

# ***AWS Serverless Architecture***

***Step-by-Step Guide***

Made by: - **Satvik Tripathi**

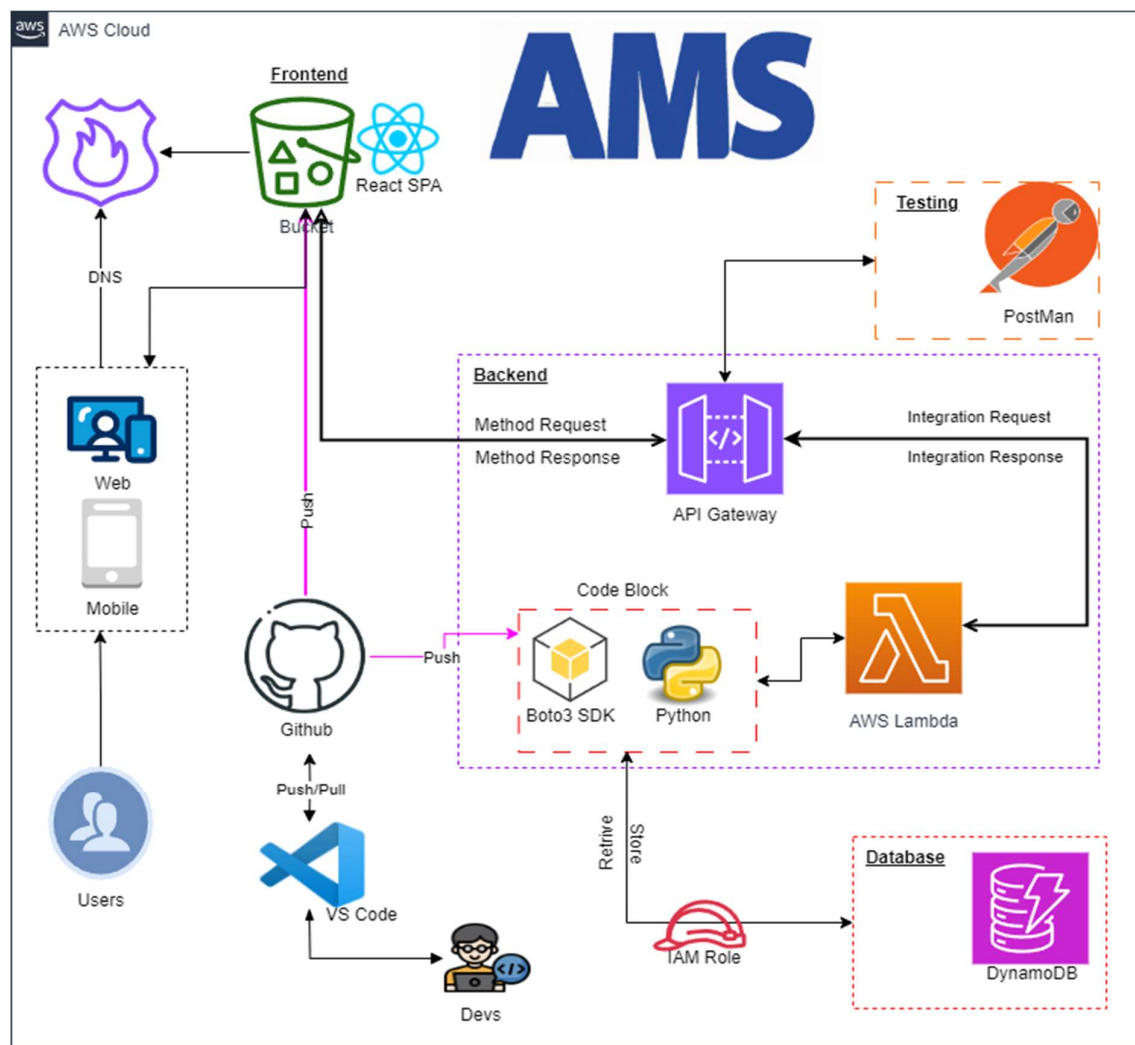
# System Design

## **Project : Attendance Management System**

### **Tech Stack:**

<b>Frontend</b>	:	React, HTML, CSS, JavaScript
<b>Backend</b>	:	Python, Boto3
<b>Databases</b>	:	AWS DynamoDB [NoSQL]
<b>DevOps Tools</b>	:	Git/GitHub
<b>IDE/Code editor</b>	:	VS Code
<b>Testing Tools</b>	:	Postman
<b>Cloud</b>	:	AWS

### **Proposed Architecture**



## **Explanation**

- **Frontend:** The frontend of the AMS app is a React single-page application (SPA) or HTML CSS JS. This means that the entire user interface is loaded in a single web page, which makes the app feel more like a native desktop application.
- **Backend:** The backend of the AMS app is serverless, which means that it is made up of small, independent functions that are run on demand. This makes the app more scalable and cost-effective.
- **API Gateway:** The API Gateway is the entry point for all requests to the AMS app. It routes requests to the appropriate backend function.
- **Integration Requests and Responses:** When a user interacts with the AMS app, their request is first sent to the API Gateway. The API Gateway then routes the request to the appropriate backend function. The backend function processes the request and sends a response back to the API Gateway. The API Gateway then sends the response back to the user.
- **Mobile:** The AMS app can also be accessed from mobile devices. The mobile app uses the same API Gateway as the web app.
- **Push Notifications:** The AMS app can send push notifications to users' devices. This is useful for things like keeping users up-to-date on the latest activity in their apps.
- **Version Control:** The AMS app uses Git for version control. This means that developers can track changes to the code and revert to previous versions if necessary.
- **Databases:** The AMS app uses DynamoDB for its database. DynamoDB is a NoSQL database that is highly scalable and fault-tolerant.
- **Security:** The AMS app uses a number of security features, including IAM roles, API keys, and encryption. These features help to protect the app and its data from unauthorized access.

## Procedure

### Part 1 [Creation of DynamoDb Table]

**Step 1 :** Install GitBash , VSCode , Nodejs on your local system

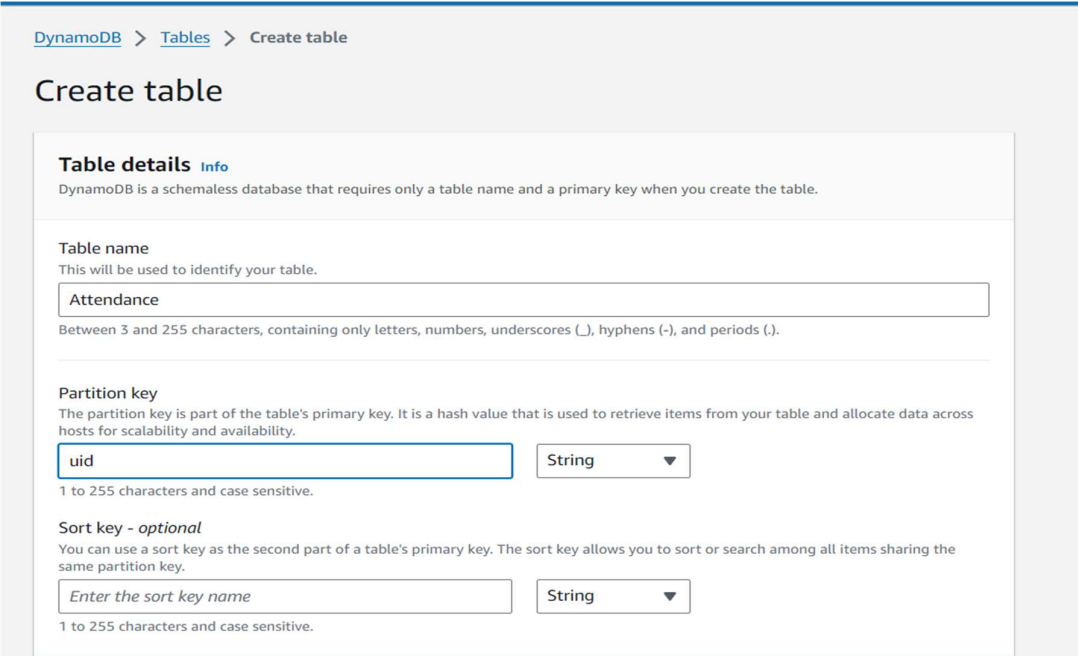
If you haven't installed them yet follow this guide

GitBash : <https://youtu.be/naL0cZnQh1g?si=wPIbKcRTaayIakZK>

VSCode : [https://youtu.be/AdzKzlp66sQ?si=\\_besF5b3SW2CoEau](https://youtu.be/AdzKzlp66sQ?si=_besF5b3SW2CoEau)

NodeJs : <https://youtu.be/JINE4D0Syqw?si=vtjgfgJ99Jh945Hc>

**Step 2 :** Login to AWS Console and go to DynamoDB  
Create a new table called Attendance and set partition key as 'uid'



DynamoDB > Tables > Create table

### Create table

**Table details** [Info](#)  
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

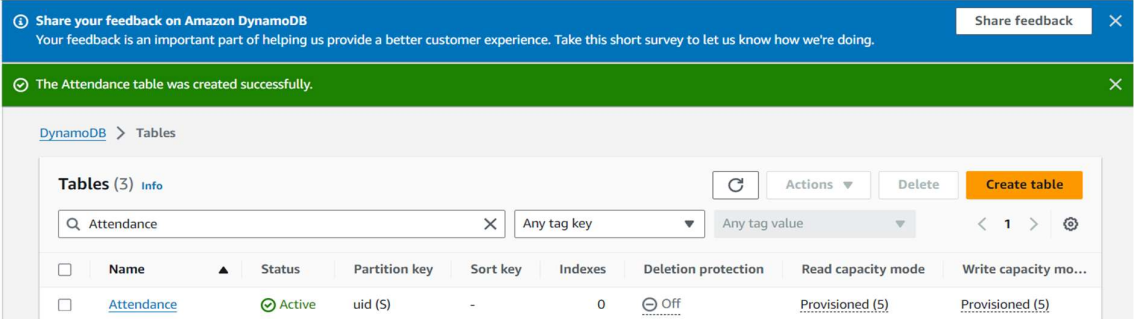
**Table name**  
This will be used to identify your table.  
  
Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.).

**Partition key**  
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.  
   
1 to 255 characters and case sensitive.

**Sort key - optional**  
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.  
   
1 to 255 characters and case sensitive.

And finally create the table

**Step 3 :** Select the table and add some dummy data to it



Share your feedback on Amazon DynamoDB  
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing. [Share feedback](#)

The Attendance table was created successfully.

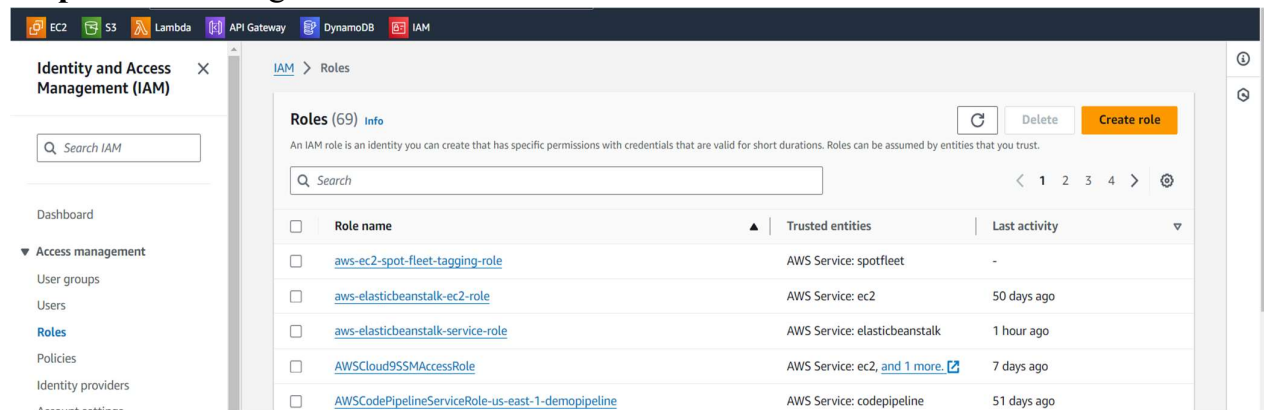
DynamoDB > Tables

**Tables (3)** [Info](#) [Refresh](#) [Actions](#) [Delete](#) [Create table](#)

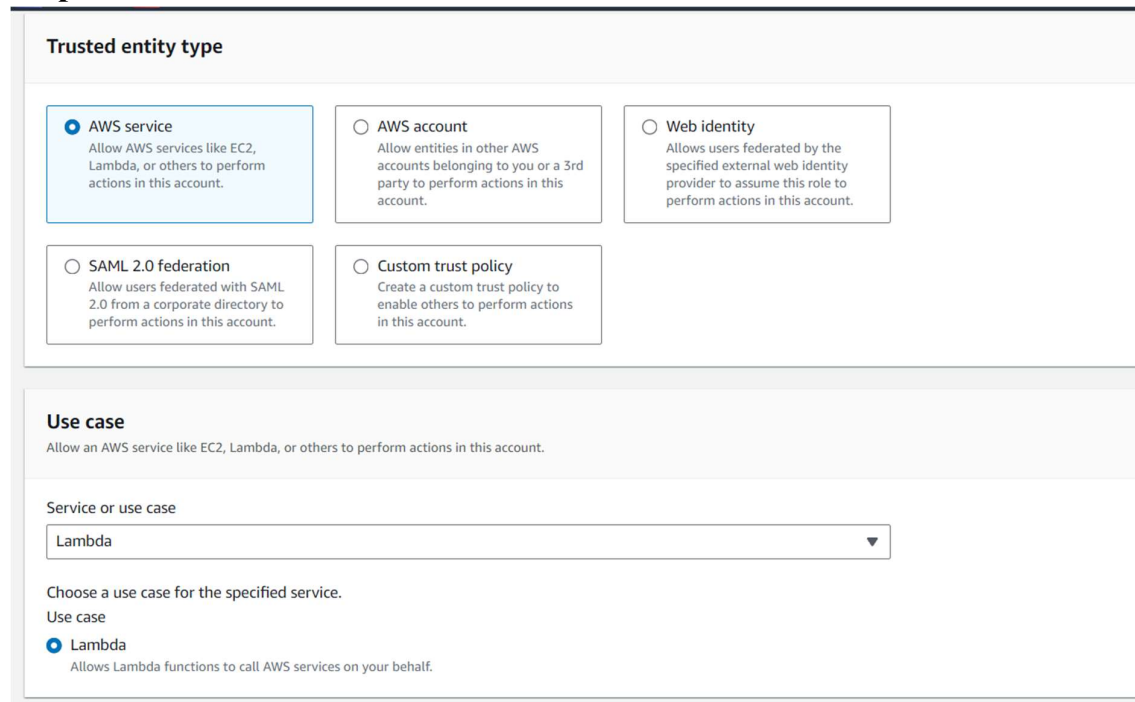
[< 1 >](#) [Filter](#)

<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mo...
<input type="checkbox"/>	<a href="#">Attendance</a>	Active	uid (S)	-	0	Off	Provisioned (5)	Provisioned (5)

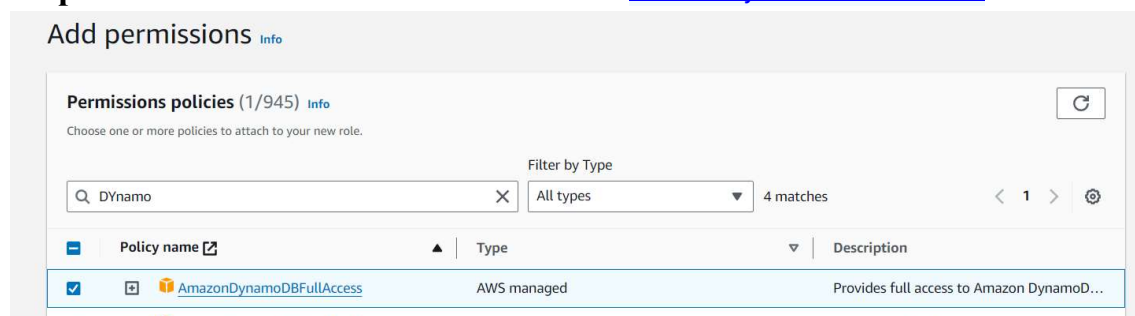
**Step 4 :** Now go to IAM service and then to IAM Role → then create role



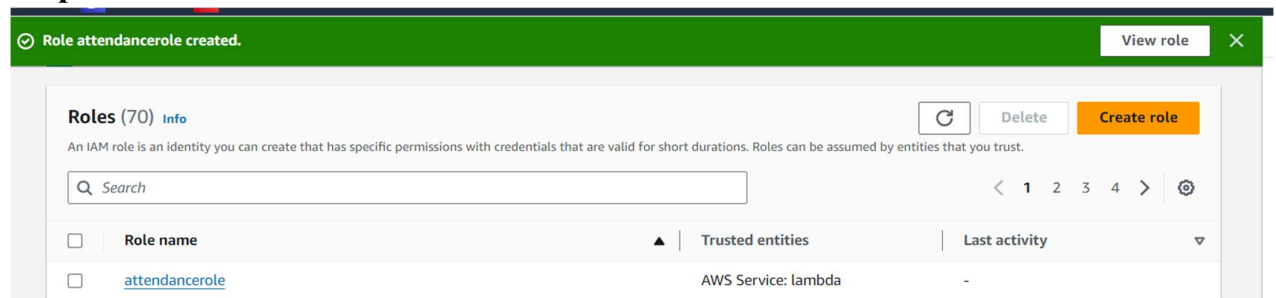
**Step 5 :** Now create a new role called “attendancerole” & usecase as Lambda



**Step 6 :** Under “Add Permission” add “[AmazonDynamoDBFullAccess](#)”

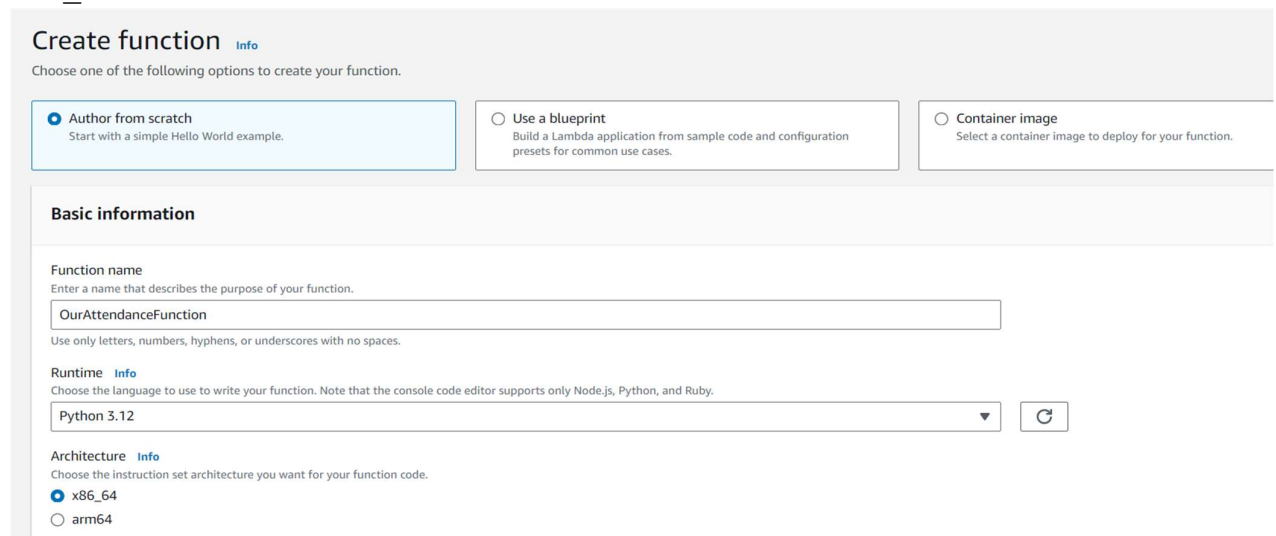


## Step 7 : Give a role name and create role

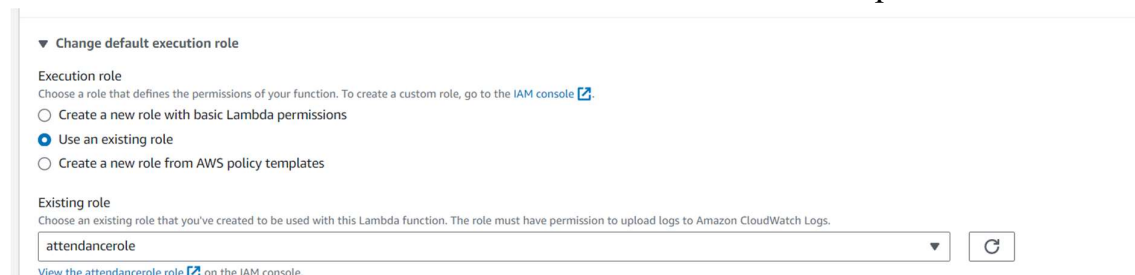


## Part 2 [Creation of Lambda Function ]

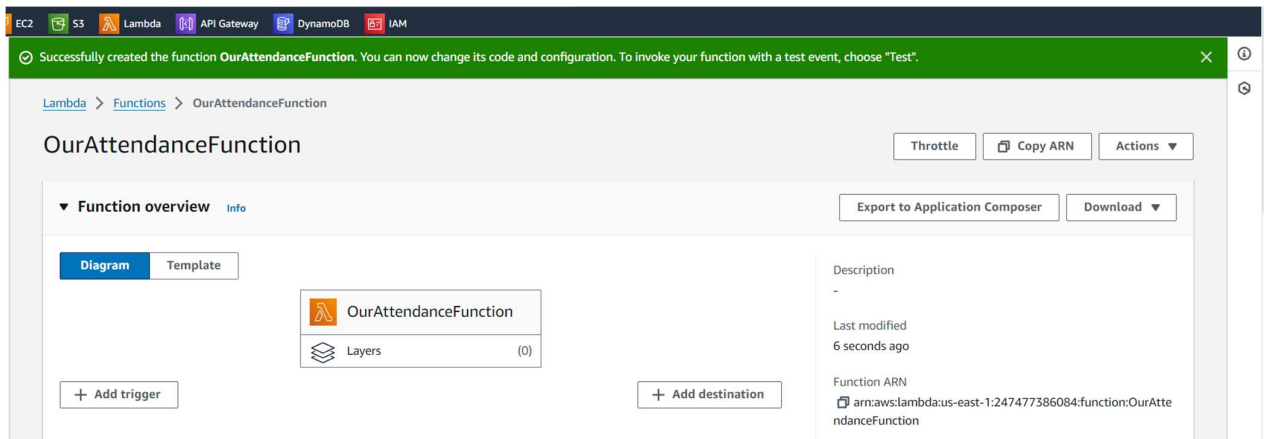
**Step 8 :** Now go to AWS Lambda , and create a new function called “OurAttendanceFunction” in the environment Python 3.12 and architecture as x86\_64



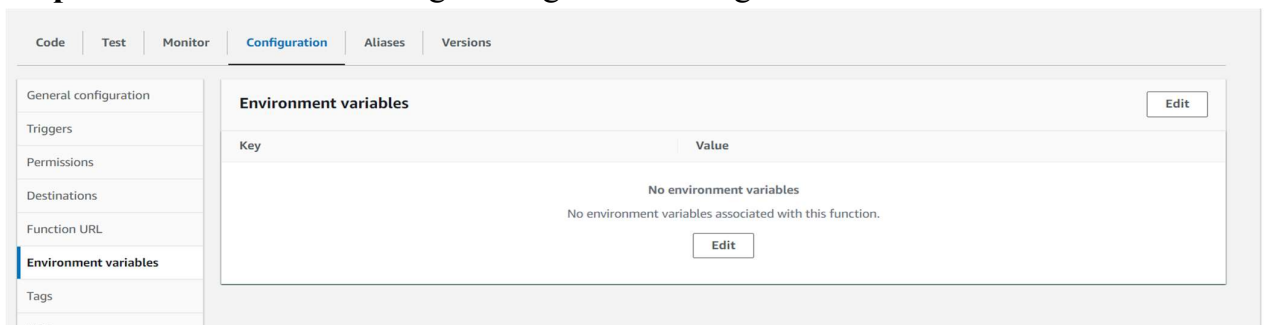
**Step 9 :** Under the default execution role , we select “Use an existing role” And we select the role “attendancerole” that we created in step 7



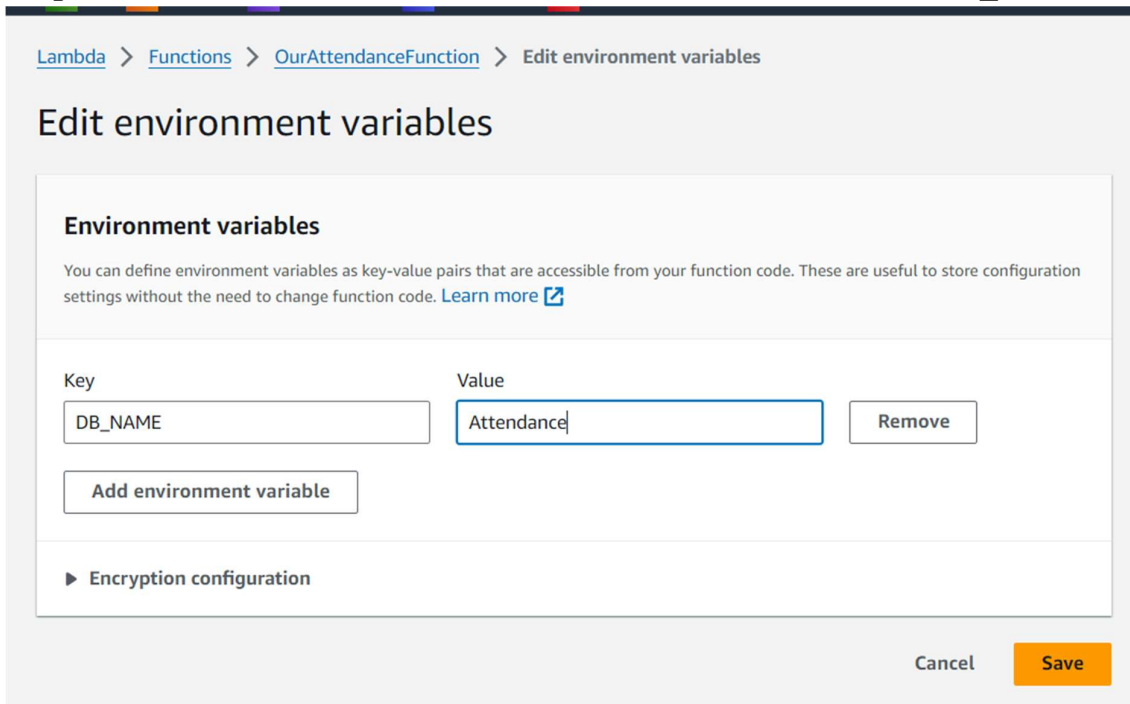
**Step 10:** Finally we click on create the function



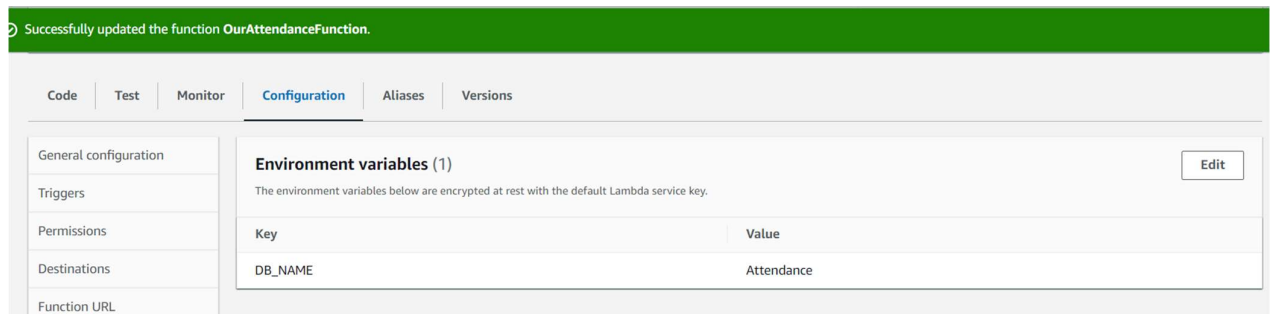
**Step 11:** Under the heading “configuration” we goto environment variables



**Step 12:** Here we click on edit and add a new variable named DB\_NAME

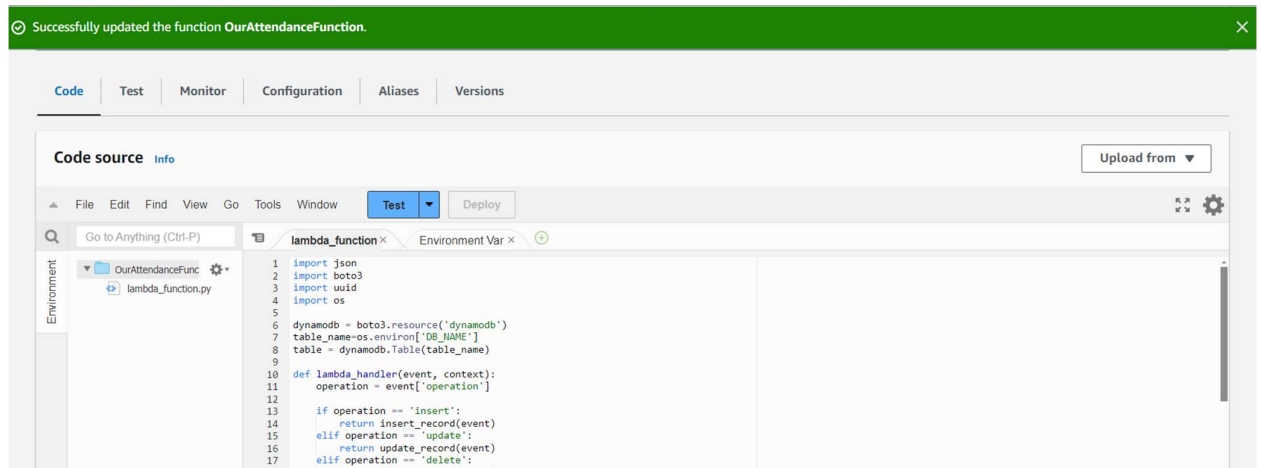


**Step 13 :** Click on save to store the env variable



**Step 14 :** In the section Code → lambda\_function.py  
 We add the code that's provided in the source code present at  
 Server/lambda\_function.py , simply copy and paste the code in the  
 browser editor

After pasting the code click on deploy



**Step 15 :** Now we start by testing the lambda function , so we goto test , we  
 assign a eventname called “inserttest” and add a requestbody as given below ,  
 finally click save & test

```

{
  "operation": "insert",
  "name": "Aarav",
  "email": "aarav@gmail.com",
  "attendance": 10,
  "date": "2024-03-18"
}

```



Test event [Info](#)

Save

Test

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

Create new event

Edit saved event

Event name

testinsert

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

Private

Shareable

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

**Step 16 :** The output should be a statuscode 200 as below

The test event testinsert was successfully saved.

Code

Test

Monitor

Configuration

Aliases

Versions

✓

Executing function: succeeded ([logs](#))

▼ Details

The area below shows the last 4 KB of the execution log.

```
{
  "statusCode": 200,
  "body": {
    "response": {
      "uid": "10e5863e-d4f1-48ed-818a-01d2eb41c116"
    }
  }
}
```

Summary

Code SHA-256  
buswXTzBxy+KEzYSEZ0J8u9gZslhCmuCyTFw62c6IzE=

Execution time  
2 seconds ago (February 15, 2024 at 02:50 PM GMT+5:30)

Note :- If there is an error go through steps 1 -16 again and check

Also tally at the dynamoDb Table

Select your table and click explore table items

**Attendance**

Partition key: uid (String) | Sort key: - | Capacity mode: Provisioned | Table status: Active

**Scan or query items**

Scan | Query

Select a table or index: Table - Attendance | Select attribute projection: All attributes

Filters

Run | Reset

Completed. Read capacity units consumed: 0.5

**Items returned (1)**

uid (String)	attendance	date	email	name
10e5863e-d4f1-48ed...	3	2024-02-14	mansikhan...	mansi Khandurie

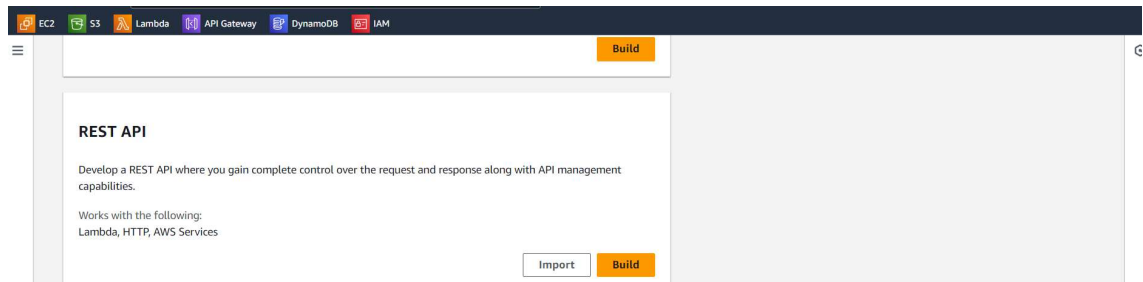
As we can see a record has being inserted so our table , i.ee lambda function works as expected.

### Part 3 [Creation of API GATEWAY]

**Step 17 :** After creation of Lambda , we goto API Gateway → Create → Rest API Creation

**APIs (6/6)**

Name	Description	ID	Protocol	API endpoint type	Created
attendanceAPI		h7wdgah8x6	REST	Regional	2024-02-14
customfun-API	Created by AWS Lambda	a5dcdgktvh	REST	Regional	2024-02-09
fasalfusion		czf5169uj1	REST	Regional	2023-11-08
fun-API	Created by AWS Lambda	ulq2pxot12	REST	Regional	2024-02-13



**Step 18 :** We assign a name to the gateway

**API details**

☒ **New API**  
Create a new REST API.

☐ **Clone existing API**  
Create a copy of an API in this AWS account.

☐ **Import API**  
Import an API from an OpenAPI definition.

☐ **Example API**  
Learn about API Gateway with an example API.

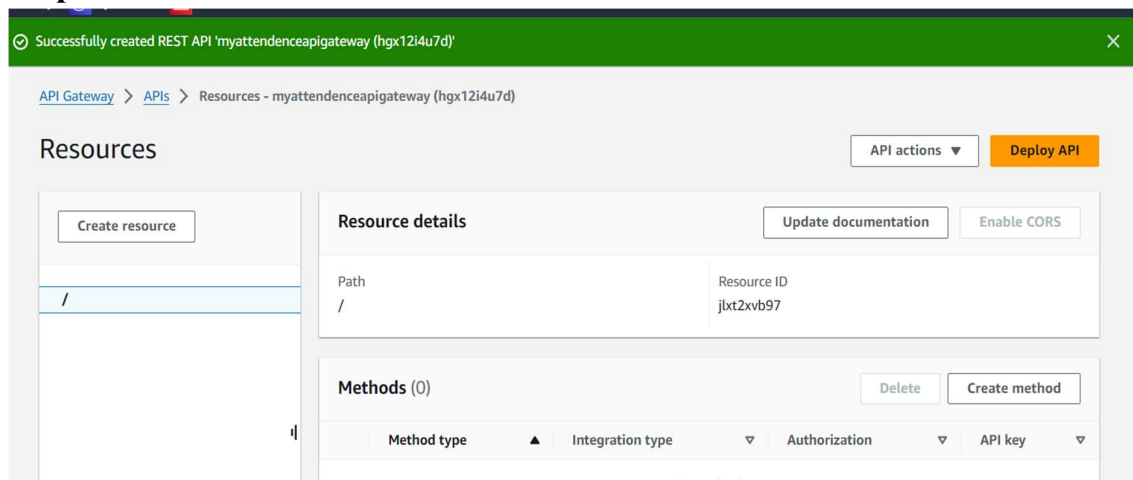
API name

Description - *optional*

API endpoint type  
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

[Cancel](#) [Create API](#)

**Step 19 :** Under the '/' Path we create a new method



**Step 20:** We select the method POST & integration as lambda


### Create method


#### Method details


Method type


POST


Integration type

☒ **Lambda function**  
Integrate your API with a Lambda function.


☐ **HTTP**  
Integrate with an existing HTTP endpoint.


☐ **Mock**  
Generate a response based on API Gateway mappings and transformations.


☐ **AWS service**  
Integrate with an AWS Service.


☐ **VPC link**  
Integrate with a resource that isn't accessible over the public internet.


**Step 21 :** We also select lambda function we created earlier & click create

☒ **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-1:247477386084:function:Our/

*Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.*

☒ **Default timeout**  
The default timeout is 29 seconds.

Cancel
Create method

**Step 22 :** We would also enable CROS by selecting '/' Path & select POST

API Gateway > APIs > Resources - myattendanceapigateway (hgx12i4u7d)

Resources

API actions
Deploy API

Create resource

/
POST

Resource details

Update documentation
Enable CORS

Path  
/
Resource ID  
jlxt2xvb97

Methods (1)

Delete
Create method

## Enable CORS

### CORS settings [Info](#)

To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for your API.

#### Gateway responses

API Gateway will configure CORS for the selected gateway responses.

☒ Default 4XX

☒ Default 5XX

#### Access-Control-Allow-Methods

☒ OPTIONS

☒ POST

#### Access-Control-Allow-Headers

API Gateway will configure CORS for the selected gateway responses.

Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token

#### Access-Control-Allow-Origin

Enter an origin that can access the resource. Use a wildcard '\*' to allow any origin to access the resource.

\*

► Additional settings

API Gateway | DynamoDB | IAM

☑ Successfully enabled CORS ✕

► Details

[API Gateway](#) > [APIs](#) > Resources - myattendanceapigateway (hgx12i4u7d)

Resources API actions ▼ Deploy API

Create resource

/

OPTIONS

POST

#### Resource details

Update documentation Enable CORS

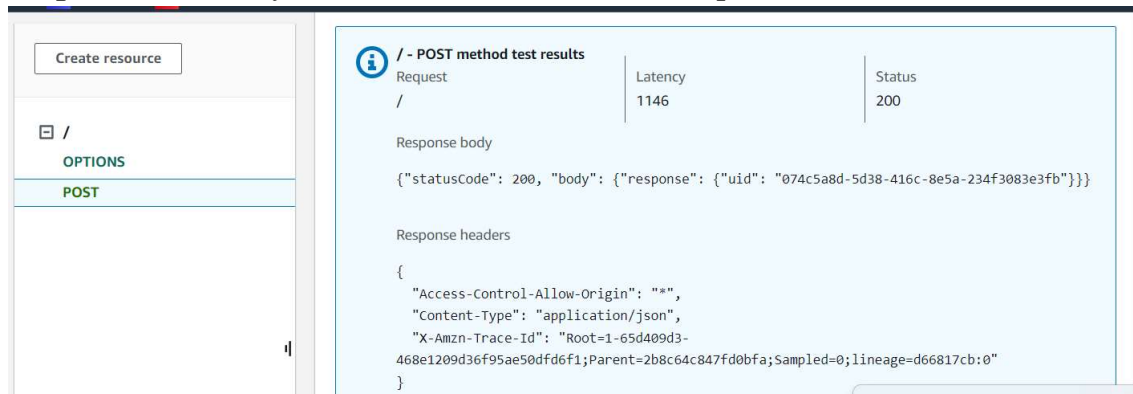
Path	Resource ID
/	jlt2xvb97

Methods (2) Delete Create method

**Step 23 :** Now to test the API Select POST Method under '/' Path and goto Test and use the following testcase

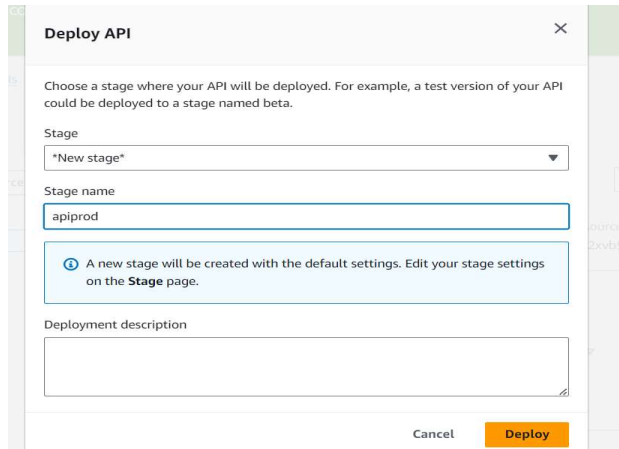
```
{
  "operation": "insert",
  "name": "Aarav",
  "email": "aarav@gmail.com",
  "attendance": 10,
  "date": "2024-03-18"
}
```

**Step 24:** Finally we click Test , and see the response



Note : → If you face error at these steps it is probably because of the configuration of post method under the resource ‘/’ go to steps 19 to 25 and retry the Part

**Step 25:** Finally Deploy API in a new Stage [ Where we find invoke Url]

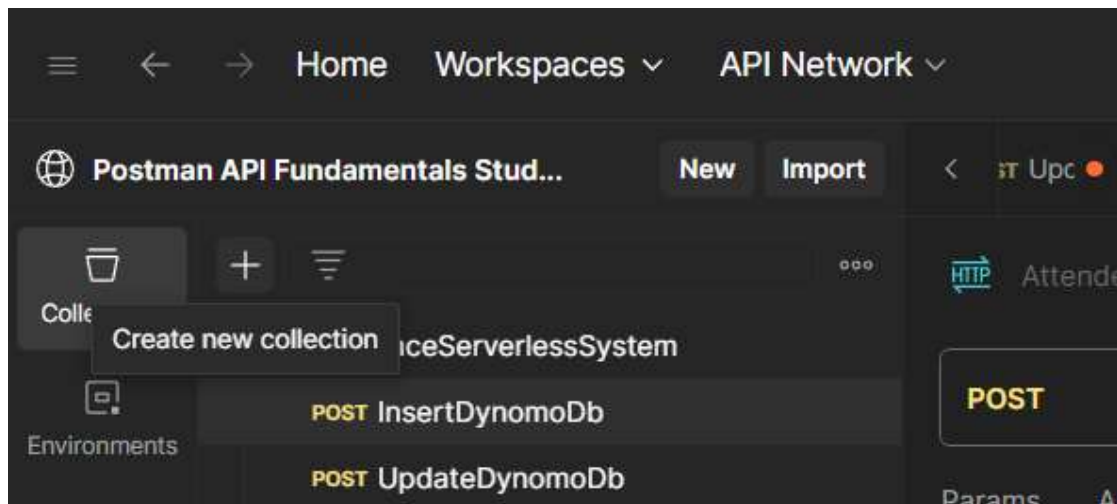


## **Part 4 [Testing with POSTMAN]**

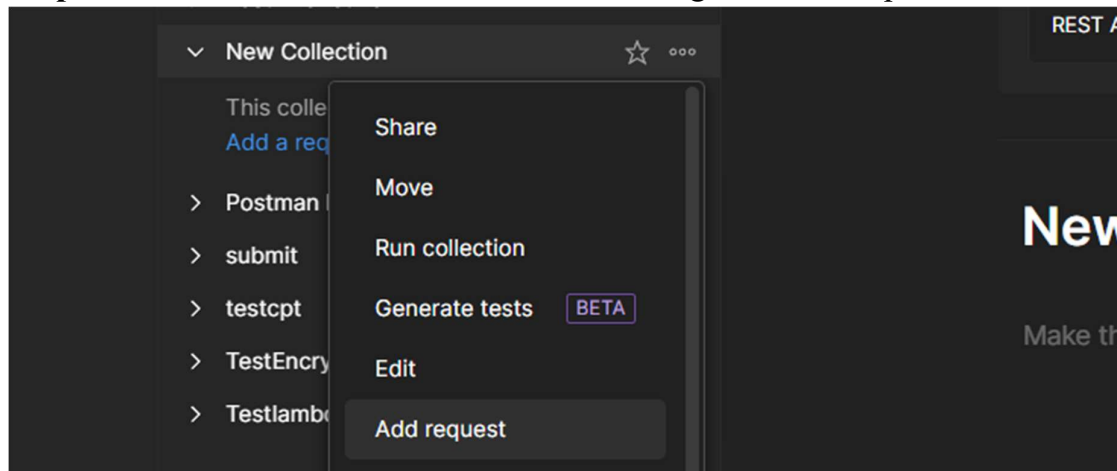
Note : Now this part is optional because you have already tested our API in the previous part however it is one of the best practises to test our API with Postman before deployment so if you wish to skip this part feel free to jump to part 5

**Perquisites:** - Install Postman from <https://www.postman.com/downloads/>

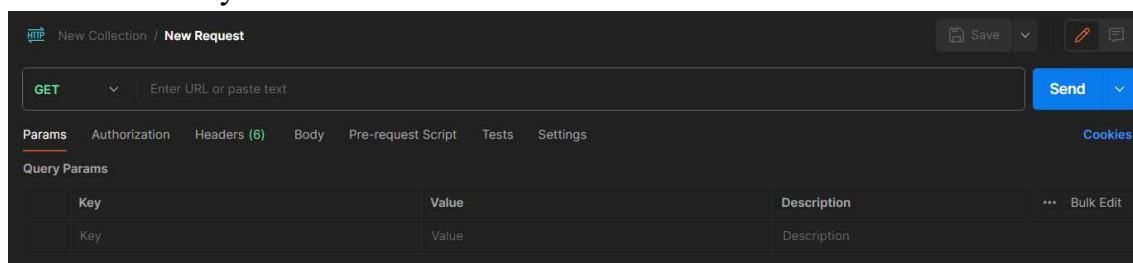
**Step 26:** Open PostMan → Create Collection



**Step 27:** Create a new Blank Collection and goto New Request



**Step 28:** In the new Request , select the POST option paste the invoke URL of API Gateway

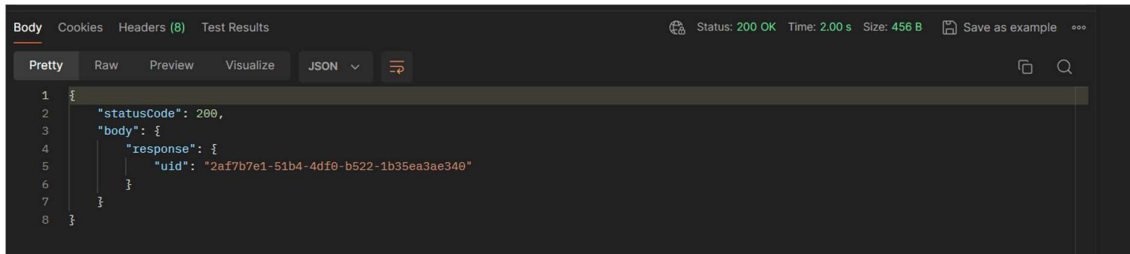


**Step 29:** Goto Body and pate the following body, under RAW, using JSON

```
{
  "operation": "insert",
  "name": "manav Khandurie",
  "email": "manavkhandurie@gmail.com",
  "attendance": 3,
  "date": "2024-02-14"
```

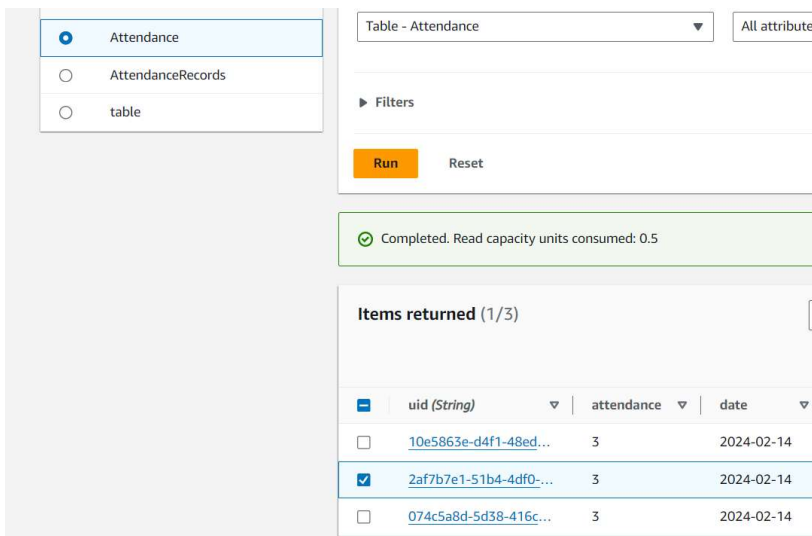
```
}
```

**Step 30:** Click on Send , and we should get a 200 Status Response



Also we will check at the db side ,

We can see that item under 2af7b7e1-51b4-4df0-b522-1b35ea3ae340 uid is added



**Step 31:** Subsequently we also check fetch & delete [ Use your UID in del ]

```
{
  "operation" : "fetch"
}
```



Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "operation": "fetch"
3 }
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 756 ms Size: 865 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "statusCode": 200,
3   "body": {
4     "records": [
5       {
6         "date": "2024-02-14",
7         "email": "mansikhandurie@gmail.com",
8         "name": "mansikhandurie",
9         "uid": "10e5863e-d4f1-48ed-818a-01d2eb41c116",
10        "attendance": 3
11      },
12      {
13        "date": "2024-02-14",
14        "email": "manavkhandurie@gmail.com",
15        "name": "manav Khandurie",
16        "uid": "2af7b7e1-51b4-4df0-b522-1b35ea3ae340",
17        "attendance": 3
18      }
19    ]
20  }
21 }
```

```
{
  "operation": "delete",
  "uid": "e29815aa-6688-41b6-8ce6-4ea614c22e10"
}
```

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "operation": "delete",
3   "uid": "2af7b7e1-51b4-4df0-b522-1b35ea3ae340"
4 }
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 397 ms Size: 456 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "statusCode": 200,
3   "body": {
4     "response": {
5       "uid": "2af7b7e1-51b4-4df0-b522-1b35ea3ae340"
6     }
7   }
8 }
```

Attendance

AttendanceRecords

table

Table - Attendance

All attributes

Filters

Run Reset

Completed. Read capacity units consumed: 0.5

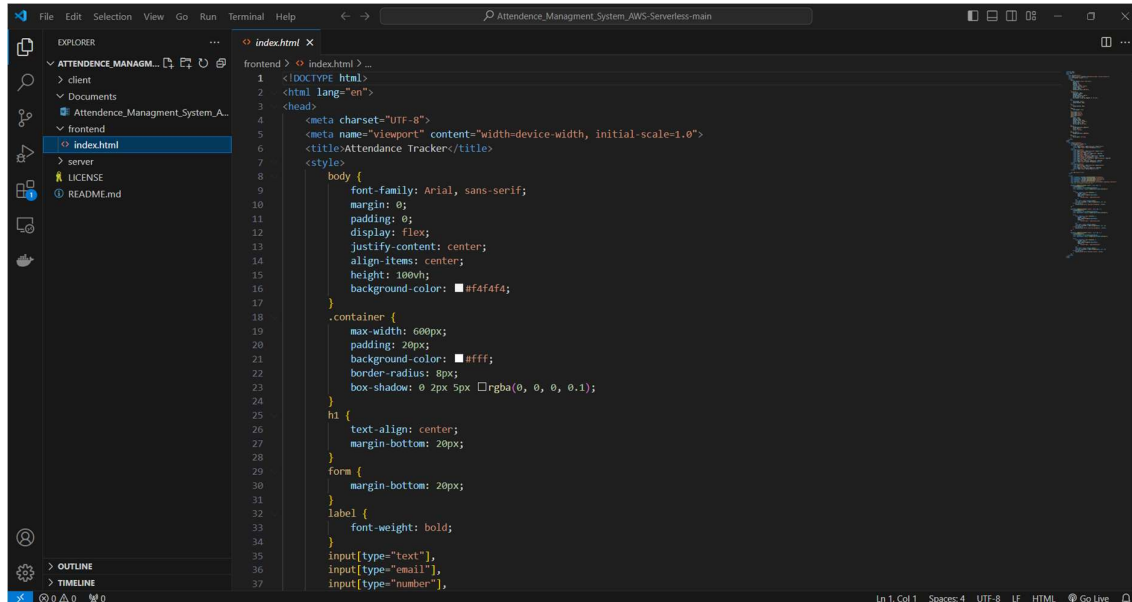
Items returned (2)

	uid (String)	attendance	date
<input type="checkbox"/>	<a href="#">10e5863e-d4f1-48ed...</a>	3	2024-02-14
<input type="checkbox"/>	<a href="#">074c5a8d-5d38-416c...</a>	3	2024-02-14

[ ID with 2af7b7e1-51b4-4df0-b522-1b35ea3ae340 deleted ]

## Part 5 [Frontend Deployment]

**Step 32:** We fetch the frontend from Source Code → Frontend & view using VS Code

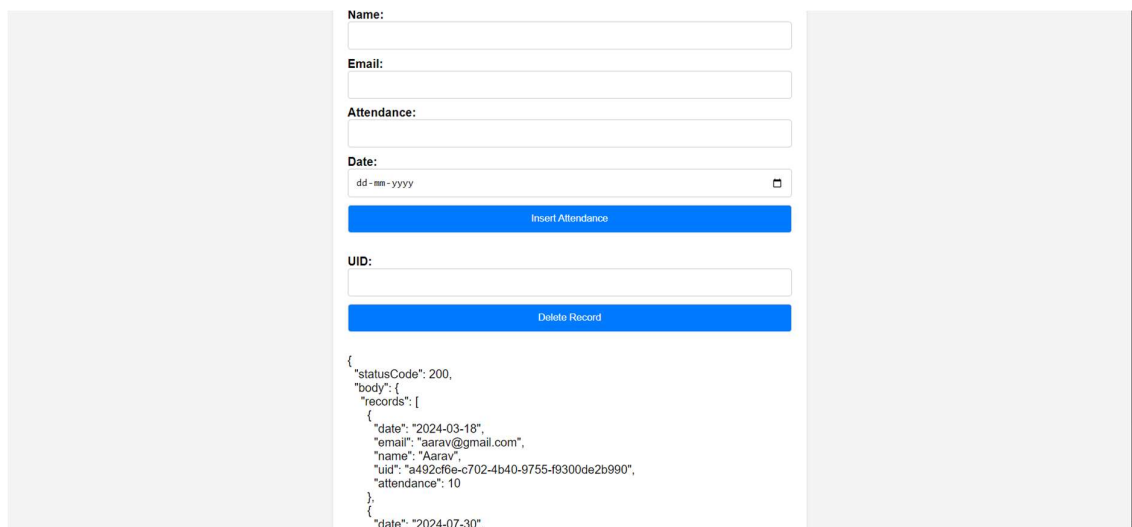


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Attendance Tracker</title>
7   <style>
8     body {
9       font-family: Arial, sans-serif;
10      margin: 0;
11      padding: 0;
12      display: flex;
13      justify-content: center;
14      align-items: center;
15      height: 100vh;
16      background-color: #f4f4f4;
17    }
18    .container {
19      max-width: 600px;
20      padding: 20px;
21      background-color: #fff;
22      border-radius: 8px;
23      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
24    }
25    h1 {
26      text-align: center;
27      margin-bottom: 20px;
28    }
29    form {
30      margin-bottom: 20px;
31    }
32    label {
33      font-weight: bold;
34    }
35    input[type="text"],
36    input[type="email"],
37    input[type="number"],
```

**Step 33:** In the source code change the URL variable to your API

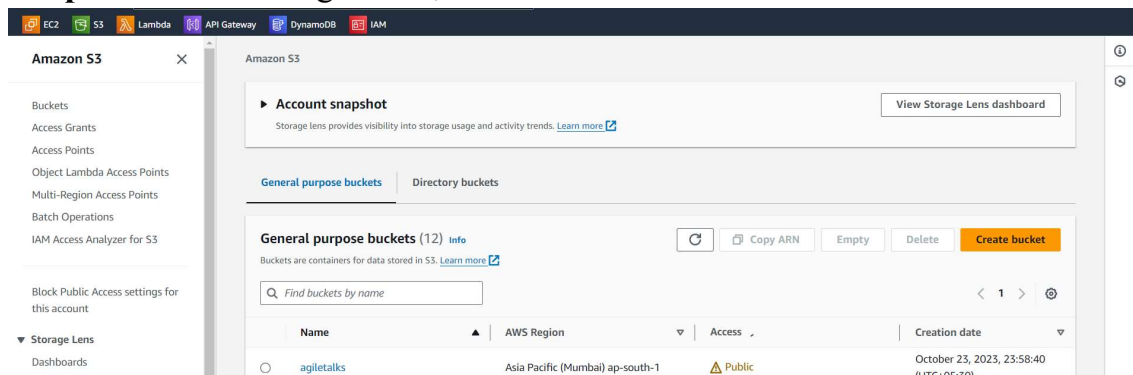
**Step 34:** Test the site locally , use live server or simply open the html page using chrome

Here we check all Functionalities insert delete and fetch

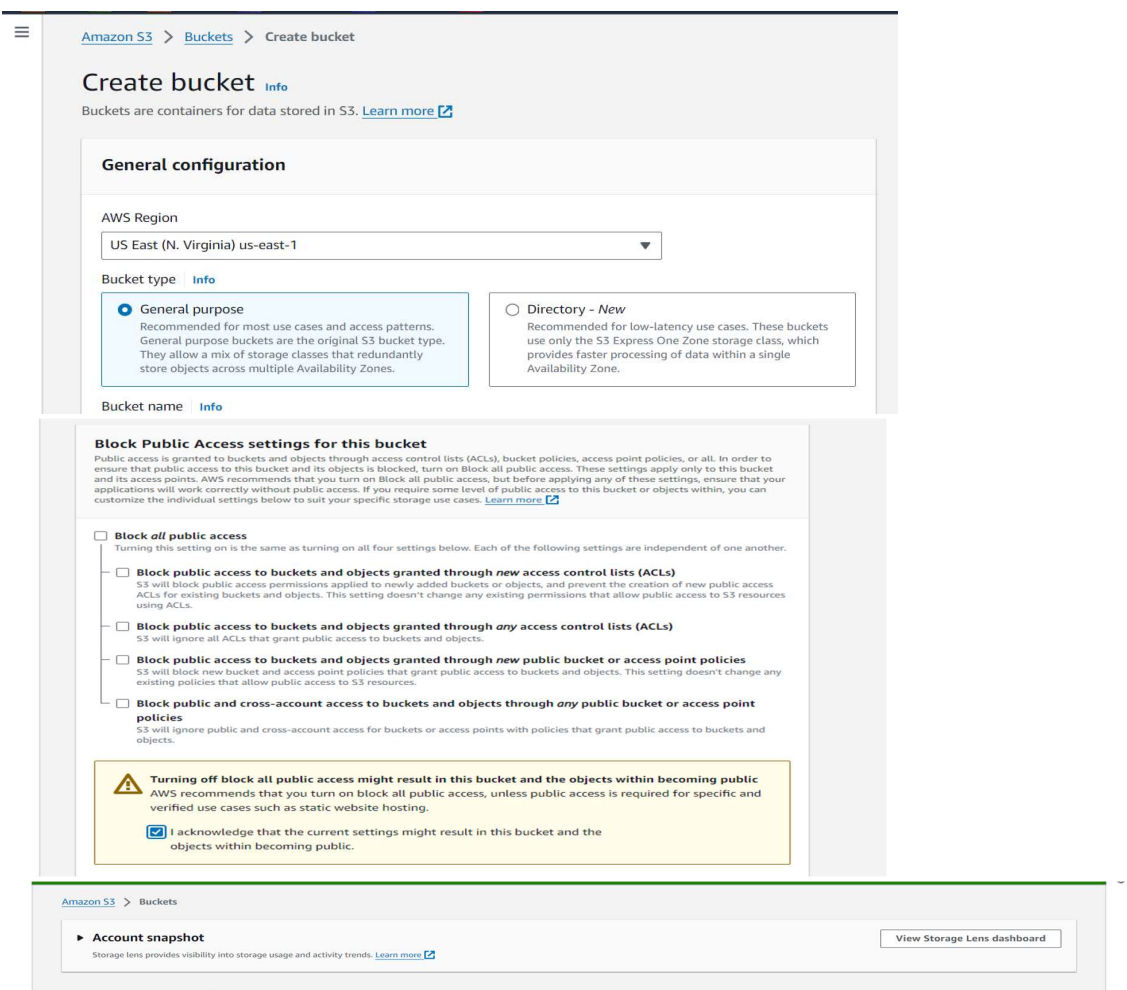


```
{
  "statusCode": 200,
  "body": {
    "records": [
      {
        "date": "2024-03-18",
        "email": "aarav@gmail.com",
        "name": "Aarav",
        "uid": "a492c8e-c702-4b40-9755-49300de2b990",
        "attendance": 10
      },
      {
        "date": "2024-07-30",
```

**Step 35:** Now we goto S3 , and create a bucket



**Step 36:** We give a unique name to the bucket , unblock the public access & create



**Step 37:** We goto the bucket we created and add the HTML page we created before , goto → Upload , Upload file , select the File and Upload

**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

## Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

**Files and folders (1 Total, 5.4 KB)** [Remove](#) [Add files](#) [Add folder](#)

All files and folders in this table will be uploaded.

Find by name

<input type="checkbox"/>	Name	Folder	Type
<input type="checkbox"/>	index.html	-	text/html

**Step 38:** Goto Permission → Bucket Policy → Edit and use the following Policy & save. Note do change the Resource to your Bucket ARN shown

“<Your-resource-arn>/\*”

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::agiletalks/*"
    }
  ]
}
```

**Bucket policy** [Edit](#) [Delete](#)

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

No policy to display.

[Copy](#)

**Step 39:** Now goto Staic Site hosting under Properties and enable it , also set the index document as index.html [ P.S. This is your html file's name that you uploaded] , we ll get a bucket endpoint url

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

☐ Disable

☒ Enable

Hosting type

☒ Host a static website

Use the bucket endpoint as the web address. [Learn more](#)

☐ Redirect requests for an object

Redirect requests to another bucket or domain. [Learn more](#)

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](#)

Index document

Specify the home or default page of the website.

index.html

Error document - optional

**Step 40:** Open the URL in the incognito mode , well see the application test it out as you wish

csmproj123.s3.ap-northeast-1.amazonaws.com/index.html

Name:

Email:

Attendance:

Date:

dd-mm-yyyy

Insert Attendance

UID:

Delete Record

```
{
  "statusCode": 200,
  "body": {
    "records": [
      {
        "date": "2024-03-18",
        "email": "aarav@gmail.com",
        "name": "Aarav",
        "uid": "a492cf6e-c702-4b40-9755-f9300de2b990",
        "attendance": 10
      },
      {
        "date": "2024-07-30",
```