

Deployment of Serverless Application on AWS

Table of Contents

1. Introduction
2. Architecture Overview
3. Components and Services
 - Amazon S3
 - Amazon CloudFront
 - Amazon Route 53
 - AWS Lambda
 - Amazon DynamoDB
4. Deployment Process
5. Security and Best Practices
6. Cost Management
7. Conclusion

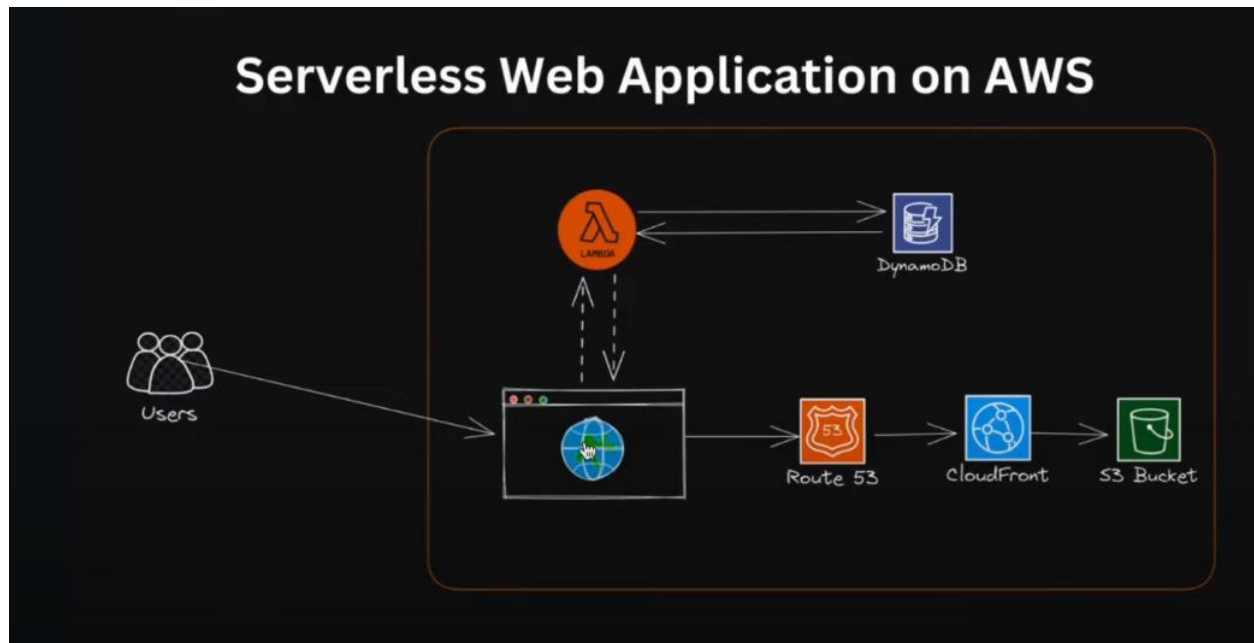
Project Description: -

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.

Create bucket [info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket name [info](#)
ManuServerless

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

Object Ownership [info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ **ACLs disabled (recommended)**
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ **ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☒ **Block all public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- ☒ **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☒ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- ☒ **Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☒ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning
☒ **Disable**
☐ **Enable**

Tags - optional (0)
You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

Activate Windows
Go to Settings to activate Windows.

Add tag

Default encryption [Info](#)
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)

- ☒ Server-side encryption with Amazon S3 managed keys (SSE-S3)
- ☐ Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- ☐ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing on the Amazon S3 pricing page](#).

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

- ☐ Disable
- ☒ Enable

► **Advanced settings**

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel **Create bucket**

Activate Windows
Go to Settings to activate Windows.

Click on Create Bucket.

Successfully created bucket "manuserverlessproject"
To upload files and folders, or to configure additional bucket settings, choose [View details](#).

Amazon S3 > Buckets

► **Account snapshot** - updated every 24 hours [All AWS Regions](#)
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[View Storage Lens dashboard](#)

General purpose buckets | Directory buckets

General purpose buckets (1) [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

Find buckets by name

Name	AWS Region	IAM Access Analyzer	Creation date
manuserverlessproject	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	May 28, 2024, 09:33:48 (UTC+05:30)

Copy ARN Empty Delete **Create bucket**

Upload static files to the bucket.

Amazon S3 > Buckets > manuserverlessproject

manuserverlessproject [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

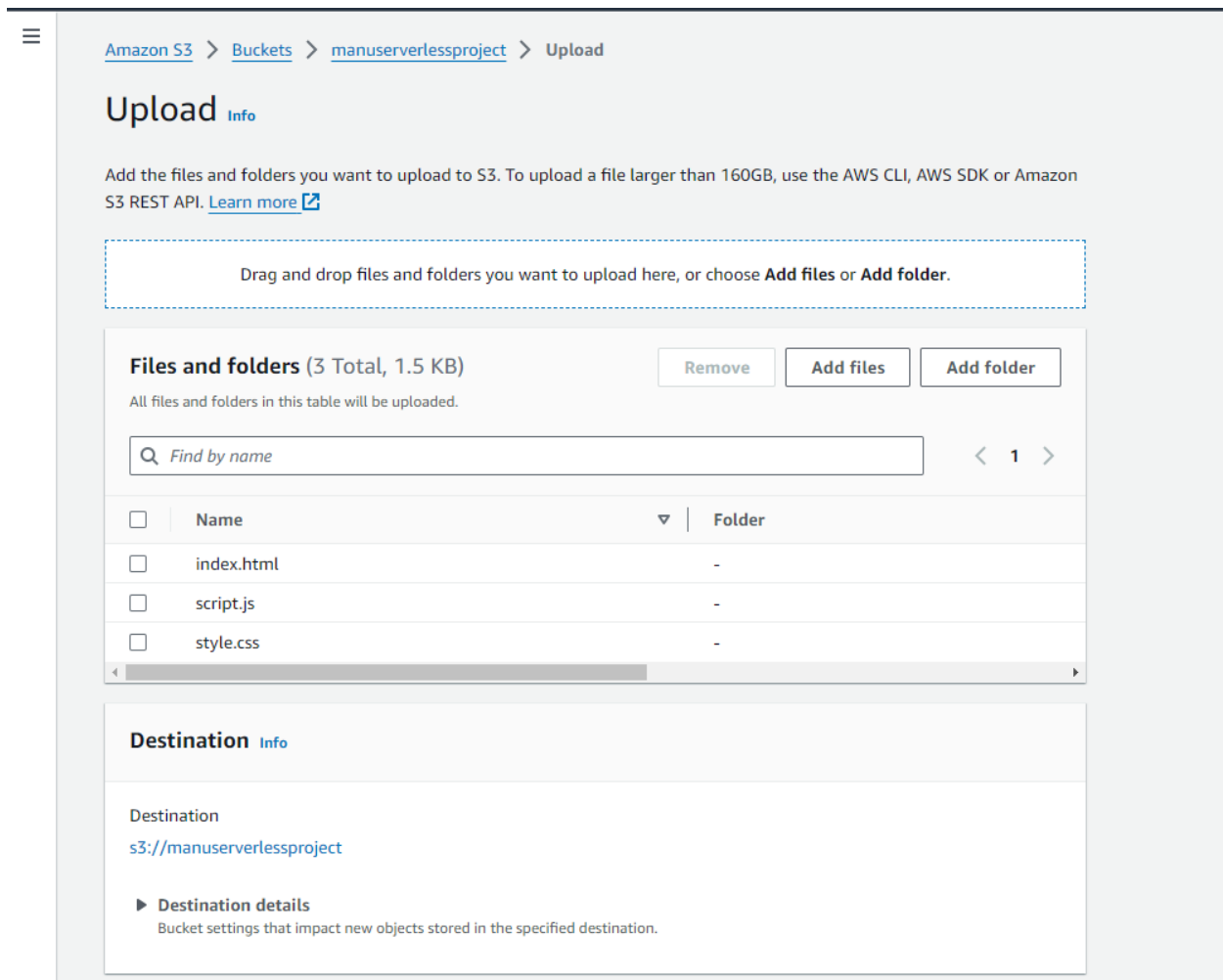
Objects (0) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

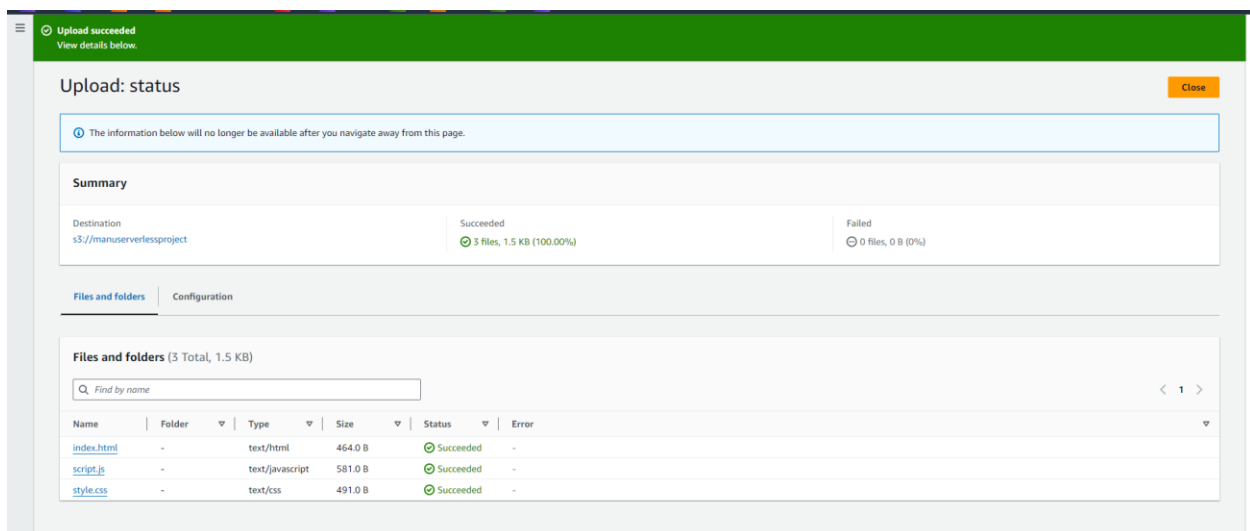
Find objects by prefix

Name	Type	Last modified	Size	Storage class
No objects You don't have any objects in this bucket.				

[Upload](#)



Click on Upload.



Configure the bucket to host a static website.

S3 bucket → properties.

The screenshot shows the 'Properties' tab of an Amazon S3 bucket. It contains several sections, each with a title, a brief description, and an 'Edit' button. The sections are: 'Send notifications to Amazon EventBridge for all events in this bucket' (set to 'Off'), 'Transfer acceleration' (set to 'Disabled'), 'Object Lock' (set to 'Disabled'), 'Requester pays' (set to 'Disabled'), and 'Static website hosting' (set to 'Disabled'). Each section also includes a 'Learn more' link. At the bottom right, there is a watermark that says 'Activate Windows Go to Settings to activate Windows.'

Edit and enable static website hosting.

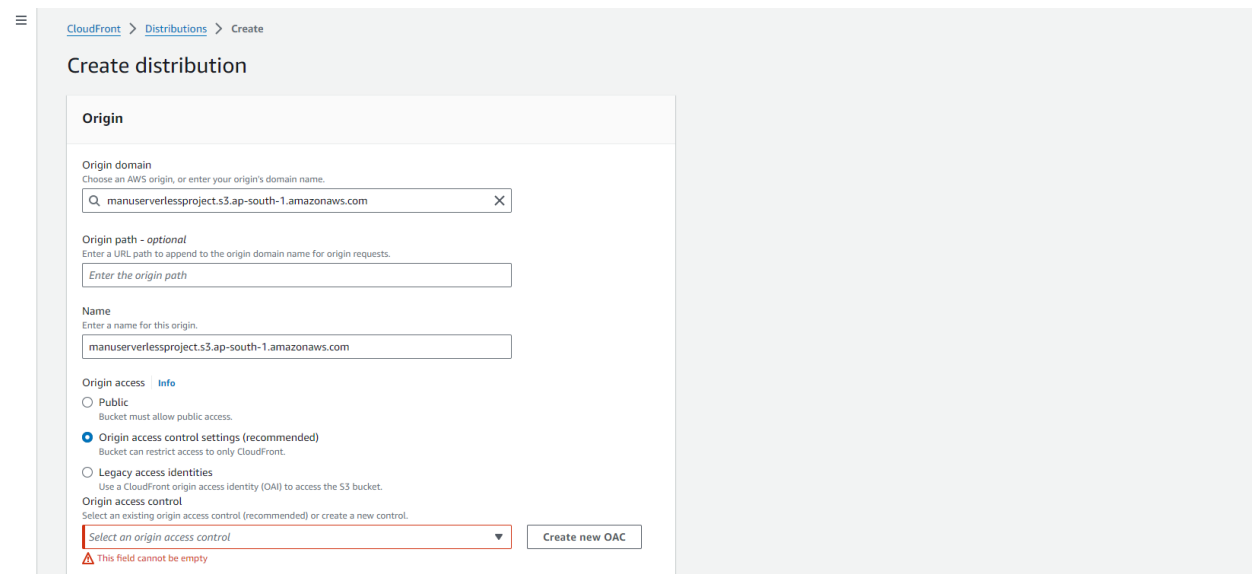
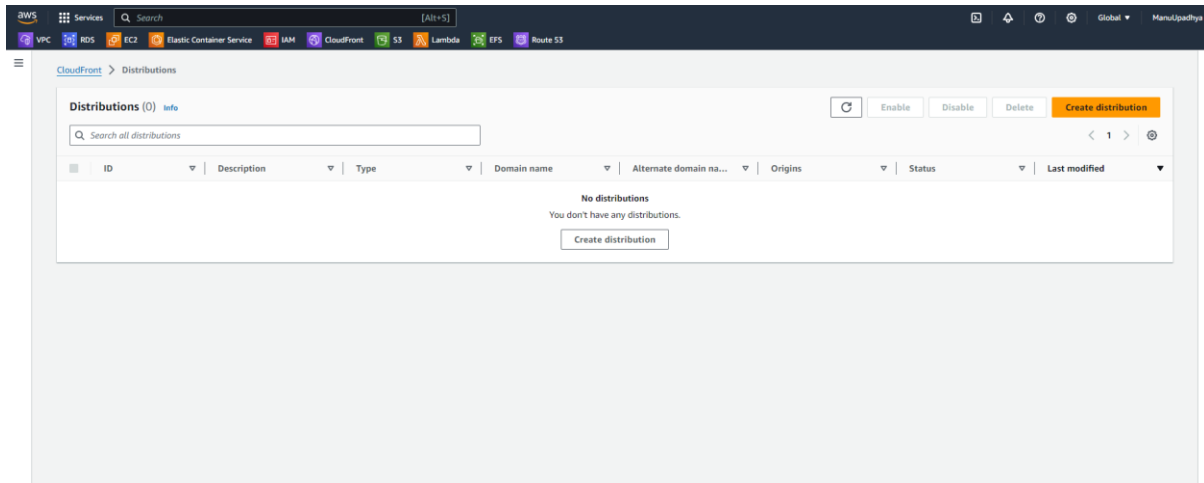
The screenshot shows the 'Edit static website hosting' page in the Amazon S3 console. The page has a breadcrumb trail: 'Amazon S3 > Buckets > manuserverlessproject > Edit static website hosting'. The main heading is 'Edit static website hosting' with 'info' links. Below this, there's a section 'Static website hosting' with a 'Learn more' link. The 'Static website hosting' section has two radio buttons: 'Disable' and 'Enable' (which is selected). Below this is the 'Hosting type' section with two radio buttons: 'Host a static website' (selected) and 'Redirect requests for an object'. The 'Host a static website' option has a sub-section with a note: 'For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access'. Below this, there are two text input fields: 'Index document' (with 'index.html' entered) and 'Error document - optional' (with 'error.html' entered). At the bottom, there's a section for 'Redirection rules - optional' with a 'Learn more' link.

Amazon CloudFront

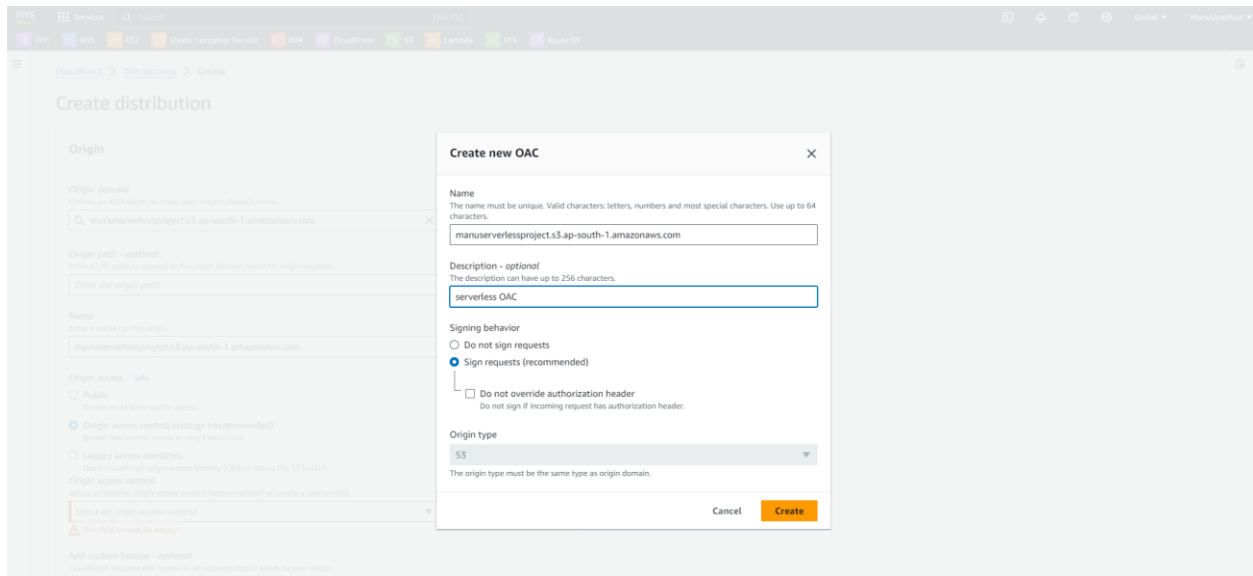
Amazon CloudFront is a content delivery network (CDN) that caches content at edge locations to reduce latency and improve load times for users.

Setup Steps:

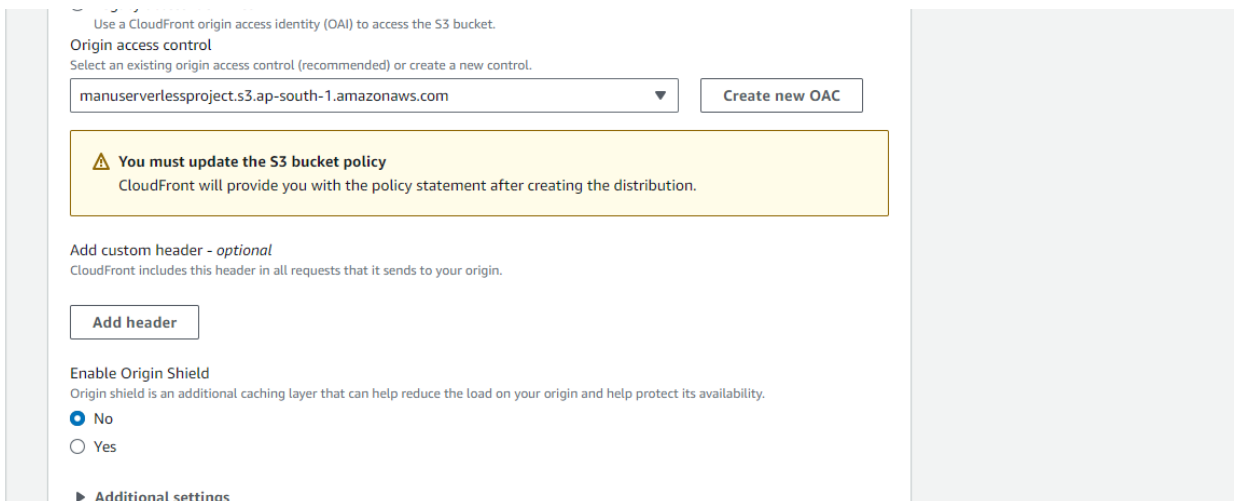
Create a CloudFront distribution → Specify the S3 bucket as the origin.



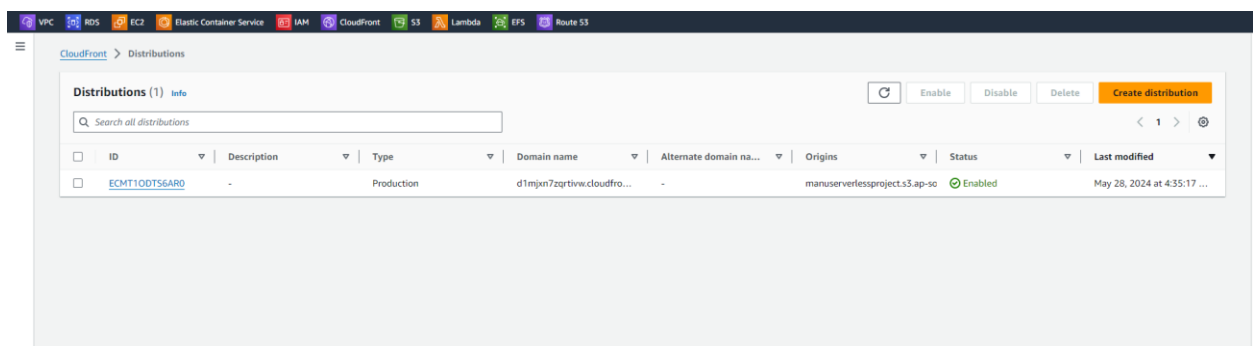
Create a new OAC



Click on create, select the created OAC.

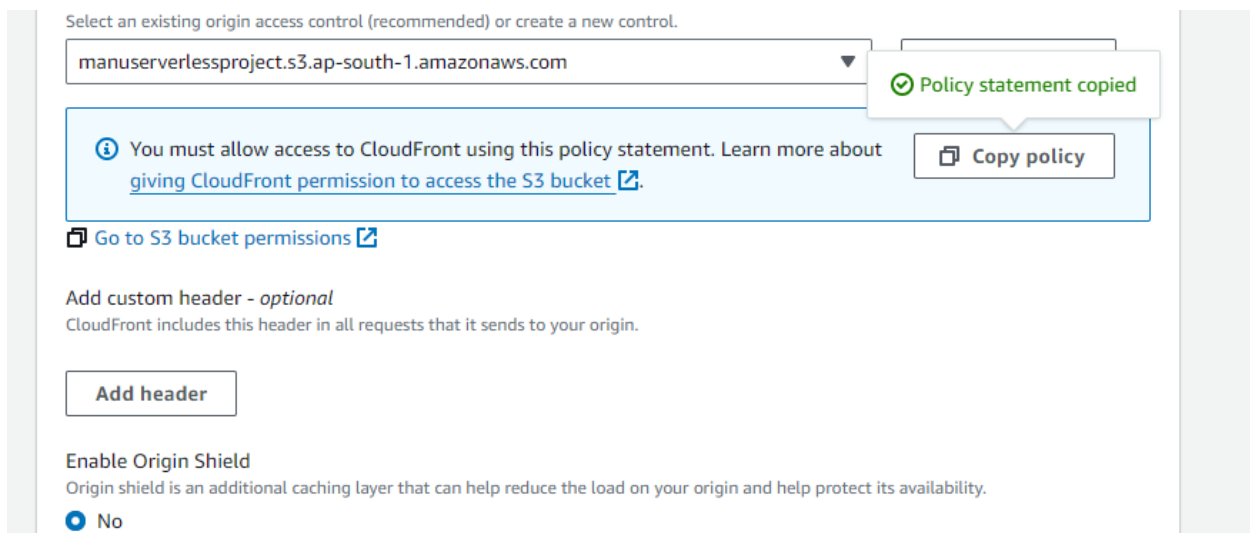
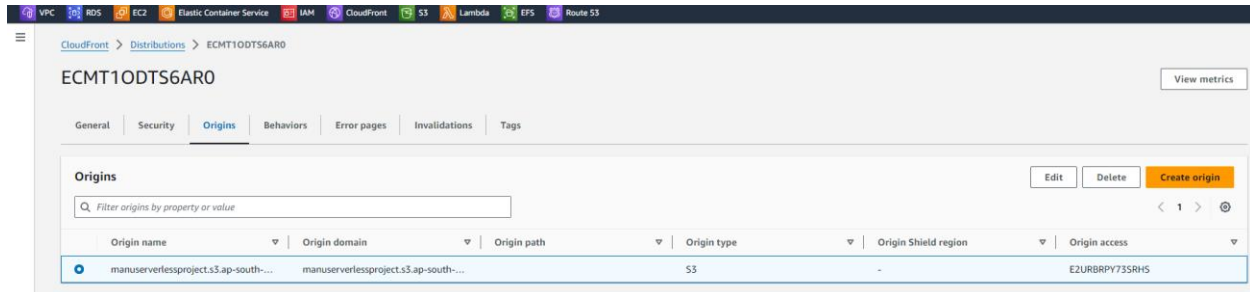


Leave everything default and click on create Distribution.



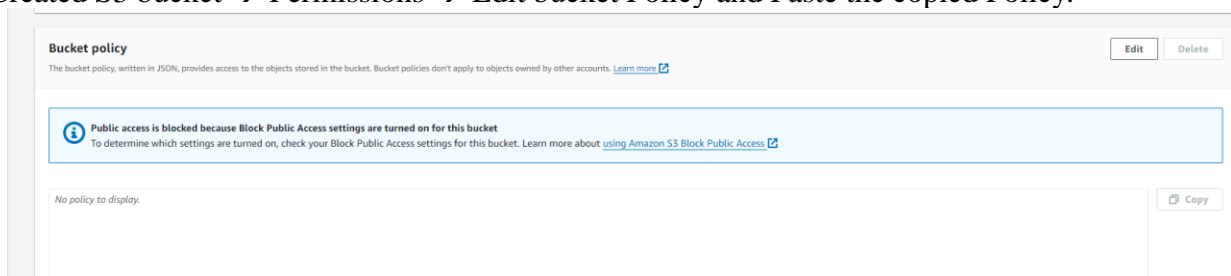
Configure caching policies and distribution settings.

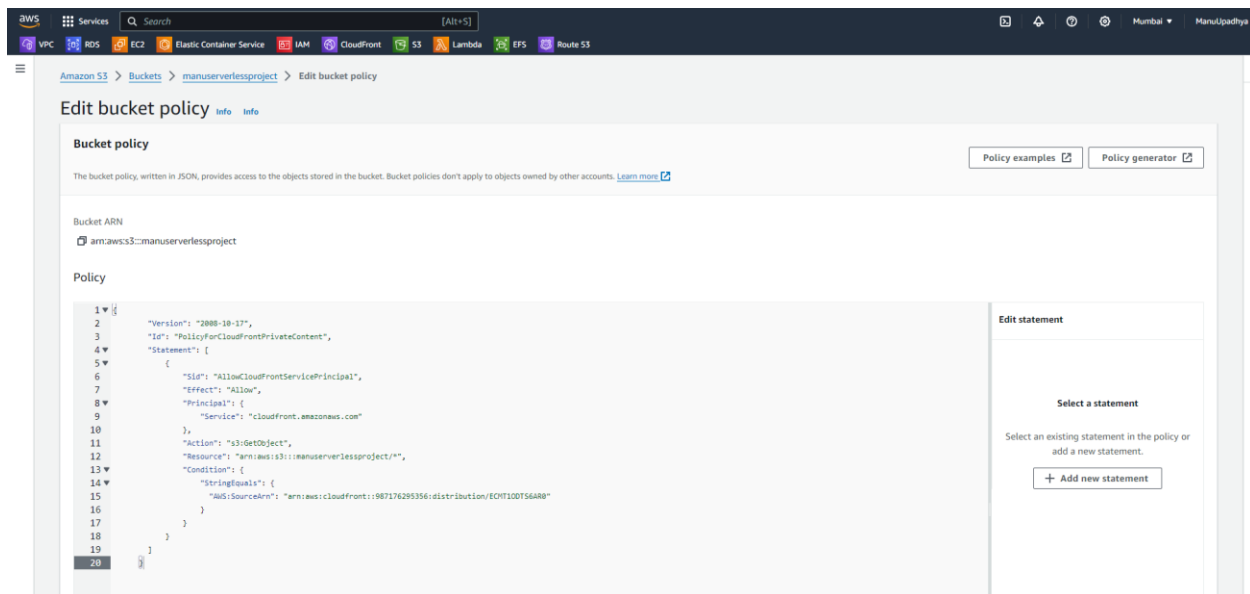
Get inside the distribution → origins → Select the created origin, click on edit.



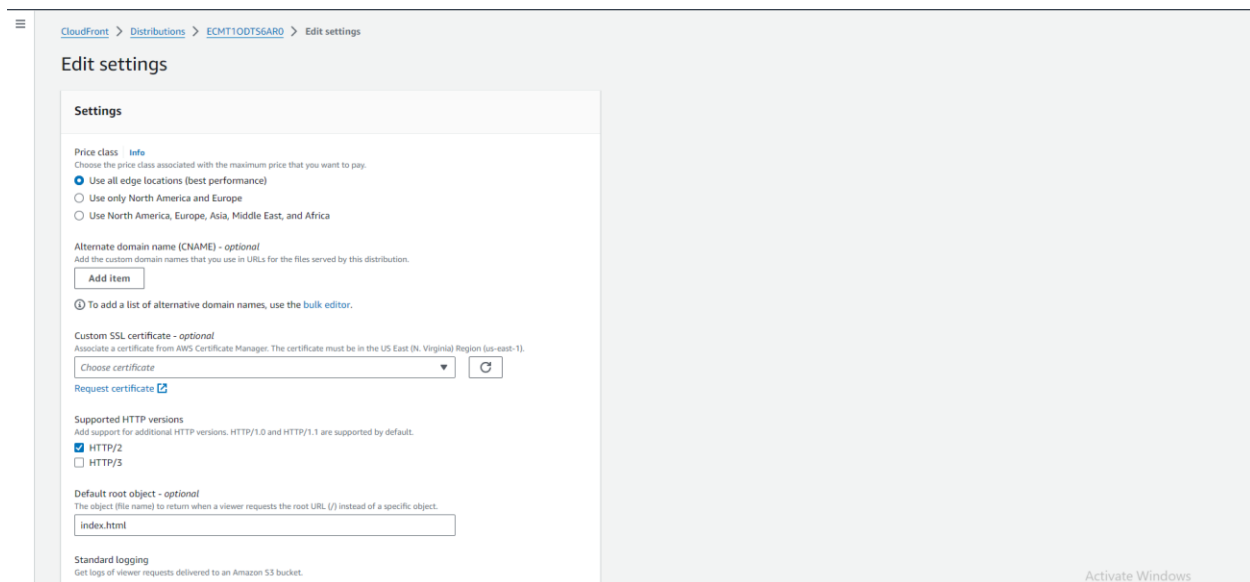
Click on copy policy. We need to paste this in our Amazon S3 bucket.

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.

manojupadhyaya11@gmail.com

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](#)

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.

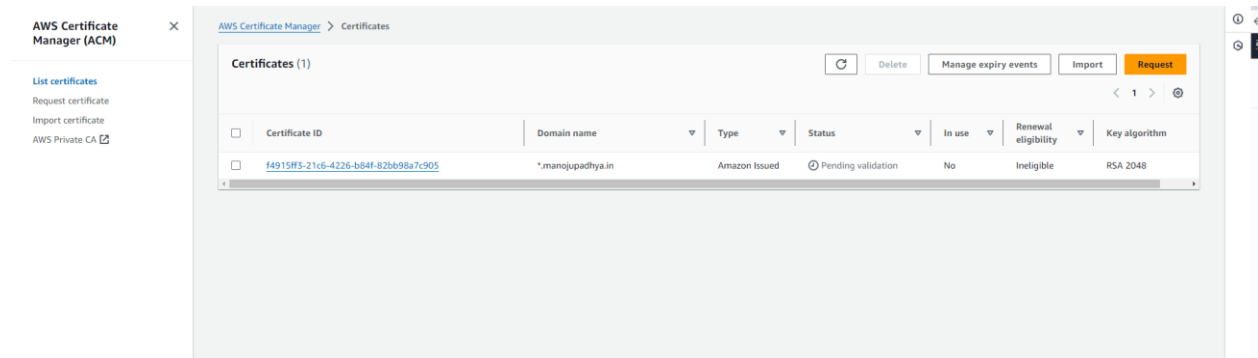
☐ ECDSA P 256

Equivalent in cryptographic strength to RSA 3072.

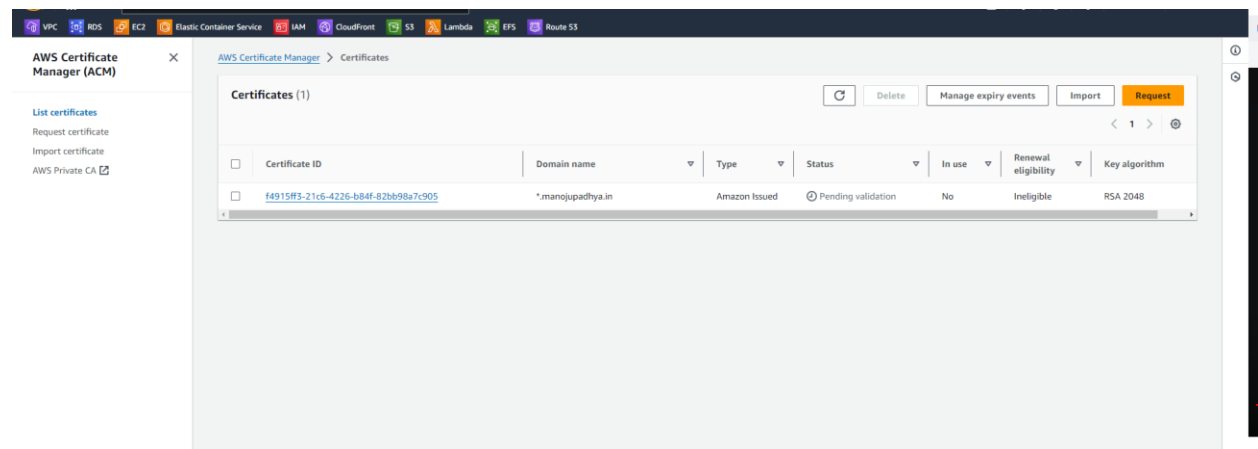
☐ ECDSA P 384

Equivalent in cryptographic strength to RSA 7680.

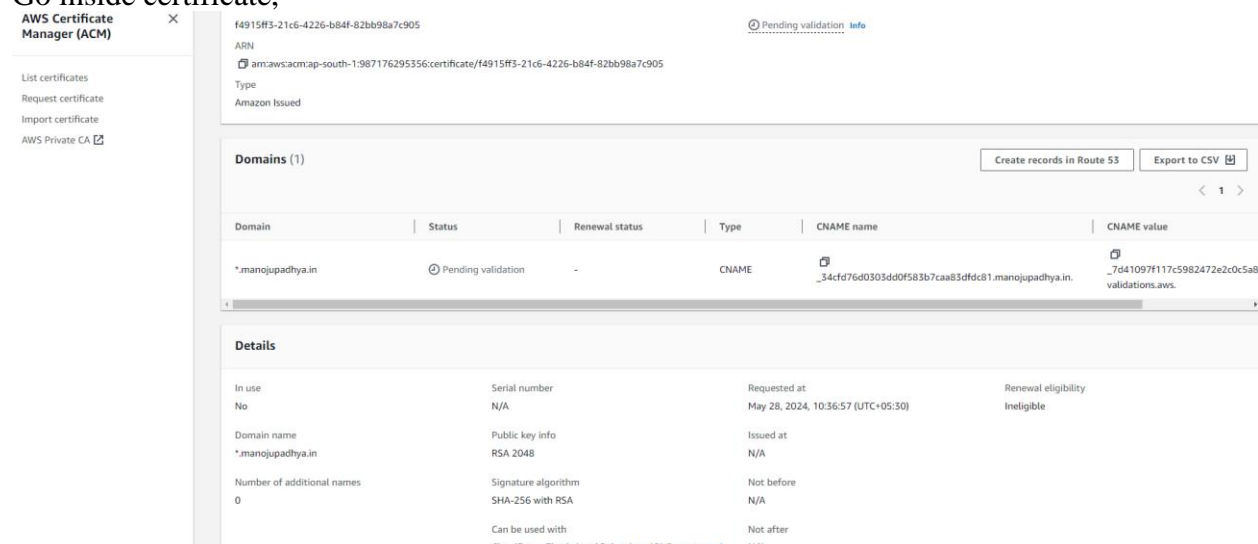
Manoj Upadhya



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.

☰

[AWS Certificate Manager](#) > [Certificates](#) > [f4915ff3-21c6-4226-b84f-82bb98a7c905](#) >

Create DNS records in Amazon Route 53

Create DNS records in Amazon Route 53 (1/1)

1 match

Validation status = Pending validation ✕

Validation status = Failed ✕

Is domain in Route 53? = Yes ✕

Clear filters

< 1 >

<input checked="" type="checkbox"/>	Domain	Validation status	Is domain in Route 53?
<input checked="" type="checkbox"/>	*.manojupadhya.in	⌚ Pending validation	Yes

Cancel **Create records**

AWS Certificate Manager (ACM) ✕

[List certificates](#)
[Request certificate](#)
[Import certificate](#)
[AWS Private CA](#)

[AWS Certificate Manager](#) > [Certificates](#)

Certificates (1) [Manage expiry events](#) [Import](#) [Request](#)


< 1 >




<input type="checkbox"/>	Certificate ID	Domain name	Type	Status	In use	Renewal eligibility	Key algorithm
<input type="checkbox"/>	f4915ff3-21c6-4226-b84f-82bb98a7c905	*.manojupadhya.in	Amazon Issued	Issued	No	Ineligible	RSA 2048

Now status changed to issued.

Select the certificate from the cloud front distribution page.

Custom SSL certificate - *optional*
Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

*.manojupadhya.in (da504e18-e464-43bf-b9a4-f779f15c98b0) 

 *.manojupadhya.in  [Request certificate](#) 

Legacy clients support - \$600/month prorated charge applies. Most customers do not need this.
CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS.

☐ Enabled

Security policy
The security policy determines the SSL or TLS protocol and the specific ciphers that CloudFront uses for HTTPS connections with viewers (clients).

☒ TLSv1.2_2021 (recommended)

☐ TLSv1.2_2019

☐ TLSv1.2_2018

☐ TLSv1.1_2016

☐ TLSv1_2016

☐ TLSv1

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.

← → ↻ ☰ greeting.manojupadhya.in ☆ ⌵ 🔍

Greeting App

Views

Name:

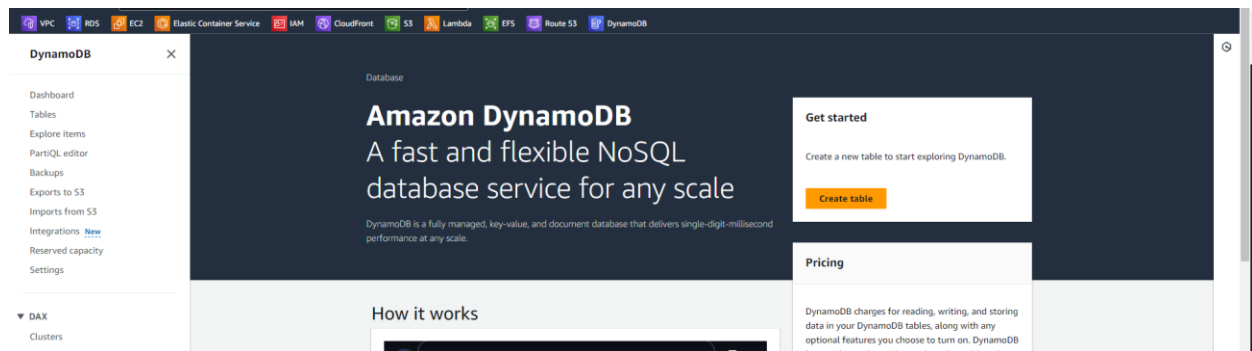
Amazon DynamoDB

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

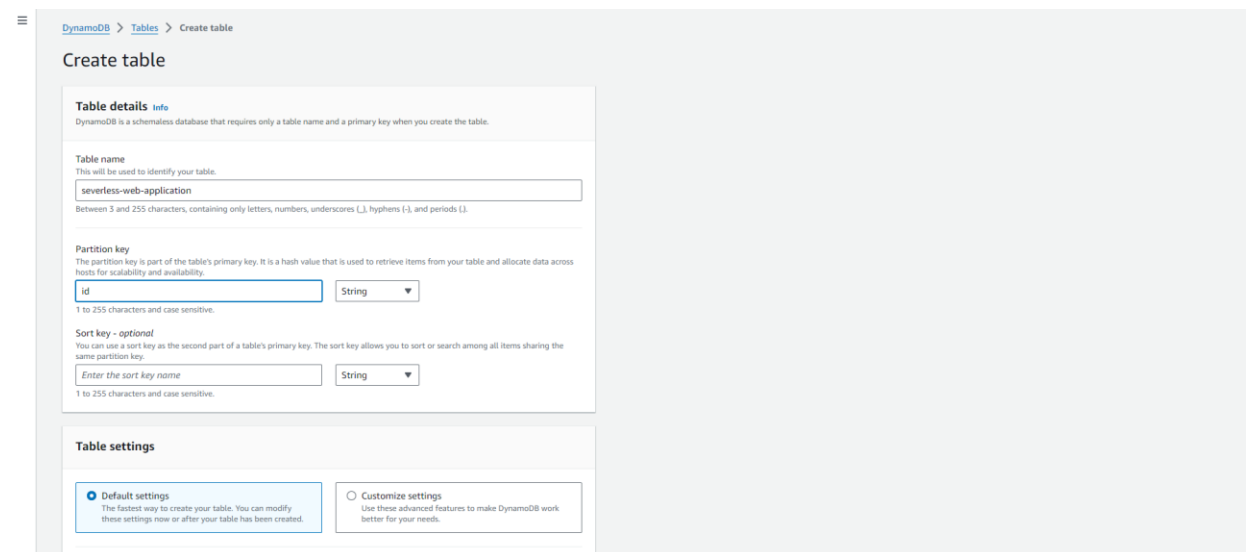
Setup Steps:

Create a DynamoDB table to store application data.

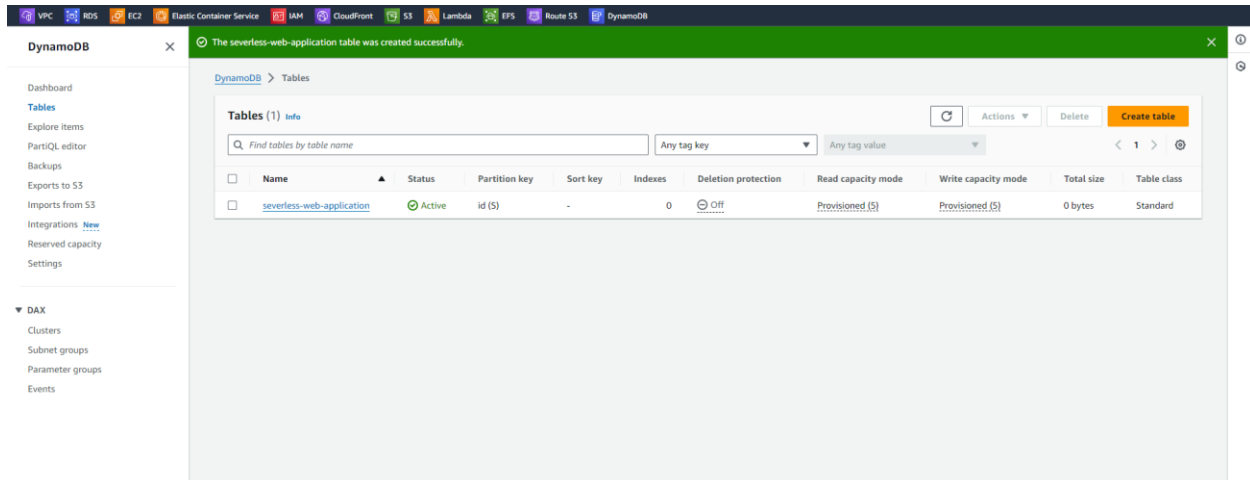
Navigate to DynamoDB from management console.



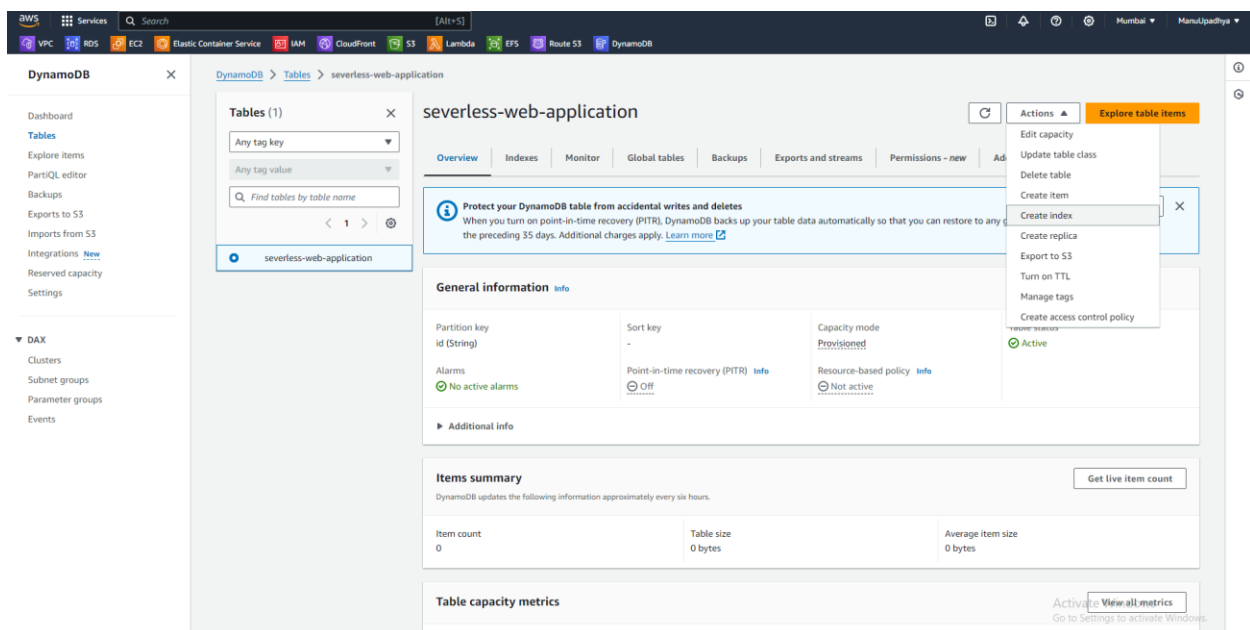
Click on the create table.



Rest are default click on create.



Now navigate inside the table → Actions → Create Item.



The screenshot shows the AWS IAM console's 'Create item' page for a DynamoDB table. The breadcrumb navigation is 'DynamoDB > Tables > severless-web-application > Create item'. There are tabs for 'Form' and 'JSON view'. A note states: 'You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)'. Below this is a table for 'Attributes' with columns 'Attribute name', 'Value', and 'Type'. The first row has 'id - Partition key' as the attribute name, '0' as the value, and 'String' as the type. The second row has 'Views' as the attribute name, '1' as the value, and 'Number' as the type. There is an 'Add new attribute' button and a 'Remove' button next to the 'Views' row. At the bottom are 'Cancel' and 'Create item' buttons.

Attribute name	Value	Type
id - Partition key	0	String
Views	1	Number

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

The screenshot shows the AWS IAM console's 'Select trusted entity' page. The breadcrumb navigation is 'IAM > Roles > Create role'. The left sidebar shows 'Step 1: Select trusted entity', 'Step 2: Add permissions', and 'Step 3: Name, review, and create'. The main heading is 'Select trusted entity [Info](#)'. Under 'Trusted entity type', there are five options: 'AWS service' (selected), 'AWS account', 'Web identity', 'SAML 2.0 Federation', and 'Custom trust policy'. Below this is the 'Use case' section, which says 'Allow an AWS service like EC2, Lambda, or others to perform actions in this account.' There is a 'Service or use case' dropdown menu with 'Lambda' selected. Below the dropdown, it says 'Choose a use case for the specified service.' and 'Use case' with 'Lambda' selected. At the bottom right are 'Cancel' and 'Next' buttons. A watermark 'Activate Windows' is visible at the bottom right.

Click on next

Step 2
[Add permissions](#)

Step 3
Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

webapplicationrole

Maximum 64 characters. Use alphanumeric and "+", "@", "-" characters.

Description

Add a short explanation for this role.

Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: "_", ":", "@", "/", "!", "#", "%", "&", "(", ")", "<", ">".

Step 1: Select trusted entities

Edit

Trust policy

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "sts:AssumeRole"
8       ],
9       "Principal": {
10        "Service": [
11          "lambda.amazonaws.com"
12        ]
13      }
14    }
15  ]
16 }
```

Step 2: Add permissions

Edit

Permissions policy summary

Activate Windows
Go to Settings to activate Windows.

Created role.

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

Access reports

- Access Analyzer
 - External access
 - Unused access
 - Analyzer settings
- Credential report
- Organization activity
- Service control policies

Role webapplicationrole created.

View role

IAM > Roles

Roles (32) info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Q webap

1 match

<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	webapplicationrole	AWS Service: lambda	-

Roles Anywhere info

Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads

Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.

X.509 Standard

Use your own existing PKI infrastructure or use [AWS Certificate Manager Private Certificate Authority](#) to authenticate identities.

Temporary credentials

Use temporary credentials with ease and benefit from the enhanced security they provide.

manojupadhyia11@gmail.com

19

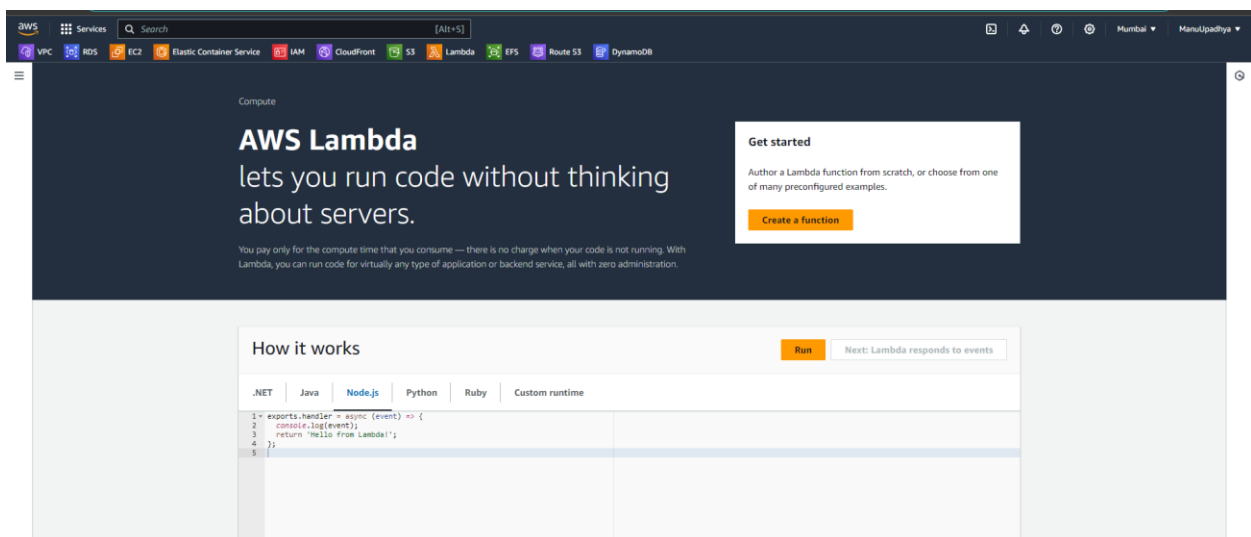
AWS Lambda

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

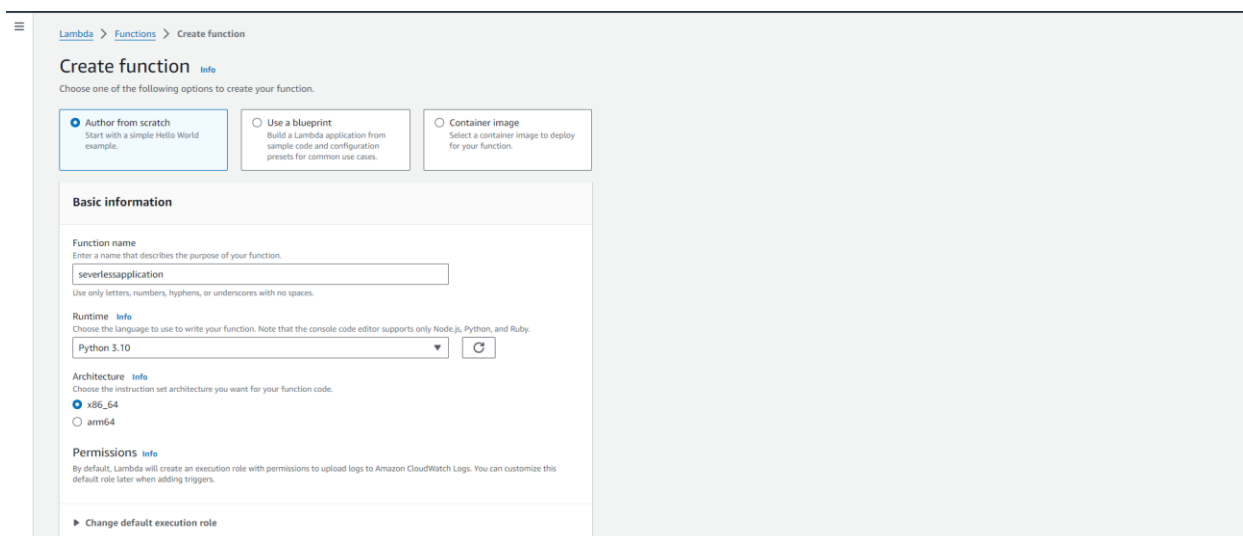
Setup Steps:

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.

manojupadhya11@gmail.com

▼ Advanced settings

☐ Enable Code signing [Info](#)
Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.

☒ Enable function URL [Info](#)
Use function URLs to assign HTTP(S) endpoints to your Lambda function.

Auth type
Choose the auth type for your function URL. [Learn more](#)

☒ AWS_IAM
Only authenticated IAM users and roles can make requests to your function URL.

☐ NONE
Lambda won't perform IAM authentication on requests to your function URL. The URL endpoint will be public unless you implement your own authorization logic in your function.

Invoke mode
Choose how your function returns responses. [Learn more](#)

☒ BUFFERED (default)
The invocation results are available when the payload is complete. Response payload max size: 6 MB

☐ RESPONSE_STREAM
Stream the invocation results. Streaming responses incurs additional costs. Refer to the documentation for payload size limitations. [Learn more](#)

☐ Configure cross-origin resource sharing (CORS)
Use CORS to allow access to your function URL from any origin. You can also use CORS to control access for specific HTTP headers and methods in requests to your function URL. By default, all origins are allowed. You can edit this after creating the function. [Learn more](#)

☐ Enable tags [Info](#)
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources, track your AWS costs, and enforce attribute-based access control.

☐ Enable VPC [Info](#)
Connect your function to a VPC to access private resources during invocation.

Cancel [Create function](#)

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

Code Test Monitor **Configuration** Aliases Versions

General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC

Execution role [Refresh](#) [Edit](#) [View role document](#)

Role name
severlessapplication-role-5y05vv0b [Link](#)

Resource summary

To view the resources and actions that your function has permission to access, choose a service.

Amazon CloudWatch Logs
3 actions, 2 resources

Select our role which is required for the project.

Code Test Monitor **Configuration** Aliases Versions

General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
RDS databases
Monitoring and operations tools
Concurrency
Asynchronous invocation
Code signing
File systems
State machines

Execution role [Refresh](#) [Edit](#) [View role document](#)

Role name
webapplicationrole [Link](#)

Resource summary

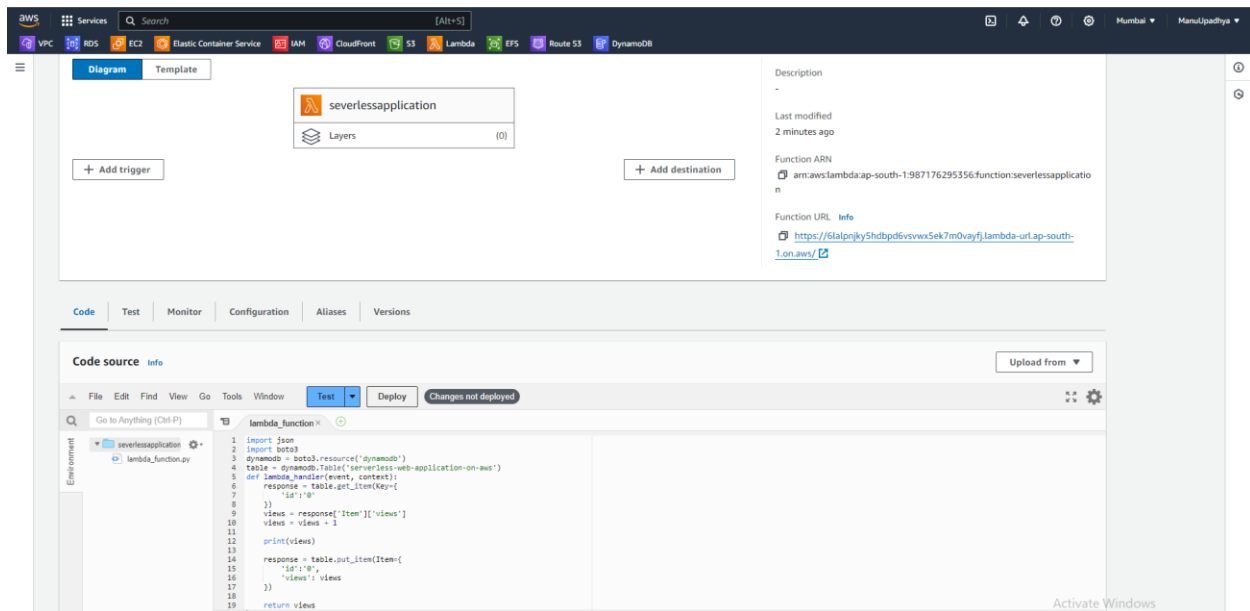
To view the resources and actions that your function has permission to access, choose a service.

AWS Application Auto Scaling
7 actions, 1 resource

By action **By resource**

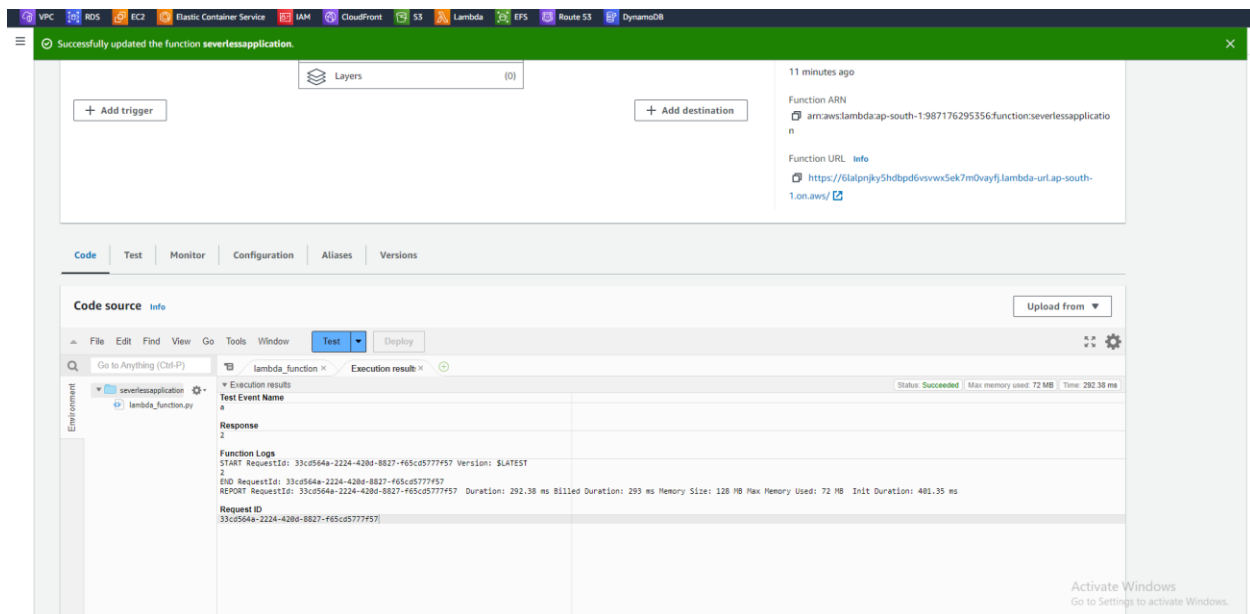
Resource	Actions
All resources	Allow: application-autoscaling:DeleteScalingPolicy Allow: application-autoscaling:DeregisterScalableTarget Allow: application-autoscaling:DescribeScalableTargets Allow: application-autoscaling:DescribeScalingActivities Allow: application-autoscaling:DescribeScalingPolicies Allow: application-autoscaling:PutScalingPolicy Allow: application-autoscaling:RegisterScalableTarget

Adding code for lambda function.

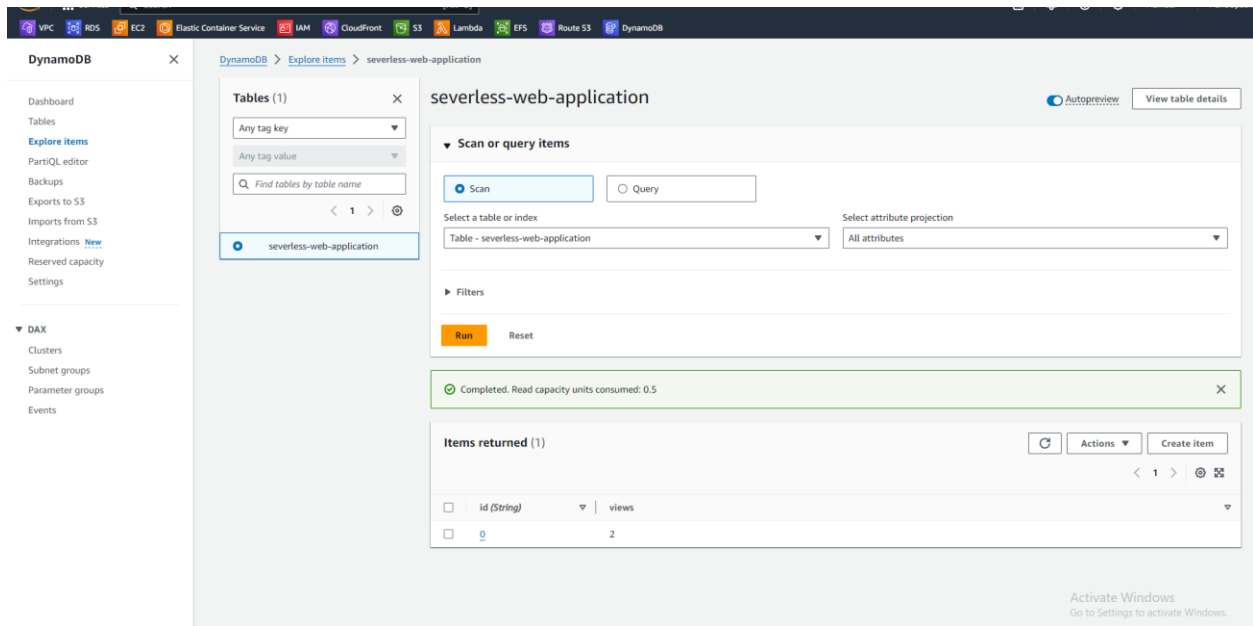


Deploy and test it again.

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



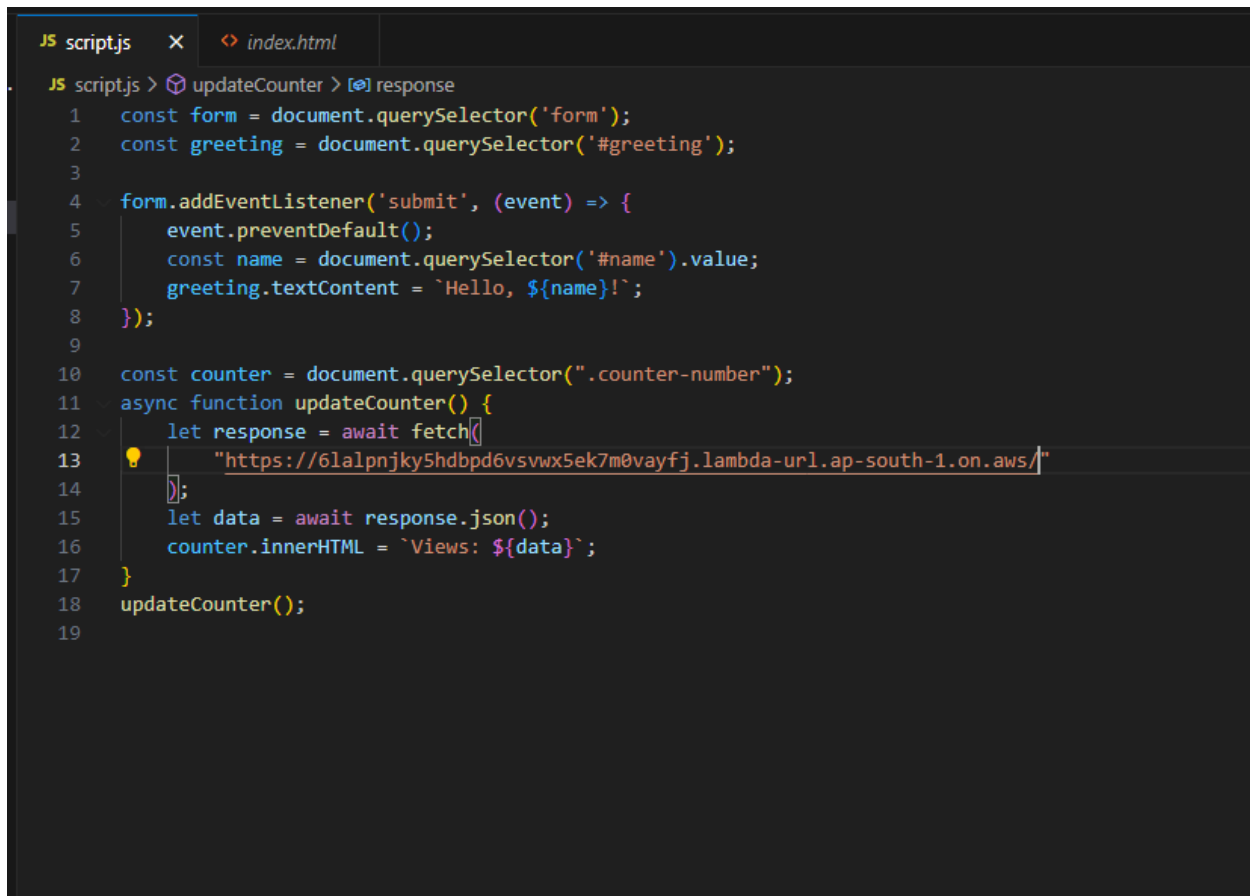
The updated view count in DynamoDB table.



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.



```
JS script.js x index.html
JS script.js > updateCounter > response
1 const form = document.querySelector('form');
2 const greeting = document.querySelector('#greeting');
3
4 form.addEventListener('submit', (event) => {
5   event.preventDefault();
6   const name = document.querySelector('#name').value;
7   greeting.textContent = `Hello, ${name}!`;
8 });
9
10 const counter = document.querySelector(".counter-number");
11 async function updateCounter() {
12   let response = await fetch(
13     "https://61alpnjky5hdbpd6vsvwx5ek7m0vayfj.lambdUrl.ap-south-1.on.aws/"
14   );
15   let data = await response.json();
16   counter.innerHTML = `Views: ${data}`;
17 }
18 updateCounter();
19
```

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.