# Build document: GEO.CA Search and Discovery Application

## Table of Contents

[Document version history](#_heading=h.3ijfejlqb2zn) [2](#_heading=h.3ijfejlqb2zn)

[Summary](#_heading=h.fetw6dprdlrw) [3](#_heading=h.fetw6dprdlrw)

[Architecture](#_heading=h.gufidh15rlkg) [3](#_heading=h.gufidh15rlkg)

[Technical Instructions](#_heading=h.eloegwc0fa94) [4](#_heading=h.eloegwc0fa94)

[Creating s3 buckets for the application and logging](#_heading=h.1fob9te) [4](#_heading=h.1fob9te)

[Creating a CodePipeline for automatic deployment](#_heading=h.3znysh7) [7](#_heading=h.3znysh7)

[Creating a Route 53 Record](#_heading=h.2et92p0) [17](#_heading=h.2et92p0)

[Creating a Certificate](#_heading=h.tyjcwt) [18](#_heading=h.tyjcwt)

[Creating a Web Application Firewall](#_heading=h.3dy6vkm) [20](#_heading=h.3dy6vkm)

[Creating a Cloudfront distribution](#_heading=h.1t3h5sf) [23](#_heading=h.1t3h5sf)

[Creating a Route 53 Alias record](#_heading=h.4d34og8) [26](#_heading=h.4d34og8)

[Securing the s3 bucket to only accept access from CloudFront](#_heading=h.17dp8vu) 27

## Document version history

| Name | Reversion Type | Date | Comments |
| --- | --- | --- | --- |
| Chris Melnick-MacDonald | Created | 2021-04-20 | Initial document creation |

## 

## 

## 

## 

## 

## 

## 

## 

## 

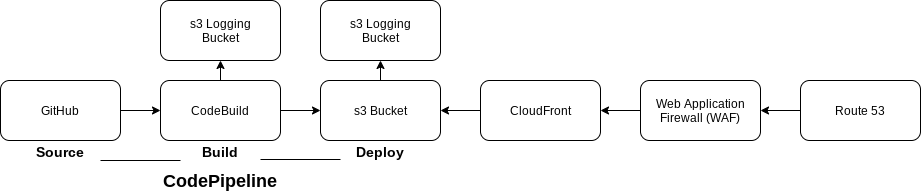
## 

## Summary

This build document describes the procedures to build the search and discovery application of geo.ca. The application is a geospatial search engine based on ReactJS and Leaflet. The application is driven by the geoCore APIs.

The application code development is hosted on Github, and is available openly. In this document, we will be automatically building our application from the GitHub branch for the environment, hosting it on AWS s3, and securing it through cloudfront and AWS web application firewall.

## Architecture



## Technical Instructions

The following technical directions will allow you to build the application in your chosen environment.

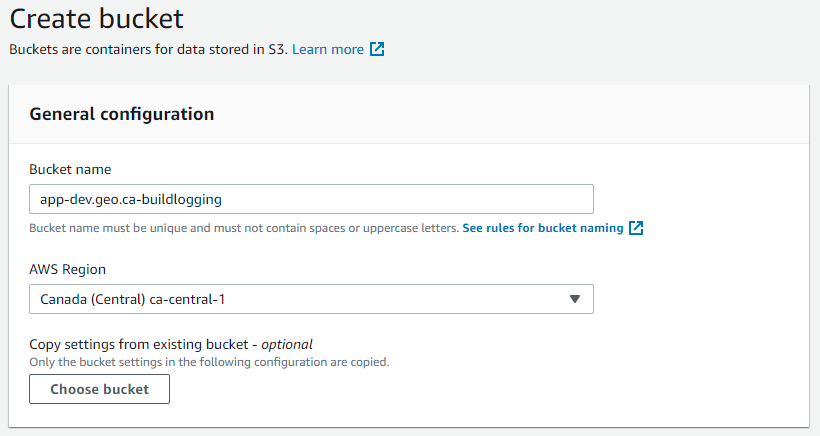
First let’s build s3 buckets for the use with our application.

*Note: Names used in the images are for illustrative purposes only.*

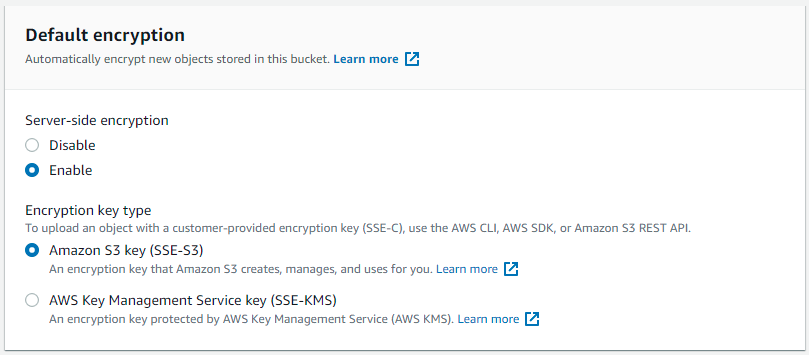
### Creating s3 buckets for the application and logging

We will need to create three s3 buckets for use with our application. One will host the application, while the other two buckets will be used for logging.

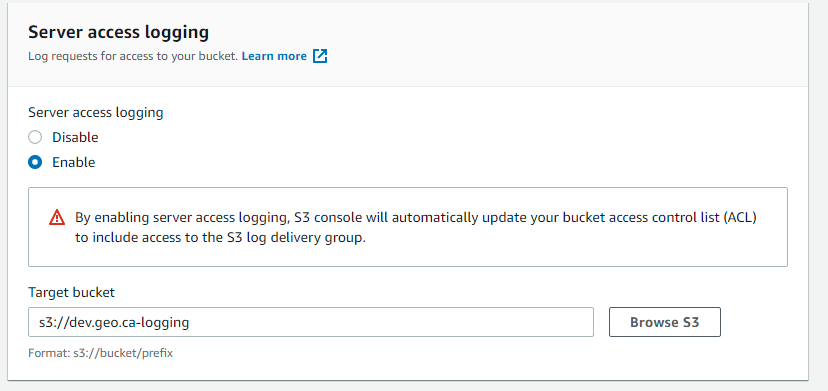
1. In the AWS Management console, navigate to the s3 interface.
2. Click Create s3 Bucket
3. For the first bucket, we will be creating a logging bucket for the logs of the CodePipeline build. For the naming convention, please include codebuild\_logging for this first bucket to easily identify the bucket for use in later stages.
4. Choose the Canada (Central) ca-central-1 region



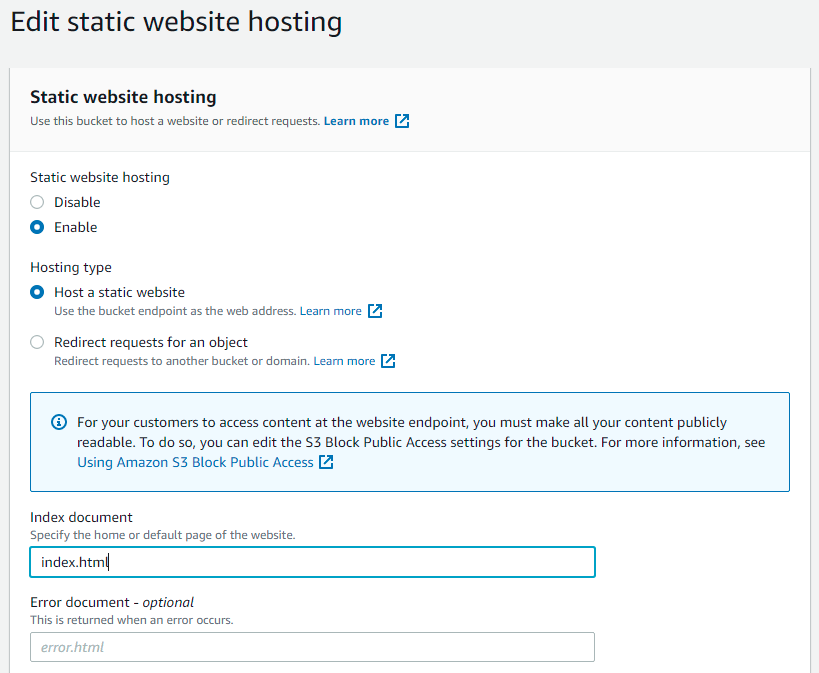
1. For our logging bucket, we will encrypt the data at rest. For default encryption, click enable and choose Amazon S3 key (SSE-S3).



1. Leave all other options as default.
2. Click Create Bucket.
3. Now we will have to create our second logging bucket for the application access logs.
4. Follow the same steps above, but include access\_log to the s3 naming convention to easily distinguish between.
5. We will need to create a third bucket, and this one will be used to deploy our application. For the bucket name, let’s name it our domain such as app-dev.geo.ca.
6. Enable Server Access Logging, and select the access\_log bucket created in the earlier step as the Target Bucket.



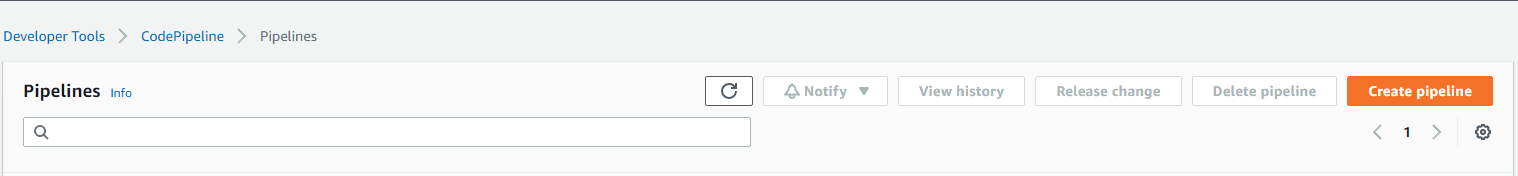
1. Leave all options as default and click Create Bucket.
2. Once created, click properties of the hosting s3 bucket from the s3 management console.
3. Enable Static website hosting



1. Click Host a static website
2. For the index document, type index.html.
3. When you return to the properties tab, copy the s3 website url for future use.

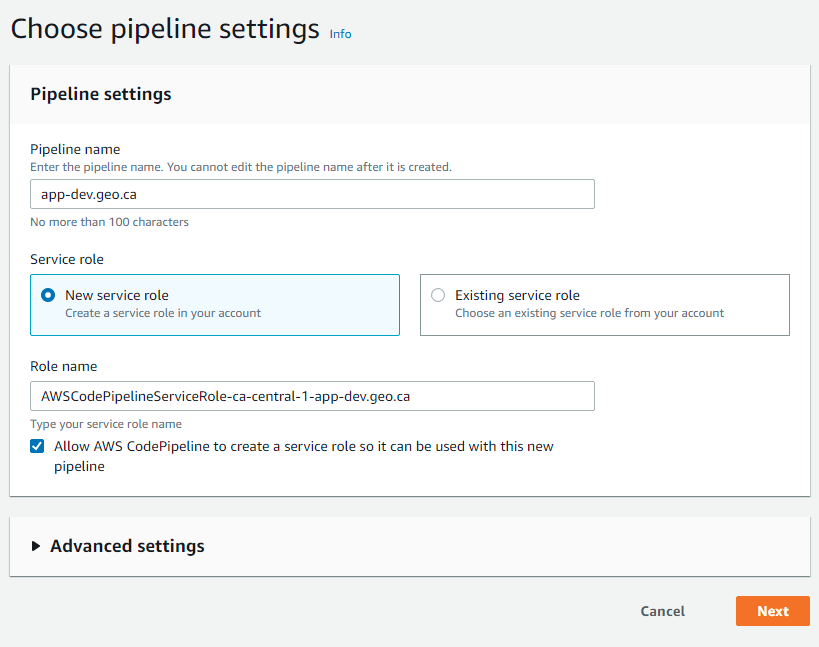
Now that we have created the s3 buckets required for the build, we will now move on to building the application.

### Creating a CodePipeline for automatic deployment

To begin the automatic deployment from the GitHub repository we will be setting up a codepipeline on AWS (Amazon Web Services)  ****

1. In the AWS management console, navigate to the CodePipeline interface.
2. Click Create Pipeline.

The pipeline settings dialog box will open.



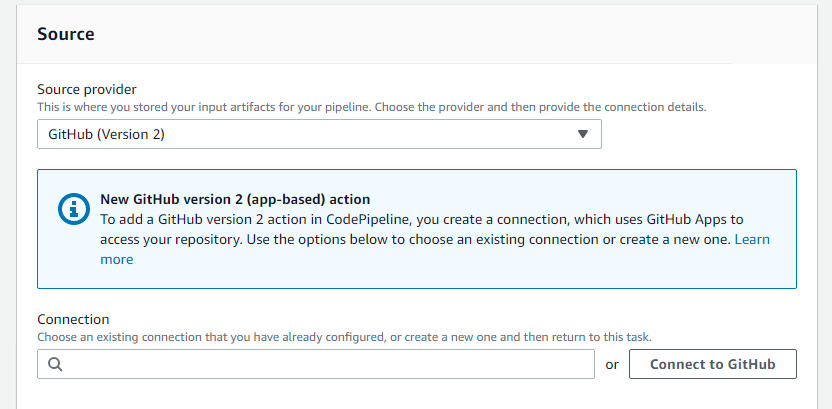
1. Add a pipeline name. For this application please follow this naming convention:

Environment

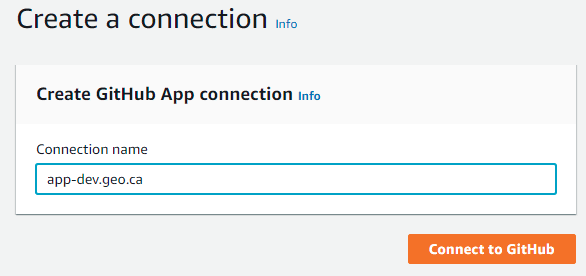
| Environment | Naming Convention |
| --- | --- |
| Development | app-dev.geo.ca |
| Staging | app-stage.geo.ca |
| Production | app.geo.ca |

1. Allow for a New service role to be created.
2. Click Next.

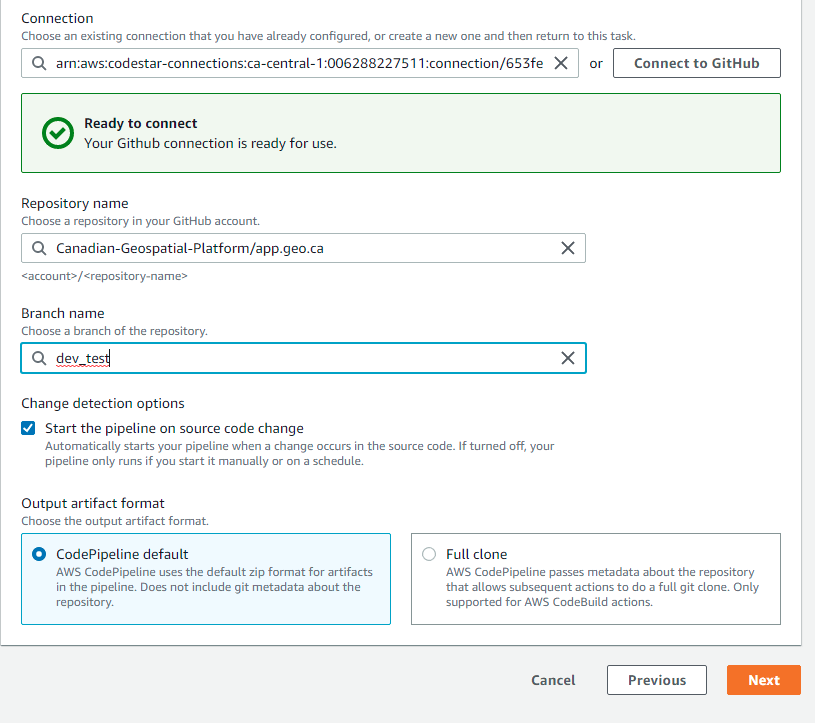
We will now connect the source of the automatic build of the application.



1. Set the source provider to GitHub (Version 2)
2. Click on Connect to GitHub



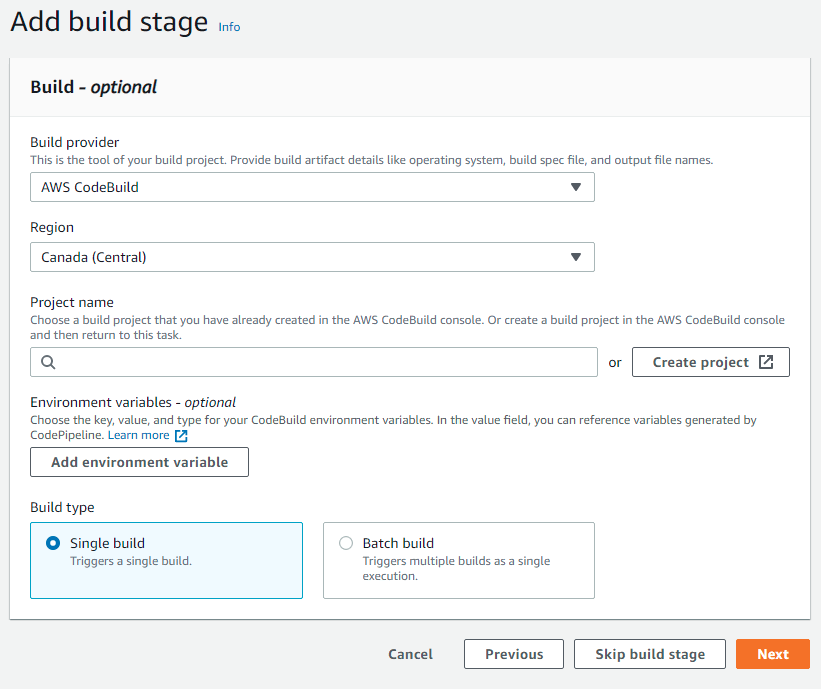
1. Name the connection the same as the environmental naming convention.
2. Click Connect to GitHub
3. Authorize the GitHub connection to the Canadian Geospatial Platform GitHub organization, and the app.geo.ca repository.



1. After the connection to GitHub has been completed, choose the repository from the Repository Name dropdown menu.
2. Select the branch you would like to use as the source of the build.
3. Select the “Start the pipeline on source code change” so that the build starts every time the GitHub repository is updated.
4. Allow the output artifacts to be placed in the CodePipeline default location.
5. Click Next.

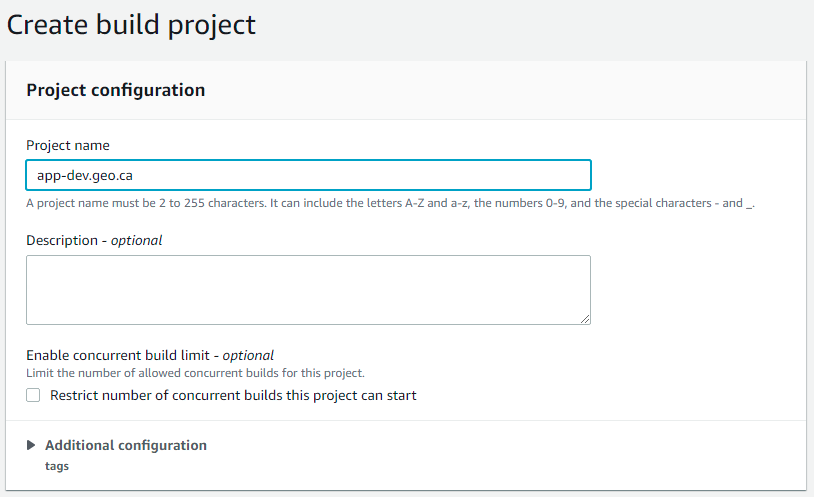
The add build stage dialog box will open.

1. Choose AWS CodeBuild as Build provider
2. Choose the Canada Central region.
3. Click Create Project

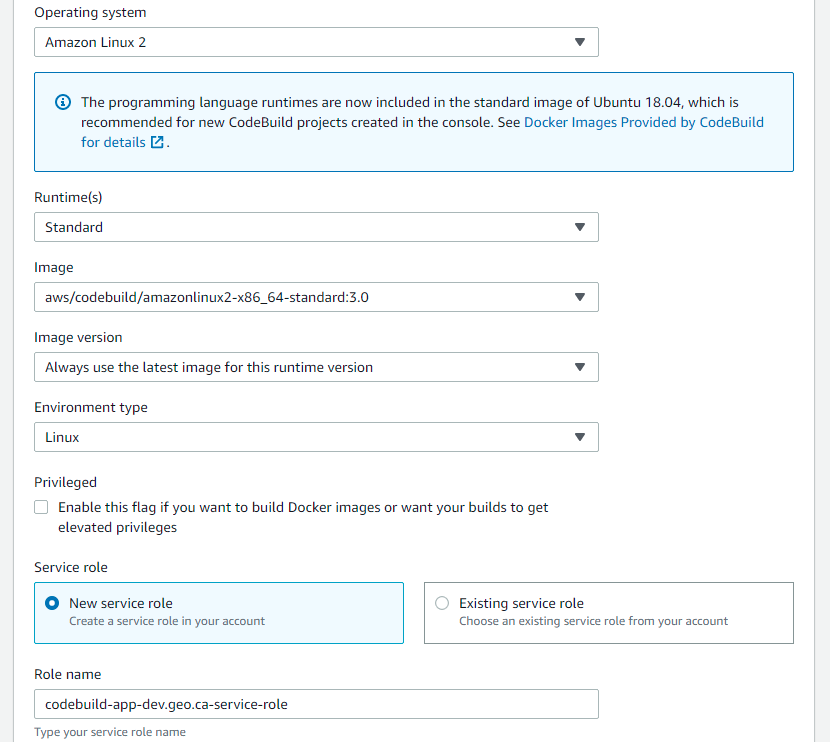


The Create build project dialog box will open.

1. Choose a project build name



1. Select Managed Image
2. For Operating system, choose Amazon Linux 2
3. Choose Standard for the Runtime.
4. Select aws/codebuild/amazonlinux2-x86\_64-standard:3.0 as the image, and Always use the latest image for this runtime version.
5. Environment type should be Linux
6. Allow CodeBuild to create a service role.



1. Click Insert build commands and switch to the editor.
2. For the build commands, insert the following code:

Version: 0.2

env:

Git-credential-helper: yes

phases:

install:

runtime-vestions:

nodejs: 12

pre\_build:

commands:

* npm install

build:

commands:

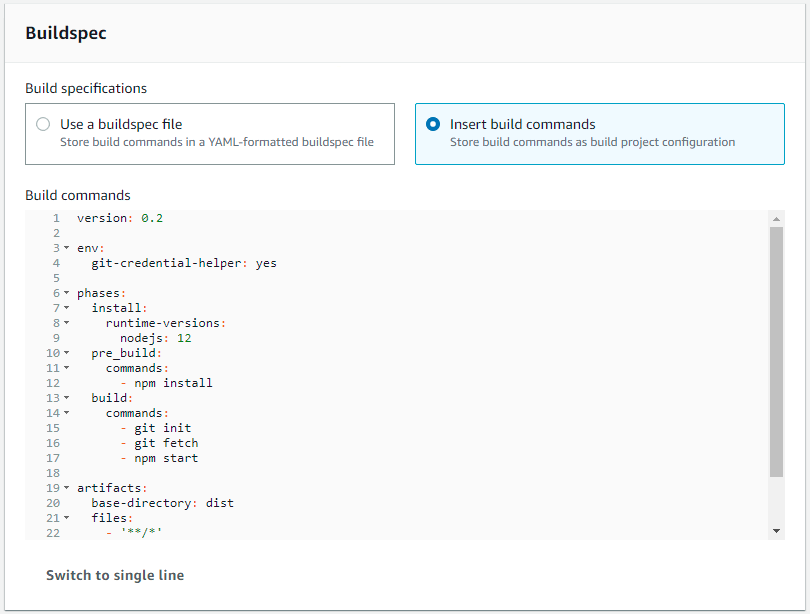
* git init
* git fetch
* npm start

artifacts:

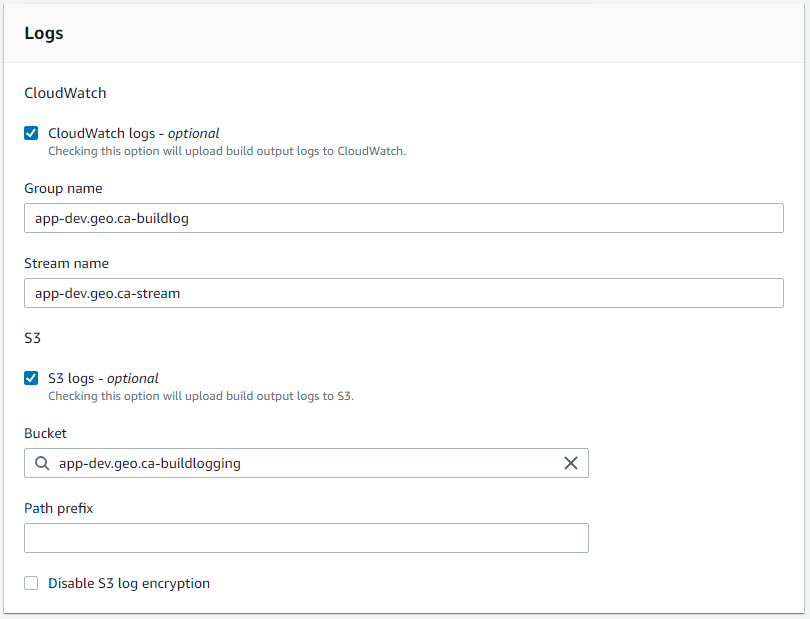
base-directory: dist

files:

* ‘\*\*/\*’

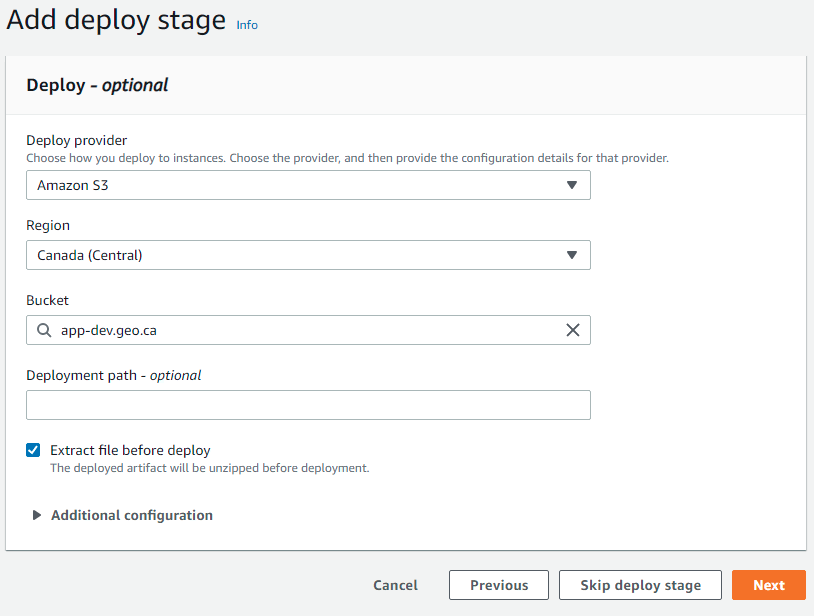


1. For the logs, for the Group name add -buildlog at the end, and -stream for the stream name.
2. Click S3 logs option
3. Choose the codebuild\_logging s3 bucket created earlier in this document.



Once the CodeBuild project configuration is complete, you will return to the CodePipeline build configuration.

1. Click Next
2. The deploy stage configuration dialog box is displayed.
3. Add Amazon S3 as the deploy provider
4. Select the Canada (Central) Region
5. Choose the bucket created for hosting the application earlier in this document.

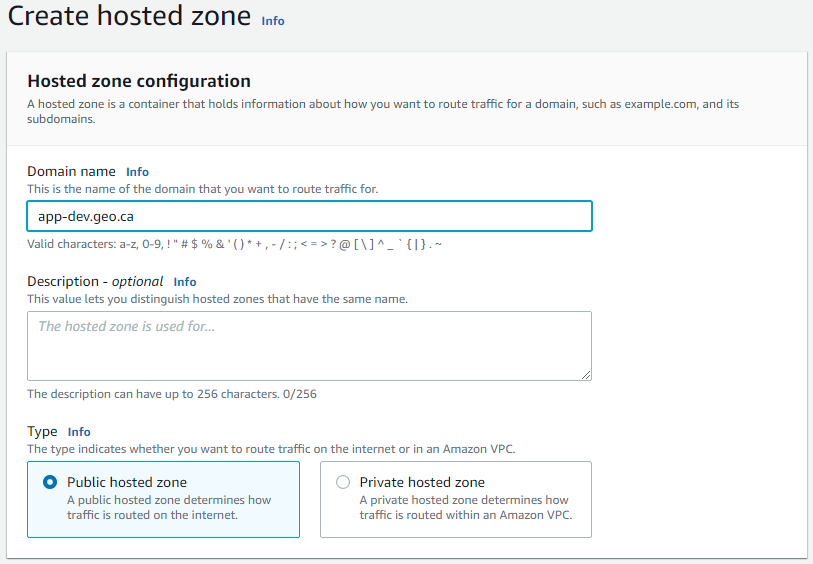


1. Click the checkbox next to Extract file before deploy
2. Click Next
3. Review the CodePipeline settings and Click Create Pipeline.

### Creating a Route 53 Record

Now we will create a Route 53 record and certificate for use with the domain.

1. In the AWS Management Console, click on Route 53.
2. Once you are on the Route 53 console, click Create hosted zone



1. Enter your Domain name for the environment.

Environment

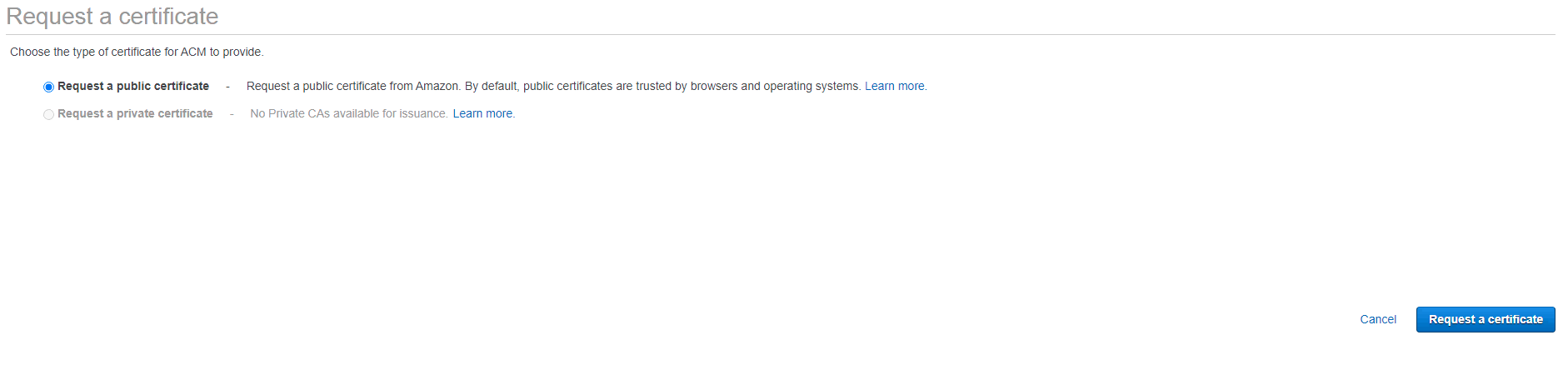
| Environment | Naming Convention |
| --- | --- |
| Development | app-dev.geo.ca |
| Staging | app-stage.geo.ca |
| Production | app.geo.ca |

1. Click Public hosted zone
2. Click create hosted zone. If this is the first time creating the environment, please forward the 4 NS records to the cloud team to redirect traffic from the main geo.ca domain.

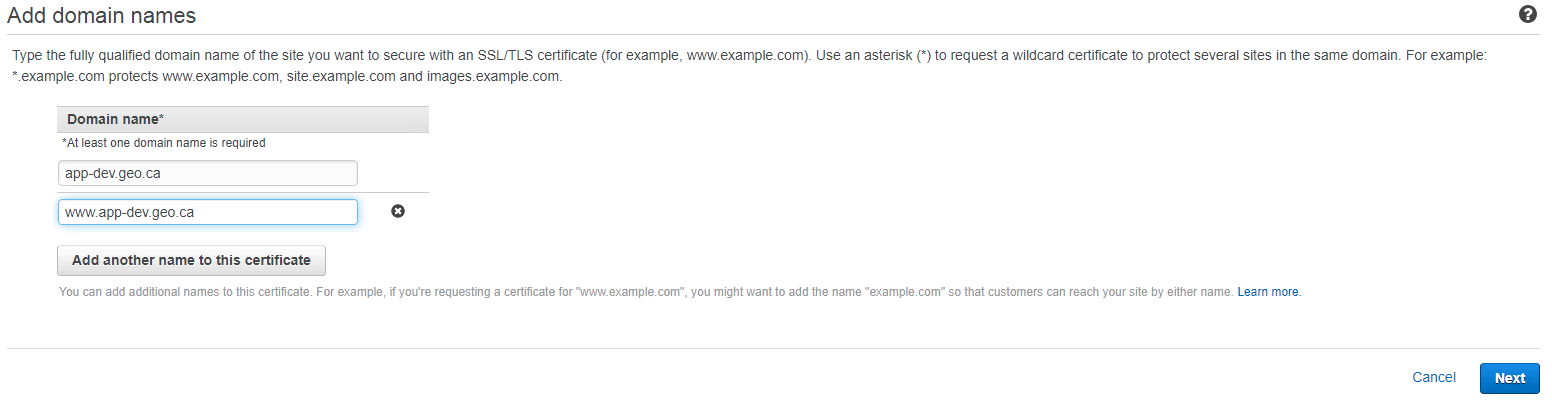
### Creating a Certificate

Next, in the AWS management console go to Certificate Manager.

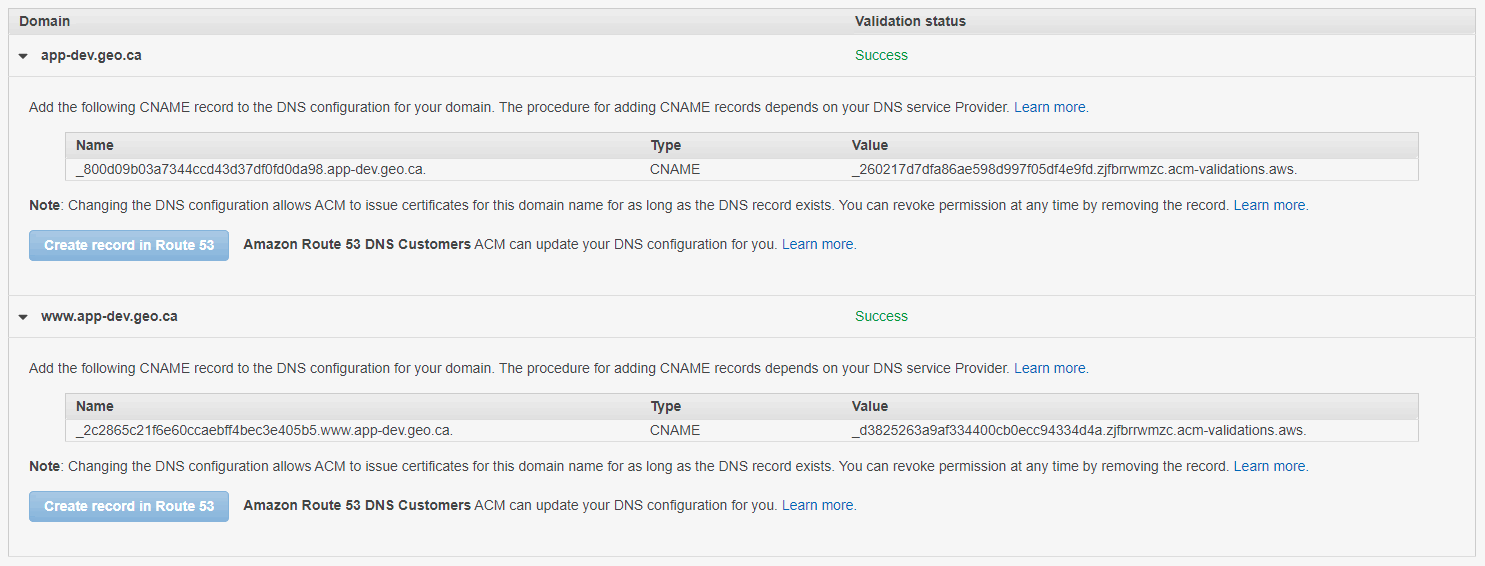
1. Select the US East (N. Virginia) us-east-1
2. Click Request a Public Certificate



1. For the Domain Name, add the environment domain name, and add a second domain name for the certificate with www added.



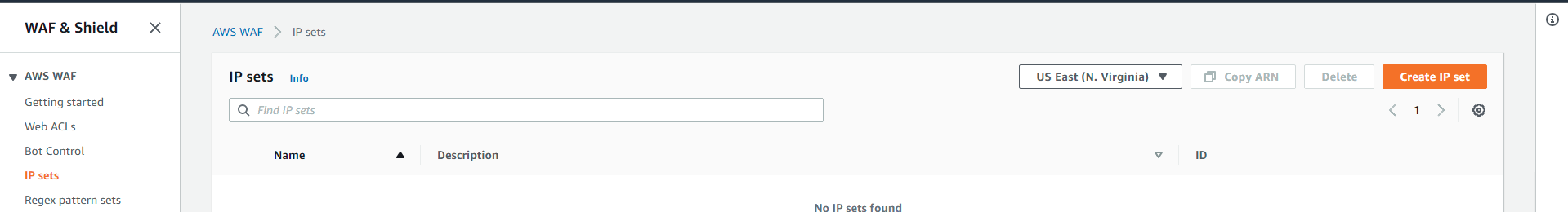
1. Click Next
2. Click DNS validation, and click Next
3. Since we’ve created a Route 53 record, for each domain name on the certificate click Create Record in Route 53 to validate the record.



### Creating a Web Application Firewall

For the Development and Staging Environment, the website/application should only be accessed internally. This can be created using the AWS WAF service to create the IP Set and the ACL rule.

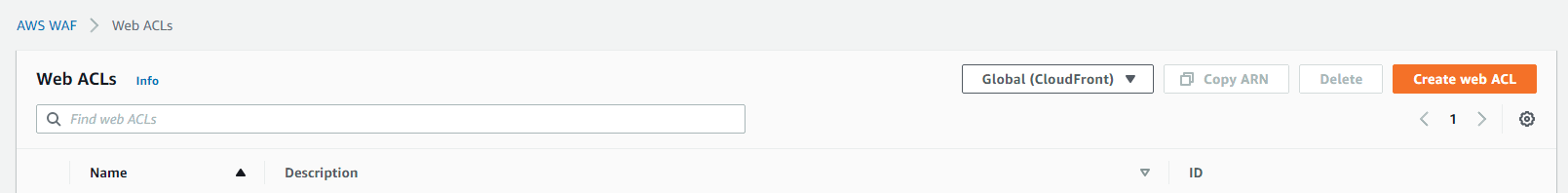
1. In the AWS Management Console, navigate to the AWS WAF tool.
2. Click IP Sets
3. Click Create IP Set



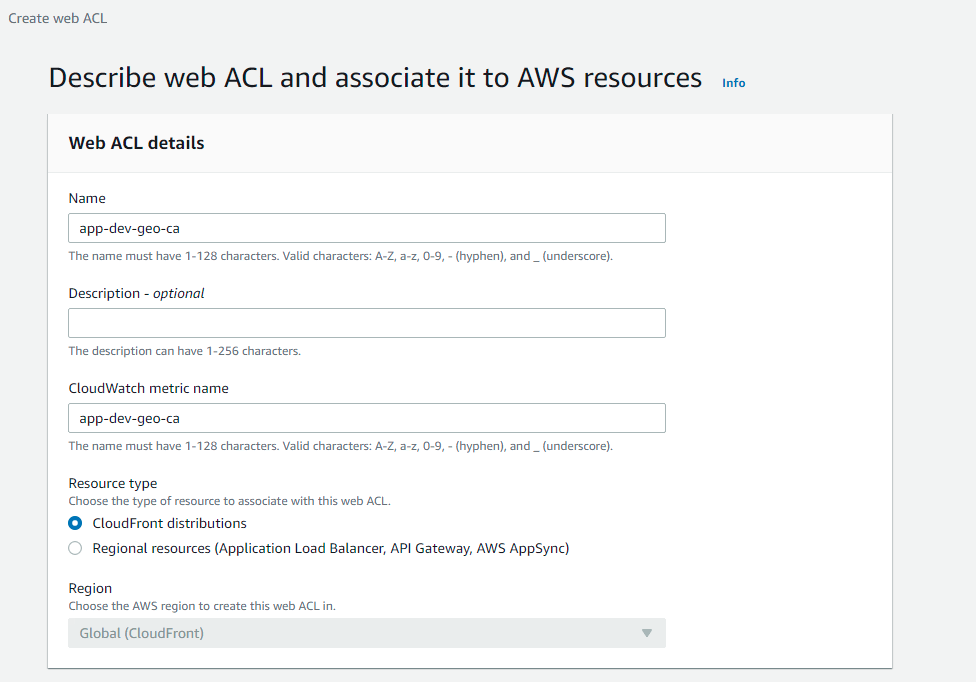
1. Name the IP Set
2. For Region select Global (Cloudfront)
3. Click Create IP Set

Next we will create the Web ACL for the development and staging environments.

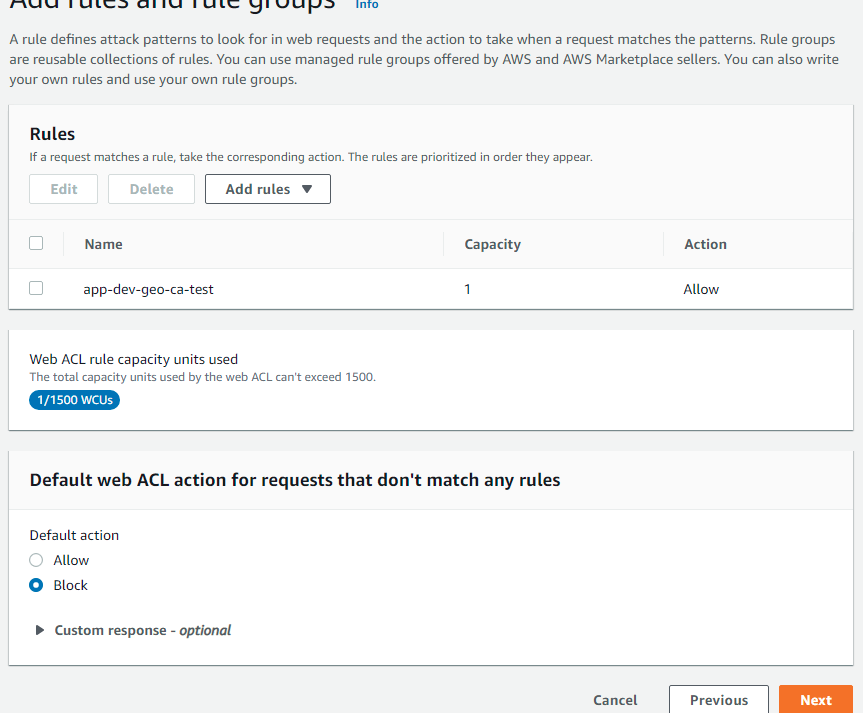
1. On the left hand menu, click Web ACLs.
2. Click Create web ACL



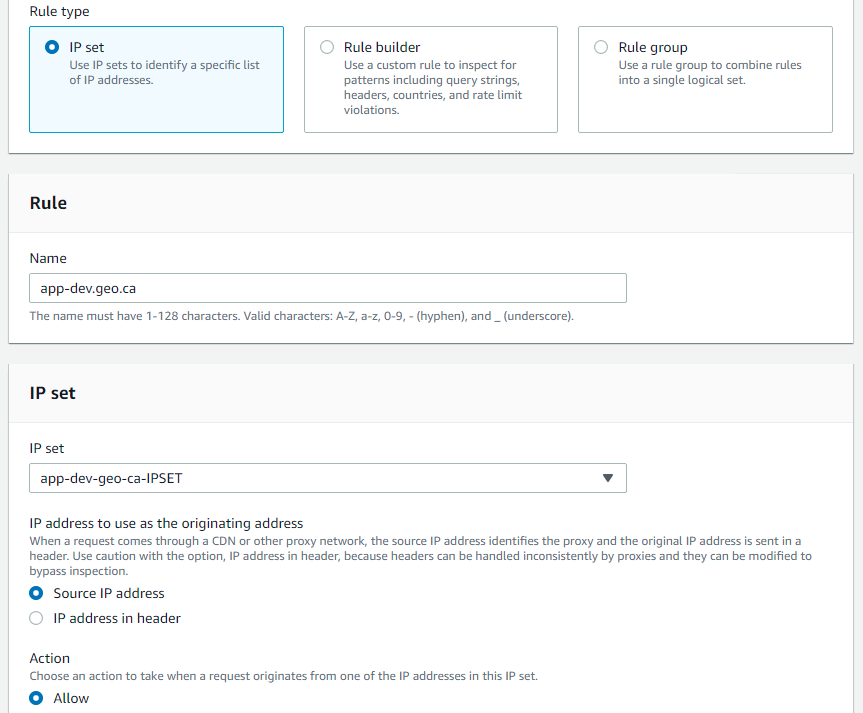
1. Click CloudFront distributions under Resource Type, which changes the Region to Global.
2. Click Next.



1. On this page click Default Action as Block



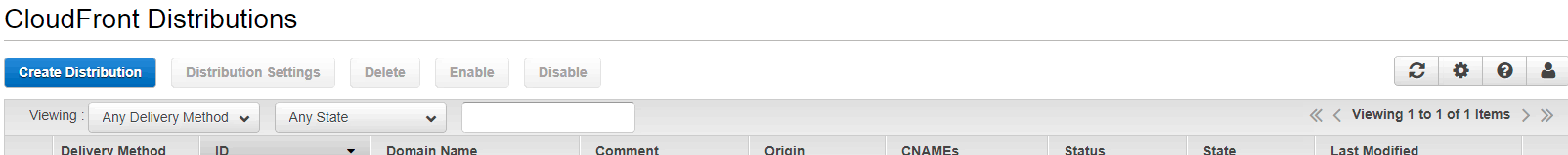
1. Click Add rules, and then click Add my own rules and rule groups



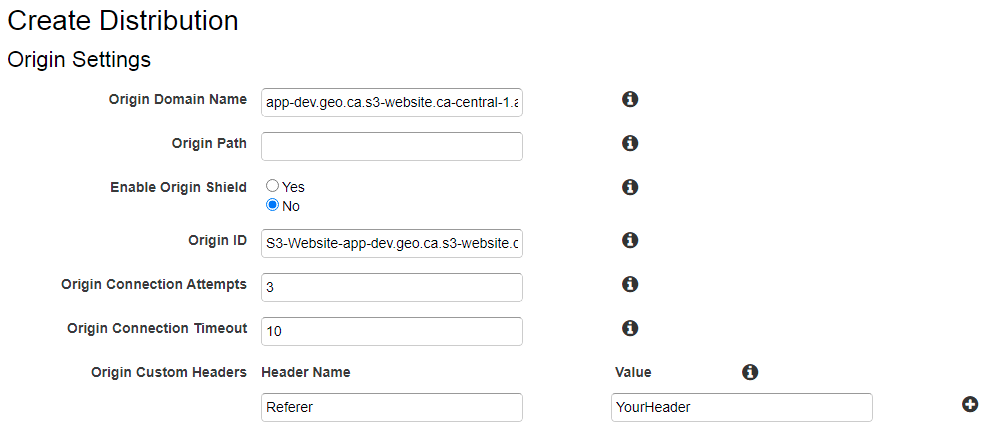
1. Click IP set
2. Name the rule by the environment domain
3. Select the IP Set created in the previous steps
4. Click Allow in the Action
5. Click Add Rule
6. You will return back to the main Web ACL configuration, and Click Next
7. Click Next and Create the Web ACL.

### Creating a Cloudfront distribution

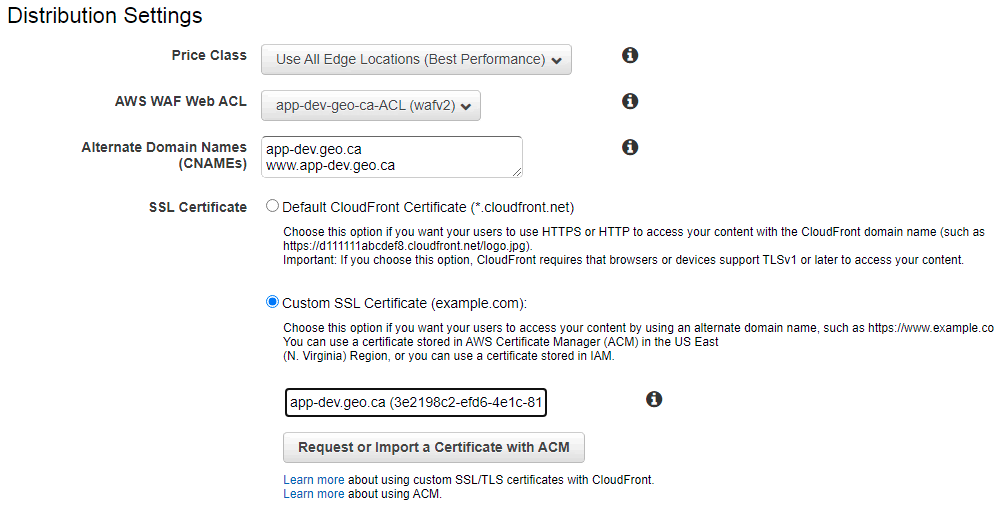
1. Navigate to the Cloudfront management console, and select Create distribution.



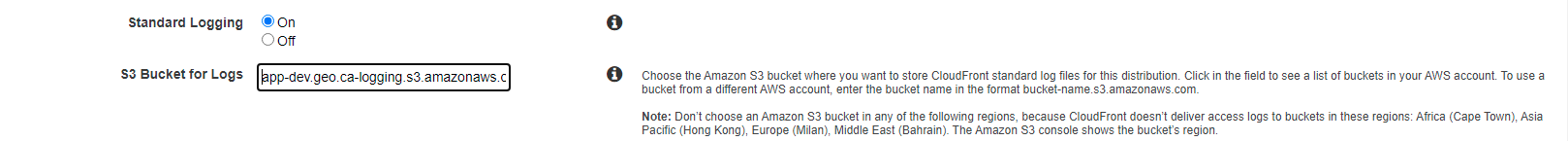
1. Click Get Started
2. For Origin Domain Name, use the s3 bucket static website url that was set up for the application hosting.
3. For Header Name, type in Referer
4. For Header Value, type a value of your choice for use to secure the s3 bucket for access only from the distribution.



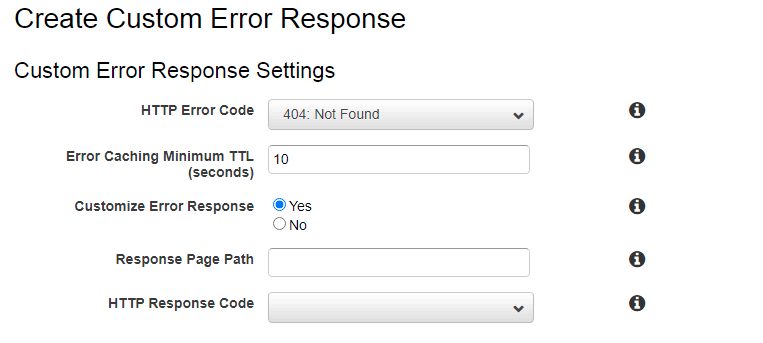
1. For the AWS WAF Web ACL Distribution setting, select the WAF ACL created in a previous step.
2. For Alternate Domain Names (CNAMEs), type the environment domain name, and add a second domain name with www added.



1. Select Redirect HTTP to HTTPS
2. Click Custom SSL Certificate and select the one created in the create certificate steps.
3. Enable Standard Logging and enter the access\_log s3 bucket created in the create s3 bucket step in the following format: \*yourbucket\*.s3.amazonaws.com



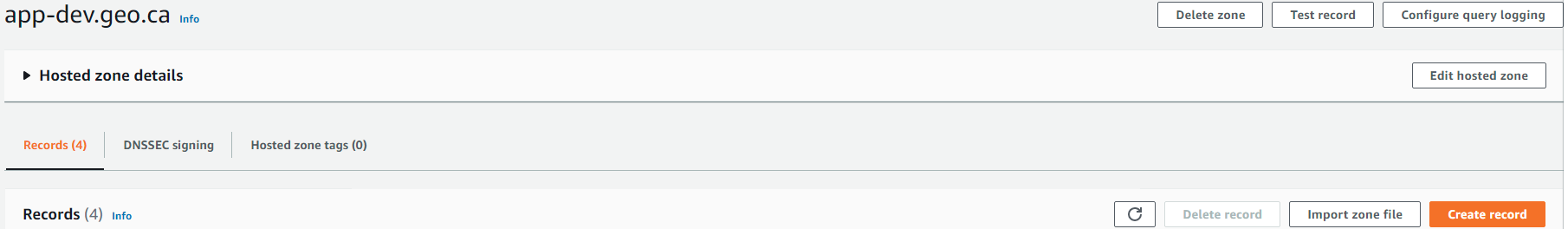
1. Click Create Distribution
2. Once the distribution is created, Click on the distribution and then click Error Pages.
3. Click Create Custom Error Response
4. Select HTTP error code 404 Not Found
5. Click Yes on Customize Error Response
6. In Response Page Path type /index.html
7. For HTTP Response code select 200 OK
8. Click create.
9. This will now route multiple paths of our application back to the application routing system.



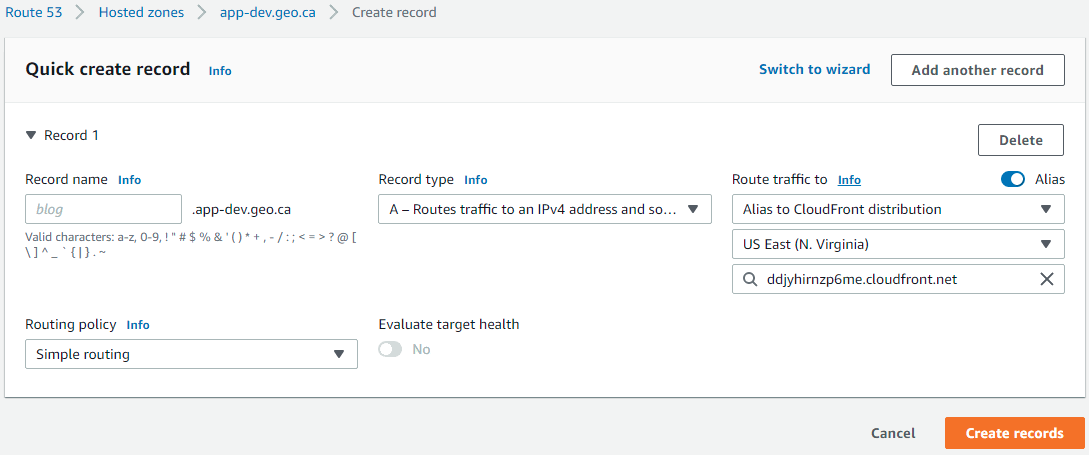
### Creating a Route 53 Alias record

In order for the website to be accessed from the browser from users there are two “A” records that need to be created. The first record is without the “www” for the URL and the second record is with “www” in the URL. Currently, the website can only be accessed by the full name including the hosted zone.

1. Using Route 53, click Hosted Zones.
2. Click on the environment domain name
3. Click Create record.



1. Select Simple routing, then click Next
2. Click Define Simple Record.
3. Record name: <leave blank>
   1. This will allow the user to access the website without typing “www” into the browser.
4. Value/Route traffic to
5. Turn on the Alias toggle on the right side of the forum.
6. Select Alias to CloudFront distribution
7. Select US East (N. Virginia)
8. Select the created Cloudfront distribution record as created in a previous step.
9. For Record type, Select A - Routes traffic to an IPv4 address and some AWS resources.
10. Click Define simple record.



### 

We will need to define another record for users who type “www” before the main domain. For this follow steps 102 to 110 while adding www as the Record name.

### Securing the s3 bucket to only accept access from CloudFront

We will be using the Referer header from the CloudFront distribution to secure the s3 bucket. To do this, navigate to the s3 console and the permissions tab of the hosting bucket.

1. In the Bucket Policy, Copy the following code replacing the Referer header in the CloudFront distribution setting

{

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

"Id": "http referer policy example",

"Statement": [

{

"Sid": "Allow get requests originating from www.app-dev.geo.ca and app-dev.geo.ca.",

"Effect": "Allow",

"Principal": "\*",

"Action": [

"s3:GetObject",

"s3:GetObjectVersion"

],

"Resource": "arn:aws:s3:::dev.geo.ca/\*",

"Condition": {

"StringLike": {

"aws:Referer": "ebf35beeac2a558cb5626e6f26eeb13f3b21b8a4"

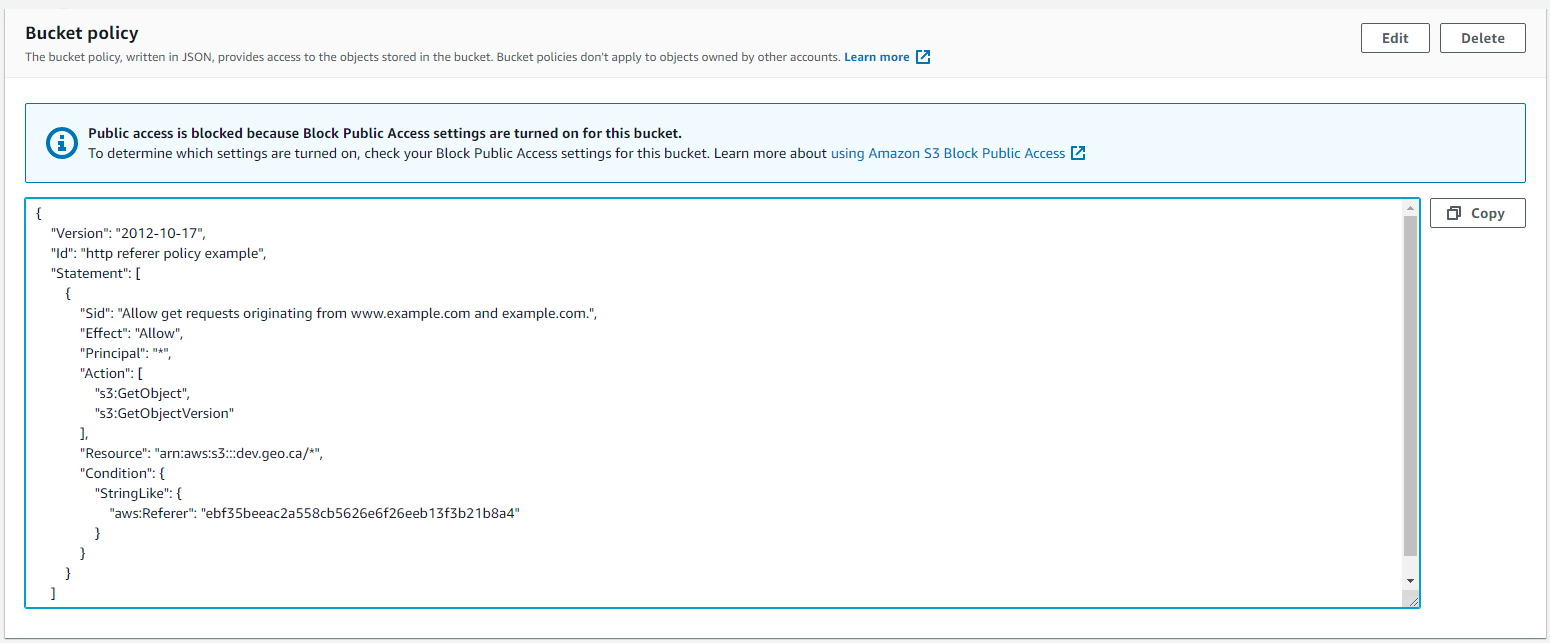
}

}

}

]

}



1. Click Save Changes

You should now be able to view the web presence site from the environment domain. Please confirm that you can not view the application from the s3 website endpoint if you are in the development or staging environments.

### 