Deployment of Serverless Application on AWS

Table of Contents

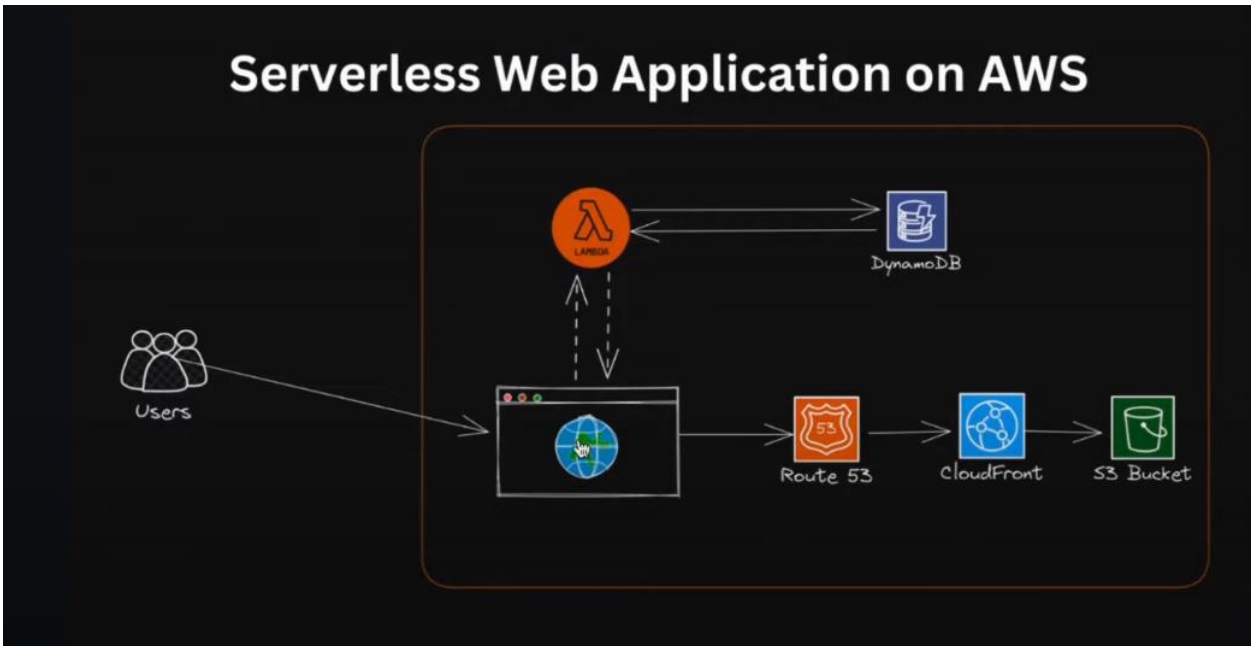
Project Description: -

In this Project, we will build a serverless web application using AWS Lambda, DynamoDB and S3. The application will allow users to create, read, update, and delete (CRUD) items from Dynamo DB table.

1. Introduction

This document provides a detailed overview of deploying a serverless web application using AWS services. It aims to outline the architecture, components, deployment process, security considerations, and cost management strategies.

2. Architecture Overview



The serverless web application leverages various AWS services to ensure scalability, high availability, and cost efficiency.

The architecture consists of the following key components:

- **Users**: End users who interact with the web application.
- **S3 Bucket**: Stores static assets (HTML, CSS, JavaScript).
- **CloudFront**: Distributes content globally with low latency.
- **Route 53**: DNS service to route user requests.
- **Lambda**: Executes backend code without provisioning servers.
- **DynamoDB**: NoSQL database for storing application data.

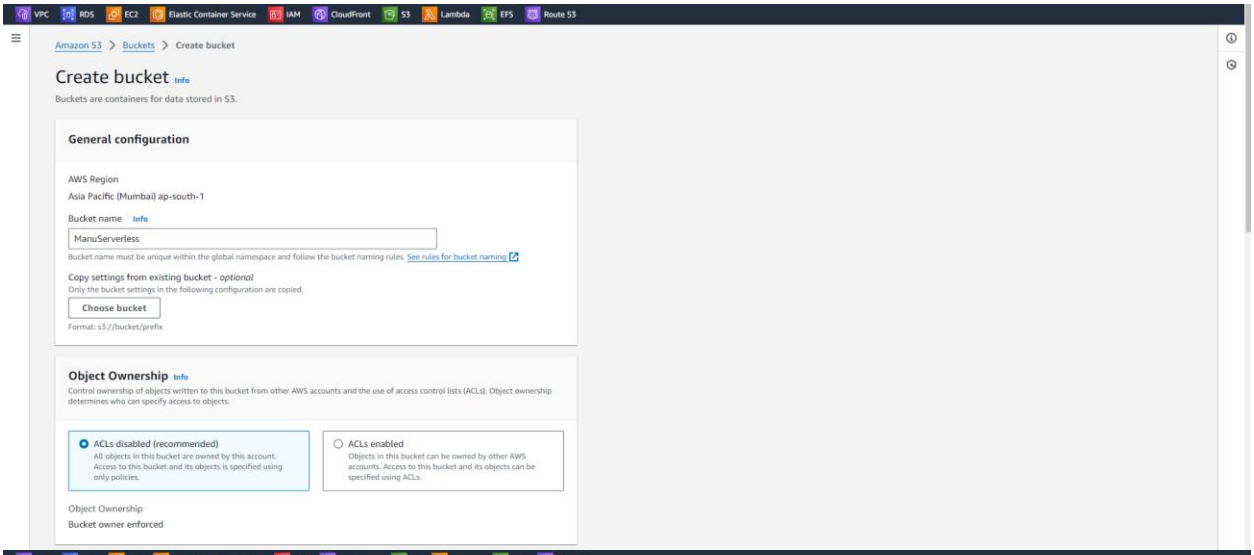
3. Components and Services

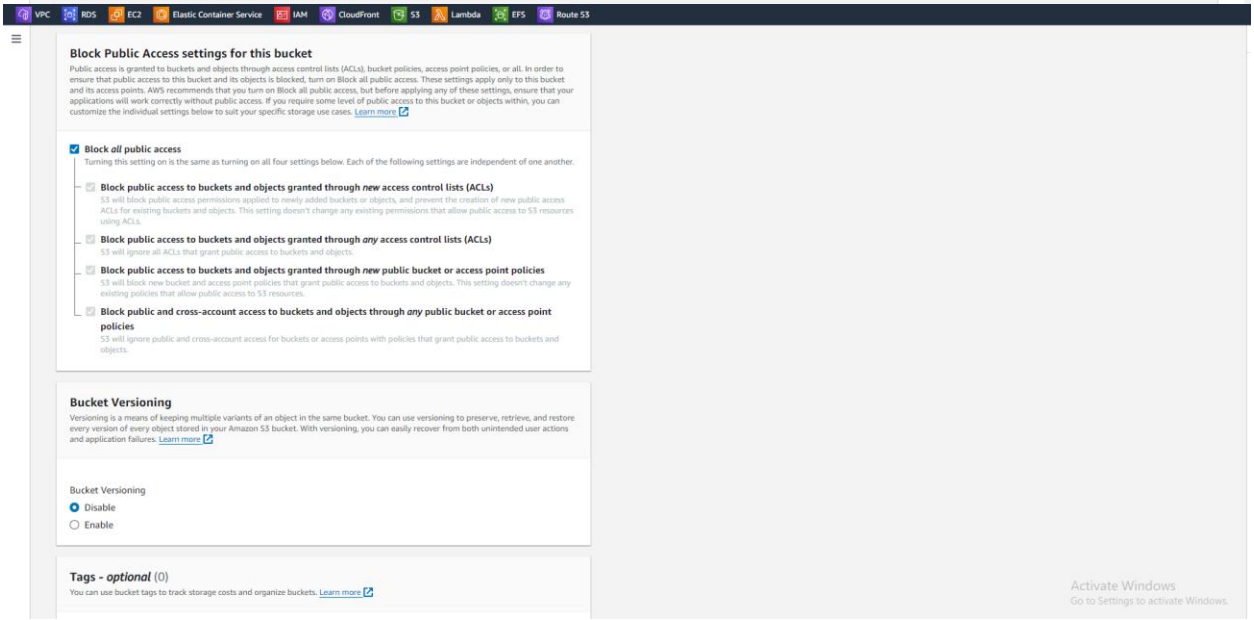
Amazon S3

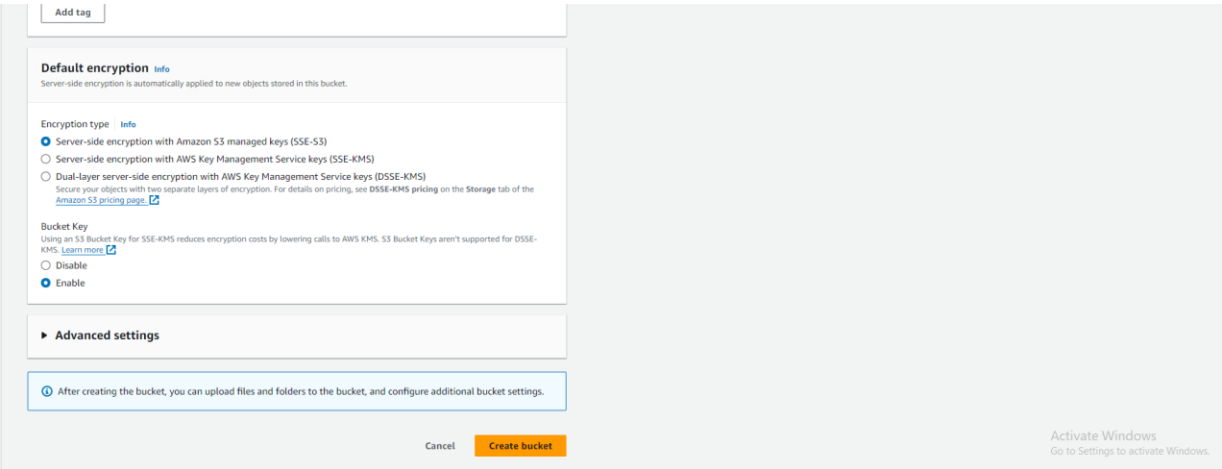
Amazon Simple Storage Service (S3) is used to store static website content, such as HTML, CSS, and JavaScript files. S3 provides high durability, availability, and scalability for static content.

Setup Steps:

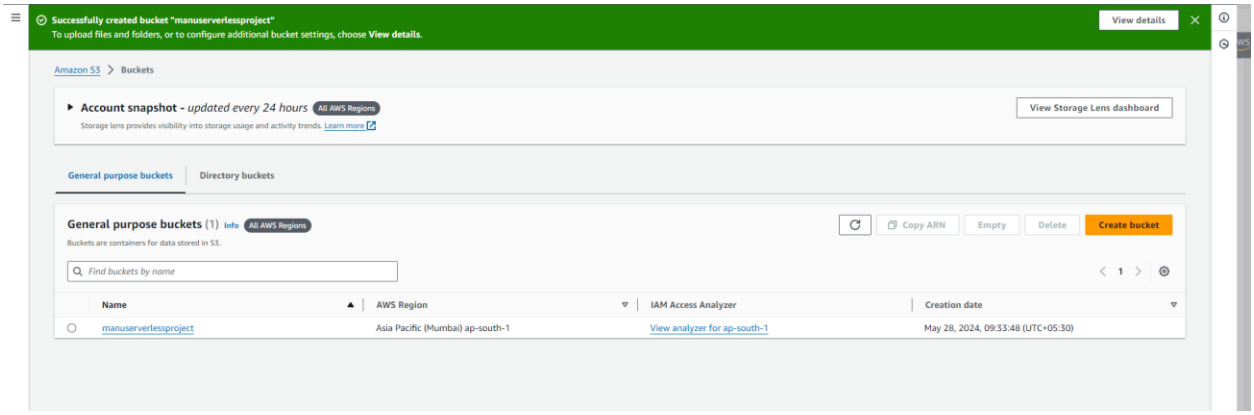
Create an S3 bucket.



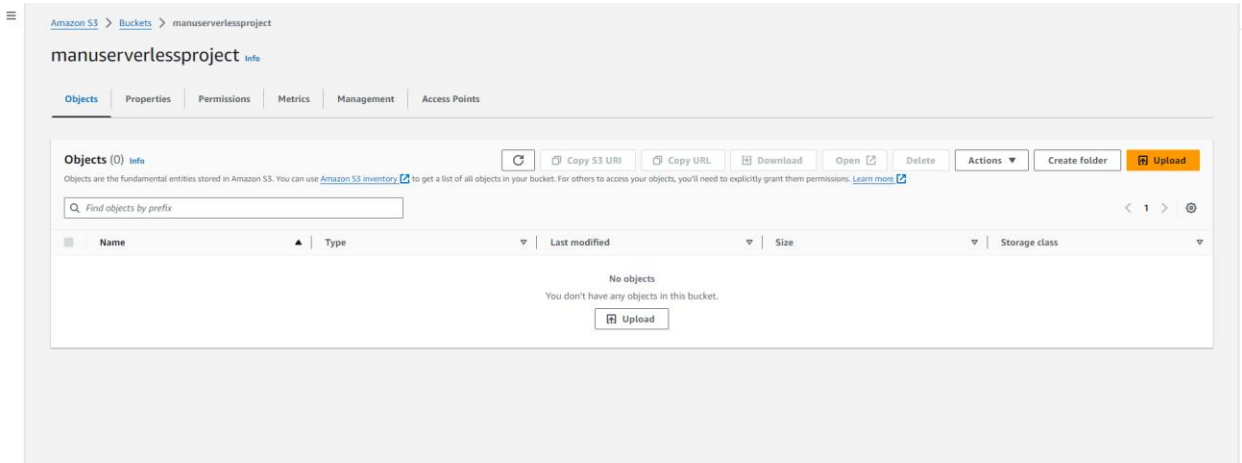


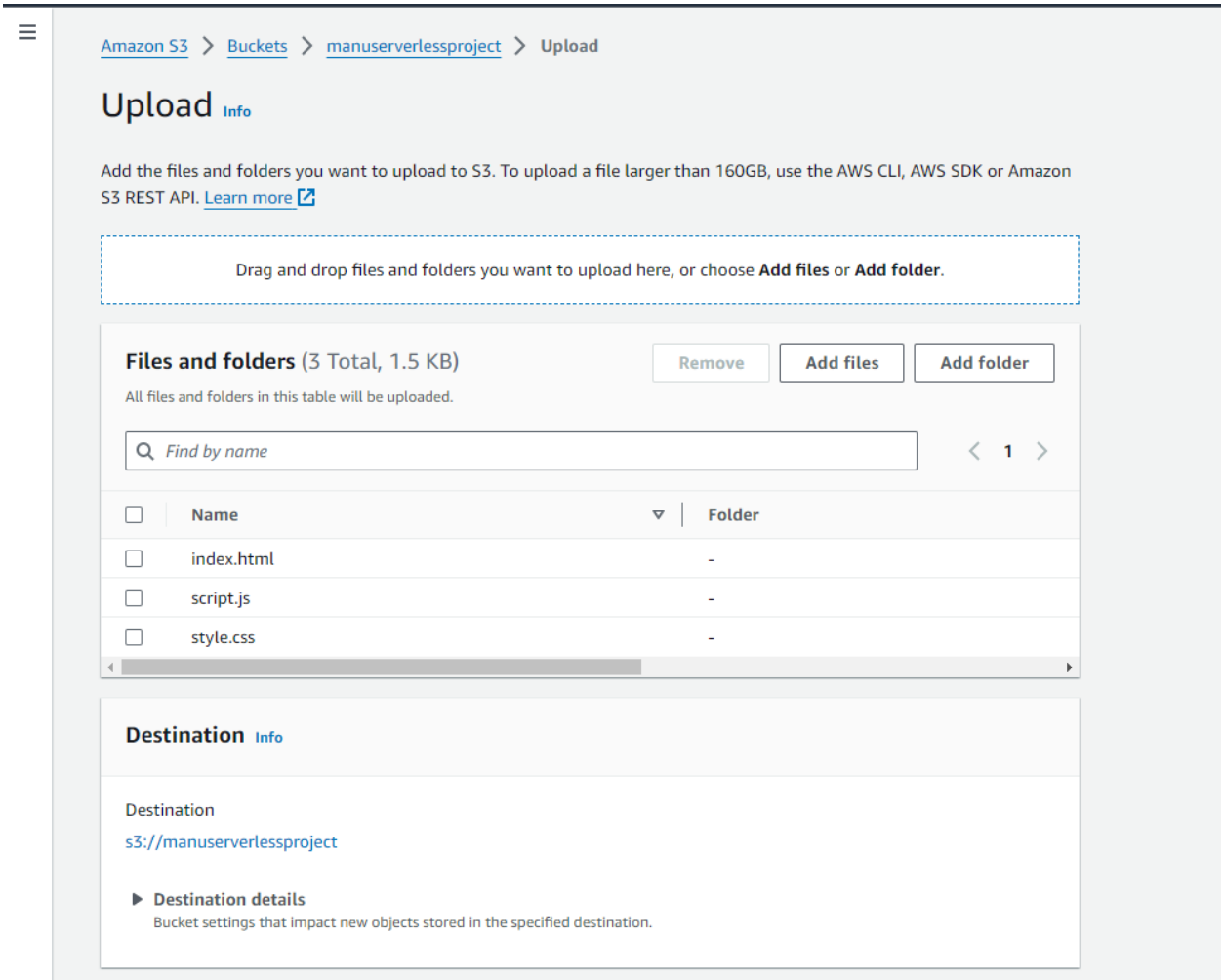


Click on Create Bucket.

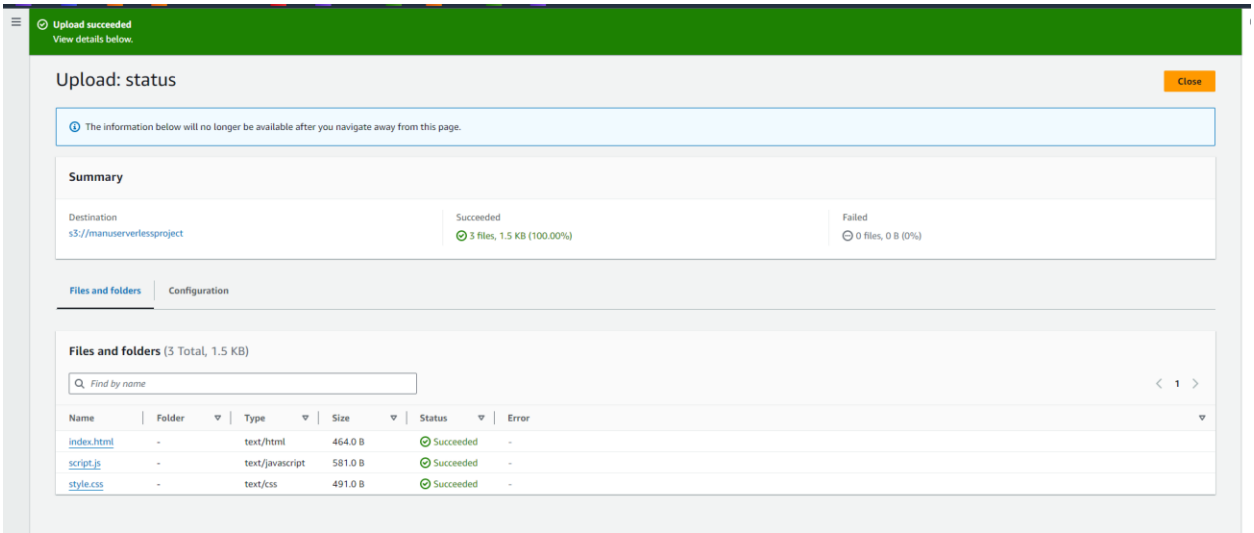


Upload static files to the bucket.



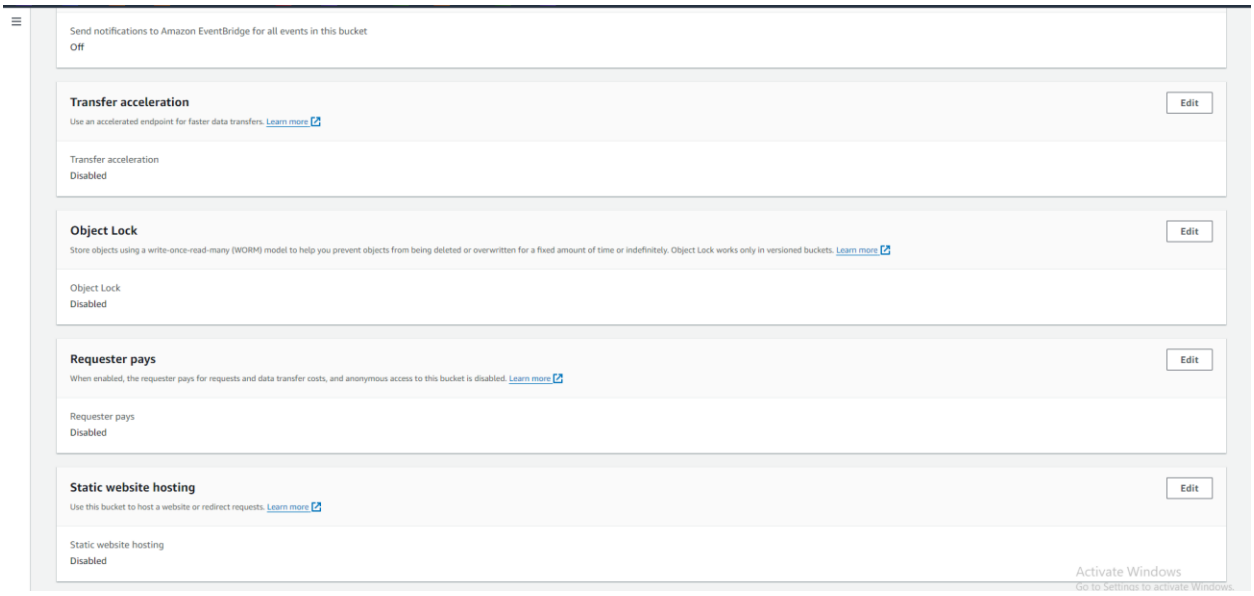


Click on Upload.

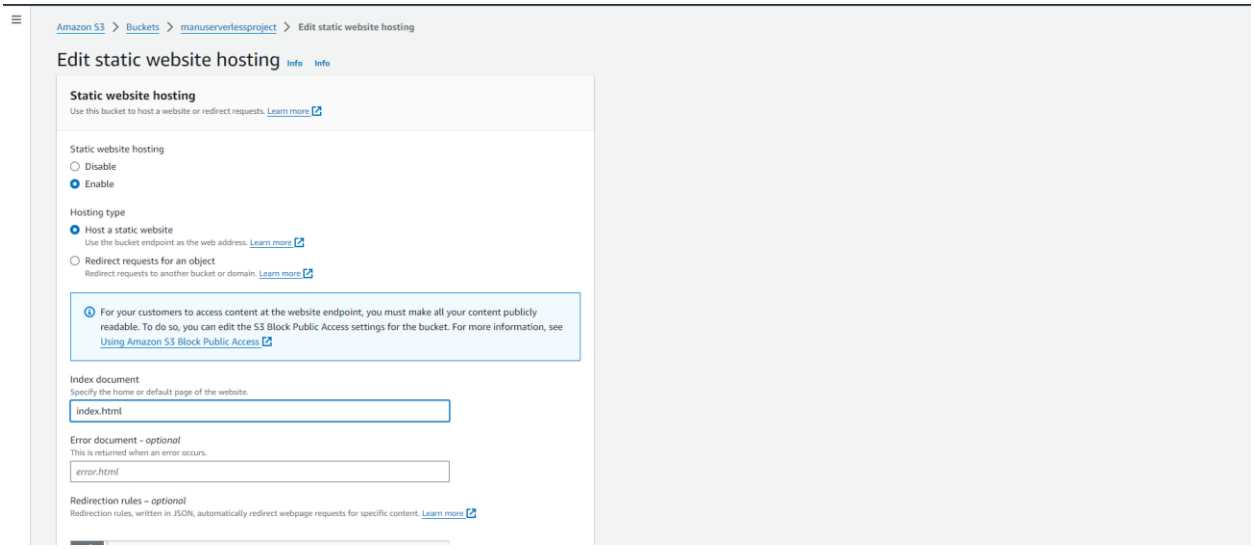


Configure the bucket to host a static website.

S3 bucket → properties.



Edit and enable static website hosting.

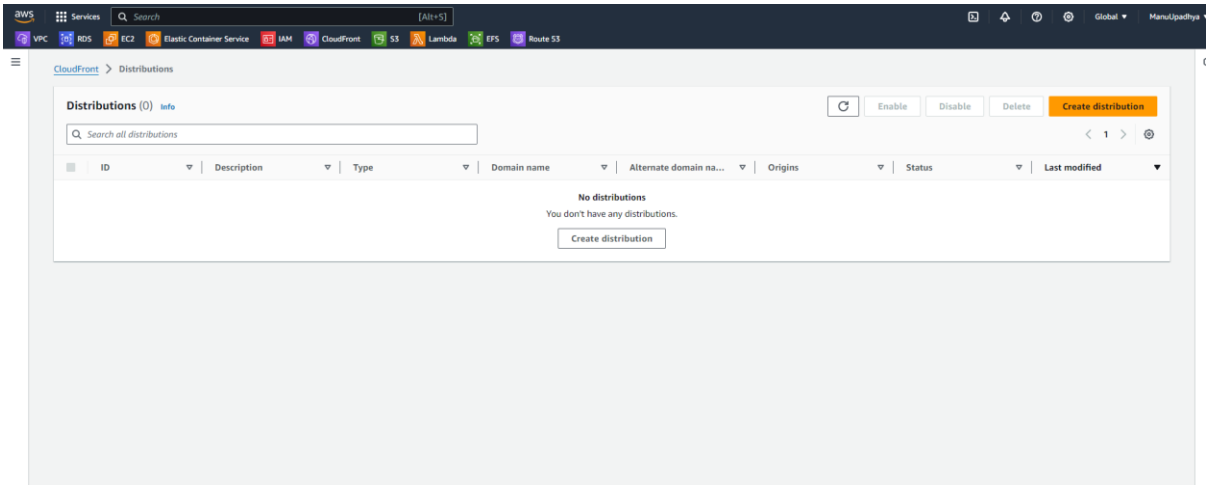


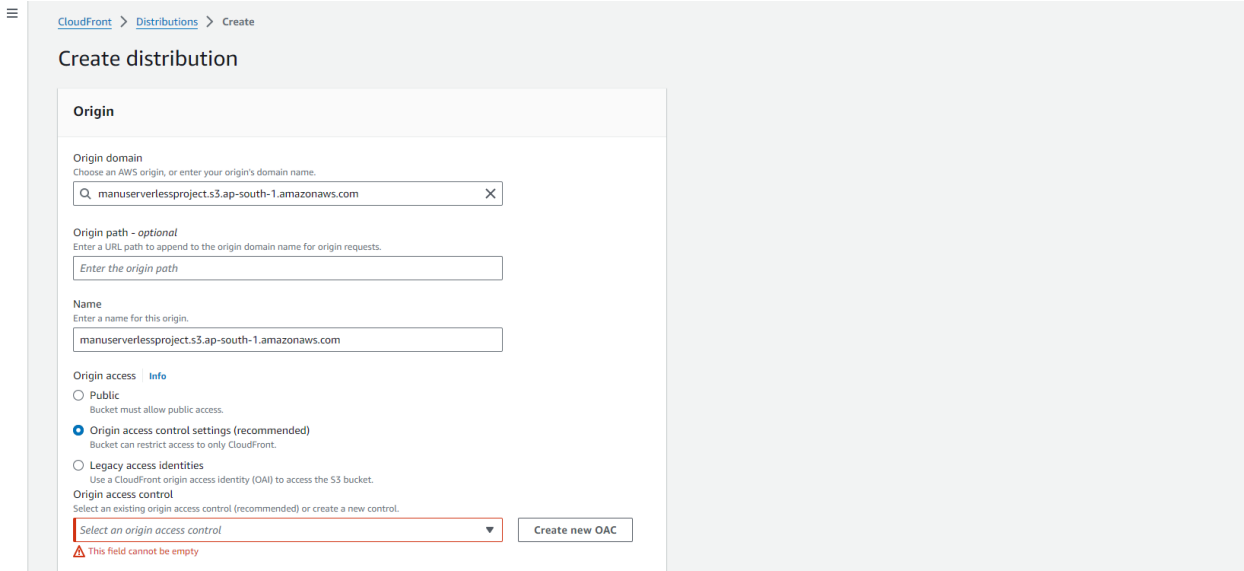
Amazon CloudFront

Amazon CloudFront is a content delivery network (CDN) that caches content at edge locations to reduce latency and improve load times for users.

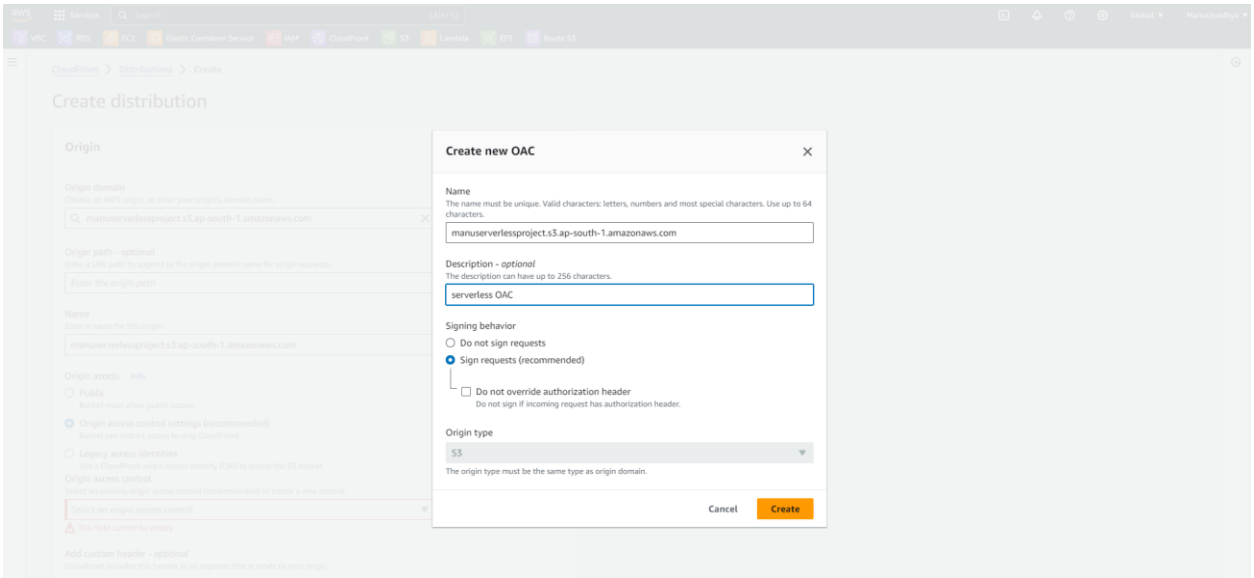
Setup Steps:

Create a CloudFront distribution→Specify the S3 bucket as the origin.

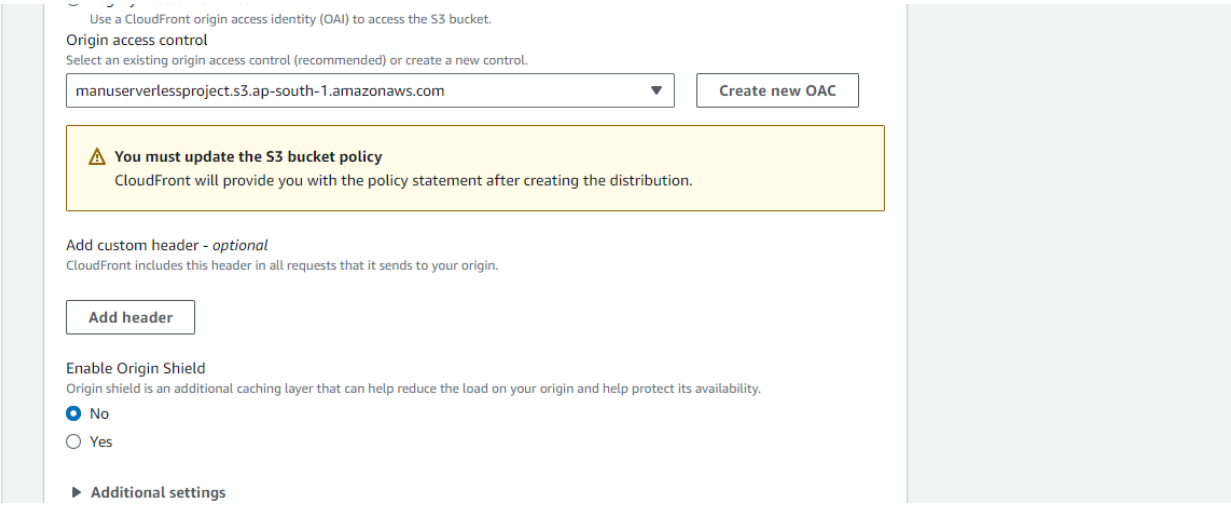




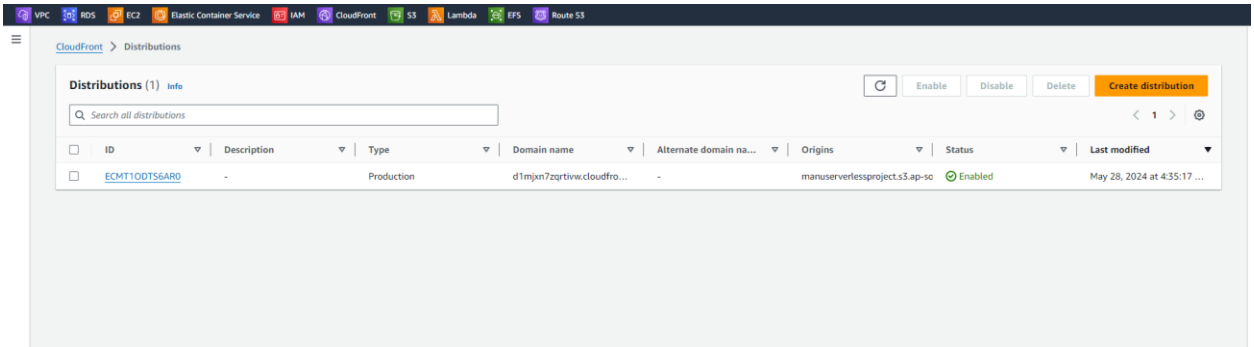
Create a new OAC



Click on create, select the created OAC.

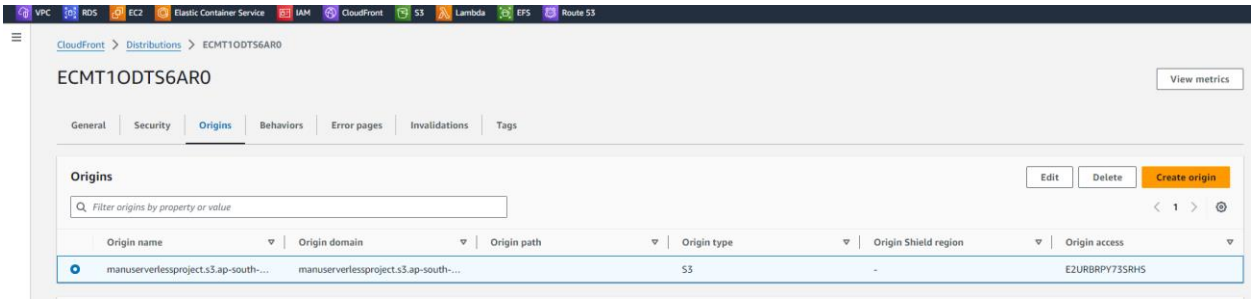


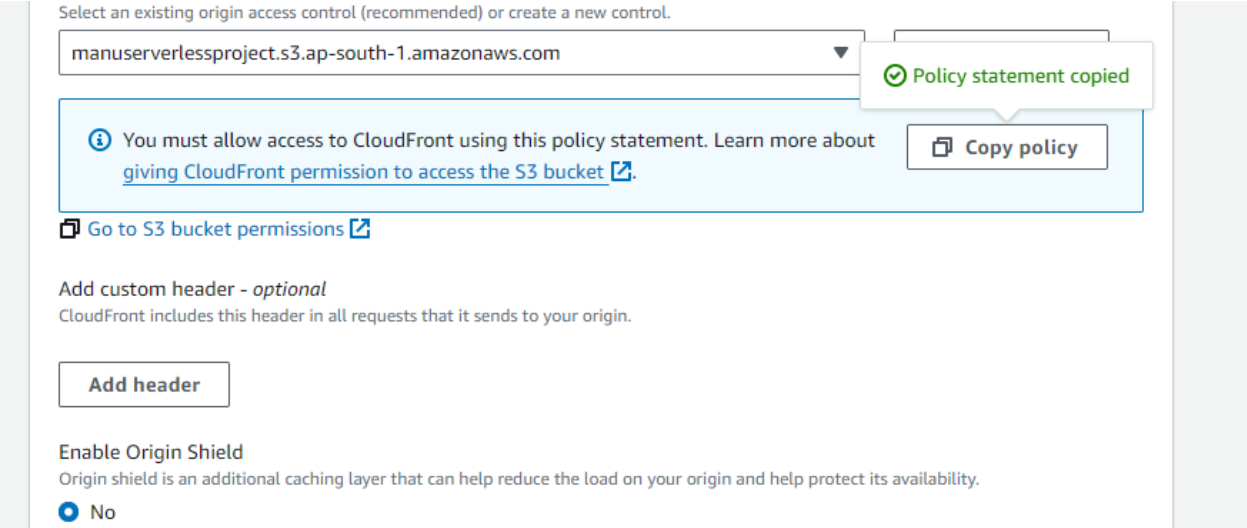
Leave everything default and click on create Distribution.



Configure caching policies and distribution settings.

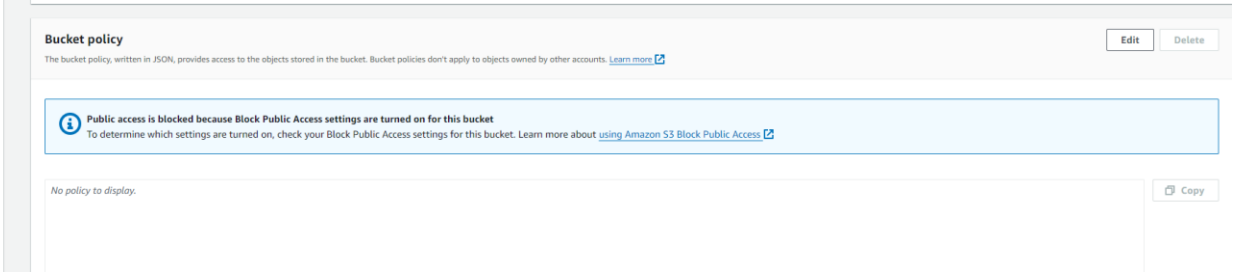
Get inside the distribution→ origins→Select the created origin, click on edit.

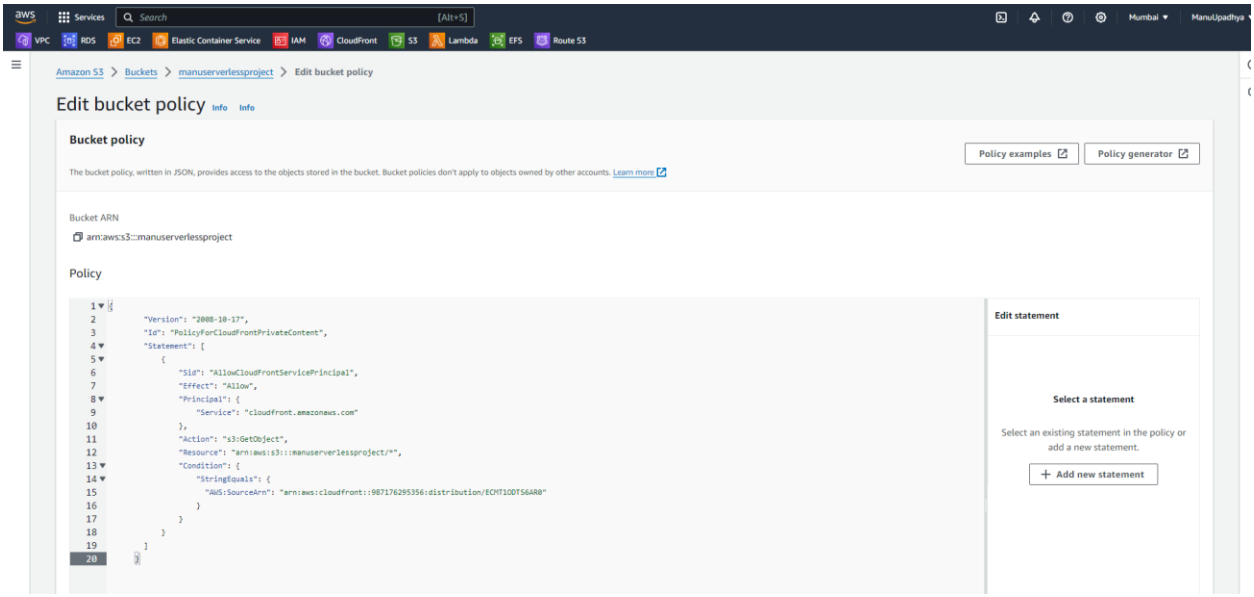




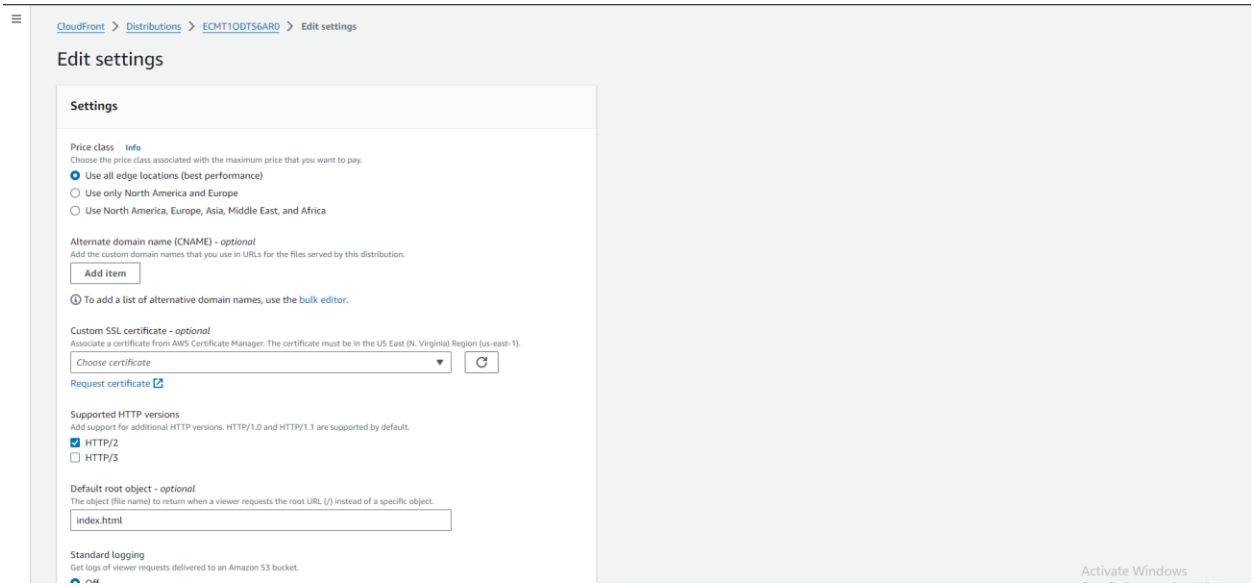
Click on copy policy. We need to paste this in our Amazon S3 bucket.

Created S3 bucket → Permissions → Edit bucket Policy and Paste the copied Policy.





Save changes. Adding Policies is Completed.



In CloudFront Distribution settings set root object as **index.html**

Amazon Route 53

Amazon Route 53 is a scalable DNS and domain name registration service. It routes user requests to the CloudFront distribution.

Setup Steps:

Register a domain or use an existing domain.

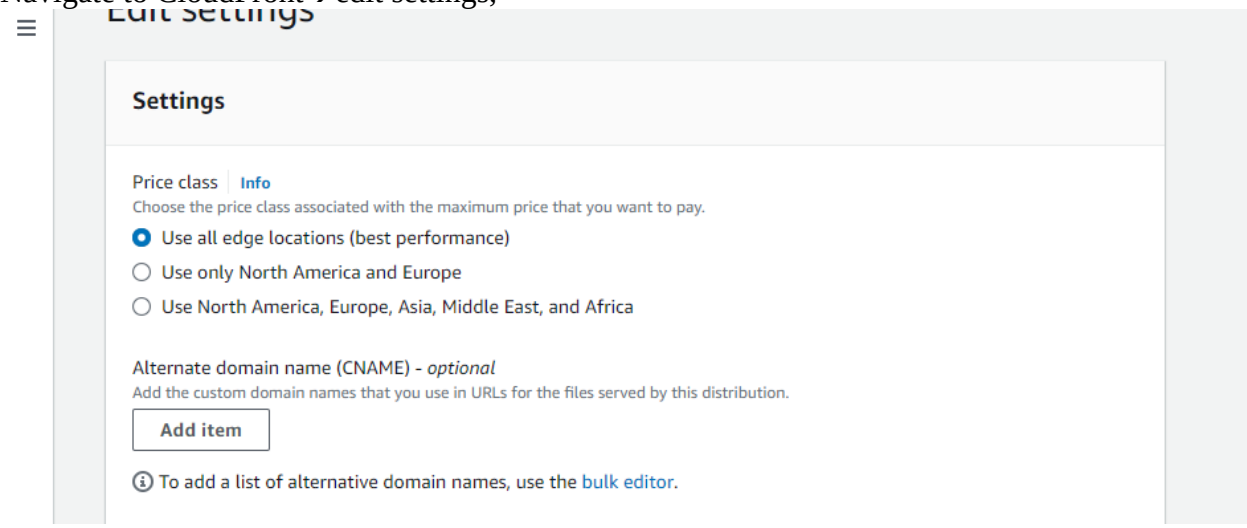
I already have a domain name so I will be using the same domain.

Create a hosted zone for the domain.

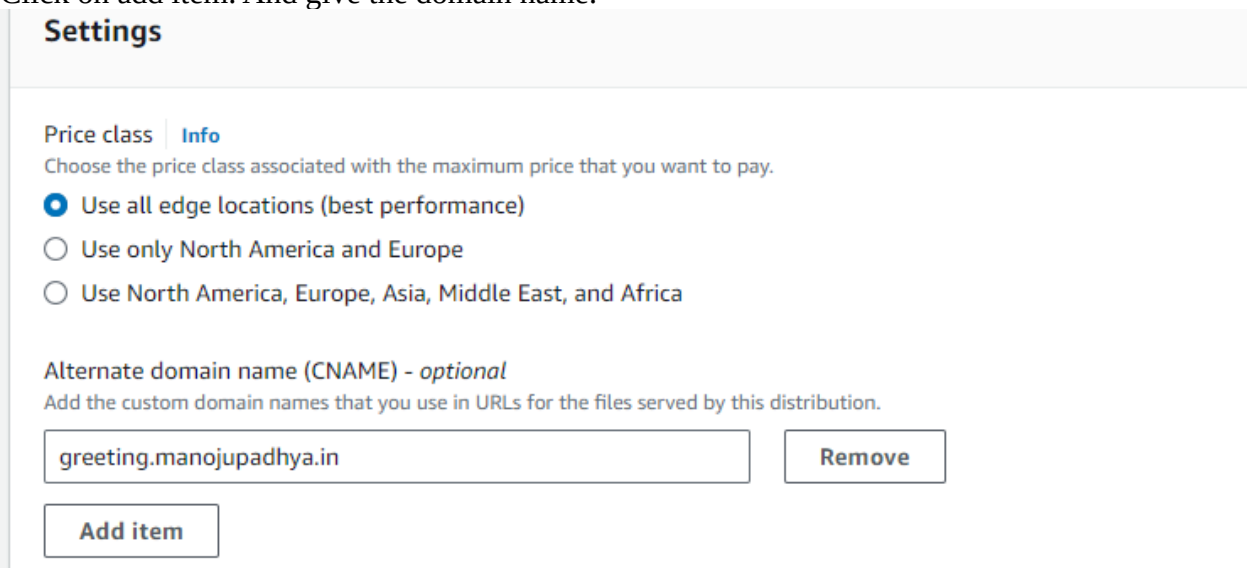
I already have a hosted zone created previously I will be using the same.

Configure DNS records to point to the CloudFront distribution.

Navigate to CloudFront→edit settings,



Click on add item. And give the domain name.



Request SSL Certificate.
Create a SSL Certificate using Amazon Certificate Manager.

AWS Certificate Manager > Certificates > Request certificate

Request certificate

Certificate type Info

ACM certificates can be used to establish secure communications access across the internet or within an internal network. Choose the type of certificate for ACM to provide.

- (●) Request a public certificate
  Request a public SSL/TLS certificate from Amazon. By default, public certificates are trusted by browsers and operating systems.
- ( ) Request a private certificate
  No private CAs available for issuance.

Requesting a private certificate requires the creation of a private certificate authority (CA). To create a private CA, visit AWS Private Certificate Authority

Cancel | Next

AWS Certificate Manager > Certificates > Request certificate > Request public certificate

Request public certificate

Domain names

Provide one or more domain names for your certificate.

Fully qualified domain name Info

*.manojupadhya.in

Add another name to this certificate

You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name.

Validation method Info

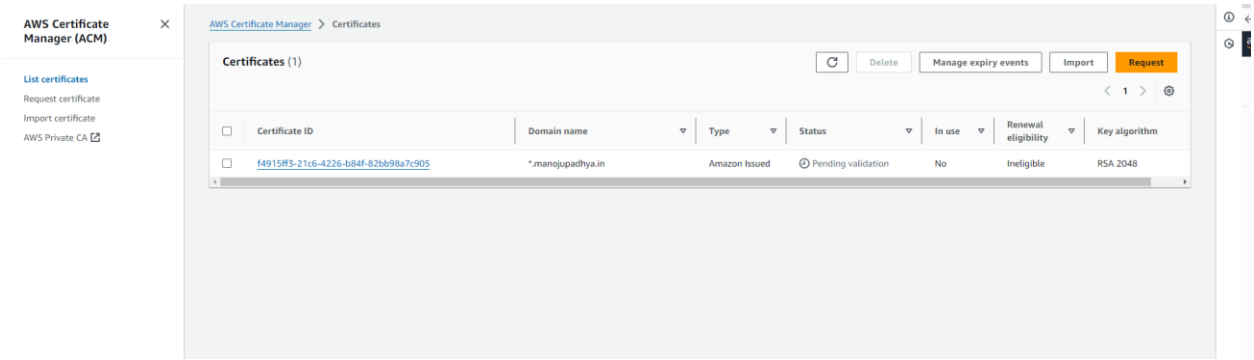
Select a method for validating domain ownership.

- (●) DNS validation - recommended
  Choose this option if you are authorized to modify the DNS configuration for the domains in your certificate request.
- ( ) Email validation
  Choose this option if you do not have permission or cannot obtain permission to modify the DNS configuration for the domains in your certificate request.

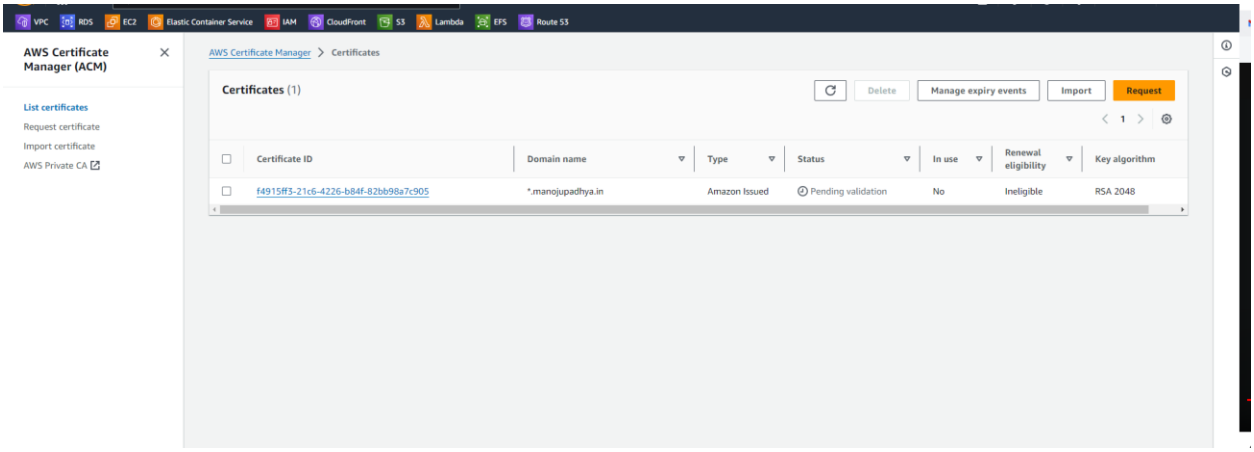
Key algorithm Info

Select an encryption algorithm. Some algorithms may not be supported by all AWS services.

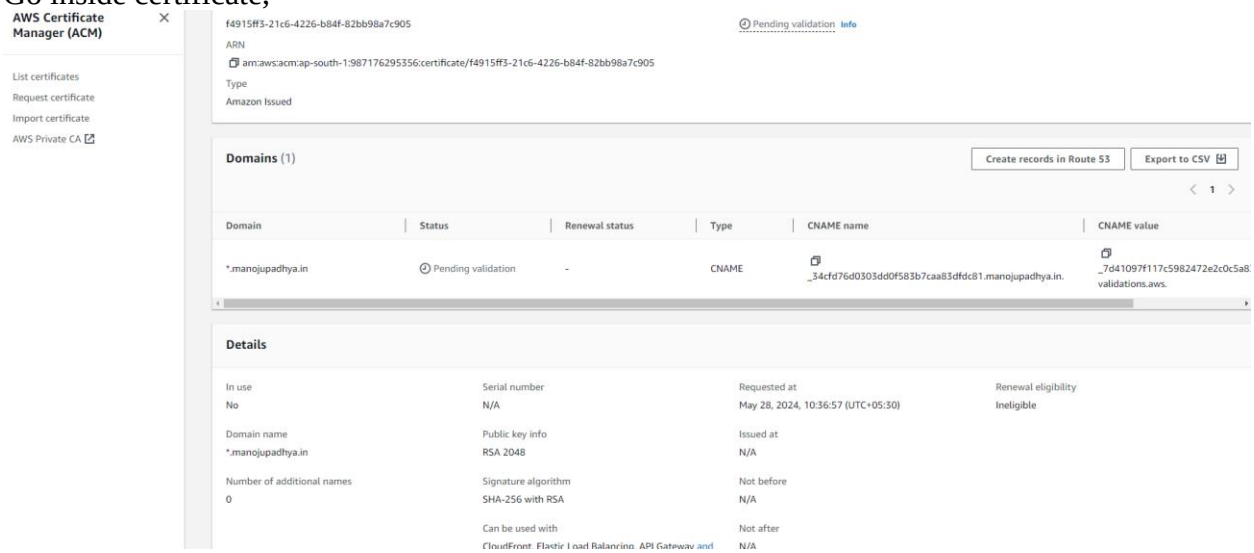
- (●) RSA 2048
  RSA is the most widely used key type.
- ( ) ECDSA P 256
  Equivalent in cryptographic strength to RSA 3072.
- ( ) ECDSA P 384
  Equivalent in cryptographic strength to RSA 7680.



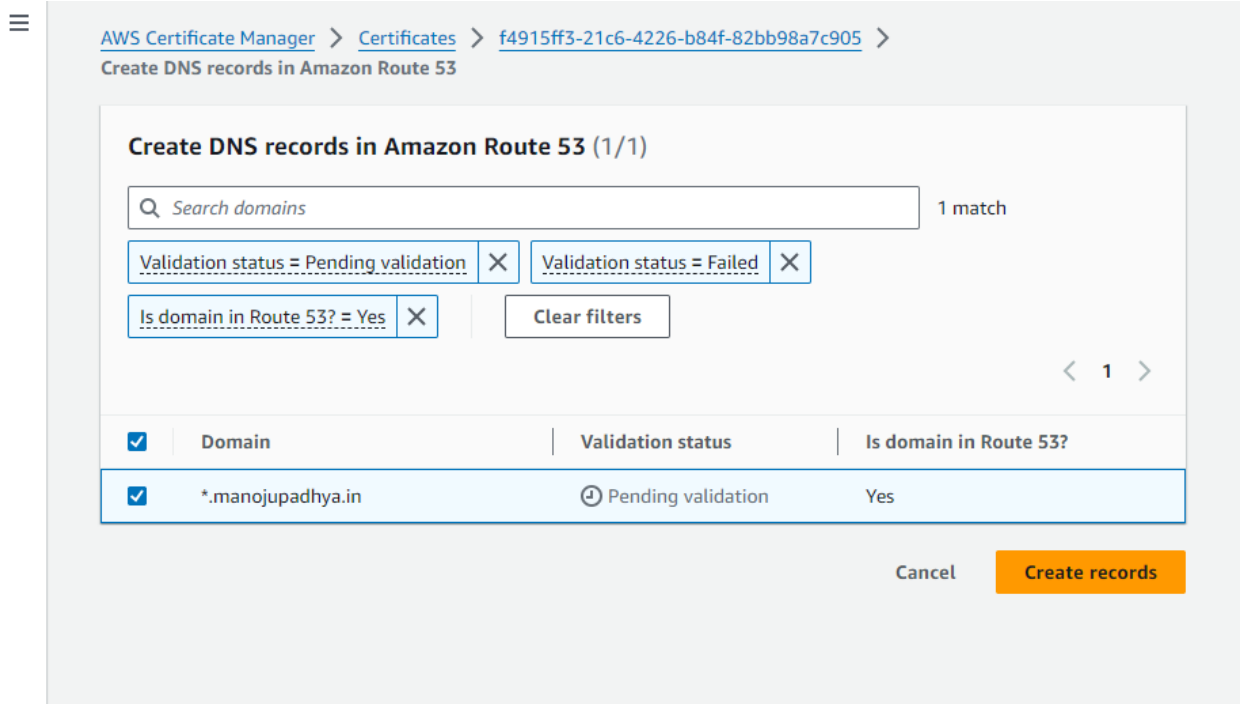
It will take some time to get certificate.

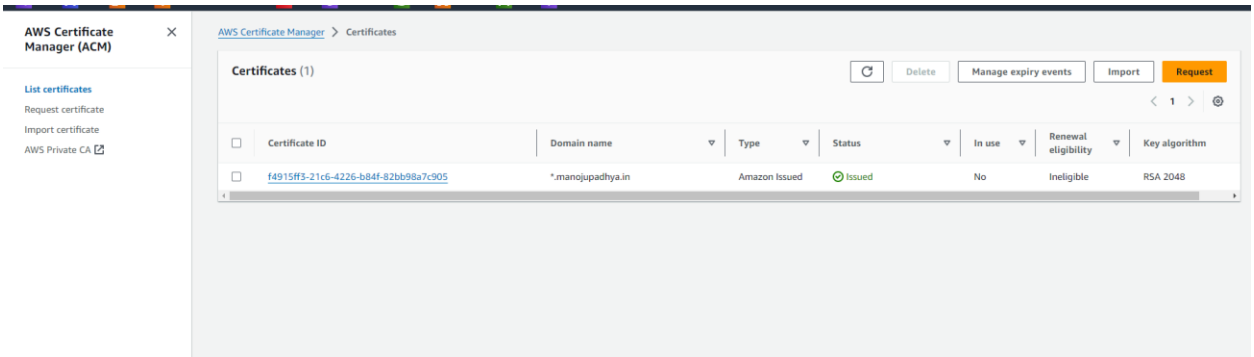


We need to validate the certificate by creating a DNS Record.
Go inside certificate,



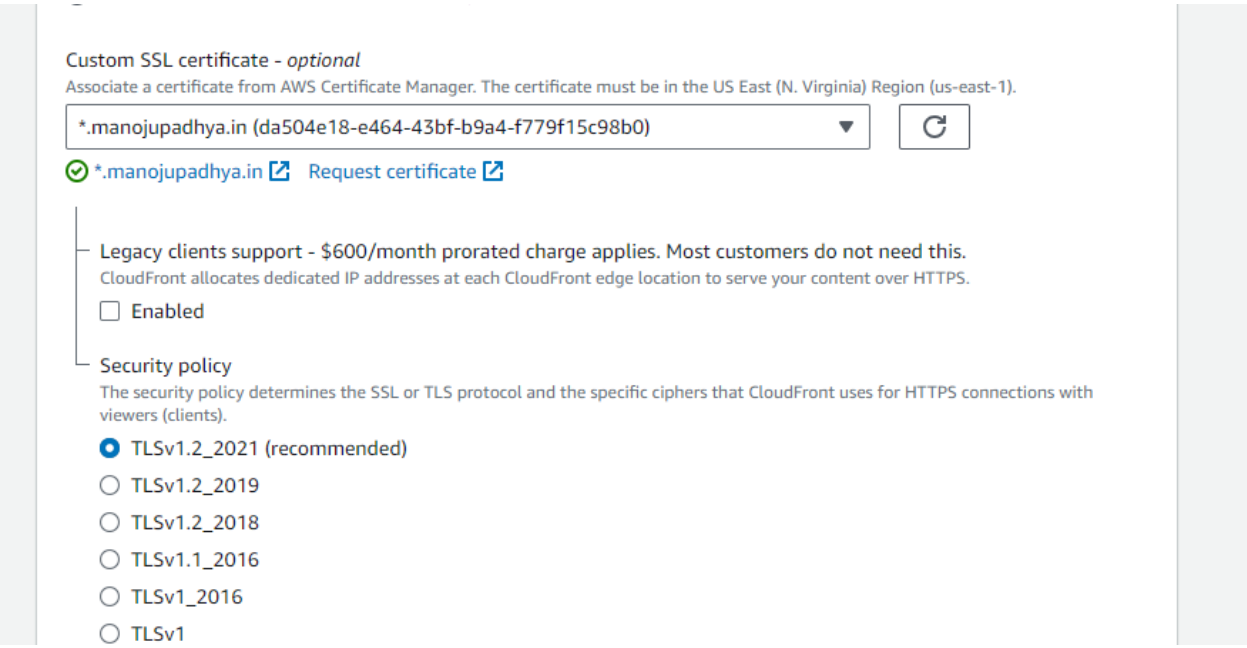
Click on create a record in Route53.





Now status changed to issued.

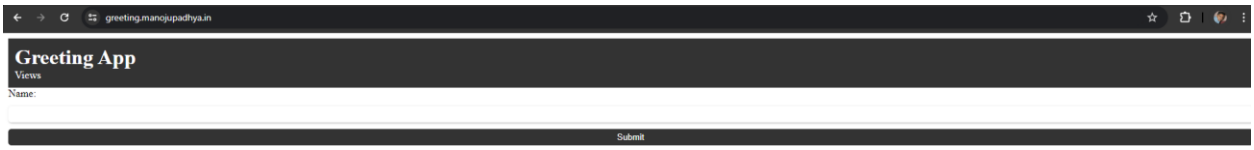
Select the certificate from the cloud front distribution page.



Click on Save changes.

We need to create a record for our domain that is for (**greetings**.manojupadhya.in) Navigate to Route 53 to create Record.

Add greeting.manojupadhya.in to direct to CloudFront distribution.

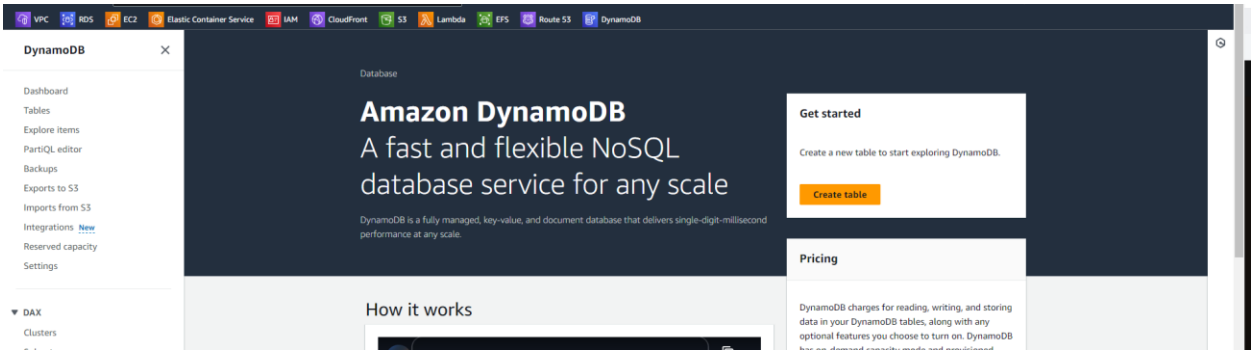


Amazon DynamoDB

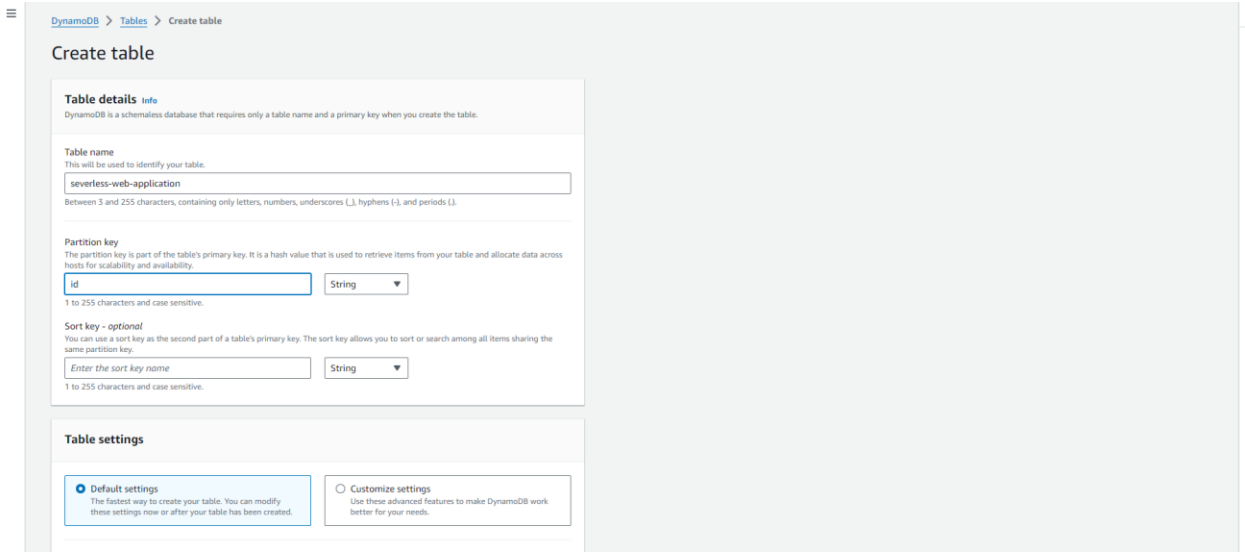
Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

Setup Steps:

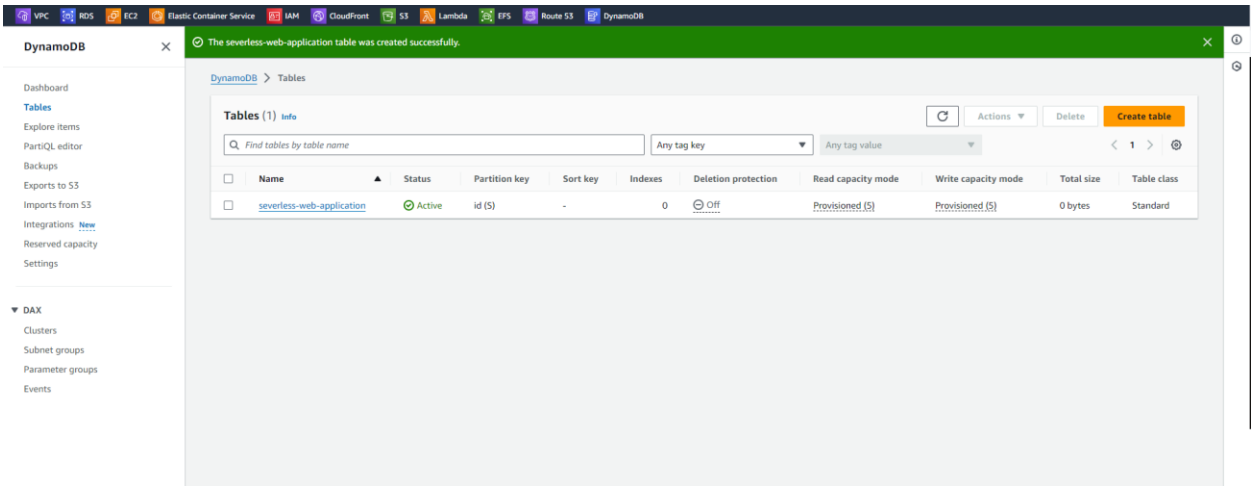
Create a DynamoDB table to store application data.
Navigate to DynamoDB from management console.



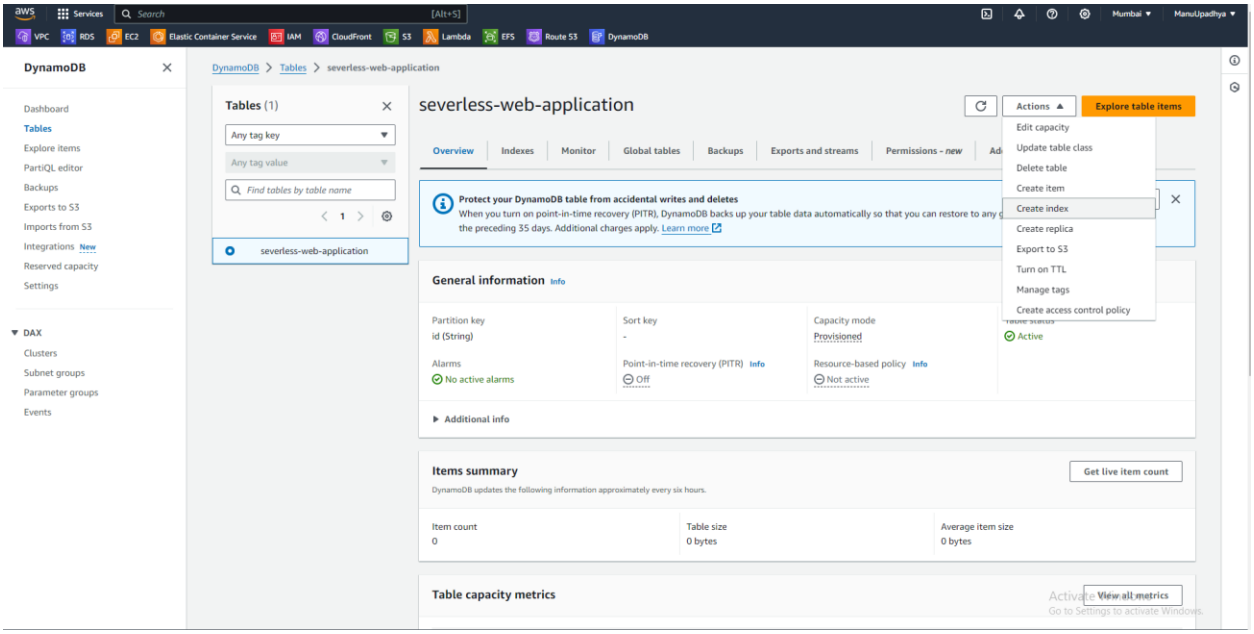
Click on the create table.

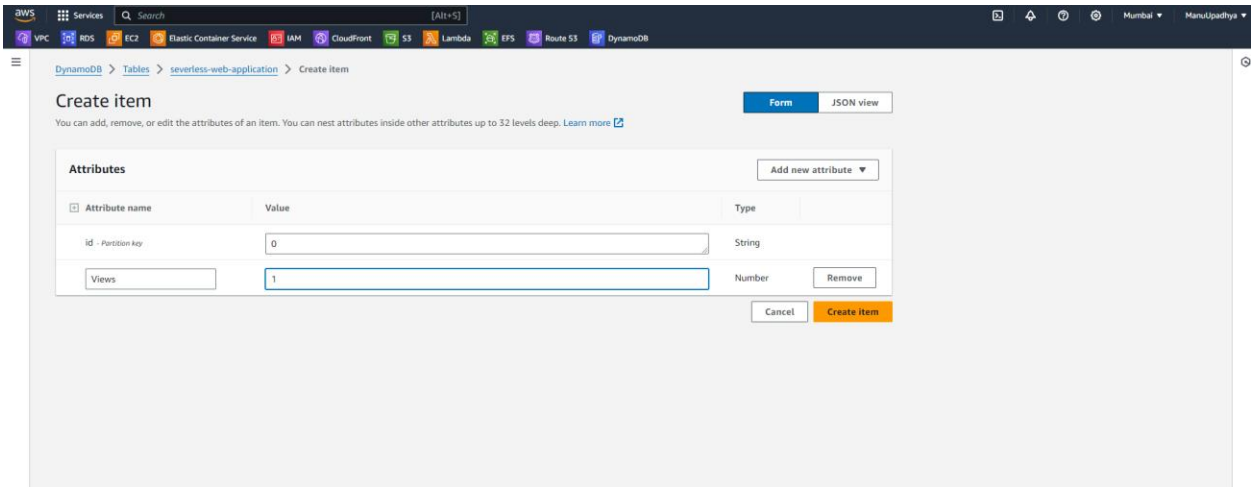


Rest are default click on create.

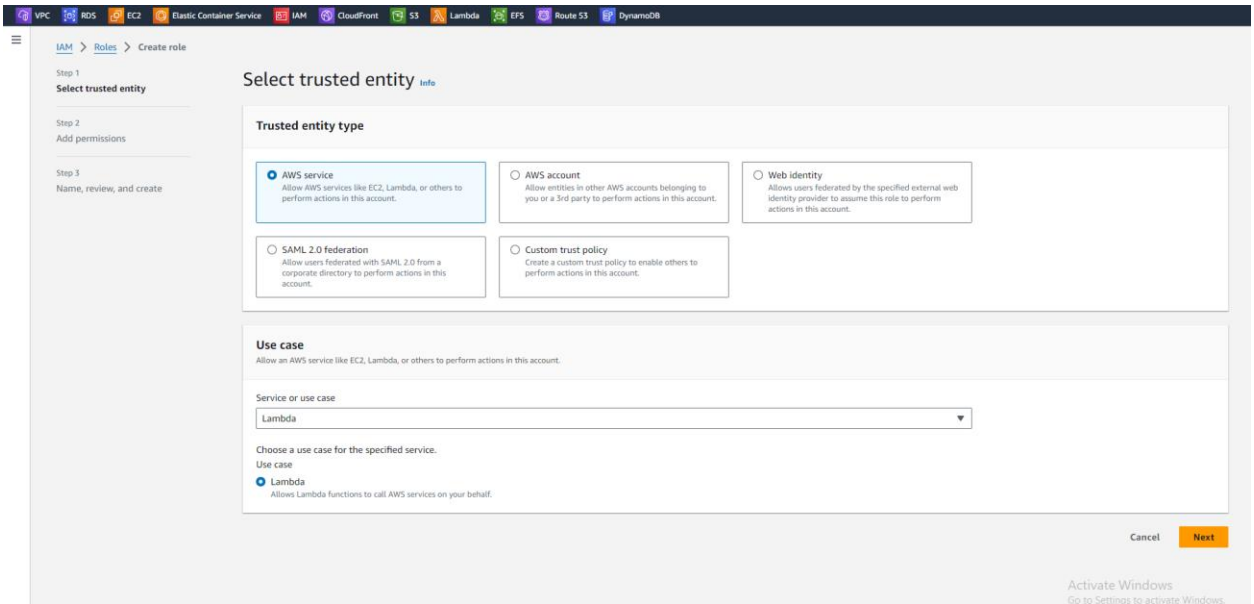


Now navigate inside the table→Actions→ Create Item.

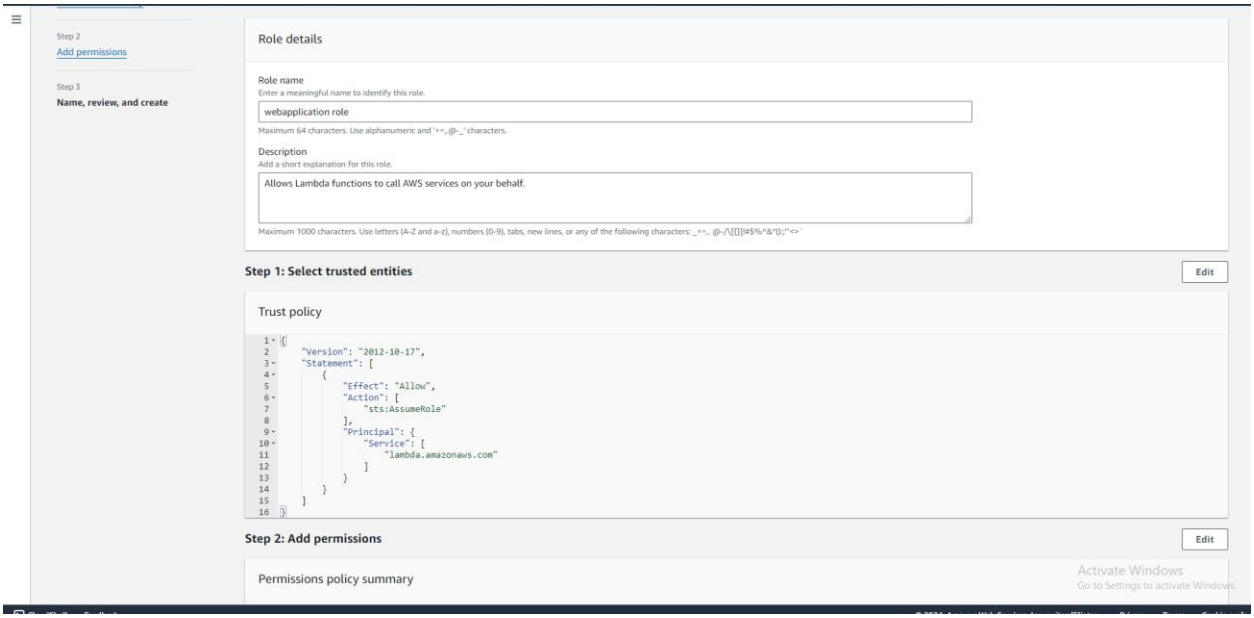




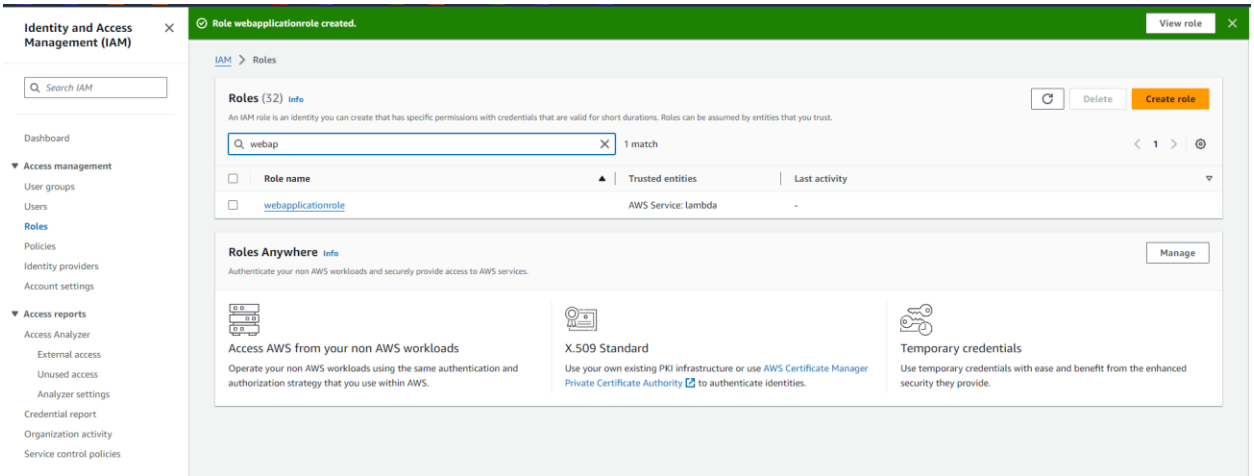
Now we will create an IAM Role with access to Lambda functions.



Click on next



Created role.



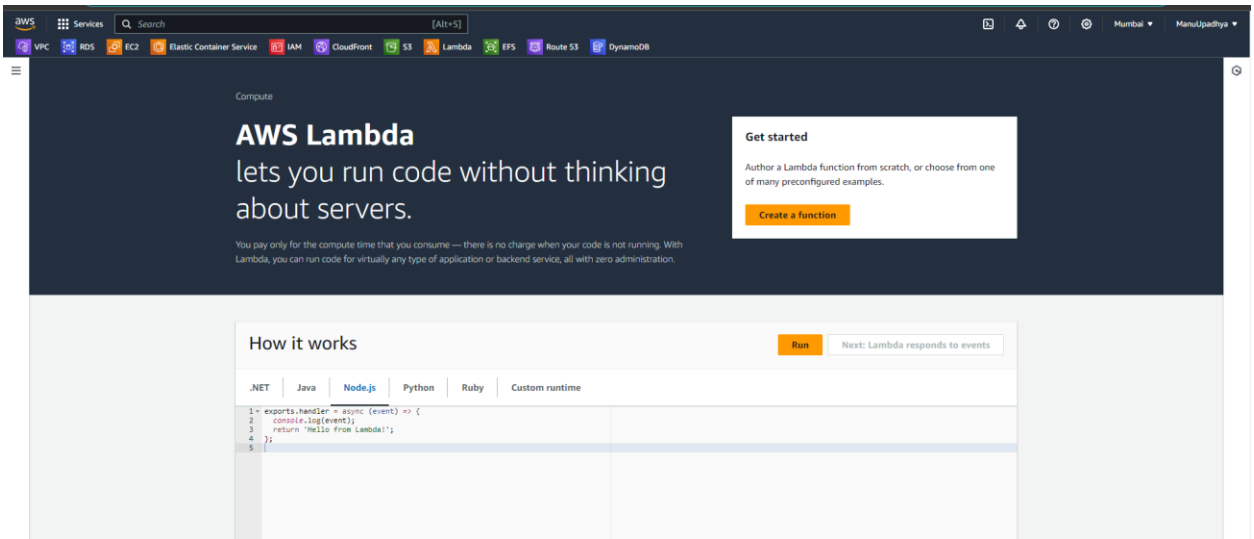
AWS Lambda

AWS Lambda allows running code without provisioning or managing servers. It executes backend logic in response to events, such as HTTP requests.

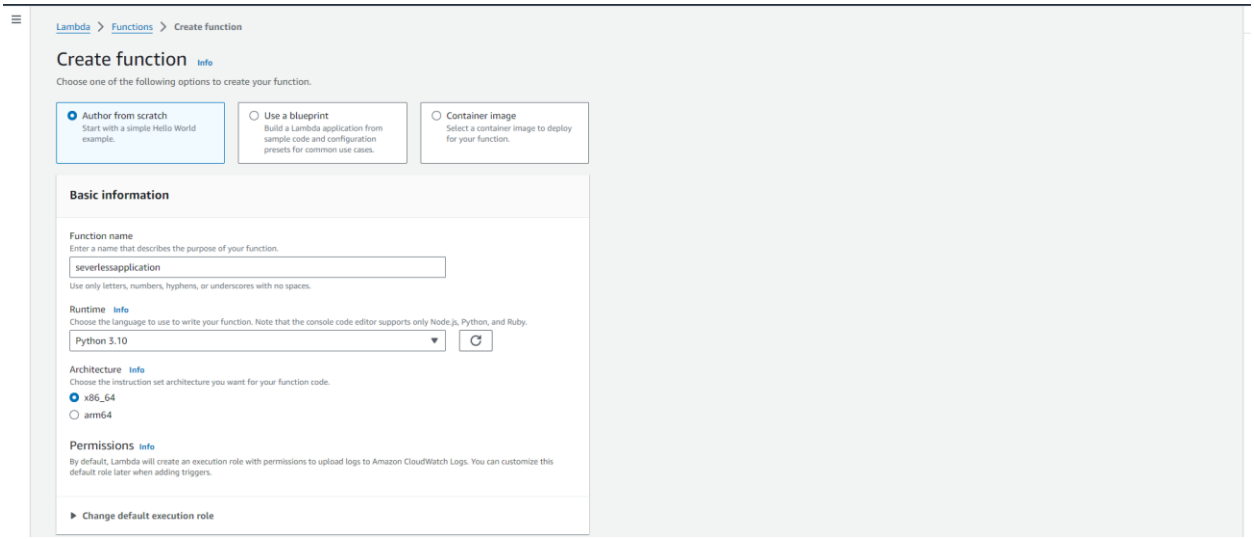
Setup Steps:

Write and test Lambda functions. Integrate Lambda functions with DynamoDB to perform view operations.

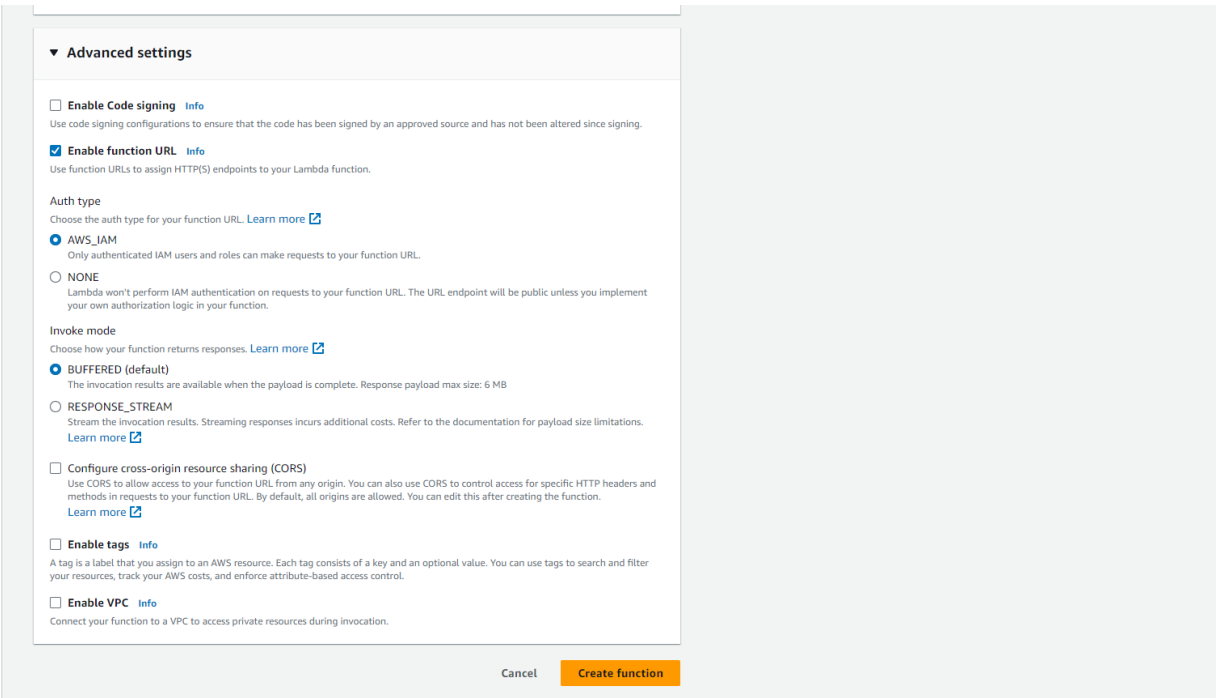
Navigate to AWS Lambda



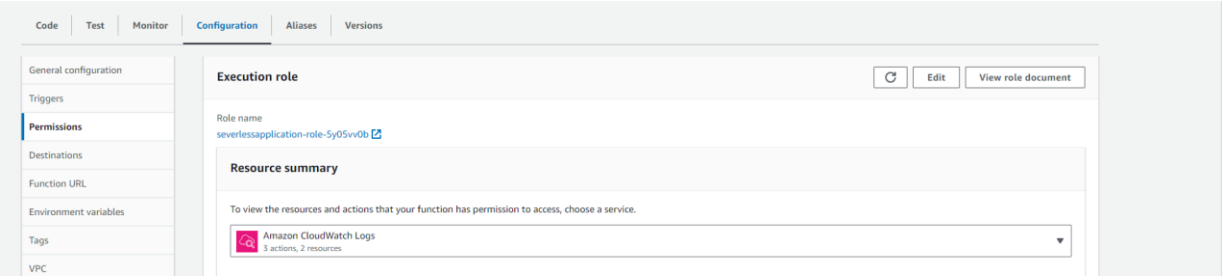
Create function



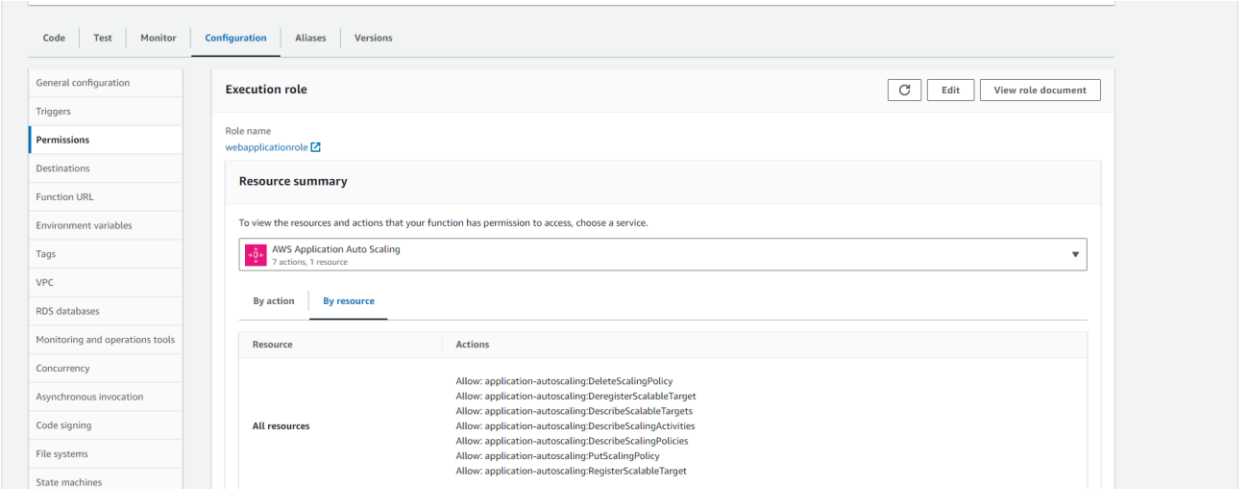
Advanced settings enable function URL.



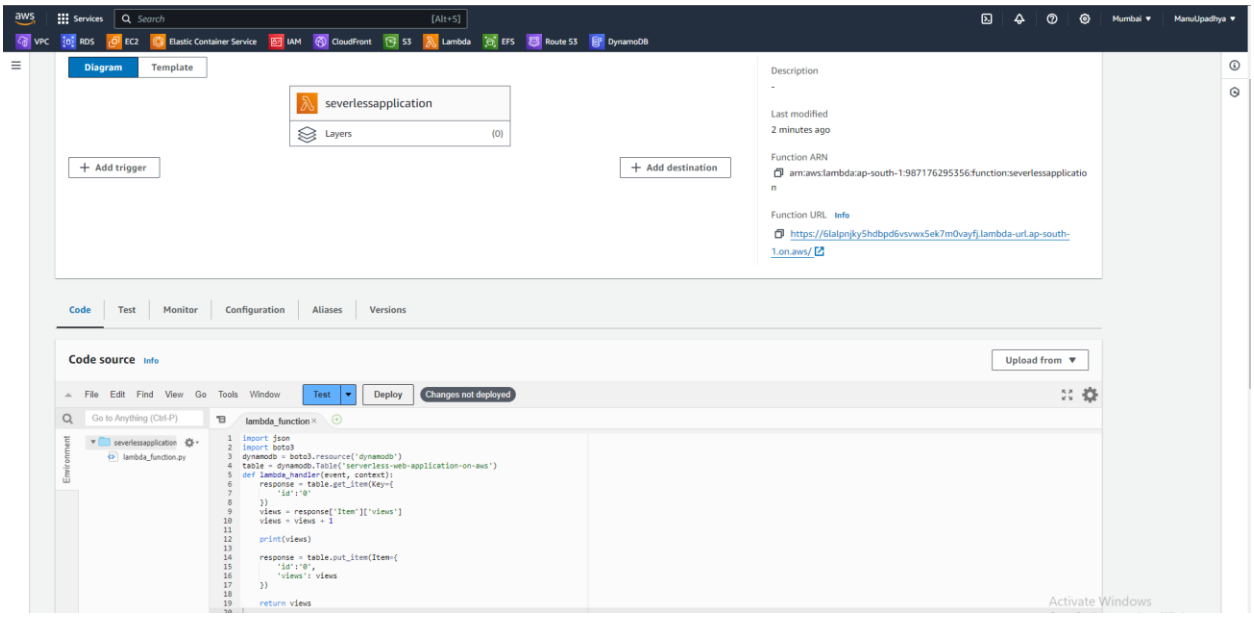
Attach created IAM role to this function by navigating to configurations→Permissions



Select our role which is required for the project.

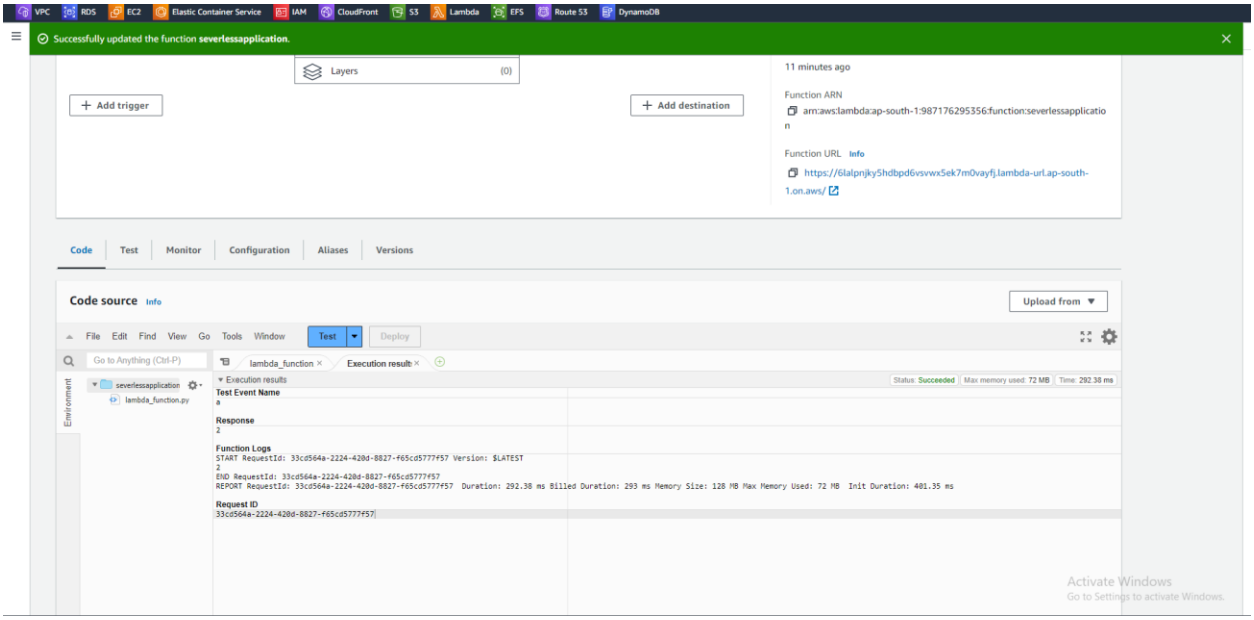


Adding code for lambda function.

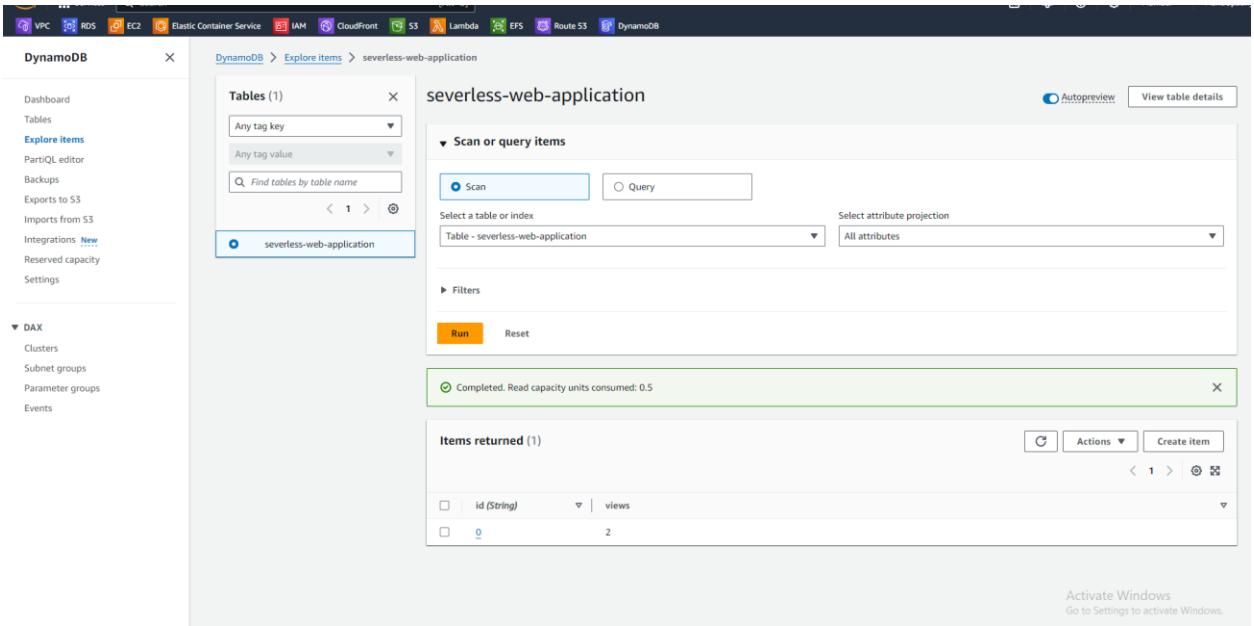


Deploy and test it again.

When test the lambda function view must be incremented as below: -



The updated view count in DynamoDB table.



Code for Lambda function is uploaded, we need function to calculate views or website visits count.

Our work is almost finished…….

Now changes in website code to get view counter to display in web page.

```javascript
const form = document.querySelector('form');
const greeting = document.querySelector('#greeting');

form.addEventListener('submit', (event) => {
    event.preventDefault();
    const name = document.querySelector('#name').value;
    greeting.textContent = `Hello, ${name}!`;
});

const counter = document.querySelector(".counter-number");
async function updateCounter() {
    let response = await fetch(
        "https://6lalpnjky5hdbpd6vsvwx5ek7m0vayfj.lambda-url.ap-south-1.on.aws/"
    );
    let data = await response.json();
    counter.innerHTML = `Views: ${data}`;
}
updateCounter();
```

Add the lamda function URL in script.js file.

Do Not forget to update the code file in S3 bucket, upload these files again….

TESTING…….

Open the DNS name of the Website…..

4. Deployment Process

Step 1: Set Up S3 Bucket

Create an S3 bucket.

Enable static website hosting.

Upload static files (index.html, style.css, app.js).

Step 2: Configure CloudFront Distribution

Create a CloudFront distribution.

Set the origin to the S3 bucket.

Configure caching behavior and distribution settings.

Step 3: Set Up Route 53

Register or use an existing domain.

Create a hosted zone.

Add DNS records pointing to the CloudFront distribution.

Step 4: Develop and Deploy Lambda Functions

Write backend code for Lambda.

Deploy Lambda functions using the AWS Management Console or CLI.

Set up API Gateway to expose Lambda endpoints.

Step 5: Integrate with DynamoDB

Create a DynamoDB table.

Write Lambda functions to interact with DynamoDB (CRUD operations).

Ensure proper IAM roles and permissions.

5. Security and Best Practices

Security

Use IAM roles and policies to restrict access.

Enable encryption for S3 and DynamoDB.

Implement WAF (Web Application Firewall) for CloudFront.

Use HTTPS for secure data transmission.

Best Practices

Optimize Lambda function performance.

Implement error handling and logging.

Monitor application using AWS CloudWatch.

Automated deployment with AWS SAM or CloudFormation.

6. Cost Management

Use AWS Cost Explorer to monitor usage.
Set up billing alerts and budgets.
Optimize resource usage (e.g., Lambda memory allocation, DynamoDB read/write capacities).

7. Conclusion

Deploying a serverless web application on AWS offers scalability, high availability, and cost efficiency. By leveraging services like S3, CloudFront, Route 53, Lambda, and DynamoDB, you can build robust applications with minimal management overhead. Proper planning and adherence to best practices ensure secure, efficient, and cost-effective deployments.