# CS 470 Module Three Assignment Guide

## Part One: Preparing the Angular Code

In Module Two, you forked and cloned the **learn-angular-from-scratch-step-by-step** code into a **lafs-web** directory on your computer. Now, you need to do some work to make that codebase ready for use on AWS. To start:

1. Open a command shell, such as PowerShell.
2. Change the directory to **lafs-web**.
3. Build the Angular application for deployment with the following command:

ng build --prod --base-href .

You may encounter some of the following issues:

1. If you receive the following error, you need to install the SASS preprocessor. See steps a and b below. Otherwise, go to step 4. For more information on this error, see [Angular CLI: 7.0.3 Cannot Find Module 'node-sass'](https://github.com/angular/angular-cli/issues/12746).

ERROR in ./src/styles.scss

Module build failed (from ./node\_modules/mini-css-extract-plugin/dist/loader.js):

ModuleBuildError: Module build failed (from ./node\_modules/sass-loader/lib/loader.js):

Error: Cannot find module 'node-sass'

* 1. npm install --save-dev node-sass
  2. ng build --prod --base-href .

1. There is a high chance that it will complain about the node version installed on your machine. To update to the latest version of node locally, you can complete the following steps:
   1. Go to the [Download Node.js](https://nodejs.org/en) webpage and get the latest installer version for your PC. Then follow the steps to install the latest node version or use **npm install npm@latest -g** in the shell prompt.
   2. Run **node -v** in the shell prompt and confirm it has a version higher than 20 installed. If it does, you can repeat step 3.
   3. If you still see the older version of node, you must execute the **nvm <version>** command in the shell prompt using the version you downloaded in place of **<version>**. Check the version in use again and then repeat step 3.
2. If there is a series of errors related to **digital envelope routines::unsupported**, you will need to run the following command in the shell prompt:

set NODE\_OPTIONS=--openssl-legacy-provider

Read more about this issue since Node v17 on the [GitHub Issues Discussion Board](https://github.com/coreui/coreui-free-react-admin-template/issues/336#issuecomment-983526097). You can now repeat step 3.

1. You are done preparing the code. What you did was tell Angular to use the current path (.) instead of the default root path (/) for routing subpages on your website. A good reference on the **base-href** and **deploy-url** build options for Angular can be found at [Angular 4: Use of base-href and deploy-url Build Options](https://shekhargulati.com/2017/07/06/angular-4-use-of-base-href-and-deploy-url-build-options/).

## Part Two: Creating an S3 Bucket

In Part One, you prepared your code to be deployed to the cloud. Now, you must create the place where your code will be deployed. In this section, you will create an S3 bucket to be your deployment target.

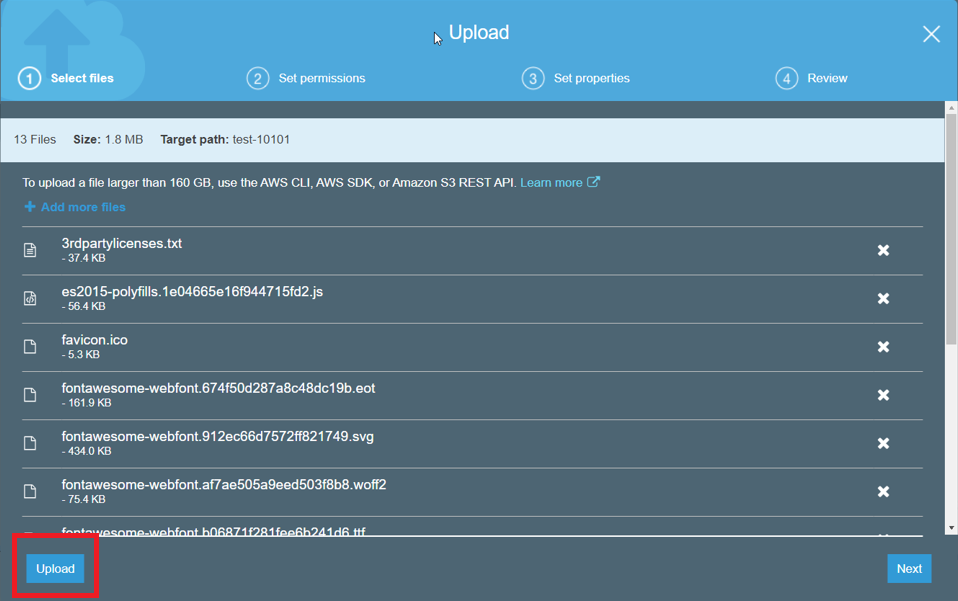
1. Log into **AWS Academy**.
2. Access the **AWS Console** by clicking the **modules** button on the left, then **Learner Lab – Foundational Services**, and then **Start Lab** to start the lab. You can tell the lab has started from the Cloud Access message.
3. Once the Lab has started and the light by the AWS at the top of the screen has turned green, click the AWS link to start the management console.
4. Type “S3” in the **Find Services** search bar and select **S3**, or select the **All Services** link. Then click **S3** under the **Storage** heading.
5. Click the orange **Create Bucket** button.
6. You must create a unique S3 bucket name. See the [Bucket Limits and Restrictions](https://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html#bucketnamingrules) webpage. Some key points to consider:
   1. Bucket names are unique by AWS Partition (World, China, and the Government). For our purposes, we operate in the World partition. Your name must be globally unique.
   2. Do not worry too much about the name. It has no real impact on your applications other than as a “locator” of your bucket. Many companies choose to generate names using a formula.
   3. If you cannot get your first choice, add some numbers or make the name slightly more unique.
7. Keep the region as **US East (N. Virginia) us-east-1.**
8. Keep the **Block All Public Access** check box checked.
9. Do not change anything in advanced settings.
10. Click **Create Bucket** at the bottom of the page.
11. The console should return you to the list of S3 buckets. You will see your new bucket listed. You will also see its region, that it is not public, and a timestamp showing when it was created.

Congratulations! You have now created your first serverless cloud “thing.”

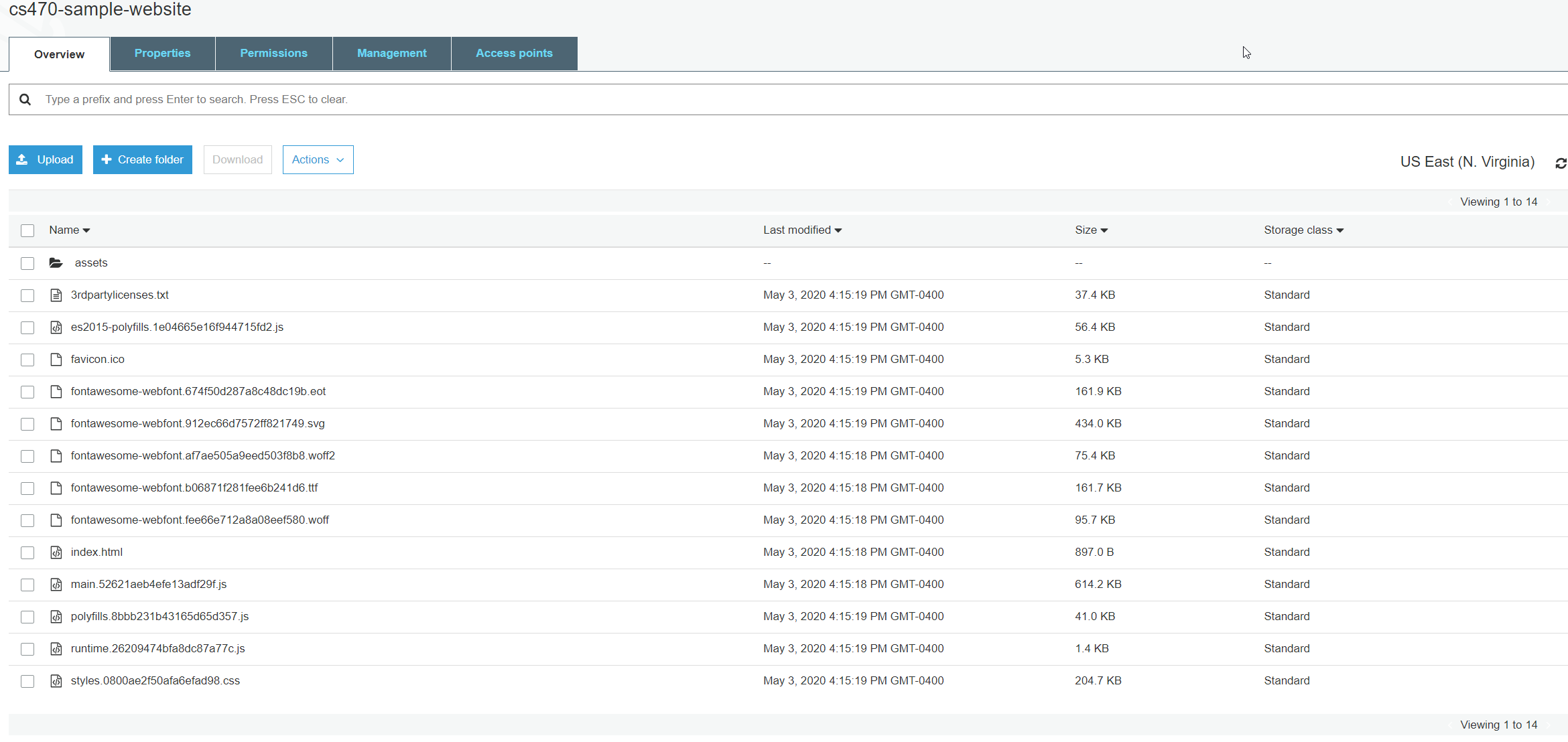
## Part Three: Deploying Your Website

In Part Two, you created your S3 bucket to host your website. Now, you will deploy your application.

1. Click the link for the bucket name.
2. Click the blue **Upload** button.
3. Drag and drop all the files from the **lafs-web/dist/learn-angular-from-scratch** directory. Make certain you select the contents of the directory, not the directory itself.
   1. If you cannot or do not want to drag and drop your files, follow these steps to upload manually:
      1. Select the **Create Folder** blue box.
      2. Type in “assets” (without quotes). Leave all other settings the same.
      3. Upload all the files in the **lafs-web/dist/learn-angular-from-scratch** directory to the S3 bucket root.
      4. Upload all the files in the **lafs-web/dist/learn-angular-from-scratch/assets** directory to the S3 bucket **assets** folder.
4. Click the blue **Upload** button in the lower-left corner. Leave all the settings as they are until the last page, and then click the blue **Upload** button. This time, the button will be in the lower-right corner.



1. You should have 14 objects in your S3 folder.

****

That’s it! You have deployed your application.

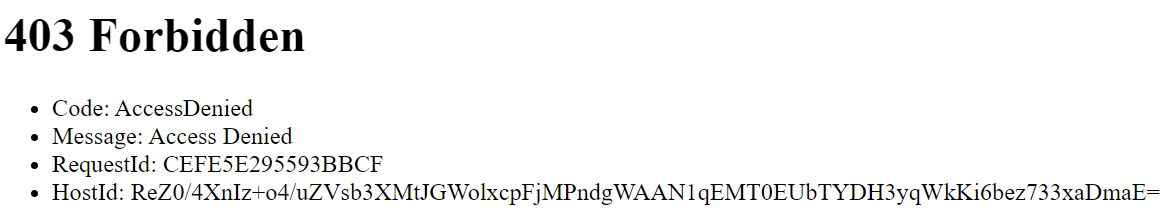
## Part Four: Configuring Your S3 Bucket as a Website

In Part Three, you deployed your application to your S3 bucket. But your application is not a website yet. In this section, you will configure your S3 bucket to operate as a website.

1. Click the orange **Close** button.
2. Select the **Properties** tab.
3. Select the **Edit** button on the **Static Website Hosting** card.
4. Select the **Enable** radio button on the **Static Website Hosting** section.
5. Select the radio button for **Host a Static Website**.
   1. Visit the [Tutorial: Configuring a Static Website on Amazon S3](https://docs.aws.amazon.com/AmazonS3/latest/userguide/HostingWebsiteOnS3Setup.html) webpage for more details on what you are about to do.
6. You will also need to address the following items:
   1. Index document. The index document is your website’s home page. Angular created this file, and it is in your folder. You need to type in “index.html”, or the **Save** button will not be enabled.
   2. Error document. You do not have a custom error page, but if you did, this is where you would put in its name.
   3. Redirection rules. Leave this blank.
7. Click **Save Changes**.
8. Copy and save the endpoint shown. It will look like this: http://{your bucket name}.s3-website-us-east-1.amazonaws.com.

Congratulations! You have configured a serverless web server. Test it by putting in the endpoint you saved into a browser.

**Note:** If you forgot to save the endpoint, don’t worry. Almost everyone does that. Just click on the **Static Website Hosting** card, and it will be there again.



Oh no, a 403! Don’t worry. You did not make a mistake. In fact, you did it right. Your website is secure by default, so no public access is allowed. You will make it public in the next set of steps.

## Part Five: Configuring Security for Your S3 Bucket Website

In Part Four, you configured your bucket to work as a website. But you are not able to access the website from a browser. It’s not a very good website. You will now configure the security for your bucket to make it public and readable.

This configuration is done in two separate steps: making your bucket public and allowing public access to the read operations.

### Making Your Bucket Public

1. Select the **Permissions** tab for your bucket.
2. This tab should have a **Block Public Access** section with a blue **Edit** button and a **Block All Public Access** box.
3. Select the **Edit** button.
4. Uncheck the **Block All Public Access** check box.
5. Click **Save Changes**.
6. Type “confirm” in the dialog box and click the orange **Confirm** button. This step is confirming to AWS that you want your S3 bucket to be public.
7. Test your endpoint again.

A 403! Again, don’t worry! You still did not make a mistake. In fact, you are still doing it right. S3 has two layers of security: the bucket and the operations on the bucket. You have just opened the security on the bucket, but you did not authorize any operations.

### Enabling the Read Access Operations

1. Create a bucket policy to enable read access.
2. Select the **Bucket Policy** button under **Permissions** for your bucket.
3. You must modify the following JSON segment and paste it into the box on the screen. You must replace “example.com” with your bucket name.

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": [

"s3:GetObject"

],

"Resource": [

"arn:aws:s3:::example.com/\*"

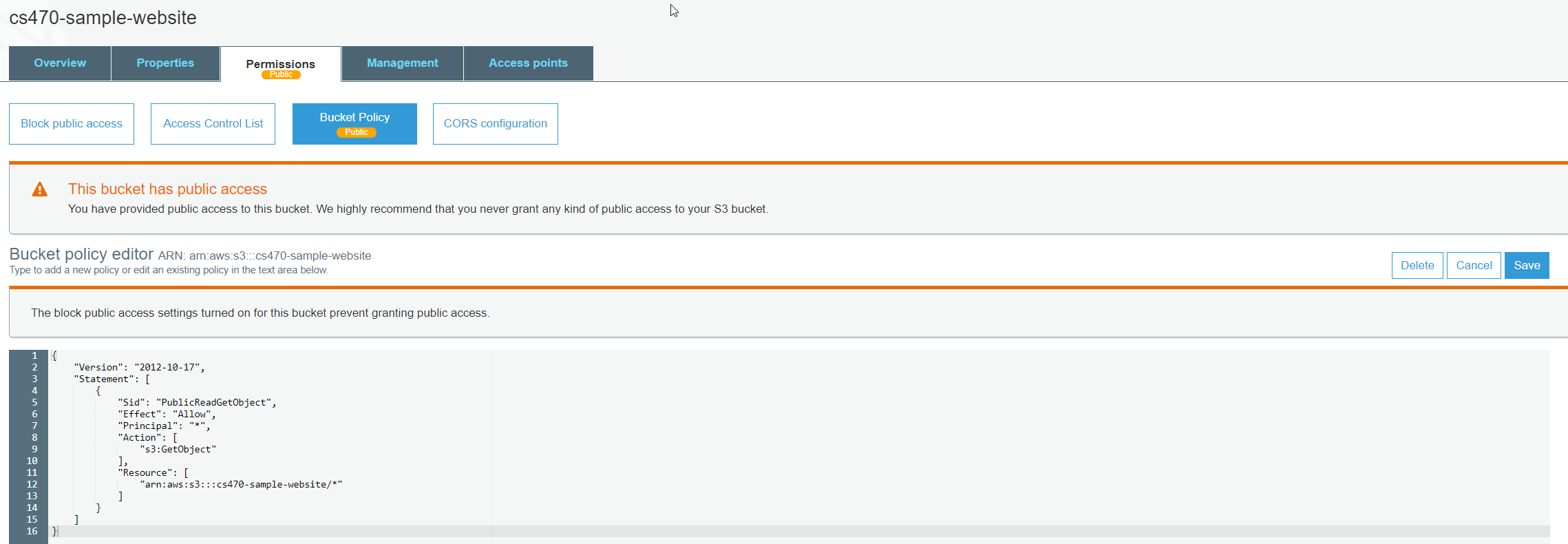
]

}

]

}

1. After you’ve pasted the JSON into the box, click **Save**.
2. If done correctly, the screen will change, and the **Permissions** tab and the **Bucket Policy** will both show the word “Public”.
3. Now, test the website in the browser again.

****

Congratulations! You migrated a totally serverless website from desktop to containers and now to the cloud.