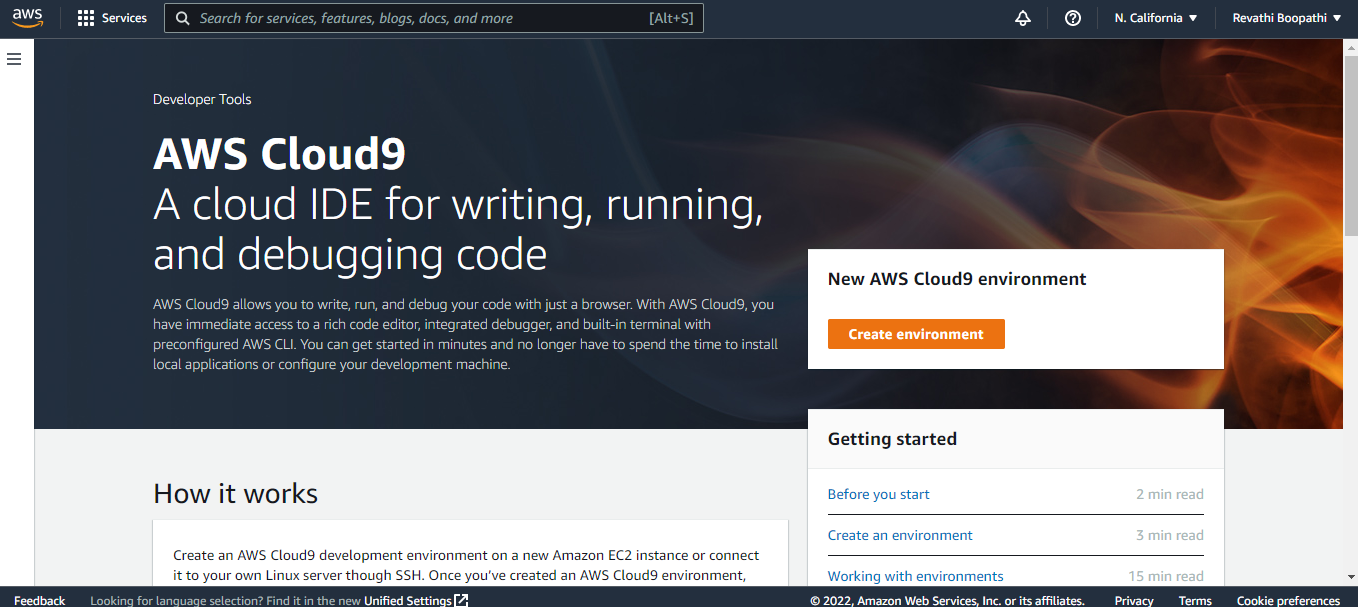
**Build a Multi-Region Serverless Application for Resilience and High Availability**

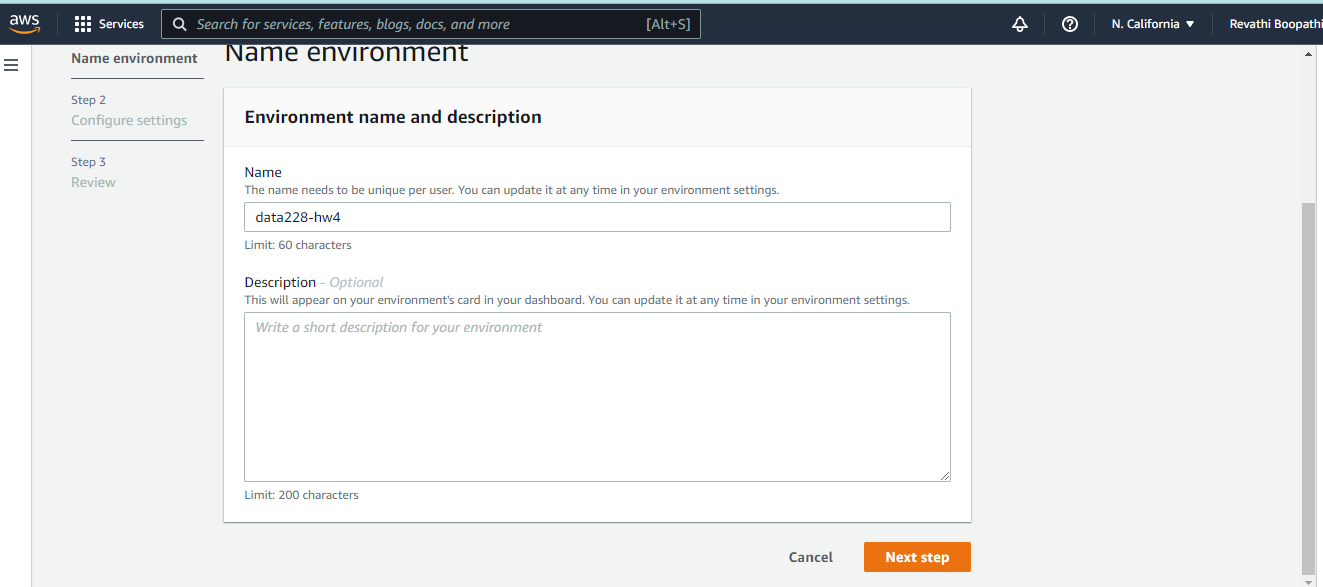
# **1. Create AWS Cloud9 Environment**

## **Step 1:** Login to AWS Console, select *eu-west-1 Region* and spin up a AWS Cloud9 environment. [https://us-west-1.console.aws.amazon.com/cloud9/home/product?region=us-west-1#](https://us-west-1.console.aws.amazon.com/cloud9/home/product?region=us-west-1)

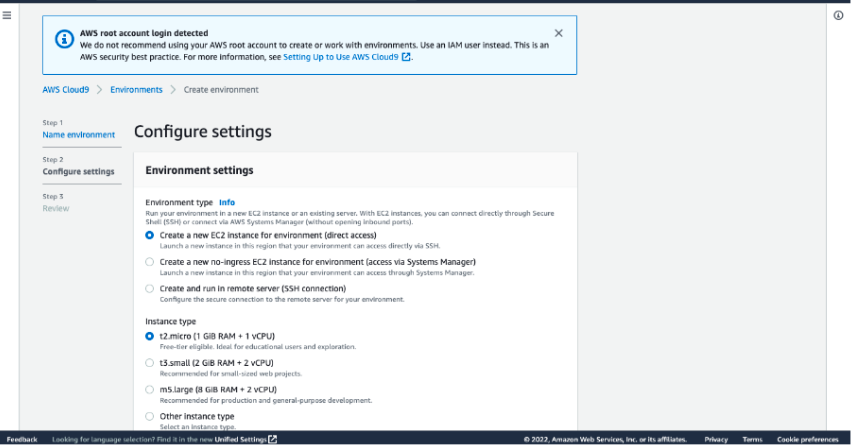
Click on ‘Create Environment’ to create a new AWS Cloud9 environment.

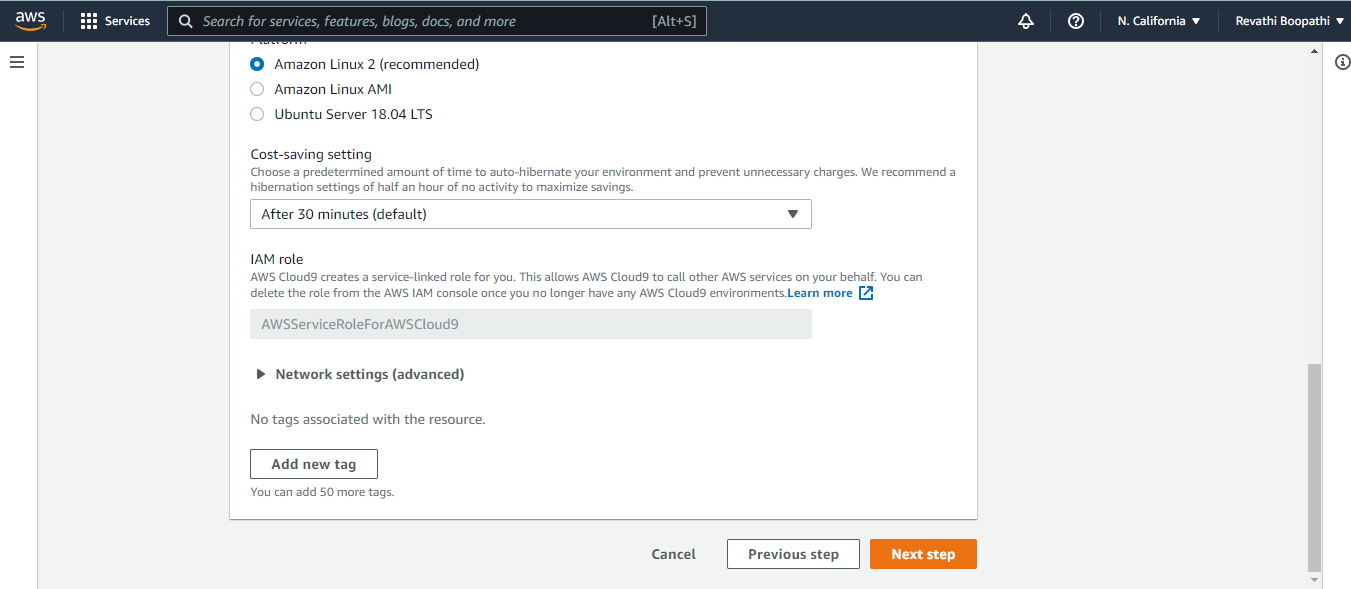


## **Step 2:** Name your AWS Cloud9 Environment



## **Step 3:** Select your EC2 instance type. We can use t2.micro for this lab. To save cost, AWS Cloud9 will spin down the EC2 when you are not using it.

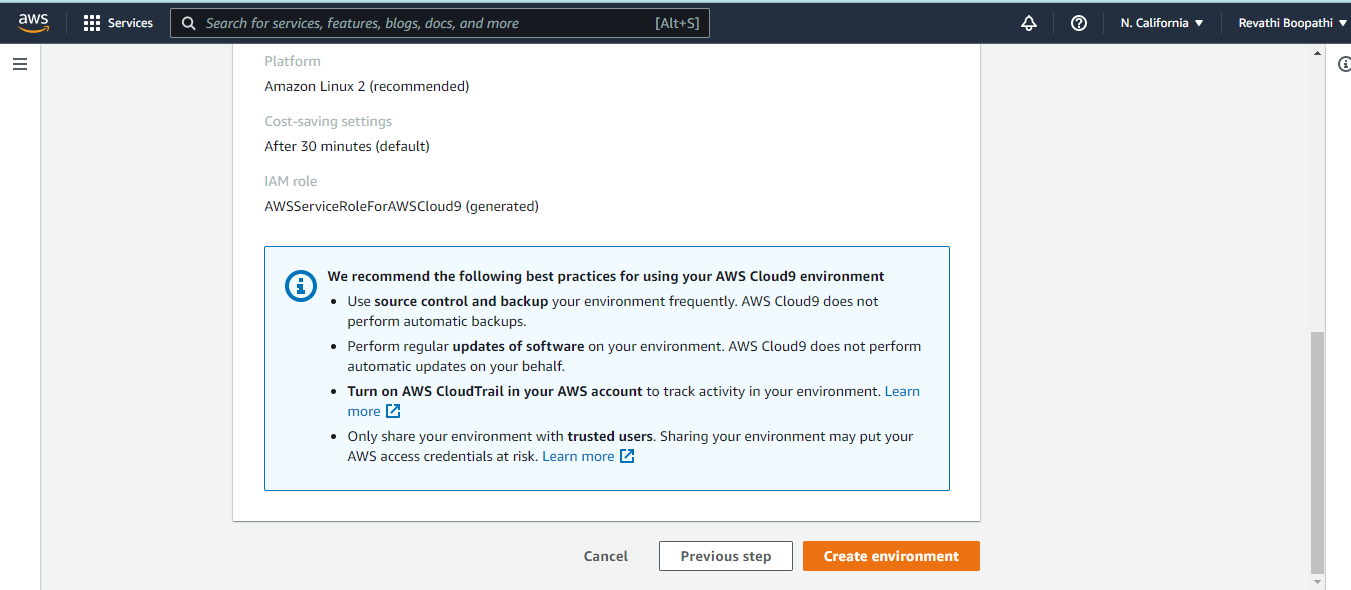




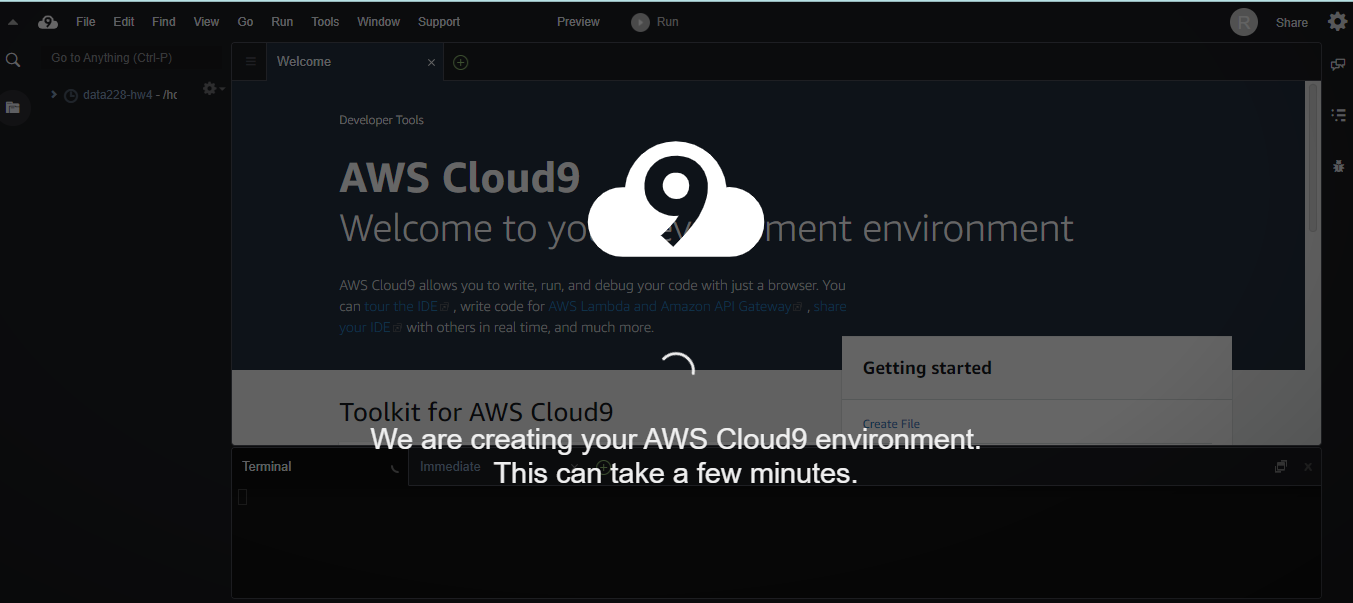
## 

## **Step 4:** Review the filled details and click "Create environment".

AWS Cloud9 will start! It would typically take 30-60s to create your AWS Cloud9 environment.



We can see that the AWS Cloud9 environment is getting created.



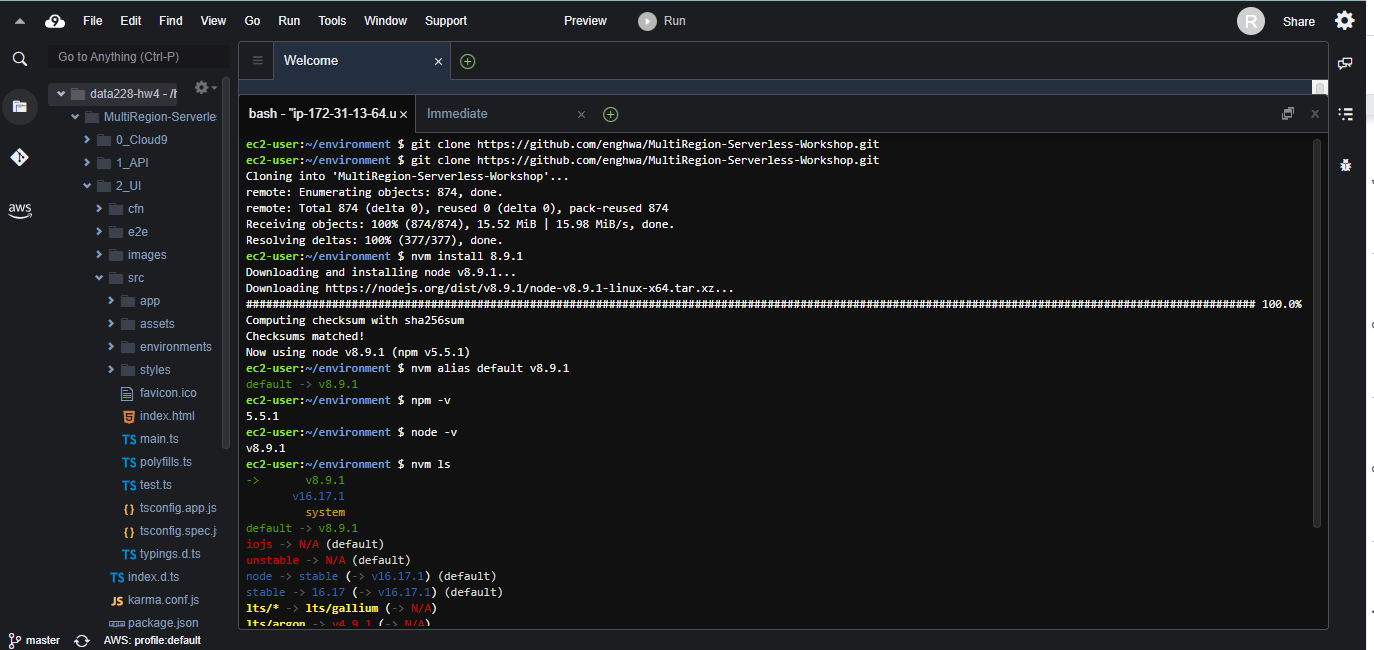
## **Step 5:** Install node and npm

Once AWS Cloud9 is up and running, execute below 2 commands in the command shell of Cloud9. This is required for the AngularJS front end.

*Command 1:* nvm install 8.9.1

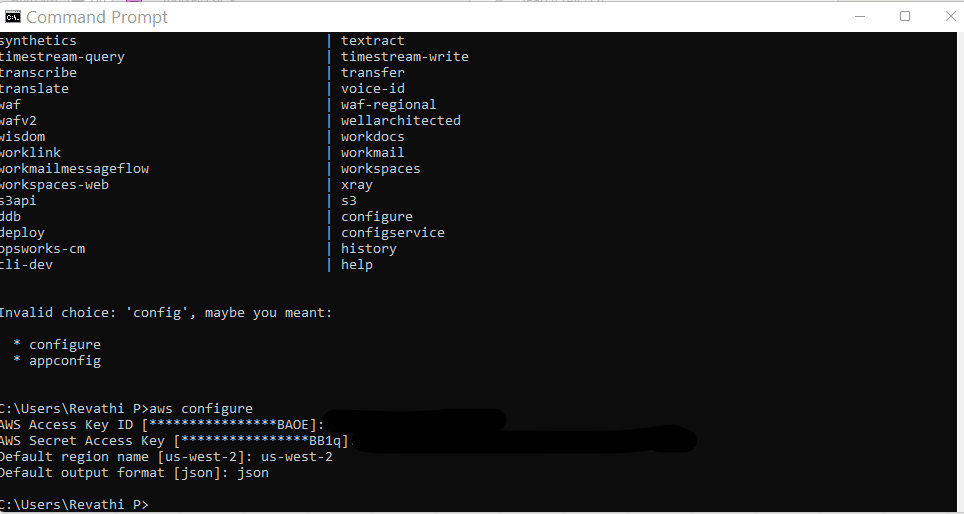
*Command 2:* nvm alias default v8.9.1

Verify that installation is done right for node and npm: npm -v, node -v, nvm ls



# **2. Build an API layer**

We have set up a CLI command prompt locally. However, used AWS Cloud9 for command execution



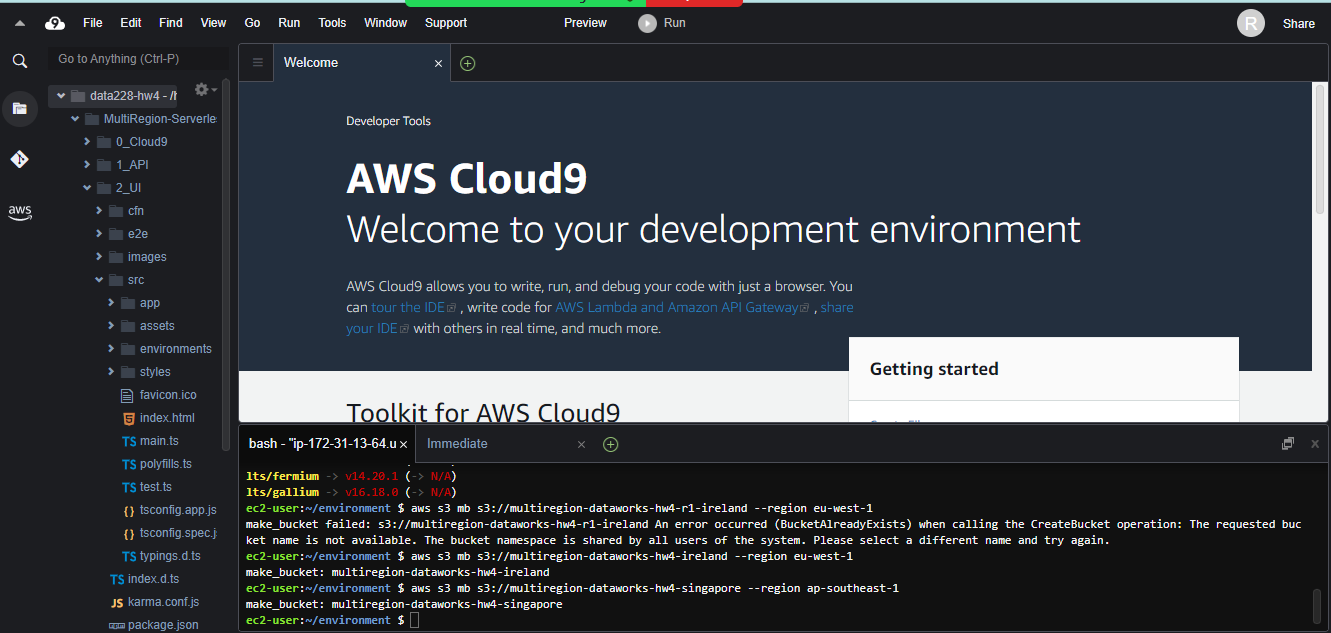
## 

## **Step 1:** Create an S3 bucket to store the app code

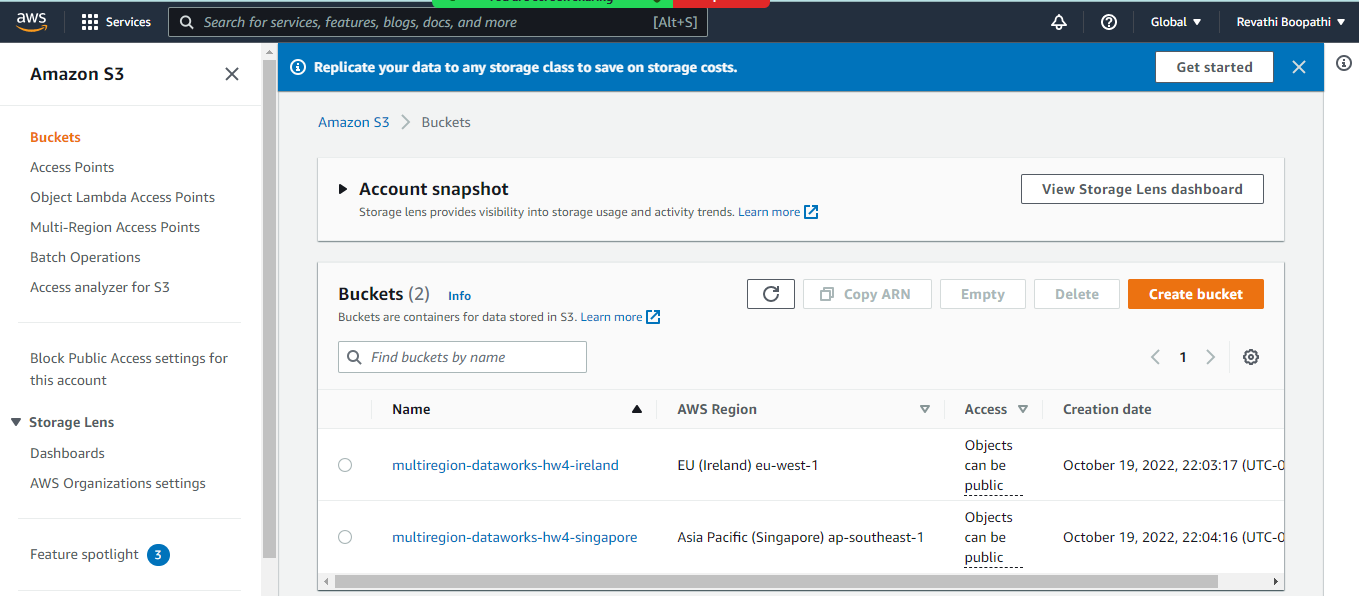
Create a bucket for each region( Ireland and Singapore ) using the CLI. Make sure S3 bucket name is unique for each of the region.

**S3 Ireland bucket name:** aws s3 mb s3://multiregion-dataworks-hw4-ireland --region eu-west-1

**S3 Singapore bucket name:**aws s3 mb s3://multiregion-dataworks-hw4-singapore --region ap-southeast-1



2 buckets are created in S3



## **Step 2:** Package up the API code and push to S3

Create 2 serverless application model templates in each of the Ireland and Singapore regions using cloud9.

Set the environment to current directory using the below:

*cd /home/ec2-user/environment/MultiRegion-Serverless-Workshop/1\_API*

Cloud formation package for ireland

aws cloudformation package \

--region eu-west-1 \

--template-file wild-rydes-api.yaml \

--output-template-file wild-rydes-api-output.yaml \

--s3-bucket multiregion-dataworks-hw4-ireland

Cloud formation package for Singapore:

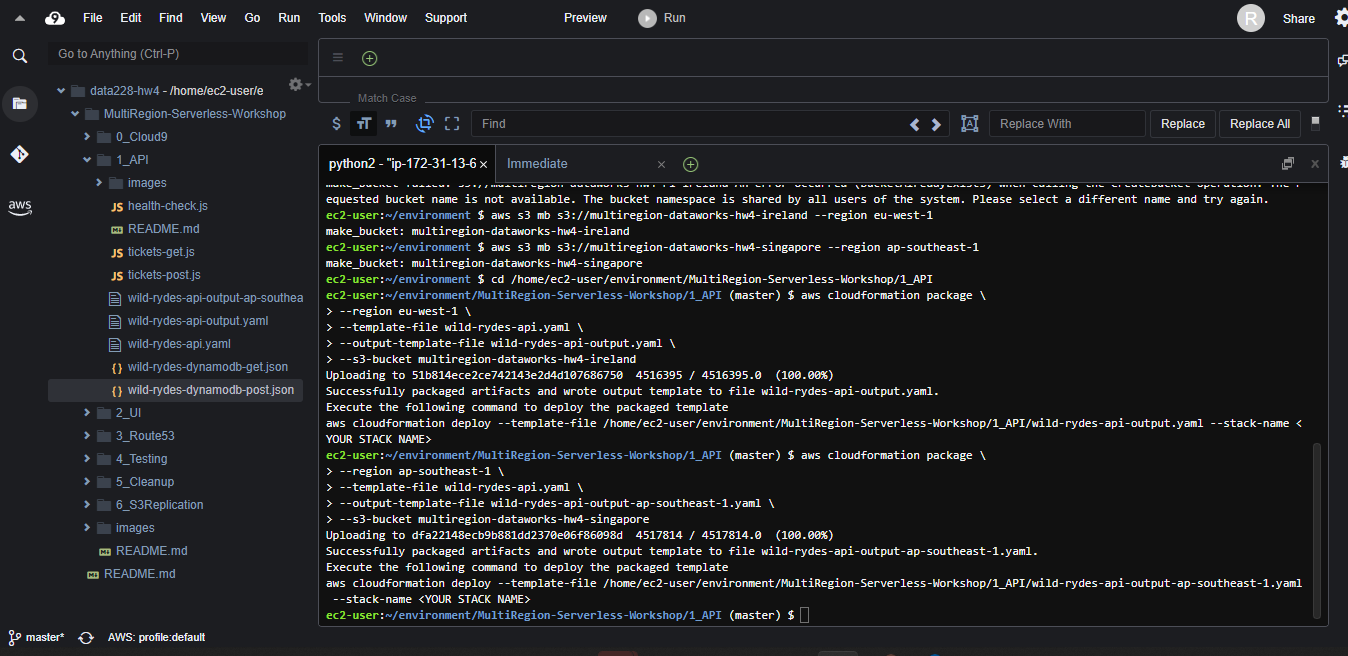
aws cloudformation package \

--region ap-southeast-1 \

--template-file wild-rydes-api.yaml \

--output-template-file wild-rydes-api-output-ap-southeast-1.yaml \

--s3-bucket multiregion-dataworks-hw4-singapore



There will be 2 cloudformation templates generated: wild-rydes-api-output-ap-southeast-1.yaml for Singapore and wild-rydes-api-output.yaml which you can see created in the left side of the IDE.

## **Step 3: Deploy a stack of resources**

Use ‘wild-rydes-api-output.yaml’ for Ireland and ‘wild-rydes-api-output-ap-southeast-1.yaml’ for Singapore to create resources in AWS.

Use the below commands for the same

Ireland:

aws cloudformation deploy \

--region eu-west-1 \

--template-file wild-rydes-api-output.yaml \

--stack-name wild-rydes-api \

--capabilities CAPABILITY\_IAM

Singapore:

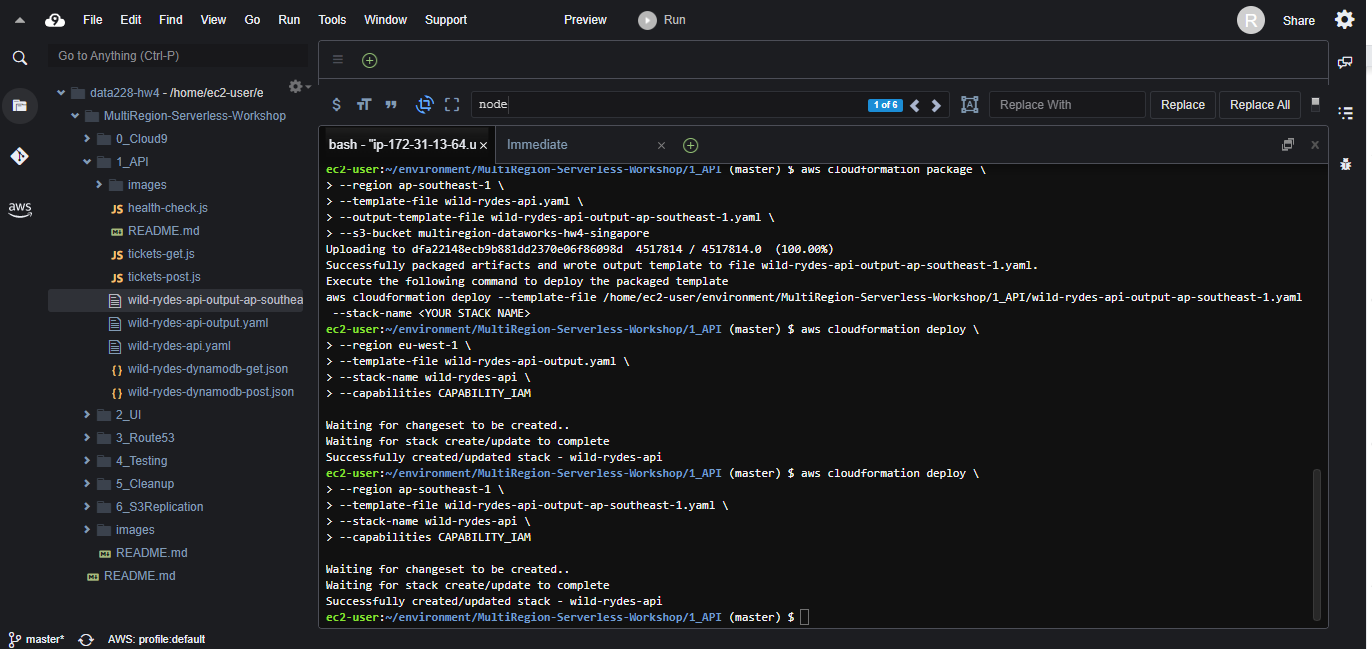
aws cloudformation deploy \

--region ap-southeast-1 \

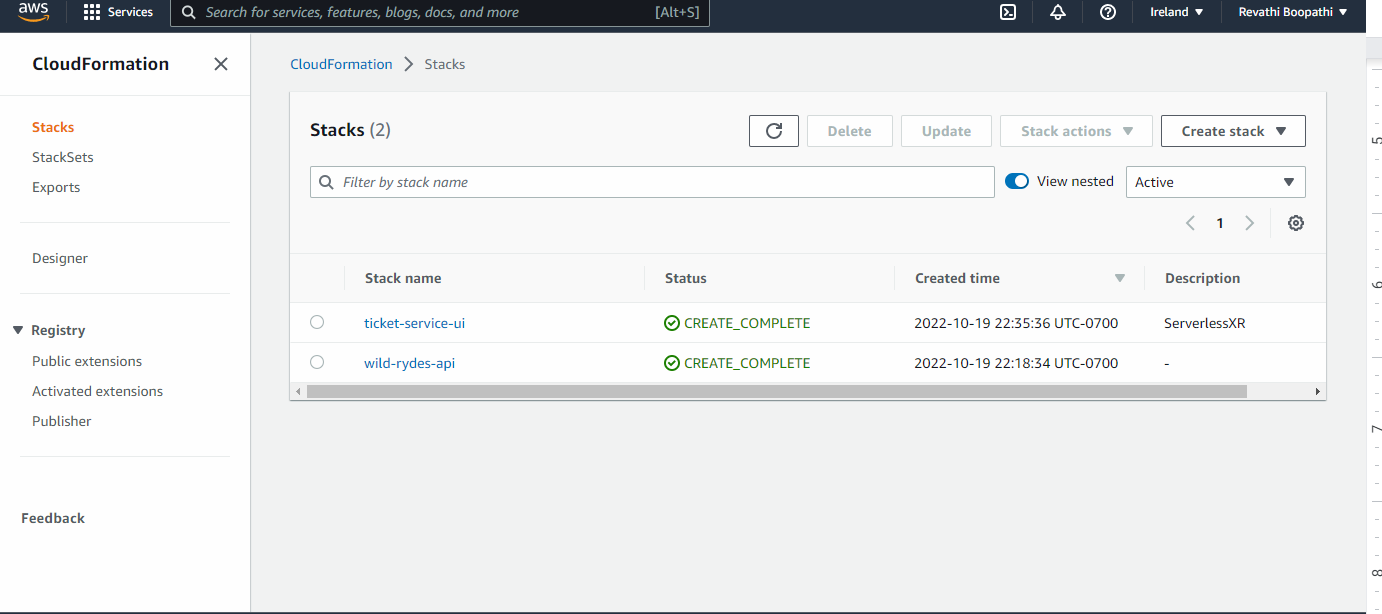
--template-file wild-rydes-api-output-ap-southeast-1.yaml \

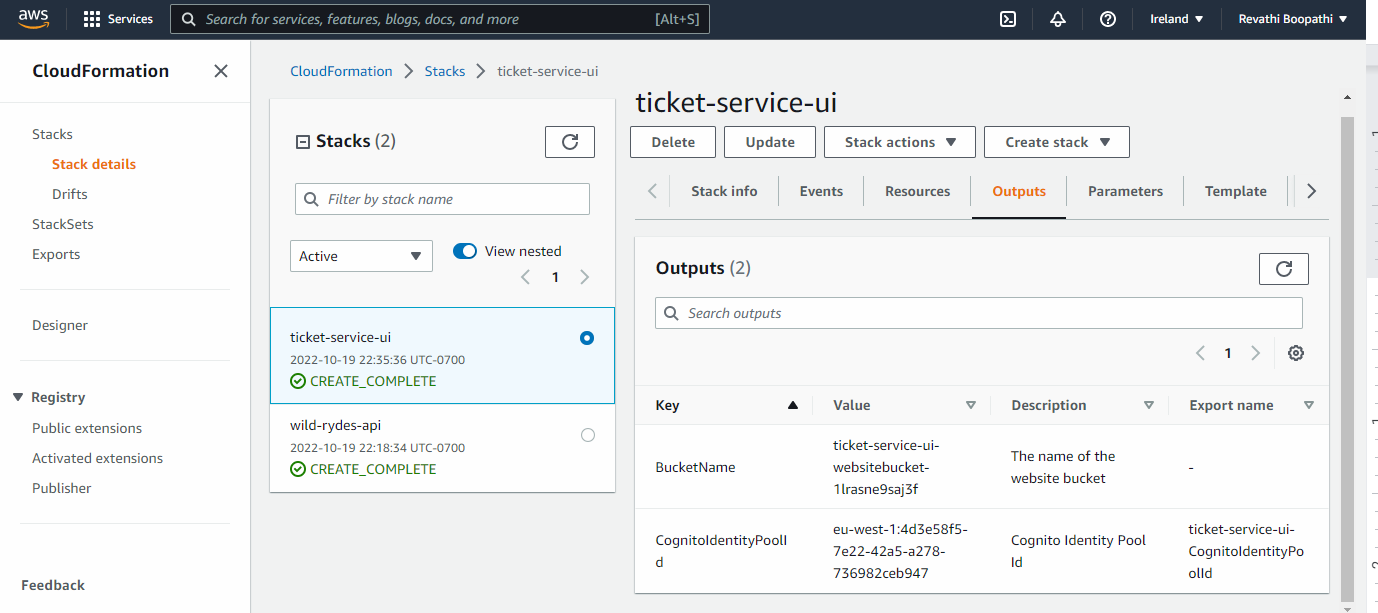
--stack-name wild-rydes-api \

--capabilities CAPABILITY\_IAM



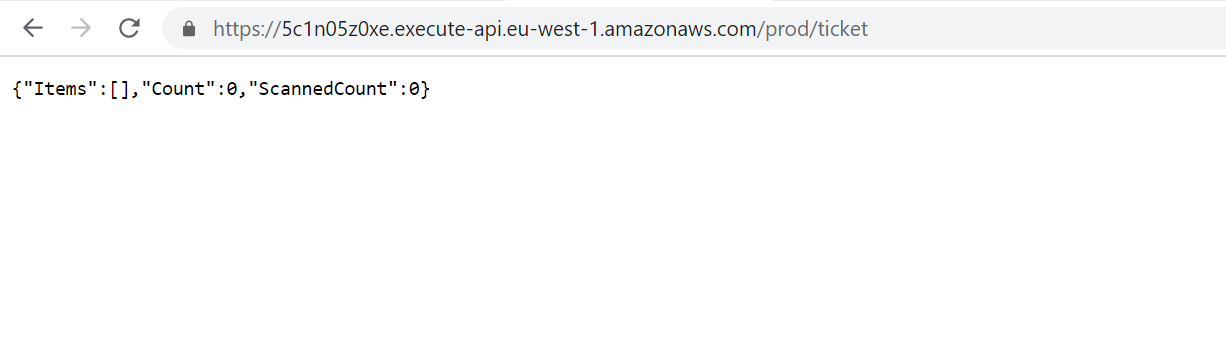
Resources are created in CloudFormation.



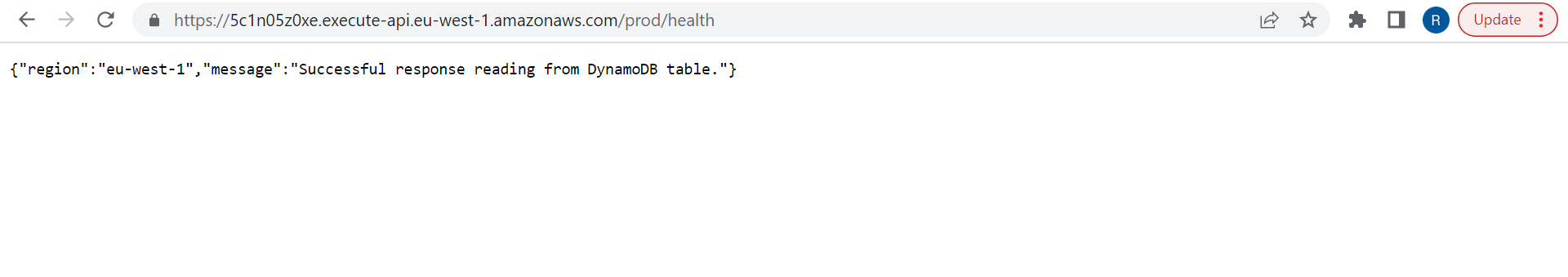


API URL for Ireland: <https://5c1n05z0xe.execute-api.eu-west-1.amazonaws.com/prod/>

[https://5c1n05z0xe.execute-api.eu-west-1.amazonaws.com/prod](https://5c1n05z0xe.execute-api.eu-west-1.amazonaws.com/prod/)/ticket



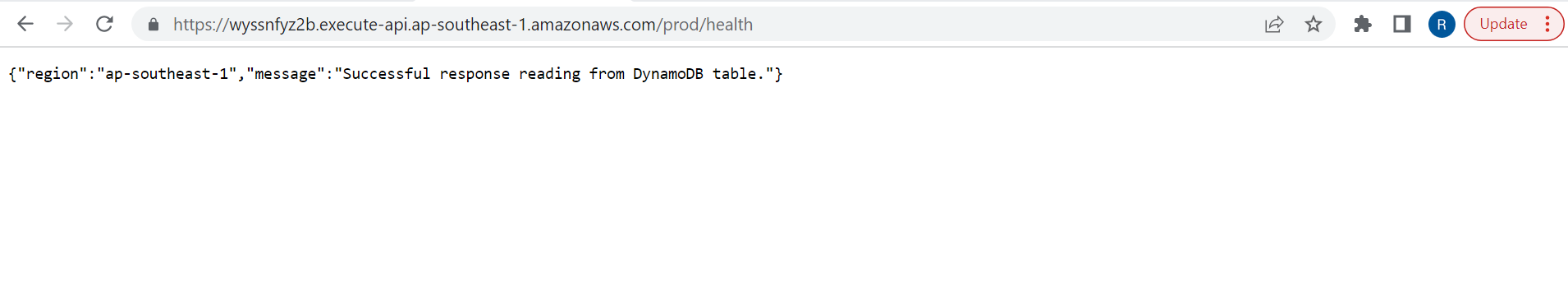
<https://5c1n05z0xe.execute-api.eu-west-1.amazonaws.com/prod/>health



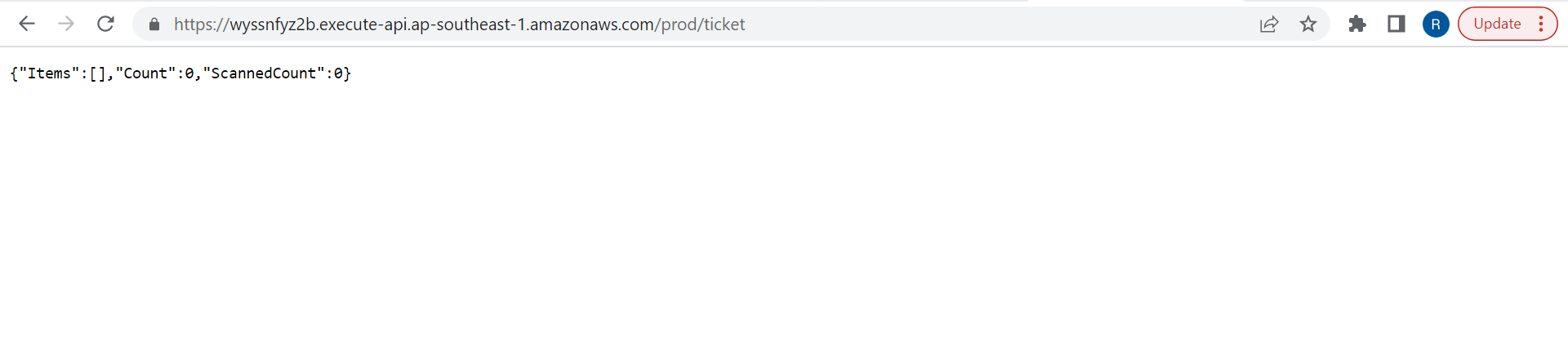
API URL for Singapore:

[https://wyssnfyz2b.execute-api.ap-southeast-1.amazonaws.com/prod/](https://wyssnfyz2b.execute-api.ap-southeast-1.amazonaws.com/prod/ticket)

<https://wyssnfyz2b.execute-api.ap-southeast-1.amazonaws.com/prod/health>



<https://wyssnfyz2b.execute-api.ap-southeast-1.amazonaws.com/prod/ticket>



**Step 4:** Enable DynamoDB Global Table using CLI

First create the dynamo DB ‘SXRTickets’ table under AWS DynamoDB>> Create Table

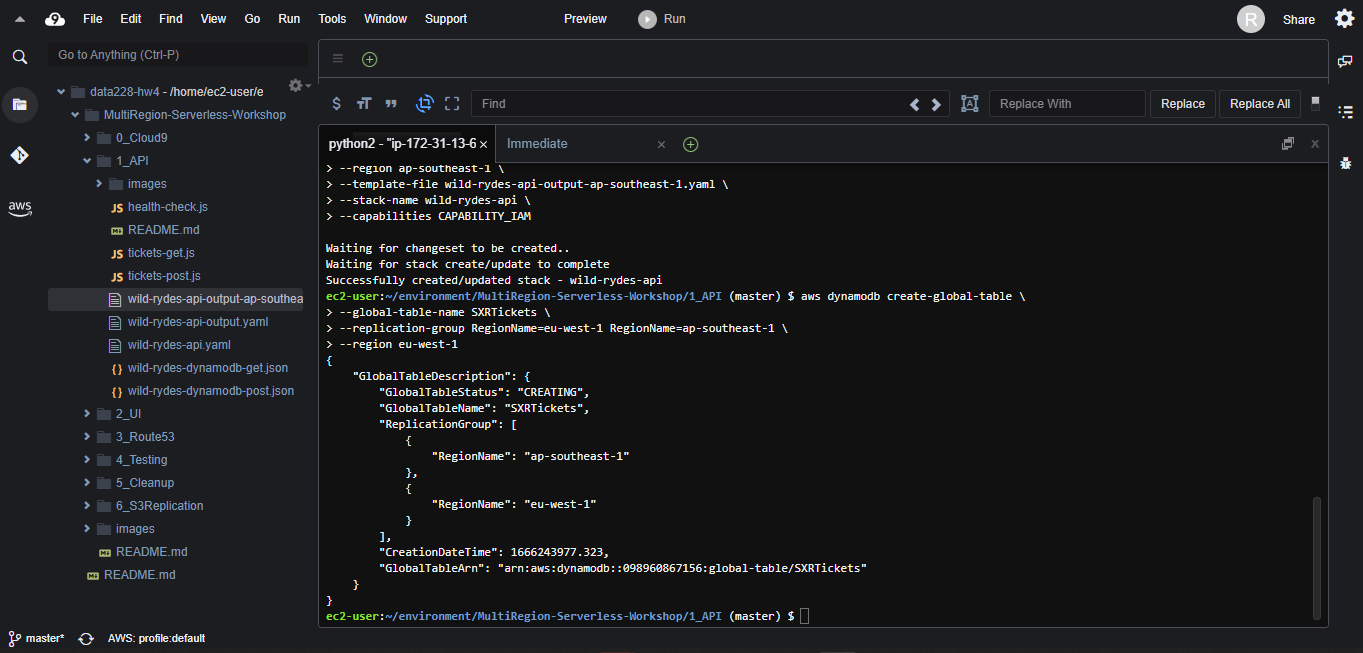
Enable DynamoDB Global Table between Ireland and Singapore using the below command

aws dynamodb create-global-table \

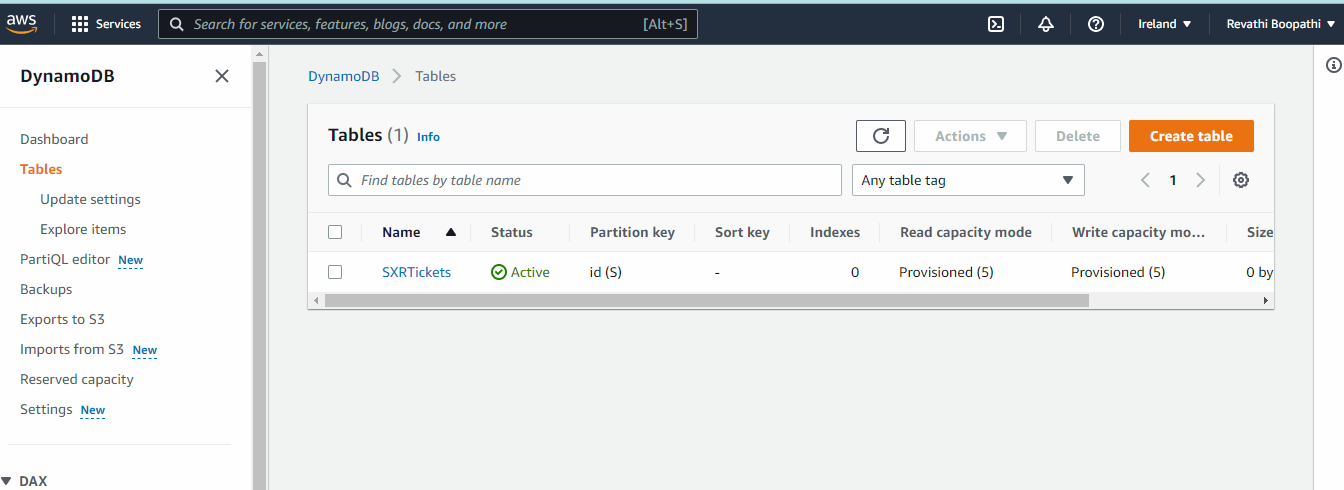
--global-table-name SXRTickets \

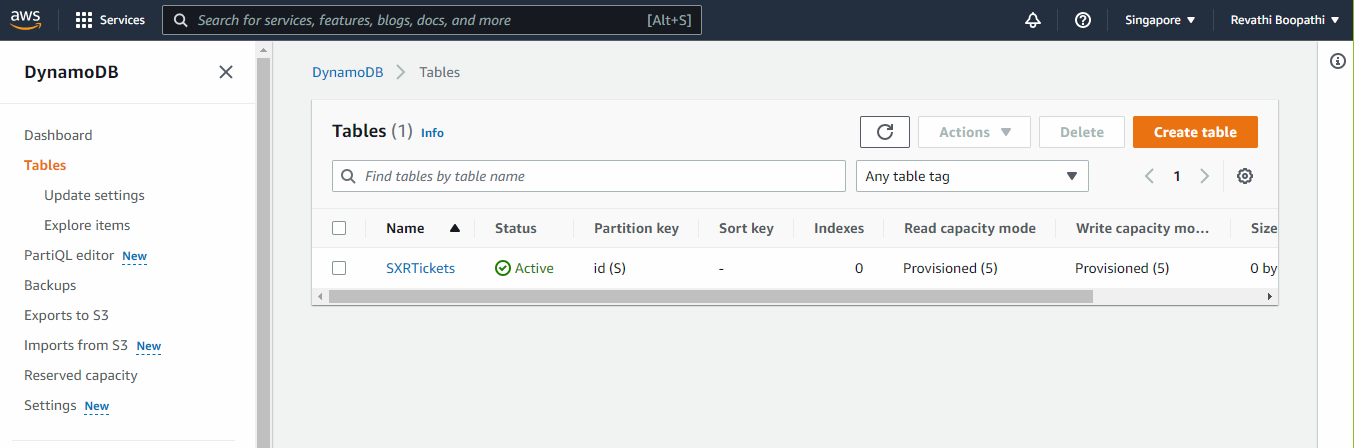
--replication-group RegionName=eu-west-1 RegionName=ap-southeast-1 \

--region eu-west-1



‘SXRTickets’ table is replicated in both Ireland and Singapore regions successfully.



In the above steps we have configured the backed components for ticketing application.

# 3. Build a UI layer

## **Step 1:** Create the AWS Cognito Identity Pool, S3 bucket and Cloudfront distribution

Now that we have a working API, let's deploy a UI that can expose this functionality to our users. Note that we will only deploy this UI in our *Primary* region (Ireland). We don't attempt to address failing over a full web application in this workshop. Our failover efforts are focused on the backend API components deployed in the first module.

Deploy the below template in the primary region (Ireland) using the aws cloudformation deploy CLI command a below.

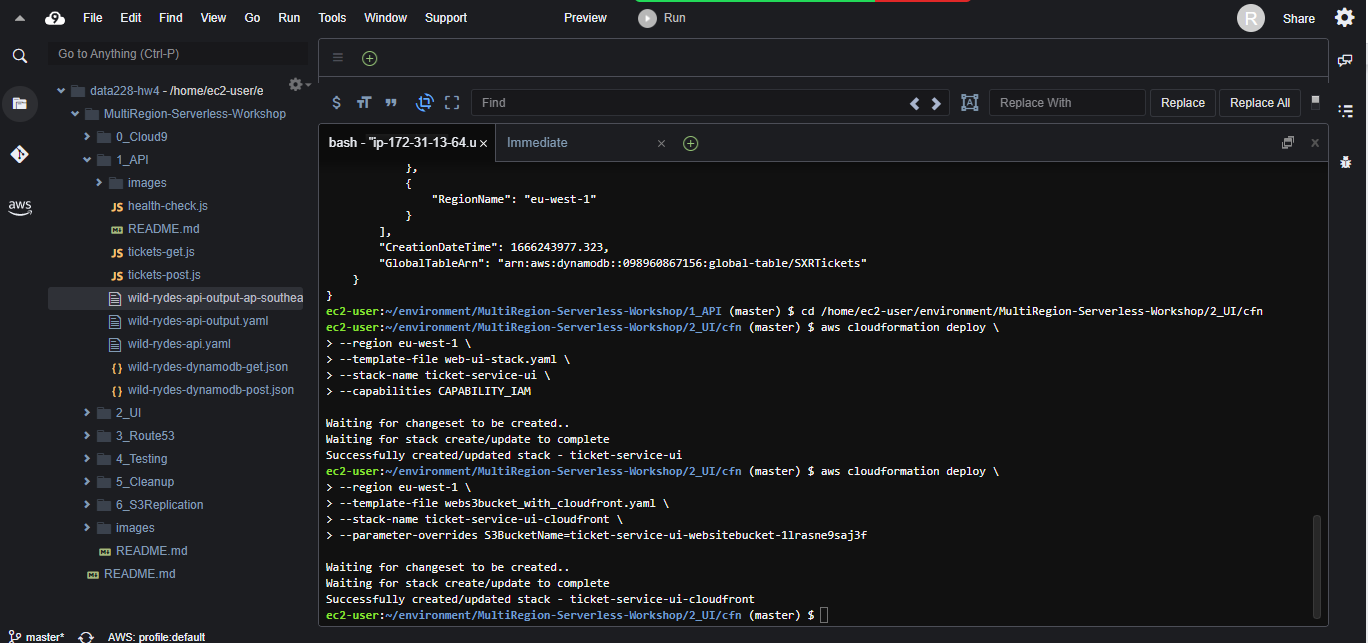
aws cloudformation deploy \

--region eu-west-1 \

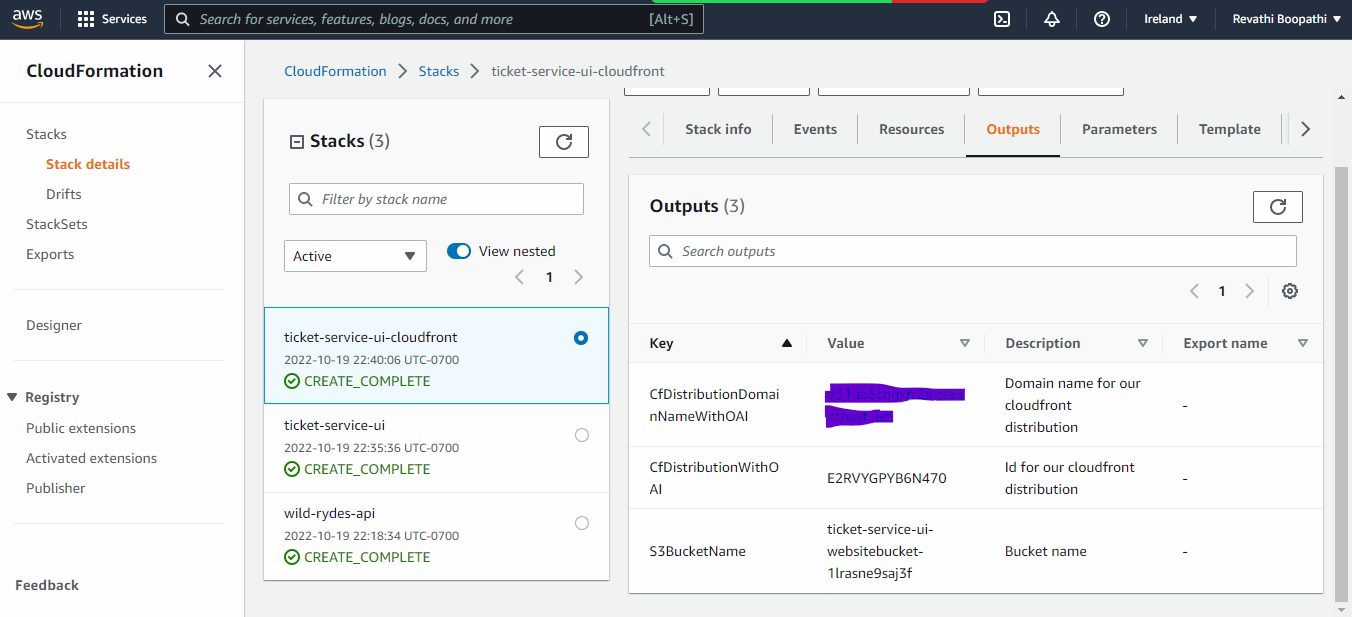
--template-file webs3bucket\_with\_cloudfront.yaml \

--stack-name ticket-service-ui-cloudfront \

--parameter-overrides S3BucketName=ticket-service-ui-websitebucket-1lrasne9saj3f

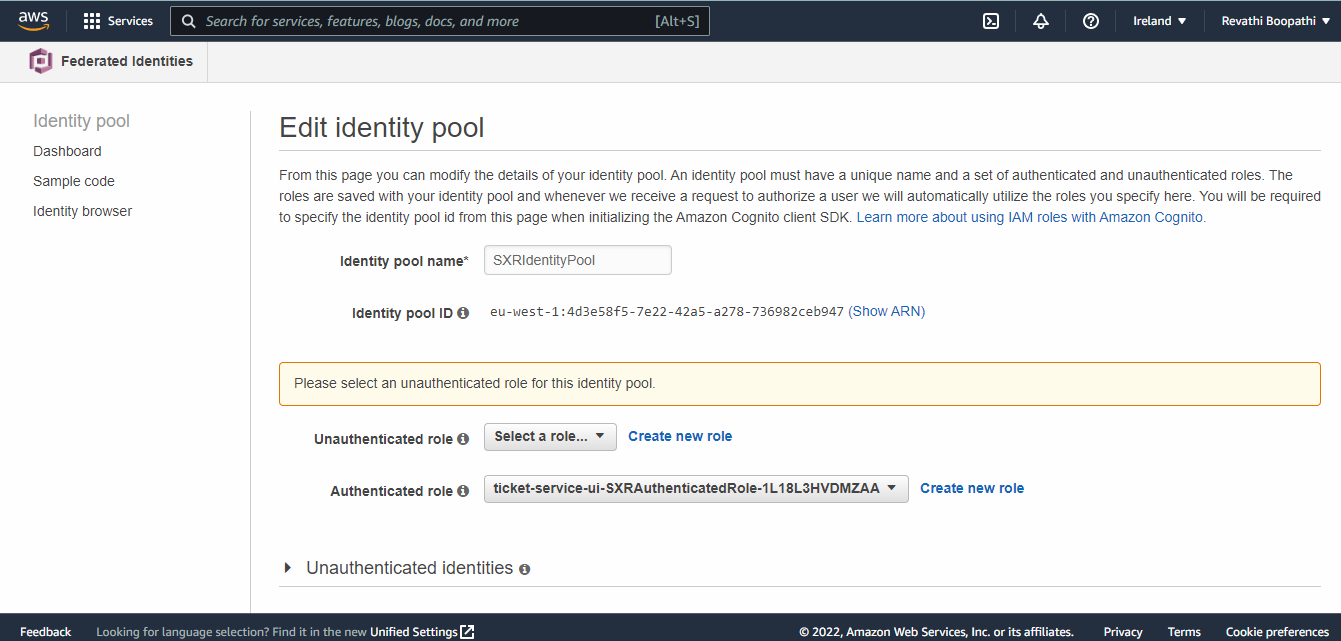


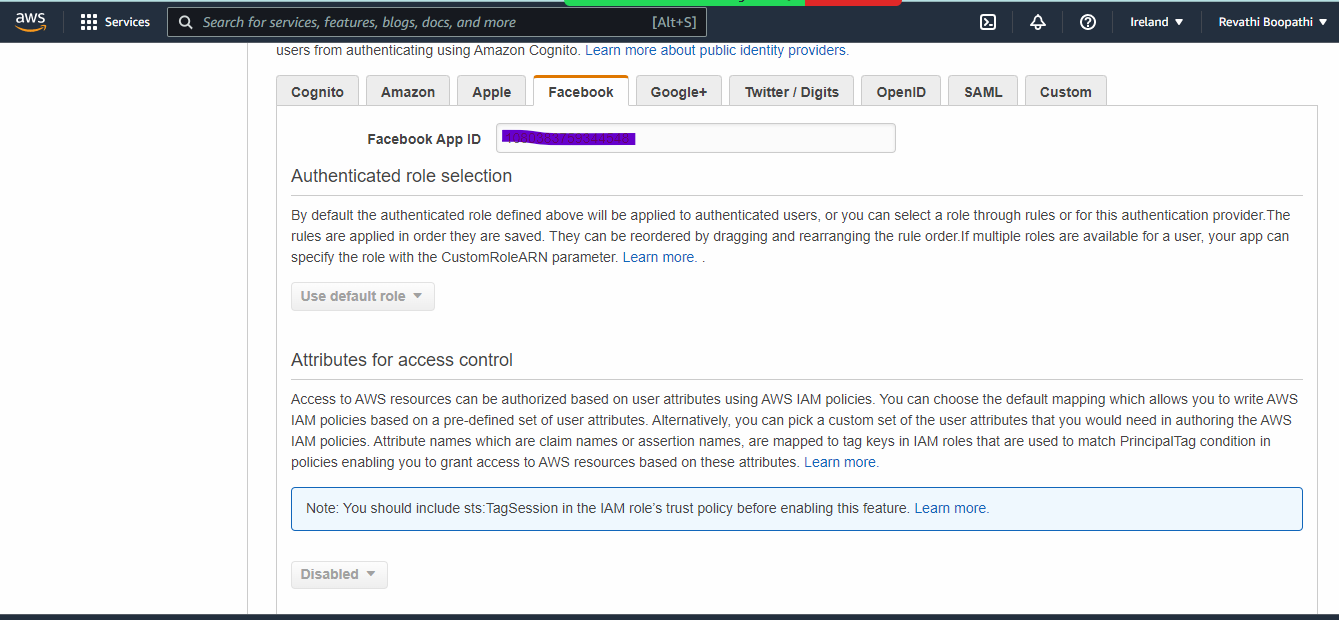
Confirm that this was created successfully and see the resource in the AWS Console. Navigate to the CloudFormation service and take a look at the Outputs tab



## **Step 2:** Configure Federated Identities with Cognito

Next, we must configure the Cognito Identity Pool to use Facebook as our identity provider. To do this, open up the AWS Console in your browser and navigate to Cognito from the menu. Select the below mentioned configurations.



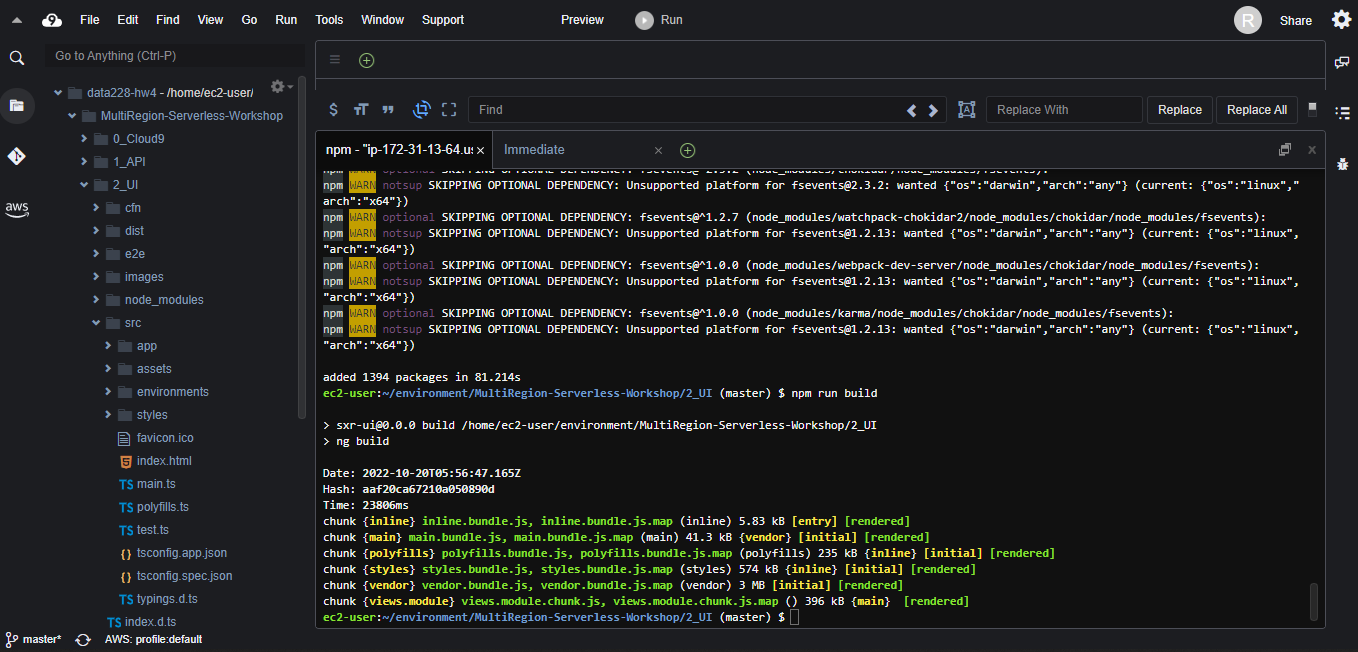


Our application will need to know the location of the API in order to push and pull data from it. In addition, the application will need to know our Facebook App ID and Cognito Identity Pool ID so it can authenticate our users.

## **Step 3: Configure and build the application code**

In your AWS Cloud9 shell, build your AngularJS project:

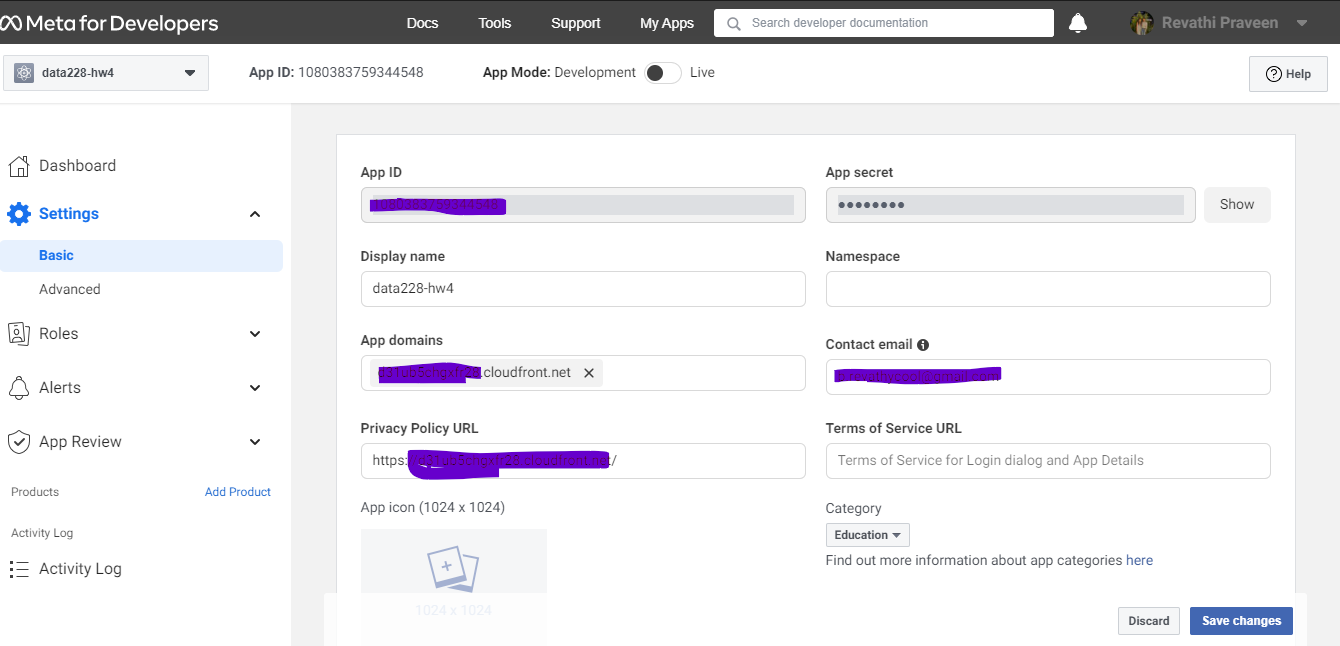
* go to the root folder of the ui project, cd /home/ec2-user/environment/MultiRegion-Serverless-Workshop/2\_UI
* run npm install to install project dependencies
* build your app with by running npm run build



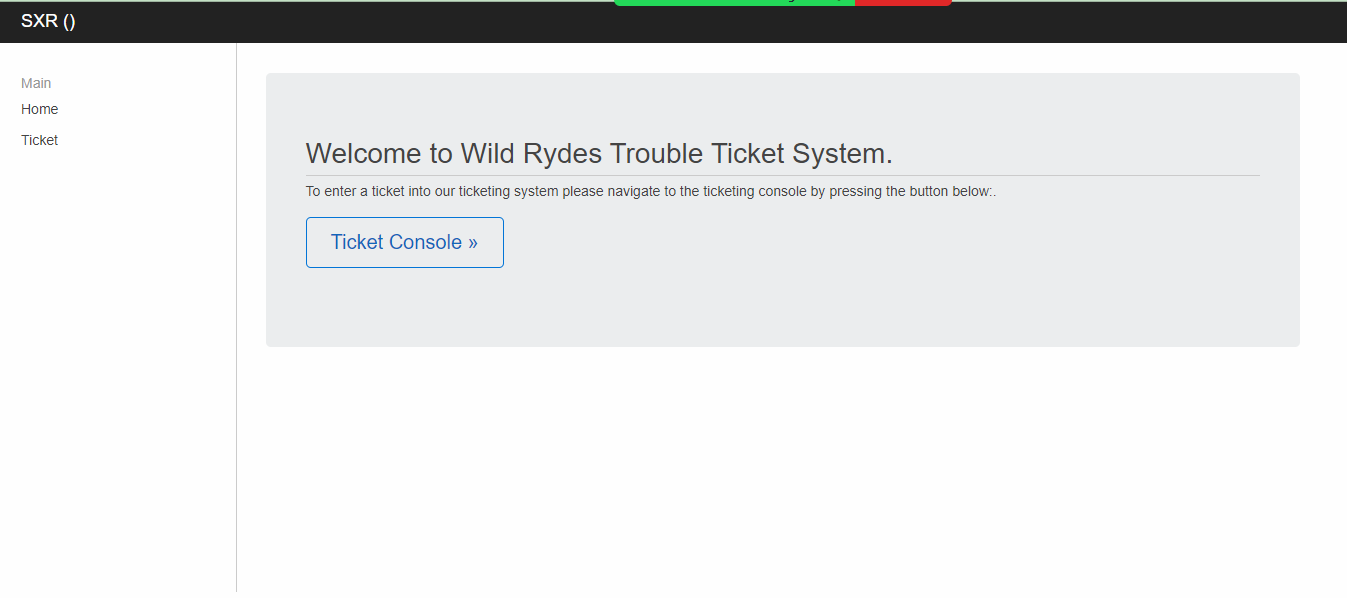
## **Step 4: Upload the application**

Now we'll set up our login with Facebook capability, this allows any Facebook users to login to our application.

Go into your Facebook Developer account and create a new application. Click on Add a New App in the upper right corner.



# 



Next, you'll need to upload the UI to the S3 website bucket specified in step 1. You can do this with:

aws s3 sync --delete dist/ s3://[bucket\_name]

# **4. Configure Active-Active Route53**

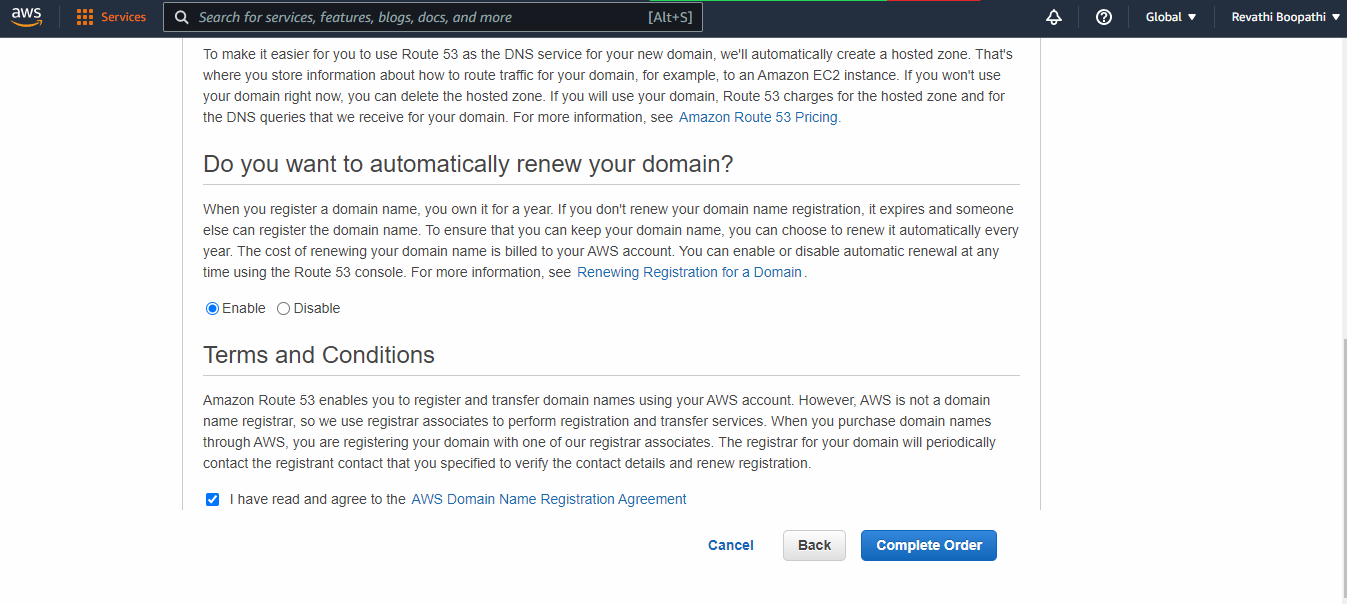
## **Step 1:** Configure Route53 Domain

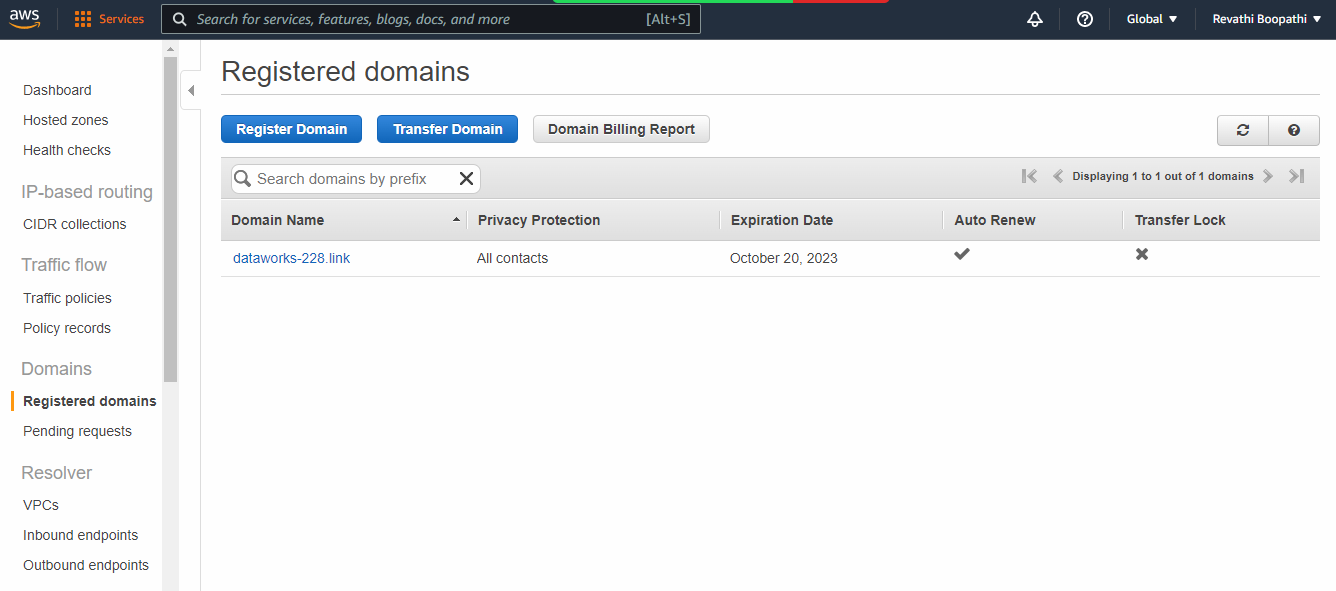
### This step will let you purchase a new Domain name from Route 53. Take note that the AWS Credit code does not cover the cost of 1 year of the domain.

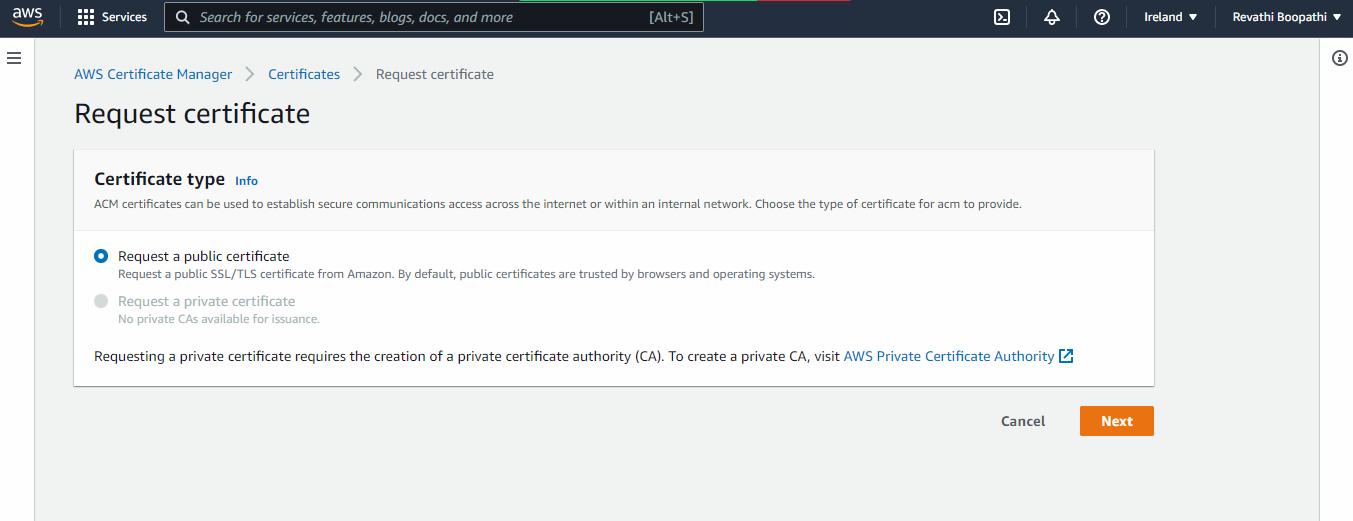
## **Step 2:** Configure Route53 Domain

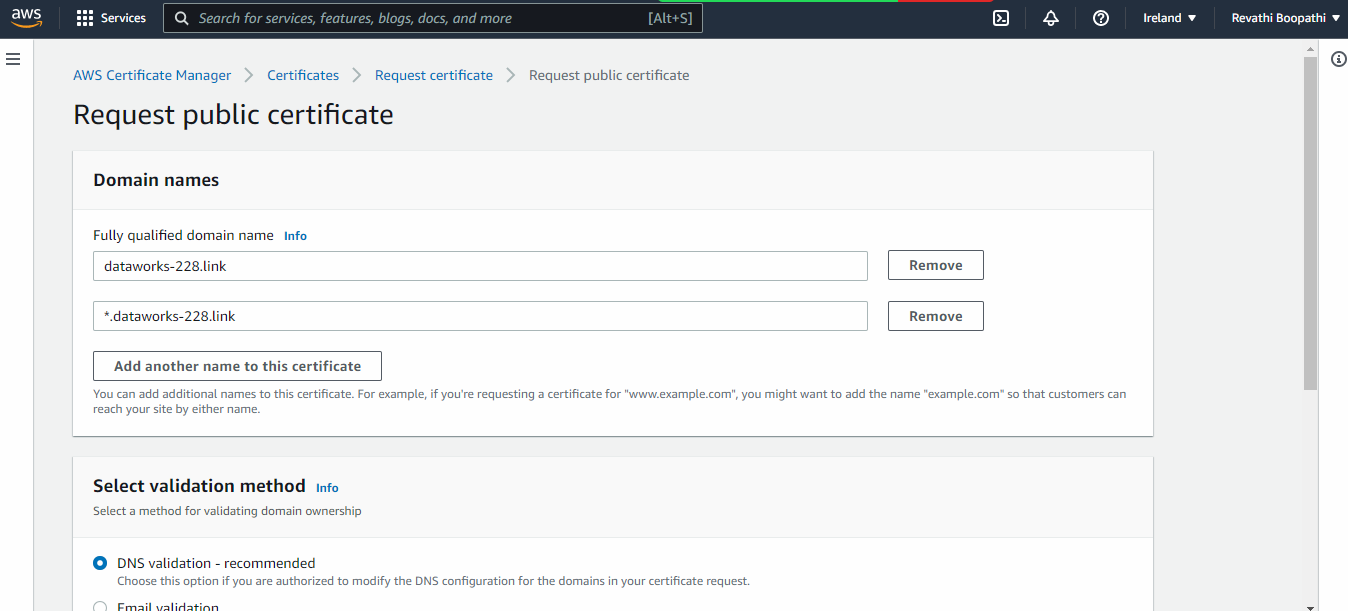
### Configure a certificate in Certificate Manager in each region

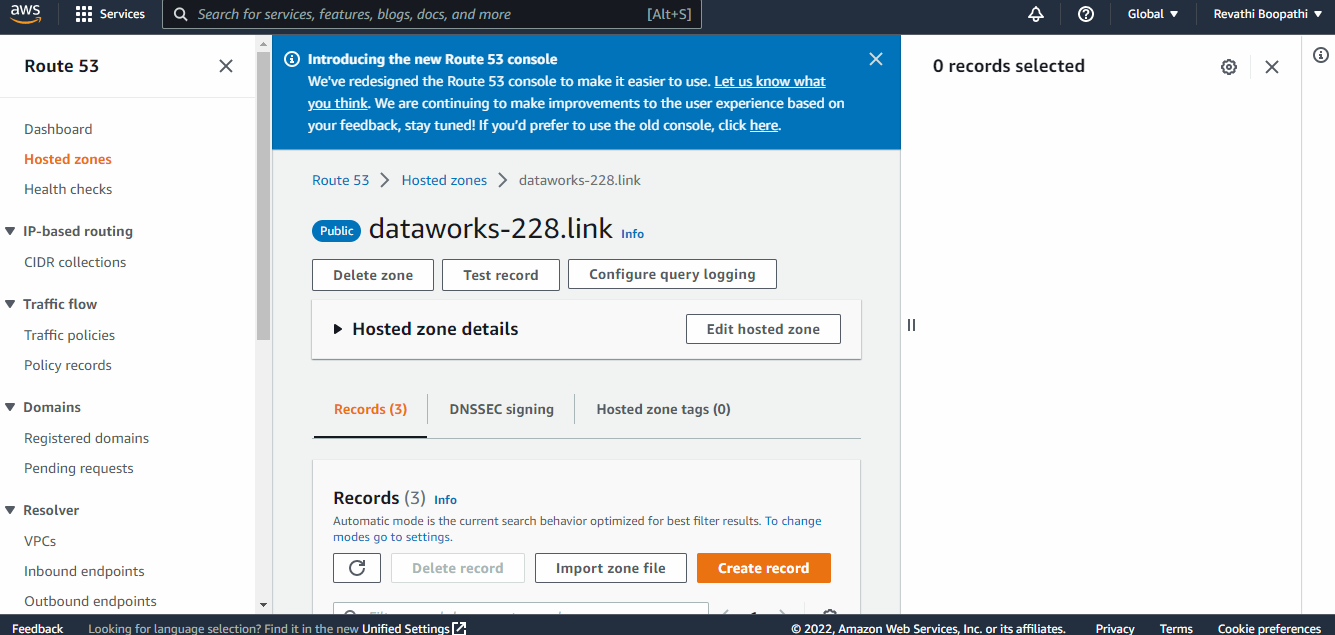
Navigate over to the *Certificate Manager* service and request a new certificate for your domain. You will specify the domain name you just created (or repurposed). Make sure to request a wildcard certificate. You will have to approve the certificate request by creating a special DNS record - follow those directions if you choose to purschase your own domain via Route 53.

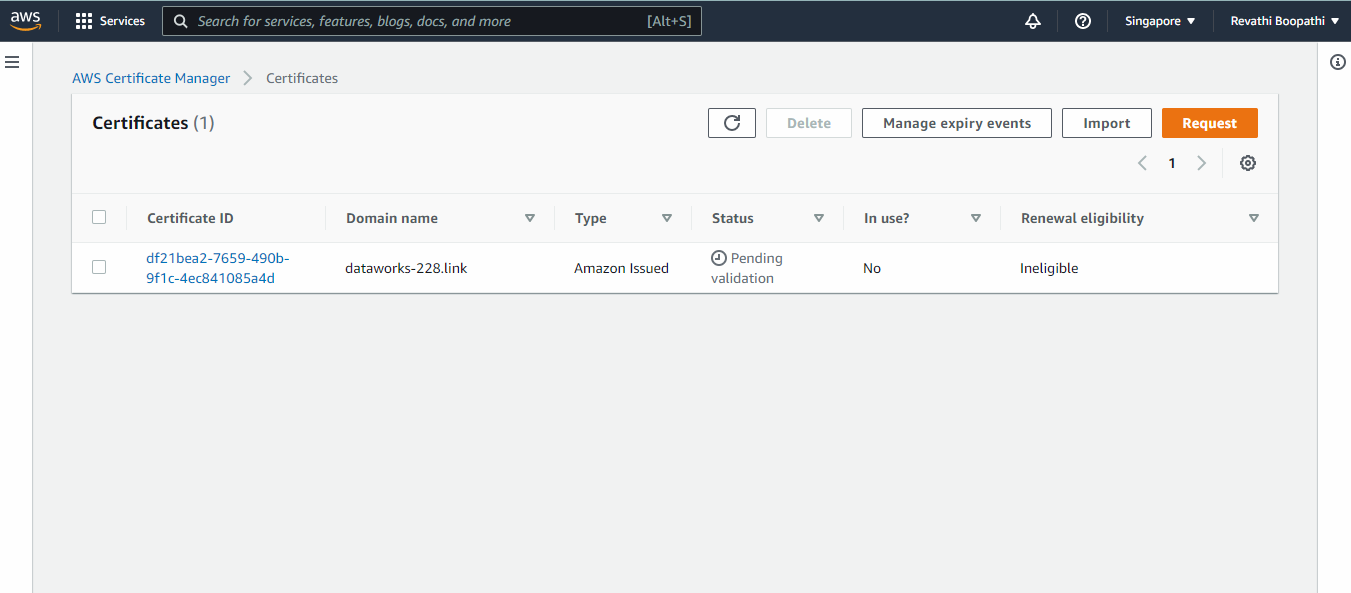


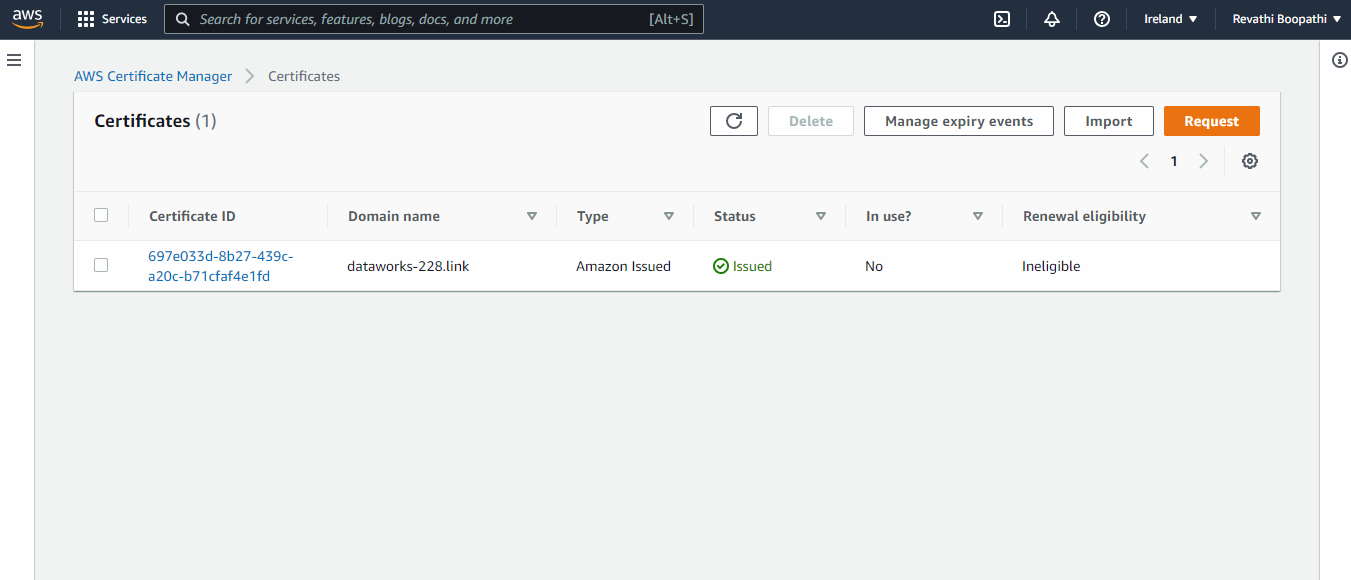












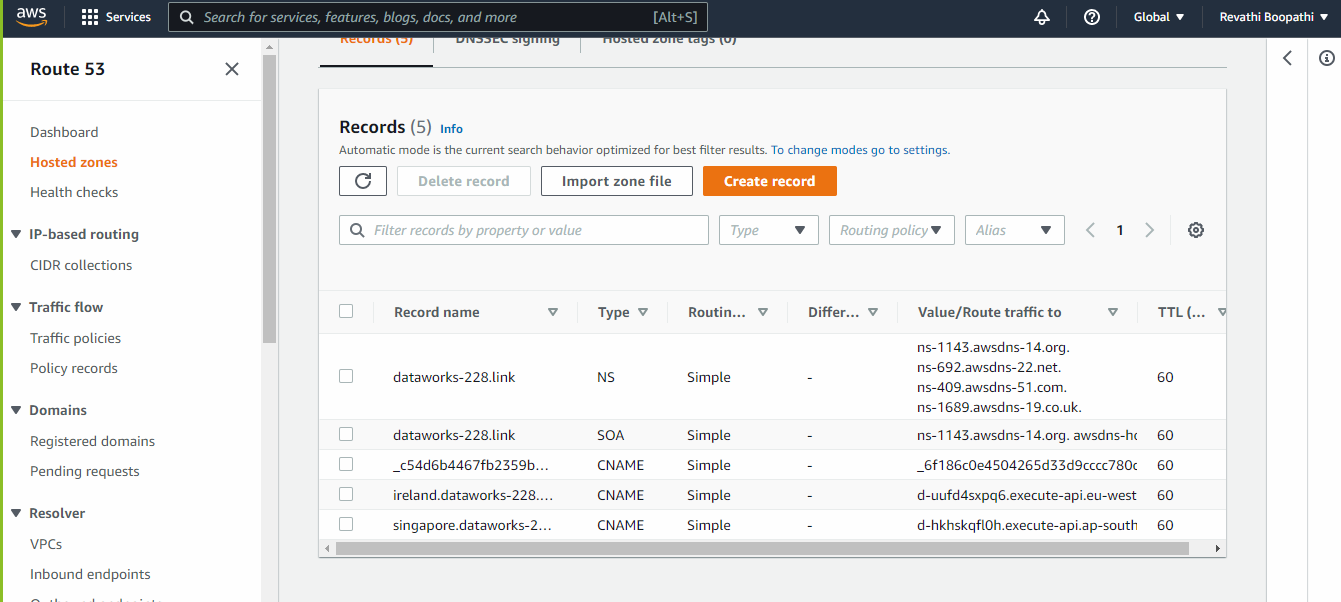


## **Step 3:** Configure Route53 Domain

Head over to the Route53 service and select Hosted zones. Choose your domain name from the list and you should see a couple of records already configured for nameservers.

Select Create Record Set and create a new CNAME record for ireland. pointing to the Target Domain Name for your corresponding API Gateway Custom Domain from the previous step. You can set the TTL to 1m (60 seconds) for the purpose of this workshop. We recommend setting ALL DNS entries to 1m (60 seconds) as the TTL.

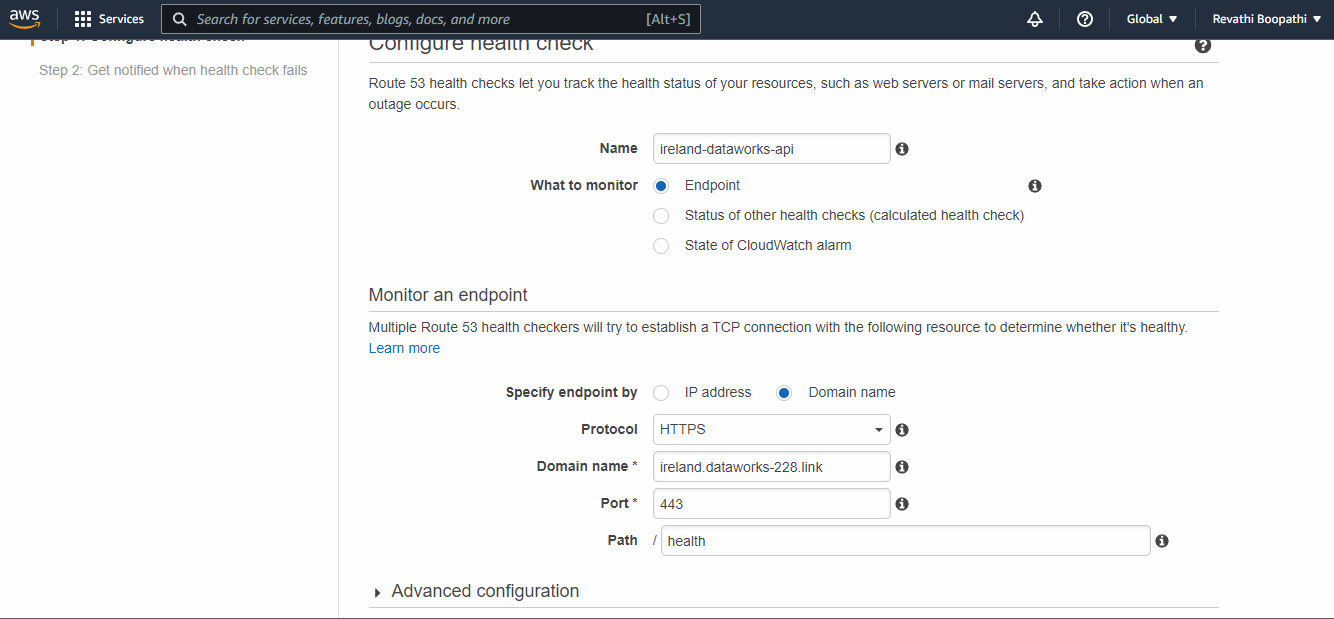
# 



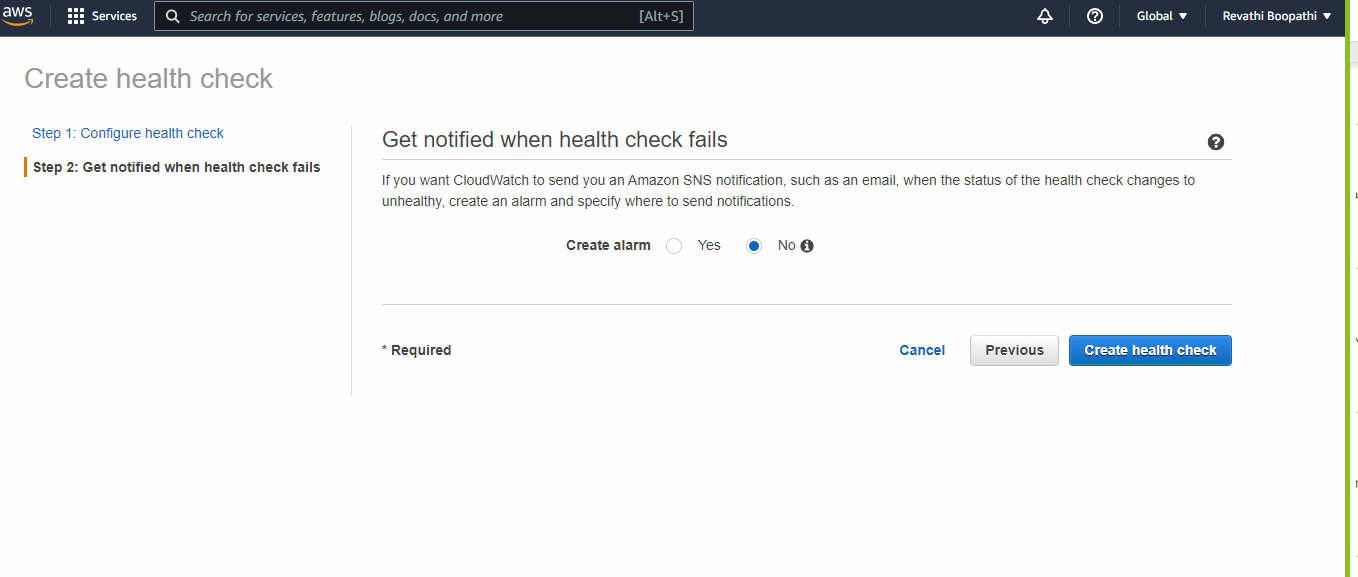
### **Configure a health check for both regions**

Navigate over to the Route53 service and choose Health checks. Create a new health check, give it an easily identifiable name e.g. ireland-api. Under *Monitor an endpoint*, specify the endpoint by domain name.

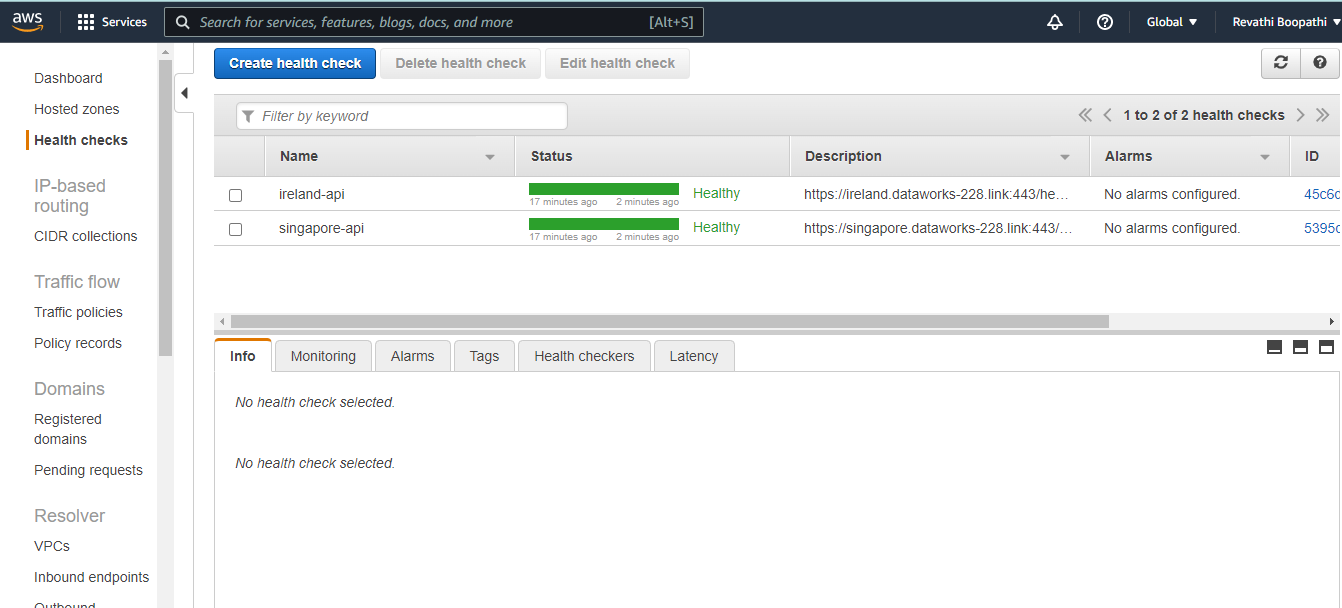
Since our API is protected by a TLS certificate you will need to change the port to 443 and the protocol to HTTPS.

You will also want to specify /health as the path as this is where our deep ping health check Lambda function is served from.

# 



# 

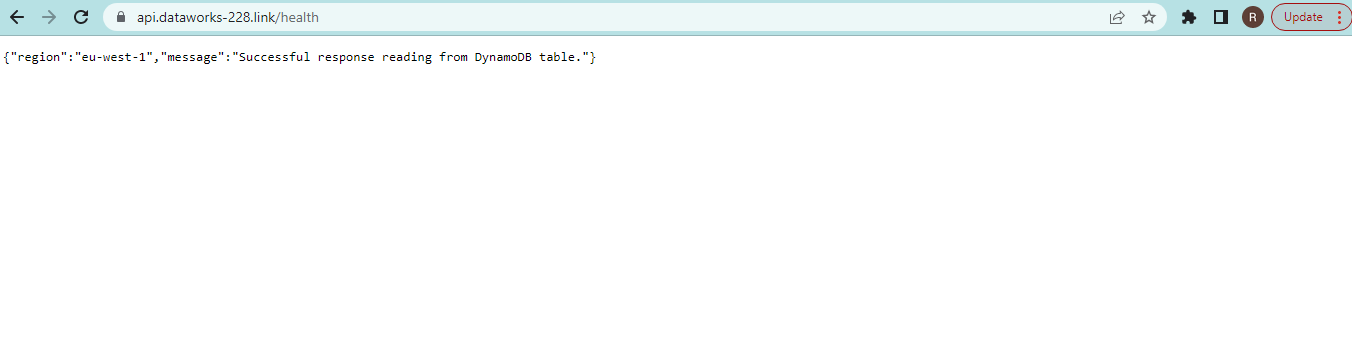


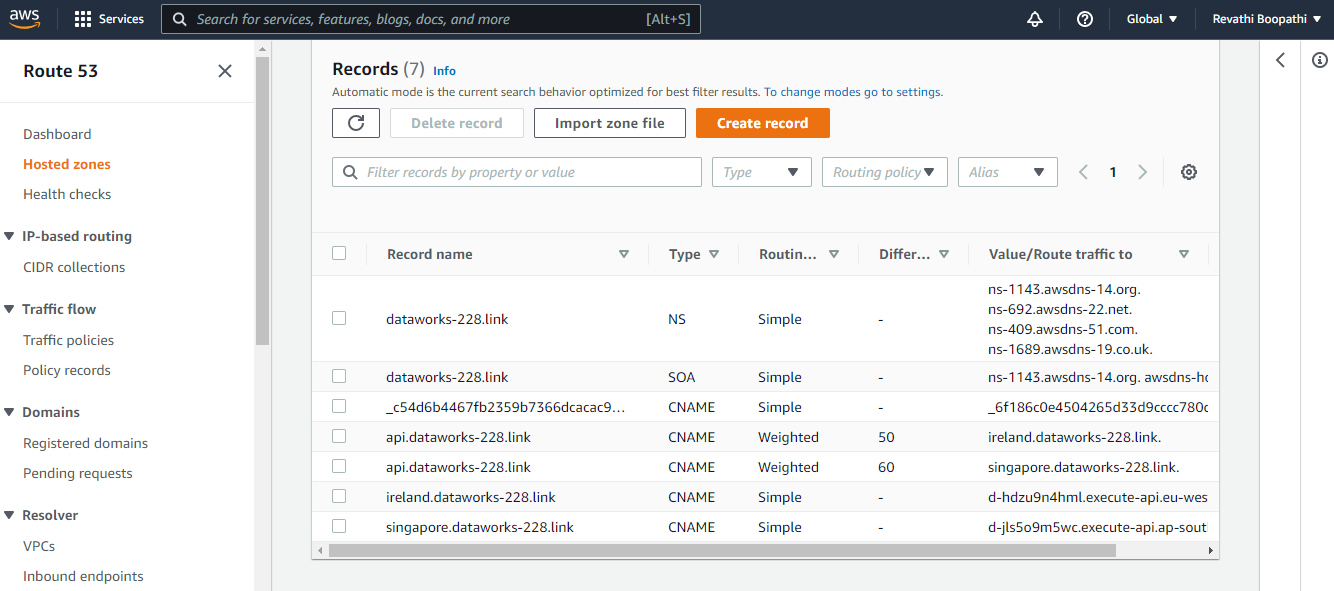
### **Configure DNS Routing Policy**

Navigate over to the Route53 service and choose Hosted zones. Choose the zone for your domain and select Create Record Set. Enter api as the name and choose CNAME as the type. Now change Alias to Yes and select the ireland. prefixed version of your domain. Since this is an alias, it should appear in the dropdown list.

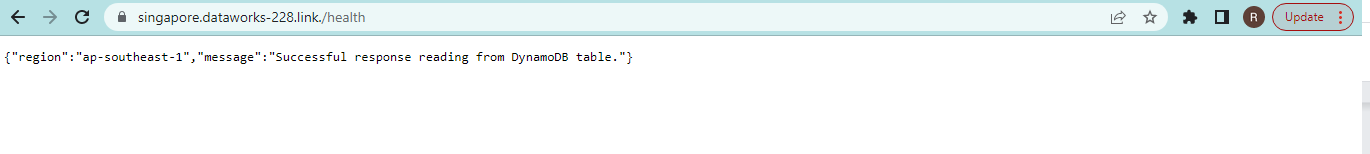
Next, choose the Weighted routing policy. You can select a value that determines the proportion of DNS queries that Route 53 responds for your Ireland record. (ex. 50) Turn on both Evaluate Target Health and Associate with Health Check then select the ireland-api health check you created previously. Hit Create.

# 





# 



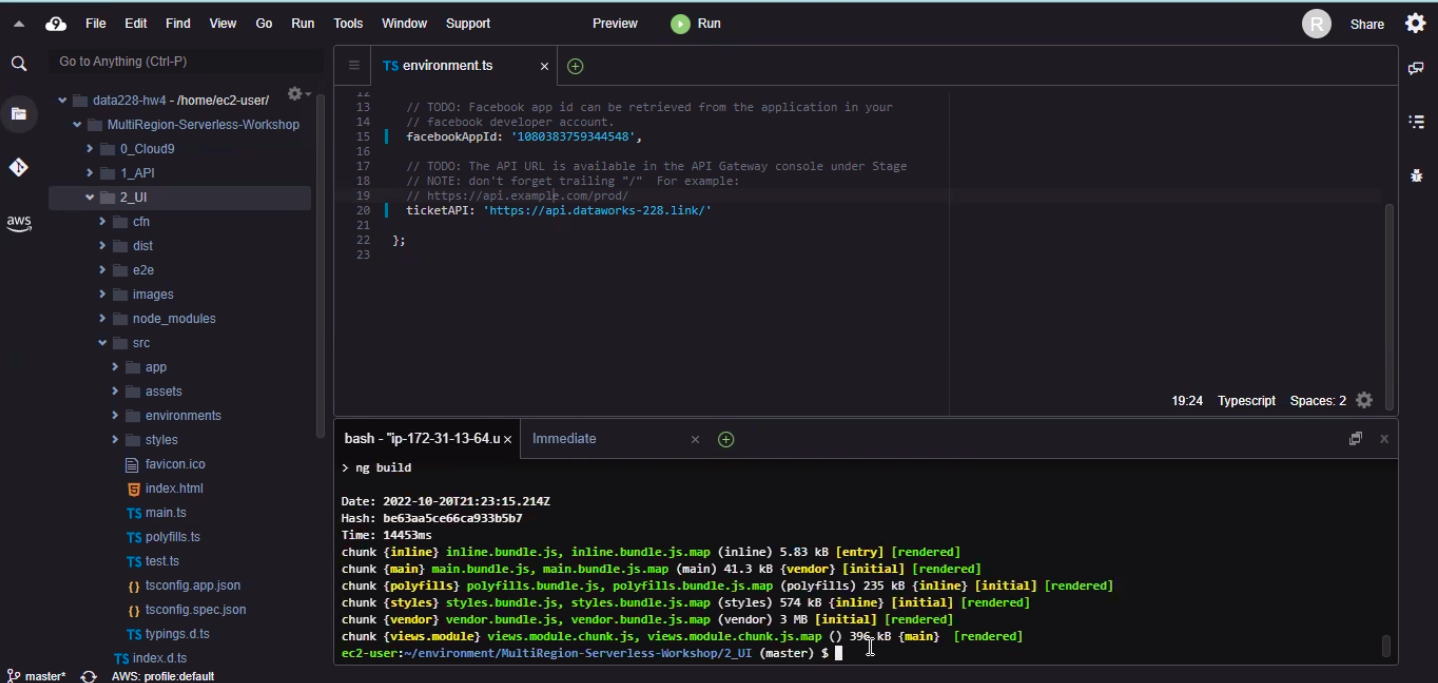
## **Step 4:** Update your UI with new API Gateway Endpoint

# 

# 

# 

npm run build is run successfully.



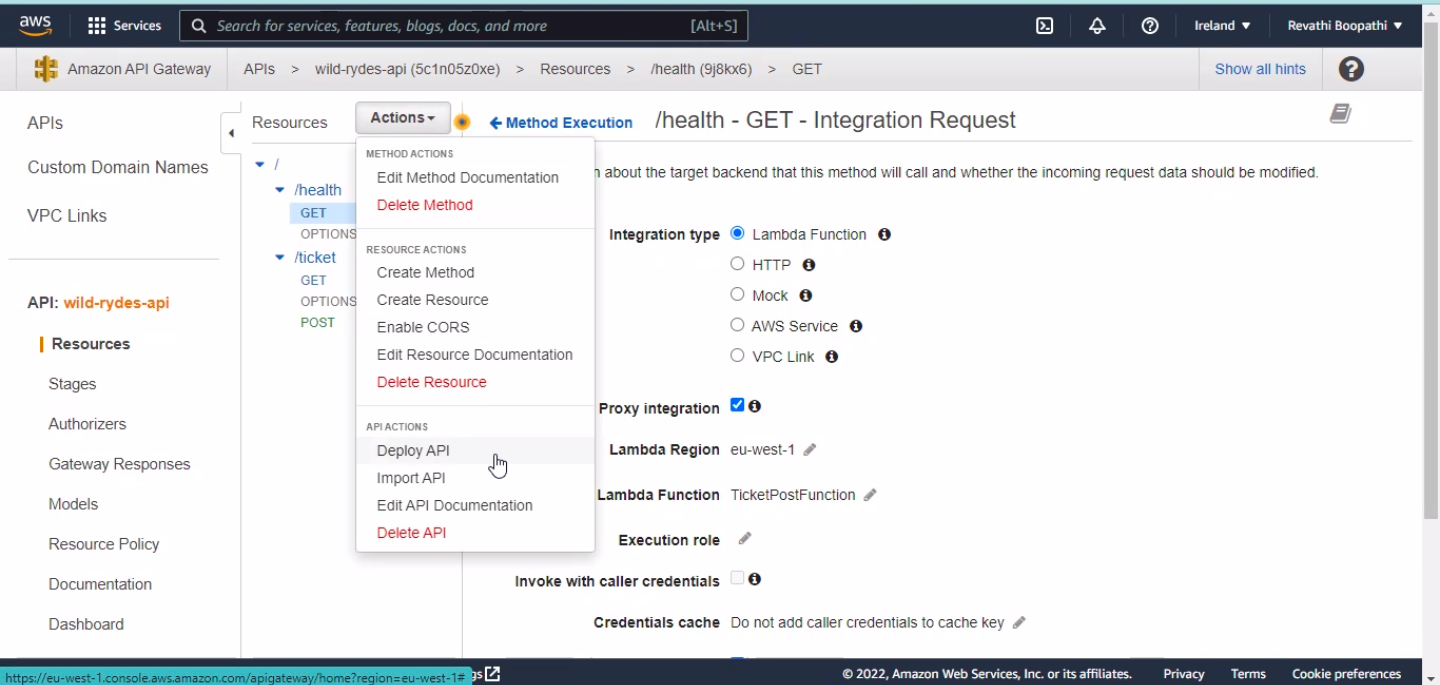
BucketName: ticket-service-ui-websitebucket-1lrasne9saj3f

aws s3 sync --delete dist/ s3://ticket-service-ui-websitebucket-1lrasne9saj3f

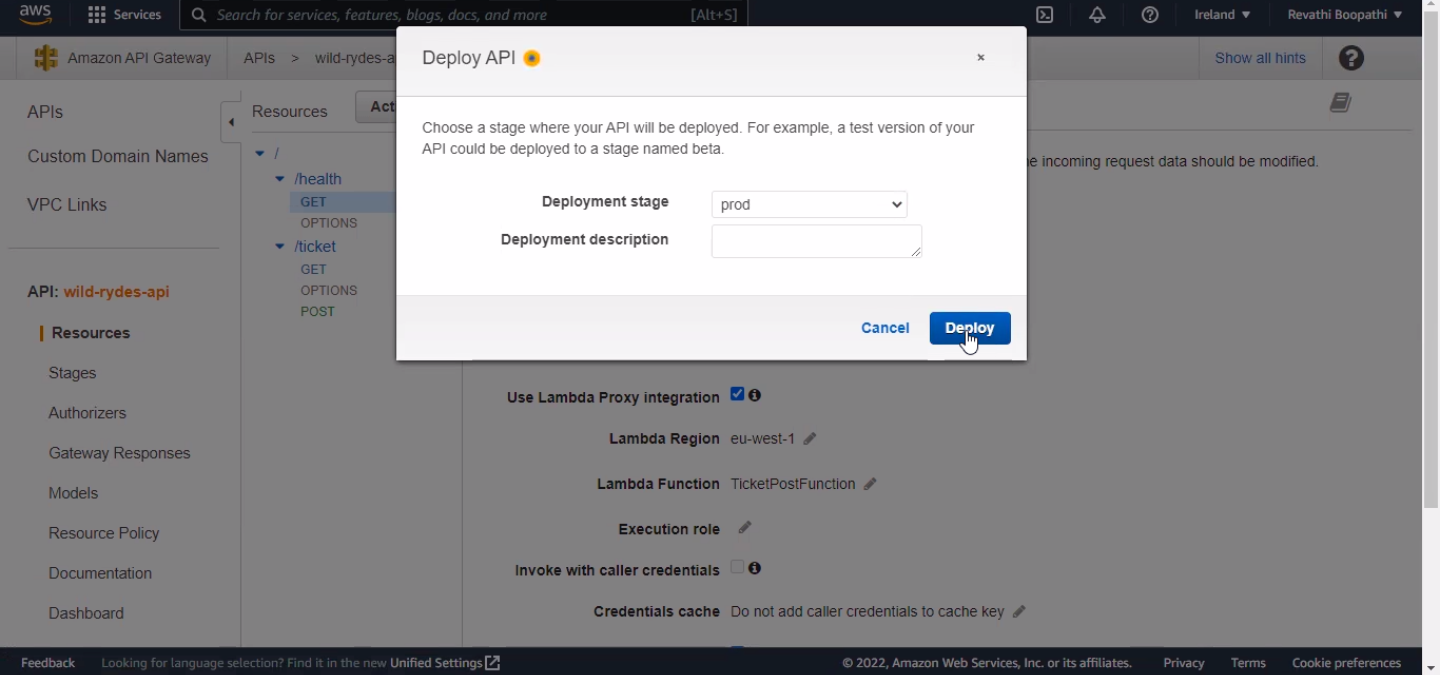
# 

# 5. Test failover

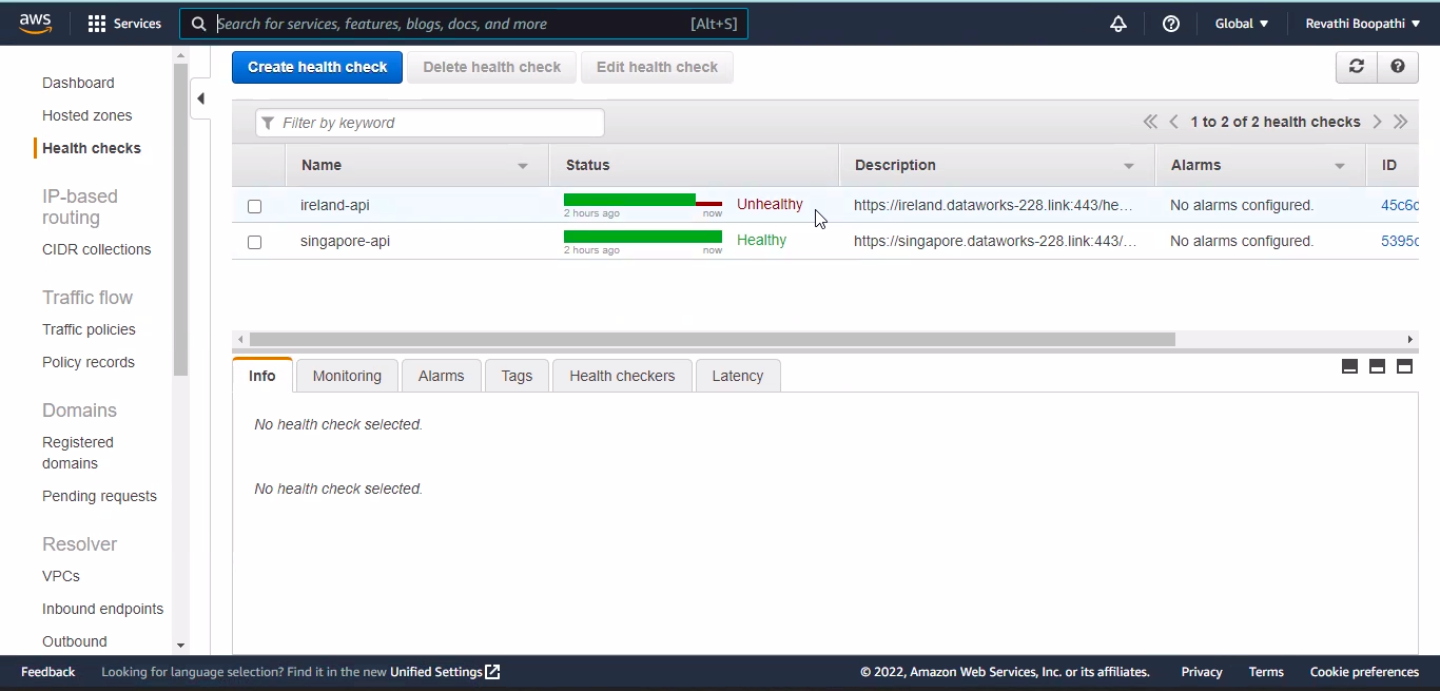
Now, we will break the primary region to check if the secondary region takes over and serves the ticketing application. Select the below configurations from custom domain names and edit the integration request as per the below image:



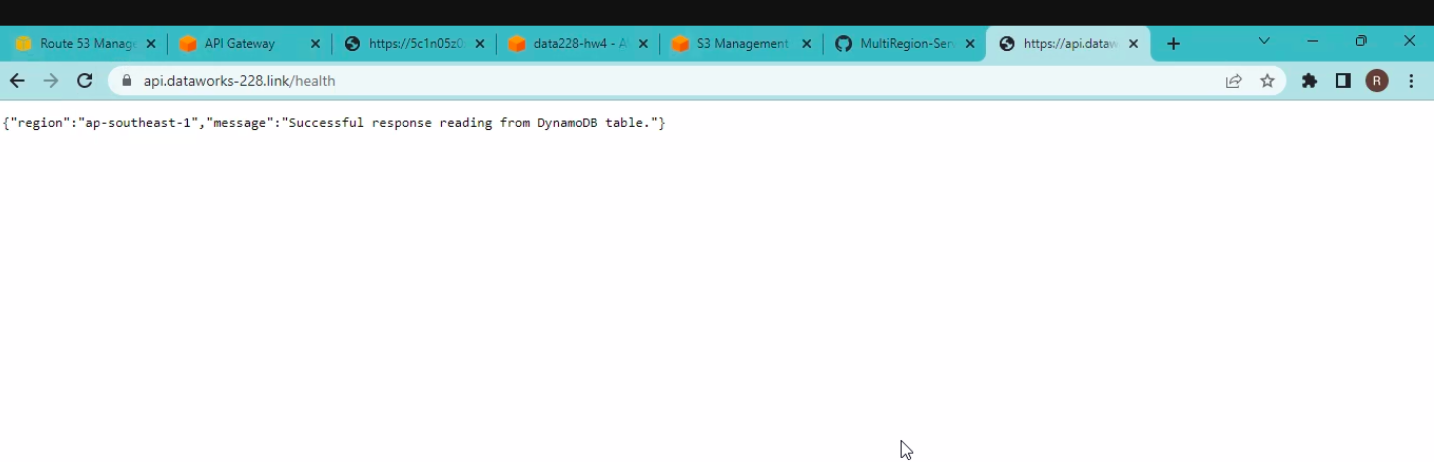
In Actions, deploy to redeploy your API to the prod stage’.



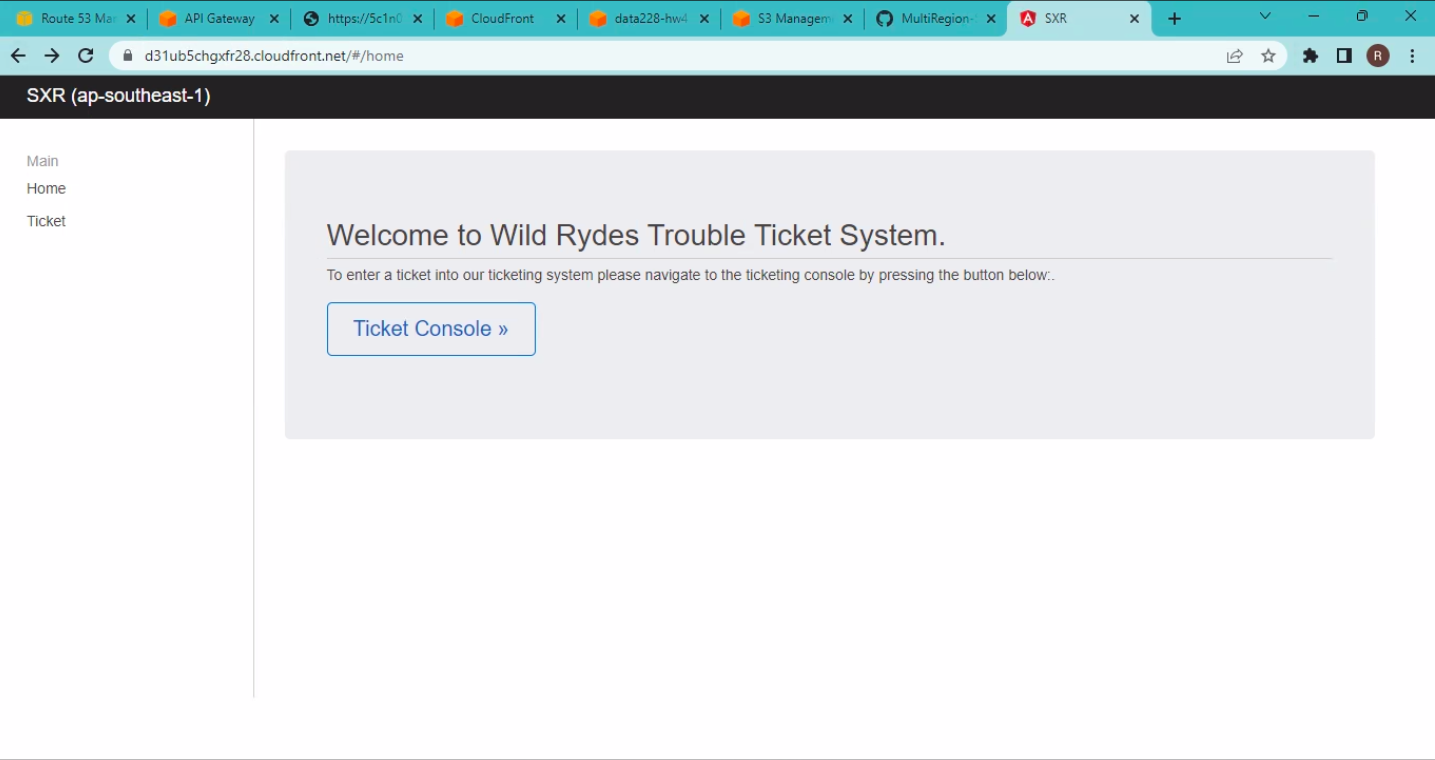
Now head over to Route53 and select Health checks. Within a few minutes, your health check should turn from green to red and display a failure.



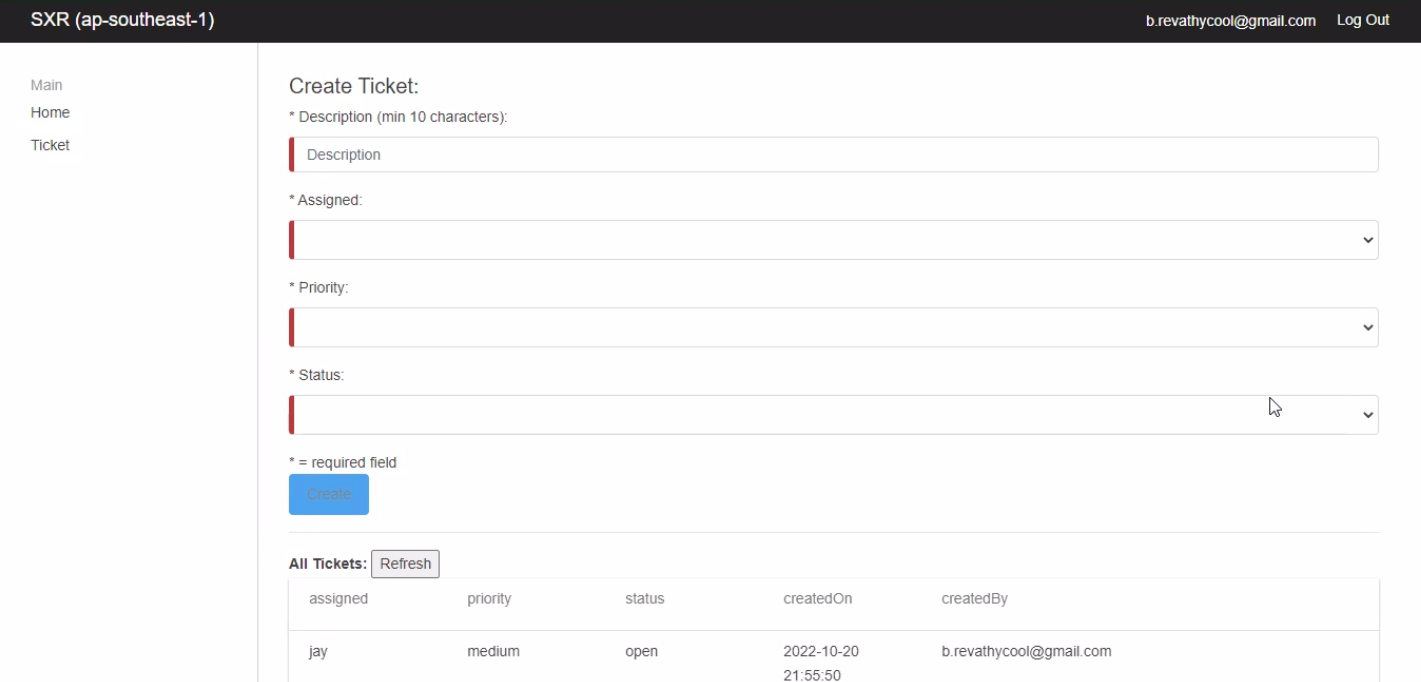
Route 53 should automatically use this information to point your domain at another region.



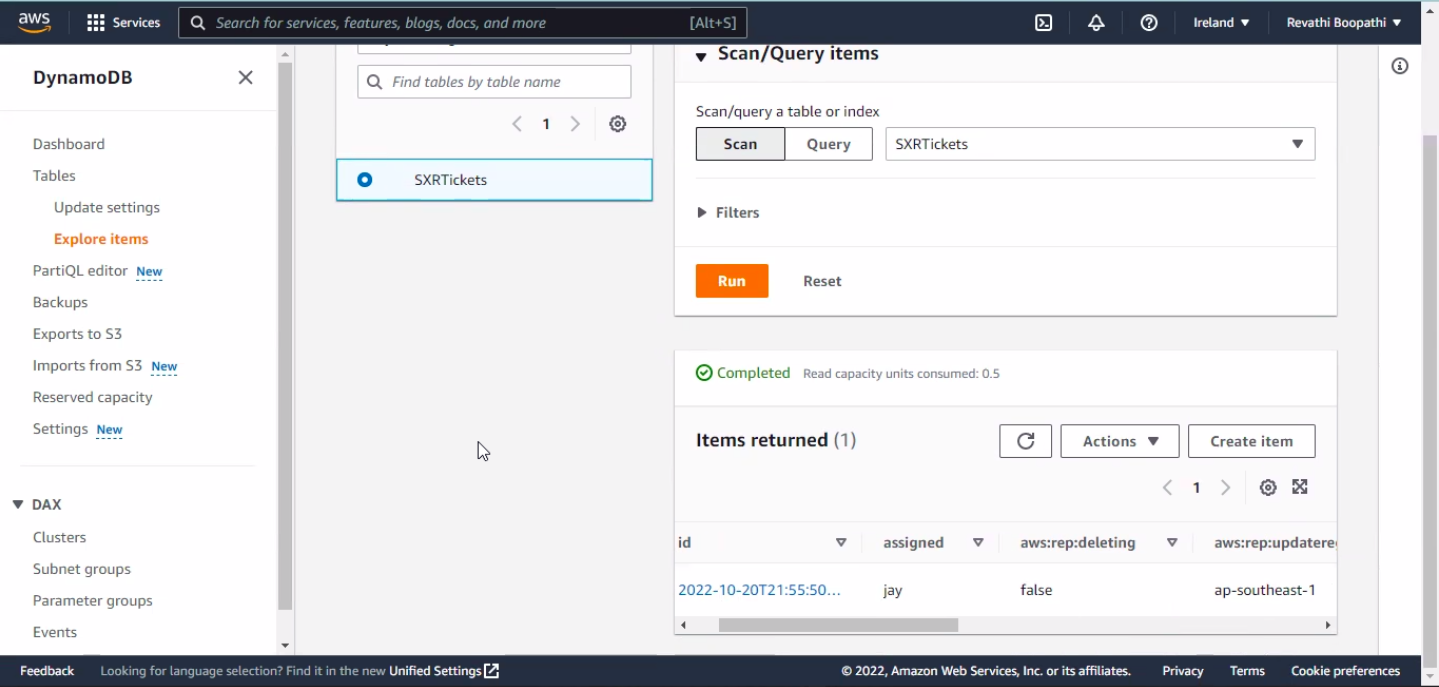
To confirm everything went as expected, go to your ticketing application (Cloudfront URL) and create a ticket again. You should see your application indicating the Singapore region.



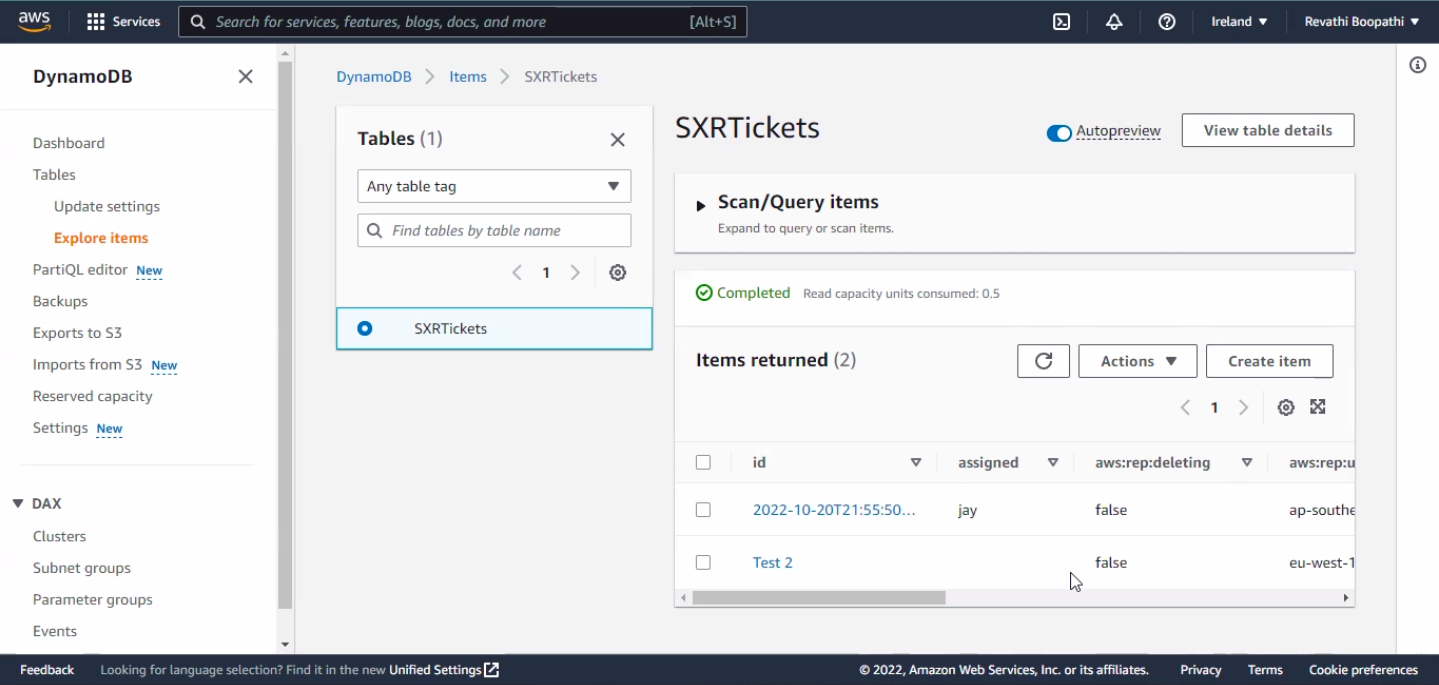
Creating ticket from the application post logging in through Facebook credentials.



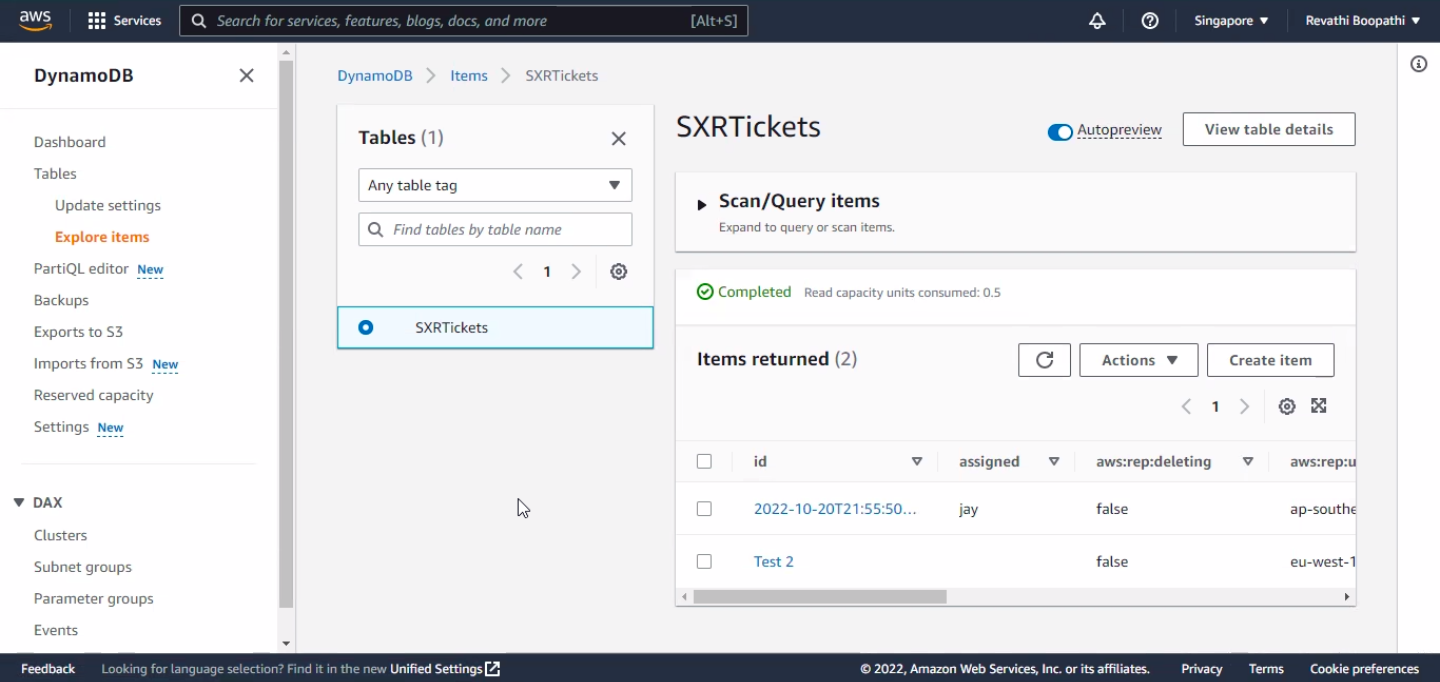
Now, go to the Dynamo db table in the AWS console and validate if the created ticket is present.



Sample ticket Test-2 is created in eu-west-1 region (Ireland)



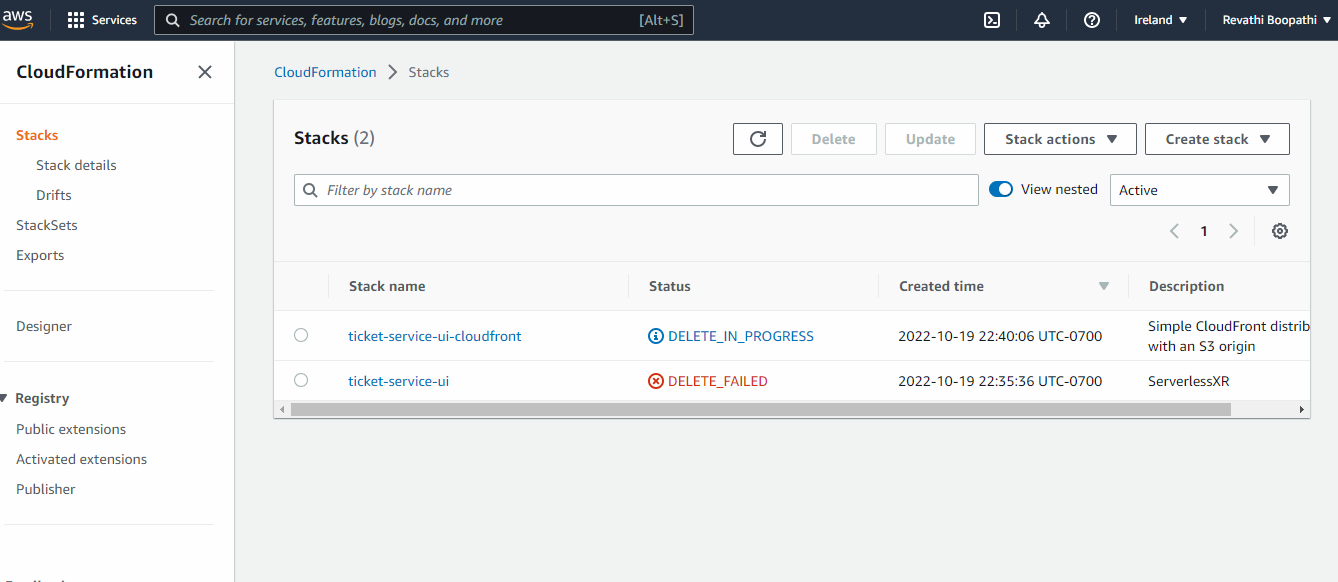
Ticket Test 2 is appearing in the Singapore region as well.



# **6. Test Template:**

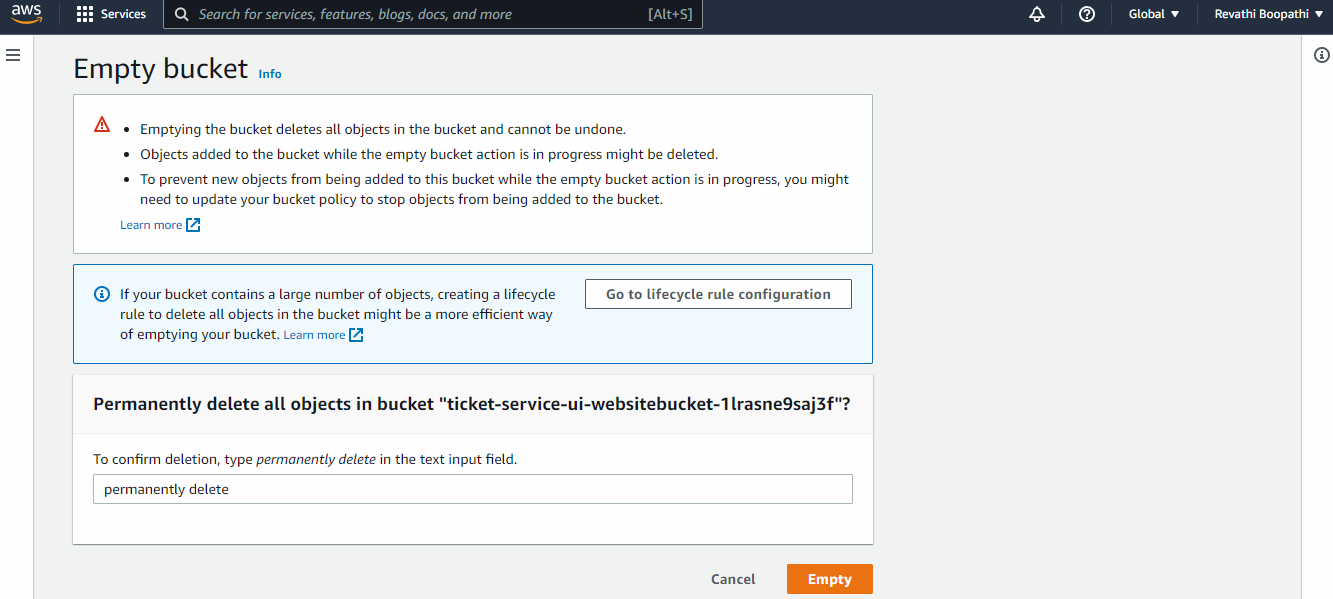
## **Module 1\_API:**

* In the console, select the correct regions
* Go into CloudFormation and select the wild-rydes-api stack and then select *delete*
* The template will be deleted along with all resources it created

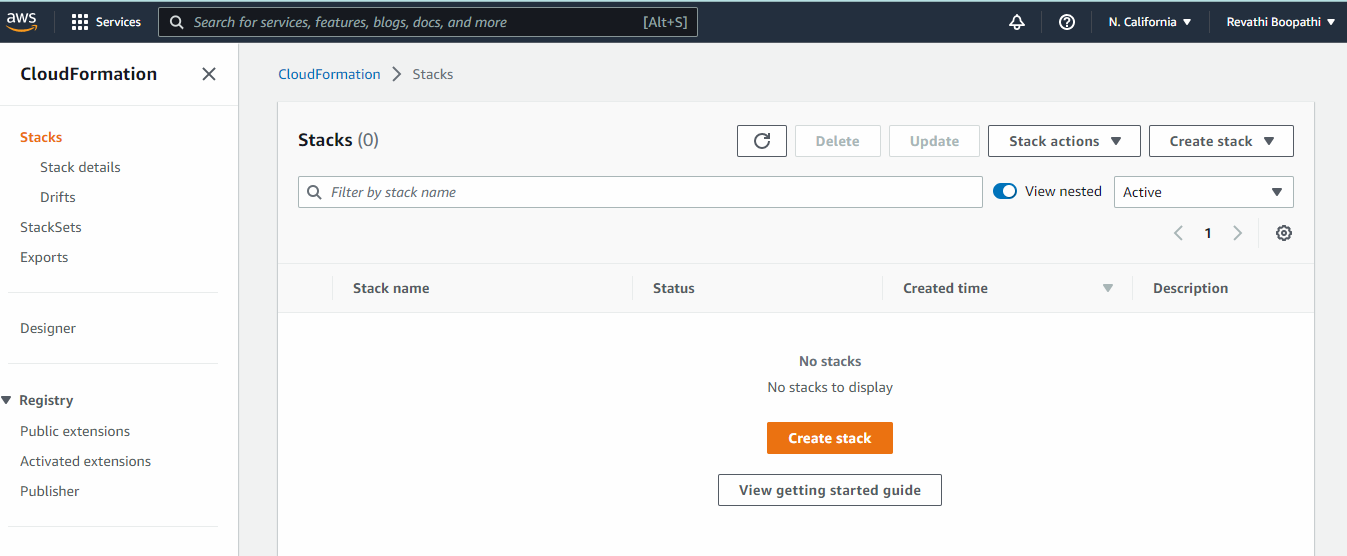


## Module 2\_UI

* In the Console, go into S3, and *Empty* but don't delete the bucket hosting your website content



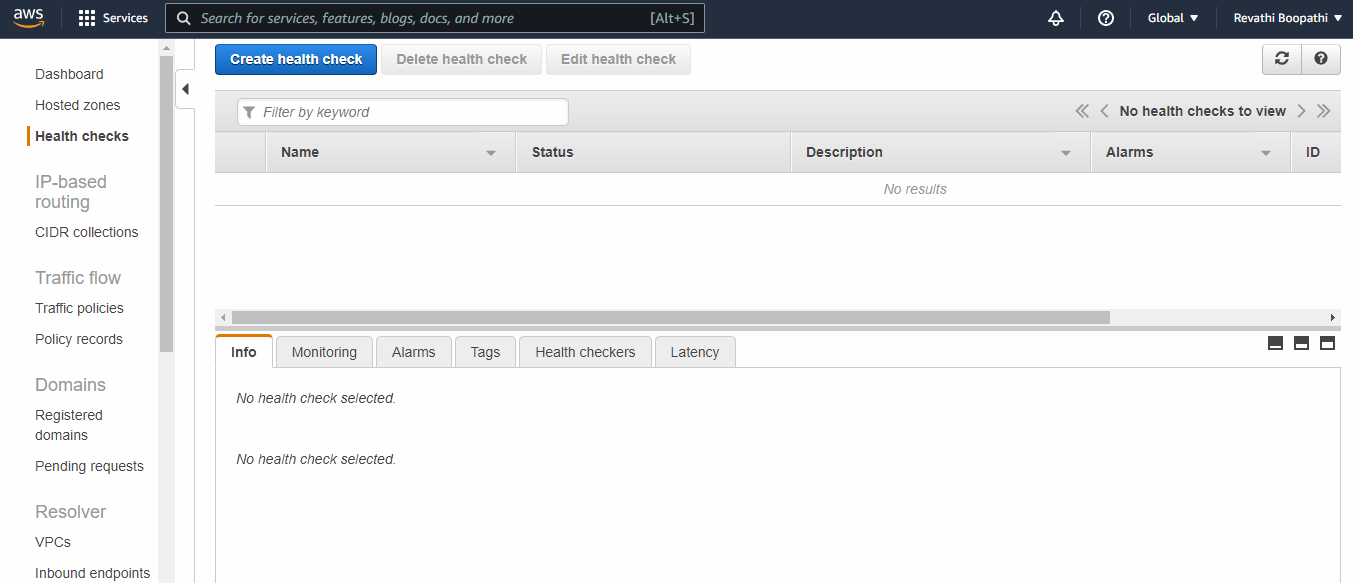
* Go to CloudFormation console, Select the web-ui-stack stack and then *delete* it.



* The template will be deleted along with all resources it created

## Module 3\_Route53

* In Route53, remove the Health Check as well as all DNS entries you created during the workshop



* In Amazon Certificate Manager, ensure all SSL certificates you created (both regions) are deleted (It was automatically deleted)

## Delete Cloud9 environment (EC2 and EBS)

* In Cloud9, select the environment you created and then click Delete (this will delete it completely, so if there is any data on it you wanted to save, ensure you take care of that first)

