

Contents

Lab 9: Serverless Application	2
Task Breakdown:	2
Task 1: Create a DynamoDB table	2
Task 2: Create an IAM role	4
Task 3: Create Lambda functions	6
Task 4: Create an API gateway	15
Task 5: Set up a Static S3 website.....	25
Important: Cleanup of all Resources	36

Lab 9: Serverless Application

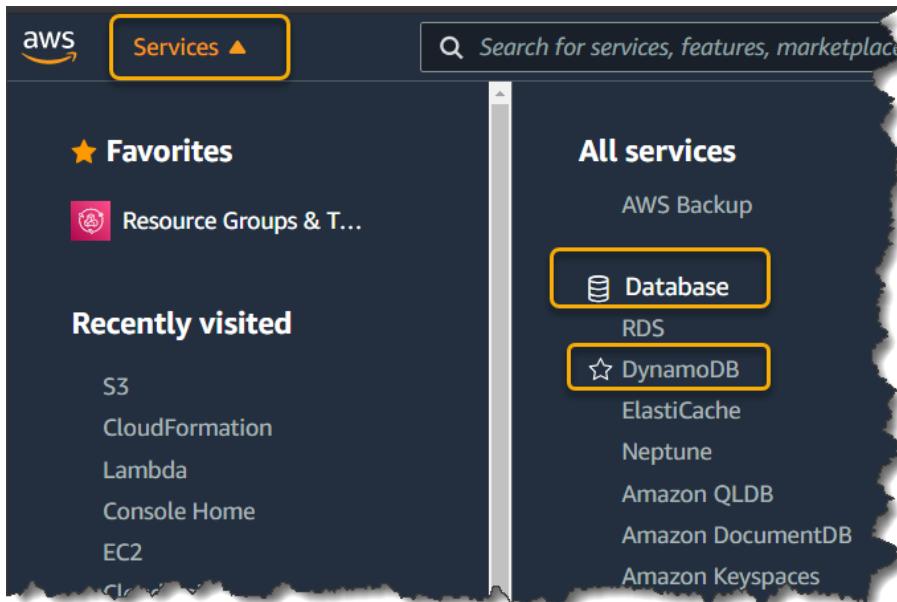
In this lab we are going to create a serverless application using S3, API Gateway, Lambda and DynamoDB

Task Breakdown:

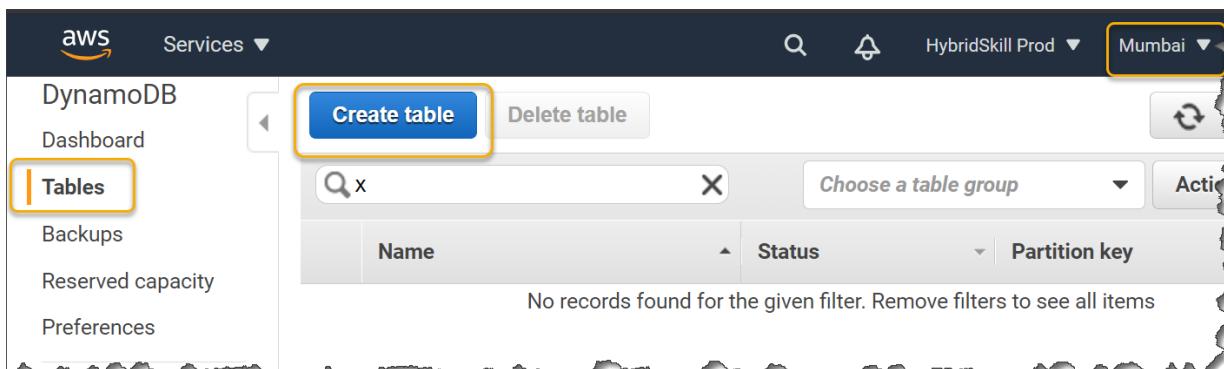
- Create a DynamoDB table
- Create an IAM role
- Create Lambda function
- Create an API gateway
- Set up a Static S3 Website

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.

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*	<input type="text" value="Employee"/>	<small>i</small>
Primary key*	Partition key	
	<input type="text" value="empid"/>	<small>String</small> <small>i</small>
<input type="checkbox"/> Add sort key		

Table settings

Default settings provide the fastest way to get started with your table. You can modify these after it's created.

Use default settings

Click **Create**.



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

Enter the information as follows.

empid : 001

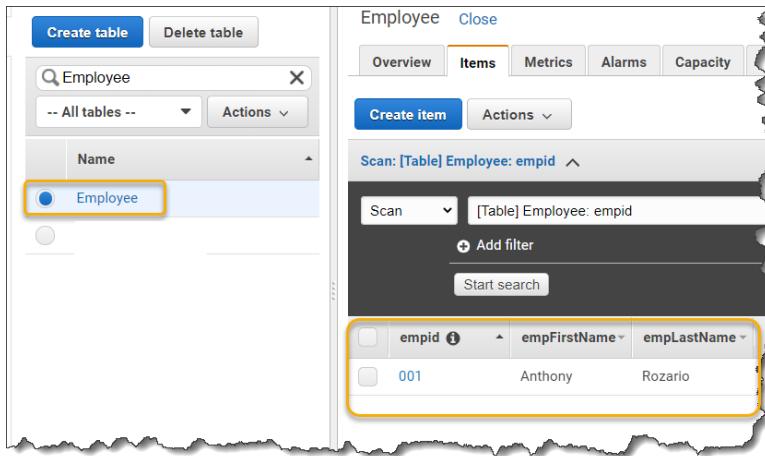
empFirstName: *Your firstname*

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

Create item

Tree	<input type="button" value="+"/>	<input type="button" value=""/>	<input type="button" value=""/>
Item {3}			
<ul style="list-style-type: none"> + empLastName String Rozario + empFirstName String Anthony + empid String 001 			
<input type="radio"/> Append <input checked="" type="radio"/> Insert			
<ul style="list-style-type: none"> + <input type="button" value="String"/> + <input type="button" value="Binary"/> + <input type="button" value="Number"/> + <input type="button" value="StringSet"/> + <input type="button" value="NumberSet"/> + <input type="button" value="BinarySet"/> + <input type="button" value="Map"/> + <input type="button" value="List"/> 			
<input type="button" value="Cancel"/> <input type="button" value="Save"/>			

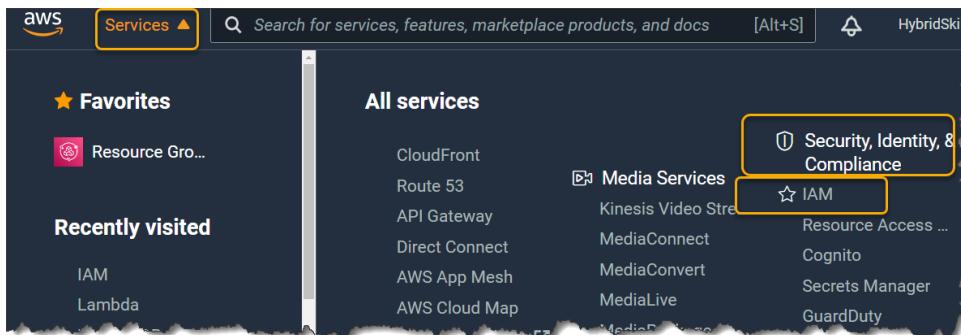
5. Your DynamoDB table should look like this.



The screenshot shows the AWS DynamoDB console. On the left, there's a sidebar with 'Create table' and 'Delete table' buttons. Below that is a search bar with 'Employee' typed in. A dropdown menu shows 'All tables' and an 'Actions' dropdown. Under 'Actions', 'Employee' is selected. The main area is titled 'Employee' with tabs for 'Overview', 'Items', 'Metrics', 'Alarms', and 'Capacity'. The 'Items' tab is selected. It shows a 'Scan' interface with a dropdown set to 'Scan' and a filter 'empid'. Below this is a table with columns 'empid', 'empFirstName', and 'empLastName'. One row is highlighted with a yellow box, showing values '001', 'Anthony', and 'Rozario' respectively.

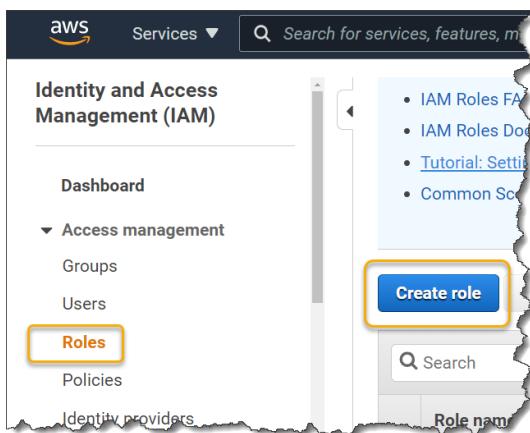
Task 2: Create an IAM role

- 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**



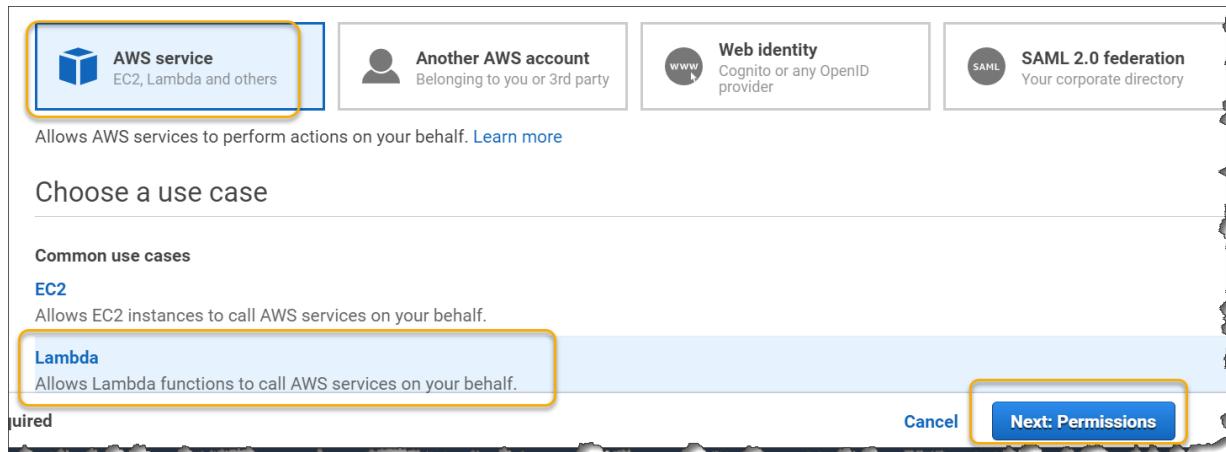
The screenshot shows the AWS services navigation bar. The 'Services' button is highlighted with a yellow box. Below it, the 'All services' section lists various AWS services. To the right, under 'Security, Identity, & Compliance', the 'IAM' service is highlighted with a yellow box. Other services listed include CloudFront, Route 53, API Gateway, Direct Connect, AWS App Mesh, AWS Cloud Map, Media Services (with Kinesis Video Stream, MediaConnect, MediaConvert, MediaLive), Resource Access Management, Cognito, Secrets Manager, and GuardDuty.

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

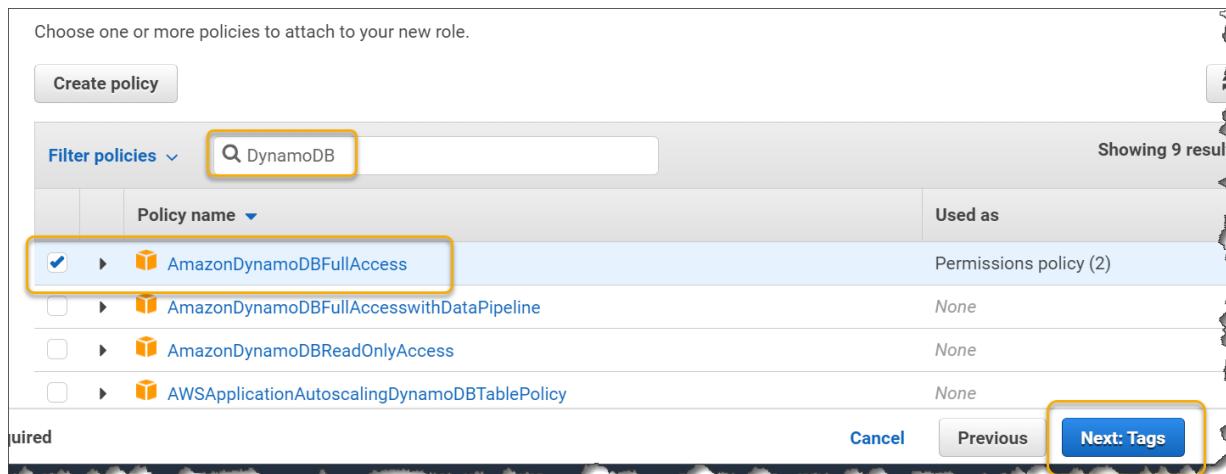


The screenshot shows the AWS IAM 'Identity and Access Management (IAM)' page. The left sidebar has 'Identity and Access Management (IAM)' at the top, followed by 'Dashboard', 'Access management' (with 'Groups' and 'Users' sub-options), 'Roles' (which is highlighted with a yellow box), 'Policies', and 'Identity providers'. The main content area shows a list of items: 'IAM Roles FA...', 'IAM Roles Doc...', 'Tutorial: Setti...', and 'Common Sc...'. Below this is a large blue 'Create role' button, which is also highlighted with a yellow box. There is also a 'Search' bar and a 'Role name' input field.

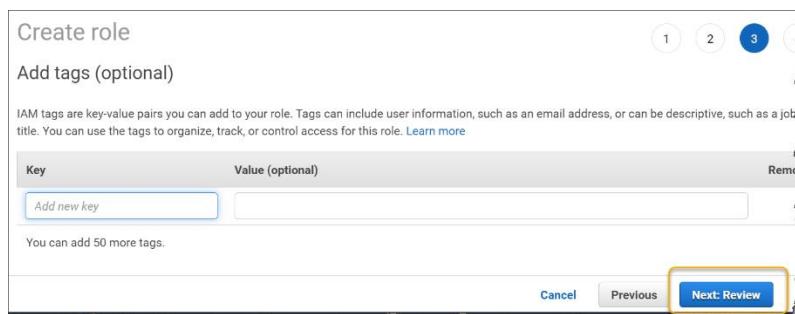
3. Select AWS service, then click on Lambda and then Next:Permissions



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**.

Create role

Review

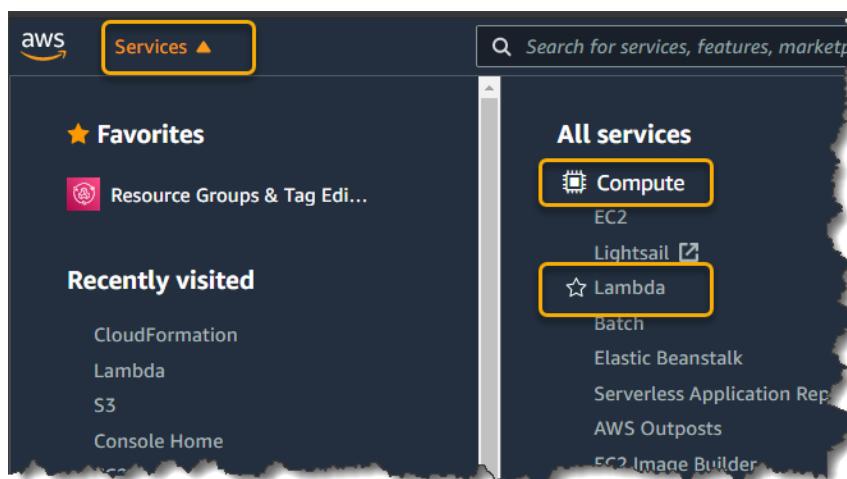
Provide the required information below and review this role before you create it.

Role name*	empDynamodDBrole
Use alphanumeric and '+,-,@-' characters. Maximum 64 characters.	
Role description	Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+,-,@-' characters.	
Trusted entities	AWS service: lambda.amazonaws.com
Policies	 AmazonDynamoDBFullAccess 

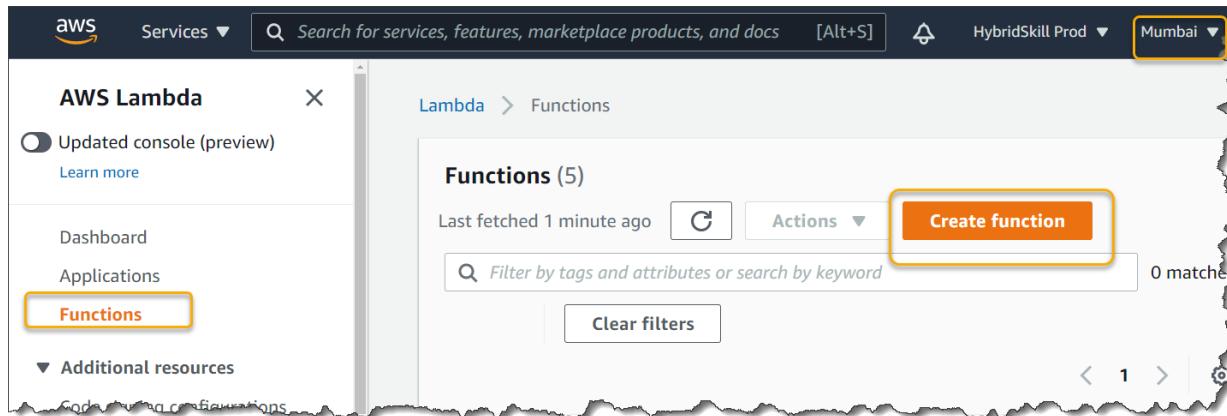
* Required Cancel Previous **Create role**

Task 3: Create Lambda functions

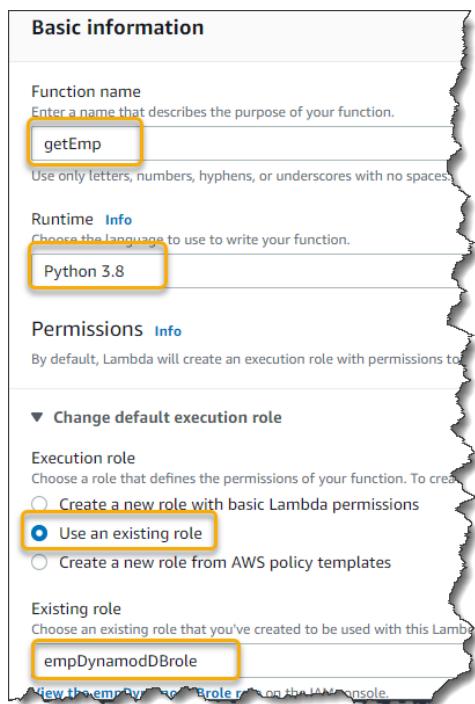
- We are now going to create 2 Lambda functions One for getting data from DynamoDB(gettemp.py) and the other for putting data(puttemp.py) into DynamoDB.
Download and unzip the code from the following location
<https://hybridskill-training.s3.amazonaws.com/genpact/serverlessapp.zip>
- Click on **Services** 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.



Basic information

Function name
Enter a name that describes the purpose of your function.
getEmp

Runtime [Info](#)
choose the language to use to write your function.
Python 3.8

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a new role or edit an existing one, click Create a new role or Use an existing role.

Create a new role with basic Lambda permissions

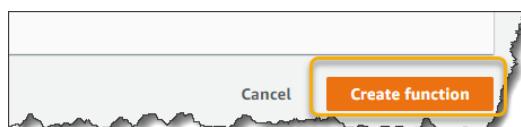
Use an existing role

Create a new role from AWS policy templates

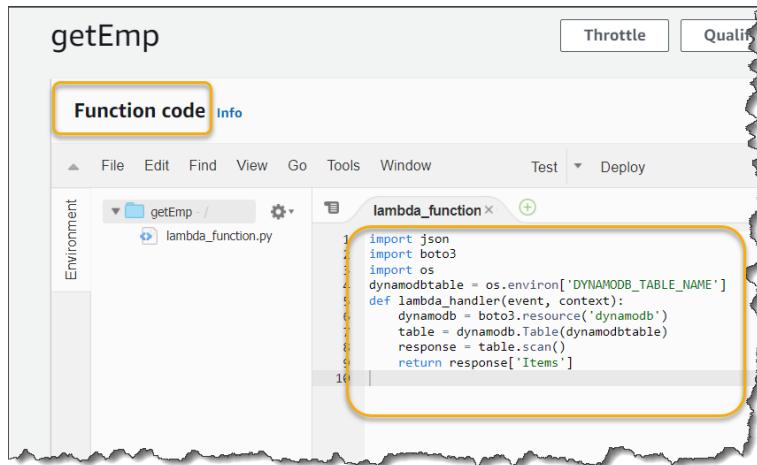
Existing role
Choose an existing role that you've created to be used with this Lambda function.

empDynamodbRole

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



```

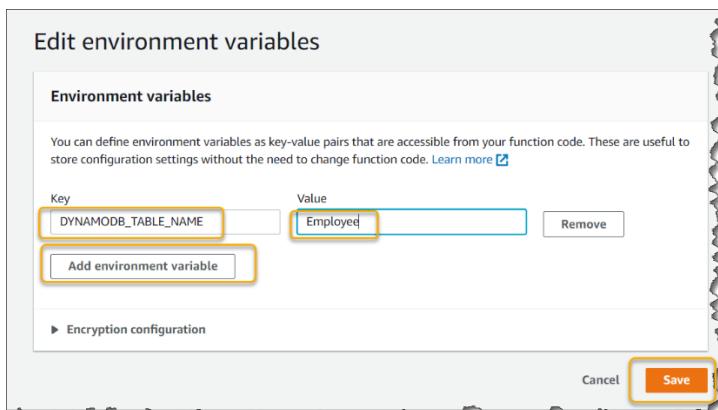
import json
import boto3
import os
dynamodbtable = os.environ['DYNAMODB_TABLE_NAME']
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table(dynamodbtable)
    response = table.scan()
    return response['Items']

```

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

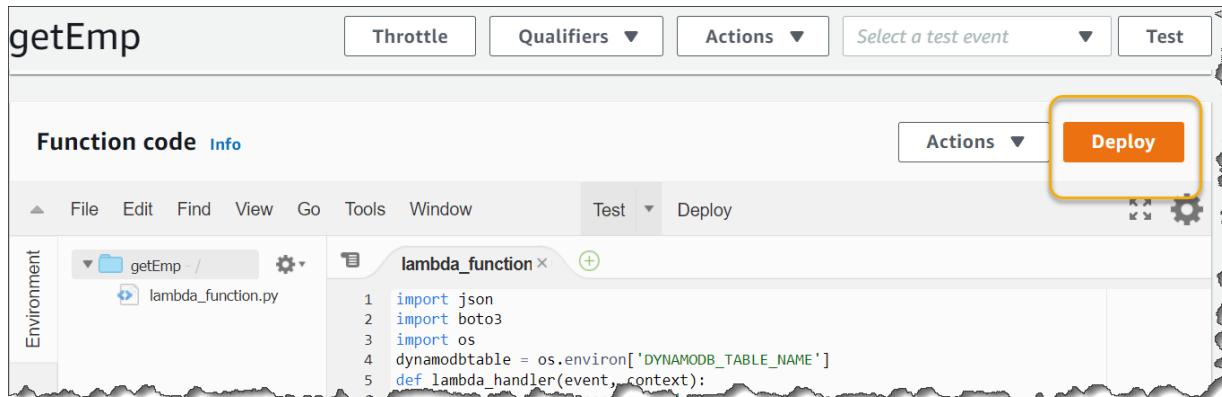


7. Click **Add environment variable** and enter the **Key** as **DYNAMODB_TABLE_NAME** and the **Value** as the name of your DynamoDB table which in our case is **Employee**. Click **Save**

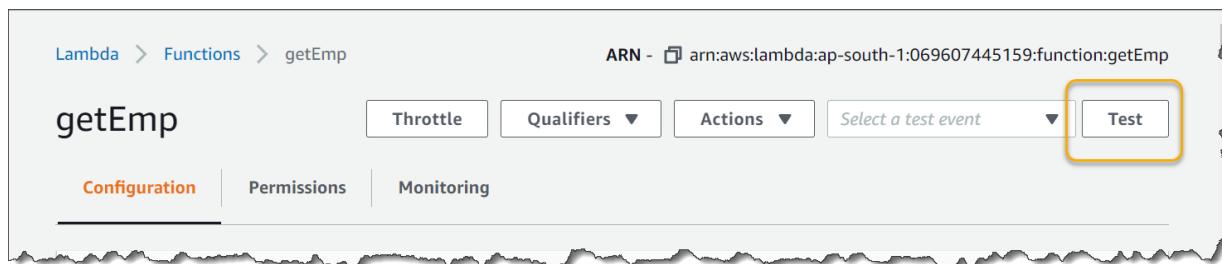


Key	Value
DYNAMODB_TABLE_NAME	Employee
Add environment variable	
Encryption configuration	
Cancel Save	

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**.

Configure test event

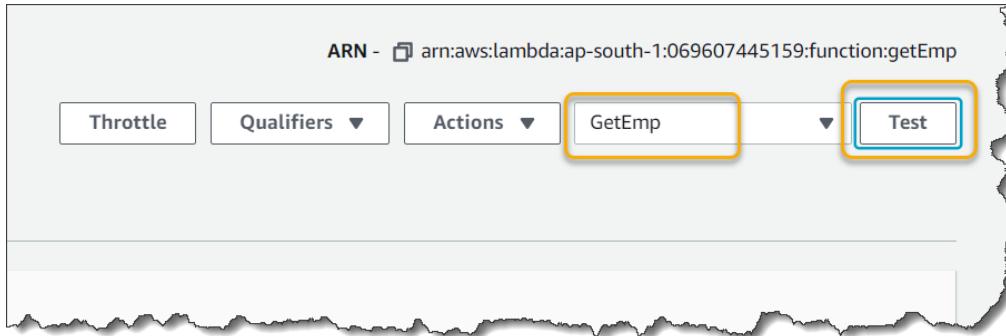
A function can have up to 10 test events. The events and test your function with the same events.

Create new test event
 Edit saved test events

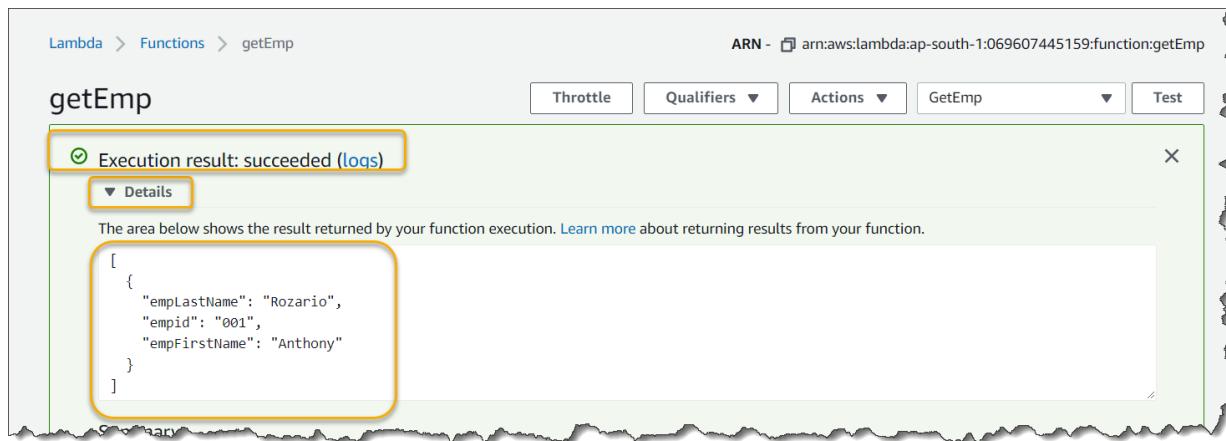
Event template

Event name
 {"key1": "value1", "key2": "value2", "key3": "value3"}
{"key1": "value1", "key2": "value2", "key3": "value3"}}

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



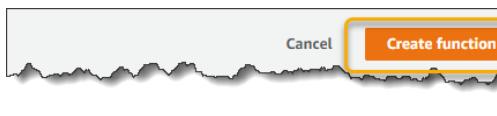
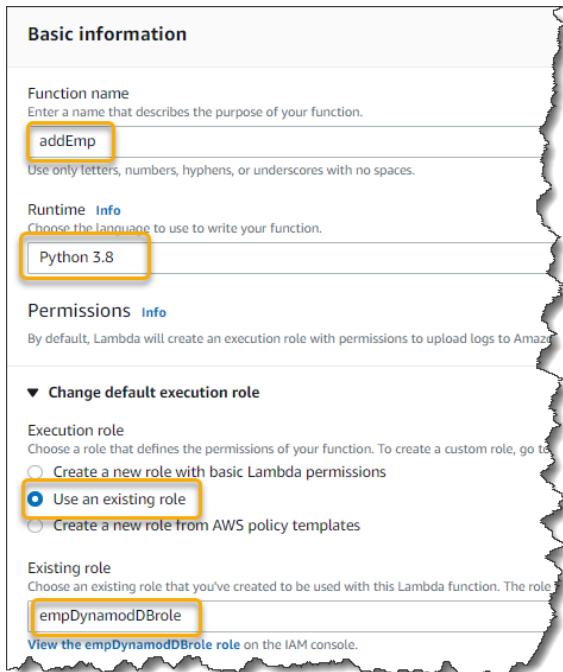
12. Expand **Details** to view the execution results.



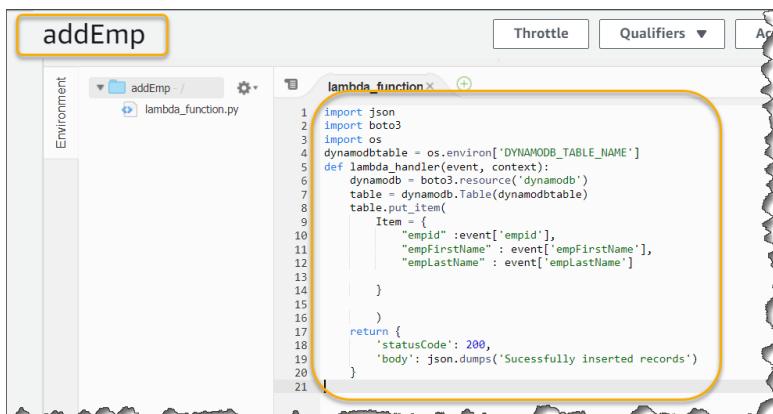
13. Let's Create the **addEmp** Lambda function next. Click **Functions** and **Create function**



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



```

import json
import boto3
import os
dynamodbtable = os.environ['DYNAMODB_TABLE_NAME']
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table(dynamodbtable)
    table.put_item(
        Item = {
            'empid' : event['empid'],
            'empFirstName' : event['empFirstName'],
            'empLastName' : event['empLastName']
        }
    )
    return {
        'statusCode': 200,
        'body': json.dumps('Sucessfully inserted records')
    }

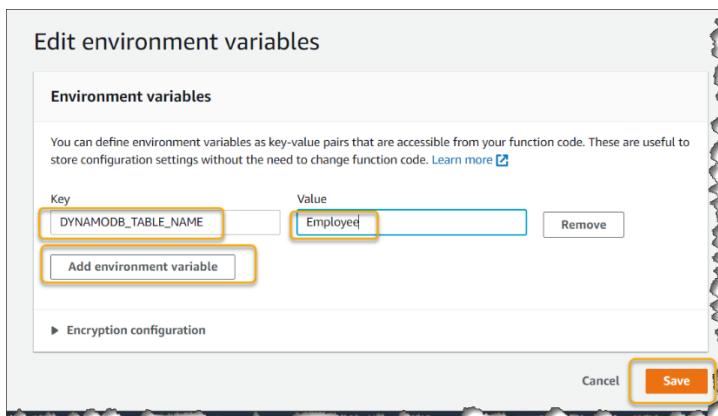
```

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



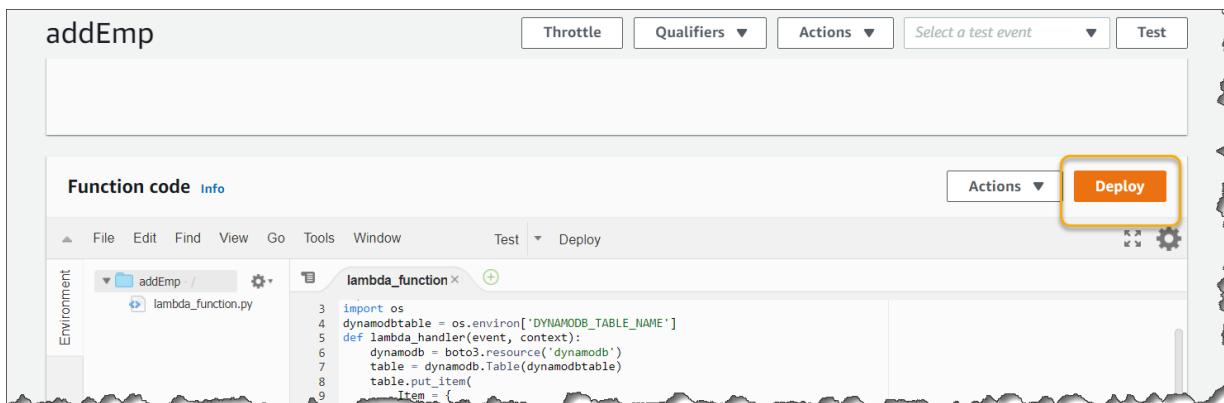
Key	Value
No environment variables	
No environment variables associated with this function	
Edit	

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**



Key	Value
DYNAMODB_TABLE_NAME	Employee
Add environment variable	
Save	

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



```

import os
dynamodbtable = os.environ['DYNAMODB_TABLE_NAME']
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table(dynamodbtable)
    table.put_item(
        Item={}
    )

```

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**.

Configure test event

A function can have up to 10 test events. The events are persisted and test your function with the same events.

Create new test event
 Edit saved test events

Event template
 hello-world

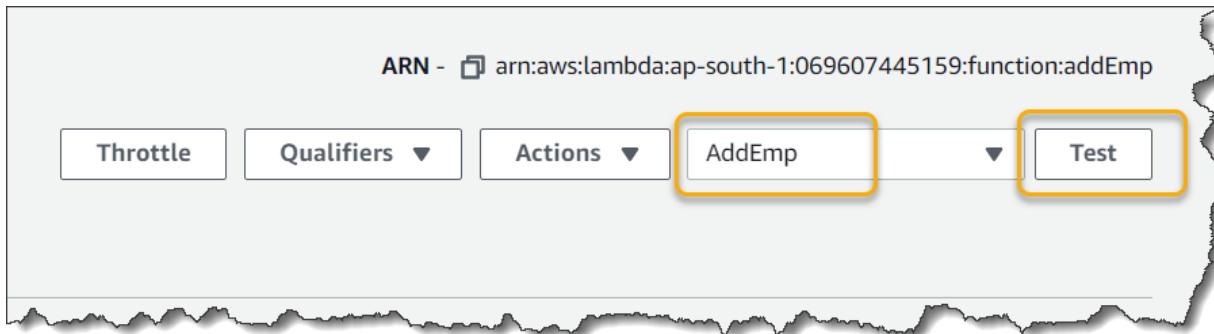
Event name
AddEmp

```

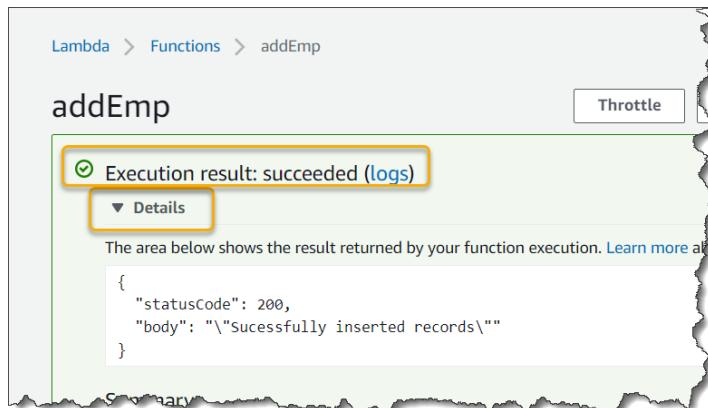
1 {
2     "empid": "002",
3     "empFirstName": "Deepak",
4     "empLastName": "Kumar"
5 }
```

Cancel Format JSON **Create**

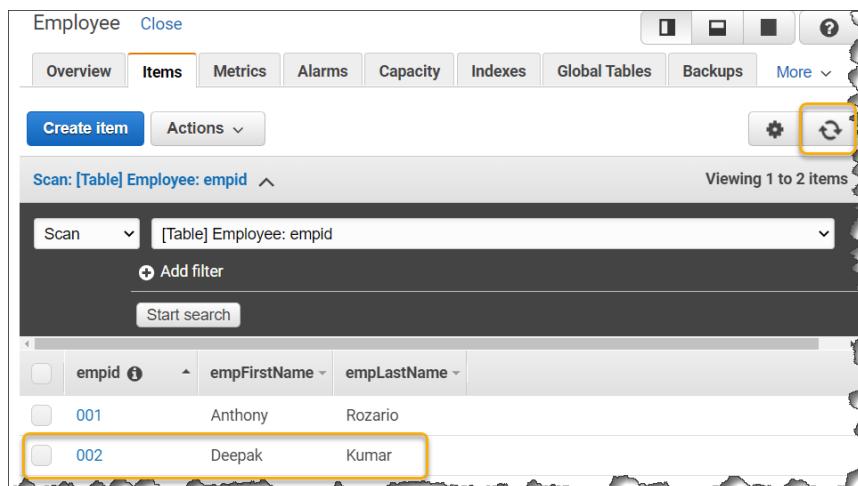
21. With the **AddEmp** event selected click **Test**.



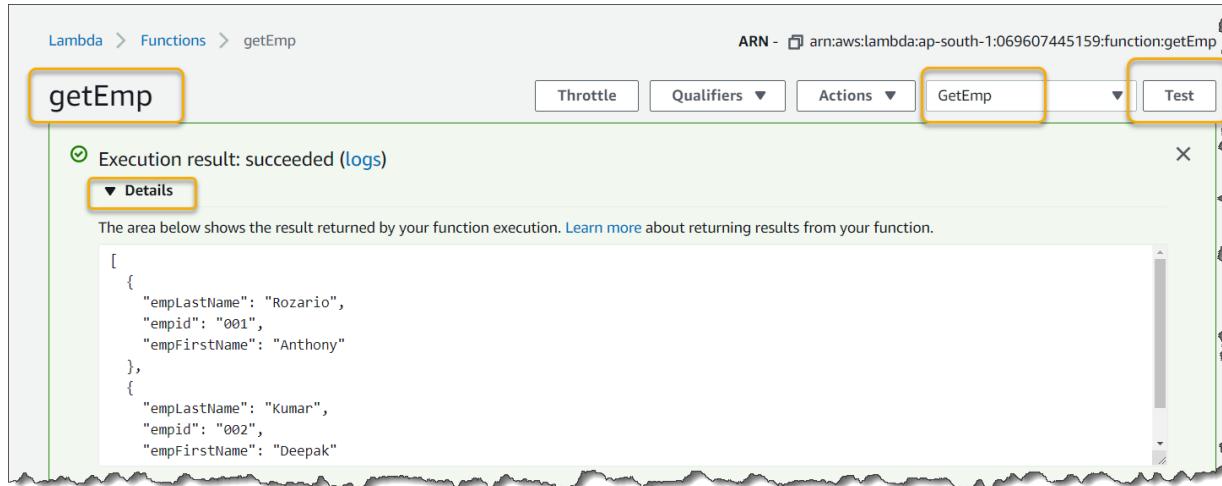
22. Expand **Details** to view the execution results.



23. Check your DynamoDB table for the new entry.

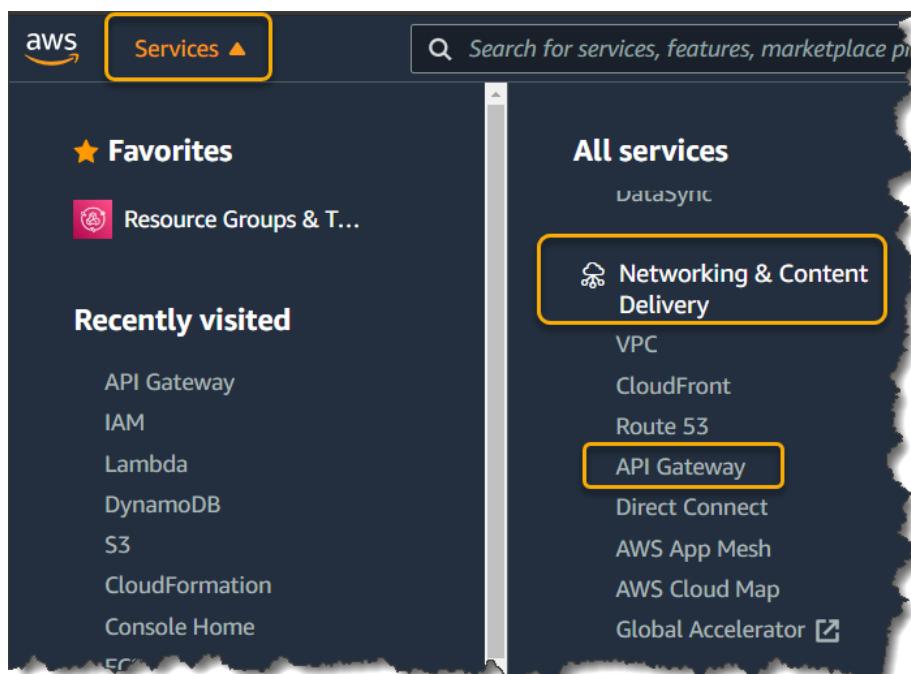


24. You can alternatively run your **getEmp** Lambda function and test if it returns the new records.

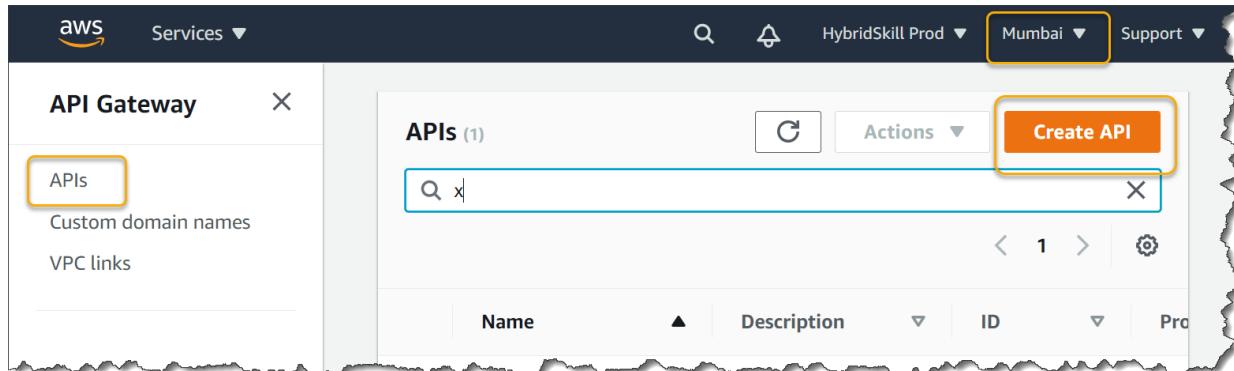


Task 4: Create an API gateway

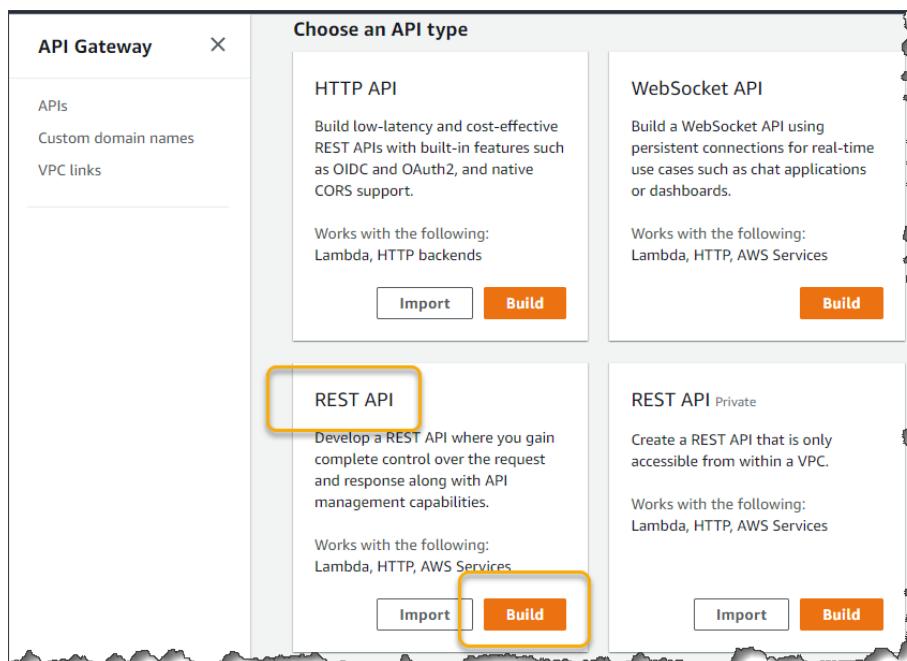
1. We will next create an API to expose our Lambda functions. Click **Services** and under **Networking & Content Delivery** click **API Gateway**



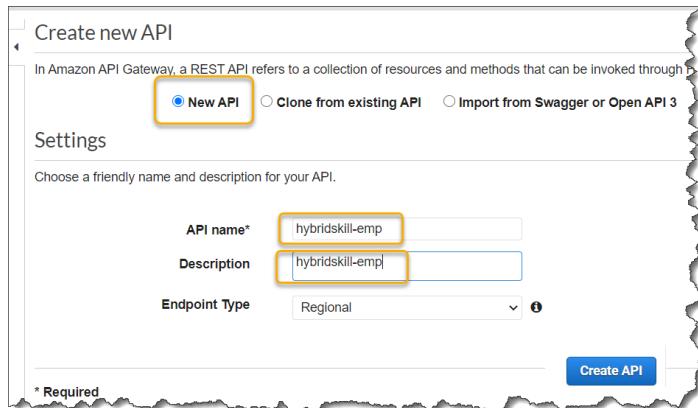
2. Make sure your **Region** is selected, click on **APIs** on the left pane and click on **Create API**. We have selected **Mumbai** as an example.



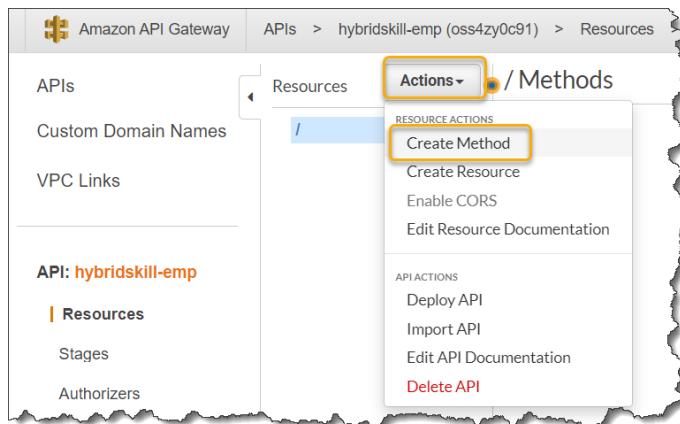
Choose the **API type** as **REST API** and click **Build**



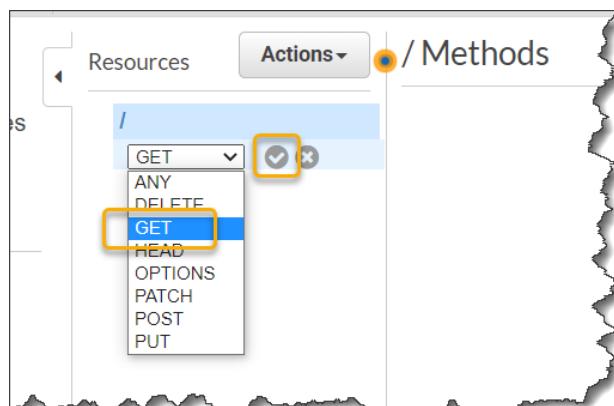
3. Select new API, enter **hybridskill-emp** as the name of your **API**, enter a description and click **Create API**



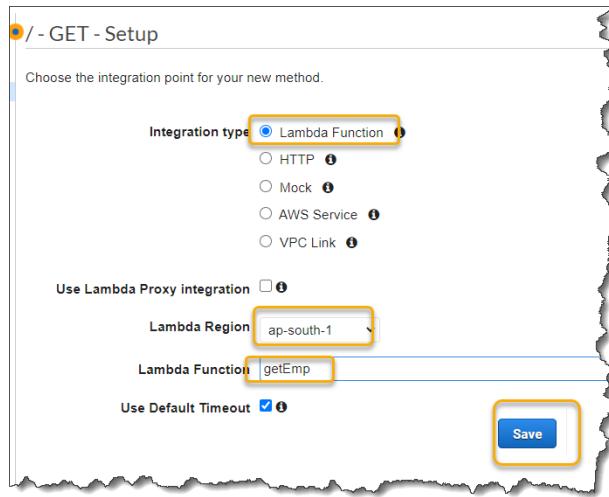
4. Select **Actions** and then **Create Method**



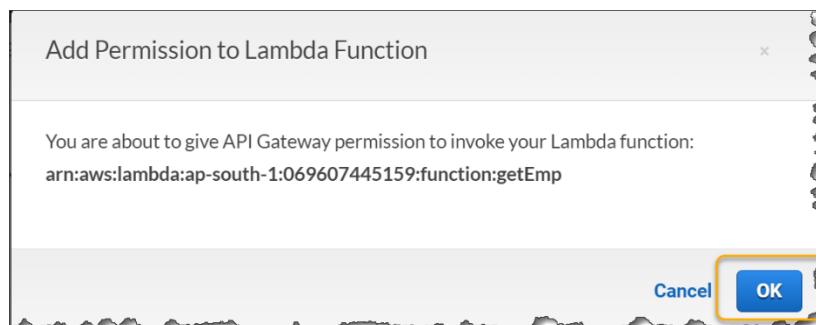
5. Select **GET** as the method and **Save**



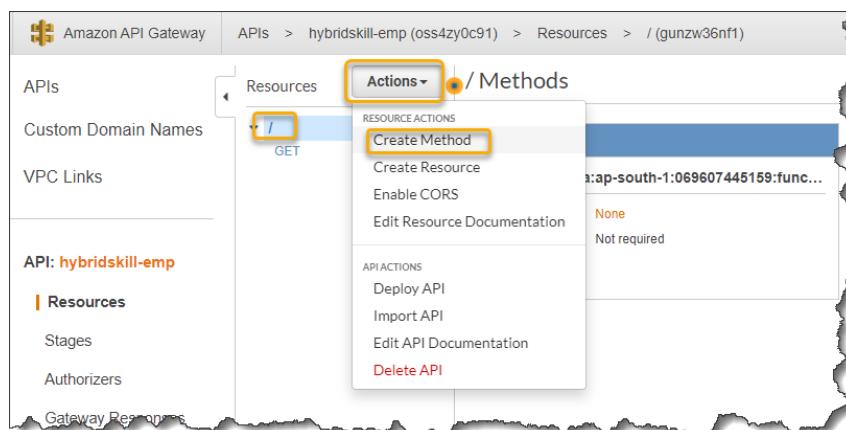
6. Select the **Integration type** as **Lambda Function**. Select **ap-south-1** as the **Lambda Region** select your **getEmp** Lambda function. Click **Save**



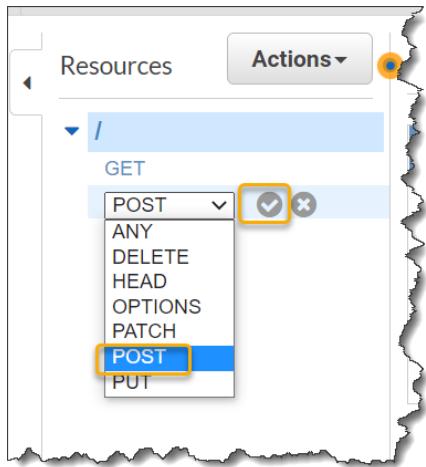
7. Click **OK** to give API gateway permissions to invoke your Lambda function.



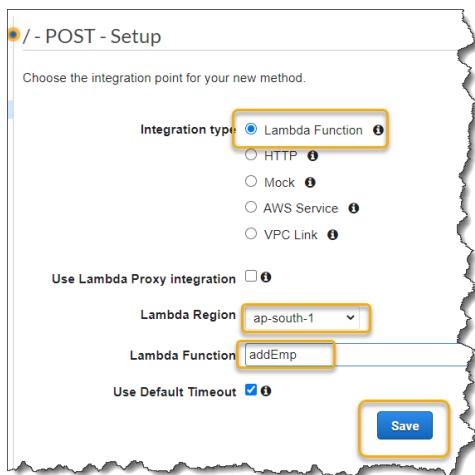
8. Lets create our POST request next. Select your API gateway and the click / and then click **Actions** and then **Create Method**.



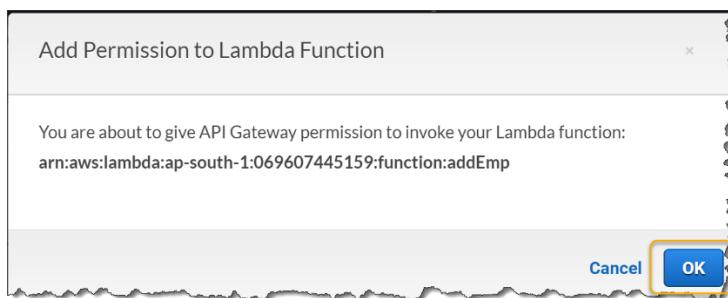
9. This time select **POST** as the method and **Save**



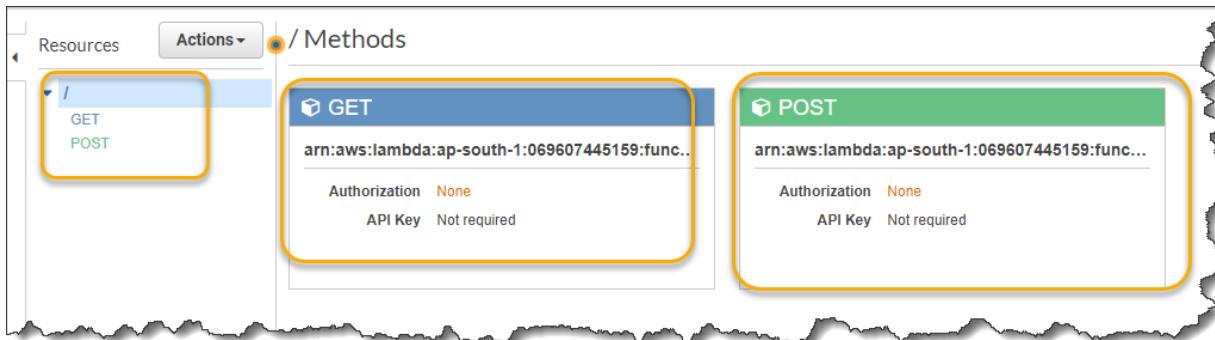
10. Select the **Integration type** as **Lambda Function**. Select **ap-south-1** as the **Lambda Region** select your **addEmp** Lambda function. Click **Save**



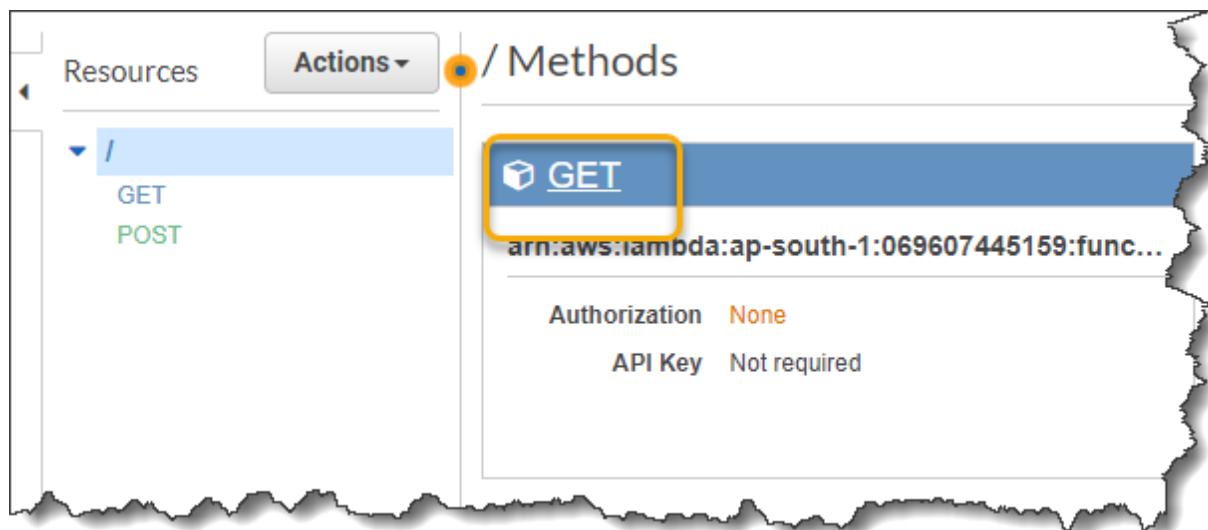
11. Click **OK** to give API gateway permissions to invoke your Lambda function



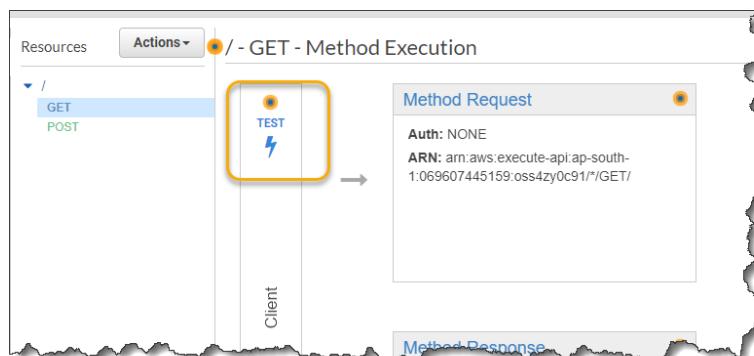
12. You should have 2 APIs setup. GET and POST.



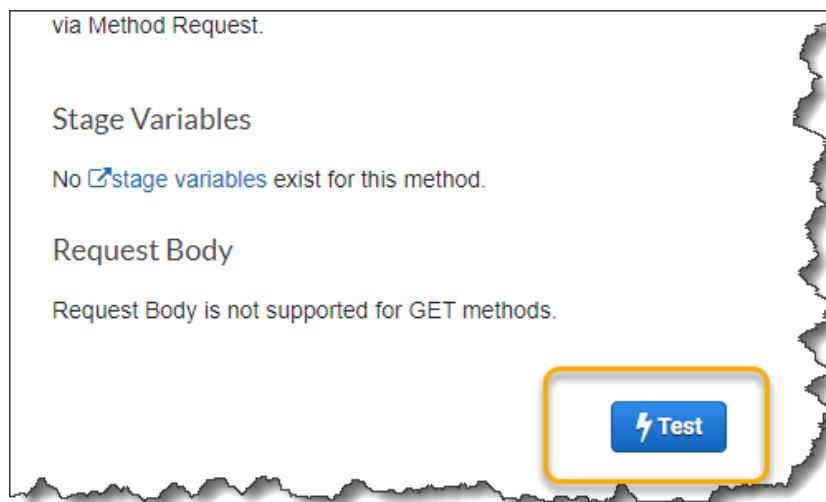
13. Lets test out the GET API first. Click on **GET**



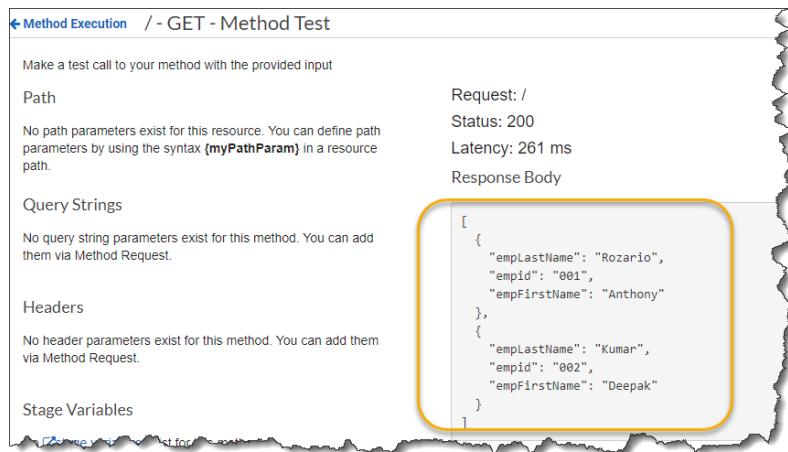
14. Next click on **Test**



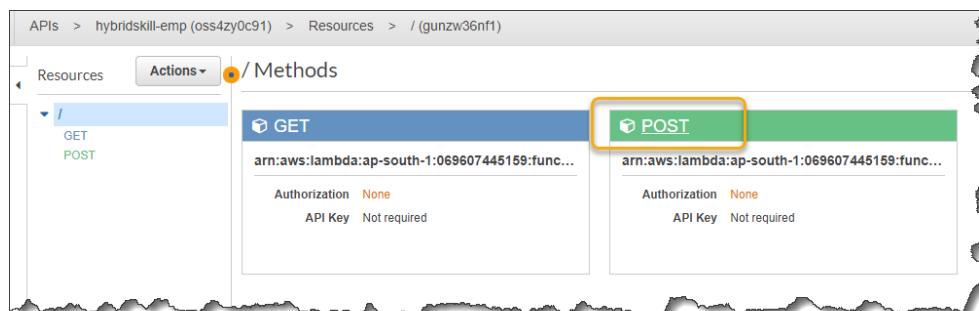
15. Scroll all the way to the bottom and click **Test**



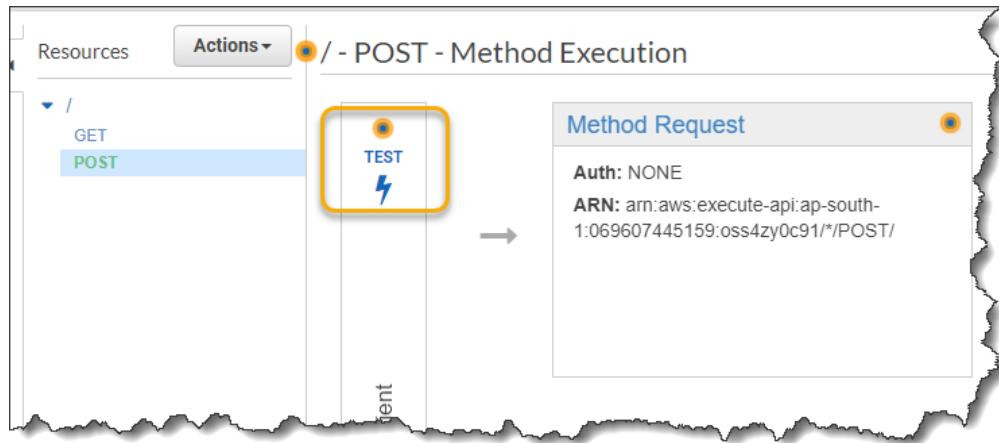
16. You should see your Employee data in the Response.



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

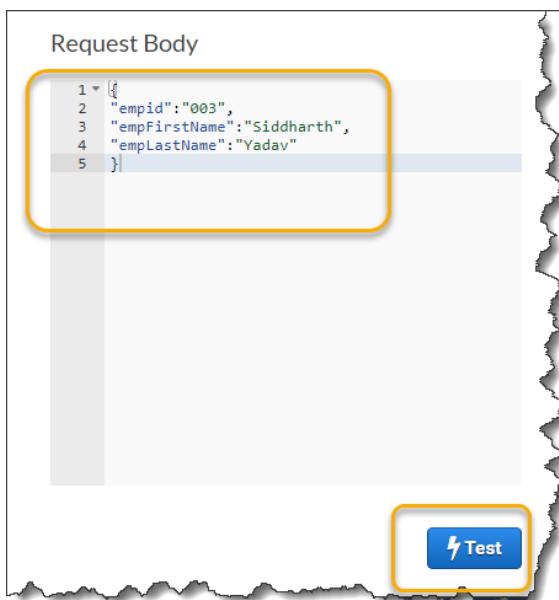


18. Click on **TEST**

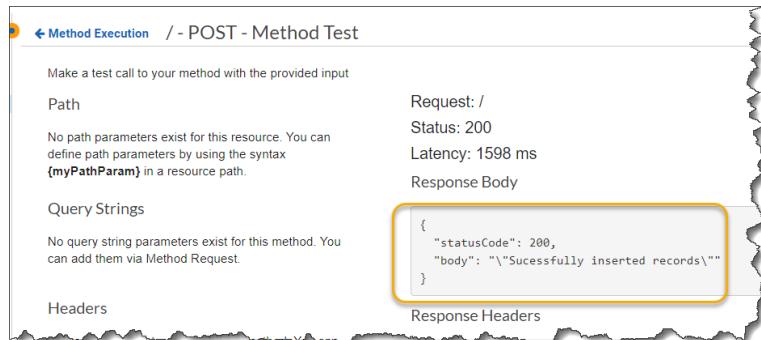


19. Scroll down, in the **Request Body**, enter the values of another employee as follows and click **Test**

```
{
    "empid": "003",
    "empFirstName": "Siddharth",
    "empLastName": "Yadav"
}
```



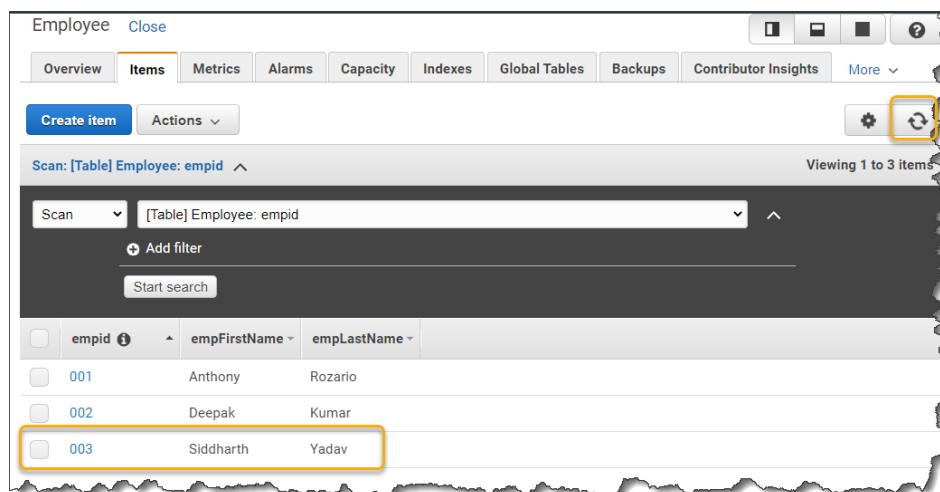
20. You should see a success in the Response.



The screenshot shows the Method Execution tool interface for a POST method. The response body is highlighted with a yellow box and contains the following JSON:

```
{
  "statusCode": 200,
  "body": "\"Successfully inserted records\""
}
```

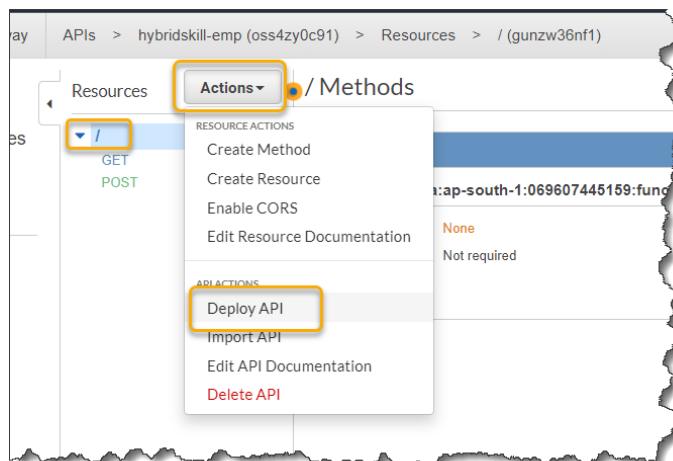
21. Check in DynamoDB for the new item.



The screenshot shows the AWS DynamoDB console for the Employee table. The table has three items:

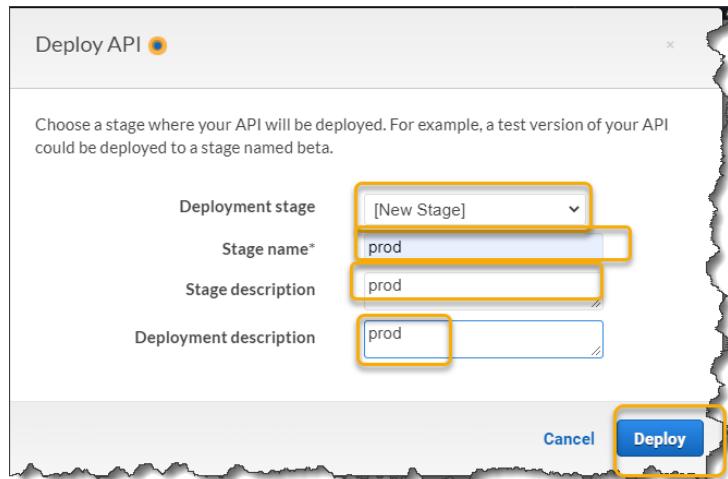
empid	empFirstName	empLastName
001	Anthony	Rozario
002	Deepak	Kumar
003	Siddharth	Yadav

22. Lets Deploy the API next. Click Deploy API

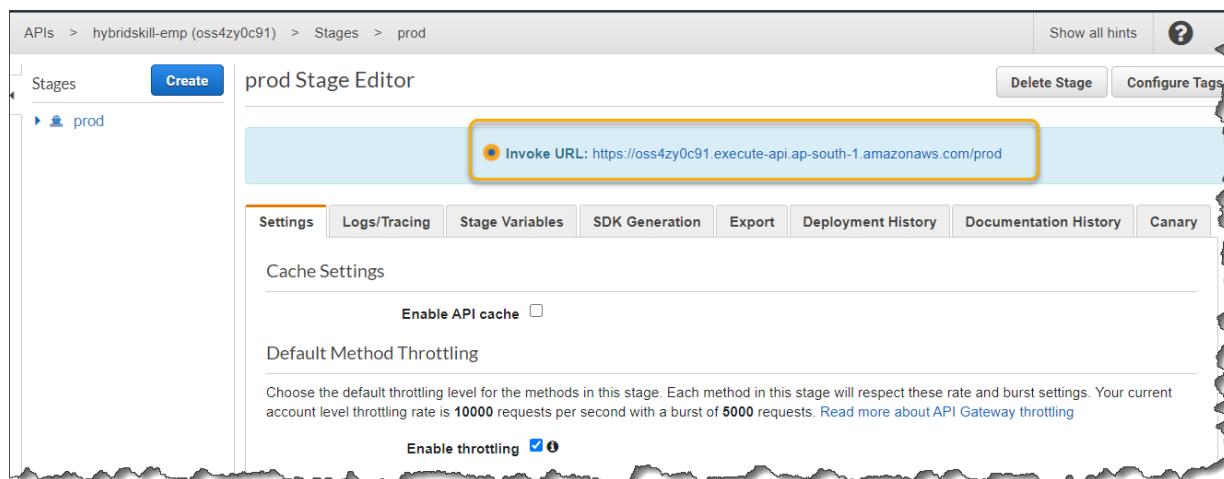


The screenshot shows the AWS API Gateway Resources page for a specific resource. The 'Actions' dropdown menu is open, and the 'Deploy API' option is highlighted with a yellow box.

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

- 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.

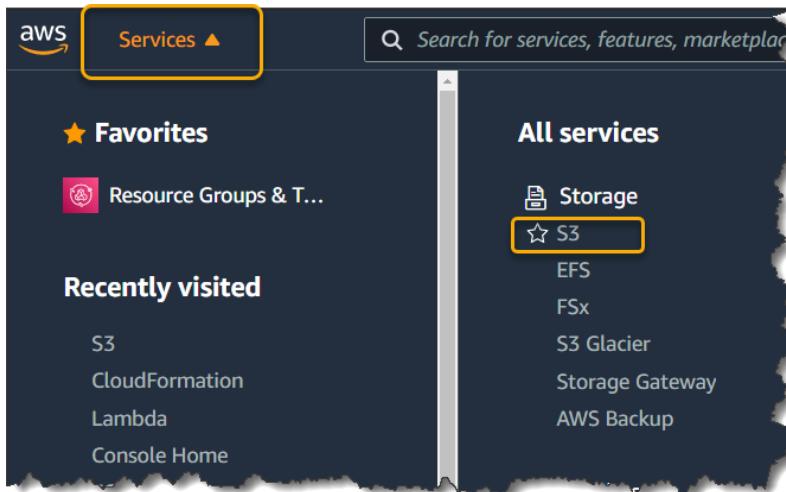
Edit the first line replacing “Your API URI” with the API gateway invoke URL you downloaded earlier.

The line should look like this.

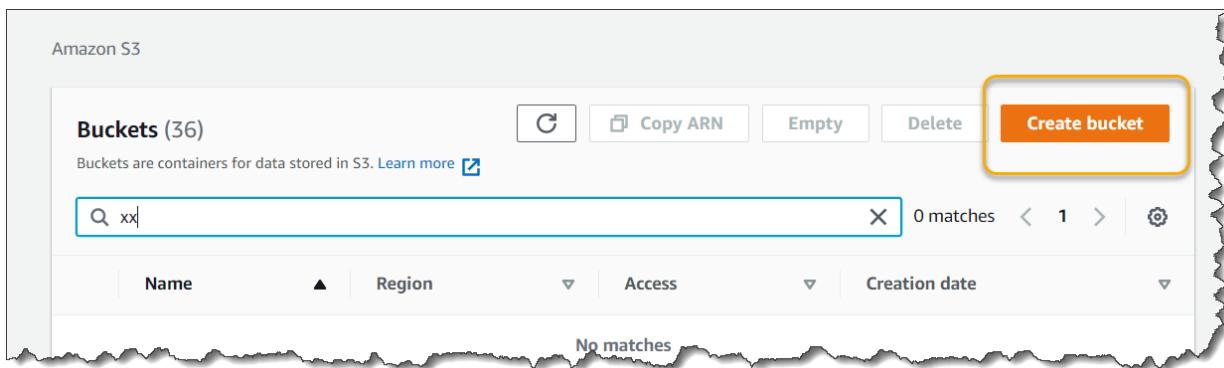
```
var API_ENDPOINT = https://oss4zy0c91.execute-api.ap-south-1.amazonaws.com/prod
```

Save the file.

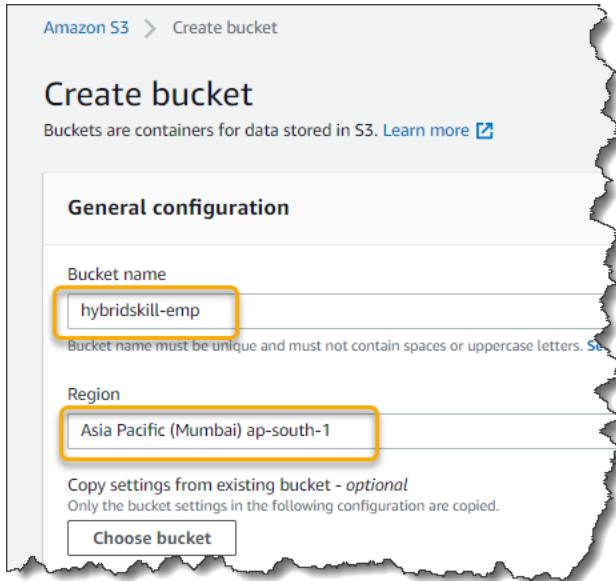
- Next create an S3 bucket. Click **Services**, click **S3**



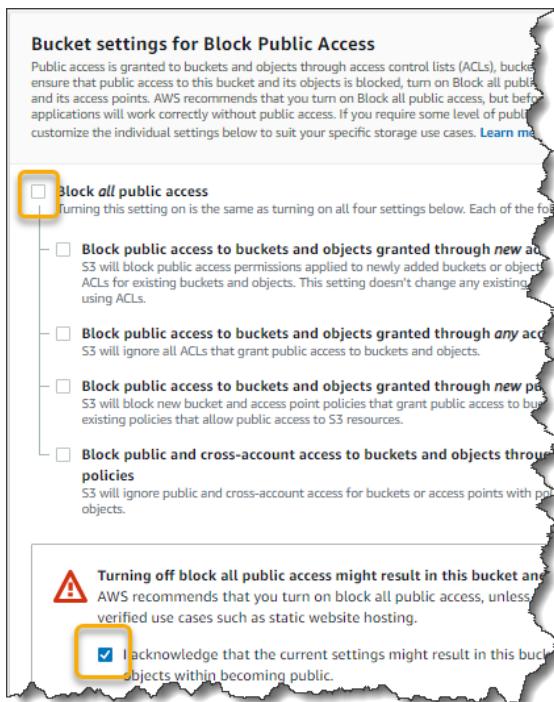
- Click **Create bucket**.



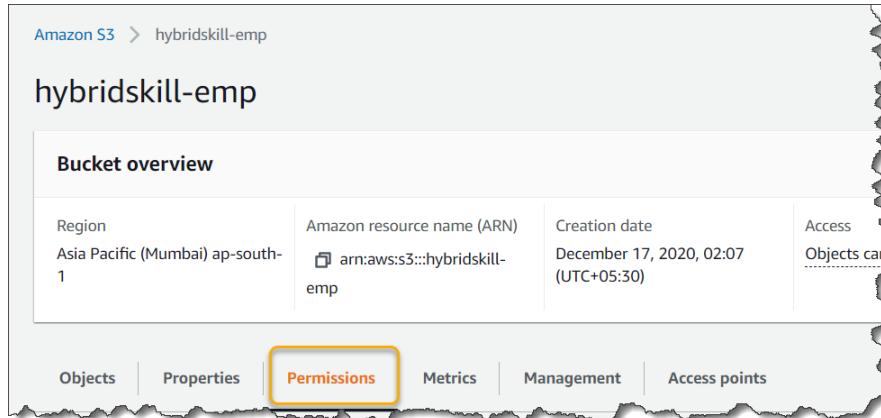
4. Enter the **Bucket name** as **hybridskill-emp**. Select the **Region** as **ap-south-1**



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



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



Amazon S3 > hybridskill-emp

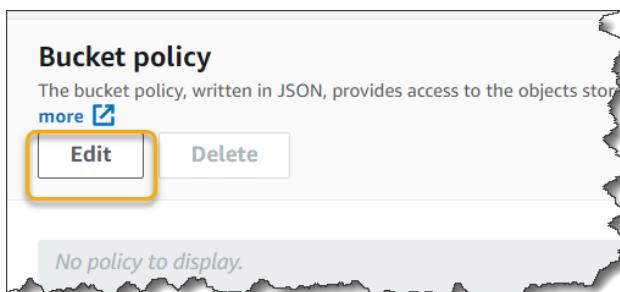
hybridskill-emp

Bucket overview

Region Asia Pacific (Mumbai) ap-south-1	Amazon resource name (ARN) arn:aws:s3:::hybridskill-emp	Creation date December 17, 2020, 02:07 (UTC+05:30)	Access Objects can
--	--	---	-----------------------

Objects Properties **Permissions** Metrics Management Access points

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



Bucket policy
The bucket policy, written in JSON, provides access to the objects stored in this bucket.

[more](#)

Edit Delete

No policy to display.

8. Click **Policy Examples**

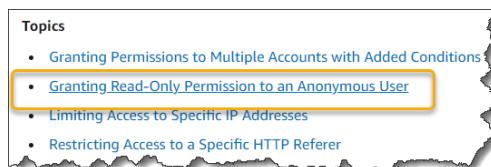


Bucket policy
The bucket policy, written in JSON, provides access to the objects stored in this bucket.

[other accounts. Learn more](#)

Policy examples [Policy generator](#)

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



Topics

- [Granting Permissions to Multiple Accounts with Added Conditions](#)
- **Granting Read-Only Permission to an Anonymous User**
- [Limiting Access to Specific IP Addresses](#)
- [Restricting Access to a Specific HTTP Referrer](#)

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

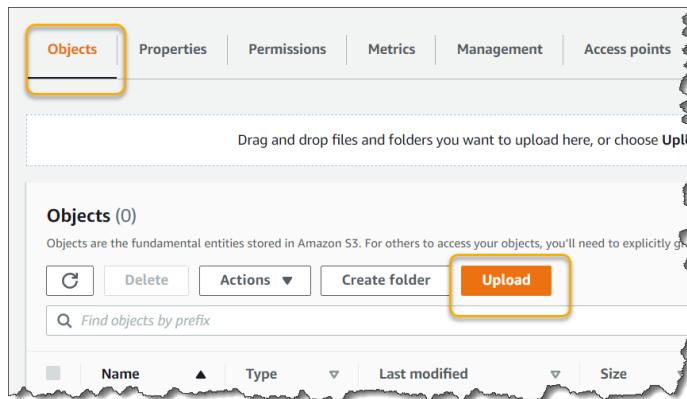


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:GetObject", "s3:GetObjectVersion"],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]
    }
  ]
}
```

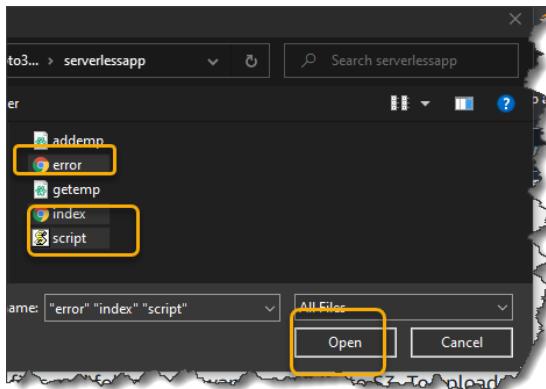
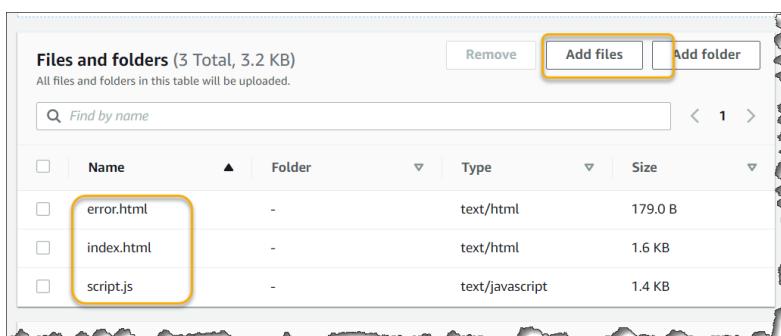
11. Replace *DOC-EXAMPLE-BUCKET* with your bucketname as follows. Click **Save Changes**



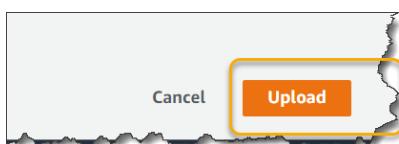
12. Next click **Objects** and click **Upload**



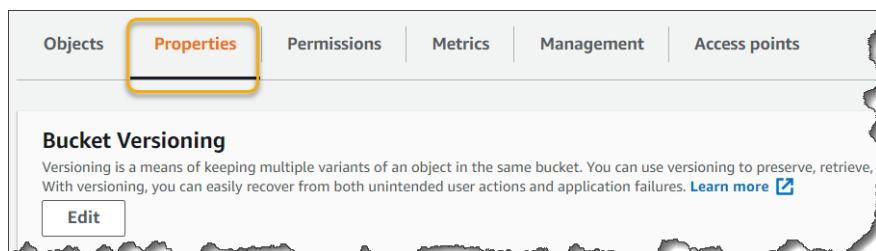
13. From the zipped folder you downloaded earlier, select **index.html**, **error.html** and **script.js** and **Upload** them into your bucket

	Name	Folder	Type	Size
<input type="checkbox"/>	error.html	-	text/html	179.0 B
<input type="checkbox"/>	index.html	-	text/html	1.6 KB
<input type="checkbox"/>	script.js	-	text/javascript	1.4 KB



14. Click **Properties**.



Bucket Versioning
 Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore objects. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

[Edit](#)

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**

Amazon S3 > hybridskill-emp > Edit static website hosting

Edit static website hosting

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

Disable
 Enable

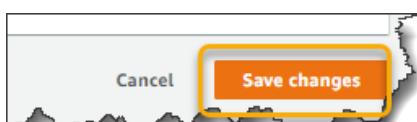
Hosting type
 Host a static website
 Use the bucket endpoint as the web address. [Learn more](#)
 Redirect requests for an object
 Redirect requests to another bucket or domain. [Learn more](#)

For your customers to access content at the website endpoint, you must make it readable. To do so, you can edit the S3 Block Public Access settings.
[Using Amazon S3 Block Public Access](#)

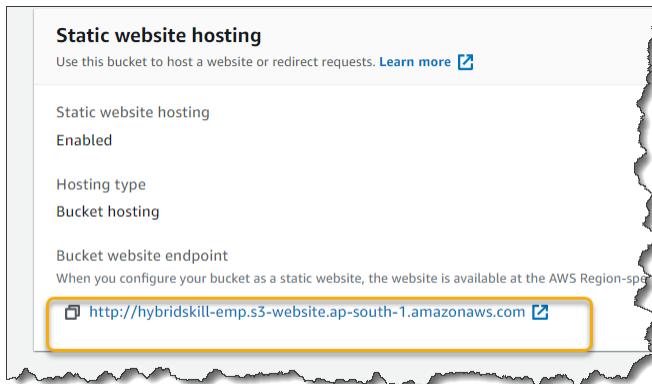
Index document
 Specify the home or default page of the website.

Error document
 This is returned when an error occurs.

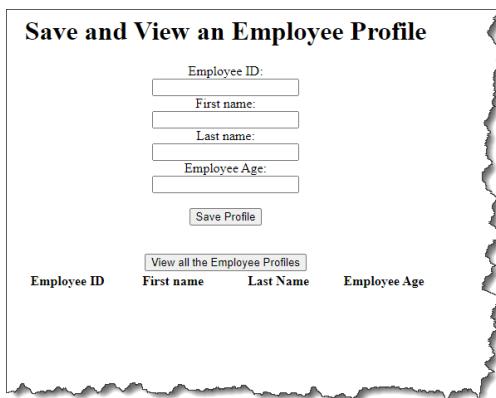
Redirection rules – optional
Set up redirection rules to automatically forward traffic from one URL to another.



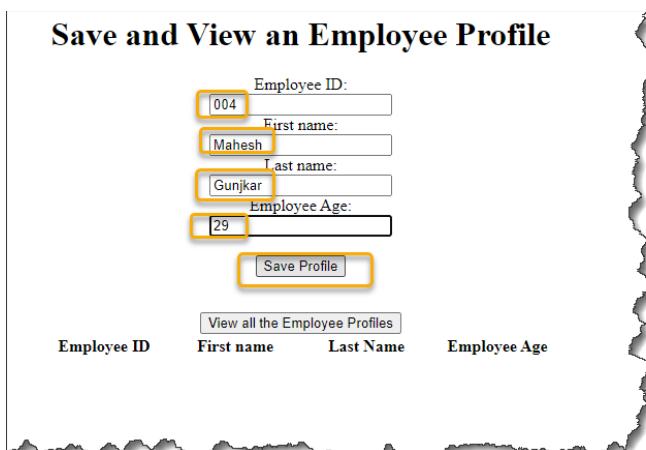
17. Find and view the bucket website endpoint on a browser.



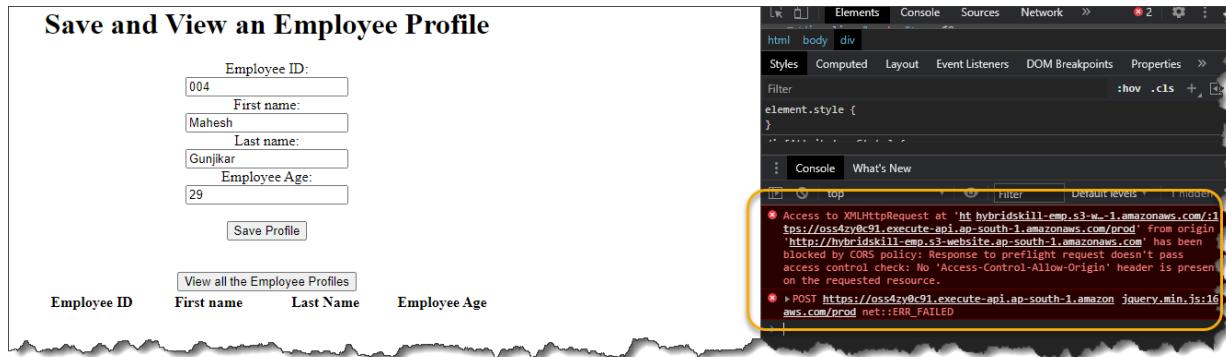
18. Your index page should be visible on a browser



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



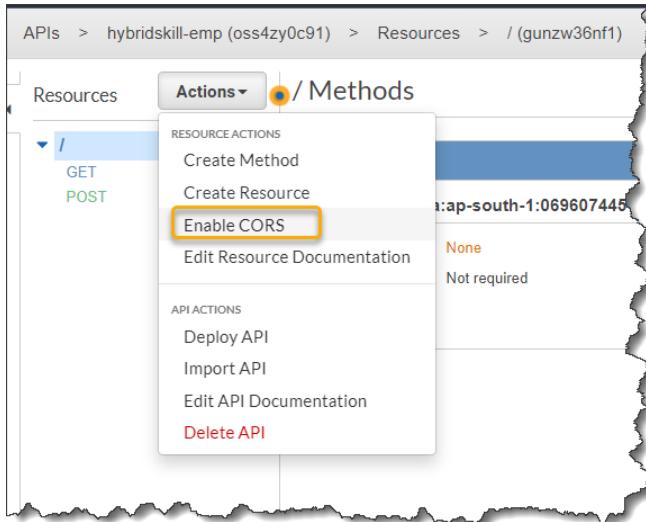
20. Inspect Element and you should see a CORS error



The screenshot shows a browser's developer tools open to the 'Console' tab. A yellow box highlights two error messages:

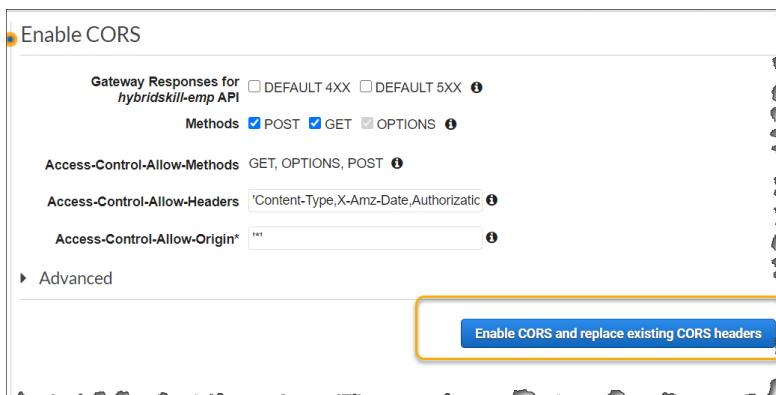
- Access to XMLHttpRequest at 'https://oss4zy0c91.execute-api.ap-south-1.amazonaws.com/prod' from origin 'http://hybridskillemp.s3-website.ap-south-1.amazonaws.com' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
- POST https://oss4zy0c91.execute-api.ap-south-1.amazonaws.com/prod jquery.min.js:16 net::ERR_FAILED

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



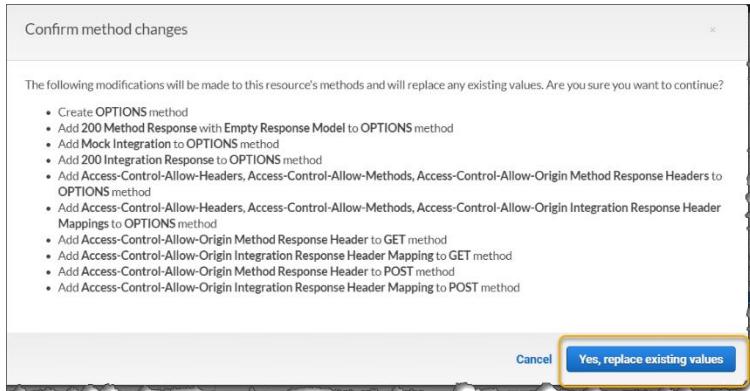
The screenshot shows the AWS API Gateway interface. A dropdown menu is open under the 'Actions' button for a specific resource. The 'Enable CORS' option is highlighted with a yellow box.

22. Click Enable CORS and replace existing headers

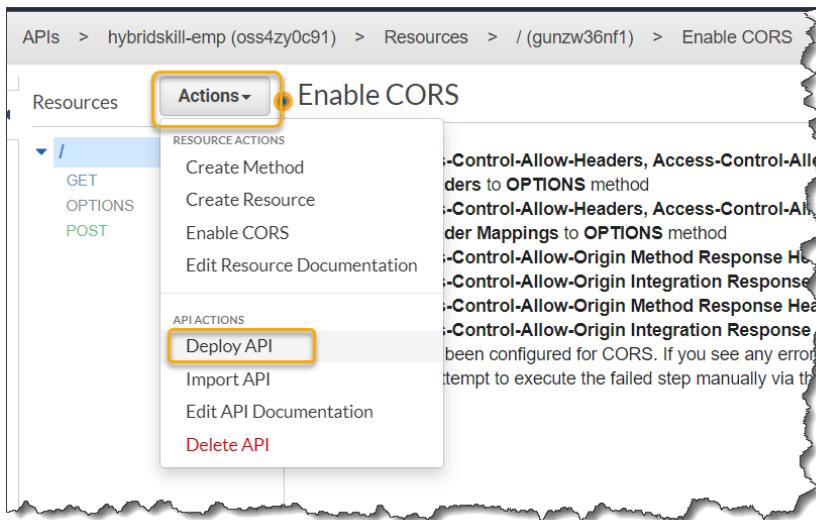


The screenshot shows the 'Enable CORS' configuration dialog. It includes fields for 'Gateway Responses for' (with checkboxes for DEFAULT 4XX and DEFAULT 5XX), 'Methods' (with checkboxes for POST, GET, and OPTIONS), and 'Access-Control-Allow-Methods' (set to GET, OPTIONS, POST). The 'Access-Control-Allow-Headers' field contains 'Content-Type,X-Amz-Date,Authorization'. The 'Access-Control-Allow-Origin*' field is empty. At the bottom is a blue button labeled 'Enable CORS and replace existing CORS headers'.

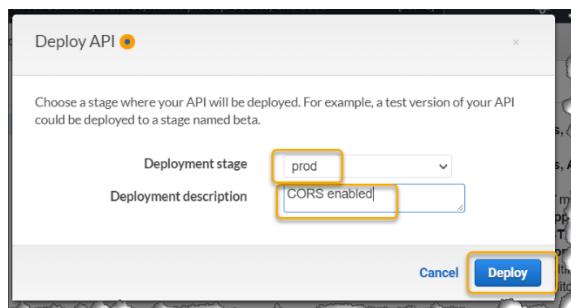
23. Click Yes, replace existing values



24. Select Actions and then Deploy API



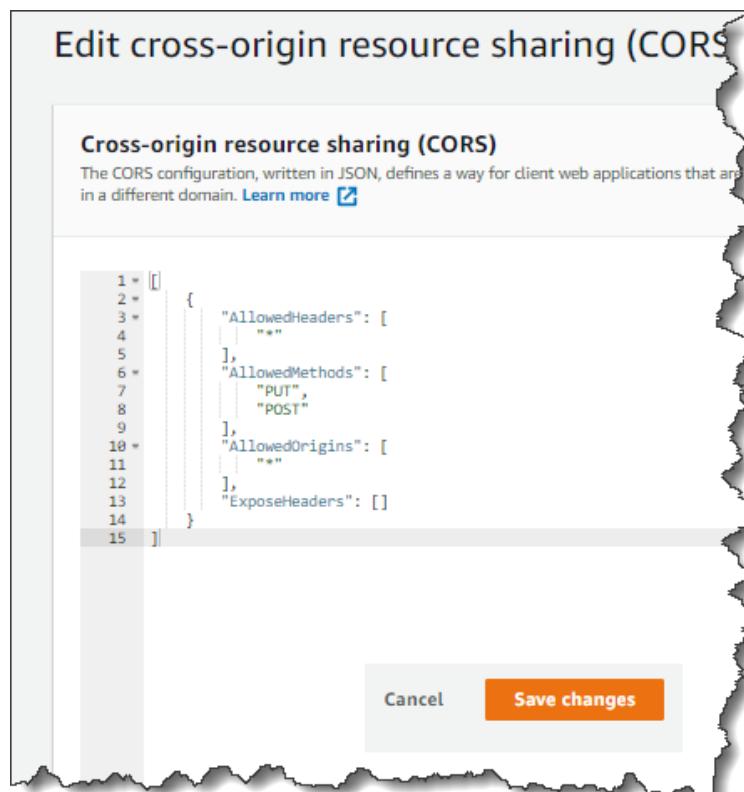
25. Select the prod stage and enter a Description. Click Deploy



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



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



The configuration JSON is as follows:

```
1: [
2:   {
3:     "AllowedHeaders": [
4:       "*"
5:     ],
6:     "AllowedMethods": [
7:       "PUT",
8:       "POST"
9:     ],
10:    "AllowedOrigins": [
11:      "*"
12:    ],
13:    "ExposeHeaders": []
14:  }
15: ]
```

At the bottom, there are 'Cancel' and 'Save changes' buttons, with 'Save changes' being highlighted with a yellow box.

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

Save and View an Employee Profile

Employee ID:	<input type="text" value="004"/>		
First name:	<input type="text" value="Mahesh"/>		
Last name:	<input type="text" value="Gunjikar"/>		
Employee Age:	<input type="text" value="29"/>		
<input type="button" value="Save Profile"/> Profile Saved!			
<input type="button" value="View all the Employee Profiles"/>			
Employee ID	First name	Last Name	Employee Age

29. You should see your employee information.

Profile Saved!		
<input type="button" value="View all the Employee Profiles"/>		
Employee ID	First name	Last Name
001	Anthony	Rozario
003	Siddharth	Yadav
002	Deepak	Kumar
004	Mahesh	Gunjikar

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