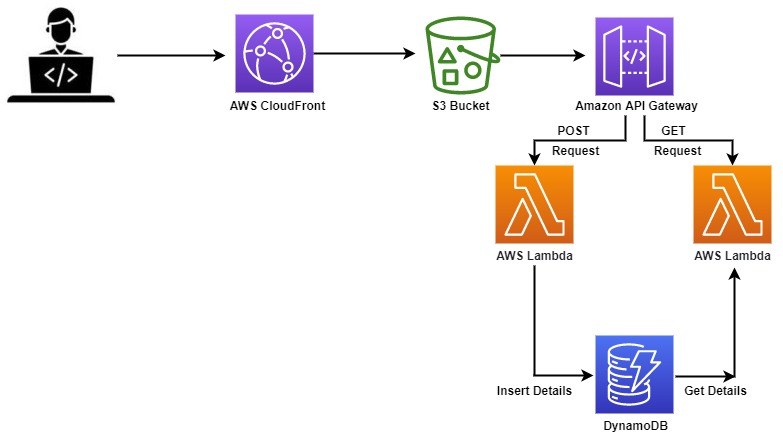
**How to Build Serverless Web Application in AWS.**

[](https://user-images.githubusercontent.com/119833411/243072109-5792a3d9-a1f6-4f2d-9287-80bb9fed0fe1.jpg)

**What is Serverless?**

* **Serverless computing is a cloud computing model where developers write and deploy code as individual functions or services without managing servers.**
* **The cloud provider takes care of infrastructure provisioning, scaling, and maintenance.**
* **Developers focus solely on writing code, and the platform automatically allocates resources and scales applications based on demand.**

**Why Serverless?**

* **serverless computing provides developers with a more streamlined and efficient development experience while delivering automatic scaling, cost savings, and improved agility.**
* **It is particularly beneficial for applications with varying workloads, sporadic usage patterns, and the need for rapid innovation.**

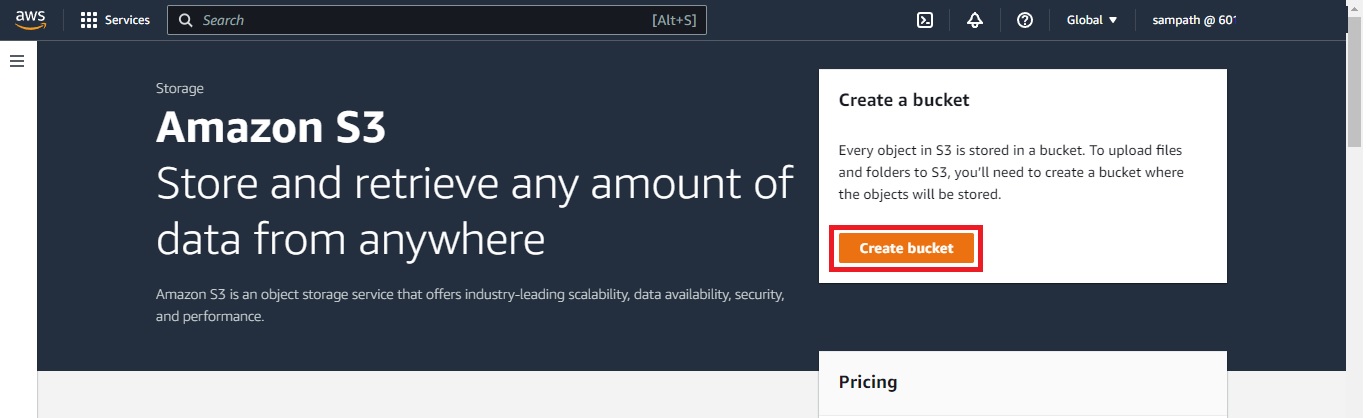
**Prerequisites.**

* **AWS Account**

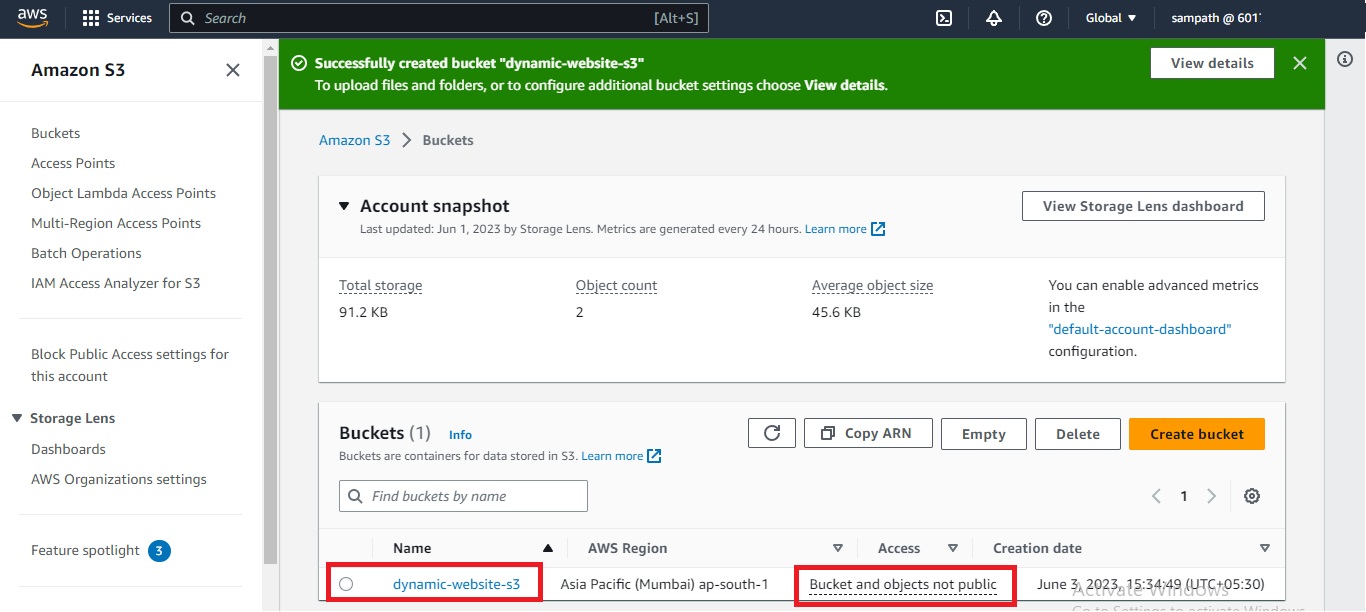
**My GitHub Repo:** <https://github.com/sampathshivakumar/Serverless-Web-Application>

**Let's start with Hosting our Application on S3 Bucket.**

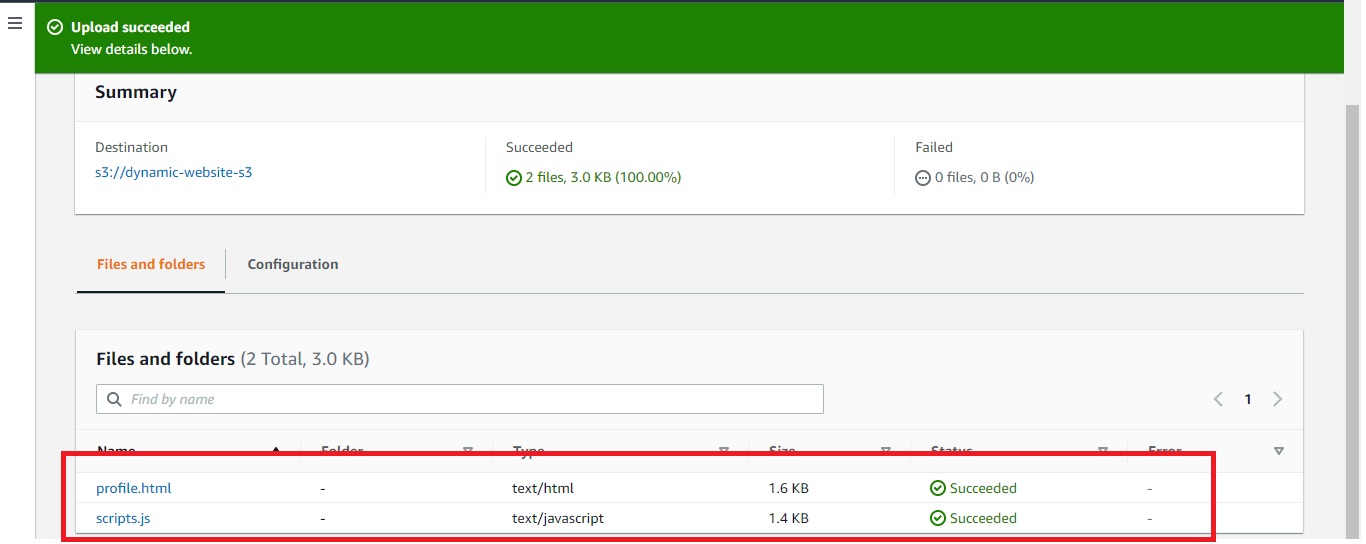
**Create a bucket**

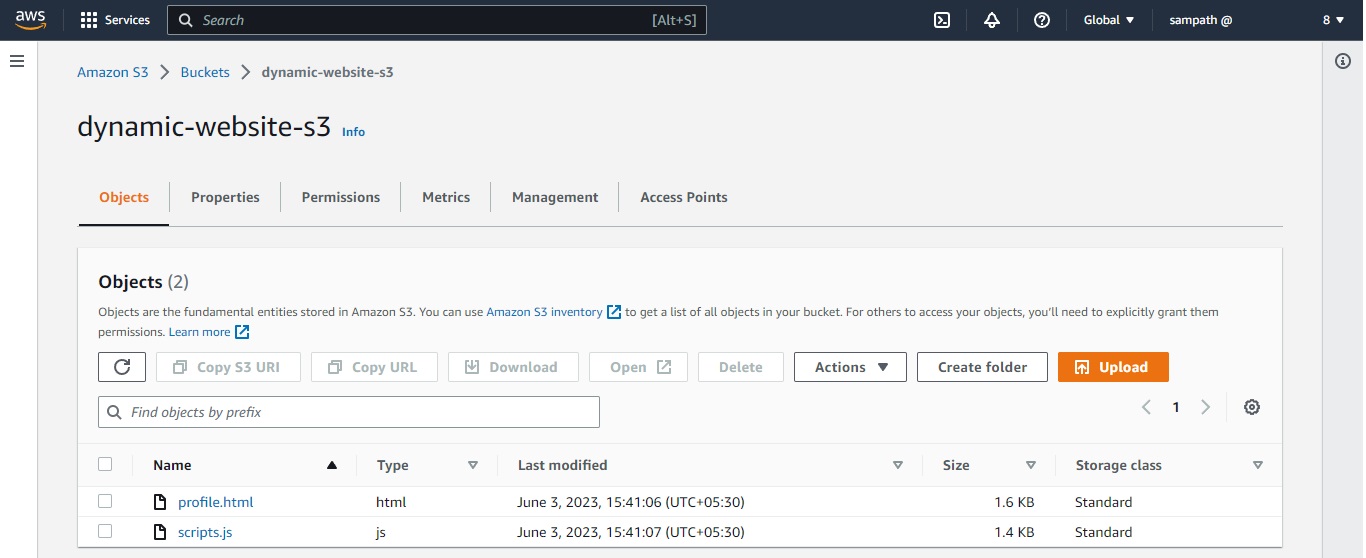
[](https://user-images.githubusercontent.com/119833411/243076437-8bf30009-2f46-4a63-9a7a-45379cf6f8ad.jpg)

**Give Bucket name "dynamic-website-s3",Tag key="owner" value="sampath", rest all looks good and clik on Create bucket.**

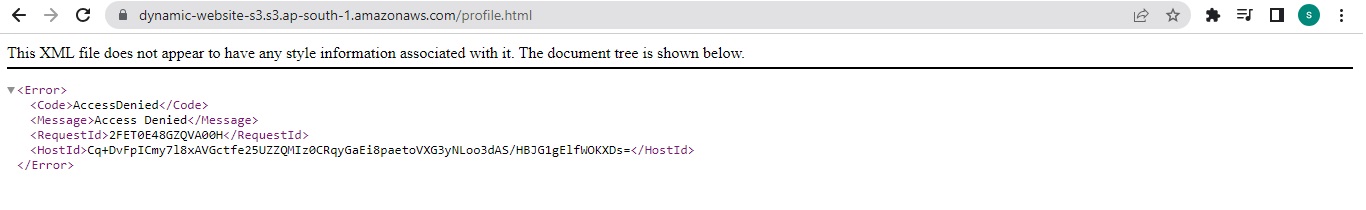
[](https://user-images.githubusercontent.com/119833411/243078802-43ff598a-6eec-43dc-a5e1-bf586e16616b.jpg)

**Now upload the Web-Content in S3 bucket**

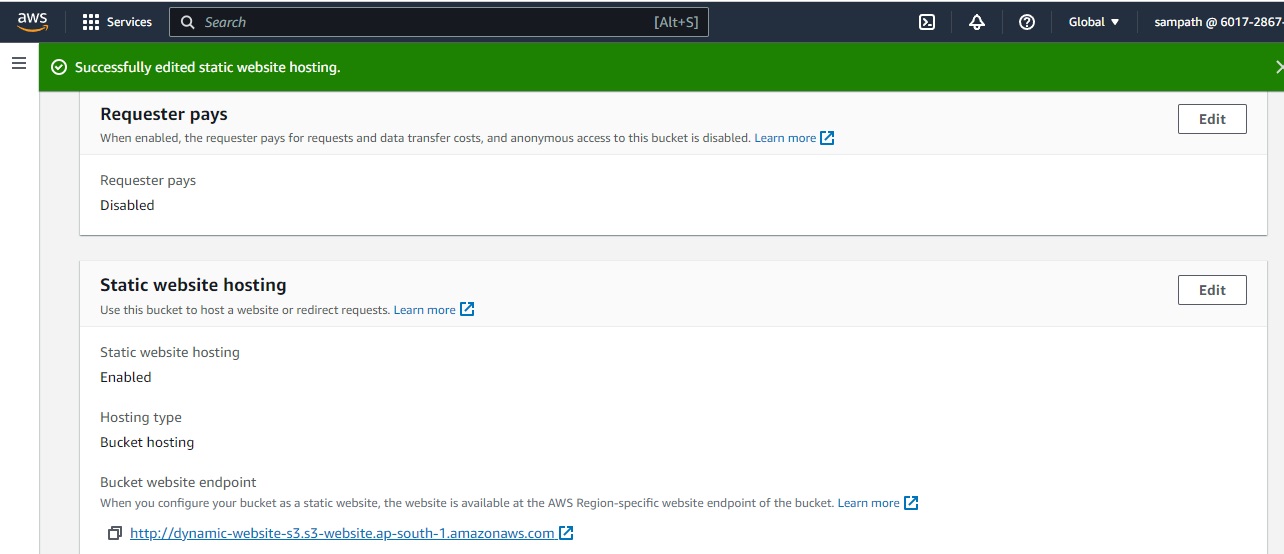
[](https://user-images.githubusercontent.com/119833411/243079053-c082f624-8719-417a-bf20-8681bdfe00c4.jpg)

[](https://user-images.githubusercontent.com/119833411/243079141-9924bd5d-0d6c-4d7b-9aa1-c27cae95dfbe.jpg)

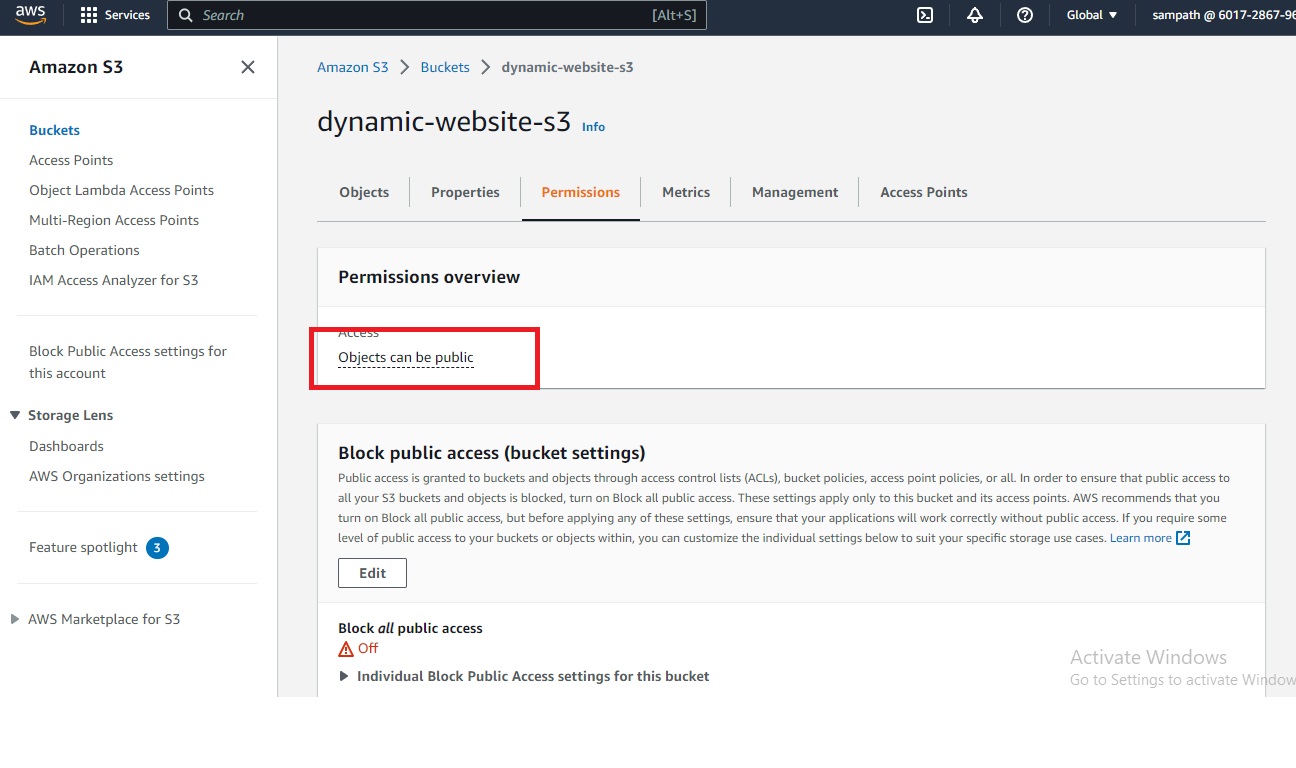
**As of now we can't access the content on S3 as my bucket is not public**

[](https://user-images.githubusercontent.com/119833411/243079285-d2ee5f5e-f5f7-465b-9d39-0e846cdf0246.jpg)

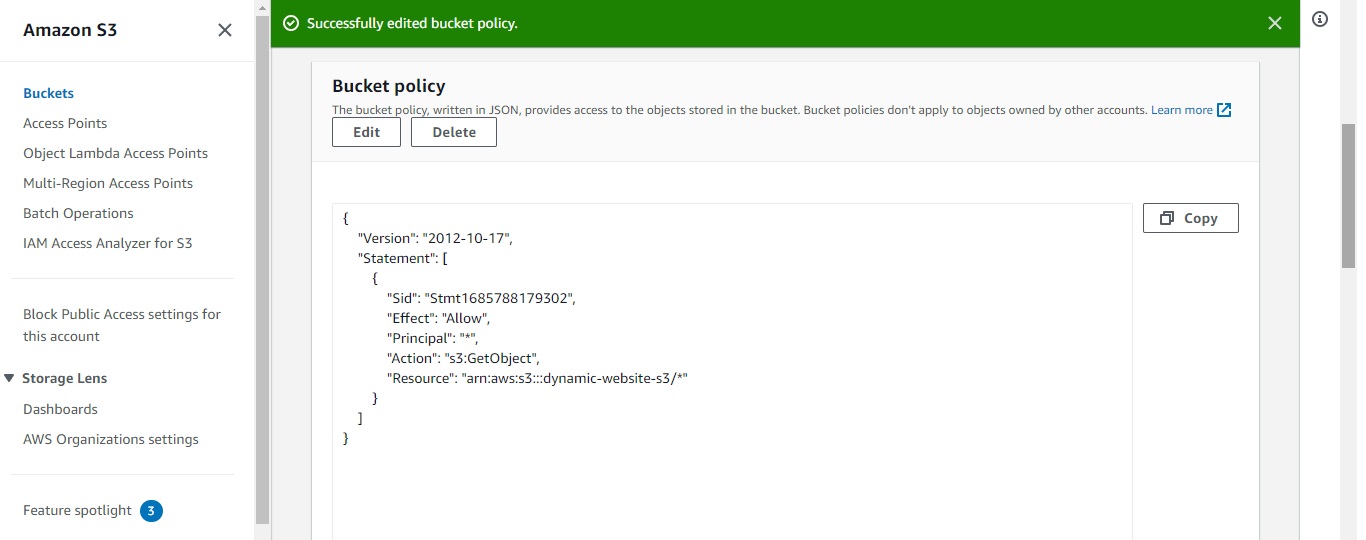
**Go to properties tab and enable static website hosting, then in Index document typed profile.html and click save changes.**

[](https://user-images.githubusercontent.com/119833411/243079658-5354f0e7-975d-4ba9-9595-94386faf2f38.jpg)

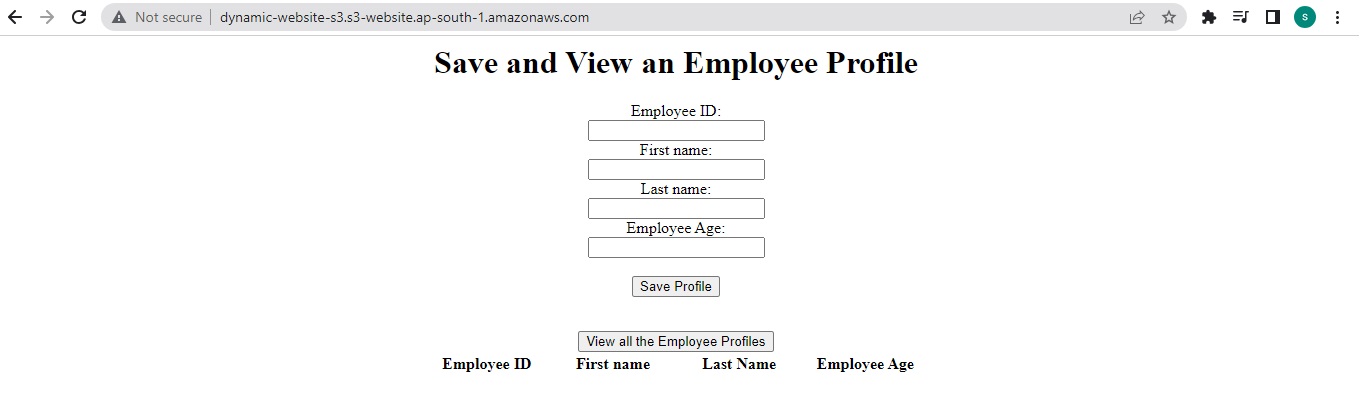
**Go to Permissions tab and edit Block public access (bucket settings) , UnBlock all public access and save settings.**

[](https://user-images.githubusercontent.com/119833411/243079856-4d1ae8a6-f6ef-4dd7-838e-0eb9ee460054.jpg)

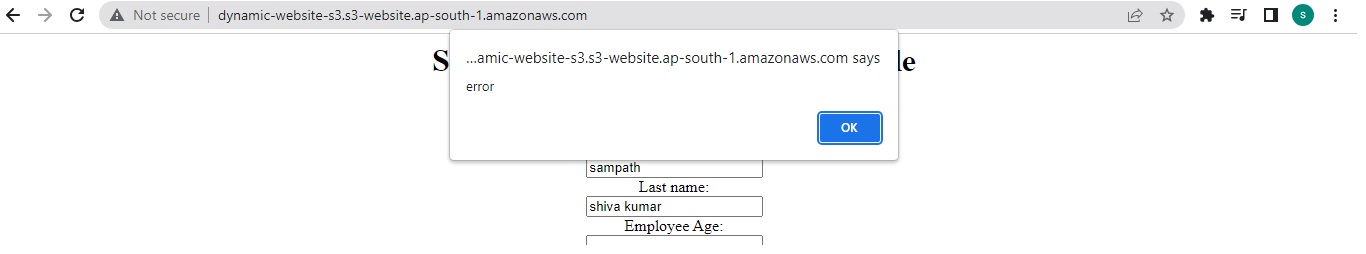
**Edit Bucket policy in Permissions tab to make contents of S3 Bucket public**

[](https://user-images.githubusercontent.com/119833411/243080690-d6eed5f3-3929-473e-94c9-27716d50c331.jpg)

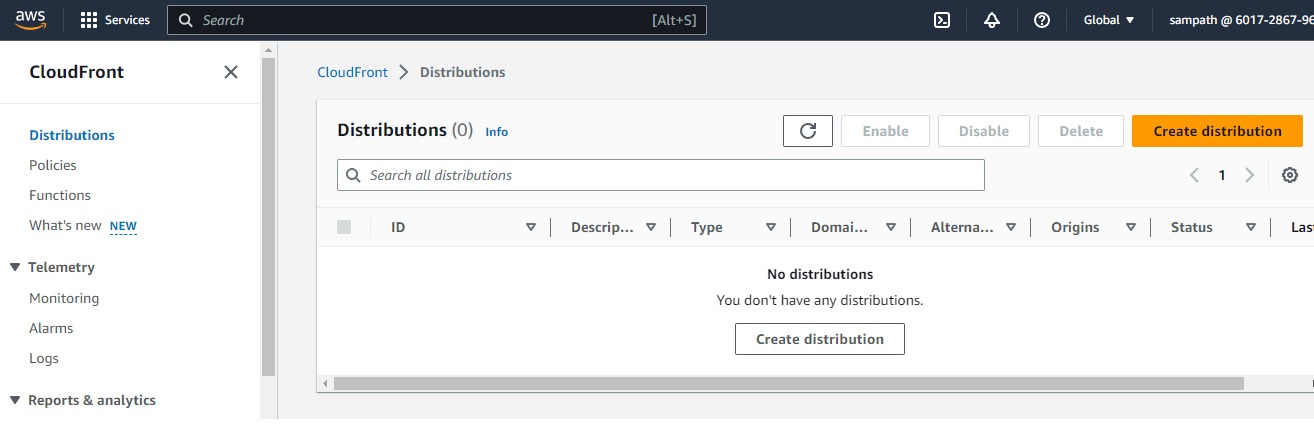
**Now we can access the url of S3 Bucket static website.**

[](https://user-images.githubusercontent.com/119833411/243080812-75b48425-7899-41a7-b529-5641ae1ecc8e.jpg)

**As of now it can't save your data as we have not connected it to any database**

[](https://user-images.githubusercontent.com/119833411/243080941-f7bb620a-03cc-42ac-90ae-6d79b9236828.jpg)

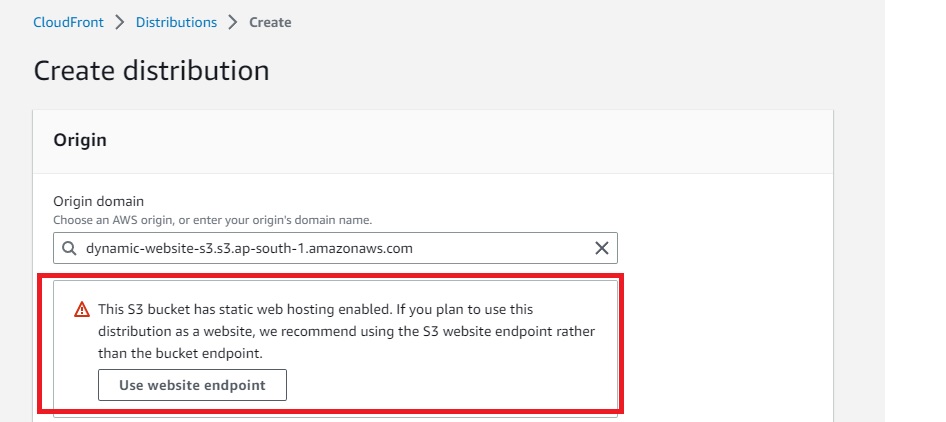
**Lets create a cloudfront and access the website using cloudfront.**

[](https://user-images.githubusercontent.com/119833411/243083039-404117ce-e9e5-4826-8410-68c7ac7f96fd.jpg)

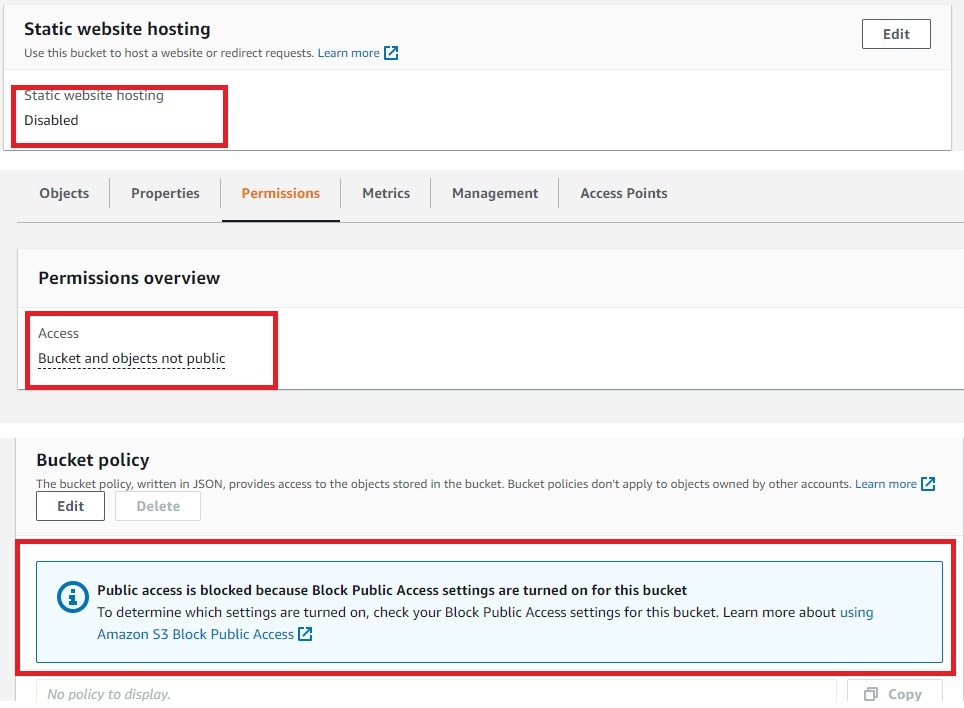
**As we have enabled static-website on S3 bucket it is asking us to go with Bucket website endpoint**

**In our previous post we went with Bucket website endpoint, so now lets go with Bucket endpoint only.**

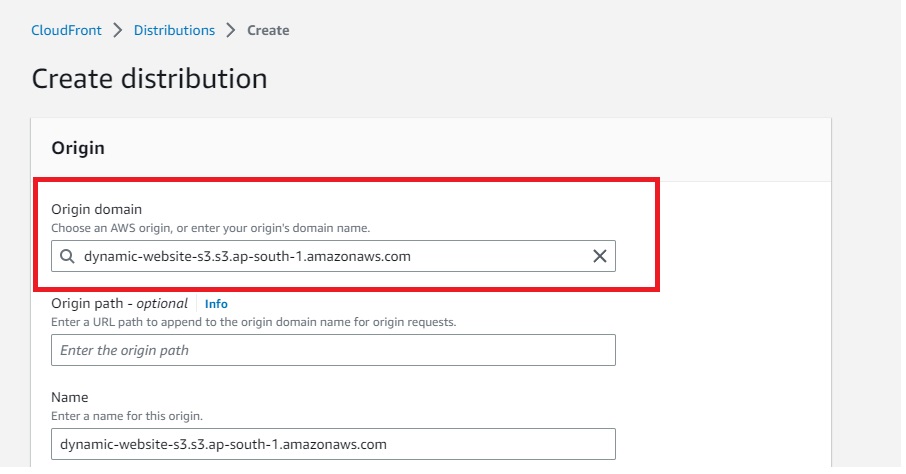
**For that we need to disable Static website hosting, delete Bucket policy and block the public access to bucket**

[](https://user-images.githubusercontent.com/119833411/243083921-e9546495-d500-4d19-ba09-82072e1e57d6.jpg)

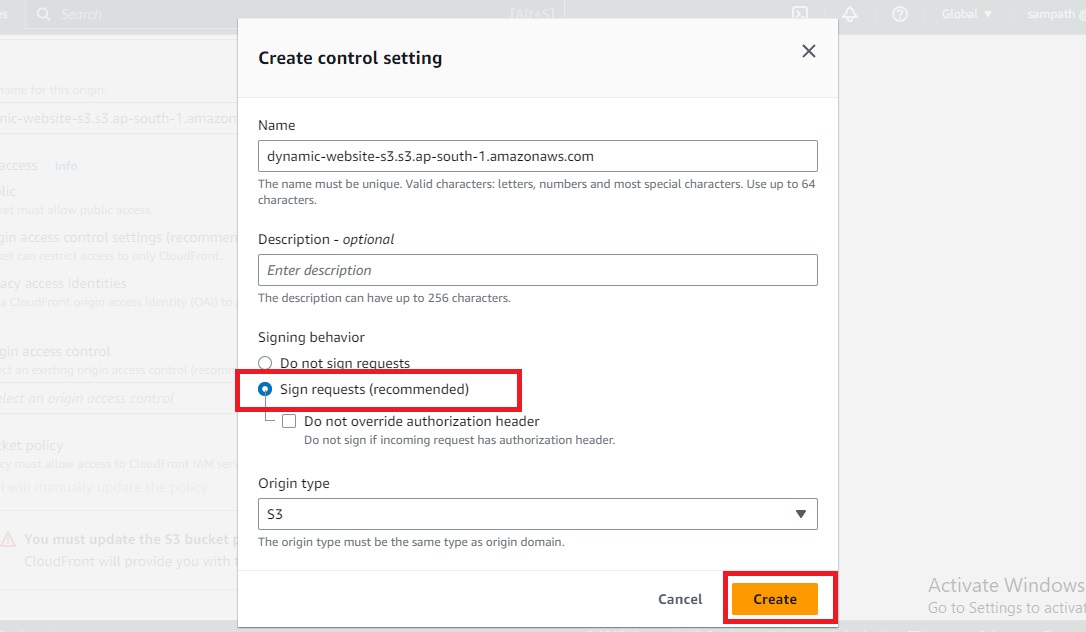
**disable Static website hosting, deleted Bucket policy, block the public access**

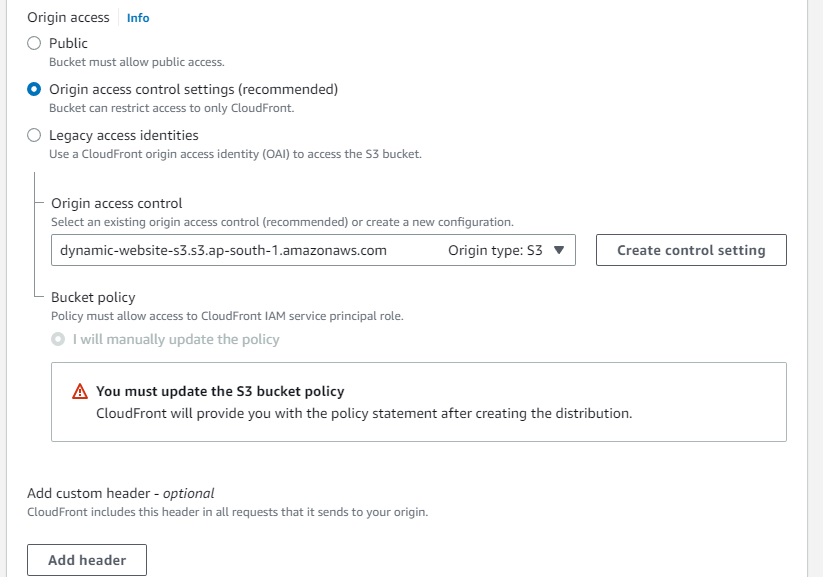
[](https://user-images.githubusercontent.com/119833411/243084404-869d820d-a8e0-497c-a46b-0fa6dcfae5d7.jpg)

**See Now its not showing any popup**

[](https://user-images.githubusercontent.com/119833411/243084580-472b29d0-d867-4d59-9e75-f8bdecfead69.jpg)

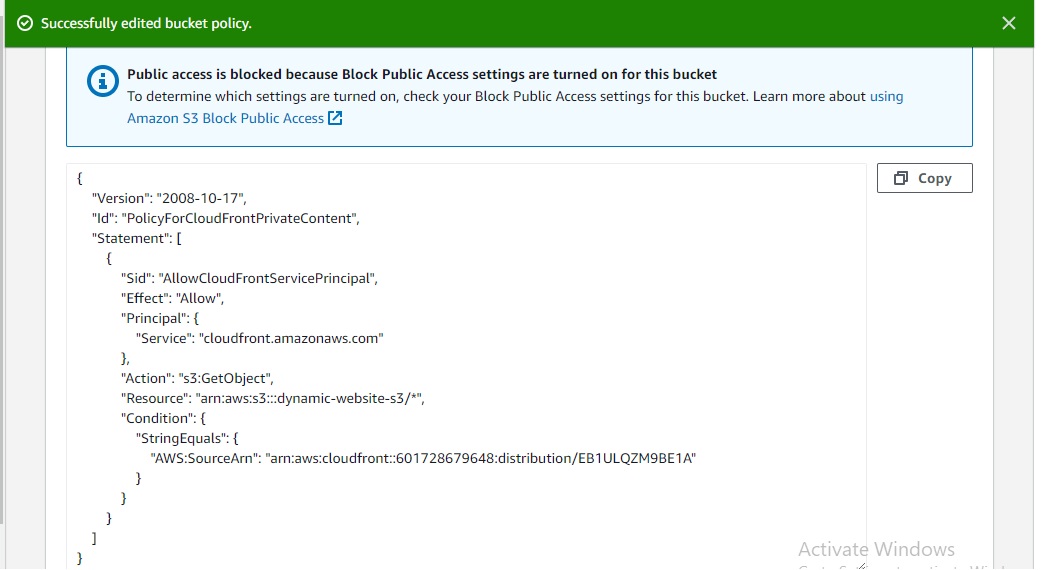
**Select Origin access type to Origin access control settings (recommended) and Create control setting and save**

[](https://user-images.githubusercontent.com/119833411/243084933-0efd72da-25c1-4967-a7b2-e159b4a8ae77.jpg)

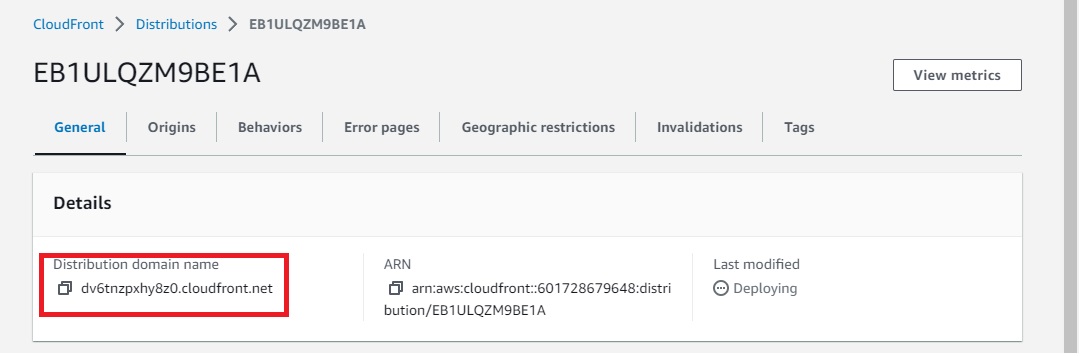
[](https://user-images.githubusercontent.com/119833411/243084974-a285824b-48c0-4775-8f99-125572eeab3a.jpg)

**After Create distribution update the bucket policy to allow cloundfront to access the s3 bucket**

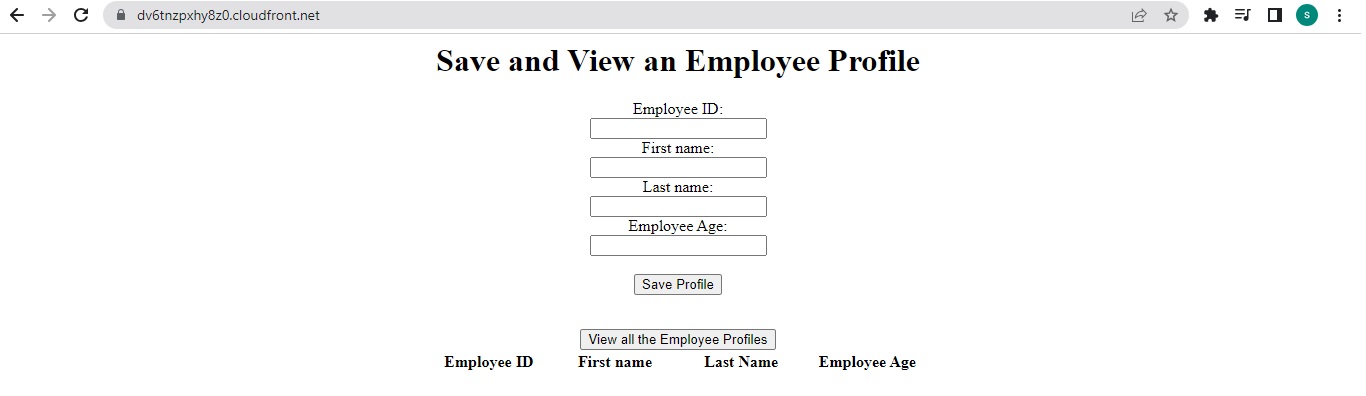
[](https://user-images.githubusercontent.com/119833411/243086335-e257a366-506b-498c-acbc-03a97edb0e51.jpg)

[](https://user-images.githubusercontent.com/119833411/243086669-ed8625b9-fb03-44fd-a40b-4fe7a54e1f58.jpg)

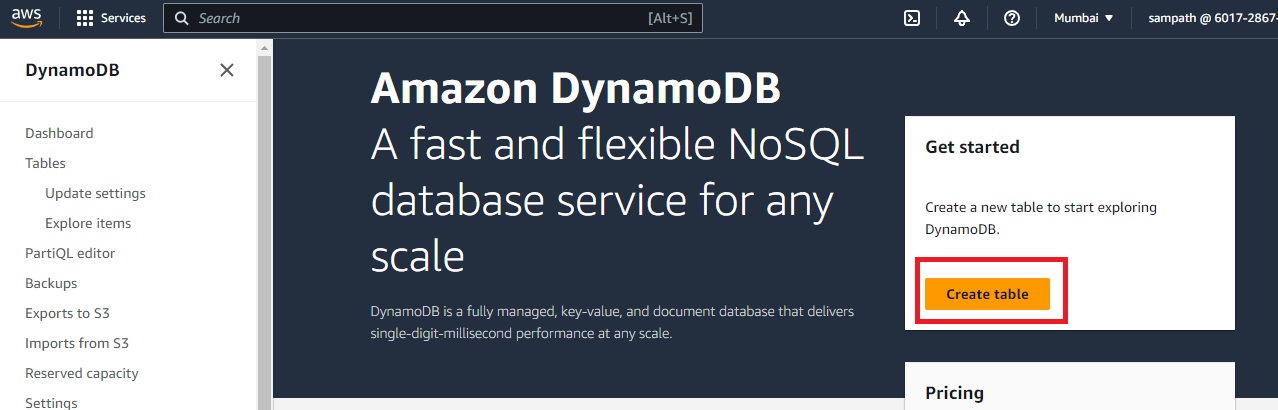
**Now the we can access the S3 Bucket using cloudfront Distribution domain name**

[](https://user-images.githubusercontent.com/119833411/243086982-7992a0d1-52a4-4f12-b643-262c9eceeaac.jpg)

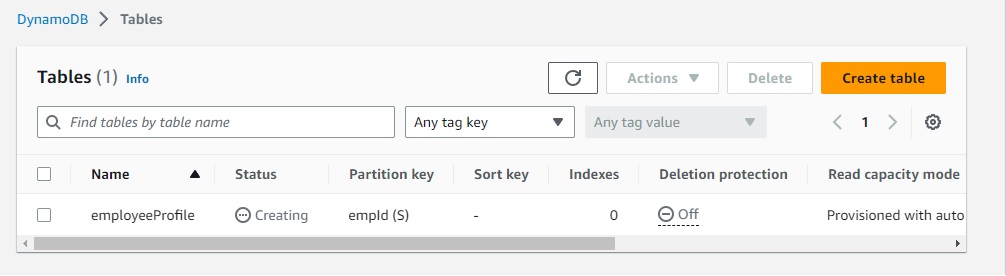
**To access the profile.html with Distribution domain name directly, mention profile.html in Default root object of settings.**

[](https://user-images.githubusercontent.com/119833411/243087038-9eebc4b1-e5af-41b8-8ed4-92987a5de349.jpg)

**Now let's create DynamoDB Table**

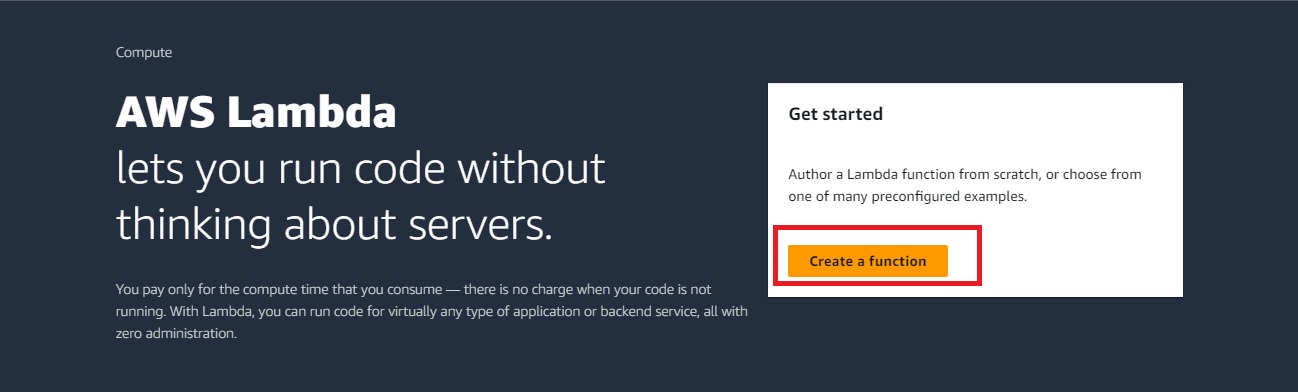
[](https://user-images.githubusercontent.com/119833411/243087680-d20bb142-5684-4f17-bfc2-9b11bbff3c60.jpg)

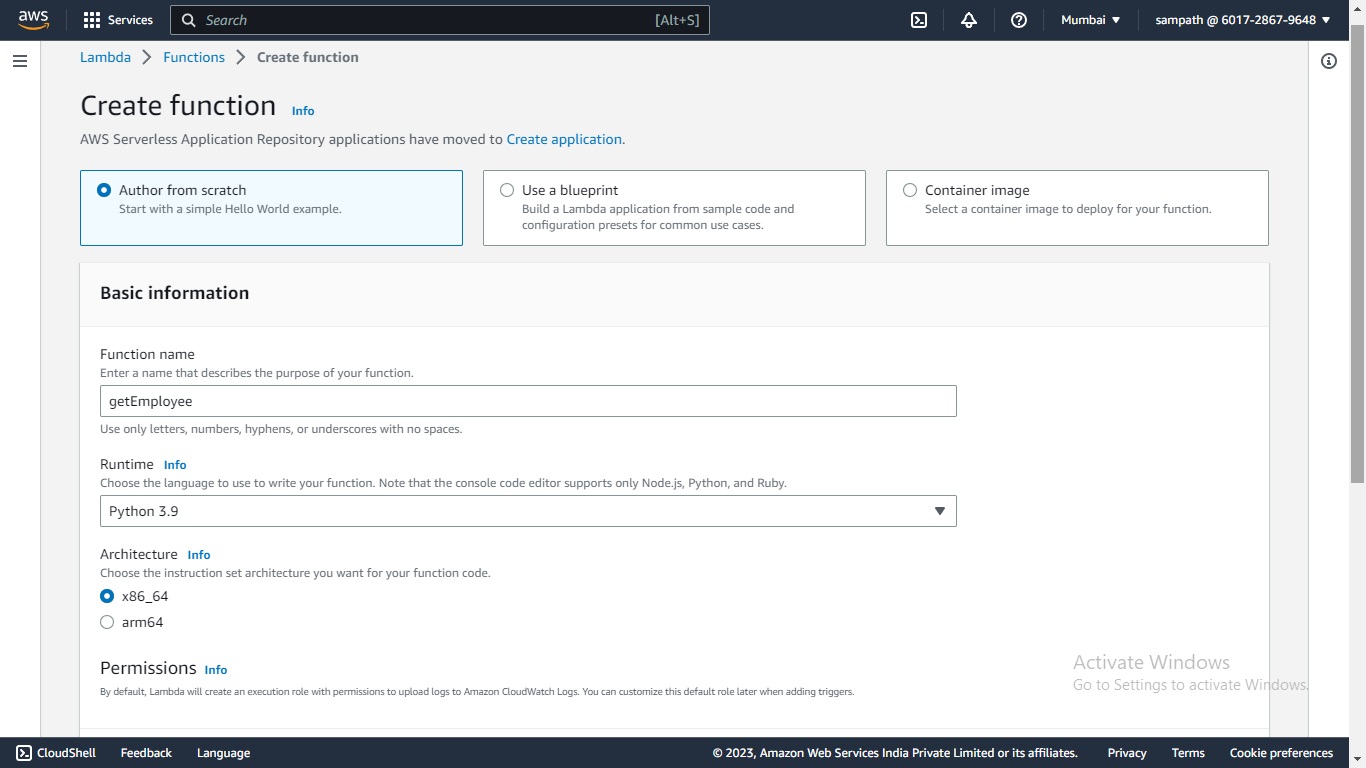
**Enter Table name = employeeProfile, Partition key = empId, tag key/value owner/sampath and create table.**

[](https://user-images.githubusercontent.com/119833411/243087952-81ddbe76-02cf-42f4-9bf4-13b75334b11e.jpg)

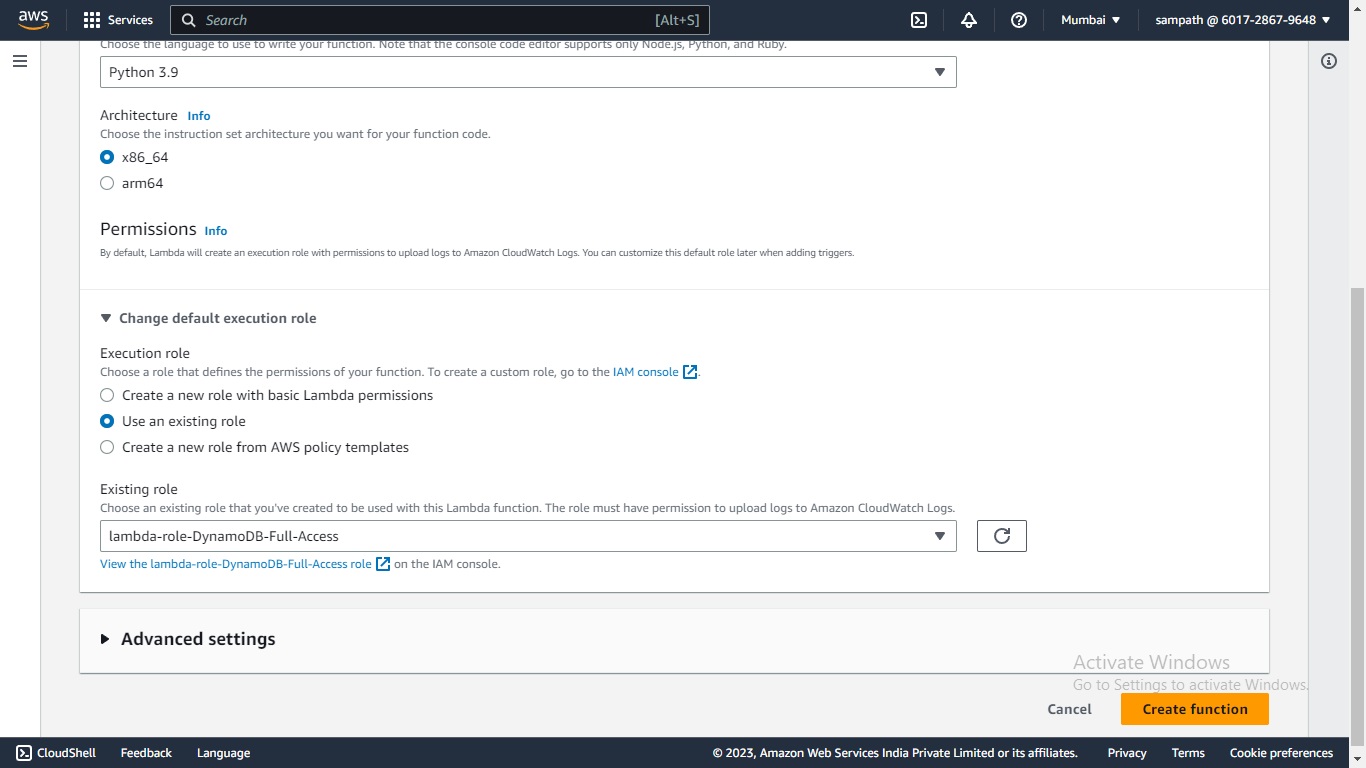
**While the DynamoDB table is creating, lets create Lambda functions**

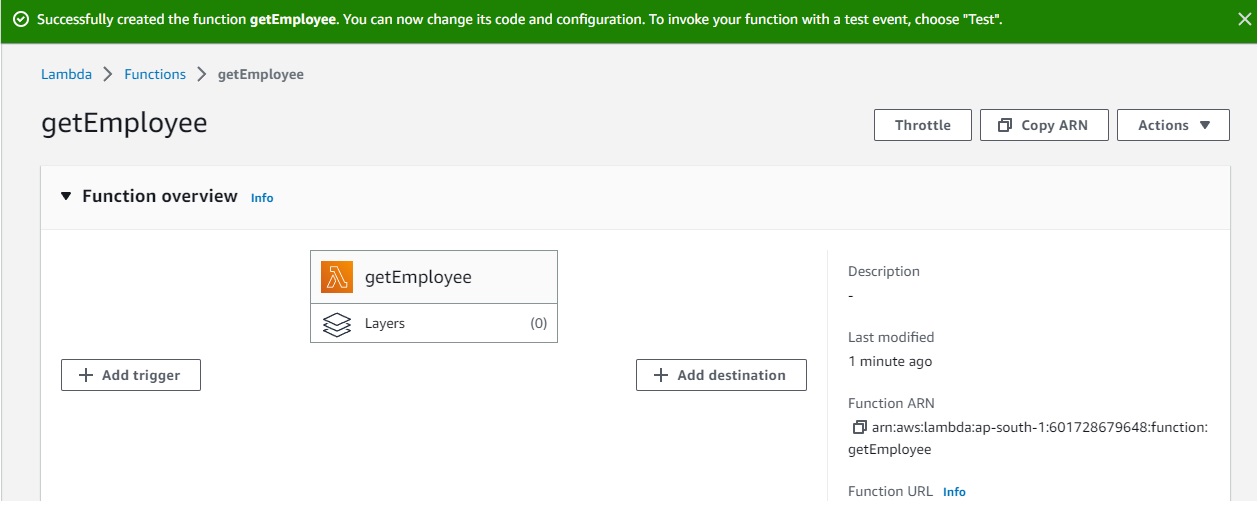
**Lets create a lambda function to get data from DynamoDB**

[](https://user-images.githubusercontent.com/119833411/243088050-7ded8597-4351-4ae9-a7cb-014a56cfbfc8.jpg)

[](https://user-images.githubusercontent.com/119833411/243088796-3a02bb2e-134e-425f-adec-a8ce78f827b5.jpg)

**Create a lambda role to access DynamoDB and select that role to lambda function**

[](https://user-images.githubusercontent.com/119833411/243088939-ee0a9128-5e88-4183-aae6-0bb58948c308.jpg)

[](https://user-images.githubusercontent.com/119833411/243089084-07c3d468-6e8f-42bd-a4d5-ef2293c76218.jpg)

**Write a small code for lambda to function and deploy**

import json

import boto3

def lambda\_handler(event, context):

dynamodb = boto3.resource('dynamodb', region\_name='ap-south-1')

table = dynamodb.Table('employeeProfile')

response = table.scan()

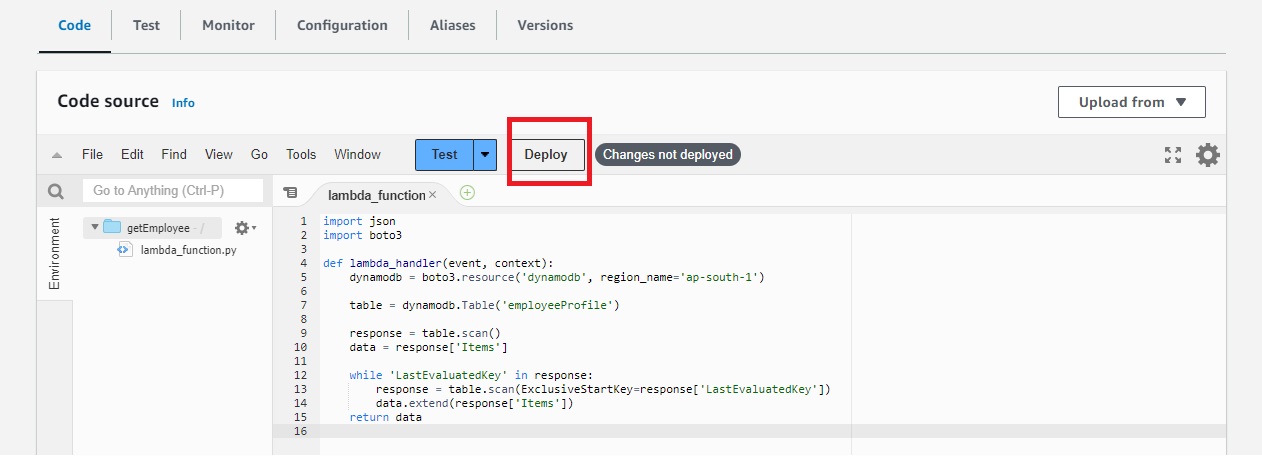
data = response['Items']

while 'LastEvaluatedKey' in response:

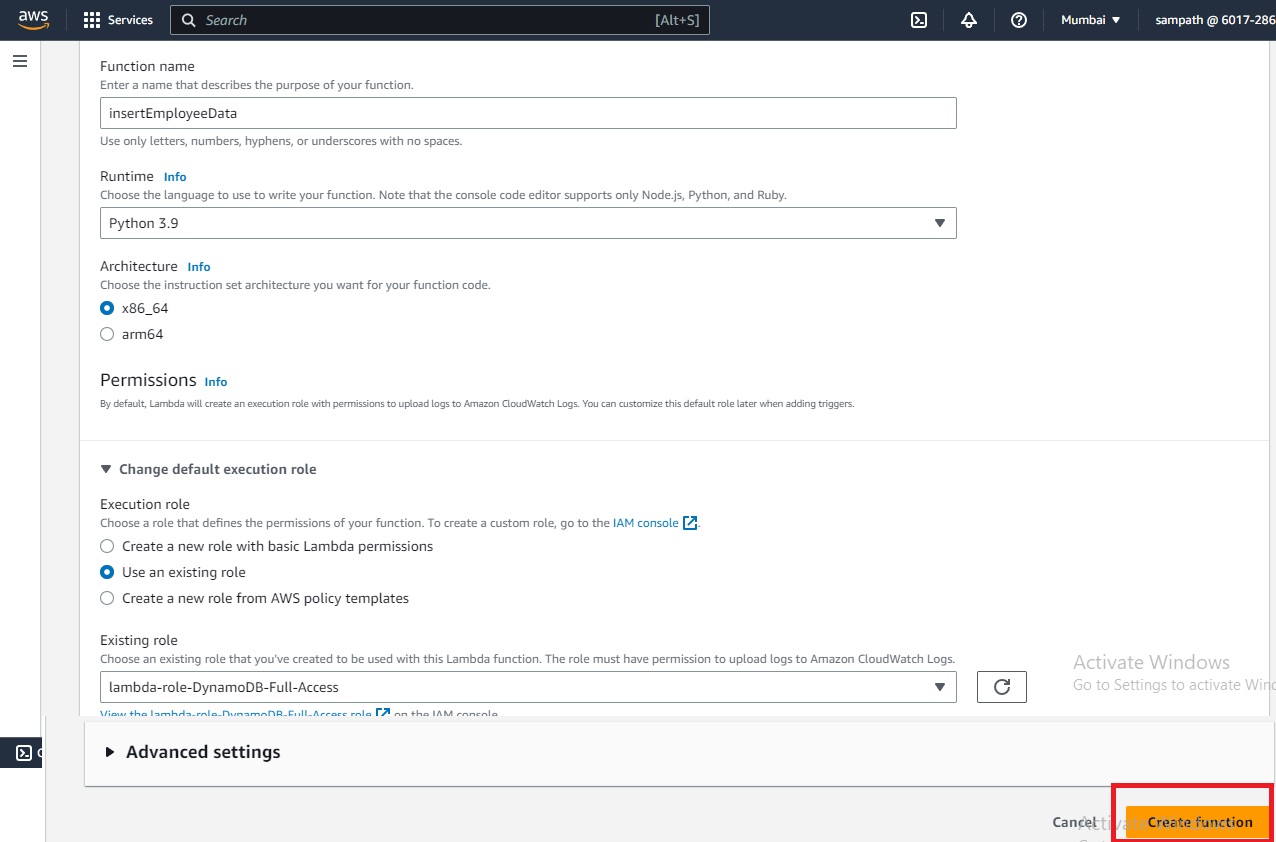
response = table.scan(ExclusiveStartKey=response['LastEvaluatedKey'])

data.extend(response['Items'])

return data

[](https://user-images.githubusercontent.com/119833411/243089756-fb990d58-8764-4541-9d48-8426c0f54700.jpg)

**Lets create a 2nd lambda function to insert data into DynamoDB**

[](https://user-images.githubusercontent.com/119833411/243090316-f19f8847-f17a-4ced-a24f-340082708b90.jpg)

**Copy and past the code in lambda function**

import json

import boto3

# create a DynamoDB object using the AWS SDK

dynamodb = boto3.resource('dynamodb')

# use the DynamoDB object to select our table

table = dynamodb.Table('employeeProfile')

# define the handler function that the Lambda service will use as an entry point

def lambda\_handler(event, context):

# extract values from the event object we got from the Lambda service and store in a variable

firstname = event['empFirstName']

id=event['empId']

lastname=event['empLastName']

age=event['empAge']

# write name and time to the DynamoDB table using the object we instantiated and save response in a variable

response = table.put\_item(

Item={

'empId': id,

'empAge':age,

'empFirstName':firstname,

'empLastName':lastname

})

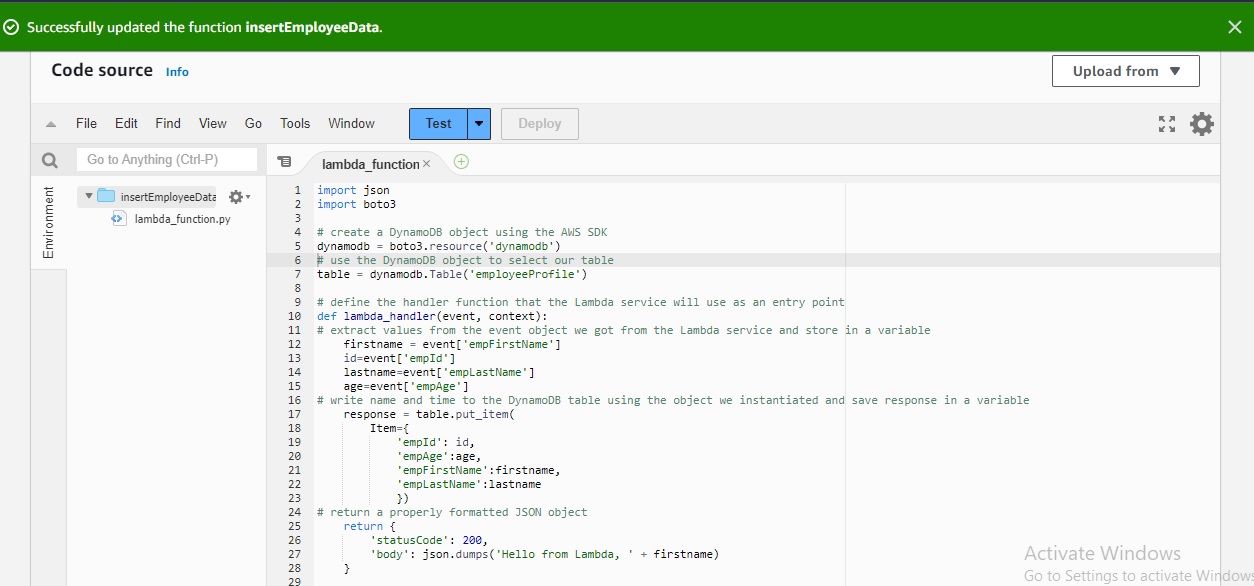
# return a properly formatted JSON object

return {

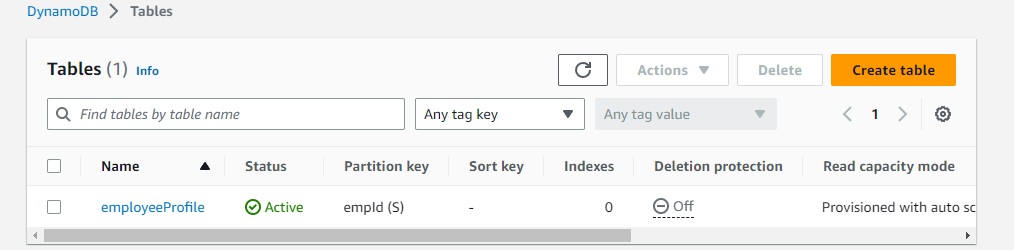
'statusCode': 200,

'body': json.dumps('Hello from Lambda, ' + firstname)

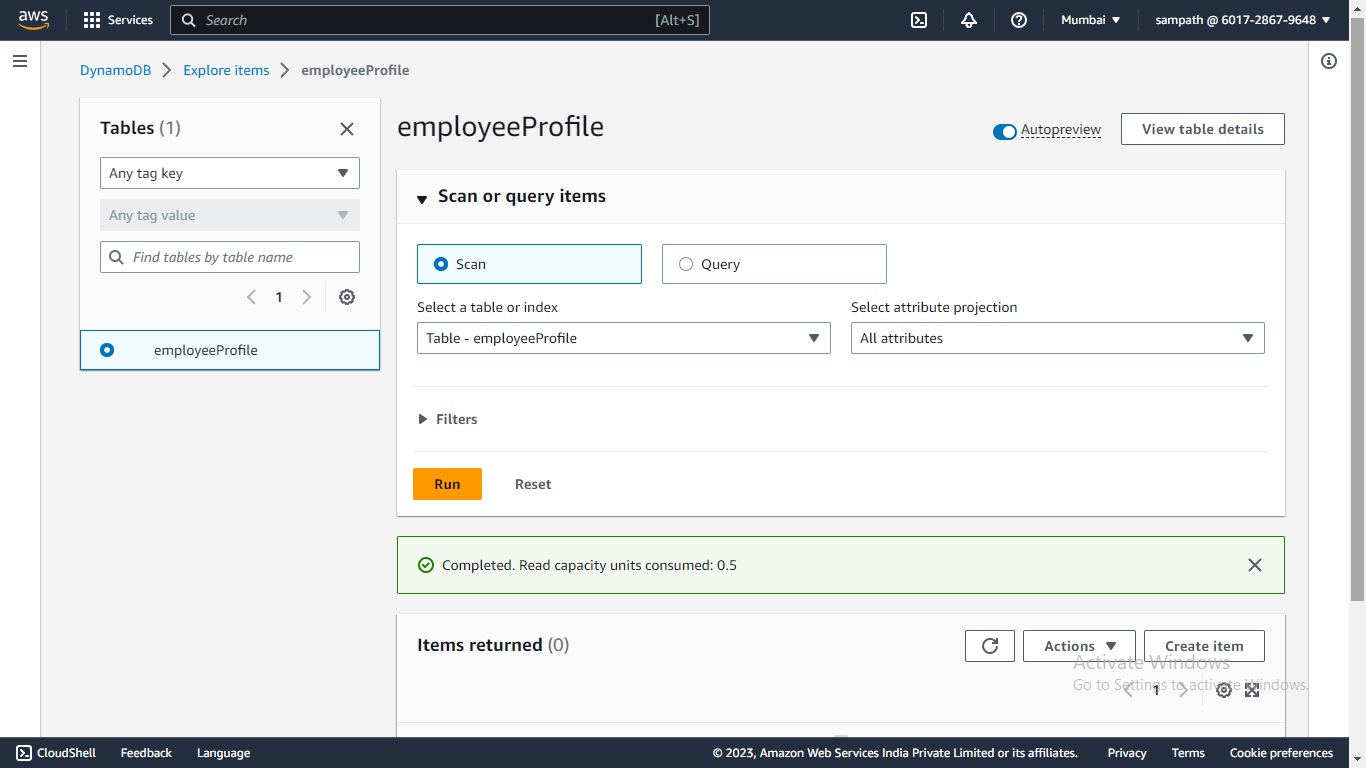
}

**Deploy** [](https://user-images.githubusercontent.com/119833411/243091021-152b272d-2562-41ed-befe-6553b45d9eb4.jpg)

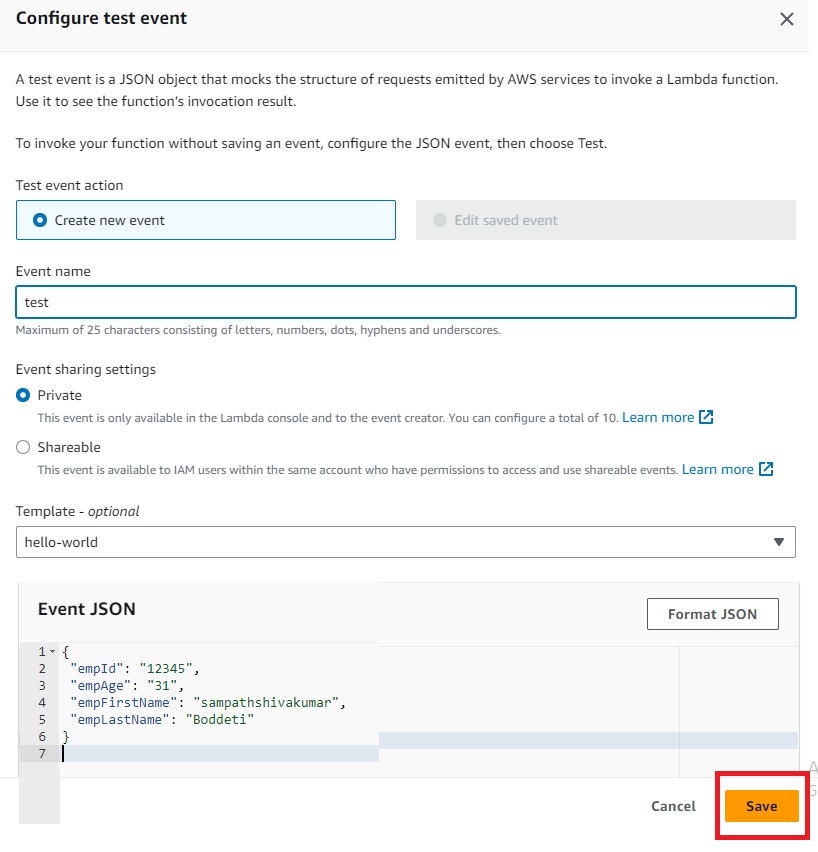
**By now our DynamoDB Table will be ready**

[](https://user-images.githubusercontent.com/119833411/243091384-9095913e-f945-45e9-885f-c4ef880dc819.jpg)

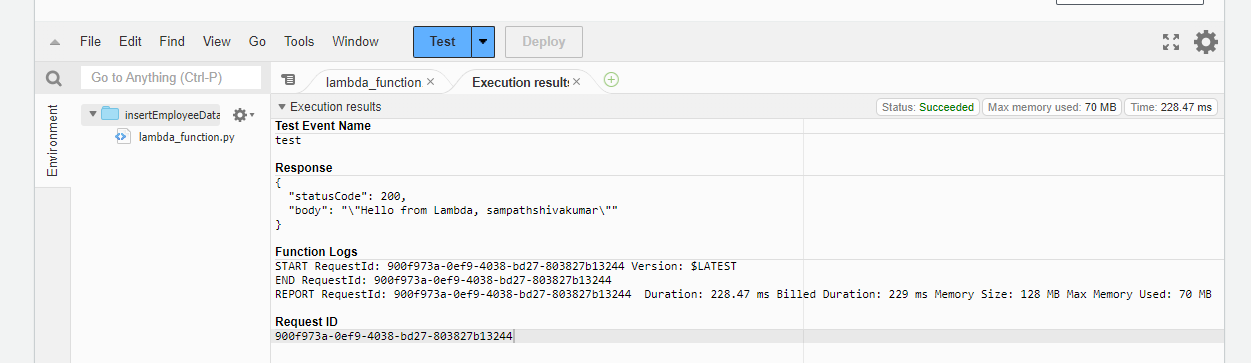
**Lets go inside DynamoDB table and Explore table items, you can see anything**

[](https://user-images.githubusercontent.com/119833411/243091706-815067bd-bb94-44f9-b12d-9f690995585d.jpg)

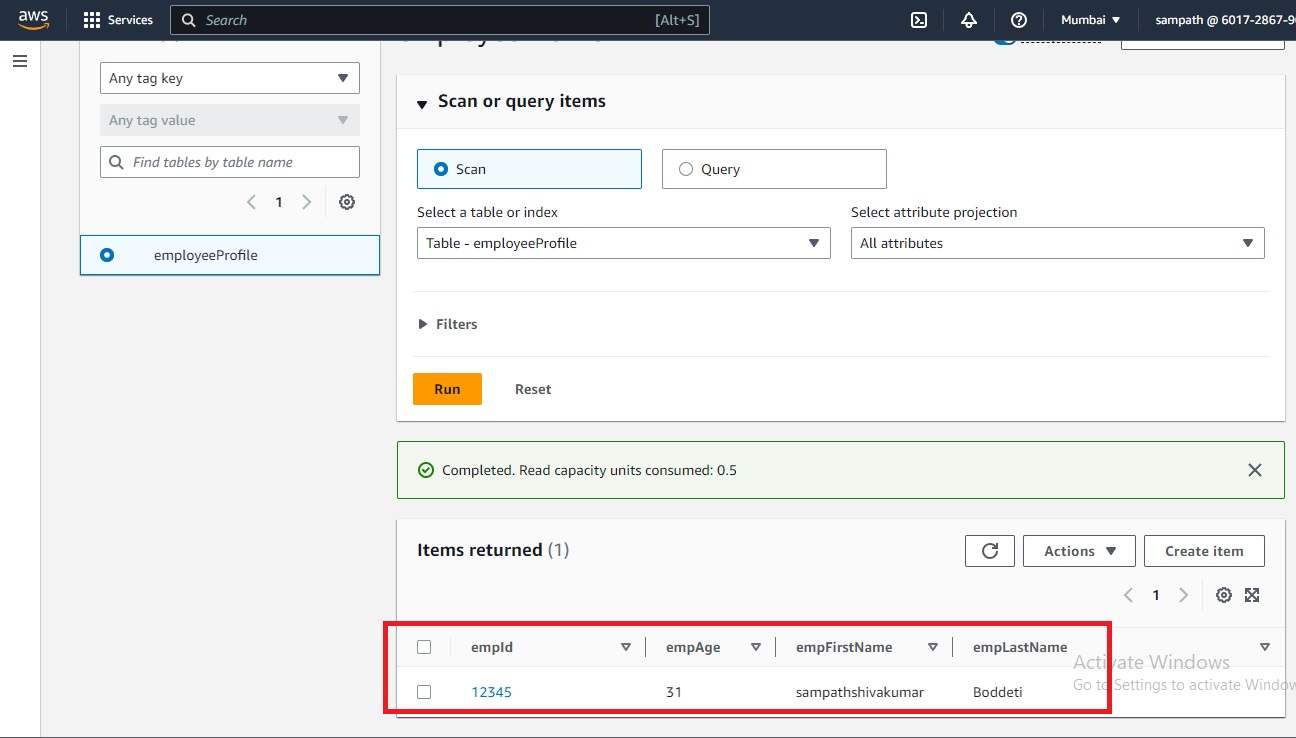
**Now lets test our 2nd lambda function, try to insert some data into DynamoDB using lambda function**

[](https://user-images.githubusercontent.com/119833411/243092189-2d2eb8fc-451e-47df-9ebb-2a7bb5a4897d.jpg)

**Test successful**

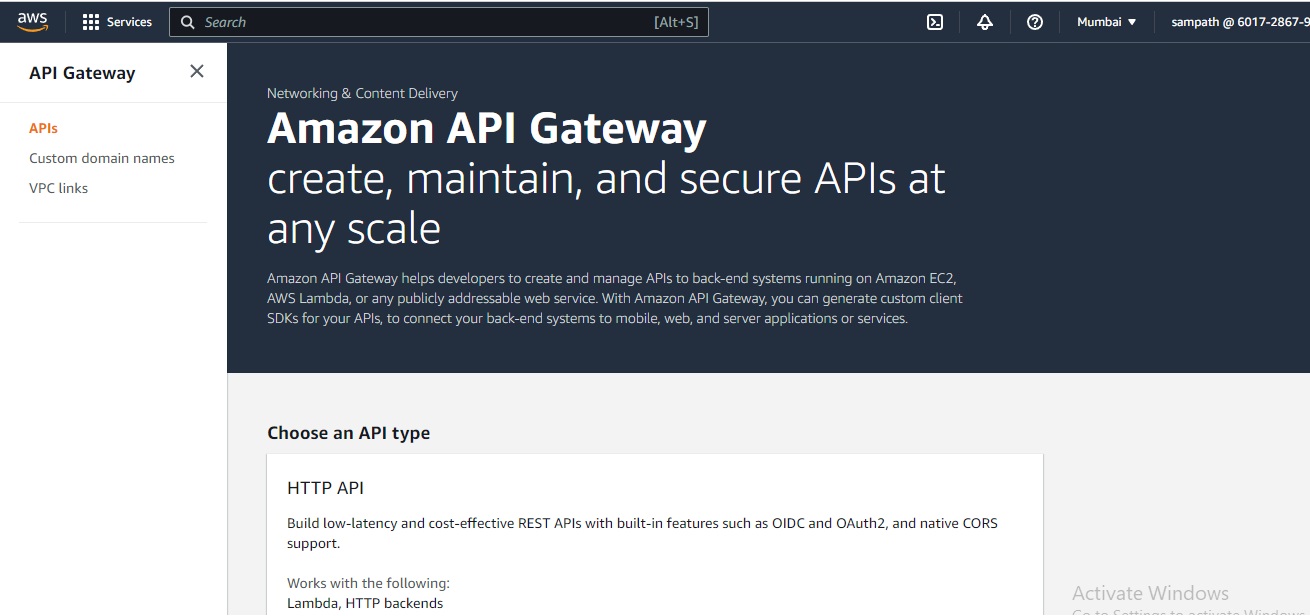
[](https://user-images.githubusercontent.com/119833411/243092423-312c0944-bec4-47cd-9f3e-04d22b782891.png)

**Lets look in to DynamoDB table**

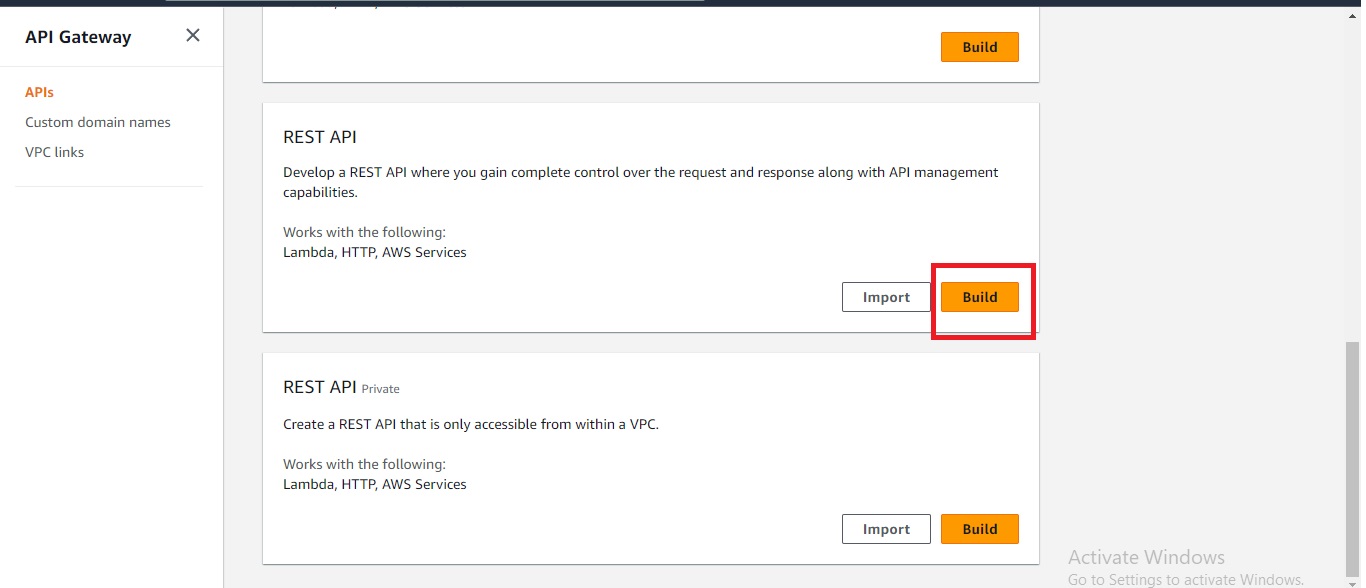
[](https://user-images.githubusercontent.com/119833411/243092638-569c9a36-42cb-4985-992a-367e33753364.jpg)

**Both lambda functions are working fine with DynamoDB, Our Backend is ready**

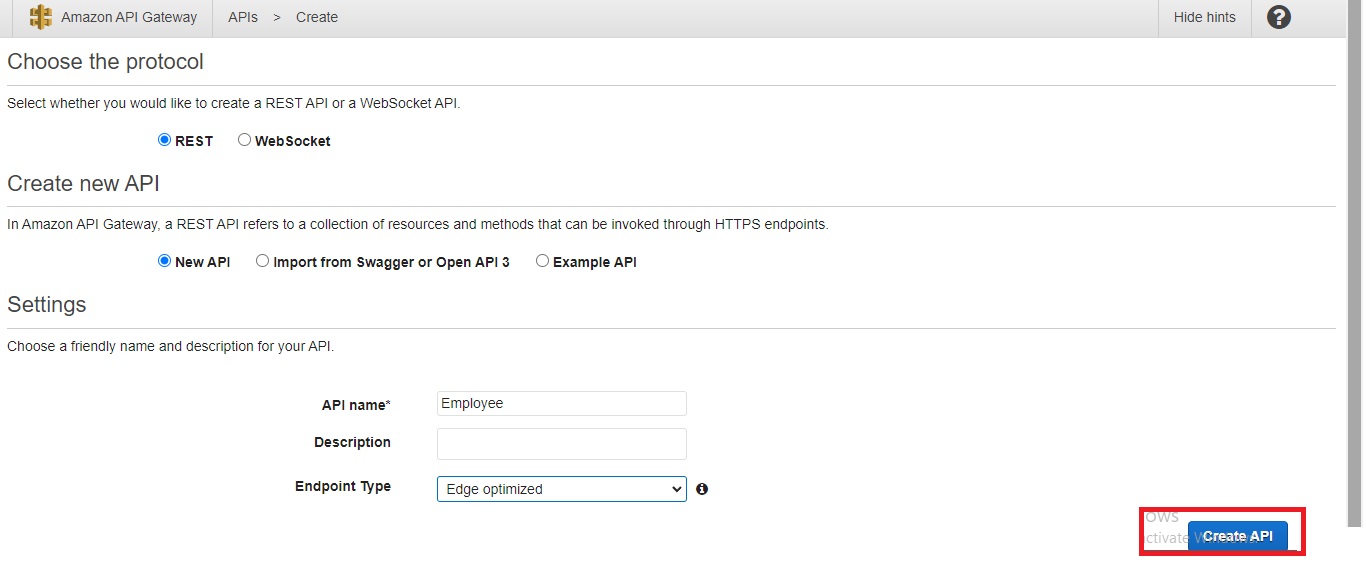
**Now we will integrate both Frount-End and Back-End part with help of API-Gateway.**

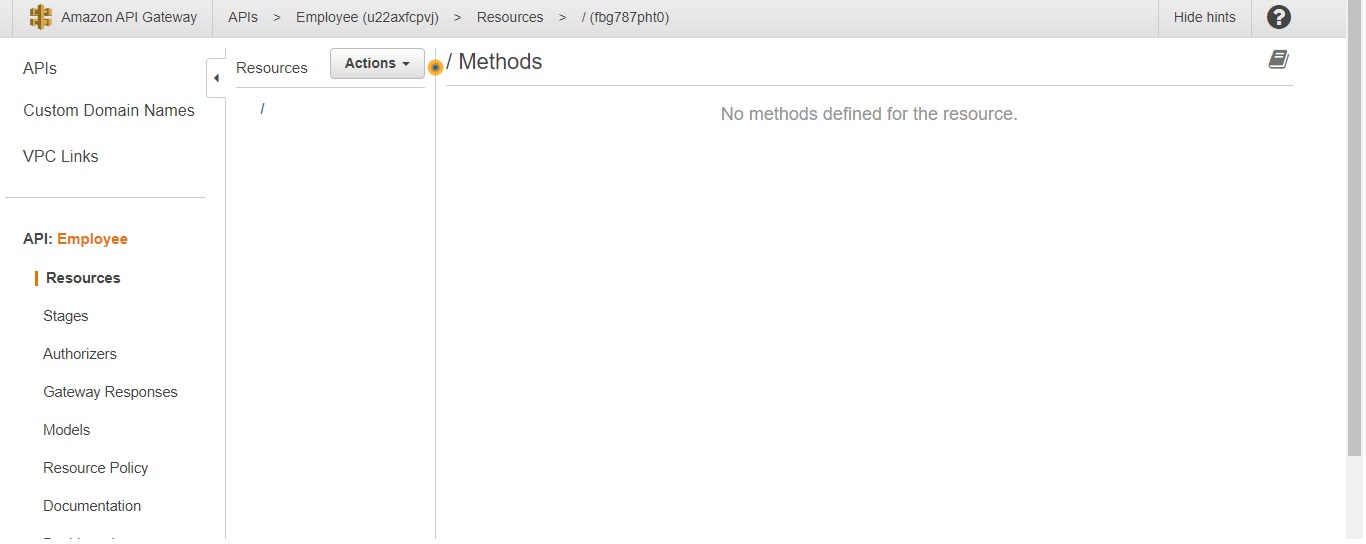
[](https://user-images.githubusercontent.com/119833411/243094241-3e9926a4-cf81-4289-bf1b-e790a08024b8.jpg)

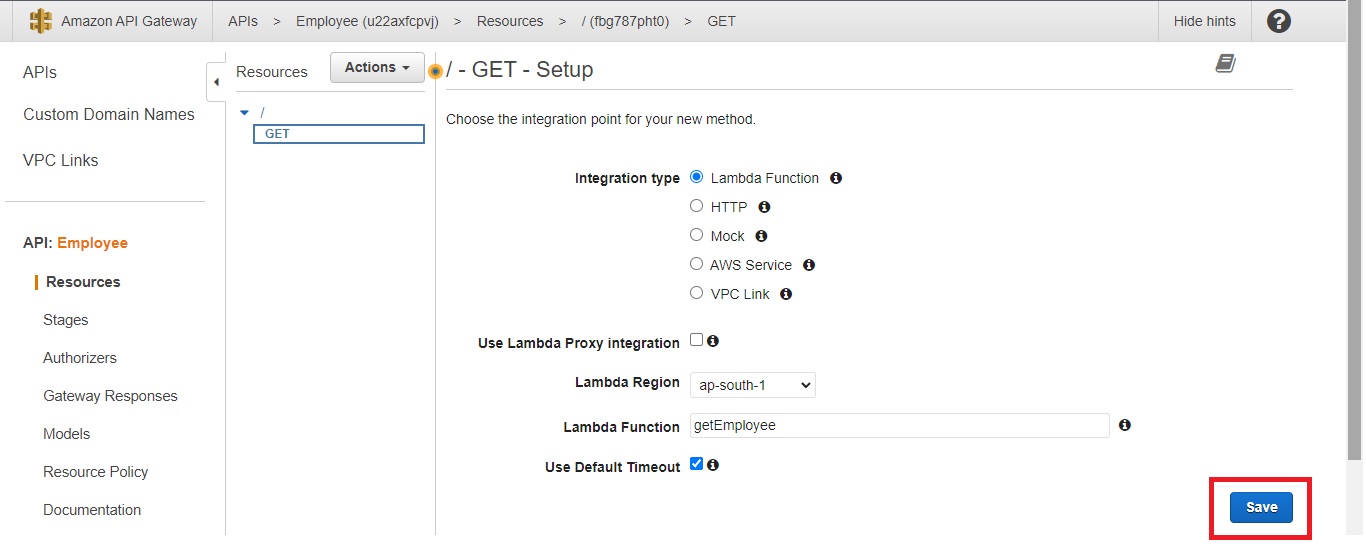
**We will create a REST-API**

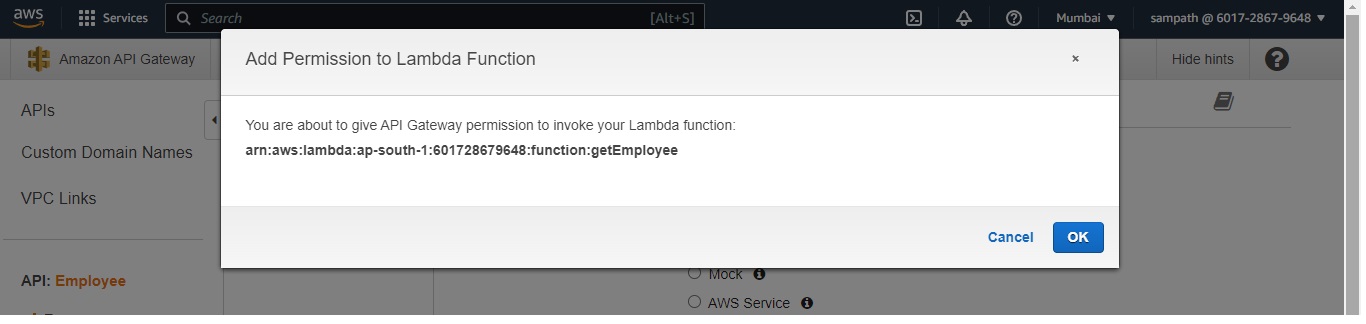
[](https://user-images.githubusercontent.com/119833411/243095098-443824a8-8cce-4b79-be85-f53f9410471f.jpg)

**Select New API, give API name and Endpoint Type and click create API.**

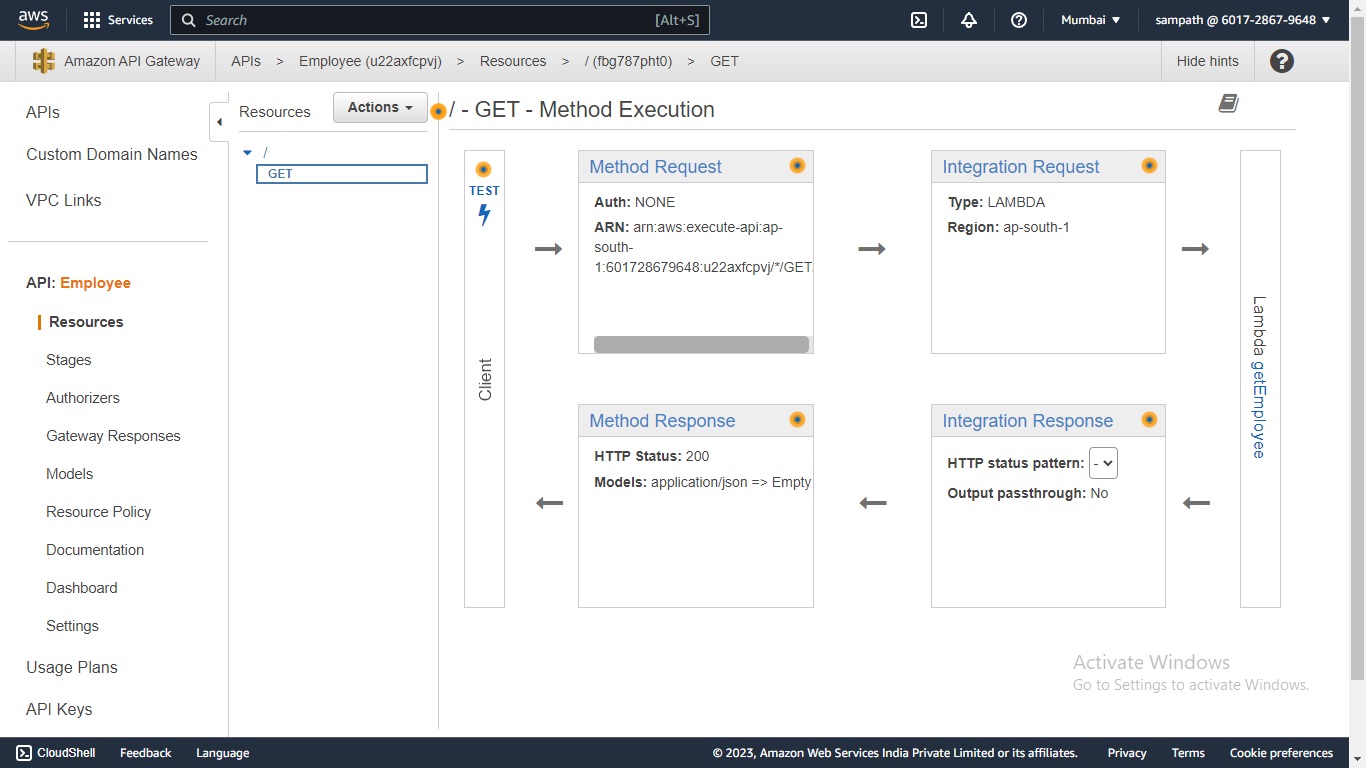
[](https://user-images.githubusercontent.com/119833411/243096277-c8236789-356a-491b-b1d8-5d75a26c0c76.jpg)

[](https://user-images.githubusercontent.com/119833411/243096857-53712a0b-959c-4921-bf55-d173910dcb20.jpg)

[](https://user-images.githubusercontent.com/119833411/243097005-94b1d528-9fe3-49e9-a2f2-2764203e3e25.jpg)

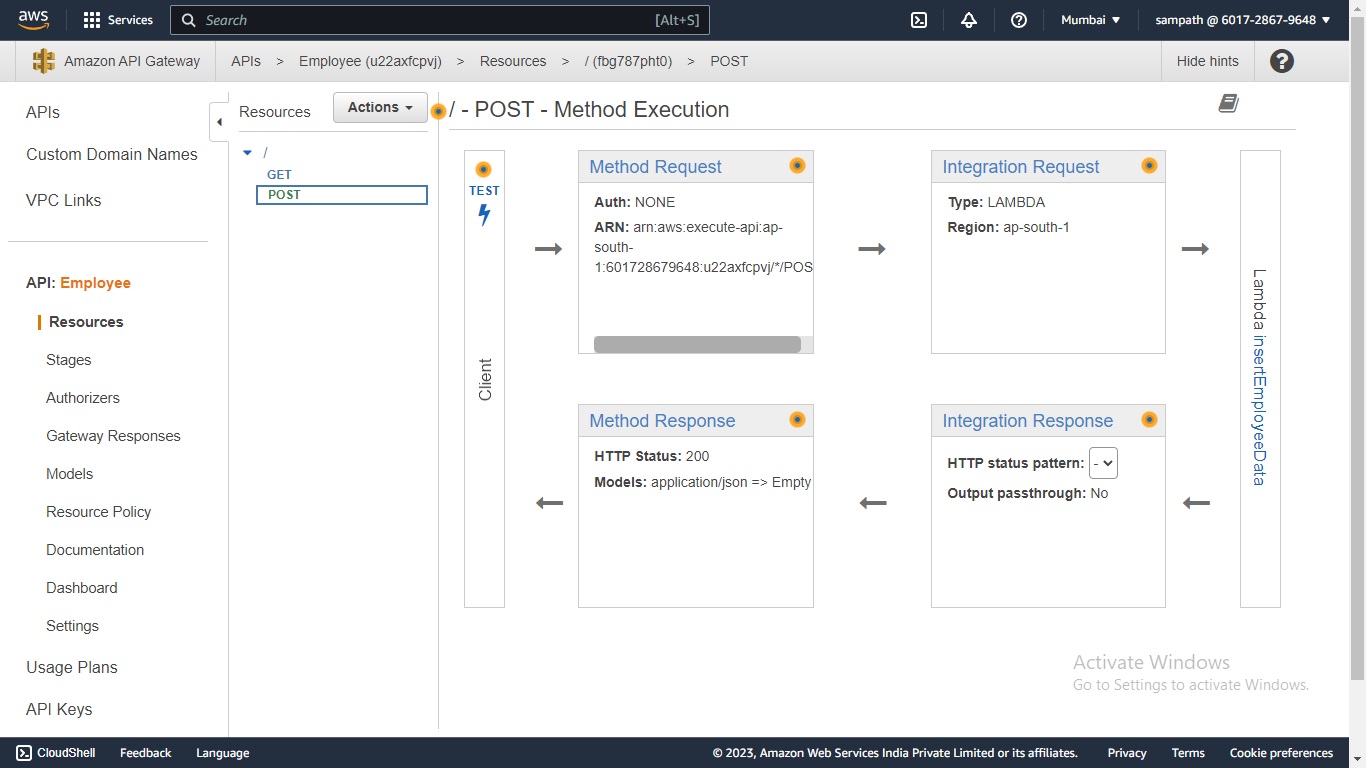
[](https://user-images.githubusercontent.com/119833411/243097052-65157a4a-7284-40dc-92c3-359828465572.jpg)

**Now my Get method is done**

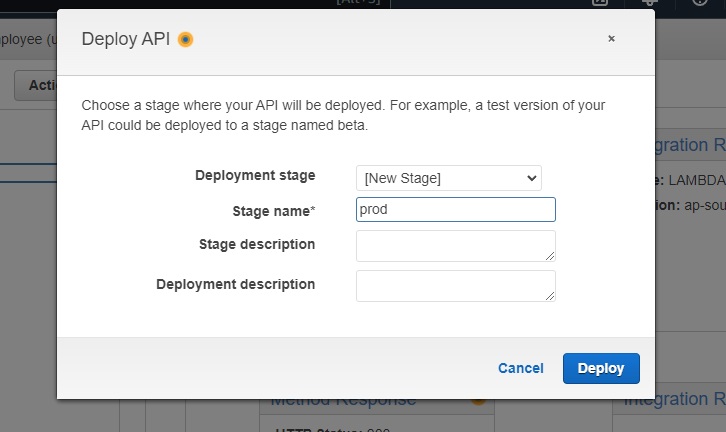
[](https://user-images.githubusercontent.com/119833411/243097190-6ecd643d-9117-4499-92ba-082060d594ee.jpg)

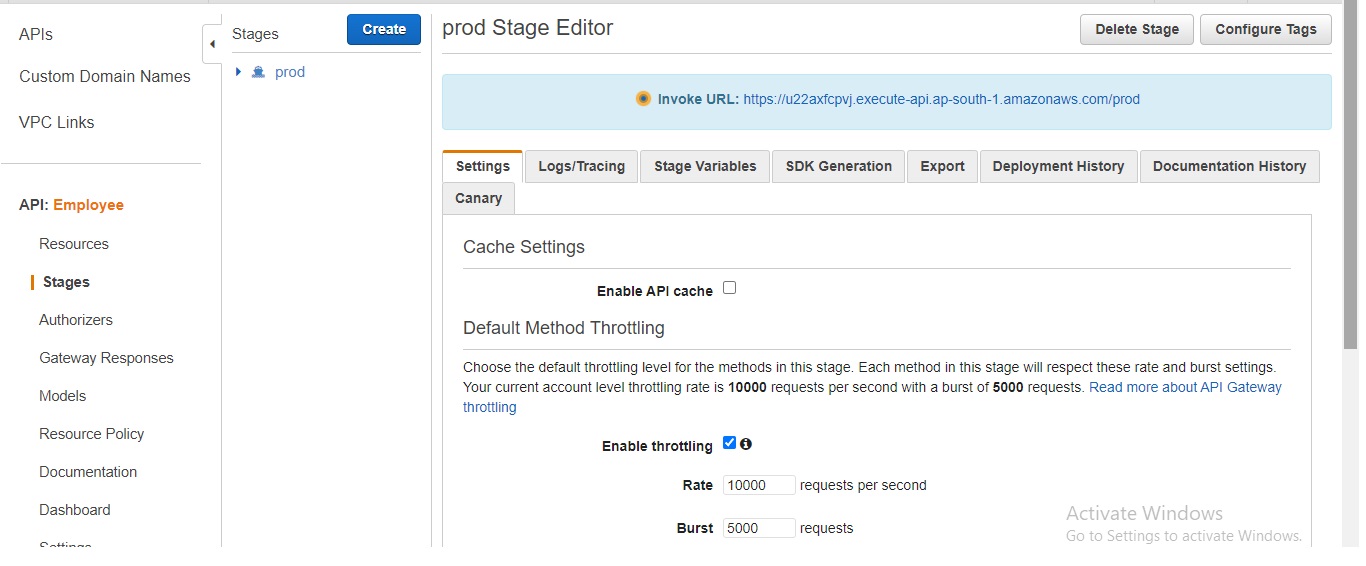
**Now lets create POST Method**

[](https://user-images.githubusercontent.com/119833411/243097306-38a09f31-4a82-44b8-848b-87821015e403.jpg)

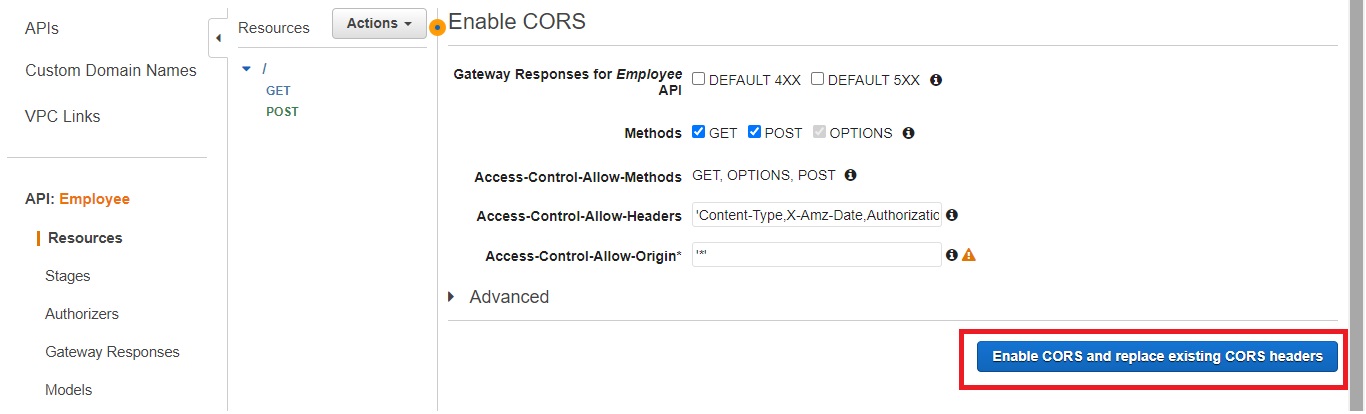
[](https://user-images.githubusercontent.com/119833411/243097356-e39657d7-6bf5-403a-ad4f-bea4f1542317.jpg)

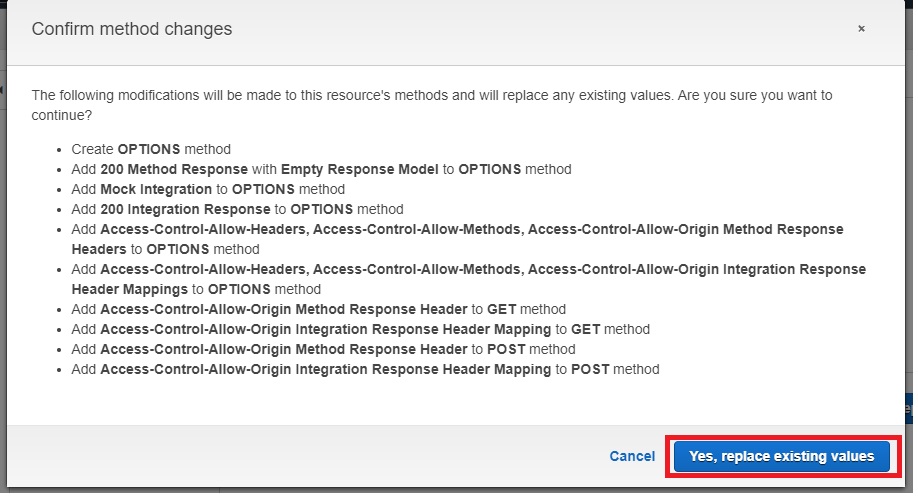
**Now we have to Deploy the API Gateway**

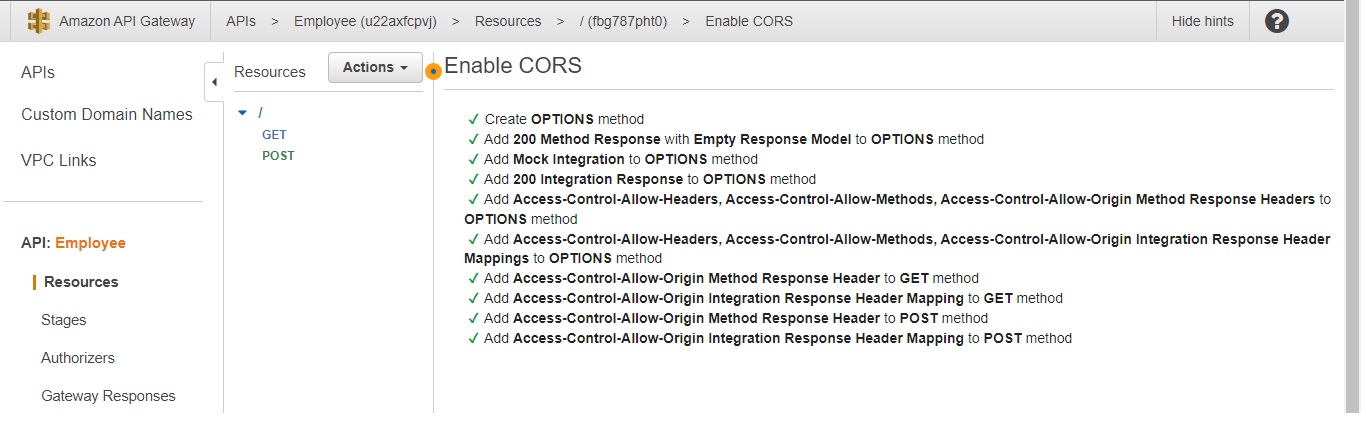
[](https://user-images.githubusercontent.com/119833411/243097585-8429c61e-7756-472e-a8b4-0f24f982242b.jpg)

[](https://user-images.githubusercontent.com/119833411/243097669-58e55890-3217-46be-aa66-d66aadcf229c.jpg)

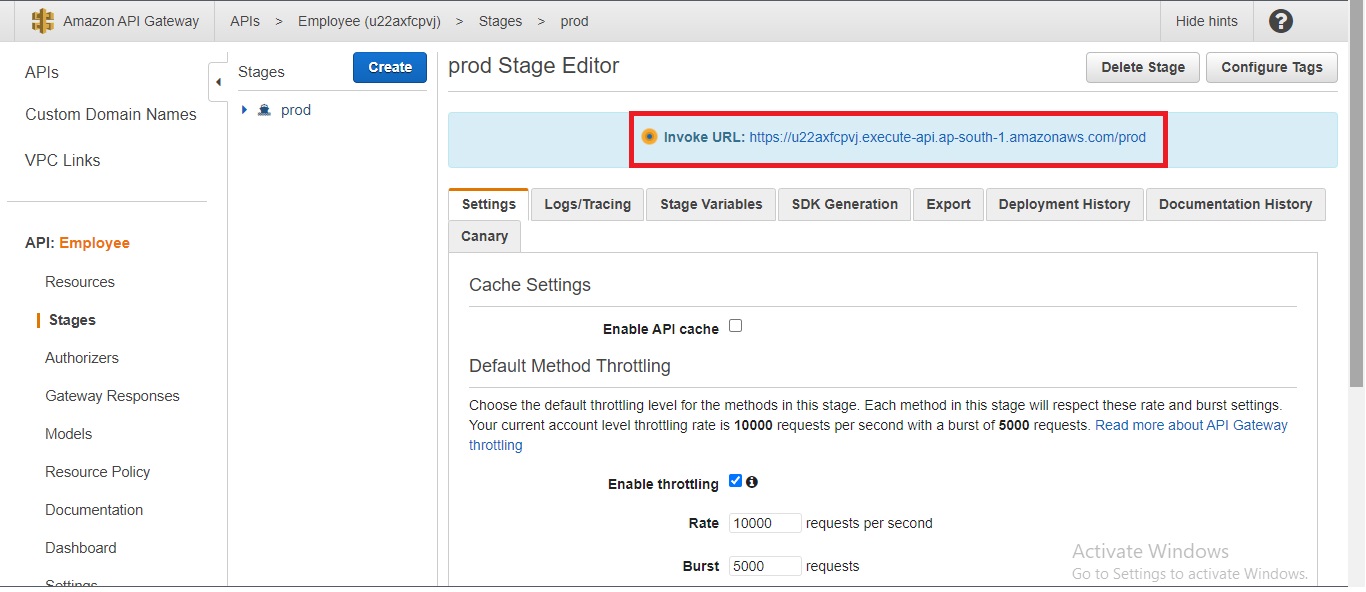
**Now lets enable CORS**

[](https://user-images.githubusercontent.com/119833411/243097725-56851fc0-4856-41a8-93d3-fe222837b2be.jpg)

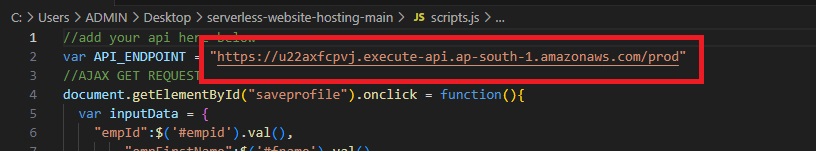
[](https://user-images.githubusercontent.com/119833411/243097820-3c5b8ce0-4d62-4ca1-9751-e724eacaf288.jpg)

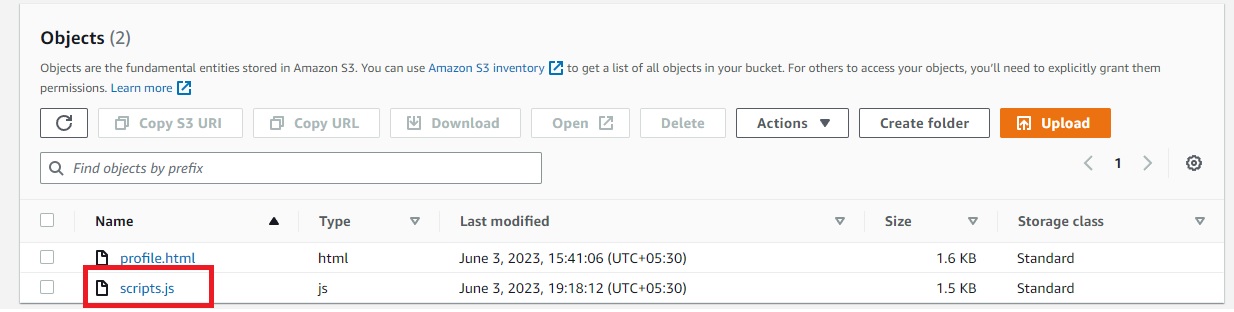
[](https://user-images.githubusercontent.com/119833411/243097875-7e221940-2c48-4d3a-a6ee-4d404fd25f17.jpg)

**Now our API-Gateway url is ready**

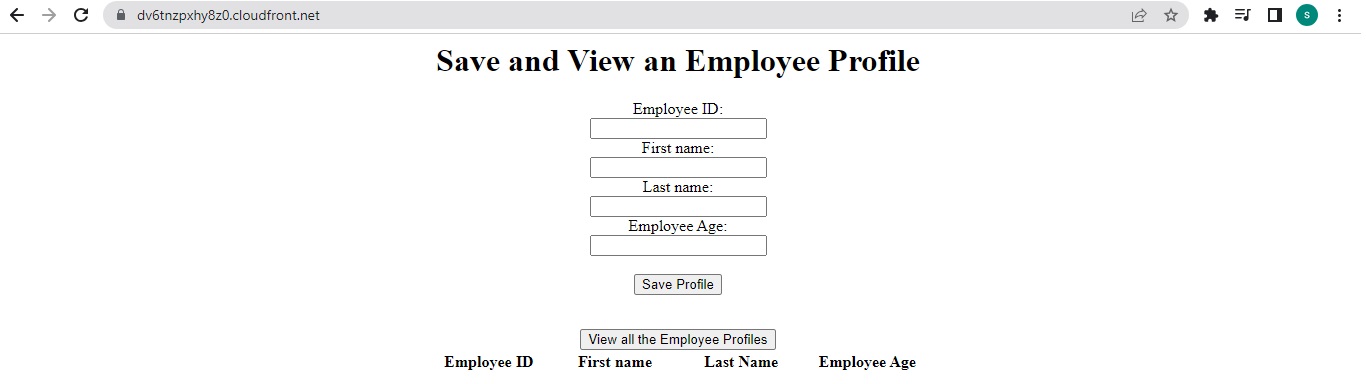
[](https://user-images.githubusercontent.com/119833411/243097980-8f5251d5-6833-49fa-95d3-cd0a449b28d0.jpg)

**We need to copy Invoke URL in Script file and again upload into S3 Bucket**

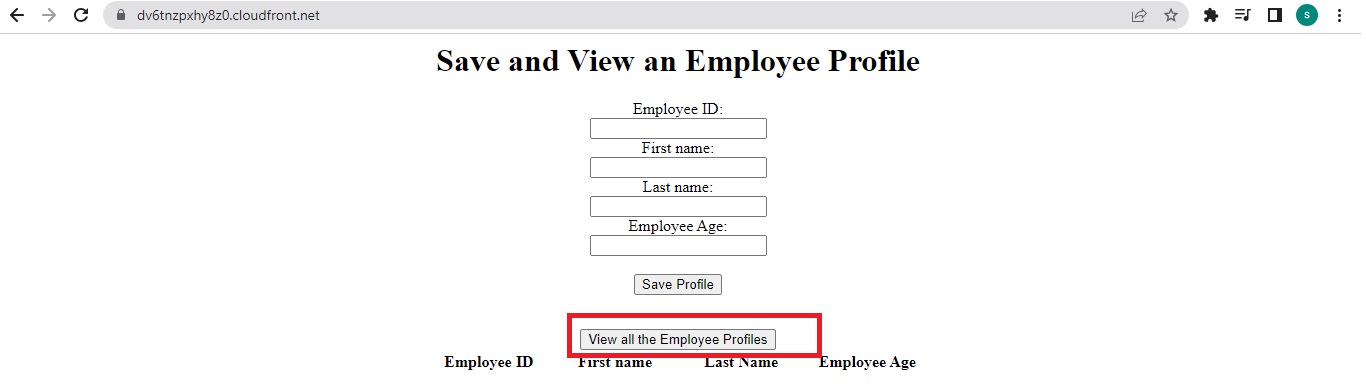
[](https://user-images.githubusercontent.com/119833411/243098669-b88d676b-d682-4826-9f55-d7c8cc2046a7.jpg)

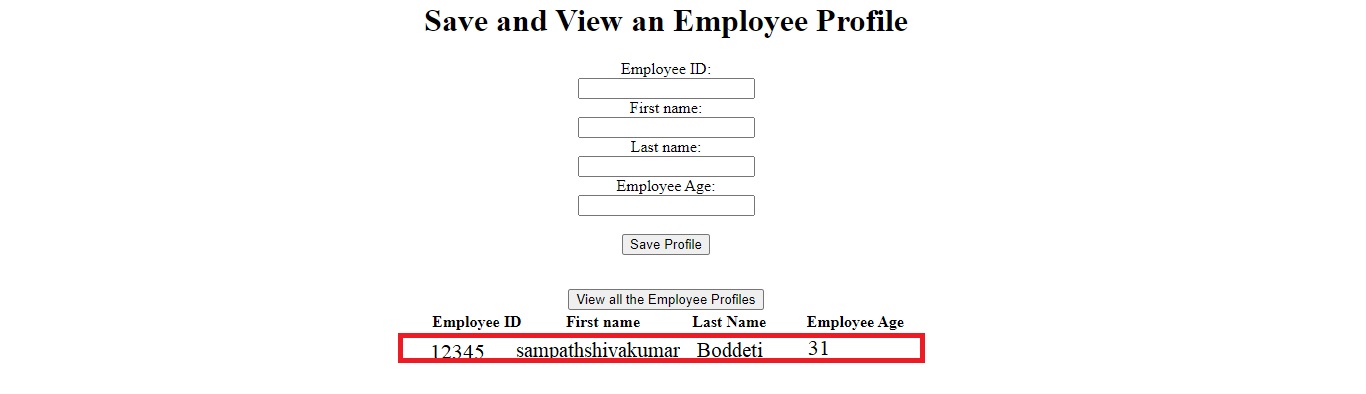
[](https://user-images.githubusercontent.com/119833411/243098875-d5bd45ec-b4a0-43c1-b8eb-85f8574170ca.jpg)

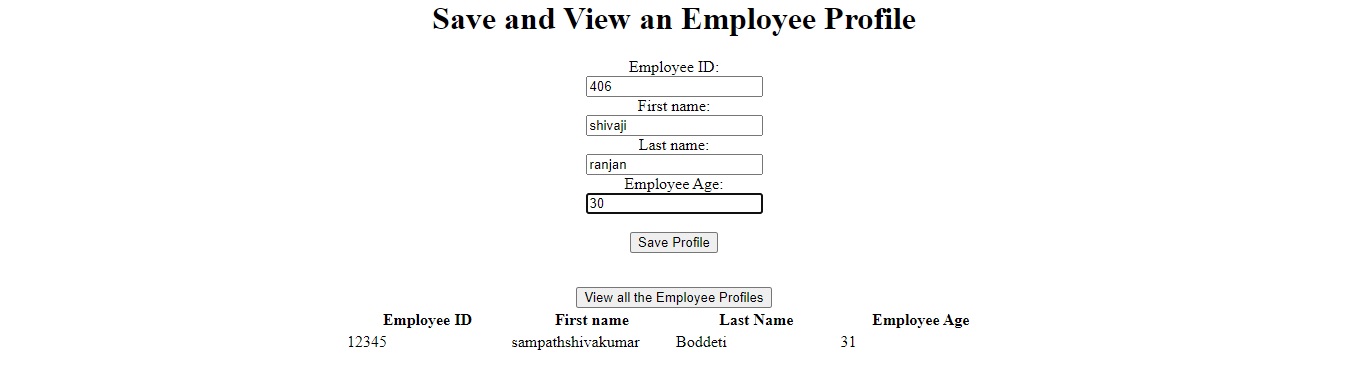
**Everything is setup. Now lets see the output.**

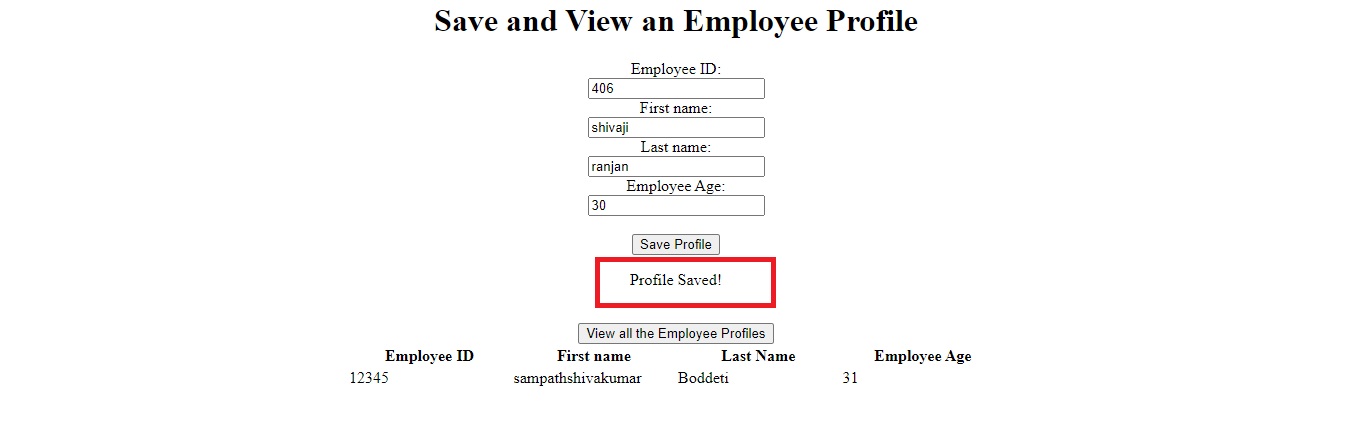
[](https://user-images.githubusercontent.com/119833411/243099194-1a70695e-7841-47d8-8207-6c1f253628cd.jpg)

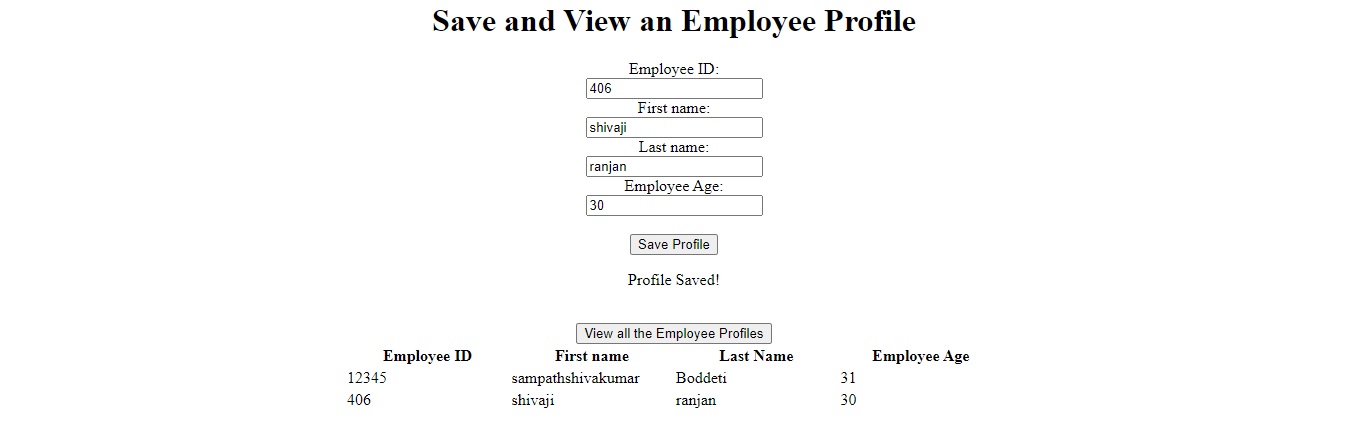
**We know that we have saved one record in DynamoDB during Test of lambda function. So if i click view all employee Profile, it should show the profile**

[](https://user-images.githubusercontent.com/119833411/243099462-f9f03999-5d46-4fc9-ad40-e65f27958b66.jpg)

[](https://user-images.githubusercontent.com/119833411/243101492-7a8c1dd2-57a2-4cb8-90d2-feb4a91feab0.jpg)

[](https://user-images.githubusercontent.com/119833411/243113894-aa729586-5fe1-4561-87f4-e425a24b864b.jpg)

[](https://user-images.githubusercontent.com/119833411/243113921-30ecfc4a-d2f7-4ba7-beaf-6efee87ccbe3.jpg)

[](https://user-images.githubusercontent.com/119833411/243113941-3fc6994f-0aae-4646-bb49-929fe6cf76c5.jpg)

**Now we can add the records.Serverless Web Application is successfully created.**

**Note: Code is taken from**[**https://github.com/nspacer/serverless-website-hosting**](https://github.com/nspacer/serverless-website-hosting)**, i have not written the code myself.**

**Thank you for reading this post! I hope you found it helpful. If you have any feedback or questions,Please connect with me on LinkedIn at**[**https://www.linkedin.com/in/sampathsivakumar-boddeti-1666b810b/**](https://www.linkedin.com/in/sampathsivakumar-boddeti-1666b810b/)**. Your feedback is valuable to me. Thank you!**