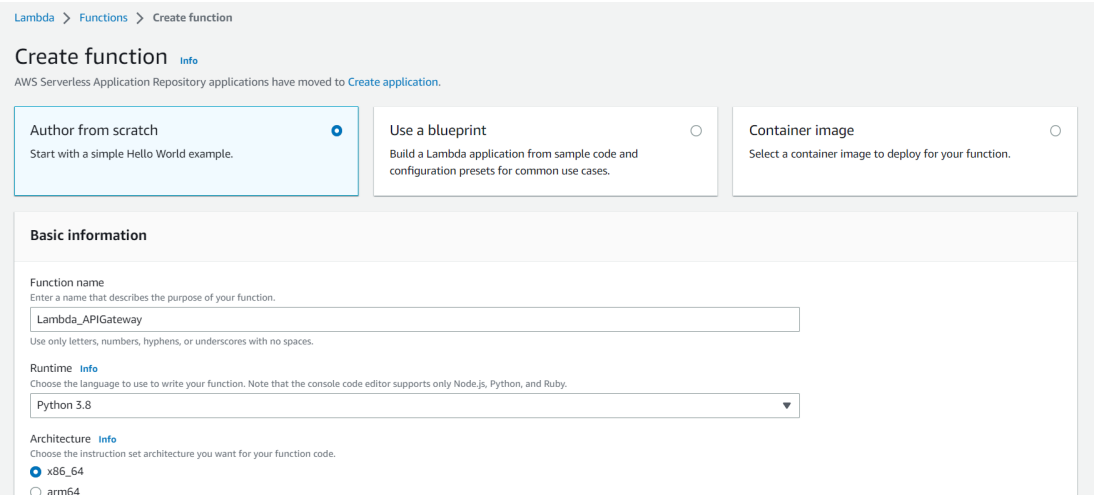
API Gateway

Problem Statement:

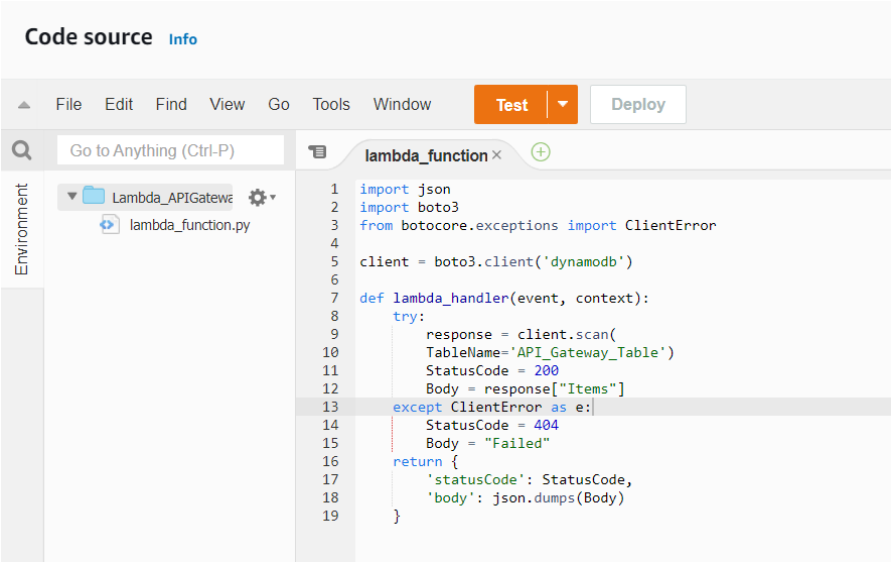
1. Create and configure a Lambda function in Python or Node.js to perform operations on a DynamoDB table.
2. Create a REST API in API Gateway to connect to your Lambda function.
3. Create a DynamoDB table and test it with your Lambda function in the console.
4. Deploy your API and test the full setup using curl in a terminal.

Solution:

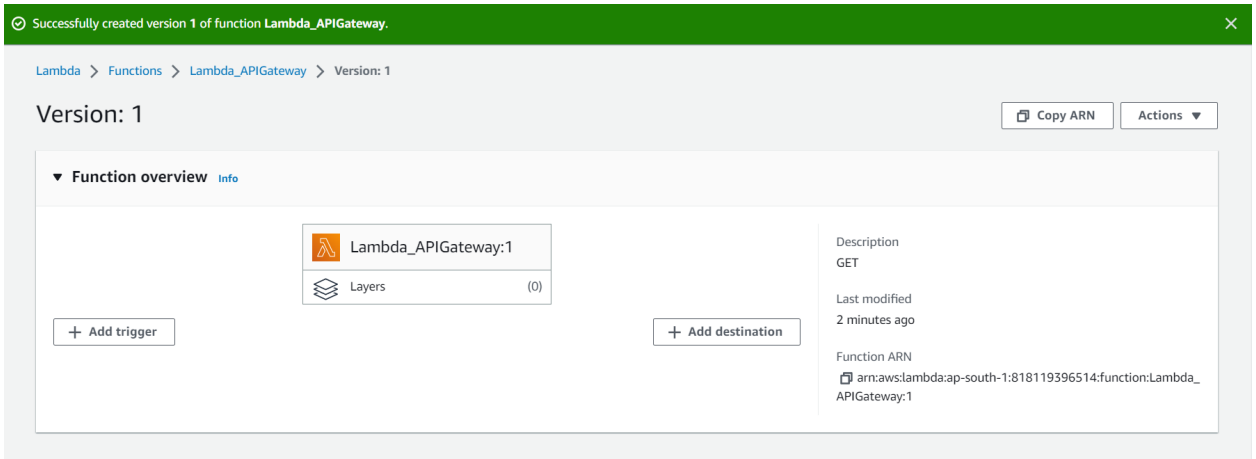
Creating a Lambda function with the Python 3.8 runtime



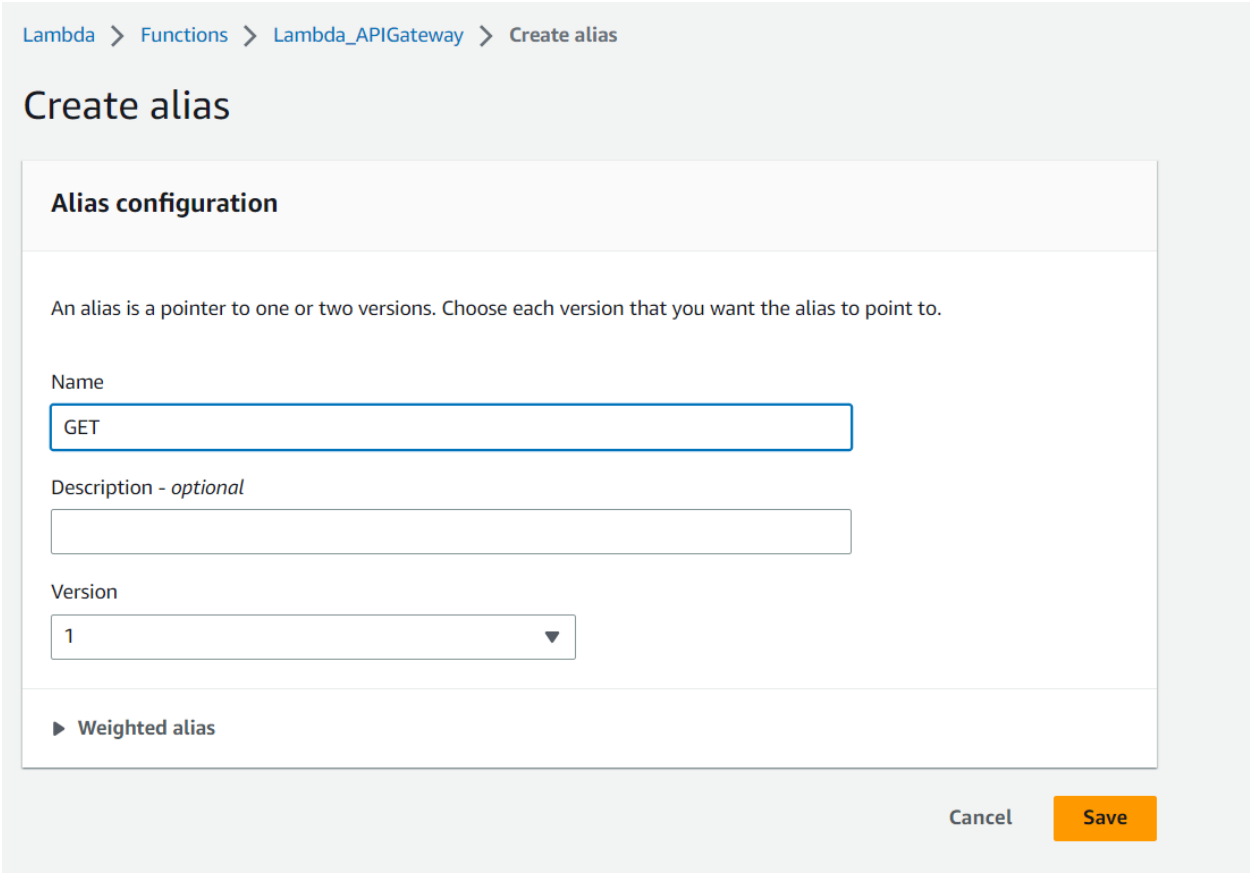
Updated the code in the lambda function to get items from the DynamoDB table whenever the function is triggered.



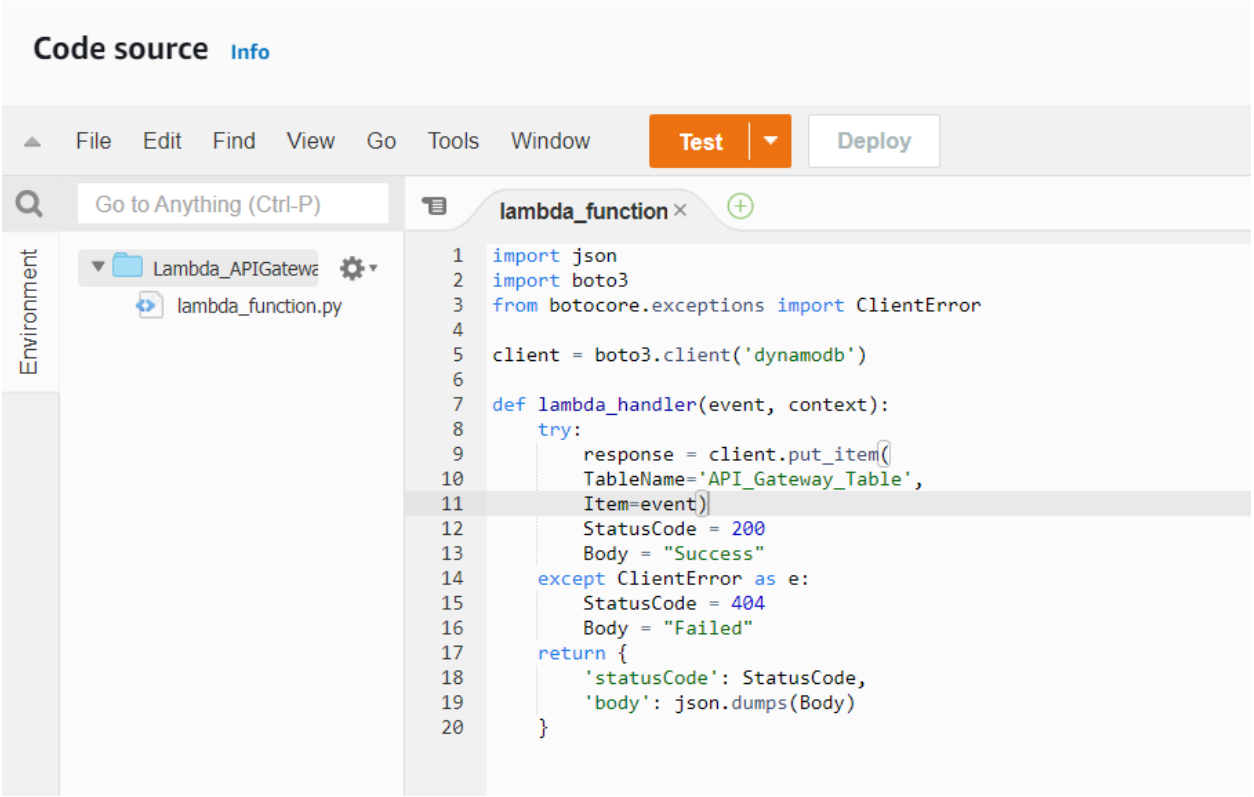
Created version for the lambda function code which implements the **GET method operation**.



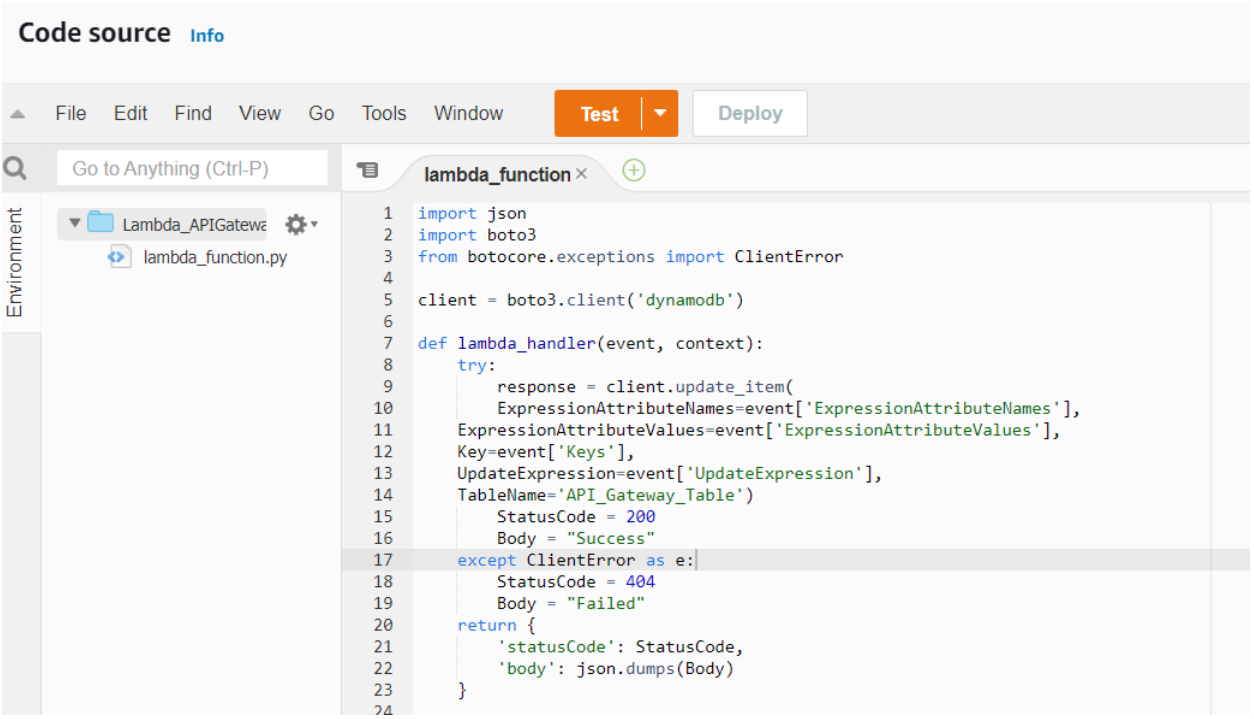
Creating an alias to the previously created version.



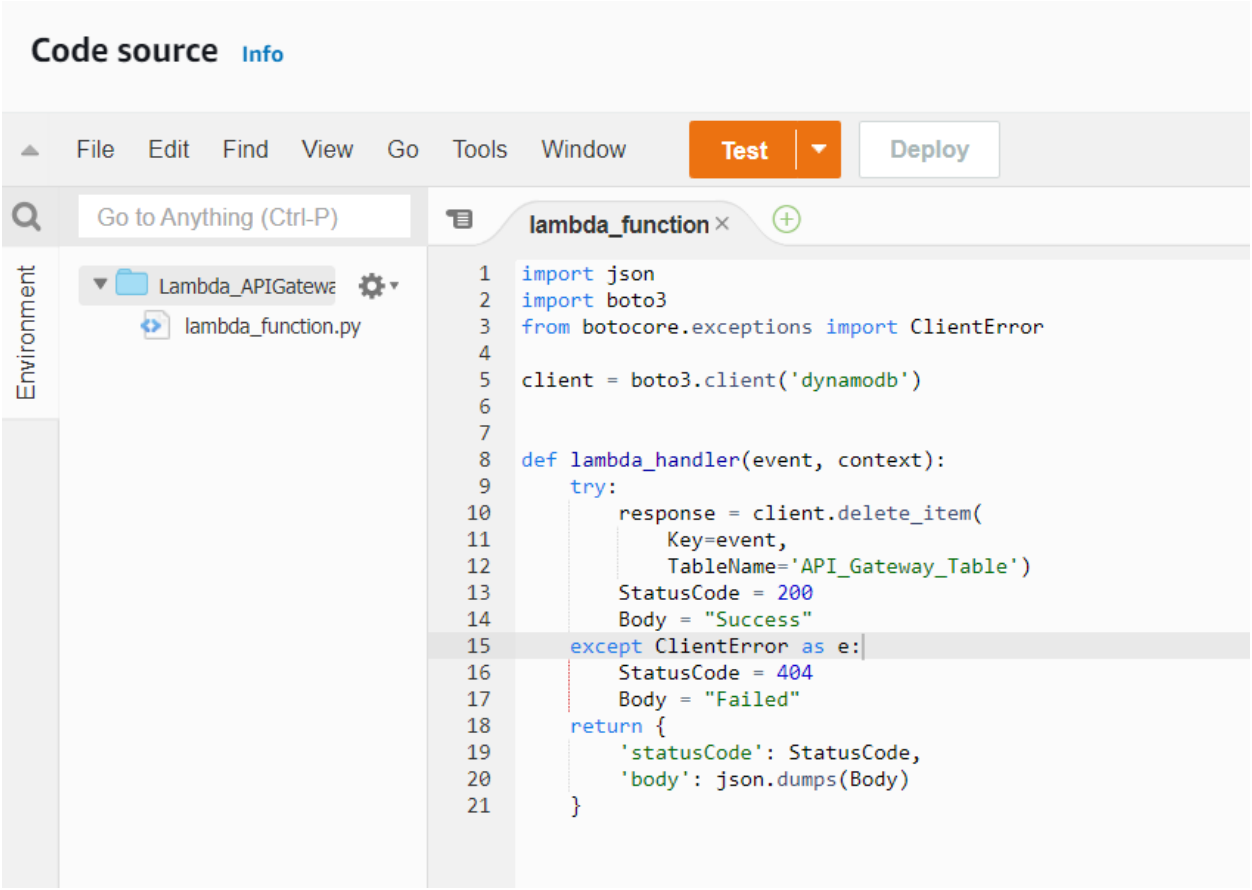
Updated the lambda function code to perform the **post method operation**. After that, I created the version and alias to the existing code.

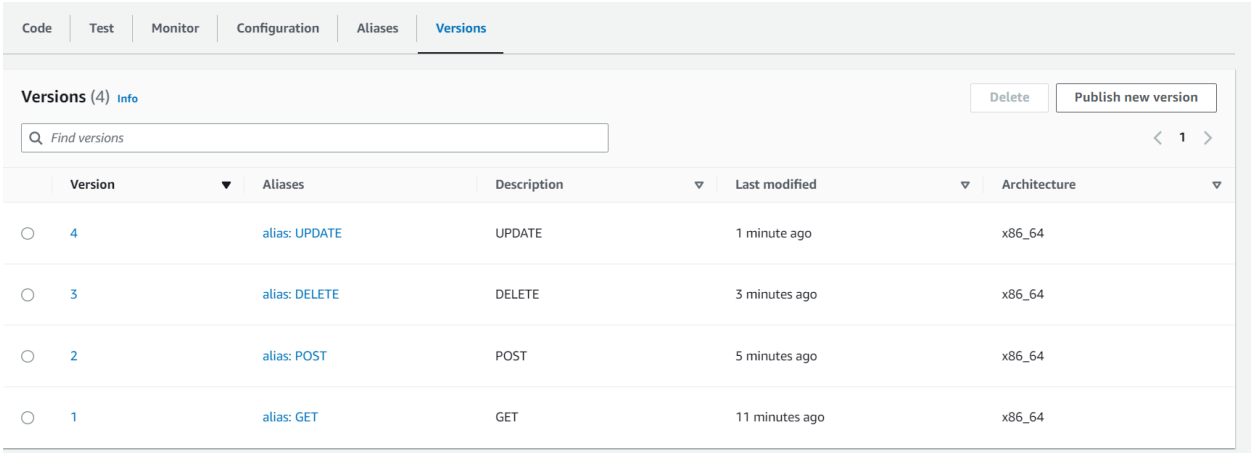


Updated the lambda function code to perform the **put method operation**. After that, I created the version and alias to the existing code.



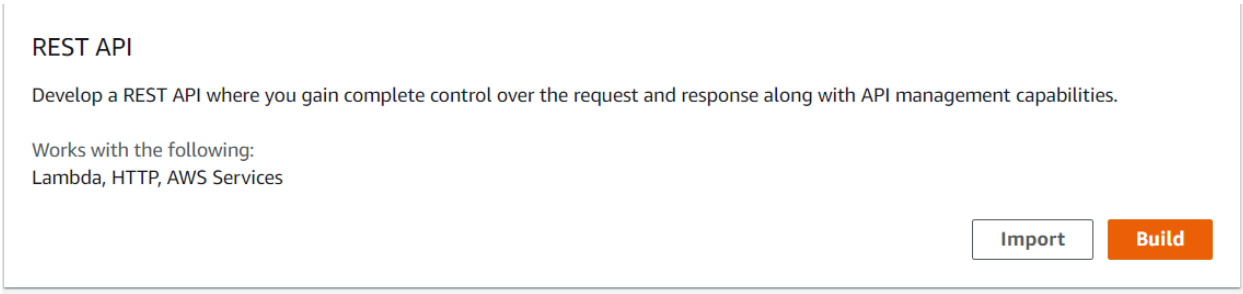
Updated the lambda function code to perform the **delete method operation**. After that, I created the version and alias to the existing code.

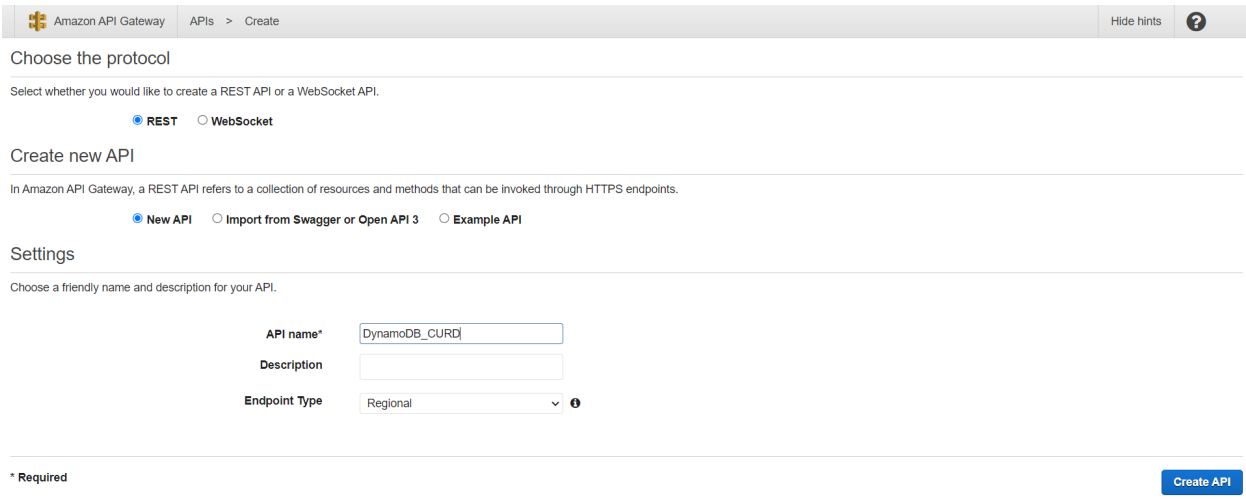




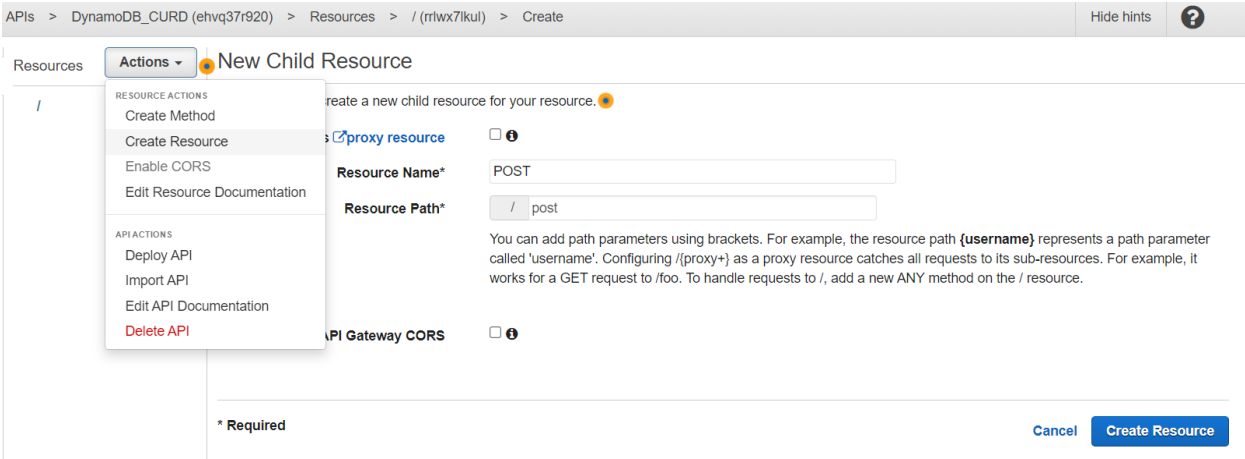
Finally to perform the CRUD operation on the Dynamodb table created all versions of the lambda function code.

Creating a API Gateway Rest API

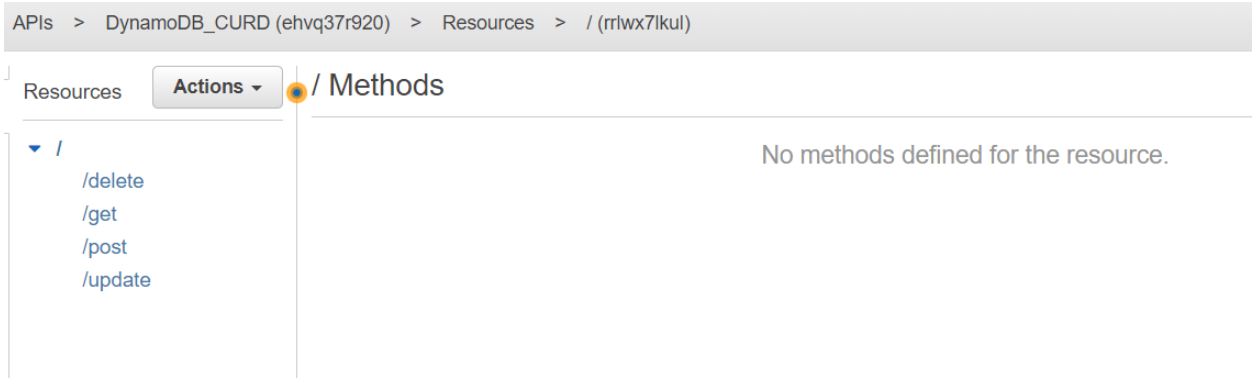




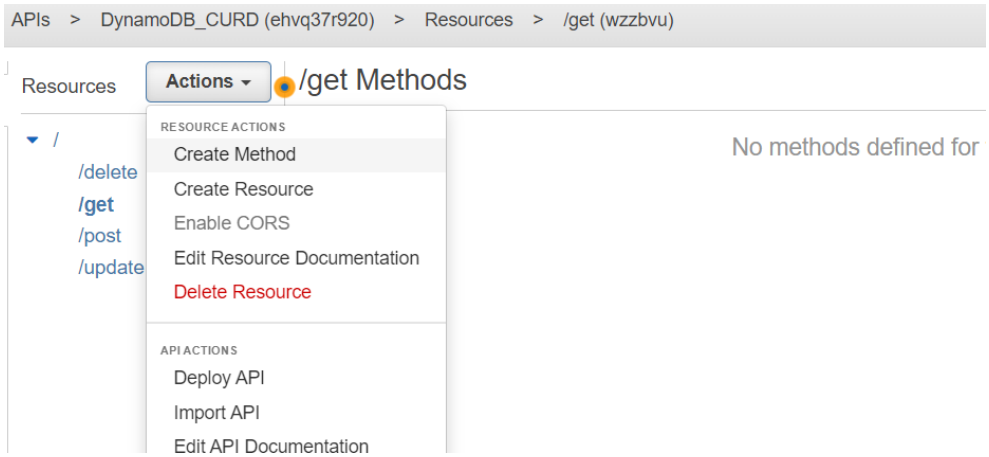
After creating the rest api creating a resource to the api which helps to perform the routing operation.



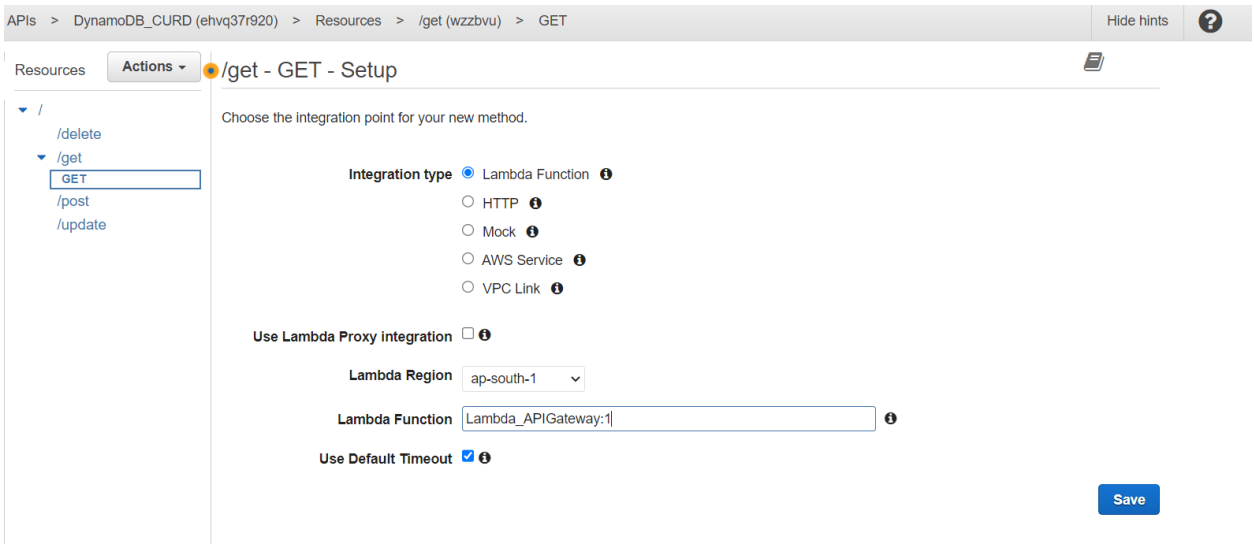
Same way finally created all routes resources now we can perform the routing with the following routes after deploying the API.



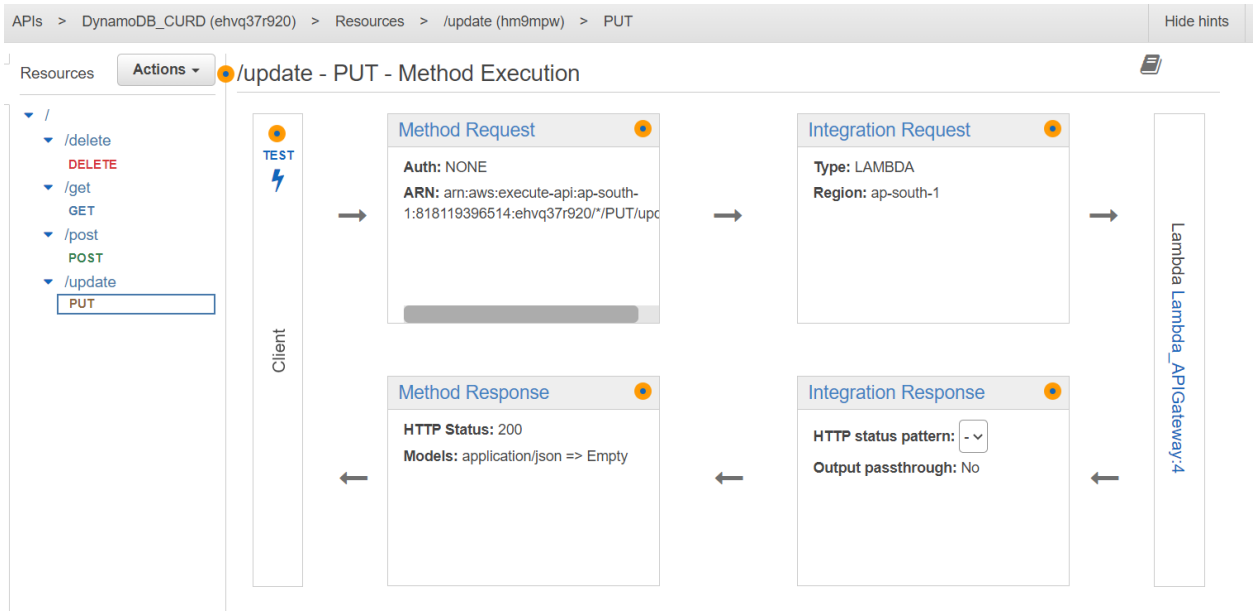
Creating a get method to the resource



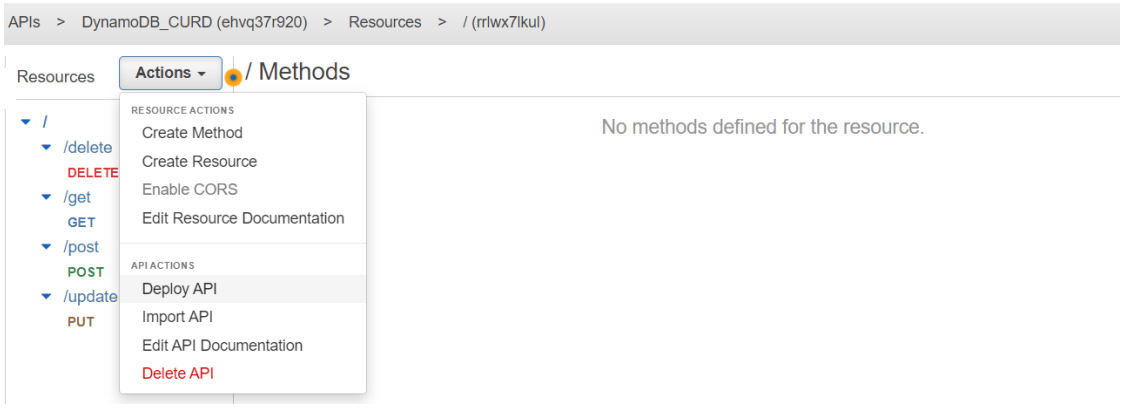
After that integrating the lambda function version to the get method



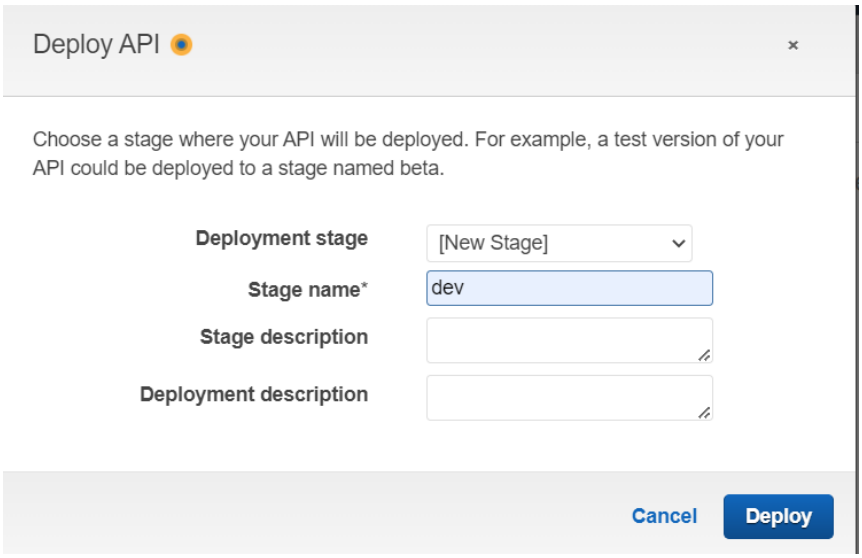
Sameway created methods and added the respective lambda function version to them.



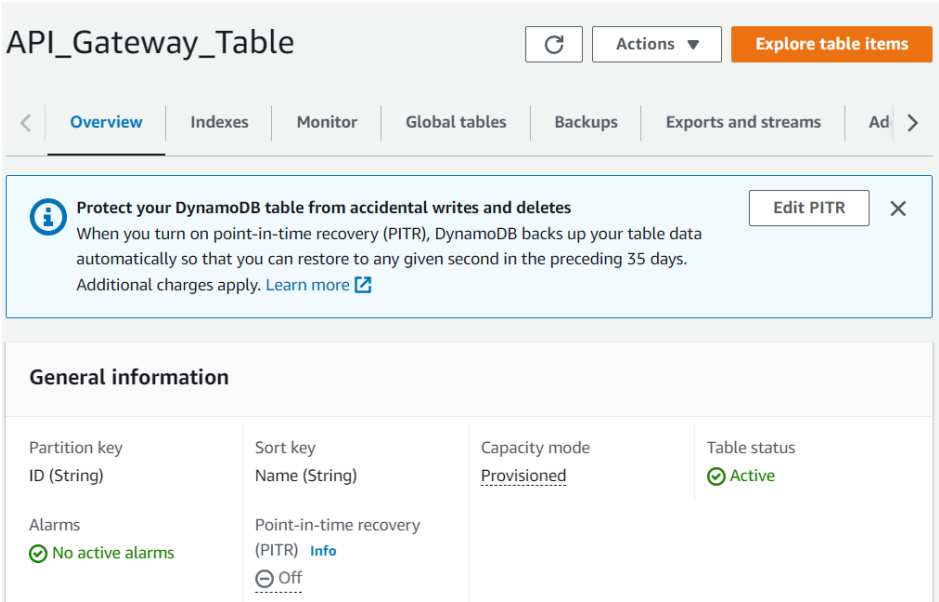
Finally deploying the api so we can trigger lambda function with the rest api.



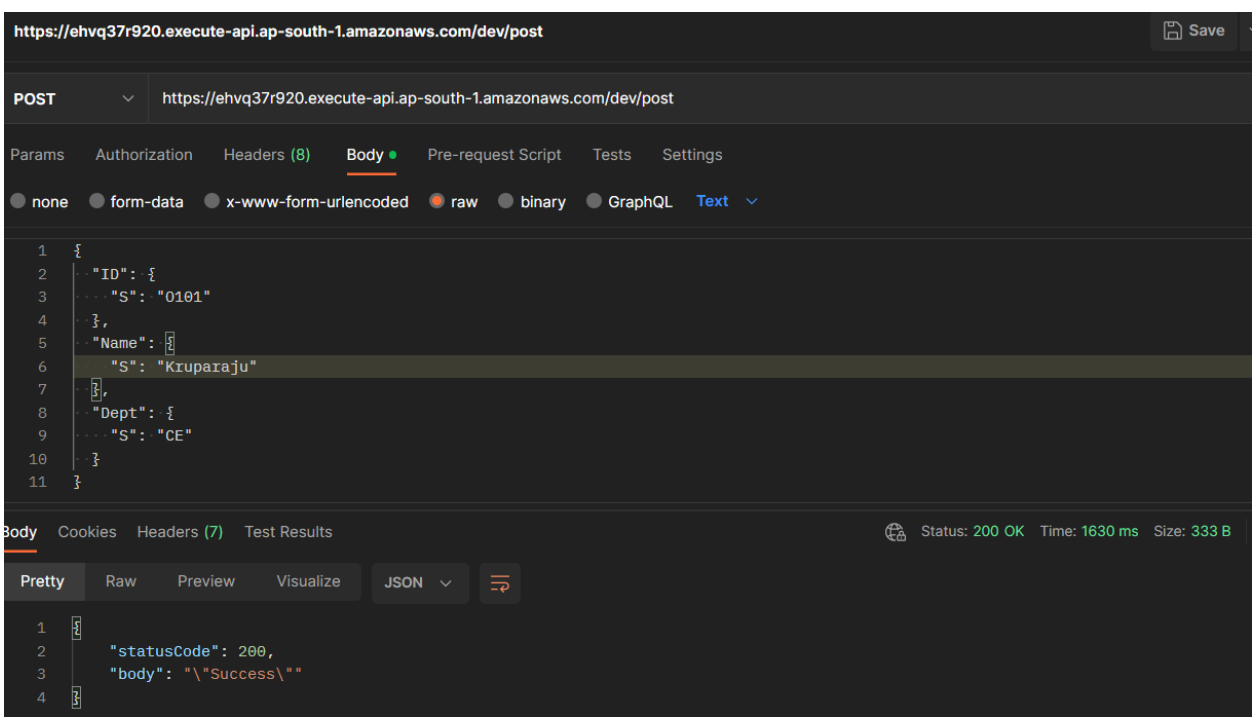
Deploying a API in the Dev stage



Created a Dynamodb table with ID as partition key and Name as a Sort key.



Performing CRUD operations with the help of a postman on the Dynamodb table. For the first time performing the post operation it will insert a new item in the DynamoDB table.



After performing the get operation it will list all items in the DynamoDB table.

GET https://ehvq37r920.execute-api.ap-south-1.amazonaws.com/dev/get Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL Text

Body Cookies Headers (7) Test Results Status: 200 OK Time: 1117 ms Size: 413 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "statusCode": 200,
    "body": "[{\"Dept\": {\"S\": \"CE\"}, \"ID\": {\"S\": \"0101\"}, \"Name\": {\"S\": \"Kruparaju\"}}]"
}
```


After performing the put operation it will update the Dept value of the item from CE to cloud engineering in the DynamoDB table.

https://ehvq37r920.execute-api.ap-south-1.amazonaws.com/dev/update Save

PUT https://ehvq37r920.execute-api.ap-south-1.amazonaws.com/dev/update

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

none form-data x-www-form-urlencoded raw binary GraphQL Text


```
{
  "ExpressionAttributeNames": {
    "#AT": "Dept"
  },
  "ExpressionAttributeValues": {
    ":t": {
        "S": "Cloud Enginnering"
    }
  },
  "Keys": {
  "ID": {
    "S": "0101"
  },
  "Name": {
    "S": "Kruparaju"
  }
},
"UpdateExpression": "SET #AT = :t"
}
```


Body Cookies Headers (7) Test Results Status: 200 OK Time: 1587 ms Size: 333 B

Pretty Raw Preview Visualize JSON


```
{
    "statusCode": 200,
    "body": "\"Success\""
}
```


After that performed the get operation to list the changes done.

GET https://ehvq37r920.execute-api.ap-south-1.amazonaws.com/dev/get Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL Text

Body Cookies Headers (7) Test Results Status: 200 OK Time: 1794 ms Size: 423 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "statusCode": 200,
    "body": "[{\"Dept\": {\"S\": \"Cloud Enginnering\"}, \"ID\": {\"S\": \"0101\"}, \"Name\": {\"S\": \"Kruparaju\"}}]"
}
```


Finally deleting the item from the dynamo db table.

DELETE https://ehvq37r920.execute-api.ap-south-1.amazonaws.com/dev/delete Send

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

none form-data x-www-form-urlencoded raw binary GraphQL Text


```
{
  "ID": {
    "S": "0101"
  },
  "Name": {
    "S": "Kruparaju"
  }
}
```


Body Cookies Headers (7) Test Results Status: 200 OK Time: 2.04 s Size: 333 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "statusCode": 200,
    "body": "\"Success\""
}
```


GET https://ehvq37r920.execute-api.ap-south-1.amazonaws.com/dev/get Send

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

Body Cookies Headers (7) Test Results Status: 200 OK Time: 1332 ms Size: 319 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "statusCode": 200,
    "body": "[]"
}
```