {
    "Id": "E2ABCDEF123XYZ",
    ...
    "S3CanonicalUserId": "abc123examplecanonicaluserid"
  }
}
```

1. **Update the S3 bucket policy:** Use the OAI to allow CloudFront to read objects. In the S3 bucket policy, set the principal to the OAI's canonical user ID or the AWS ARN for that user. Example JSON policy (replace `<OAICanonicalUserID>` with the `S3CanonicalUserId` you obtained, and `<bucket-name>` with your bucket):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudFrontRead",
      "Effect": "Allow",
      "Principal": {
        "CanonicalUser": "<OAICanonicalUserID>"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-portfolio-bucket/*"
    }
  ]
}
```

Save this policy JSON to a file, for example `bucket-policy.json`, then apply it:

```
aws s3api put-bucket-policy --bucket my-portfolio-bucket --policy file://
bucket-policy.json
```

This grants CloudFront (via the OAI) permission to read any file in the bucket. Now **remove any public-read ACLs** from your objects (if you used `--acl public-read` earlier). You can revoke public access on objects or simply rely on CloudFront fetching via the OAI. For example, to make an object private:

```
aws s3api put-object-acl --bucket my-portfolio-bucket --key index.html --acl
private
```

1. **Verify bucket access:** With the policy in place, ensure the bucket is not publicly accessible. You should be able to read files only via CloudFront. To test, try retrieving an object using the S3 public URL; it should be denied or forbidden. (Since you have not set up CloudFront yet, this is just a policy check.)

Step 4: Create and Configure the CloudFront Distribution

Now set up a CloudFront distribution to serve your site over HTTPS using the S3 bucket as the origin.

1. **Create a new distribution:** In the AWS Management Console, go to CloudFront and click **Create Distribution**. For the origin domain, enter your S3 bucket. If using OAI (and no Route 53 alias), use the REST endpoint format (e.g. `my-portfolio-bucket.s3.amazonaws.com`) and **not** the website endpoint. Then, under "Restrict Bucket Access", select "Yes" and choose the OAI you created (or create a new one here by picking "Create a New Identity"). This tells CloudFront to use the OAI credentials to fetch from S3.

2. **Default Root Object:** In the distribution settings, set the *Default Root Object* to `index.html`. This ensures that a request to the site root returns `index.html`.

3. **Origin Protocol Policy:** Choose `HTTPS Only` or `Match Viewer`. (HTTPS Only ensures CloudFront always connects to S3 over HTTPS.)

4. **Viewer Protocol Policy:** Under "Behavior", set the policy to **Redirect HTTP to HTTPS**. This forces visitors to use HTTPS.

5. **Enable caching and performance settings (optional):** You can leave the default cache behaviors, or adjust TTL and caching based on your static content needs. For example, set longer cache TTLs for images/CSS that change infrequently.

6. **Alternate Domain Names (CNAMEs) and SSL Certificate (if using custom domain):**

7. If you have a custom domain (e.g. `www.example.com`), add it under *Alternate Domain Names*.
8. Request or attach an SSL/TLS certificate: Go to AWS Certificate Manager (ACM) in the us-east-1 region (N. Virginia) and request a public certificate for your domain name. Validate it via email or DNS (DNS if you can manage records with your DNS provider). Once issued, return to CloudFront and in the *SSL Certificate* section choose the ACM certificate.
9. Finally, configure your DNS provider (outside Route 53) to point your domain to the CloudFront distribution domain (it looks like `d1234abcdef8.cloudfront.net`). Typically, this is a CNAME record `www.example.com -> d1234abcdef8.cloudfront.net`.

If you choose **not** to use a custom domain, CloudFront provides a default `*.cloudfront.net` domain with built-in HTTPS support, so no extra certificate is needed.

1. **Finalize and create:** Review all settings and create the distribution. CloudFront will say "In Progress" as it deploys. This can take 5–15 minutes. While it's deploying, you can prepare the error pages or other settings.

Step 5: Configure Custom Error Pages

To provide friendly error responses (such as a 404 page), configure CloudFront to serve your `404.html` when a file is not found.

1. **S3 Error Document:** Ensure your bucket's static website configuration has `404.html` as the error document (you set this in Step 1). This means if CloudFront fetches the website endpoint and gets a 404, S3 will return `404.html`.

2. **CloudFront Custom Error Responses:** In the CloudFront distribution settings (under the *Error Pages* tab), add custom error responses:

3. Add a response for **HTTP 404**. Set the *Response Page Path* to `/404.html` and *HTTP Response Code* to `404`. You may also set a minimal TTL (e.g. 60 seconds) for the error so CloudFront will periodically retry rather than caching the 404 forever.
4. Add a response for **HTTP 403** (Forbidden). Sometimes S3 returns 403 for missing objects when the origin is not a website endpoint. Configure the 403 to use `/404.html` as well, with response code `404`.
5. (Optional) Configure other codes (e.g. 500) if you have custom pages.

These settings ensure that if someone requests a missing page, CloudFront will return your custom `404.html` instead of a generic AWS error.

Step 6: Testing the Deployment

After the CloudFront distribution status shows as **Deployed**, test your website:

- **Access the site:** Open a browser and go to your domain. If you set a custom domain, use that (e.g. `https://www.example.com`). Otherwise use the CloudFront URL (e.g. `https://d1234abcdef8.cloudfront.net`). You should see your portfolio's homepage (index page) load over HTTPS (check for the lock icon).

- **Test HTTPS:** Verify that the connection is secure. If you used a custom domain, ensure the certificate is valid for that domain. Browsers should show a secure padlock.

- **Test content and assets:** Navigate through your site and ensure all CSS, JS, and images load correctly. They should all be coming from the CloudFront URL or your domain.

- **Test the error page:** Try accessing a non-existent page (e.g. `https://your-site.com/no-such-page.html`). You should see your `404.html` content and a browser status code of 404. This confirms the custom error configuration is working.

- **CloudFront invalidation (if needed):** Whenever you update content in S3 and need to push changes immediately, either wait for the cache TTL to expire or create an invalidation. For example, to invalidate all files:

```
aws cloudfront create-invalidation --distribution-id E2ABCDEF123XYZ --paths
"/*"
```

Replace the distribution ID with your CloudFront's ID (from the console or CLI). This forces CloudFront to fetch fresh files from S3.

- **Verify no direct S3 access:** As a final check, try using the S3 bucket website endpoint (http) or REST endpoint (http) to fetch your site. With OAI in place, direct S3 access should be denied or blocked. Only the CloudFront URL should serve your content.

Optional Enhancements

Continuous Deployment (CI/CD)

Automate updates by connecting your source control to S3 and CloudFront:

- **Git-based Deployment:** Use a CI tool (GitHub Actions, GitLab CI, etc.) with AWS credentials to run `aws s3 sync` on each push. For example, a GitHub Actions workflow might check out your code, configure AWS credentials, and run:

```
aws s3 sync ./website-files/ s3://my-portfolio-bucket/
aws cloudfront create-invalidation --distribution-id E2ABCDEF123XYZ --paths
"/*"
```

This automatically uploads new content and clears the CloudFront cache on each commit.

- **AWS CodePipeline/CodeBuild:** Set up a pipeline that triggers on changes in your code repository. CodeBuild can run build and deploy commands (similar to above) and push to S3, then invoke a CloudFront invalidation.

Monitoring and Logging

Keep an eye on site performance and security:

- **CloudFront Access Logs:** Enable CloudFront standard logs to an S3 bucket. Each request (edge location hits) is logged. This helps analyze traffic patterns or diagnose issues. You can enable logs in the *General* settings of the distribution and specify a log bucket.

- **S3 Server Access Logs (optional):** You can also enable S3 access logs on the bucket to record requests from CloudFront. This is less common when using CloudFront, but available if needed.

- **CloudWatch Metrics:** CloudFront publishes metrics (Requests, 4xxErrorRate, 5xxErrorRate, etc.) to CloudWatch. In the CloudWatch console, you can create dashboards or set alarms on these metrics (for example, alert if 5xx errors spike).

- **Alarms/Alerts:** Set up CloudWatch Alarms on error rates or usage to notify you of unusual patterns. For example, an alarm if `4xxErrorRate > 1%` for 10 minutes might indicate broken links or a DDoS attempt.

- **Certificate Expiration:** If using a custom domain, ensure the ACM certificate is renewed before expiration. AWS managed certificates renew automatically, but double-check expiration dates or set a reminder.

- **Security Best Practices:** Keep “Block Public Access” enabled on the S3 bucket (with OAI). Do not expose other S3 permissions. Rotate IAM keys if you use them for deployment. Regularly review the bucket policy and CloudFront settings.

Conclusion

You have now set up a static portfolio website on AWS S3 with CloudFront delivering content securely over HTTPS. The architecture ensures that all content is cached at edge locations, providing fast load times for visitors globally, while maintaining control and security over the content via IAM policies. With the optional CI/CD and monitoring in place, updates to your portfolio can be deployed automatically, and the site's performance and security can be continuously observed. Following these steps, anyone can replicate the setup for their own static site without relying on AWS Route 53.
