Contents

Lab 9: Serverless Application

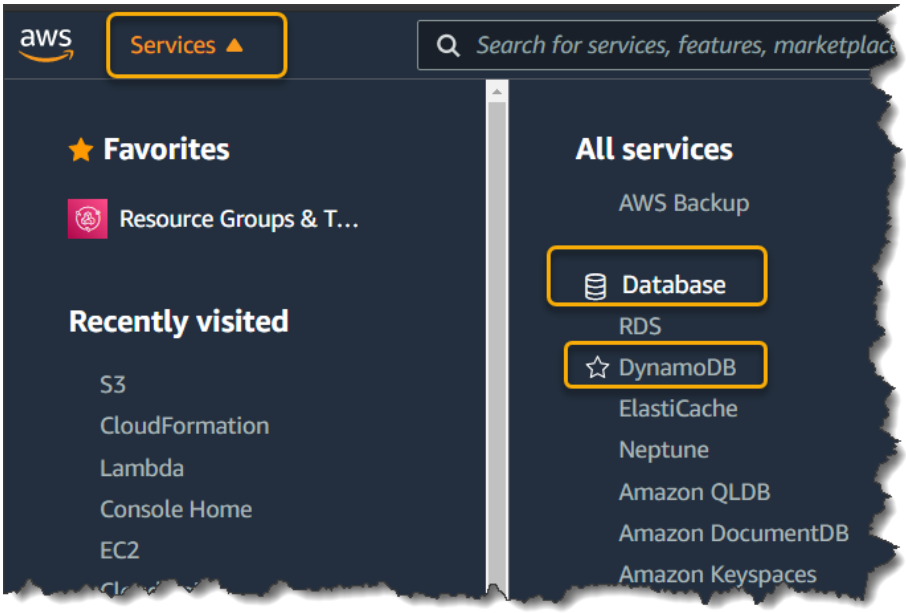
In this lab we are going to create a serverless application using S3, API Gateway, Lambda and DynamoDB

Task Breakdown:

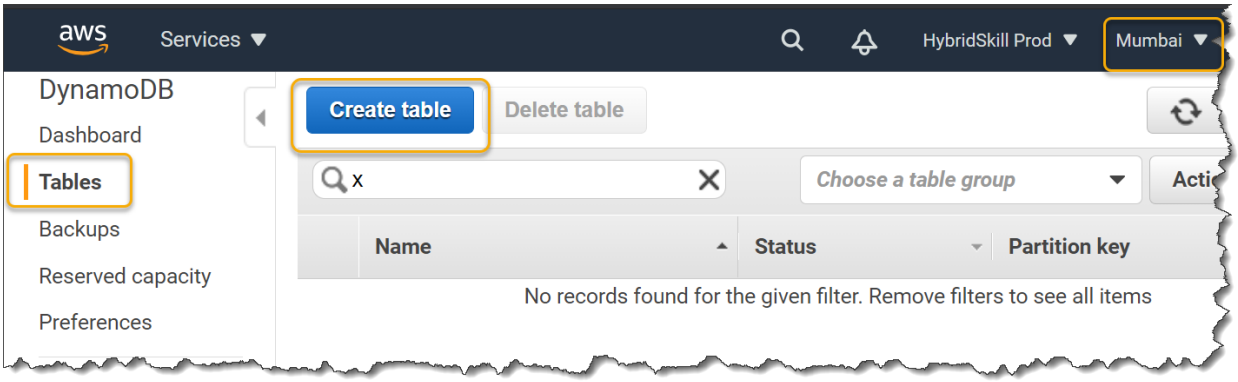
- Create a DynamoDB table
- Create an IAM role
- Create Lambda function
- Create an API gateway
- Set up a Static S3 Website

Task 1: Create a DynamoDB table

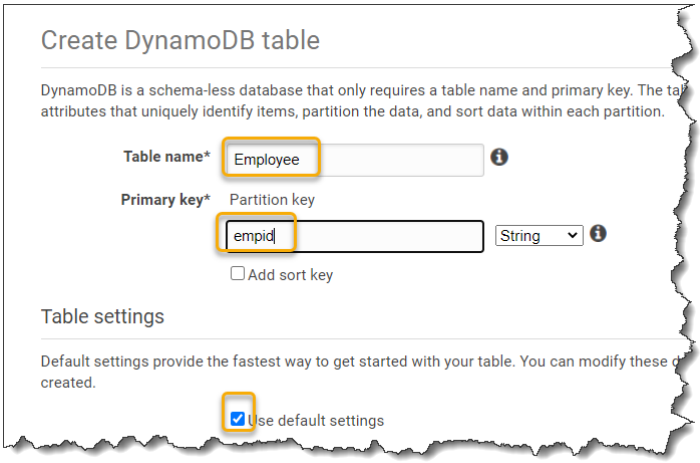
1. Click on **Services** and under **Databases** click on **DynamoDB**



2. Make sure your **Region** is selected, click on **Tables** on the left pane and click on **Create table**. We have selected **Mumbai** as an example.



3. Enter **Employee** as the **Table name**. and **empid** as the **Primary key.** Leave all other options as default.



Click **Create**.

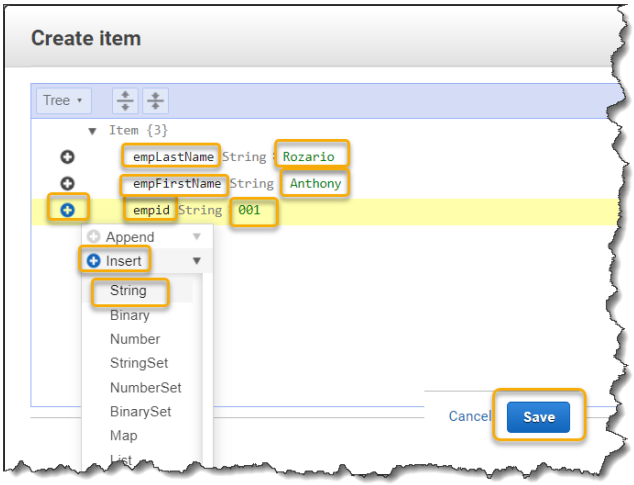
4. Click the **+ sign** and select **Insert** and choose **String** as the value.

Enter the information as follows.

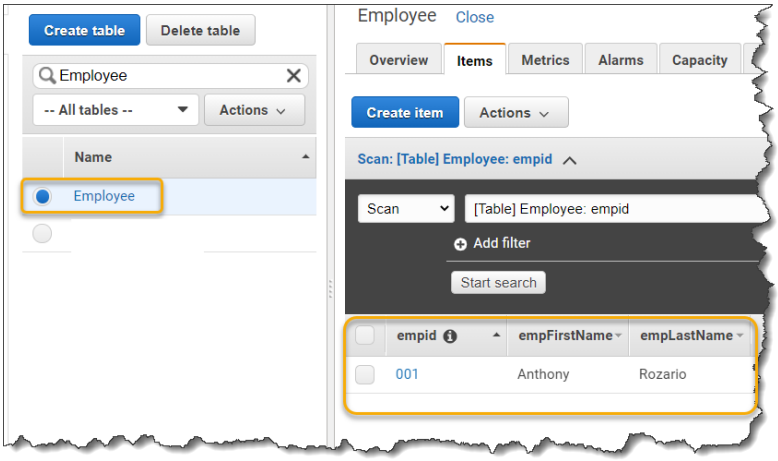
empid : 001

empFirstName: *Your firstname*

empLastname: *YourLastname.* **Click Save**

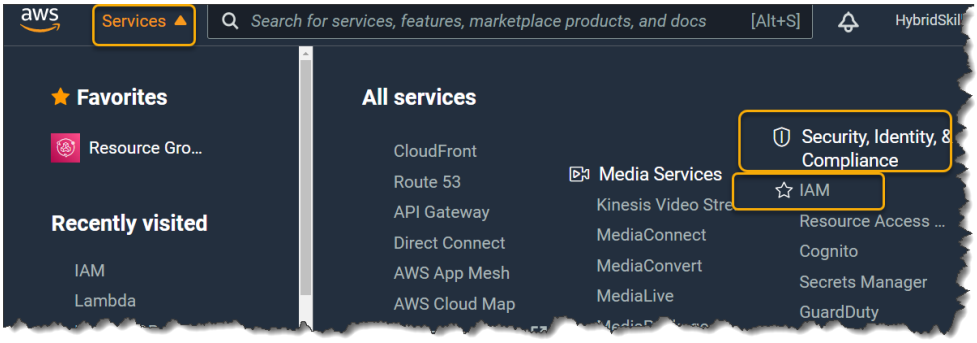


5. Your DynamoDB table should look like this.

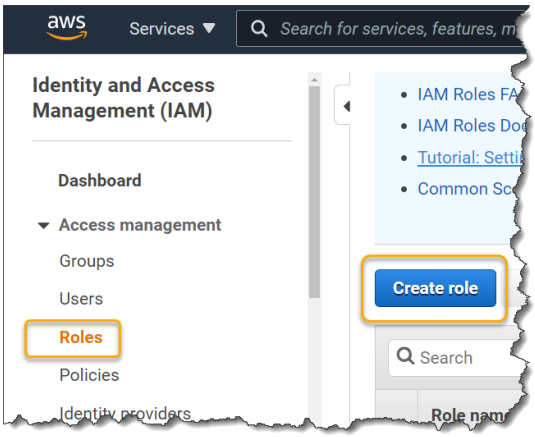


Task 2: Create an IAM role

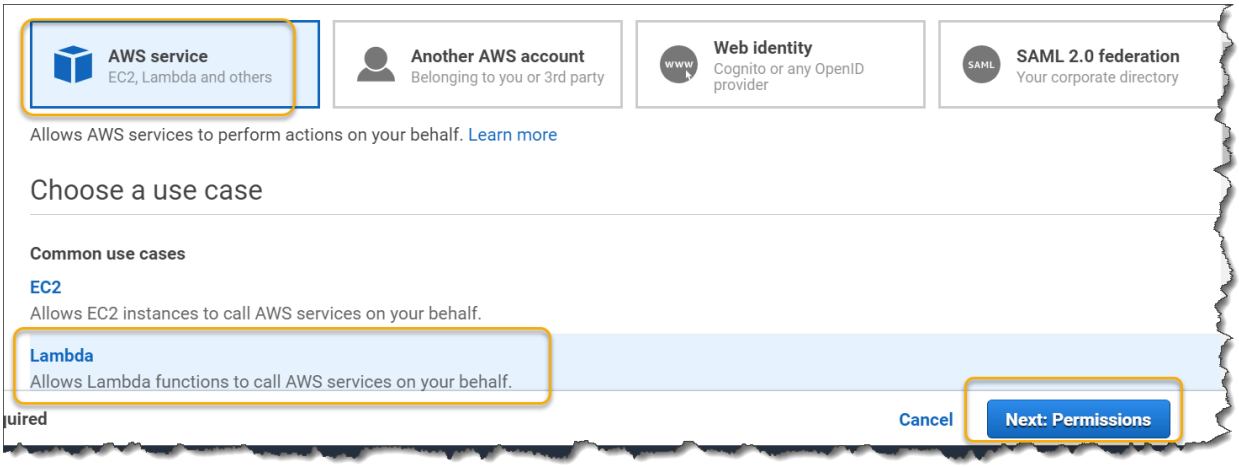
1. We are now going to create an **IAM role** that will allow our Lambda functions to read/write into DynamoDB. Click on **Services** and then under **Security, Identity & Compliance** click on **IAM**



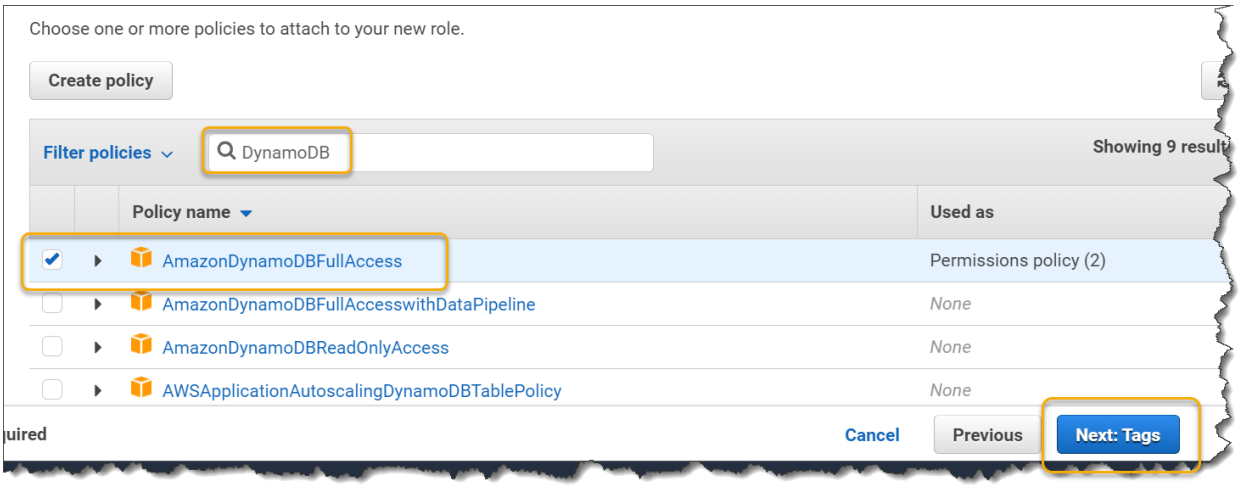
2. On the left pane find and Click **Roles** and then on **Create role**



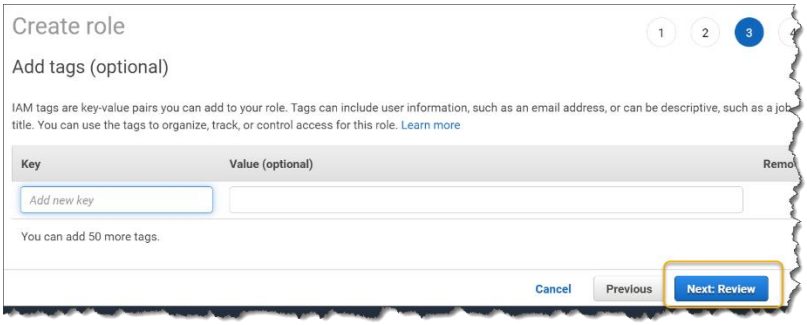
3. Select **AWS service**, then click on **Lambda** and then **Next:Permissons**



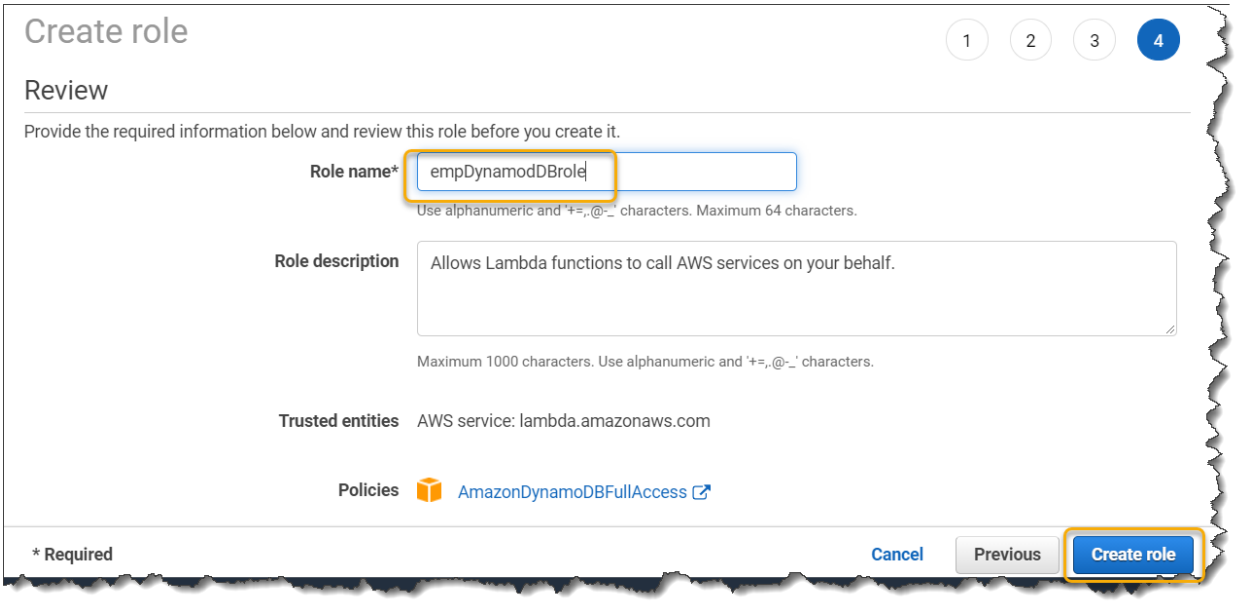
4. Search for **DynamoDB** and then select the policy **AmazonDynamoDBFullAccess** and click on **Next:Tags**



5. Click **Next:Review**



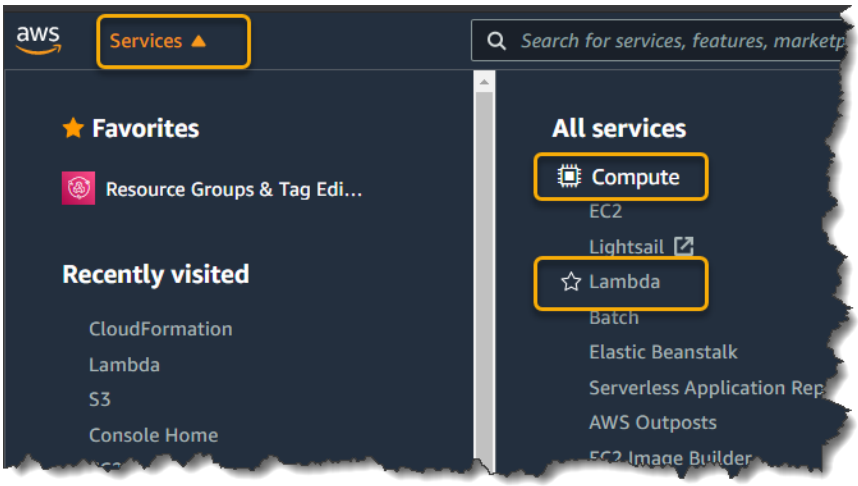
6. Give **empDynamoDBrole** as the **Role name** as click **Create role**.



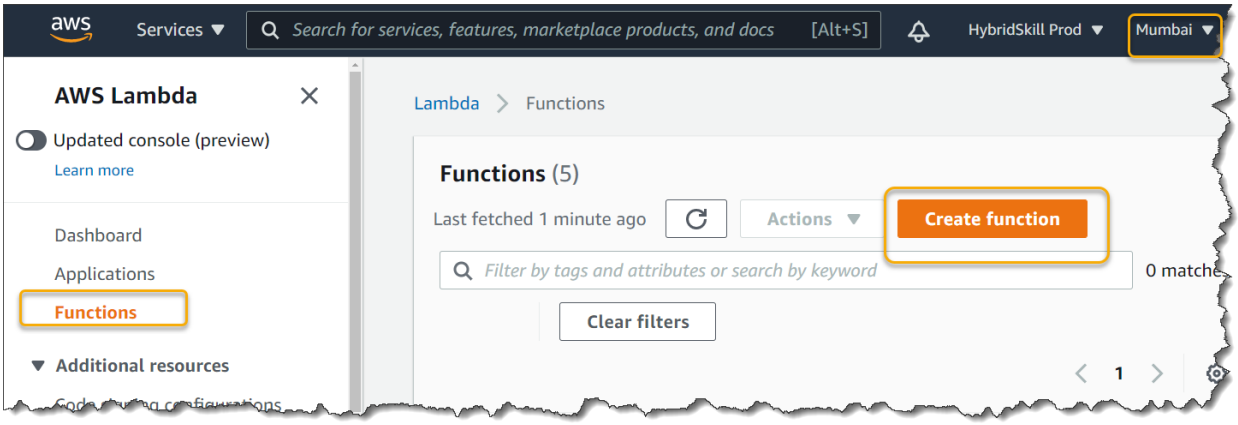
Task 3: Create Lambda functions

1. We are now going to create 2 Lambda functions One for getting data from DynamoDB(getemp.py) and the other for putting data(putemp.py) into DynamoDB.
   Download and unzip the code from the following location
   https://hybridskill-training.s3.amazonaws.com/genpact/serverlessapp.zip

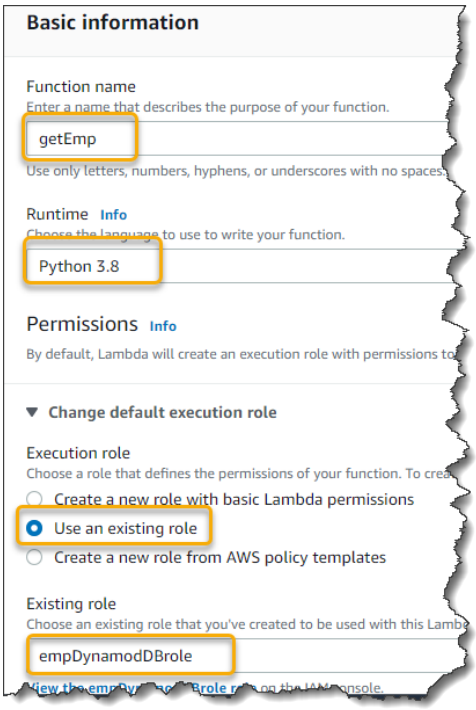
2. Click on **Services** and then under **Compute** click on **Lambda**



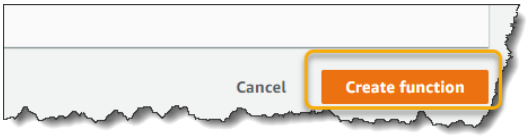
3. Make sure your **Region** is selected, click on **Functions** on the left pane and click on **Create function**. We have selected **Mumbai** as an example



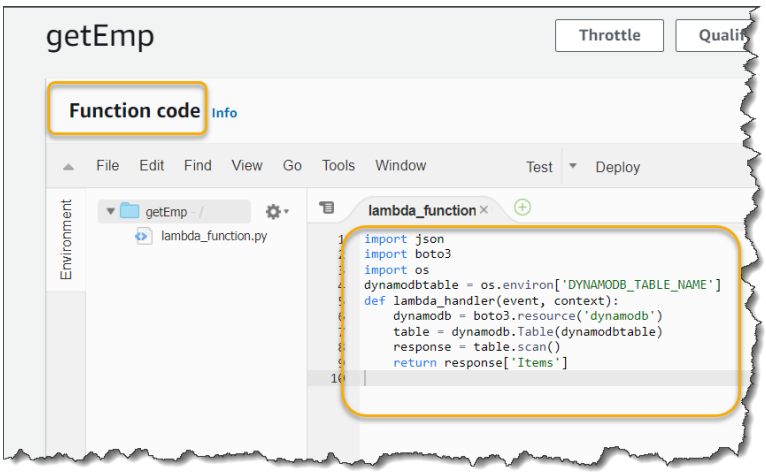
4. Enter **getEmp** as the **Function name**. Choose the **Runtime** as **Python 3.8.** for **Execution role** choose **Use an existing role** and choose the **empDynamodbRole** you created earlier.



Click **Create function.**

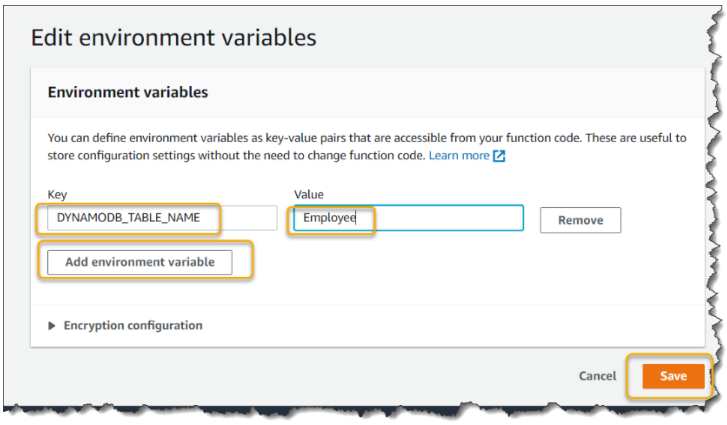


5. Open and copy.paste the contents of **getemp.py***(from the zip you downloaded earlier)* into the function code editor as shown below

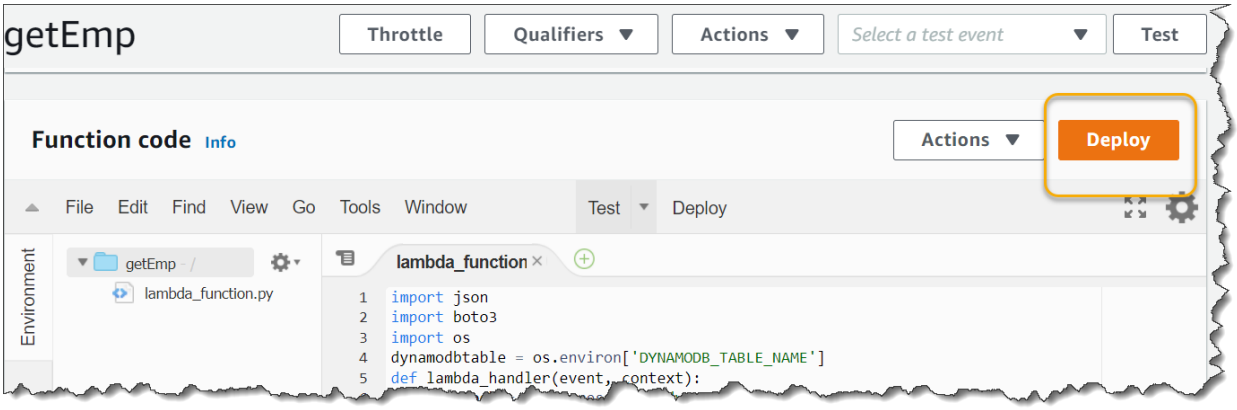


6. Scroll down to **Environment variables** and click **Edit**

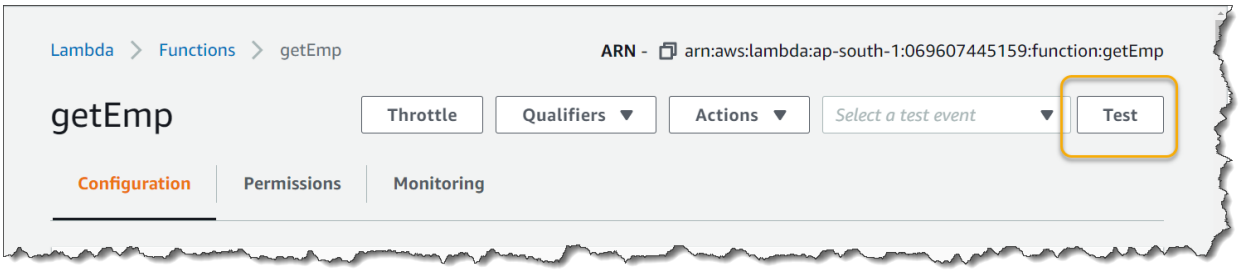
7. Click **Add environment variable** and enter the **Key** as **DYNAMODB_TABLE_NAME** and the **Value** as the name of your DynamoDB table which in our case is **Employee.** Click Save



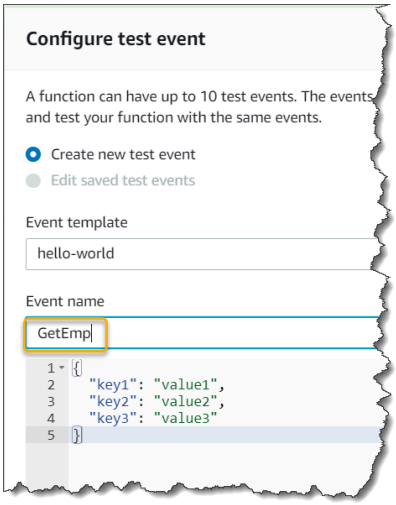
8. Scroll back up and click **Deploy**



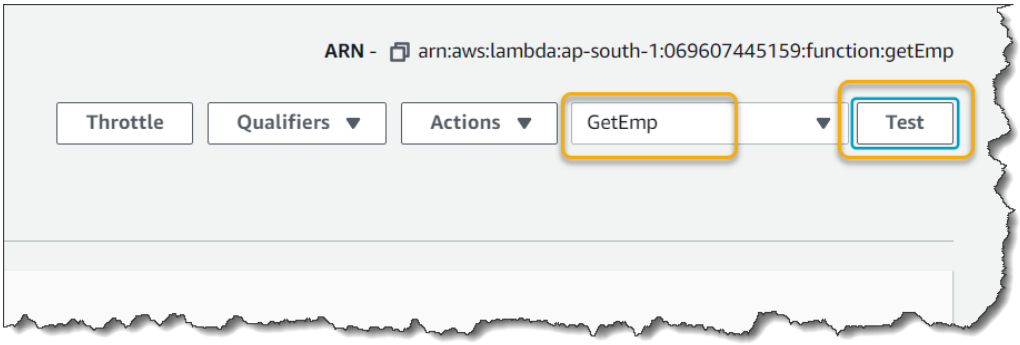
9. Next lets test out our Lambda function. Scroll up to the top and click **Test.**



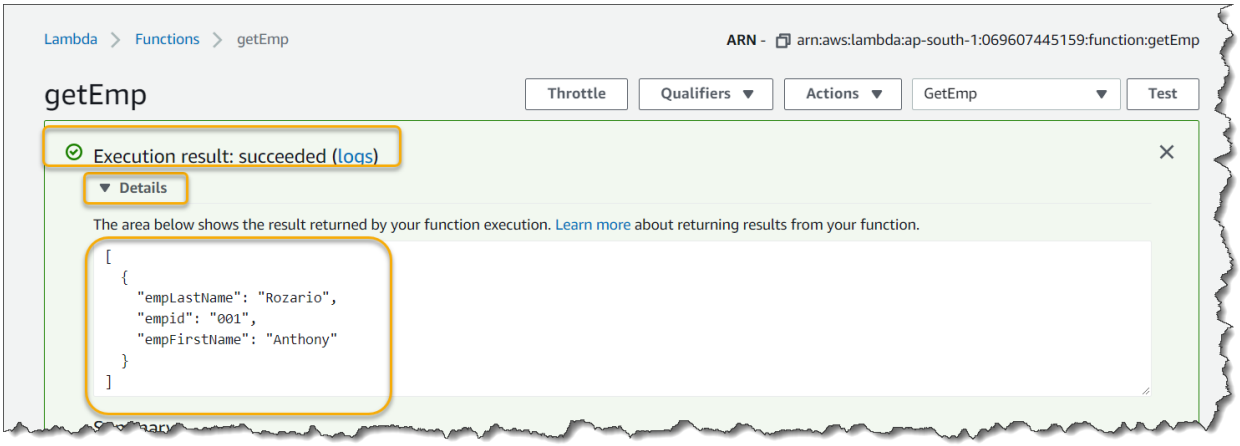
10. Enter **GetEmp** as the name of the test event, leave the values as default and click **Create.**



11. With the **GetEmp** event selected click **Test.**

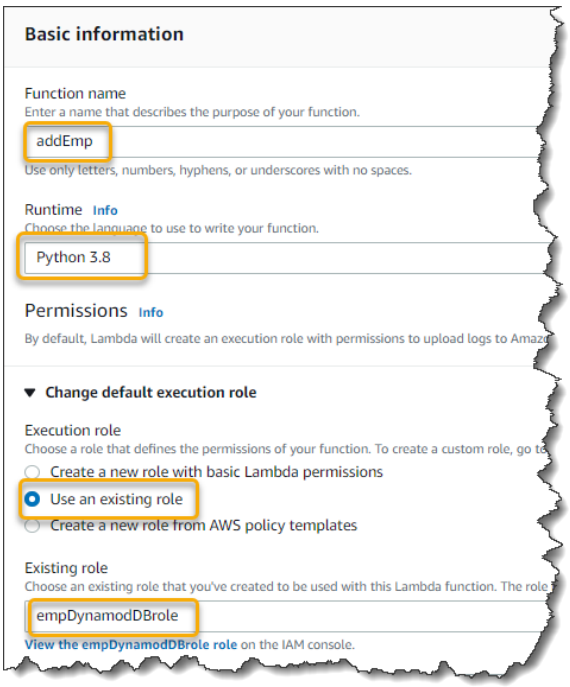


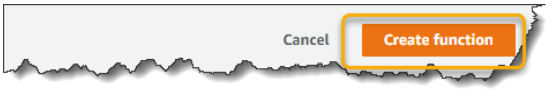
12. Expand **Details** to view the execution results.



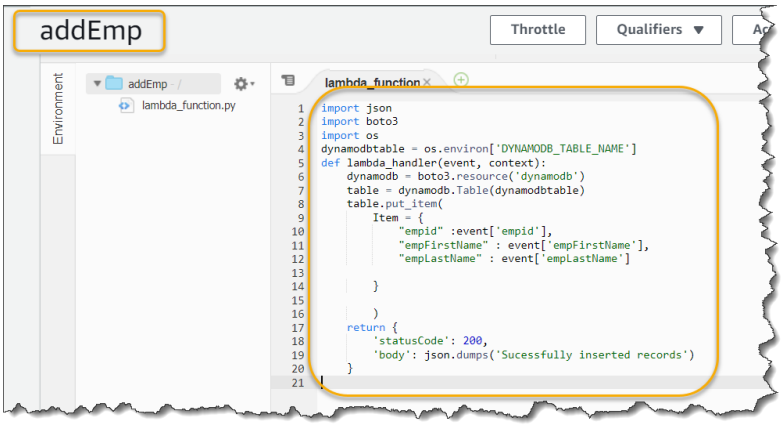
13. Let's Create the a**ddEmp** Lambda function next. Click **Functions** and **Create function**

14. This time give the **Function name** as **addEmp** and as before select the **runtime** as **Python.3.8** and select **empDynamoDBrole** as the **execution role**. Click **Create function**



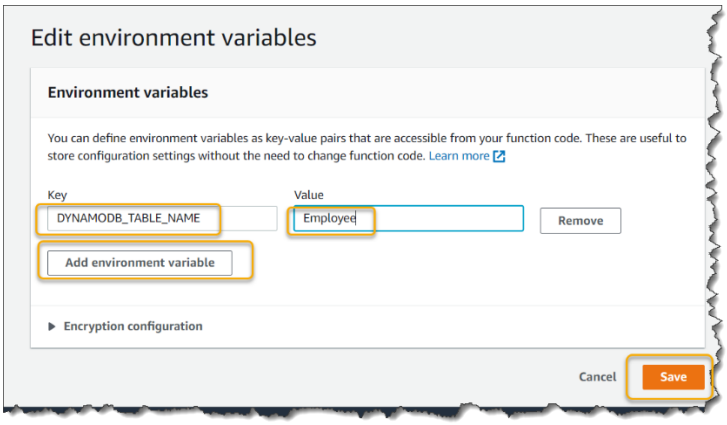


15. Open and copy.paste the contents of **addemp.py***(from the zip you downloaded earlier)* into the function code editor as shown below

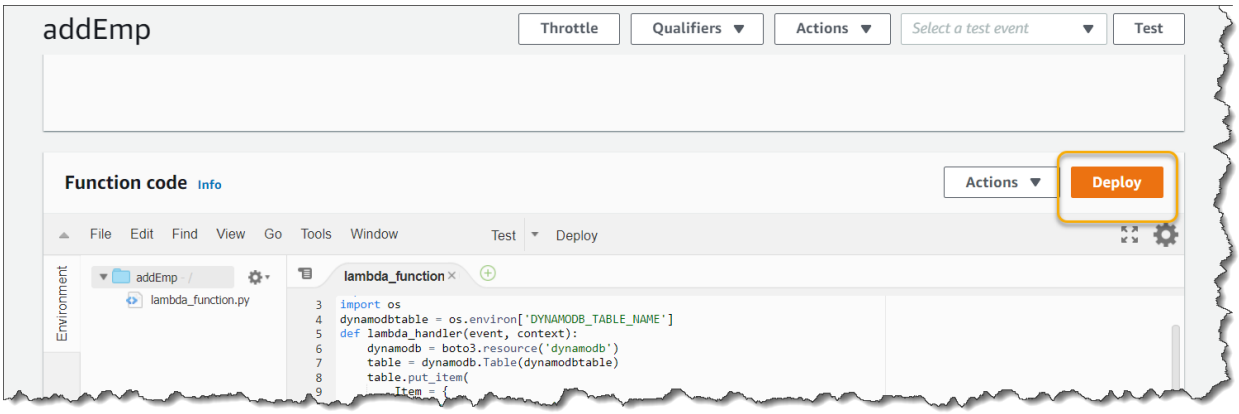


16. Scroll down to **Environment variables** and click **Edit**

17. Click **Add environment variable** and enter the **Key** as **DYNAMODB_TABLE_NAME** and the **Value** as the name of your DynamoDB table which in our case is **Employee.** Click **Save**



18. Scroll back up and click **Deploy**



19. Next lets test out our Lambda function. Scroll up to the top and click **Test.**

20. Enter **AddEmp** as the name of the test event, Enter the values of another employee as follows

{

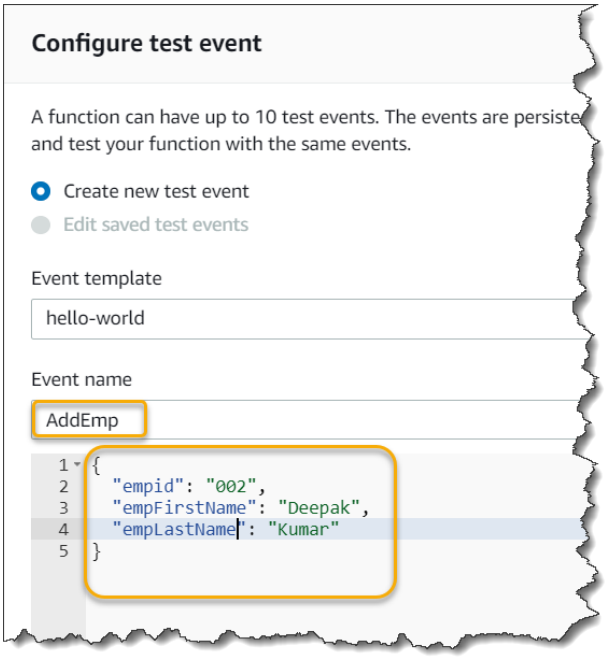
“empid”:”002”,

“empFirstName”:”Deepak”,

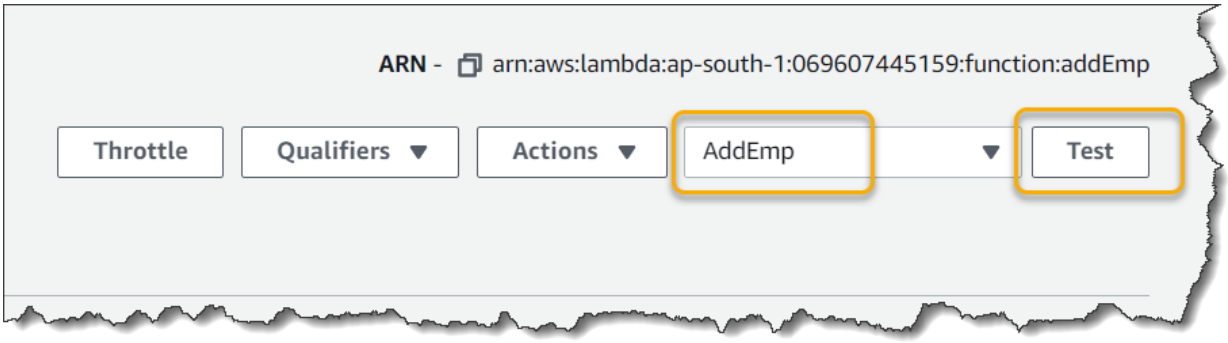
“empLastName”:”Kumar”

}

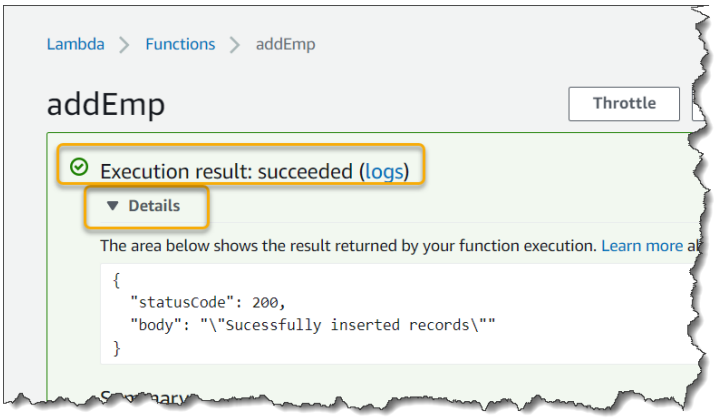
click **Create.**



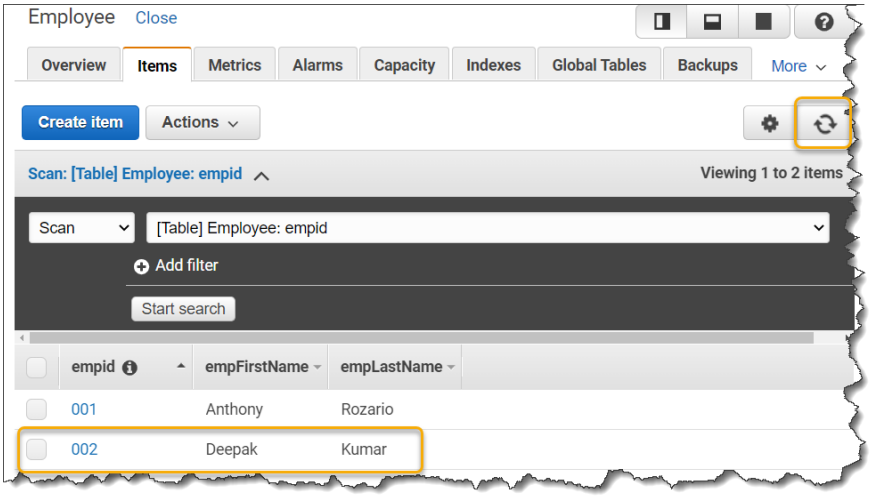
21. With the **AddEmp** event selected click **Test.**



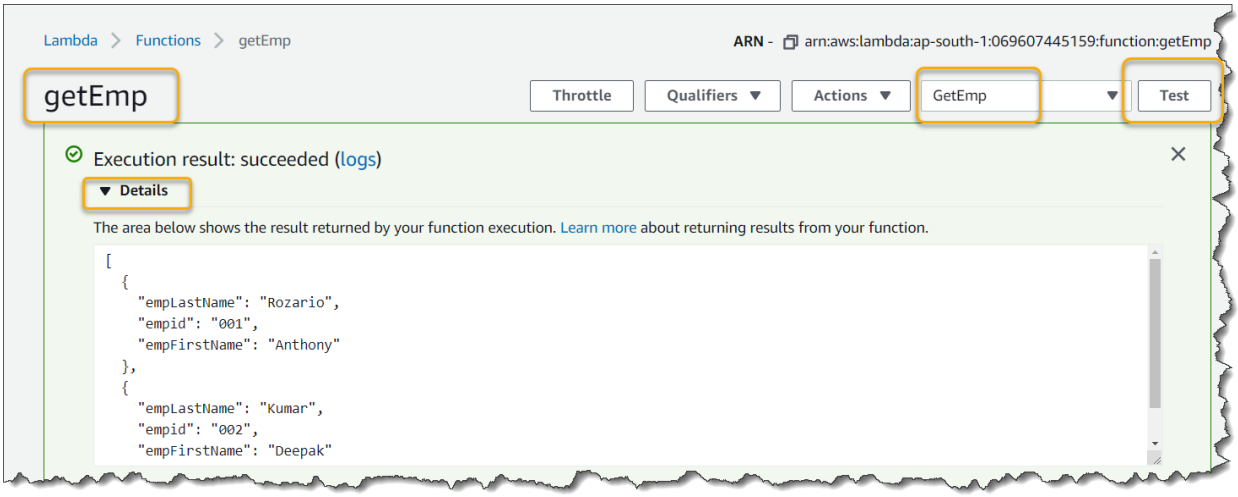
22. Expand **Details** to view the execution results.



23. Check your DynamoDB table for the new entry.

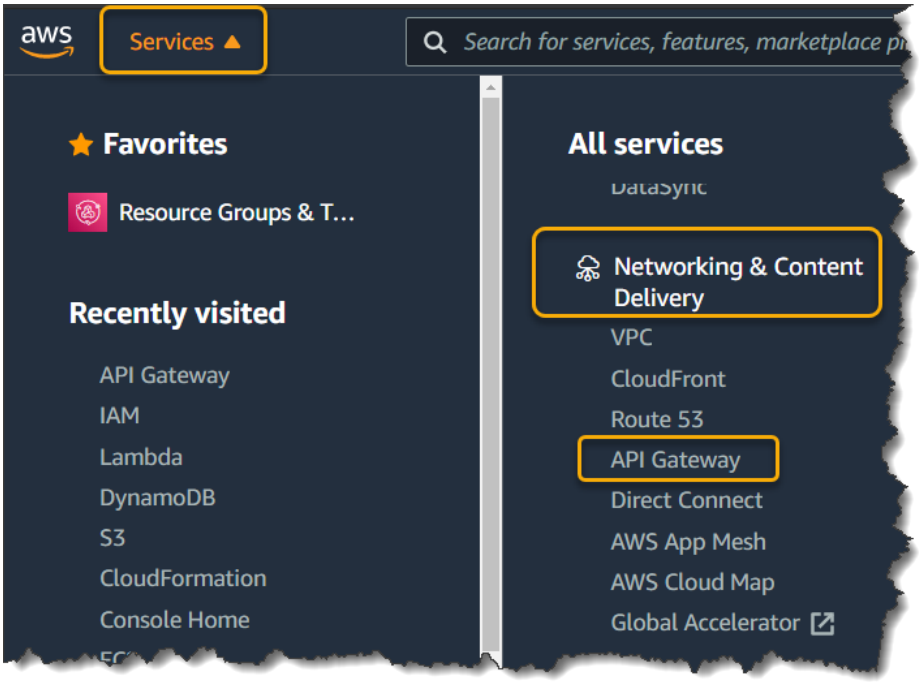


24. You can alternatively run your **getEmp** Lambda function and test if it returns the new records.

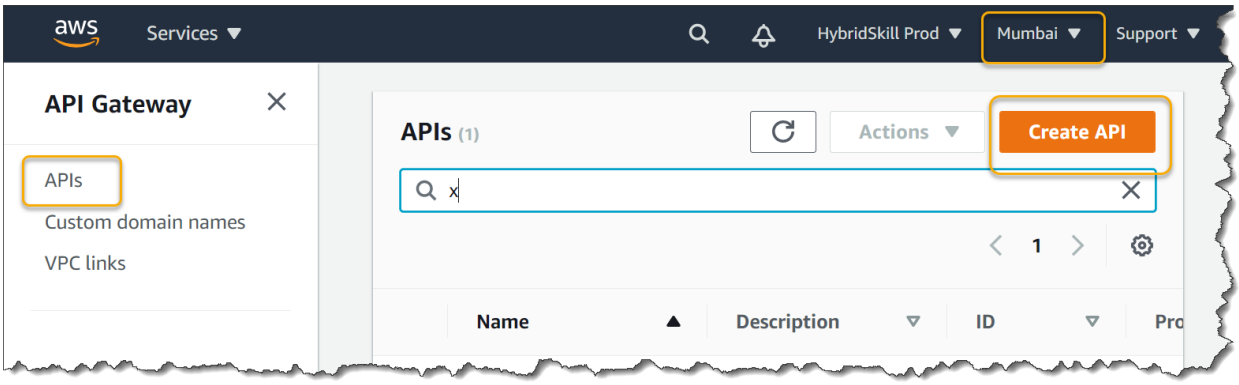


Task 4: Create an API gateway

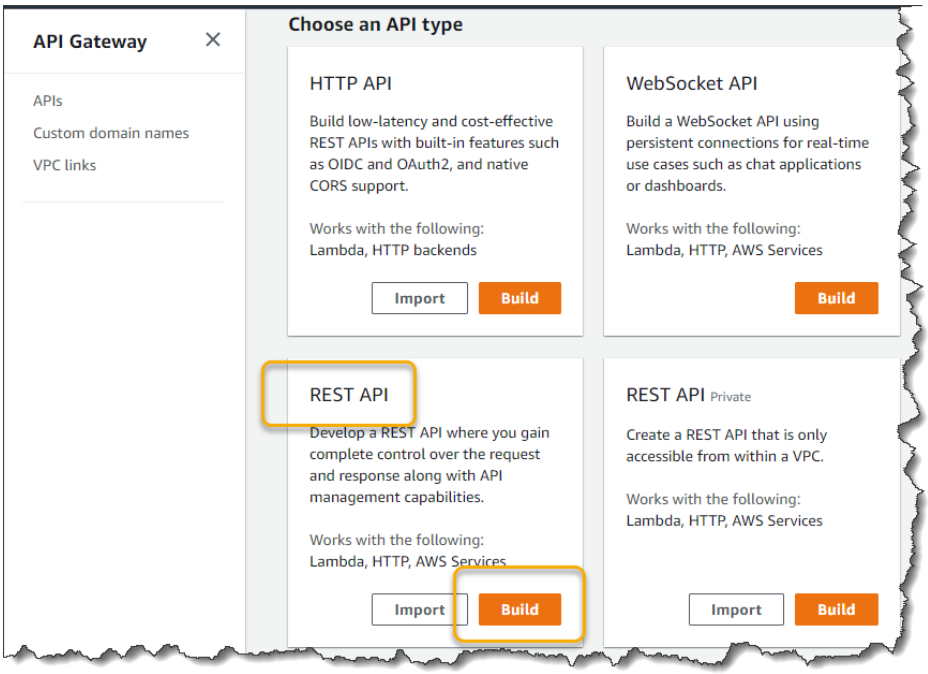
1. We will next create an API to expose our Lambda functions. Click **Services** and under **Networking & Content Delivery** click **API Gateway**



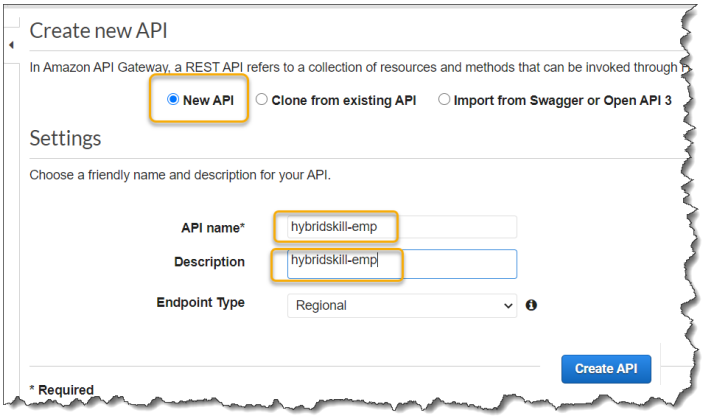
2. Make sure your **Region** is selected, click on **APIs** on the left pane and click on **Create API**. We have selected **Mumbai** as an example.



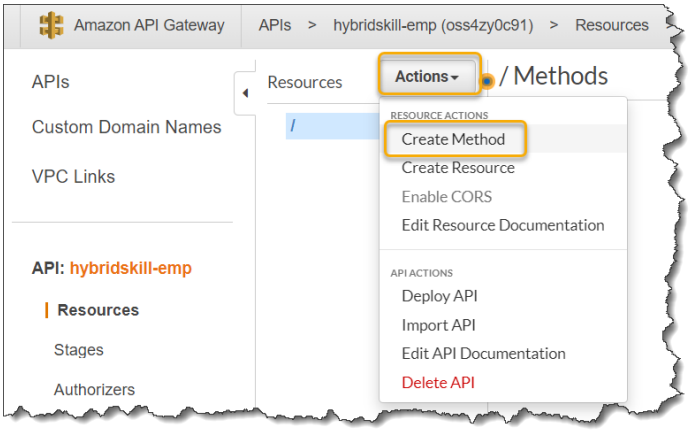
Choose the **API type** as **REST API** and click **Build**



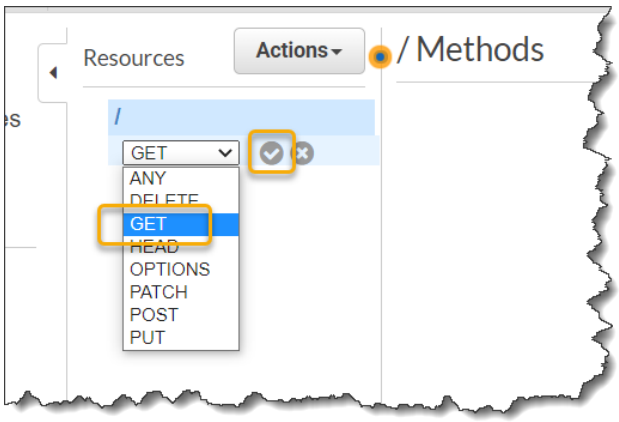
3. Select new API, enter **hybridskill-emp** as the name of your **API**, enter a description and click **Create API**



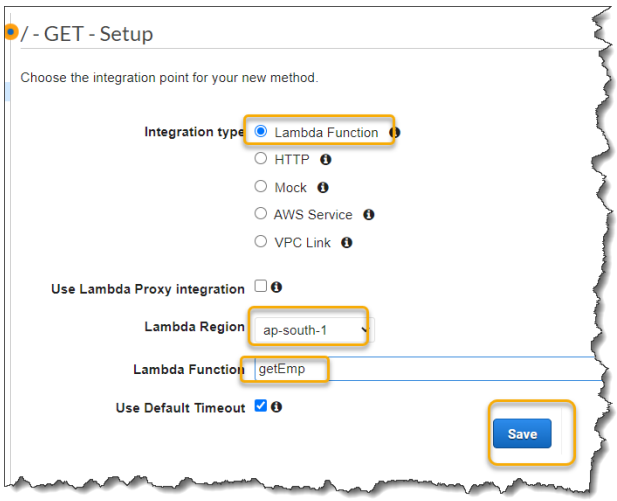
4. Select **Actions** and then **Create Method**



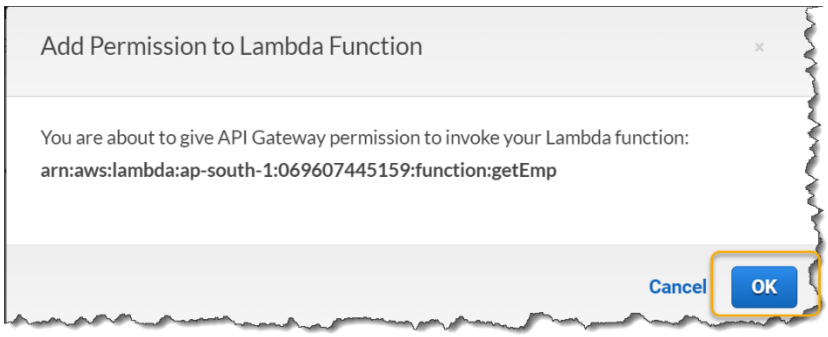
5. Select **GET** as the method and **Save**



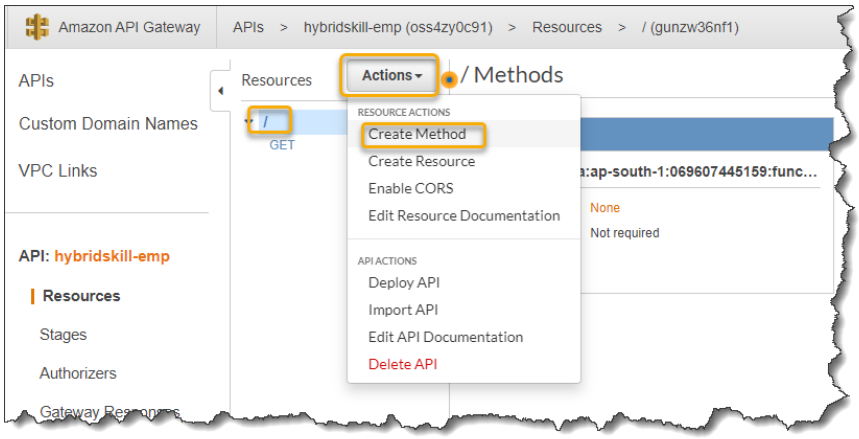
6. Select the **Integration type** as **Lambda Function.** Select **ap-south-1** as the **Lambda Region** select your **getEmp** Lambda function**.** Click **Save**



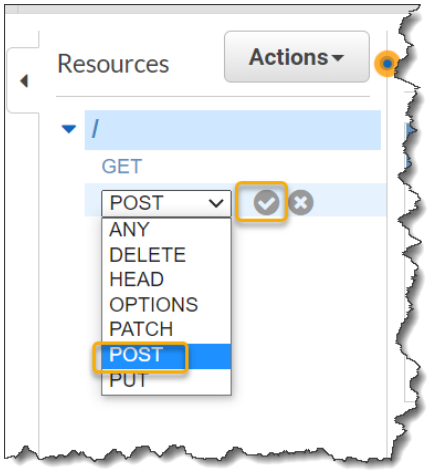
7. Click **OK** to give API gateway permissions to invoke your Lambda function.



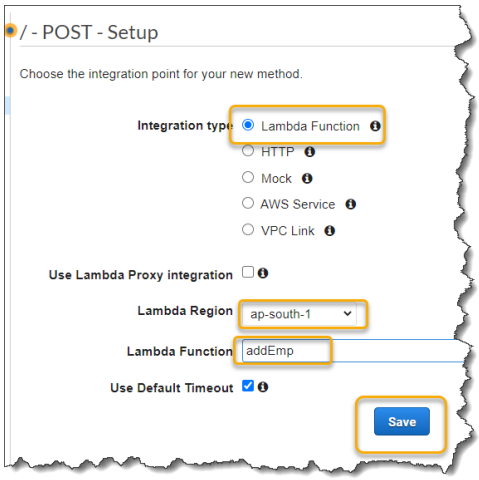
8. Lets create our POST request next. Select your API gateway and the click **/** and then click **Actions** and then **Create Method.**



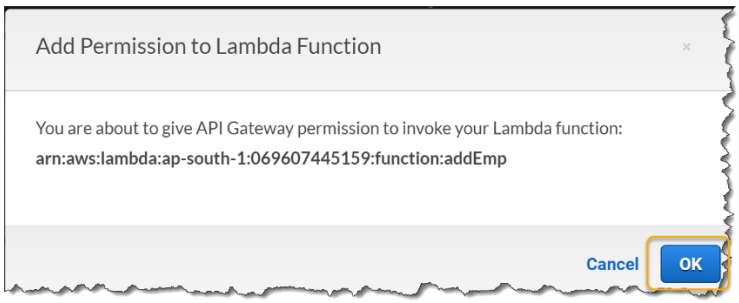
9. This time select **POST** as the method and **Save**



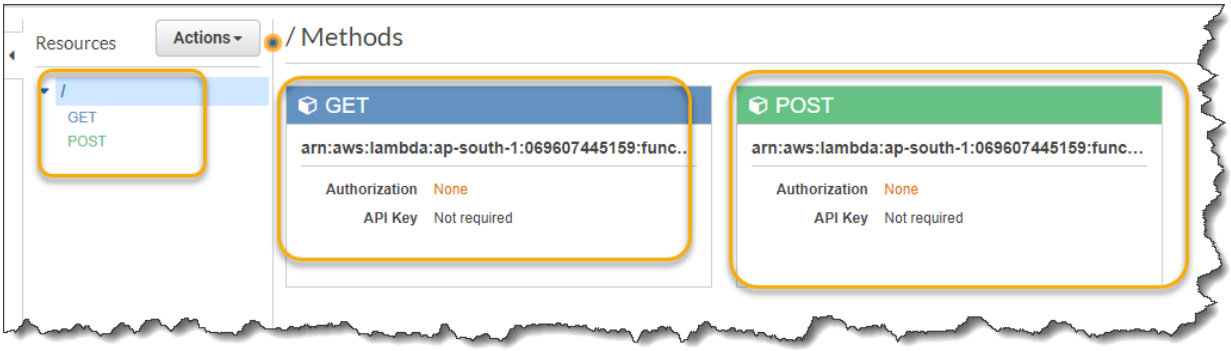
10. Select the **Integration type** as **Lambda Function.** Select **ap-south-1** as the **Lambda Region** select your **addEmp** Lambda function**.** Click **Save**



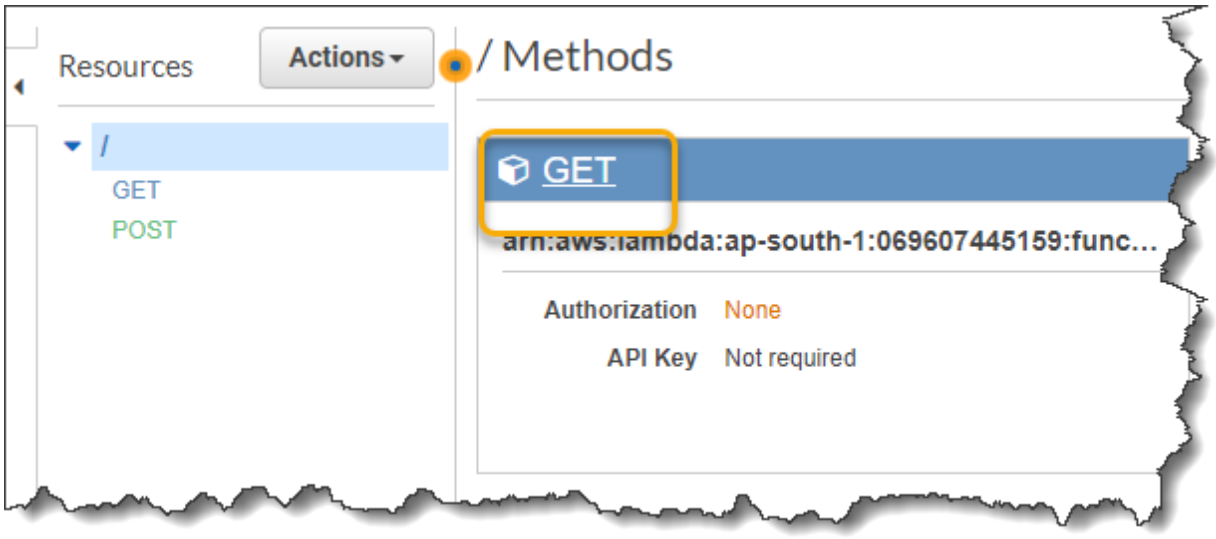
11. Click **OK** to give API gateway permissions to invoke your Lambda function



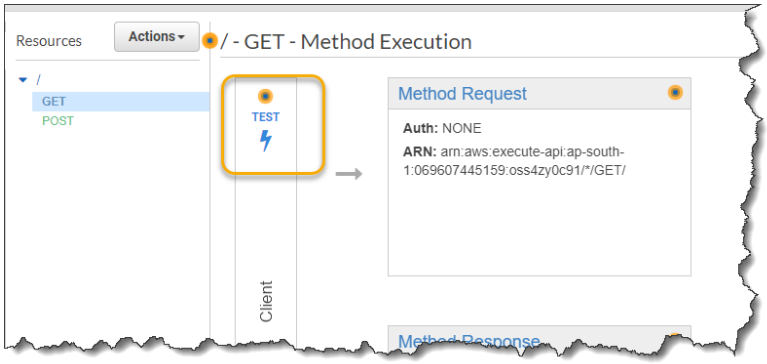
12. You should have 2 APIs setup. GET and POST.



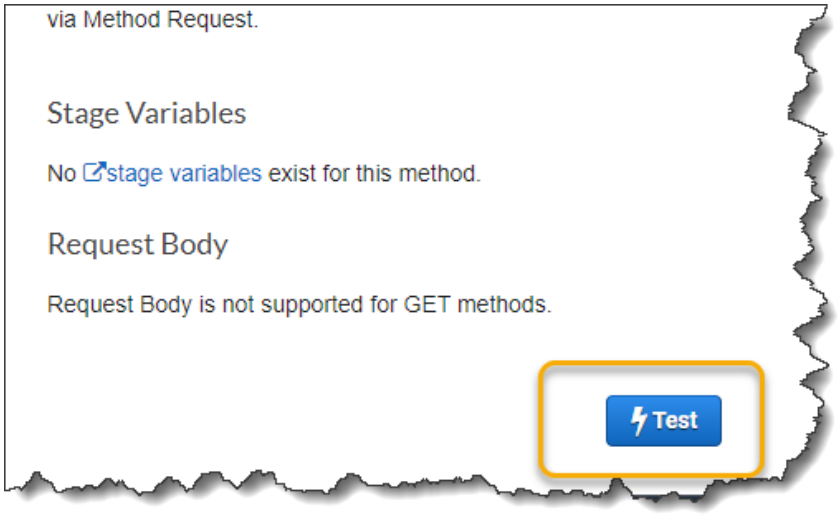
13. Lets test out the GET API first. Click on **GET**



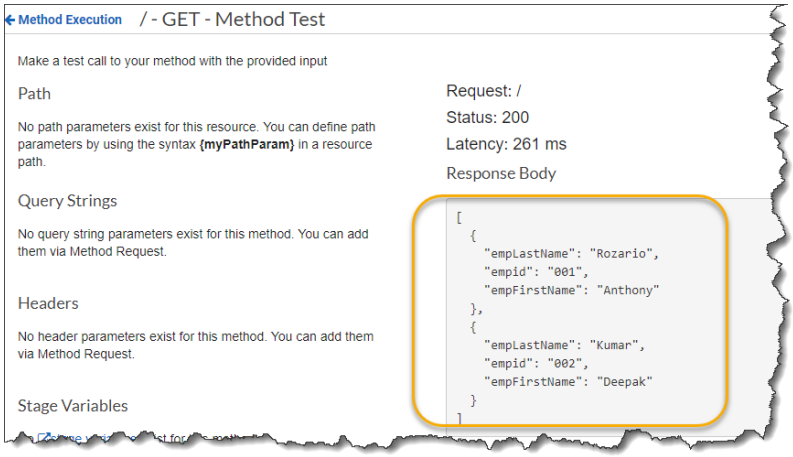
14. Next click on **Test**



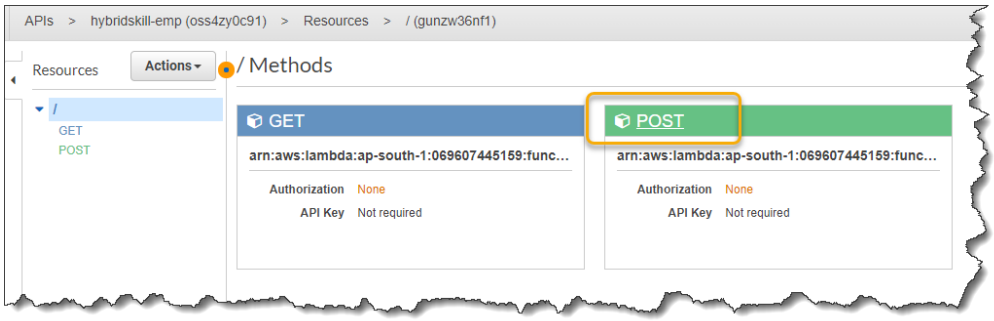
15. Scroll all the way to the bottom and click **Test**



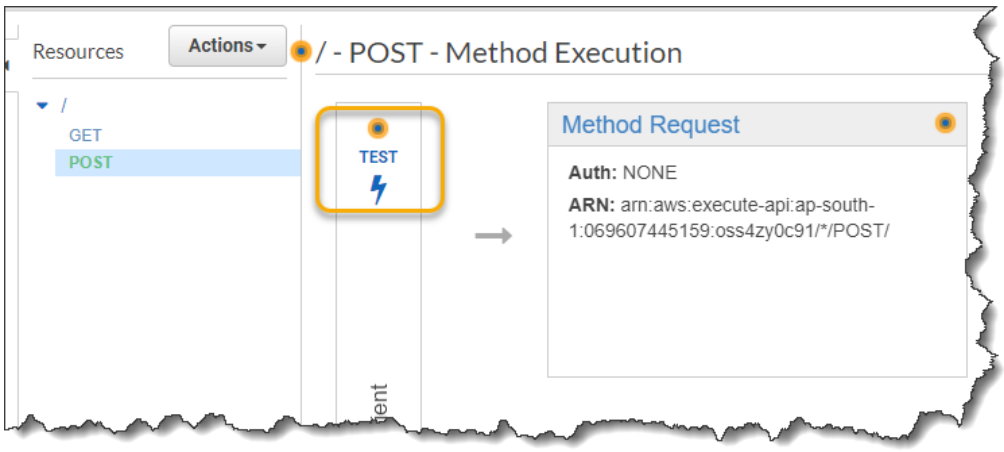
16. You should see your Employee data in the Response.



17. Lets test out our POST API next. Click on **POST**



18. Click on **TEST**



19. Scroll down, in the **Request Body**, enter the values of another employee as follows and click **Test**

{

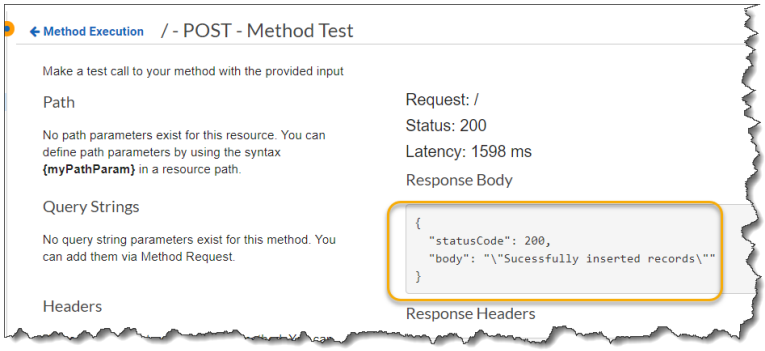
“empid”:”003”,

“empFirstName”:”Siddharth”,

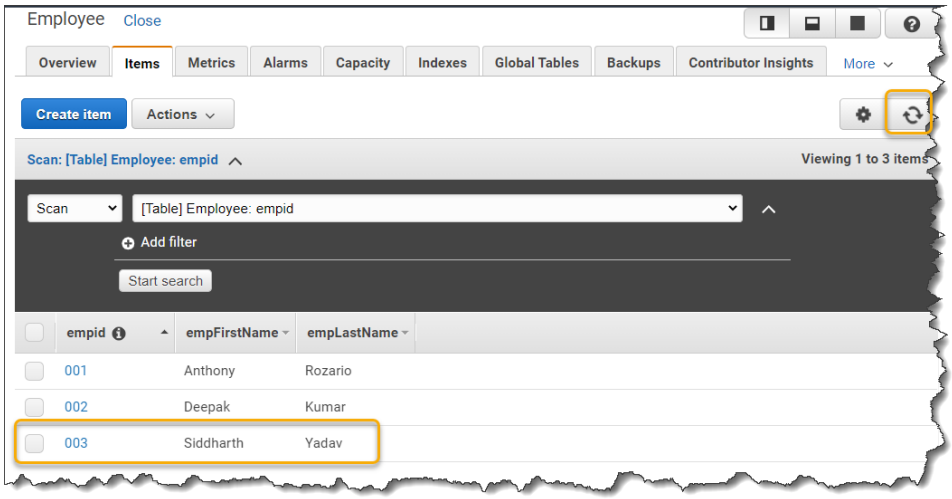
“empLastName”:”Yadav”

}

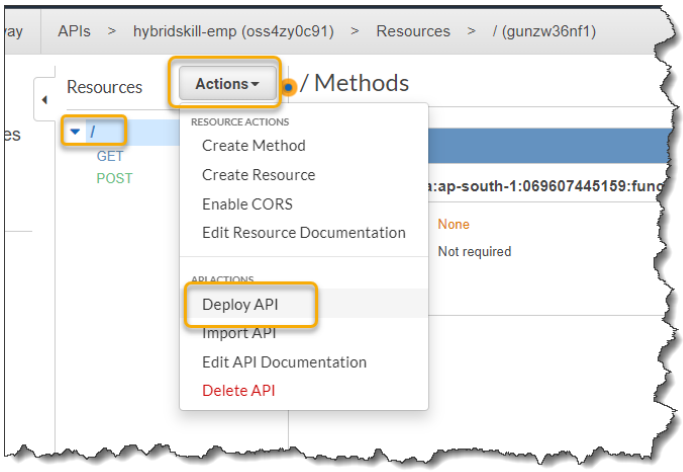
20. You should see a success in the **Response**.



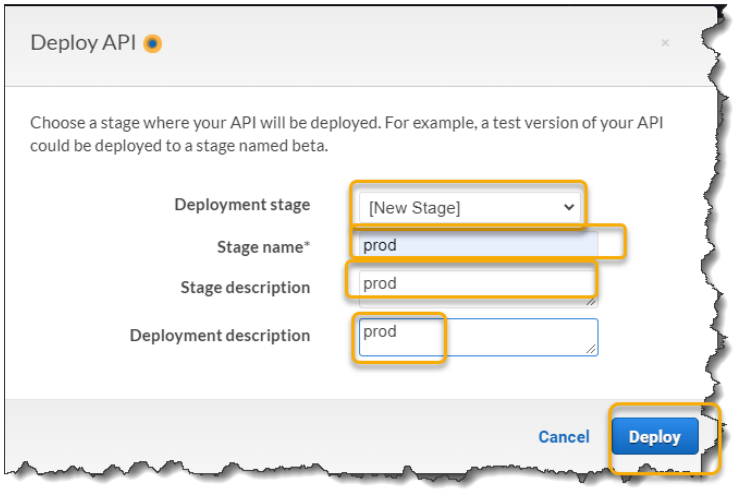
21. Check in DynamoDB for the new item.



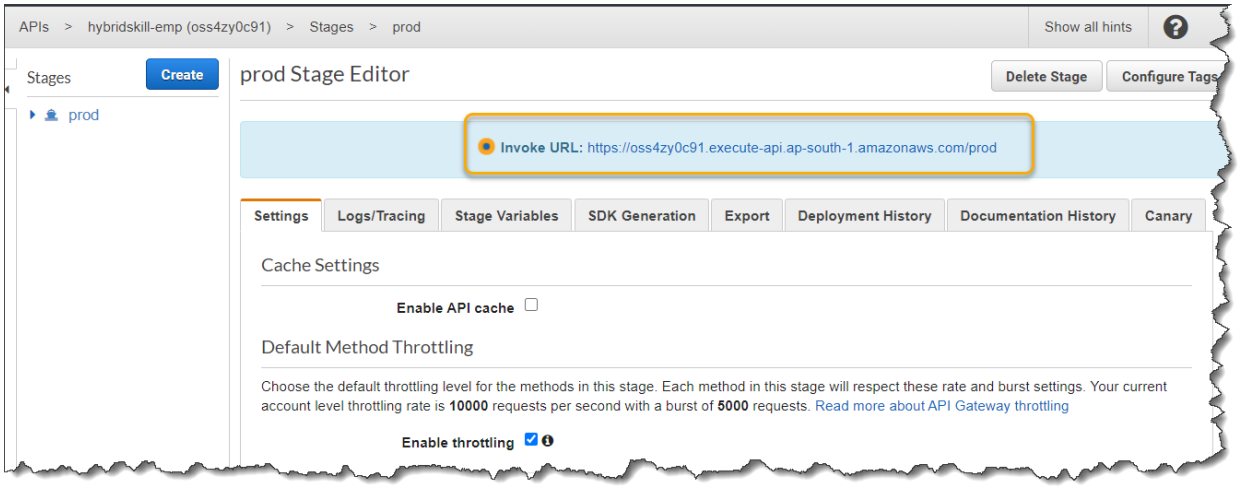
22. Lets Deploy the API next. Click **Deploy API**



23. Select **New stage** and enter the **stage name** as **prod**, enter a **description** and click **Deploy**



24. Copy the **Invoke URL** we will use this next.



Task 5: Set up a Static S3 website

1. We will now setup our frontend on a static S3 website. Open script.js*(from the zip you downloaded earlier)* using an Editor of your choice.

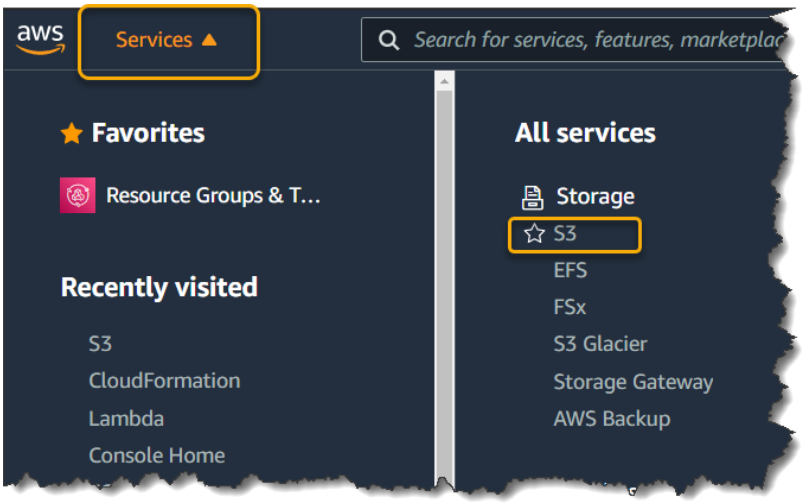
Edit the first line replacing "Your API URI" with the API gateway invoke URL you downloaded earlier.

The line should look like this.

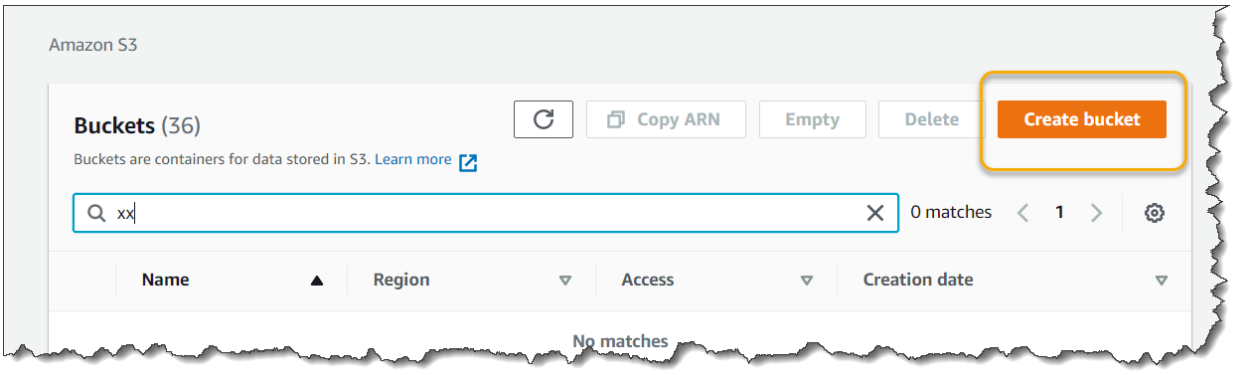
var API_ENDPOINT = https://oss4zy0c91.execute-api.ap-south-1.amazonaws.com/prod

Save the file.

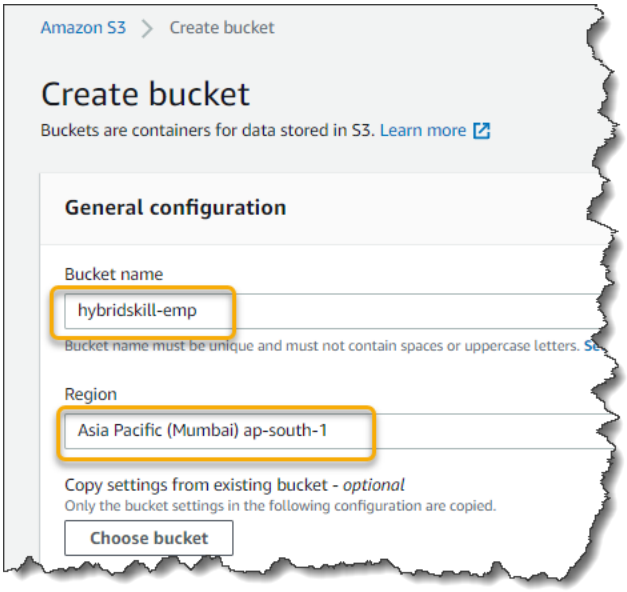
2. Next create an S3 bucket. Click **Services**, click **S3**



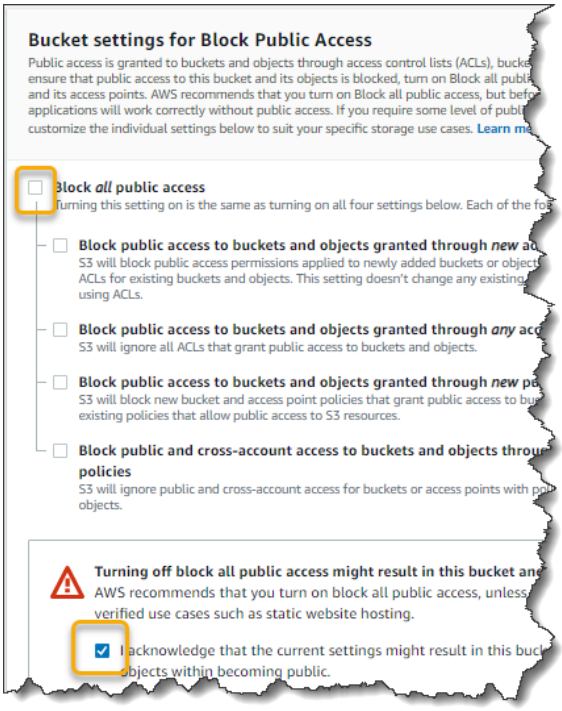
3. Click **Create bucket**.



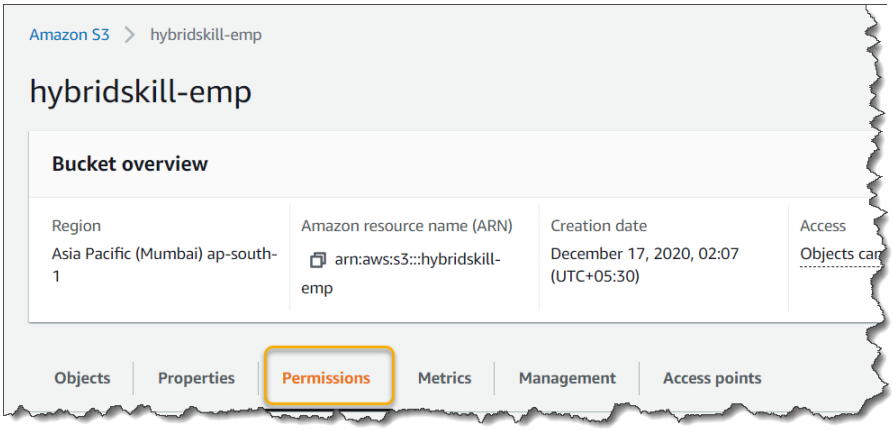
4. Enter the **Bucket name** as **hybridskill-emp**. Select the **Region** as **ap-south-1**



5. **Uncheck** the options for **Block all Public access** and acknowledge.



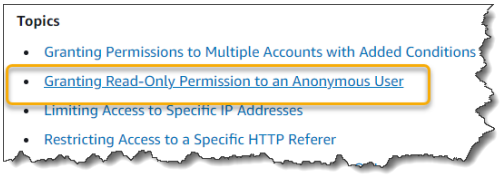
6. Select your bucket and click **Permissions**



7. Under **Bucket policy** click **Edit**

8. Click **Policy Examples**

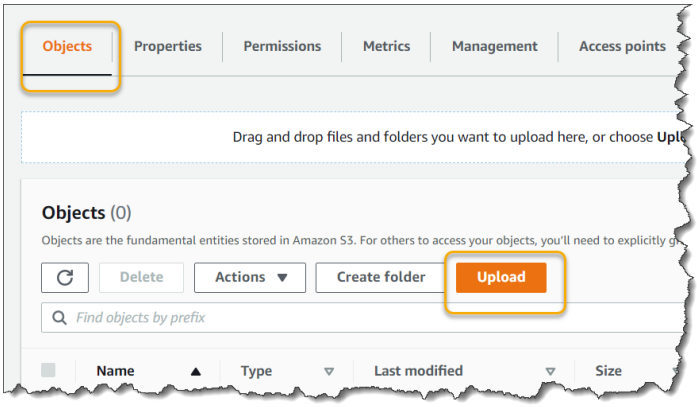
9. Select **Granting Read-only Permission to an Anonymous User**



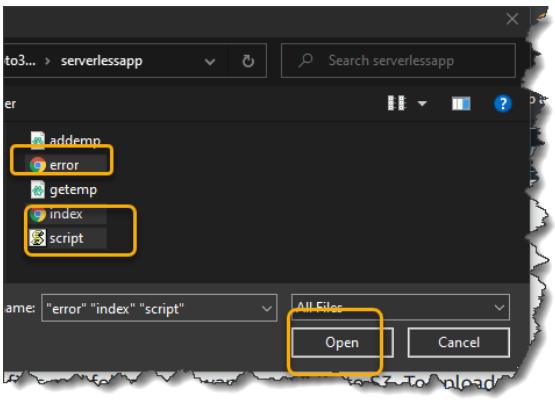
10. Copy the code and paste into your bucket policy editor

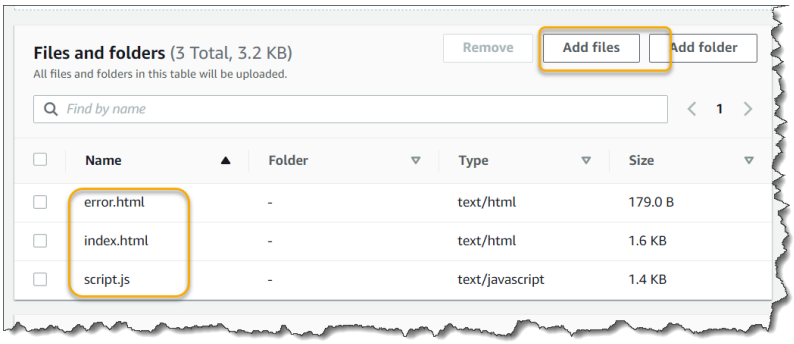
11. Replace *DOC-EXAMPLE-BUCKET* with your bucketname as follows. Click **Save Changes**

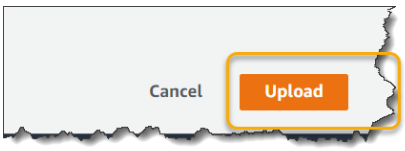
12. Next click **Objects** and click **Upload**



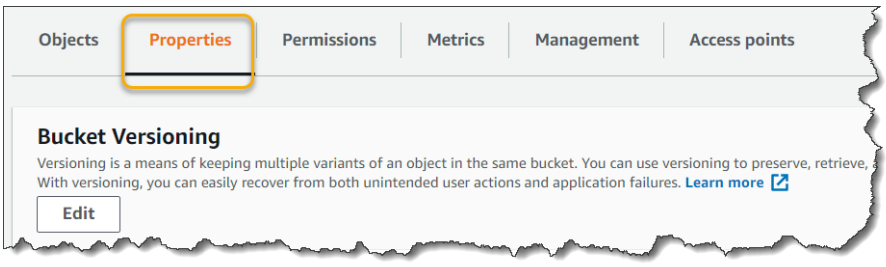
13. From the zipped folder you downloaded earlier, select **index.html, error.html** and **script.js** and **Upload** them into your bucket





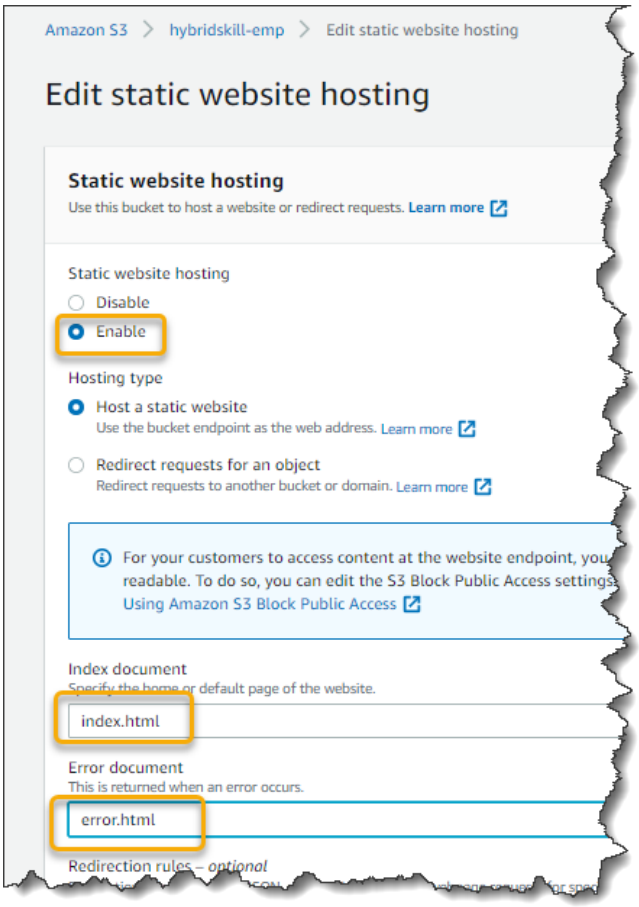


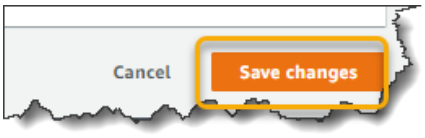
14. Click **Properties**.



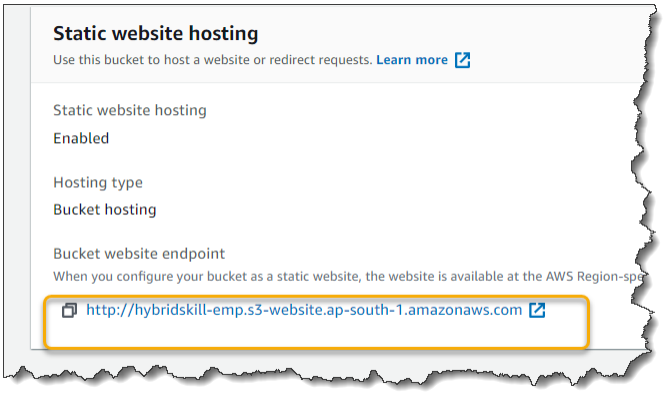
15. Scroll all the way down to **Static website hosting** and click **Edit**

16. Click **Enable** and enter **index.html** as the **Index document** and **Error.html** as the **Error document.** Click **Save changes**

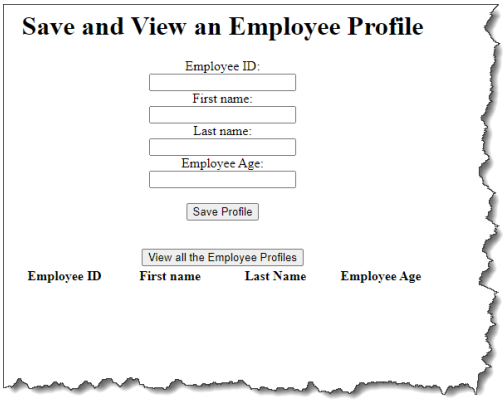




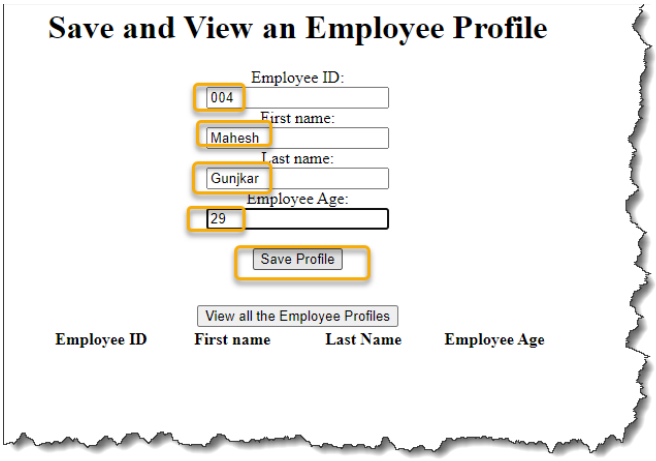
17. Find and view the bucket website endpoint on a browser.



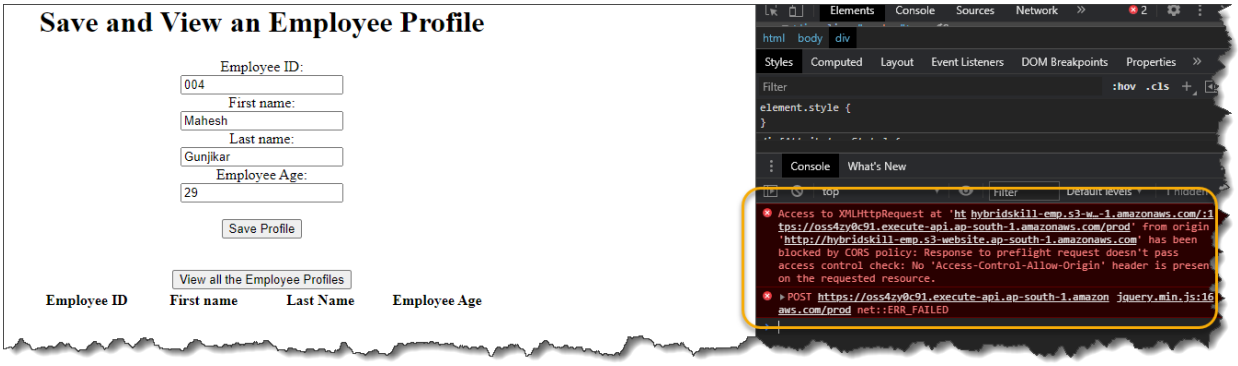
18. Your index page should be visible on a browser



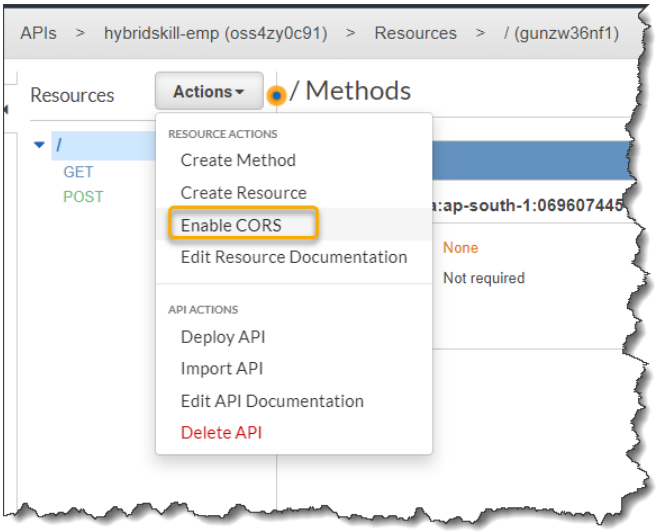
19. Provide **Employee details** and click **Save Profile**



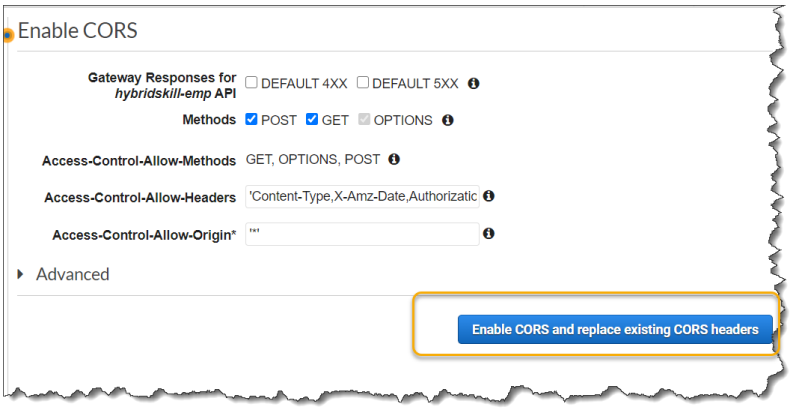
20. Inspect Element and you should see a CORS error



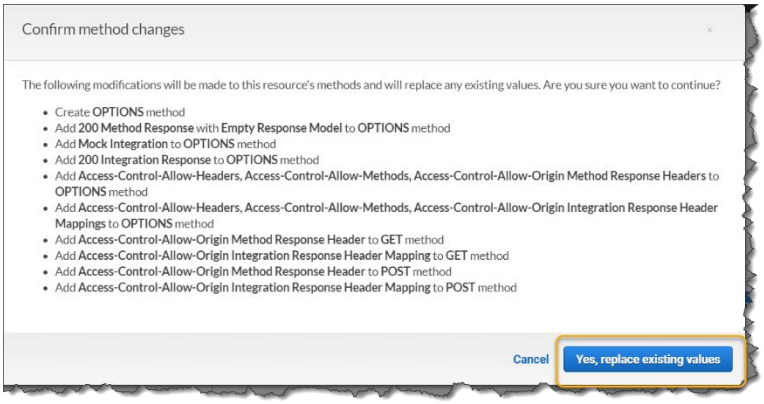
21. Head back to **API gateway** and select **Actions** and **Enable CORS**



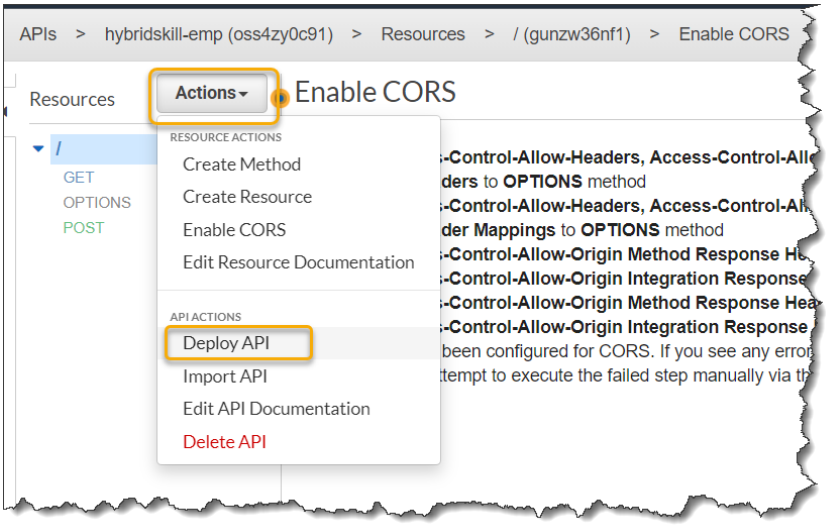
22. Click **Enable CORS and replace existing headers**



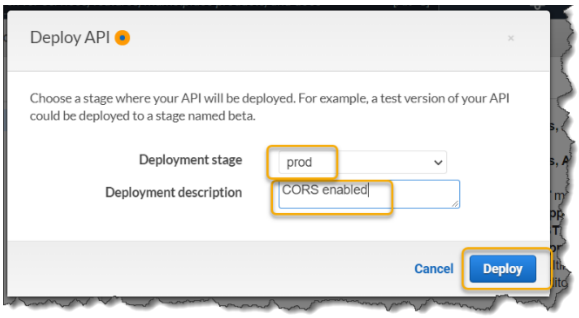
23. Click **Yes, replace existing values**



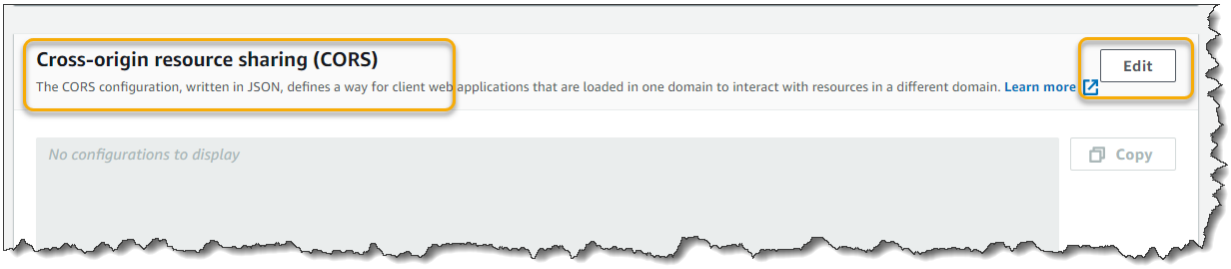
24. Select **Actions** and then **Deploy API**



25. Select the **prod** stage and enter a **Description.** Click **Deploy**

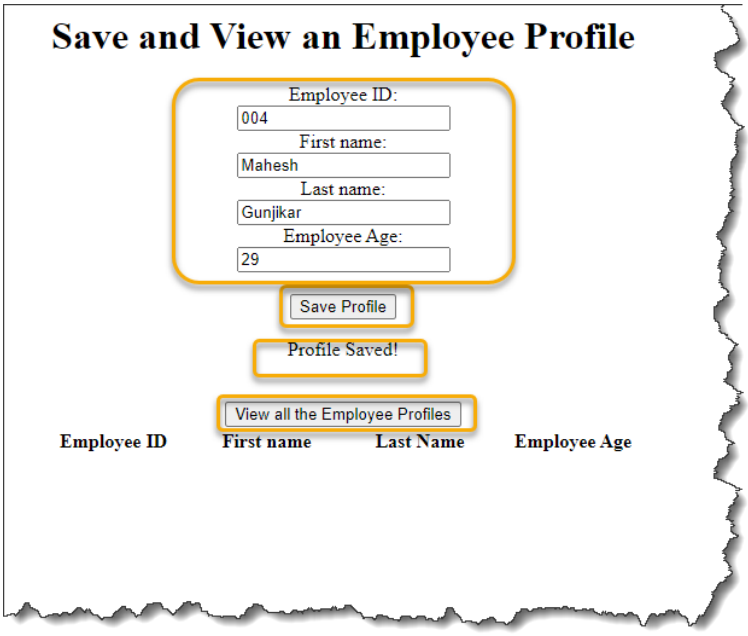


26. You will also have to enable CORS on your bucket

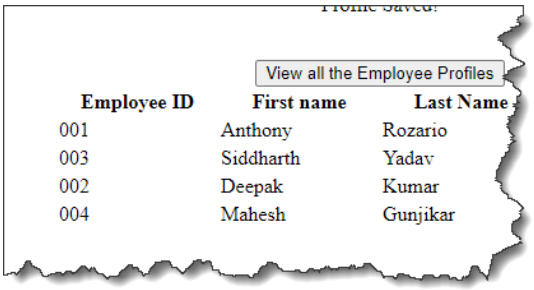


27. Enter the following configuration and click **Save changes**

28. Retry entering the employee information. This time it should be successful.



29. You should see your employee information.



Important: Cleanup of all Resources

Next let's follow this checklist make sure all resources are cleaned up. to prevent billing to your account.

- **Lambda functions**
- **API gateway**
- **DynamoDB table**
- **S3 bucket**