AWS Serverless Deployment

AWS serverless deployment involves deploying applications that leverage serverless services like AWS Lambda, Amazon API Gateway, Amazon S3, and Amazon DynamoDB, without the need to provision or manage servers. This approach allows developers to focus on writing code while AWS handles the underlying infrastructure.

AWS Serverless deployment diagram

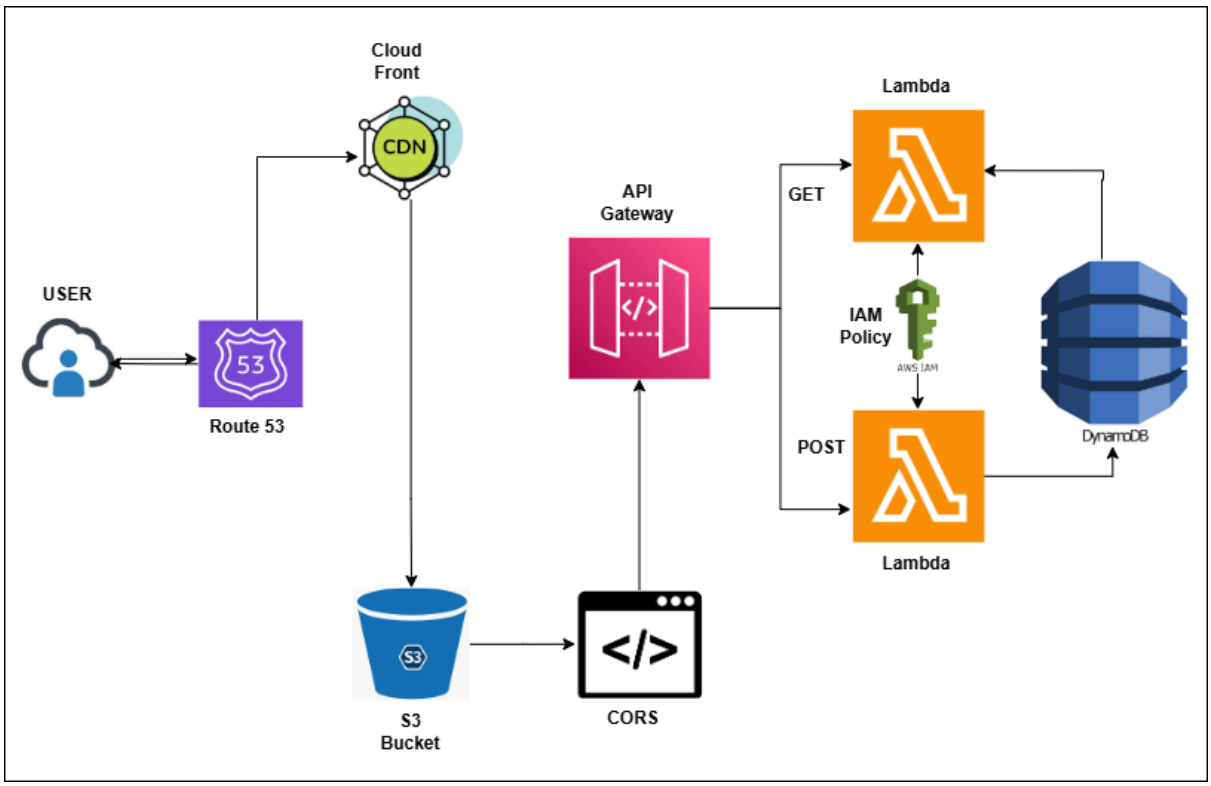


Table of Contents

AWS Serverless deployment diagram 1
Step1: Creating Application 2
Step2: S3 bucket creation 2
Step3: Creating Table in Dynamo DB 3
Step4: Creating Lambda function: 4
Step5: Creating API Gateway 5
Step6: Enabling CloudFront Distribution (CDN) 9
Step7: Hosting Application in Route53 13

Step1: Creating Application

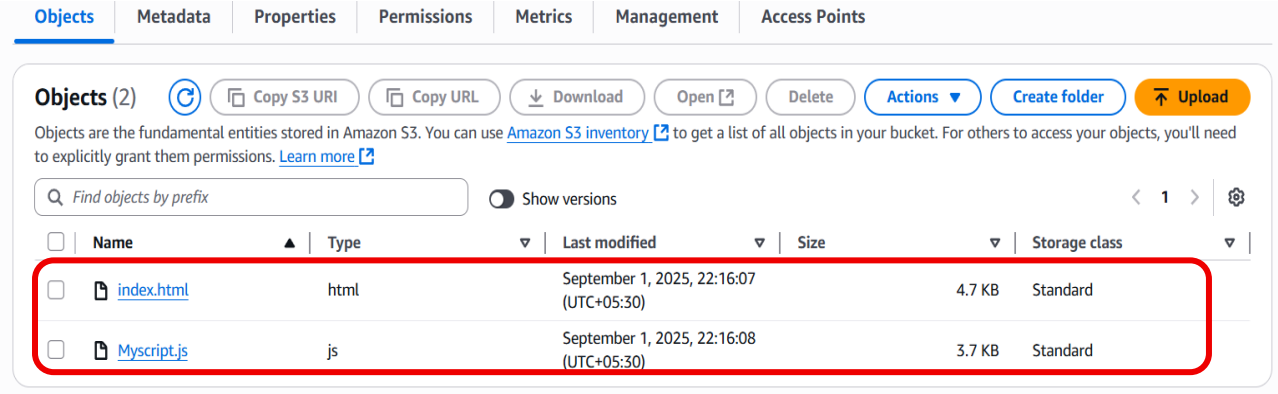
Developed the Project which includes Frontend and backend process. Frontend includes, posting and retrieving the Employee details in API gateway using AJAX and enabling the CORS. Backend includes API gateway is used to trigger the lambda function to insert and select the Employee records in Dynamo DB.

Step2: S3 bucket creation

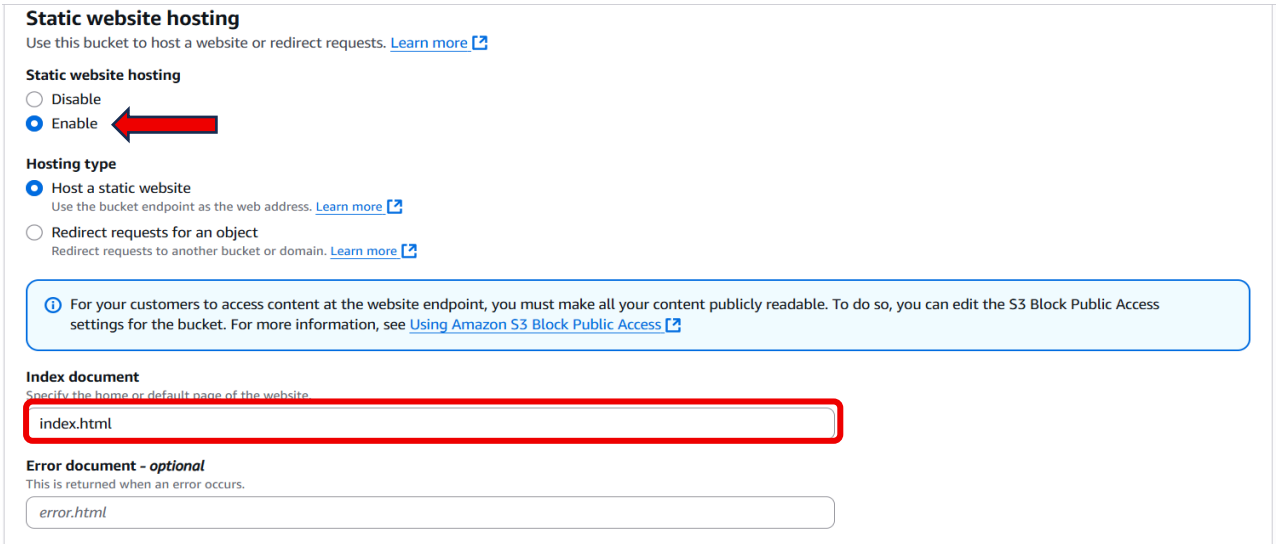
An Amazon S3 bucket is a fundamental storage resource within AWS S3. It functions as a container for storing objects, which are essentially files (like images, documents, or application data) along with any associated metadata.

- Download the Project from My Github Repo
- Create the S3 bucket with unique name and upload the Project inside the s3 bucket.
- Enable the S3 bucket public access rule required for static website hosting.
- To make the S3 bucket has private , we are using cloud front distribution (CDN).
- Enabling the static website hosting configuration.
- Attach AWS S3 bucket Policy to get the objects accessed by the users.

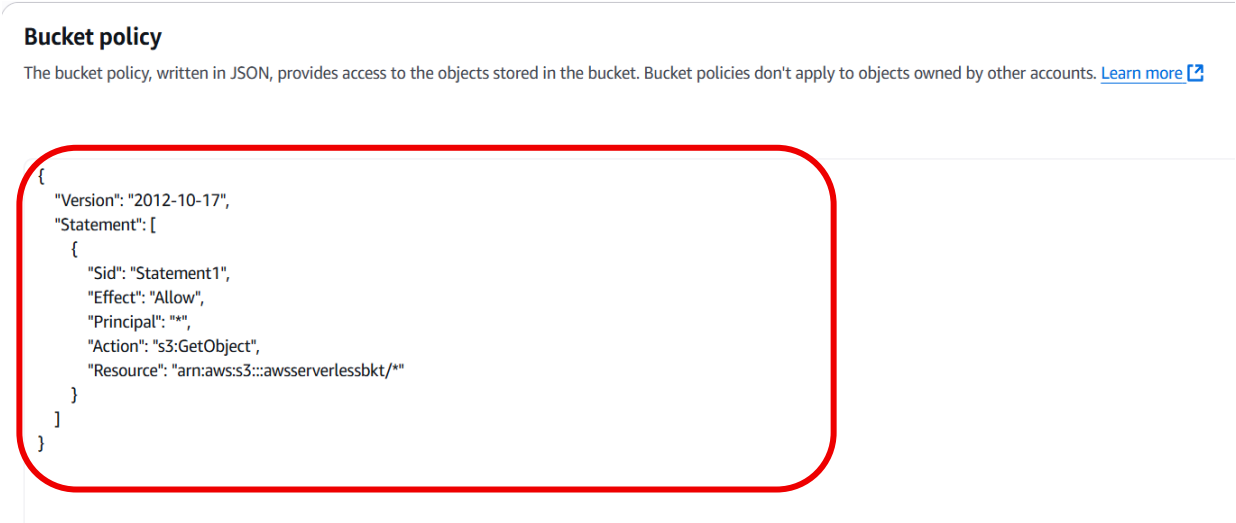
- Uploaded files in S3 bucket



- Enabling static website hosting



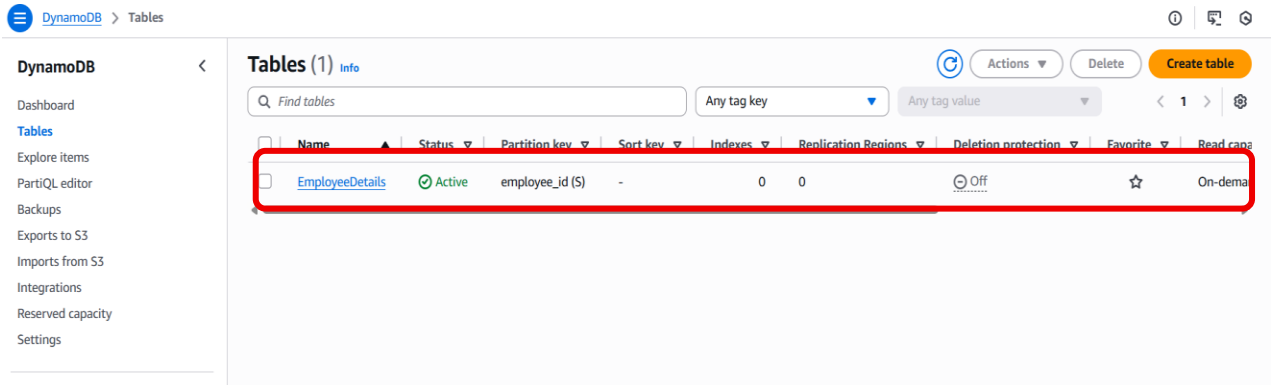
- S3 Bucket policy



Step3: Creating Table in Dynamo DB

Amazon DynamoDB is a fully managed, serverless NoSQL database service offered by Amazon Web Services (AWS). It is designed to provide fast and predictable performance with seamless scalability, making it suitable for high-performance, internet-scale applications.

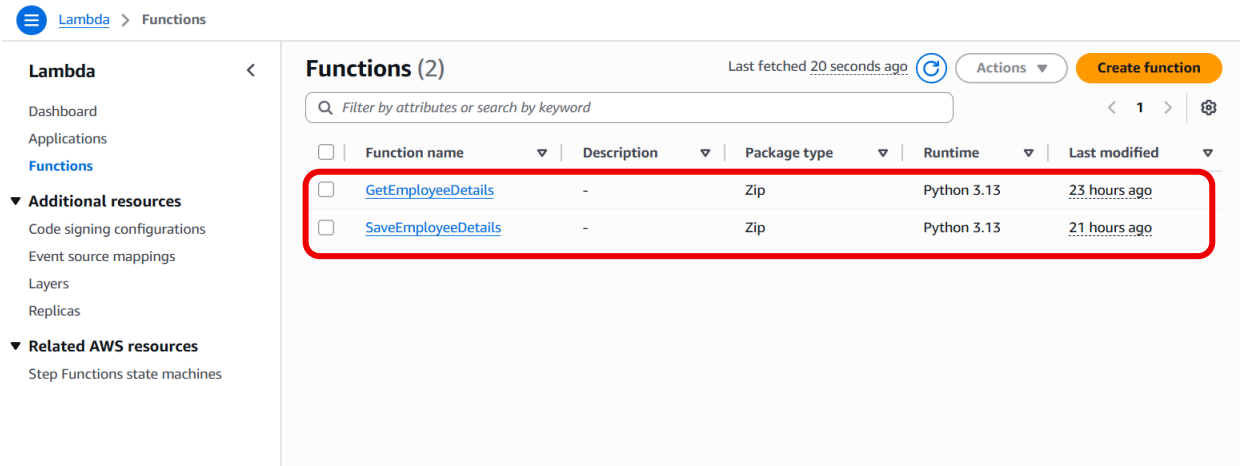
- Create Employee Table in DynamoDB



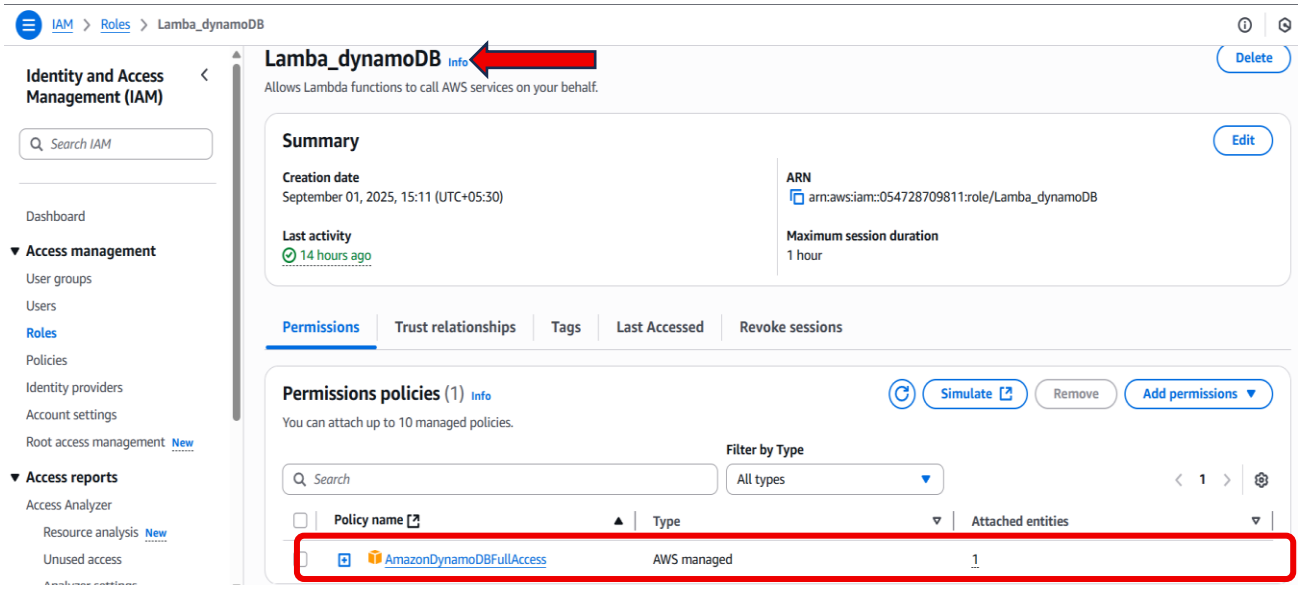
Step4: Creating Lambda function:

AWS Lambda is a serverless compute service that enables users to run code without provisioning or managing servers. It is a core component of Amazon Web Services (AWS) and operates on a "pay-per-use" model, meaning charges are incurred only when the code is actively running.

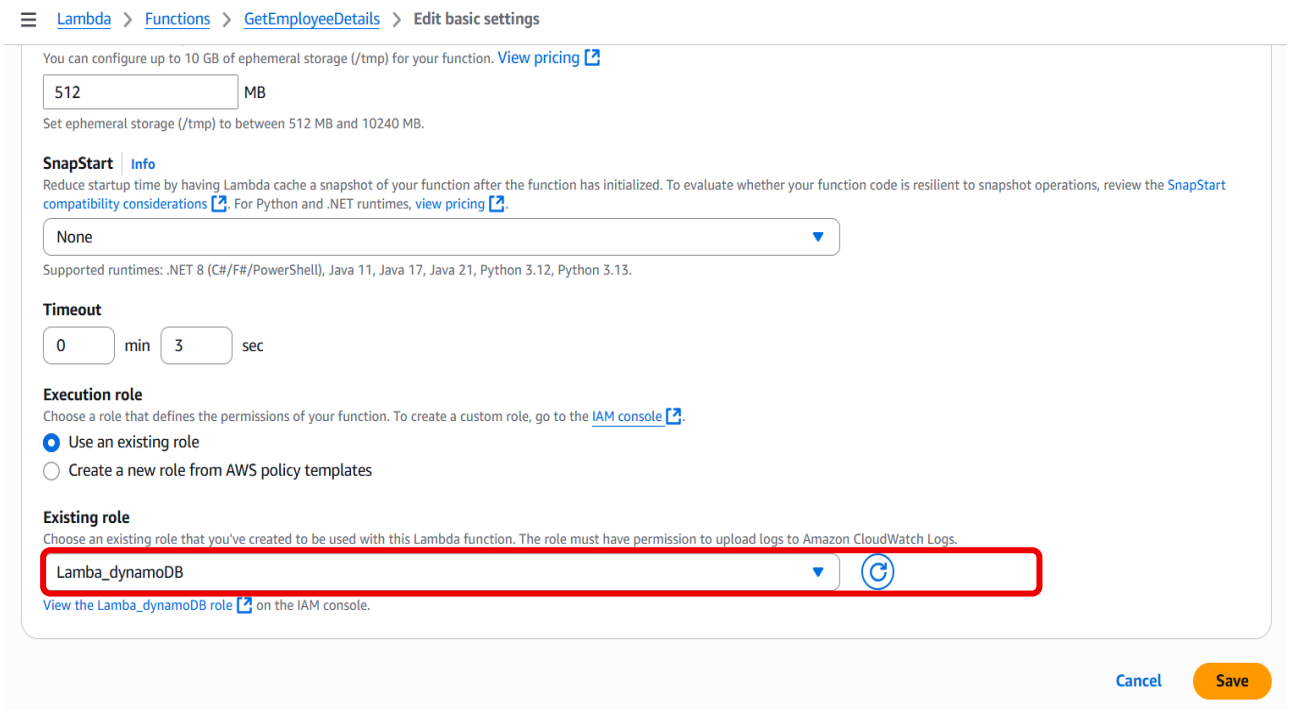
- Creating the lambda function to automate the Insertion the Selection process from Dynamo DB.
- Creating the Event handler for getting Employee details and posting Employee details from DynamoDB in lambda function using Python.



- Creating IAM role for accessing the Dynamo DB.



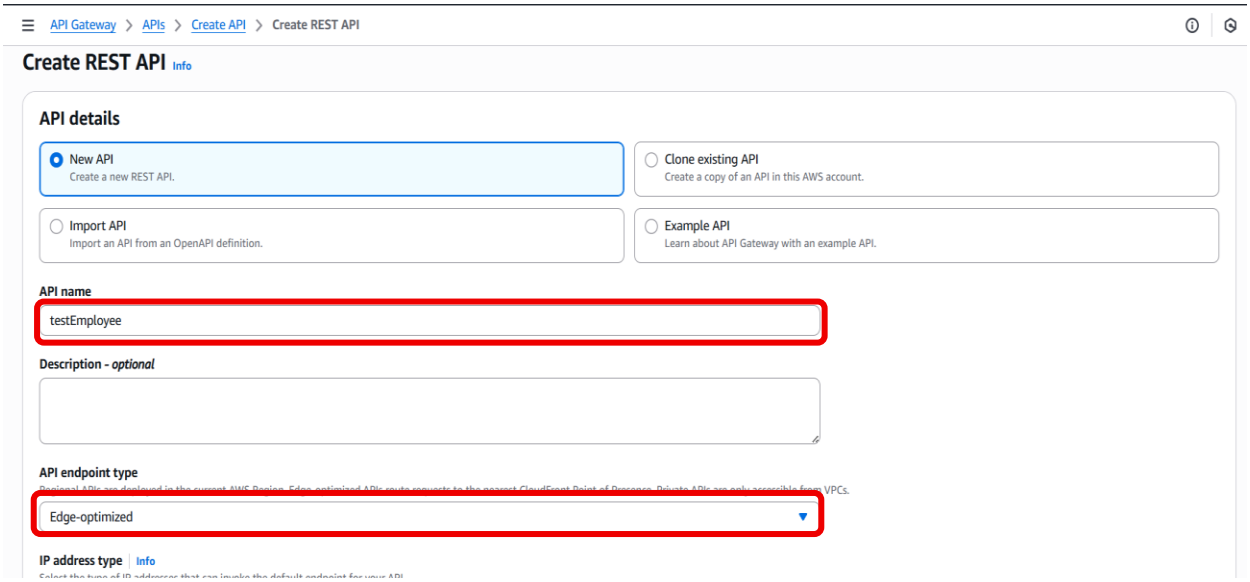
- Attach the IAM role in Lambda function.



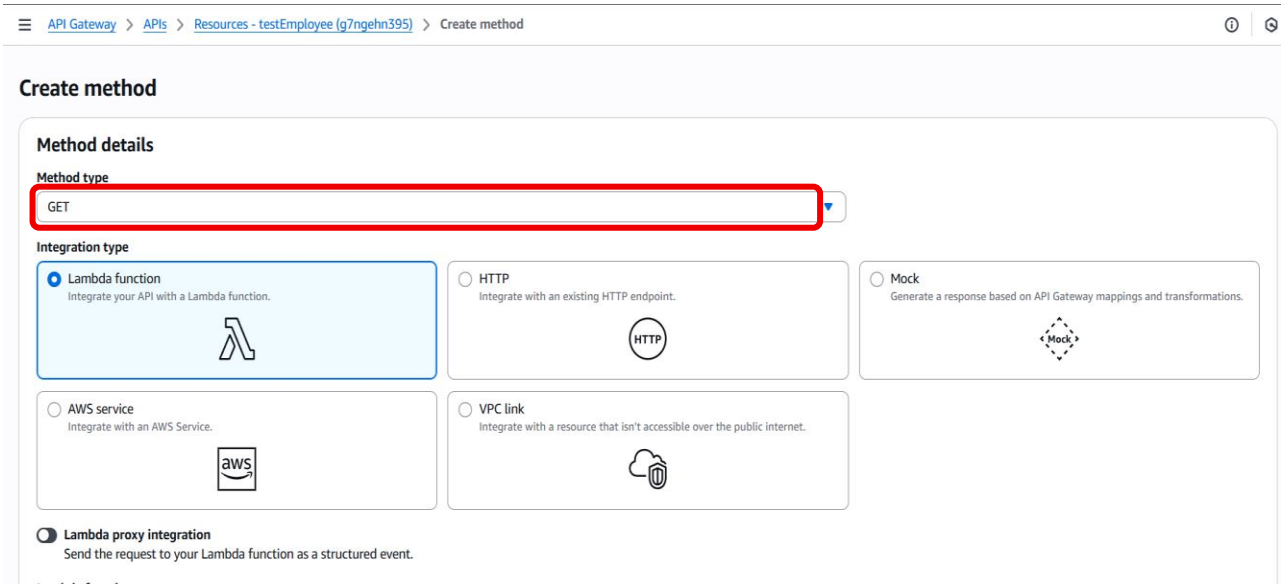
Step5: Creating API Gateway

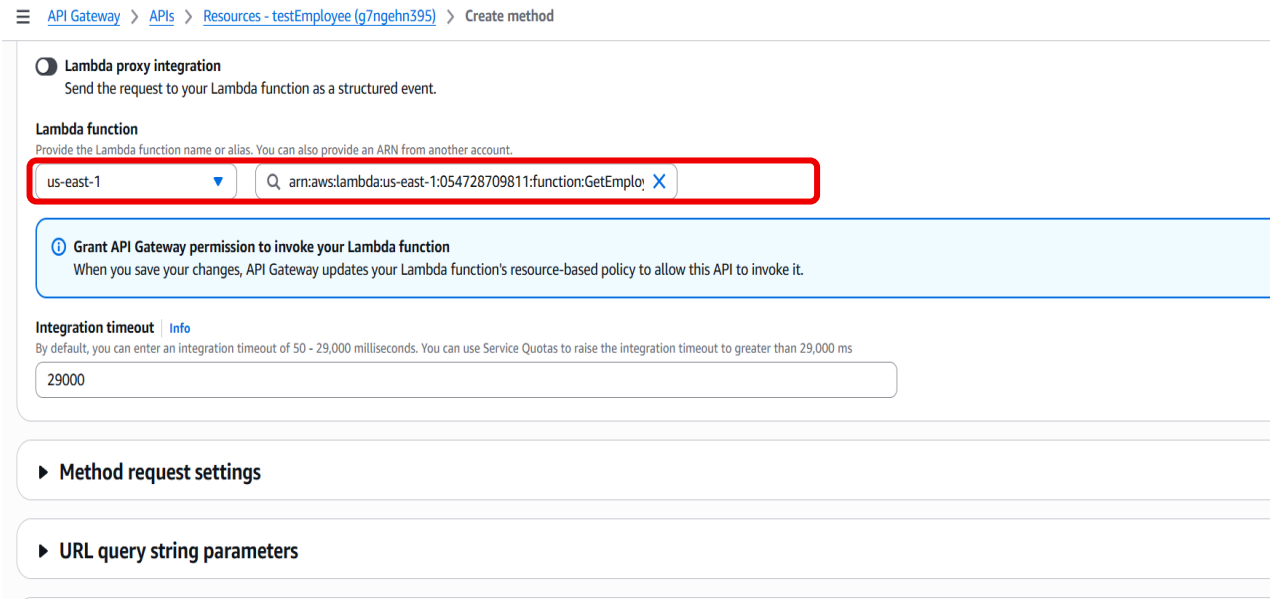
AWS API Gateway is a fully managed service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. It acts as a "front door" for applications to access data, business logic, or functionality from backend services.

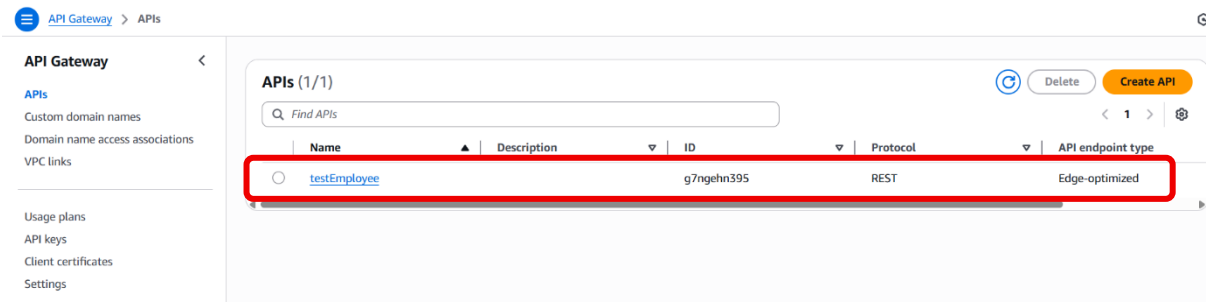
- Creating API to trigger the lambda function for GET and POST Method

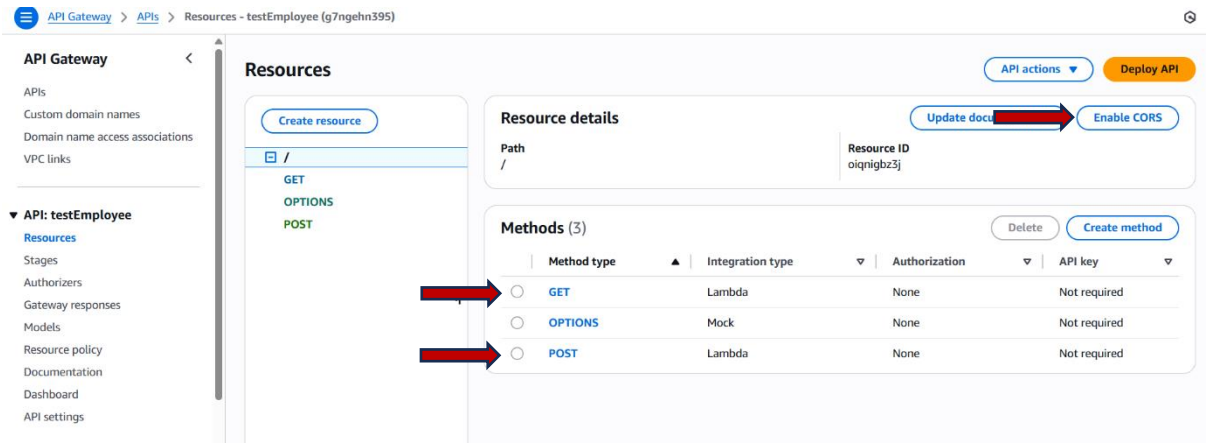


- Create GET and POST Method and integrate it with lambda functions.

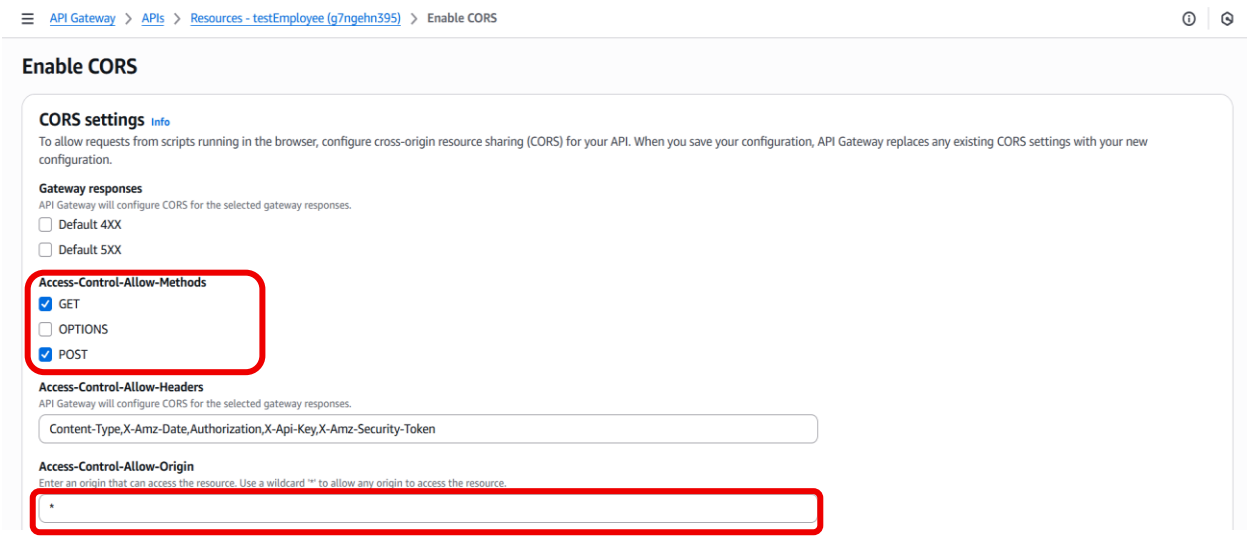



API Gateway > APIs > Resources - testEmployee (g7ngehn395) > Create method
Lambda proxy integration
Send the request to your Lambda function as a structured event.
Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.
us-east-1
arn:aws:lambda:us-east-1:054728709811:function:GetEmplo
Grant API Gateway permission to invoke your Lambda function
When you save your changes, API Gateway updates your Lambda function's resource-based policy to allow this API to invoke it.
Integration timeout
Info
By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms
29000
Method request settings
URL query string parameters

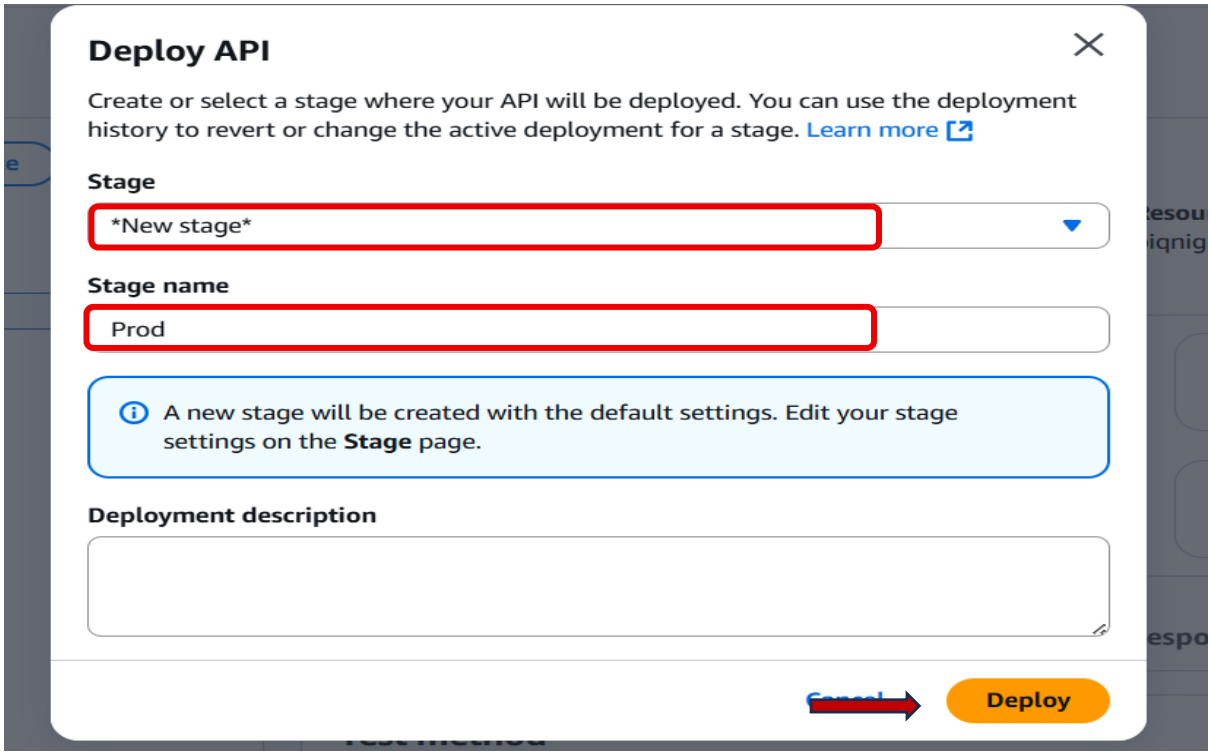

API Gateway > APIs
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
Usage plans
API keys
Client certificates
Settings
APIs (1/1)
Delete
Create API
Find APIs
Name
Description
ID
Protocol
API endpoint type
testEmployee
g7ngehn395
REST
Edge-optimized


API Gateway > APIs > Resources - testEmployee (g7ngehn395)
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
API: testEmployee
Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation
Dashboard
API settings
Resources
API actions
Deploy API
Create resource
/
GET
OPTIONS
POST
Resource details
Update docu
Enable CORS
Path
/
Resource ID
oiqnigbz3j
Methods (3)
Delete
Create method
Method type
Integration type
Authorization
API key
GET
Lambda
None
Not required
OPTIONS
Mock
None
Not required
POST
Lambda
None
Not required

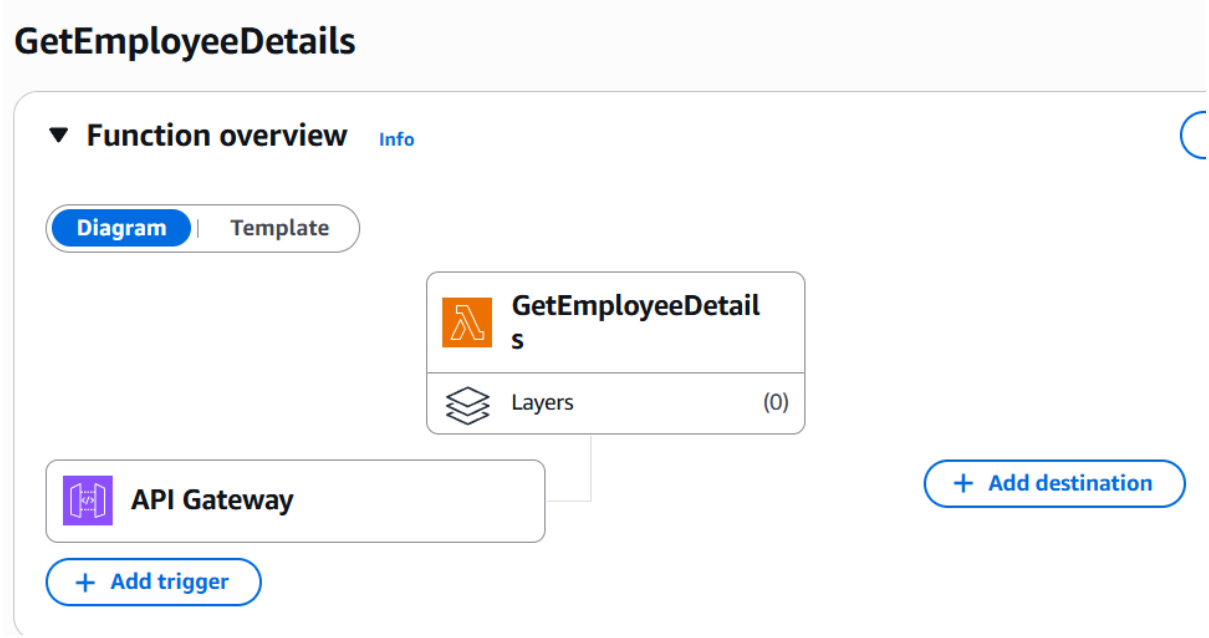
- Enabling CORS , to communicate Frontend AJAX call to API endpoint.

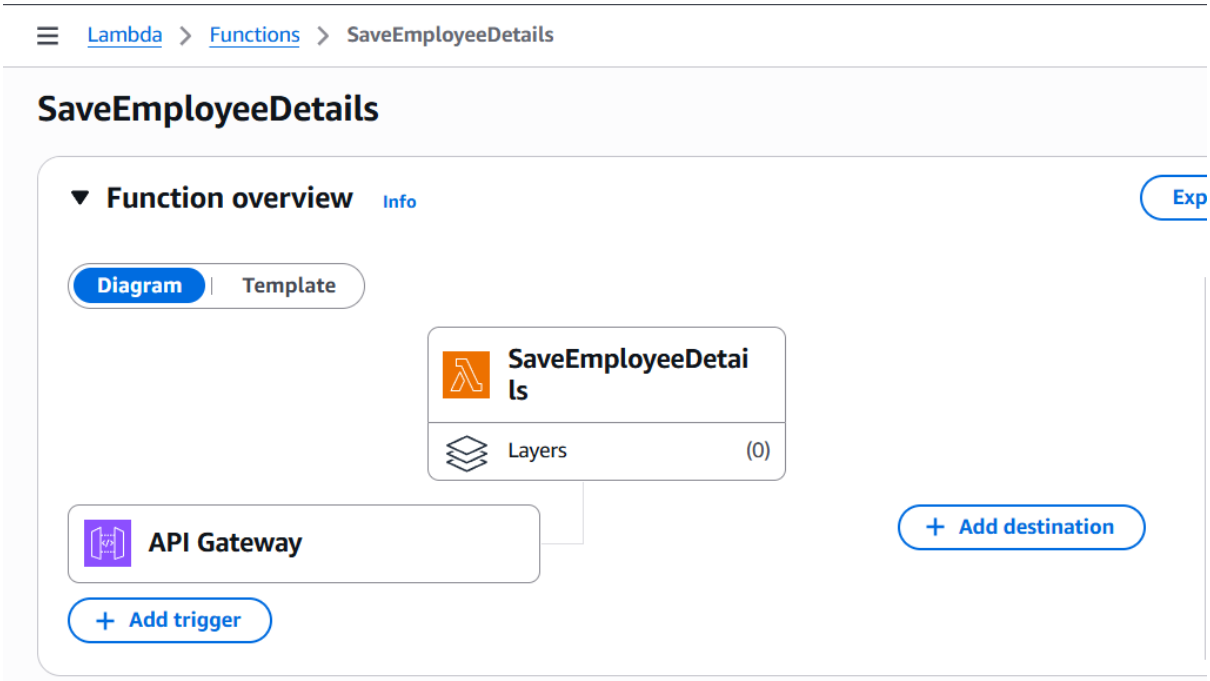


- API Deployment



- API trigger to Lambda function

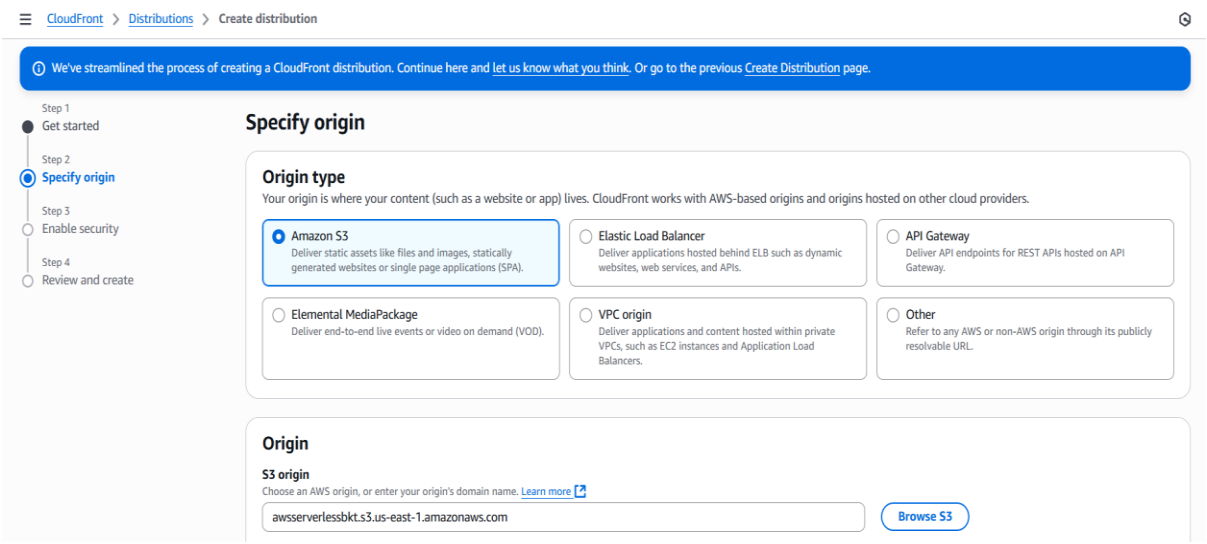




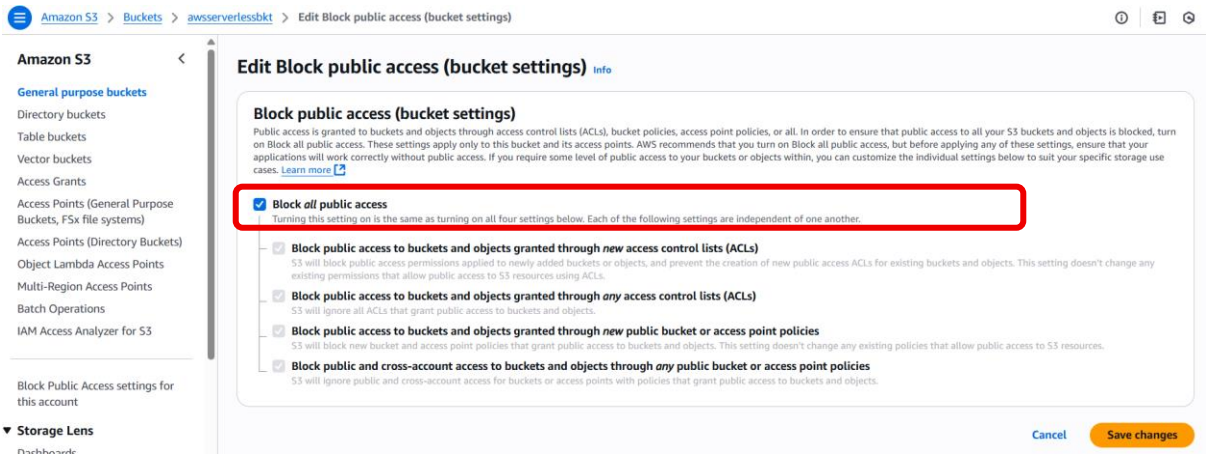
Step6: Enabling CloudFront Distribution (CDN)

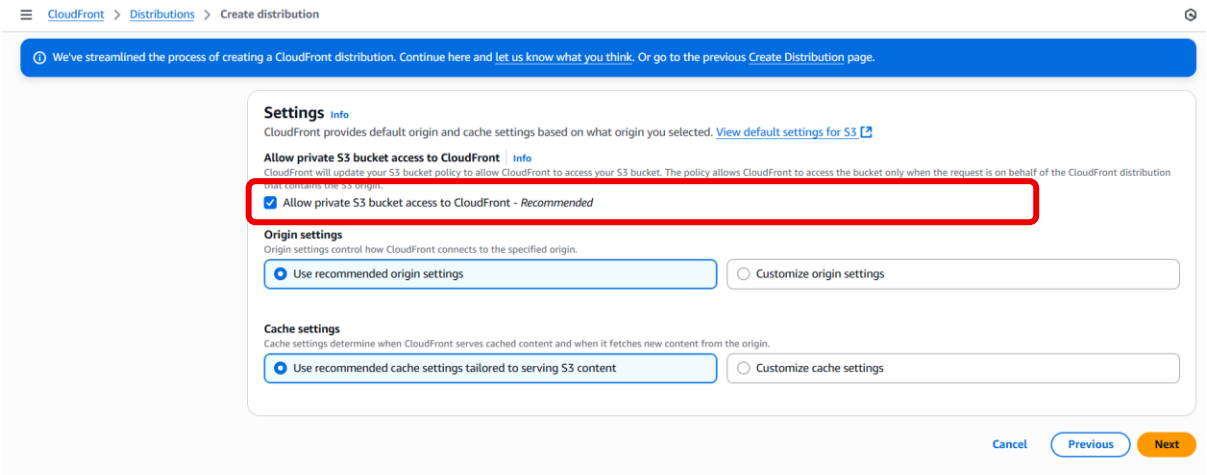
- Amazon CloudFront is a content delivery network service that accelerates the delivery of your static and dynamic web content to users globally.
- It achieves this by caching content at edge locations (data centers) around the world, bringing the content closer to your users and reducing latency

- Create Cloud front distribution with S3 bucket



- Making S3 bucket as private.





- S3 bucket policy after configuring CDN

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::awsserverlessbkt/*"
    },
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::awsserverlessbkt/*",
      "Condition": {
        "ArnLike": {
```

- Run CDN through Http. Copy the domain and run in browser.
- To make this hosting secure, we will go for SSL certification enabled, and hosted this distribution through Route 53.

Creating Hosted Zone

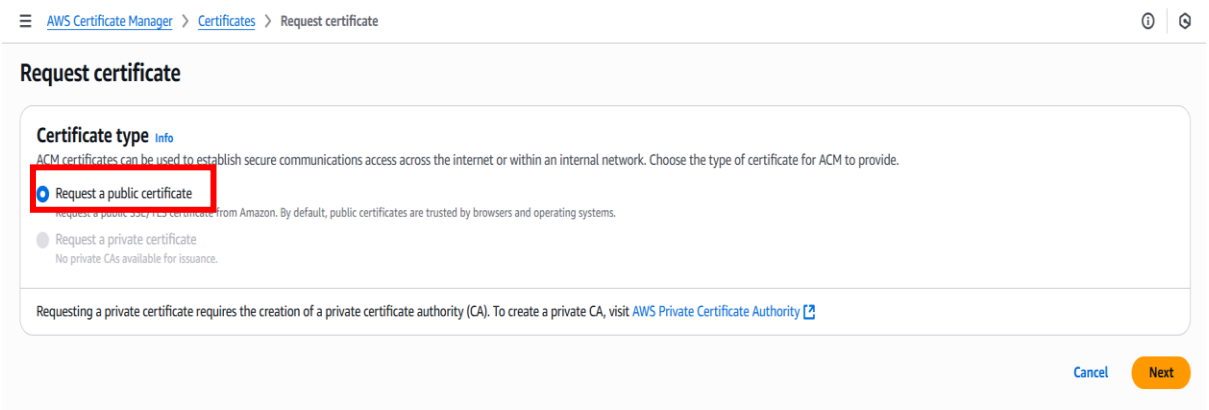
→ Purchase the Domain in GoDaddy website and Create the Hosted Zone with the same Domain name in Route 53.

→ Automatically Namespaces will be created inside the Hosted Zone.

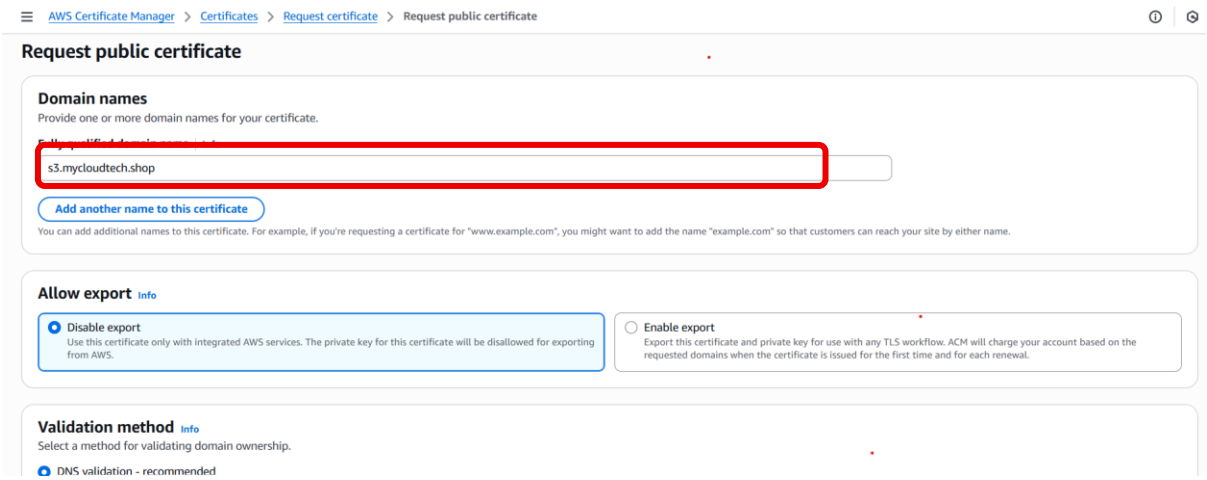
→ Copy the namespaces and paste inside the GoDaddy Domain website.

Enabling SSL Certificate (ACM)

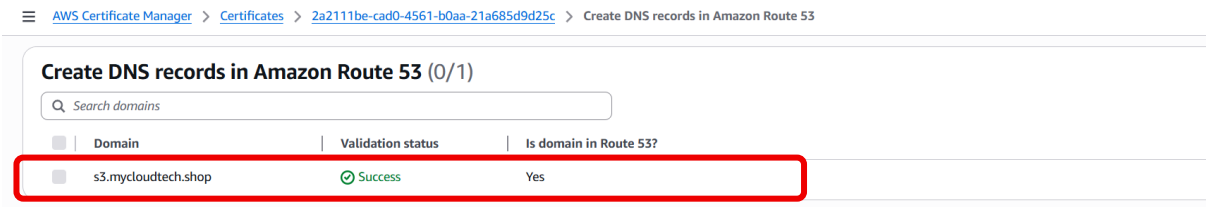
➡ Request a Public Certificate



→ Provide the fully qualified Domain Name (contains sub domain and main domain)



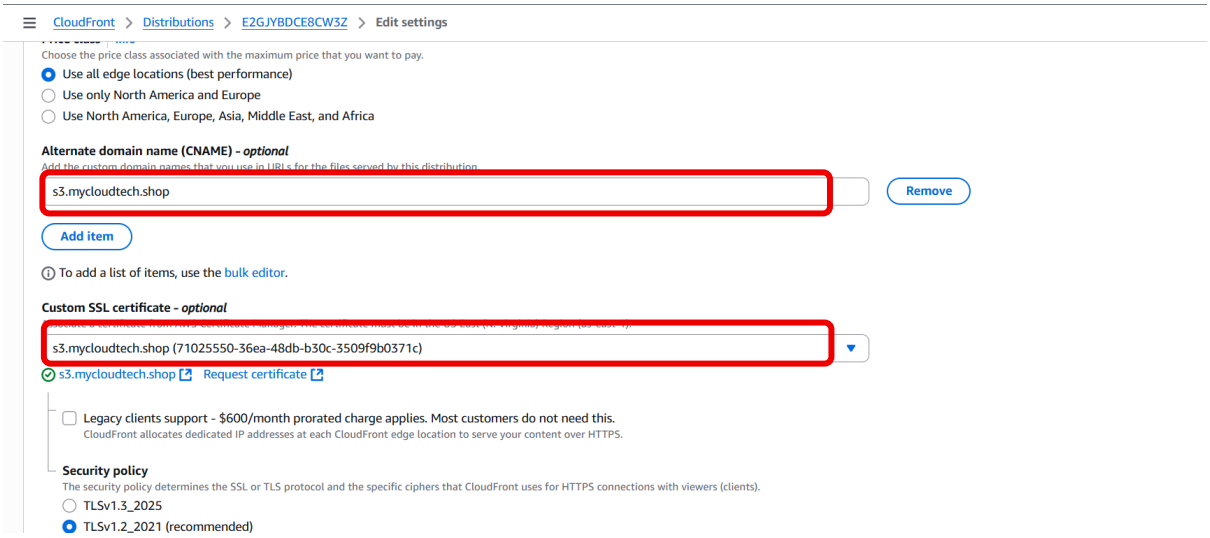
→ DNS Record creation

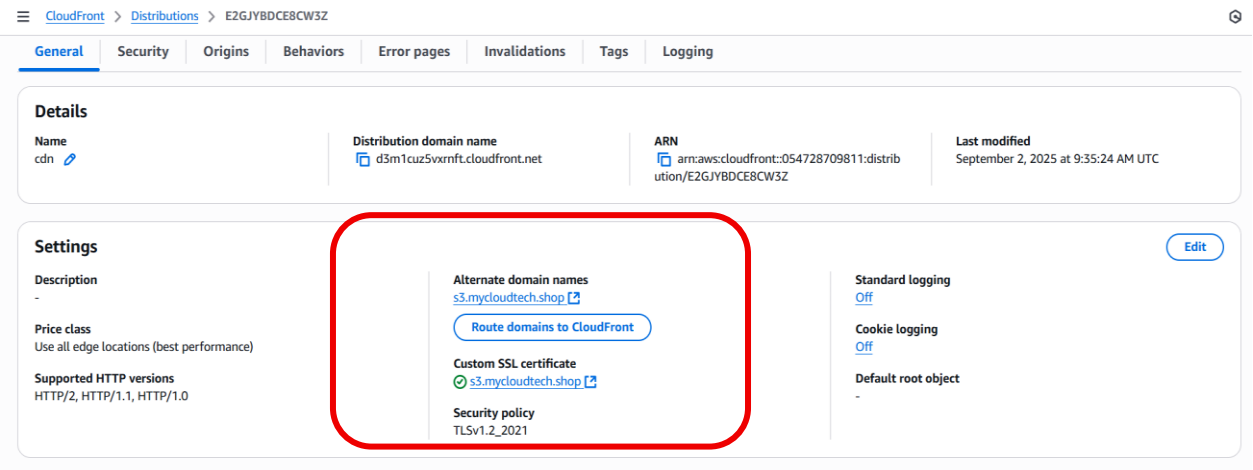


DNS Record is Created in Route 53 and Request for public certificate (SSL) is Issued.

Adding Domain in CloudFront

→ Enter Fully qualified Domain Name which we have created for SSL.



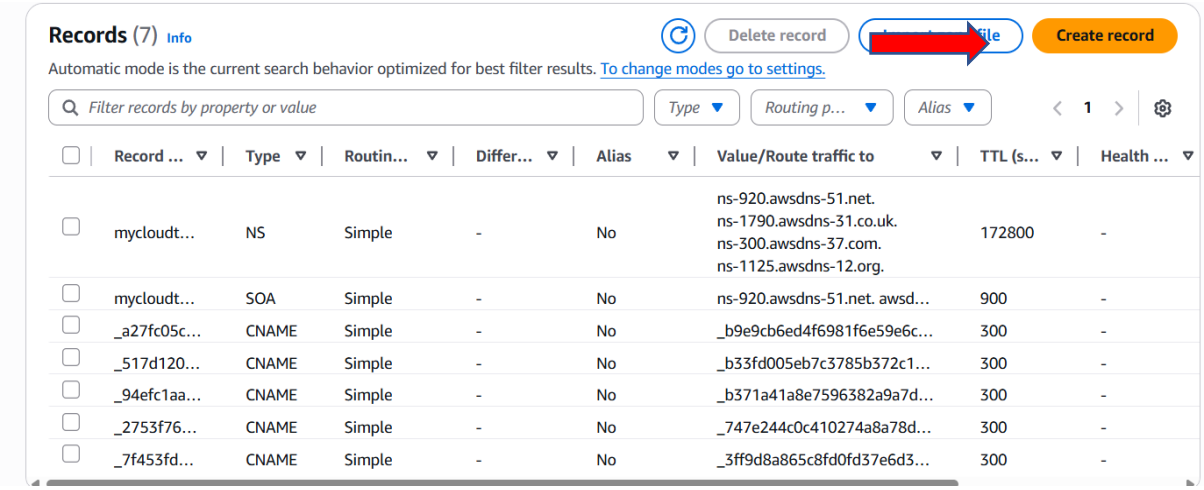


→ SSL Certificate is enabled for CloudFront.

→ Now, Route the CloudFront Distribution in Route 53.

Step7: Hosting Application in Route53

- Amazon Route 53 is to act as a highly available and scalable Domain Name System (DNS) web service for routing internet traffic to applications and resources
- It translates human-readable domain names (like "www.example.com") into the numerical IP addresses that computers use to connect. Additionally, Route 53 offers domain registration and health checking capabilities.



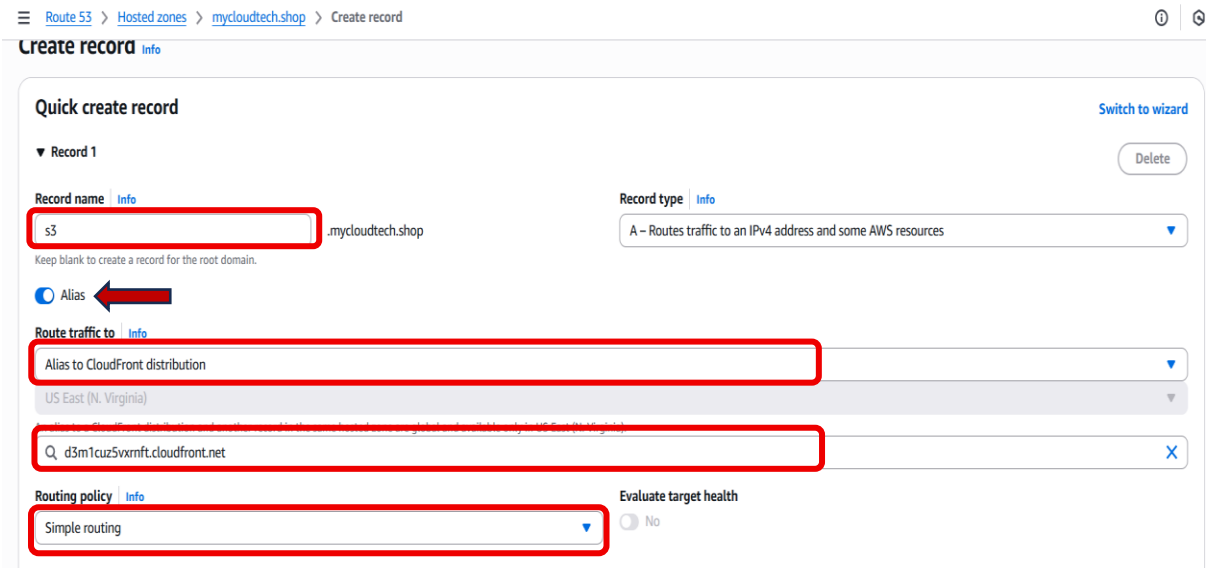
Create the Record in Hosted Zone:

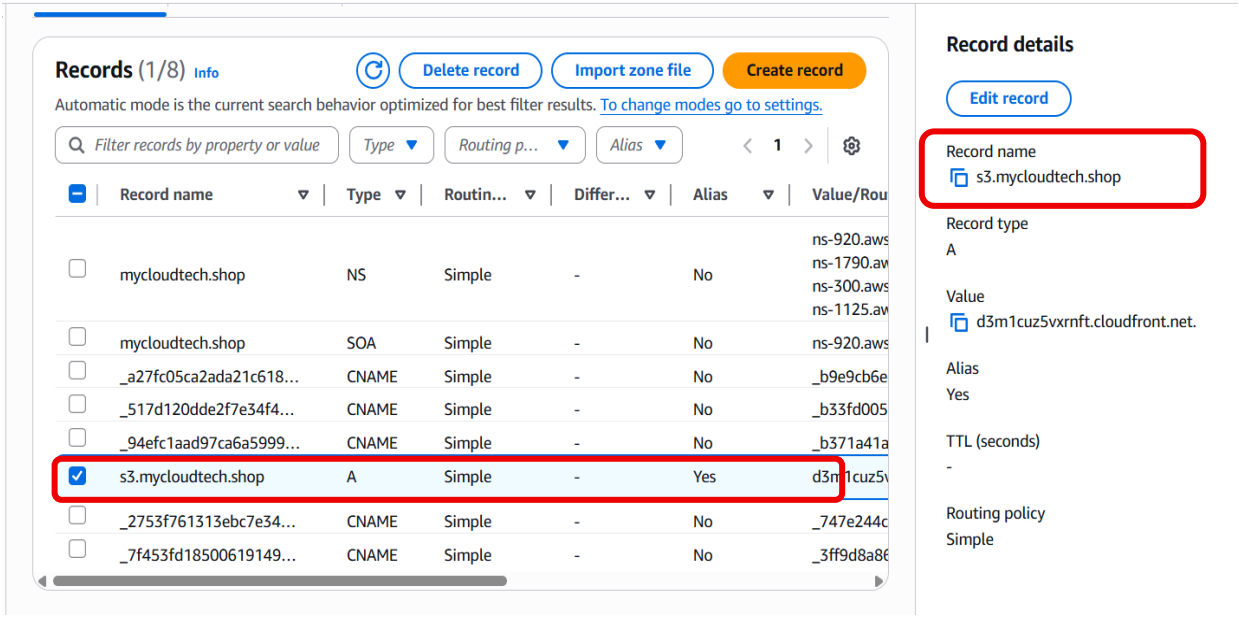
→ Select the Record type

→ Enable Alias

→ Route the traffic: Alias to CloudFront distribution

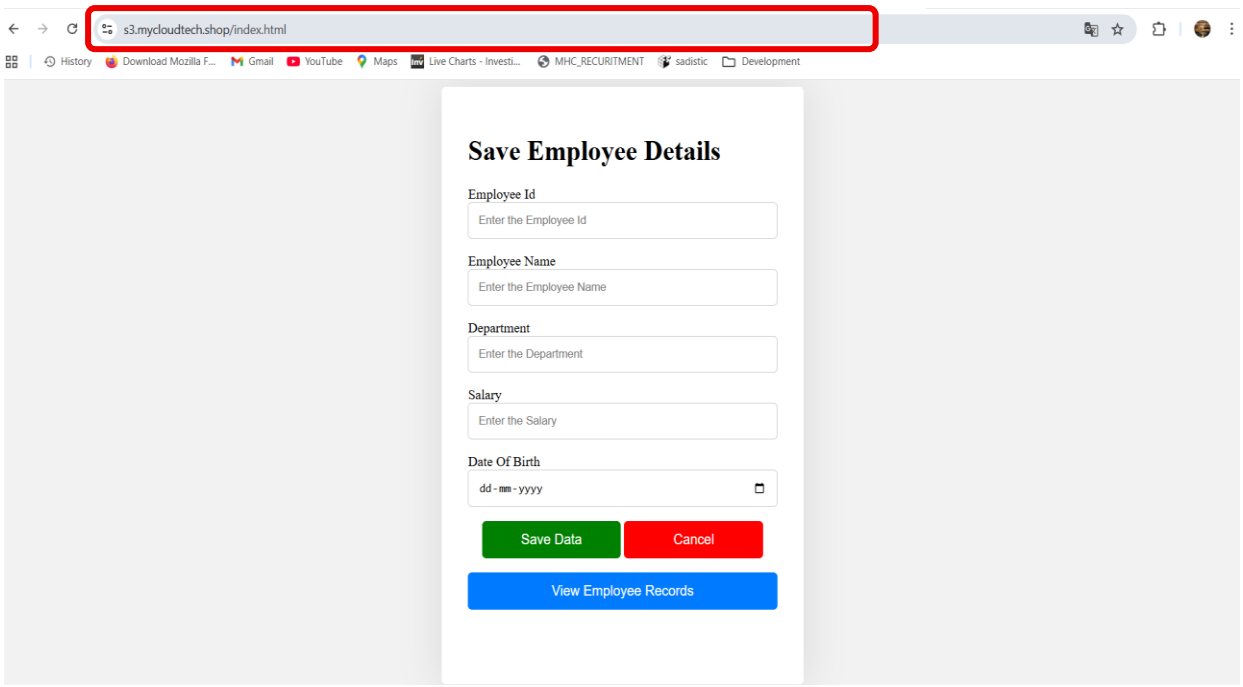
→ Selecting Simple routing policy.

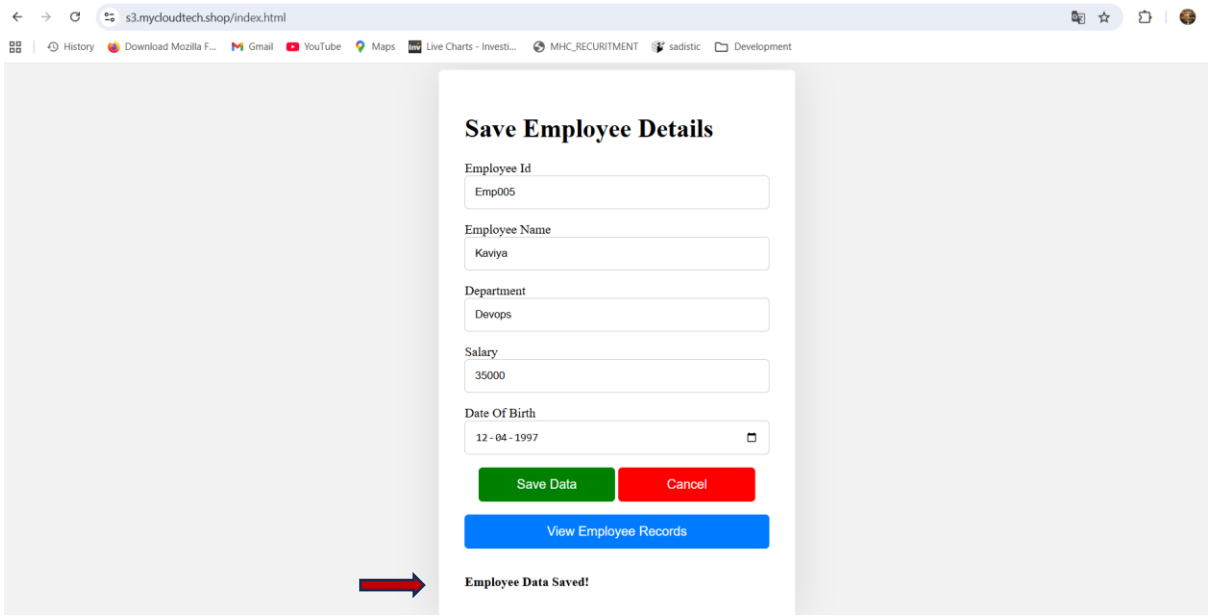


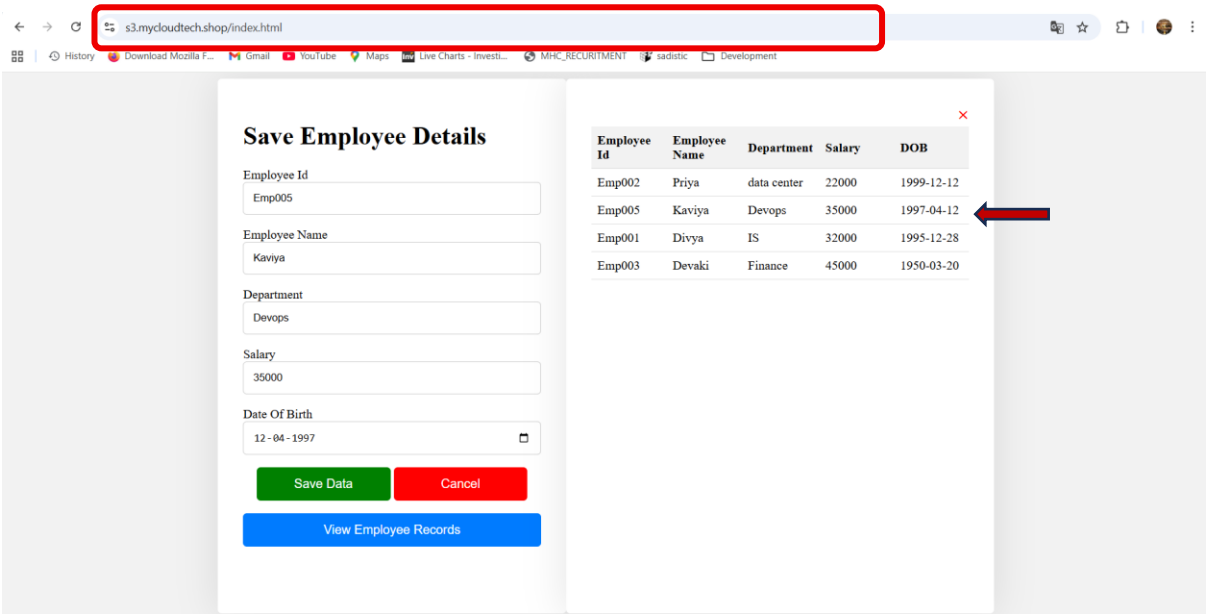


Now our application is hosted in Route 53, lets run the application.

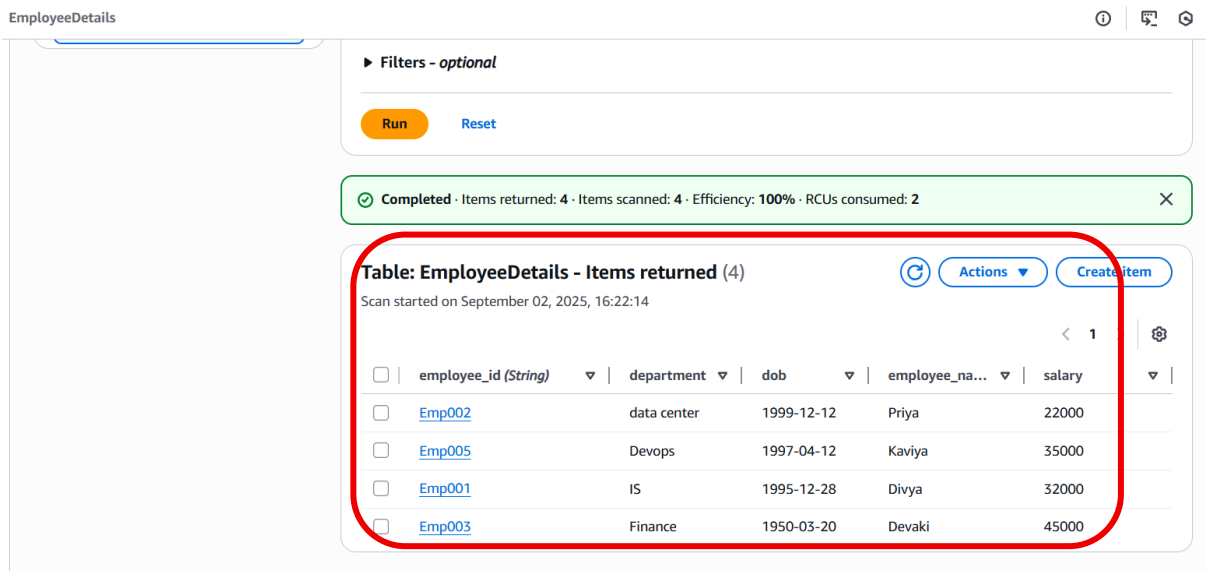
Run: s3.mycloudtech.shop


s3.mycloudtech.shop/index.html
History
Download Mozilla F...
Gmail
YouTube
Maps
Live Charts - Investi...
MHC_RECURITMENT
sadistic
Development
Save Employee Details
Employee Id
Enter the Employee Id
Employee Name
Enter the Employee Name
Department
Enter the Department
Salary
Enter the Salary
Date Of Birth
dd-mm-yyyy
Save Data
Cancel
View Employee Records


s3.mycloudtech.shop/index.html
History
Download Mozilla F...
Gmail
YouTube
Maps
Live Charts - Investi...
MHC_RECURITMENT
sadistic
Development
Save Employee Details
Employee Id
Emp005
Employee Name
Kaviya
Department
Devops
Salary
35000
Date Of Birth
12-04-1997
Save Data
Cancel
View Employee Records
Employee Data Saved!



- Checking records in Dynamo DB



AWS Serverless Project is Successfully created and deployed.