Deploying to S3/CloudFront

This guide walks through how to host and publish your next Gatsby site to AWS using S3. Optionally - but very recommended - you can add CloudFront, a global CDN to make your site *even faster*.

Getting Started: AWS CLI

Create an IAM account with administration permissions and create an access id and secret for it. You'll need these in the next step.

Install the AWS CLI and configure it (ensure Python is installed before running these commands):

```
pip install awscli
aws configure
```

The AWS CLI will now prompt you for the key and secret, so add them.

Setting up: S3

Now that your Gatsby site is up and running and AWS access is sorted out, you'll need to add hosting and make the site live on AWS.

First, install the Gatsby S3 plugin:

```
npm install gatsby-plugin-s3
```

Add it to your `gatsby-config.js`: (don't forget to change the bucket name)

```
plugins: [
  {
    resolve: `gatsby-plugin-s3`,
    options: {
      bucketName: "my-website-bucket",
    },
  },
]
```

And finally, add the deployment script to your `package.json`:

```
"scripts": {
   ...
   "deploy": "gatsby-plugin-s3 deploy"
}
```

That's it! Run `npm run build && npm run deploy` to do a build and have it immediately deployed to S3!

Deploying with `.env` variables

Some deployments require passing environment variables. To deploy with environment variables, update the deployment script to your `package.json`:

```
"scripts" : {
    ...
    "deploy": "npm run -n \"-r dotenv/config\" && npm run build && gatsby-plugin-s3 deploy"
}
```

This command requires `dotenv` first, runs build next, and finally deploys to S3. `dotenv`, a dependency of Gatsby that doesn't need to be installed directly, loads environment variables and makes them available globally.

If you have multiple AWS profiles in your machine, you can deploy by declaring your `AWS_PROFILE` before the deploy script:

```
AWS_PROFILE=yourprofilename npm run deploy
```

Setting up: CloudFront

CloudFront is a global CDN and can be used to make your blazing fast Gatsby site load *even faster*, particularly for first-time visitors. Additionally, CloudFront provides the easiest way to give your S3 bucket a custom domain name and HTTPS support.

There are a couple of things that you need to consider when using gatsby-plugin-s3 to deploy a site which uses CloudFront.

There are two ways that you can connect CloudFront to an S3 origin. The most obvious way, which the AWS Console will suggest, is to type the bucket name in the Origin Domain Name field. This sets up an S3 origin, and allows you to configure CloudFront to use IAM to access your bucket. Unfortunately, it also makes it impossible to perform serverside (301/302) redirects, and it also means that directory indexes (having index.html be served when someone tries to access a directory) will only work in the root directory. You might not initially notice these issues, because Gatsby's clientside JavaScript compensates for the latter and plugins such as `gatsby-plugin-meta-redirect` can compensate for the former. But just because you can't see these issues, doesn't mean they won't affect search engines.

In order for all the features of your site to work correctly, you must instead use your S3 bucket's Static Website Hosting Endpoint as the CloudFront origin. This does (sadly) mean that your bucket will have to be configured for public-read, because when CloudFront is using an S3 Static Website Hosting Endpoint address as the Origin, it's incapable of authenticating via IAM.

gatsby-plugin-s3 configuration

In the gatsby-plugin-s3 configuration file, there are a couple of optional parameters that you can usually leave blank, `protocol` and `hostname`. But when you're using CloudFront, these parameters are vital for ensuring redirects work correctly. If you omit these parameters, redirects will be performed relative to your S3 Static Website Hosting Endpoint. This means if a user visits your site via the CloudFront URL and hits a redirect, they will be redirected to your S3 Static Website Hosting Endpoint instead. This will disable HTTPS and (more importantly) will display an ugly and unprofessional URL in the user's address bar.

By specifying the `protocol` and `hostname` parameters, you can cause redirects to be applied relative to a domain of your choosing.

```
{
    resolve: `gatsby-plugin-s3`,
    options: {
        bucketName: "my-example-bucket",
        protocol: "https",
        hostname: "www.example.com",
    },
}
```

If you use your site's URL elsewhere in gatsby-config.js, you can define the URL once at the top of the config:

```
const siteAddress = new URL("https://www.example.com")
```

And then in the Gatsby config you can reference it like so:

```
{
    resolve: `gatsby-plugin-s3`,
    options: {
        bucketName: "my-example-bucket",
        protocol: siteAddress.protocol.slice(0, -1),
        hostname: siteAddress.hostname,
    },
}
```

If you need the full address elsewhere in your config, you can access it via `siteAddress.href`.

For automatic setup of builds that are deployed straight to S3:

References:

- Gatsby on AWS, the right way
- Using CloudFront with gatsby-plugin-s3
- Publishing Your Next Gatsby Site to AWS With AWS Amplify
- Escalade Sports: From $5000 to $5/month in Hosting With Gatsby
- Deploy your Gatsby.js Site to AWS S3