**PROJECT-2**

**Deploying Website using AWS Console**

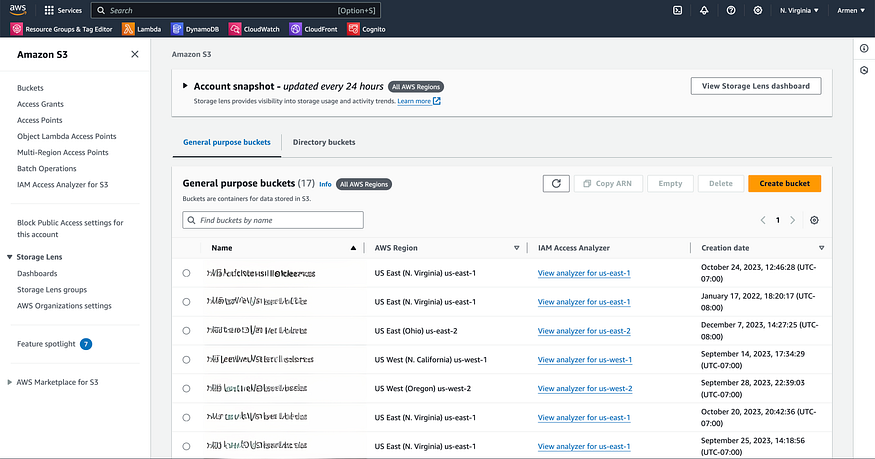
This section covers deploying the website using the AWS Console. While not the recommended method, it’s might be useful for understanding core concepts. Also this method doesn’t require you to know any coding knowledge.

**Prerequisites**

* An AWS account

**Configure the S3 bucket**

Open a browser and go to the AWS console, and login.



1. Navigate to the Amazon S3 Service.
2. Click on the **Create New Bucket** option.
3. In the Create Bucket page, under General configuration, select **General purpose** from the bucket options.
4. Name your bucket something that you will remember, I have named it *“temp-console-website”*, you could name it the same things to make following steps easier.
5. In the **Object Ownership** section, select ACLs disabled.
6. In the **Block Public Access settings for this bucket** section, uncheck the **Block all pubic access**, and make sure all the other four checkboxes have been unchecked as well. You will notice a yellow alert box popping up with the following message: *Turning off block all public access might result in this bucket and the objects within becoming public*
7. Select or check the **I acknowledge that the current settings might result in this bucket and the objects within becoming public** box.
8. In the **Bucket Versioning** select Disable versioning.
9. everything else should be set to default, select **Create bucket**
10. This is an optional step, but to make things simple for us at the end where we need to destroy all resource, add a tag. Scroll down to the **Tags** section and add a tag by click on edit, and adding a new tag with the key value of *temp-project*.

**Note**

The name is crucial to remember, as we’ll use it again when specifying the bucket Resource value in the next section. This ensures correct resource linking in later steps.

**Upload the files to S3**

1. Select the newly created bucket from the bucket list page. If you set the name to be *“temp-console-website”* then that will show up in the list.
2. Click on the **Upload** button
3. Select **add files**, then select all the files in the *dist* folder from the NextJs folder, and then click the **Upload** button to upload all the files. a progress bar will be displayed showing the upload progression. Once upload to the next step.
4. Go to the **Properties** tab, then scroll down to the **Static website hosting** section and click Edit.
5. Under the **Static website hosting** section select *Enable*.
6. For the **Hosting type** select *Host a static website*
7. There is an input field under **Index document**, enter index.html in that field.
8. enter error.html for the **Error document** but this is optional.
9. Click **Save changes**
10. Go into the **Permissions** tab
11. Scroll down until you see the **Bucket Policy** section. Click on the **Edit** button, then paste the following policy in there

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Effect": "Allow",  
 "Principal": "\*",  
 "Action": "s3:GetObject",  
 "Resource": "arn:aws:s3:::temp-console-website/\*"  
 }  
 ]  
}

12. Click **Save changes**

**Critical**

the “temp-console-website” is the name I chose, if you have chosen a different name, then replace this value in the JSON object.

We’ve completed the S3 configuration. To view the website directly from S3, go to the **Properties** tab and scroll to **Static website hosting**. At the bottom, you’ll find a URL under “Bucket website endpoint”. The URL should look similar to this:

[http://temp-console-website.s3-website-us-east-1.amazonaws.com](http://temp-console-website.s3-website-us-east-1.amazonaws.com/)

you should see the same NextJs page we saw when we ran locally.

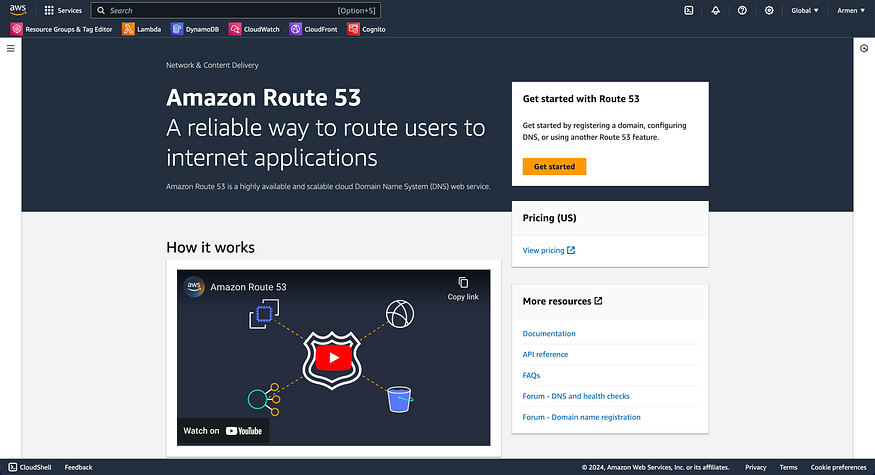
**Configuring DNS**

In this section we will configure to point a domain name to this CloudFront distribution or our website. If you already have a domain you purchased from a third party domain provider like GoDaddy, Namecheap, Google domain etc you have to login and change some of the settings.

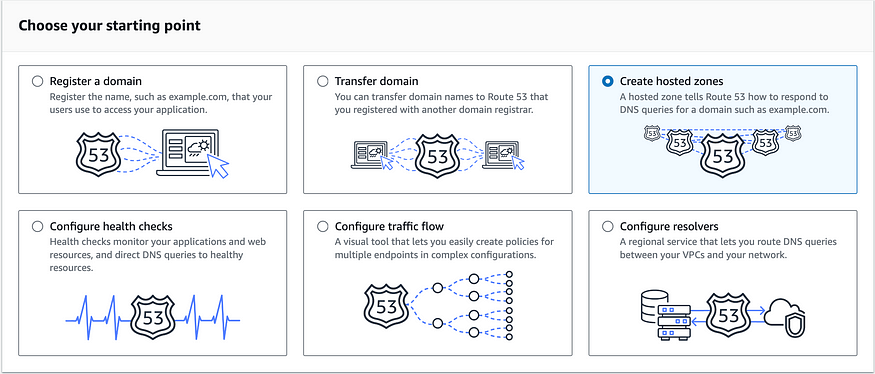
We will be moving back and forth between DNS, CloudFront and Certificate Manager in order to get this configured

First lets add the domain to AWS Route 53. Route 53 is a DNS service provided by Amazon. Lets add our domain. For this example I have a test domain named redrobotexample.com. This is a domain I use to try and test different applications.

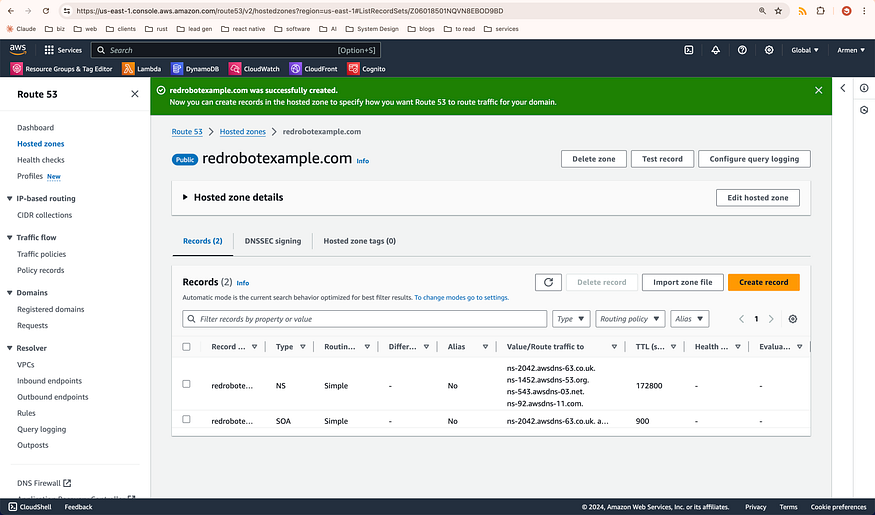
Go to the Route 53 page:



1. Click on the **Get started** button, and you will be presented with the following view:



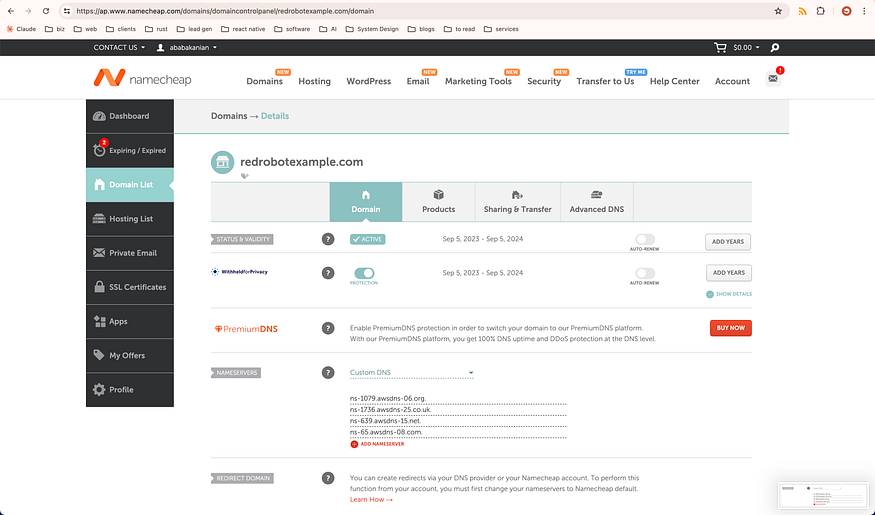
1. Select the **Create hosted zones** option and click **Get started**
2. In the Domain name field enter your domain name, ie redrobotexample.com, and then click **Create hosted zone**. Once done you will be presented with this page:



4. Expand the **Hosted zone details** area by clicking on the triangle next to it. you will be presented with a few details. From that list, you need to copy the **Name servers** value and paste them in your 3rd party domain provider DNS provider. In your 3rd party domain provider, find where you can define custom dns servers and paste the values, which should look like this:

ns-2042.awsdns-63.co.uk  
ns-1452.awsdns-53.org  
ns-543.awsdns-03.net  
ns-92.awsdns-11.com

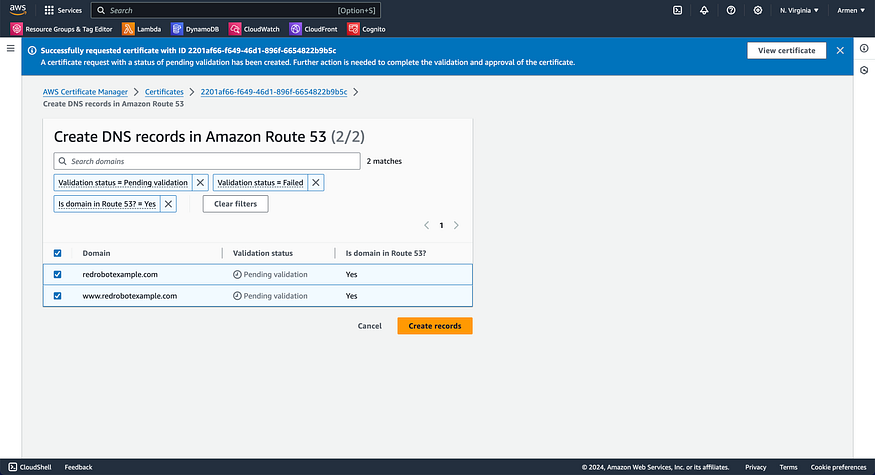
For example, in namecheap this is the section where you define custom DNS values:



5. Scroll down to the **Tags** section and add a tag and select the *temp-project* value.

**Creating a Certificates**

1. Open a new tab or browser windows, and go to the AWS Certificate Manager or (ACM)
2. Click on **Request certificate**,
3. In the next page select **Request a public certificate**
4. In the next page, for the **Fully qualified domain name** enter your domain name, ie redrobotexample.com in our example,
5. In the same section click on **add another name** for the certificate and enter the same domain but add the the www subdomain to it. So for our example it would be www.redrobotexample.com then click **Request**
6. In the next page, scroll down to where it says **Domains**, you will notice two CNAME values are presented — these need to be added ro your DNS entry so AWS certificate Manager can verify that the domain belongs to us. S we need to add those to the Route 53 entries and we can do that by clicking on the **Create records in Route 53** in that same view (you can do this manually but AWS made things simple for us)
7. In the next page, since we have been using the same domain name for our DNS values, AWS will filter the right DNS entries for us to select.



Click the **Create Records in Route53**

8. In the other tab, that has the Route 53 page open, verify the two new Route53 entries have been added to the record list

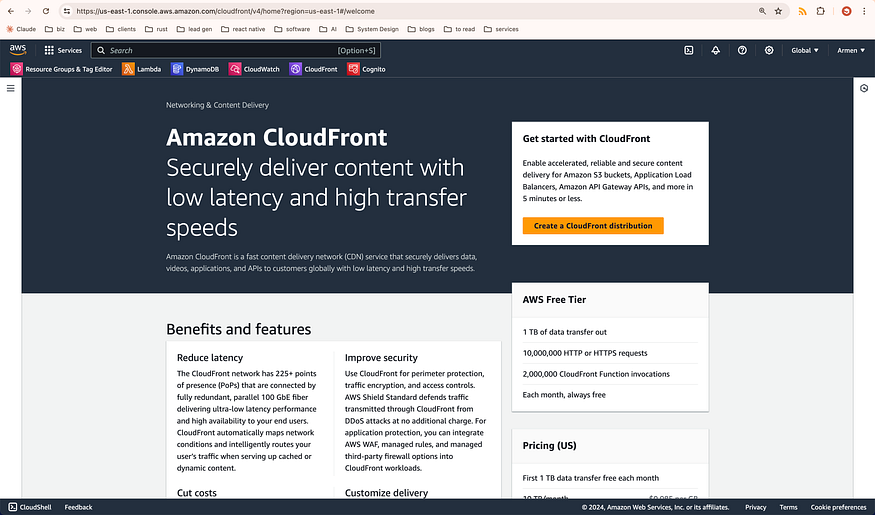
9. After this, we need to wait for the status of this certificate to change from “pending validation” to validated. In the Certificate Manager page, check the value of **Status**. Once the status changes to “Issued” move to the next step.

10. Scroll down to the **Tags** section and add a tag by click on edit, and adding a new tag with the key value of *temp-project*.

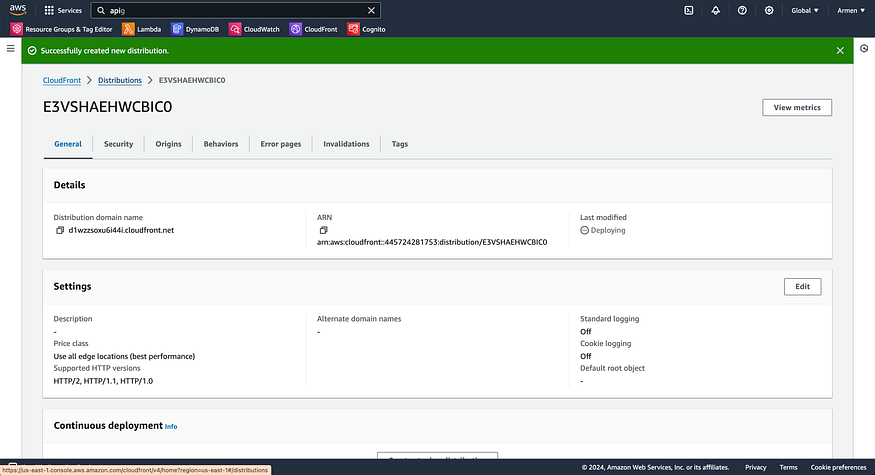
**Configuring CloudFront**

We have defined our DNS records and created our certificates, we would like to update the page so it points to a domain instead of the S3 generated URL. For that we need to setup a AWS CloudFront entry. CloudFront is a CDN service provided by AWS.

Go to the AWS CloudFront page



1. Then click on the **Create a CloudFront Distribution**, the **Create distribution** page will show up
2. In the **Origin domain** field or dropdown, click on the field and the dropdown list will appear. From that list select the S3 entry we created in the last part. If you picked the same name from this guide, it should look something like this: temp-console-website.s3.us-east-1.amazonaws.com
3. A pop should be displayed that says “This S3 bucket has static web hosting enabled. If you plan to use this distribution as a website, we recommend using the S3 website endpoint rather than the bucket endpoint.” Click on the **Use website endpoint** button to do this.
4. In the Web Application Firewall (WAF) section, select *Do not enable security protections*
5. Do not change any of the other configurations, we will come back here to update the config once we define our domain and certificates but for now, go ahead and click **Create distribution**.
6. Once distribution is done, you can copy the URL value under **Distribution domain name** and paste it in the browser. the value should look like something like this d1wzzsoxu6i44i.cloudfront.net You should see the same page now as before
7. In the **Custom SSL certificate — optional** field, select the certificate we just created.
8. In the Alternative Domain names, add your domain name with and without the www subdomain, so essentially we are adding two values. In this example it would be: redrobotexample.com
9. Scroll down to the **Tags** section and add a tag by click on edit, and adding a new tag with the key value of *temp-project*.
10. click on **Create records**



**Note**

It might be easier to open a secondary tab, and go to the CloudFront page and check the name to make sure the value is the correct one.

**Updating DNS**

The last step is to update our DNS to point to the CloudFront CDN. With the certificates and CDN in place, we can add a record in our Route53 to our CloudFront dist.

1. In the same Route 53 page, under the Records section click on **Create record**.
2. For the **Record Type** select *A — Route TRaffic to an IPv4 address*
3. Enable the *Alias* toggle
4. For the **Route traffic to** select the *Alias to CloudFront Distribution*
5. and then on the bottom selector, select the CloudFront distribution we created in the last step.

you can do the same once more but add the www subdomain as well.

We are done with out configuration, If you followed the steps exactly as described you should be able to go to your domain, (in our case redrobotexample.com) and view your website.

**Destroying all resources**

As you can see, the steps involved to get this up and running are tedious, and error prone and most importantly not repeatable. You will also notice that destroying these resources suffer from the same problem as we have to manually go to each service and destroy individual services.

Before doing thing using IaC, lets delete/destroy all the created resources.

1. Go to the **Resource groups** page
2. Click **Create resource group**
3. Select the **Tag based**
4. Under the **Grouping criteria** enter *temp-project* in the tag field
5. Click **Create group**
6. Then after creation, select the group
7. On the group’s detail you will see each resource listed. The order of the resource might be different for you, so lets go over each resource by type and describe how to completely destroy them — the order which you need to destroy resource is important since we have created dependencies between resources.
8. Open the Route 53 resource on a new tab or window.
9. Check the 3 Record that have the record Type A and the two CNAME’s. **DO NOT DELETE** the NS and SOA records.
10. close the tab, now open the CloudFront resource in a new tab
11. Remember the distribution name, and go into the Distribution list view and select that item and click **Disable**
12. Once Disabled, then click **Delete**
13. Close the tab, now open the S3 resource
14. Select all files from the list and click **Delete**
15. A new page is presented, which will ask for your confirmation, enter the text *permanently delete* in the bottom text box then click **Delete objects**
16. Once deleted, click on the **Buckets** item to view the list of Buckets
17. Select the bucket (should be named temp-console-website) and click **Delete**
18. A similar confirmation page is shown, enter the bucket name in the text field to confirm bucket deletion and click **Delete bucket**
19. Close the tab, now open the certificate manager view
20. Click on the **Delete**. In the confirmation pop-up enter delete and then Delete again.
21. close the tab, verify that all resource have been deleted from the resource group view
22. If no other resource is left continue to the next section.

At the end of the next section you will see that destroying resources is a single command.

**Using AWS CDK**

By leveraging IaC, you can automate and replicate all the setup and configuration work typically done through the console. This approach significantly speeds up your website deployment process. It allows you to maintain both your website structure and content in code, enabling quick redeployment when you need to add a blog post or change content.

A key advantage of this method is its elimination of the need for a database to store and retrieve website content, or a server to render it. Instead, your entire site can be managed and updated directly through code. This approach simplifies maintenance, reduces costs, and improves scalability and performance and you can use a code subversion system like git to track all changes including the content.

Lets automate all of the steps we took using AWS CDK, which is an IaC or Infrastructure as code framework. All the steps we performed manually it can be automated and also we can add logic as well so based on some condition, ie if it’s a production deployment we can add a few other steps vs if it’s a test deployment we can skip a few other steps.

**Prerequisites**

* An AWS account
* Node.js and npm installed
* AWS cli and cdk setup and configured
* Basic knowledge in Javascript and Typescript

**Setup, Code and Deploy**

1. Initialize a new CDK project:

mkdir temp-project && cd temp-project  
mkdir infrastructure && cd infrastructure  
cdk init app --language typescript

in addition, move the frontend folder we created from the first section of the guide into temp-project folder.

2. Replace the content of the file lib/infrastructure-stack.ts with:

import \* as cdk from "aws-cdk-lib"  
import { Construct } from "constructs"  
import \* as route53 from "aws-cdk-lib/aws-route53"  
import \* as acm from "aws-cdk-lib/aws-certificatemanager"  
import \* as s3 from "aws-cdk-lib/aws-s3"  
import \* as s3deploy from "aws-cdk-lib/aws-s3-deployment"  
import \* as cloudfront from "aws-cdk-lib/aws-cloudfront"  
import \* as route53Targets from "aws-cdk-lib/aws-route53-targets"  
   
export class InfrastrucureStack extends cdk.Stack {  
 constructor(scope: Construct, id: string, props?: cdk.StackProps) {  
 super(scope, id, props)  
 const domain = "redrobotexample.com"  
 const subdomain = `www.${domain}`  
   
 // fetch route53 zone  
 const zone = route53.HostedZone.fromLookup(this, "zone", {  
 domainName: domain,  
 })  
   
 // this will create a certificate  
 const certificate = new acm.Certificate(this, "certificate", {  
 domainName: domain,  
 subjectAlternativeNames: [domain, subdomain],  
 validation: acm.CertificateValidation.fromDns(zone),  
 })  
   
 // viewer certificate  
 const viewerCertificate = cloudfront.ViewerCertificate.fromAcmCertificate(  
 certificate,  
 {  
 aliases: [domain, subdomain],  
 }  
 )  
   
 // bucket where website dist will reside  
 const bucket = new s3.Bucket(this, "WebsiteBucket", {  
 websiteIndexDocument: "index.html",  
 websiteErrorDocument: "404.html",  
 publicReadAccess: true,  
 blockPublicAccess: {  
 blockPublicAcls: false,  
 blockPublicPolicy: false,  
 ignorePublicAcls: false,  
 restrictPublicBuckets: false,  
 },  
 removalPolicy: cdk.RemovalPolicy.DESTROY,  
 autoDeleteObjects: true,  
 })  
   
 const distro = new cloudfront.CloudFrontWebDistribution(  
 this,  
 "WebsiteCloudfrontDist",  
 {  
 viewerCertificate,  
 originConfigs: [  
 {  
 s3OriginSource: {  
 s3BucketSource: bucket,  
 },  
 behaviors: [  
 {  
 isDefaultBehavior: true,  
 },  
 ],  
 },  
 ],  
 }  
 )  
   
 // s3 construct to deploy the website dist content  
 new s3deploy.BucketDeployment(this, "WebsiteDeploy", {  
 destinationBucket: bucket,  
 sources: [s3deploy.Source.asset("../frontend/dist")],  
 distribution: distro,  
 distributionPaths: ["/\*"],  
 memoryLimit: 512,  
 })  
   
 new route53.ARecord(this, "route53Domain", {  
 zone,  
 recordName: domain,  
 target: route53.RecordTarget.fromAlias(  
 new route53Targets.CloudFrontTarget(distro)  
 ),  
 })  
   
 new route53.ARecord(this, "route53FullUrl", {  
 zone,  
 recordName: "www",  
 target: route53.RecordTarget.fromAlias(  
 new route53Targets.CloudFrontTarget(distro)  
 ),  
 })  
 }  
}

3. now deploy the stack by typing cdk deploy

Once deployment open the browser and go to the domain and view the page. That is it!

And to destroy all the resources just run cdk destroy And enter **y** to confirm you want to delete. And that is it! all resource will been destroyed according to the dependency tree CDK keeps track of.

As you can see, this method requires more knowledge but triumphs the manual method for all cases.

……………………………………………………………………………………………………………………………………………………………