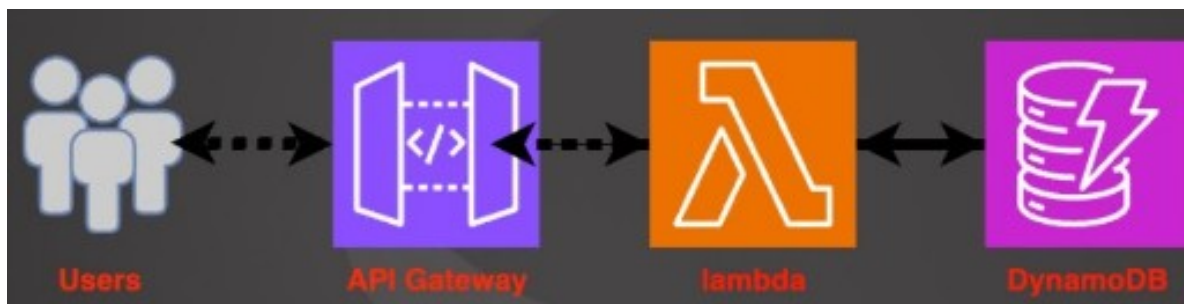

Project: Serverless Web Application with AWS Lambda and API Gateway

Description

In this project, you'll build a serverless web application using AWS Lambda for backend logic and Amazon API Gateway to expose APIs. You'll also integrate Amazon DynamoDB for data storage and use AWS IAM for security and permissions. This project will give you hands-on experience with AWS serverless architecture, allowing you to create scalable and cost-efficient applications.



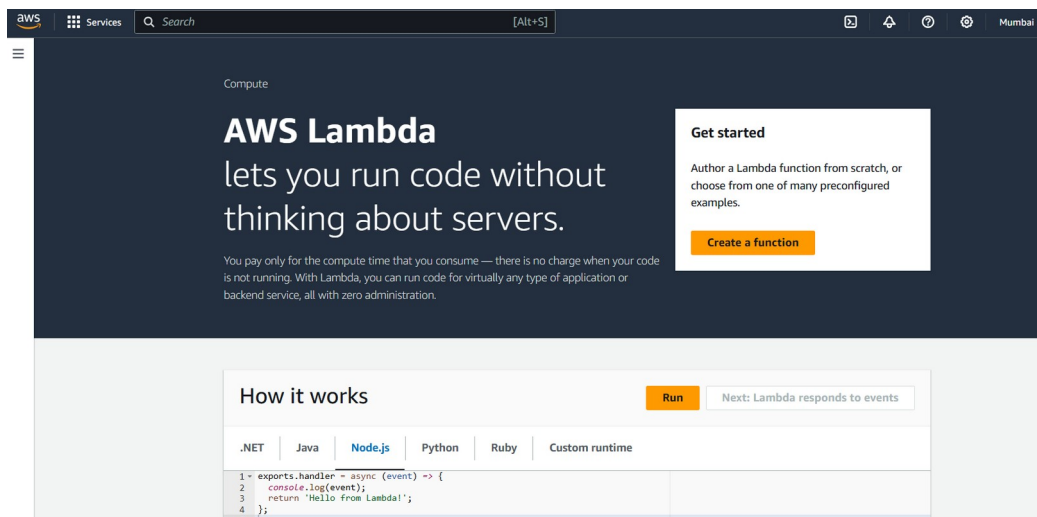
Step-by-Step Guide

Prerequisites

- AWS account
- Basic knowledge of programming (Python, Node.js, or another supported Lambda language)
- Basic understanding of RESTful APIs

Step 1: Set Up AWS Lambda

1. **Create a Lambda Function:**
 - o Open the AWS Management Console.
 - o Navigate to the Lambda service.
 - o Click "Create function".
 - o Choose "Author from scratch".
 - o Enter a function name (e.g., MyServerlessAppFunction).
 - o Choose the runtime (e.g., Python 3.8).
 - o Choose or create an execution role with basic Lambda permissions.
 - o Click "Create function".



[Lambda](#) > [Functions](#) > **Create function**

Create function [Info](#)

Choose one of the following options to create your function.

☒ **Author from scratch**
Start with a simple Hello World example.

☐ **Use a blueprint**
Build a Lambda sample code and presets for common use cases.

Basic information

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

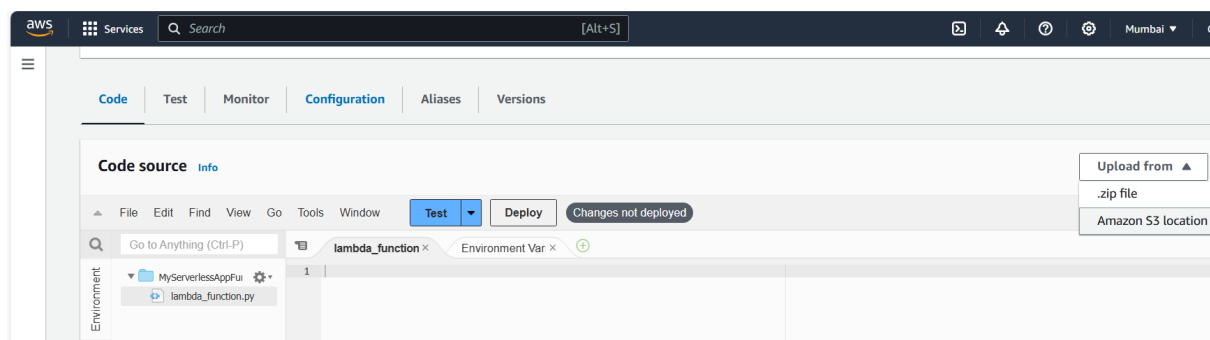
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console only shows the first few runtimes.

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.

☒ **x86_64**

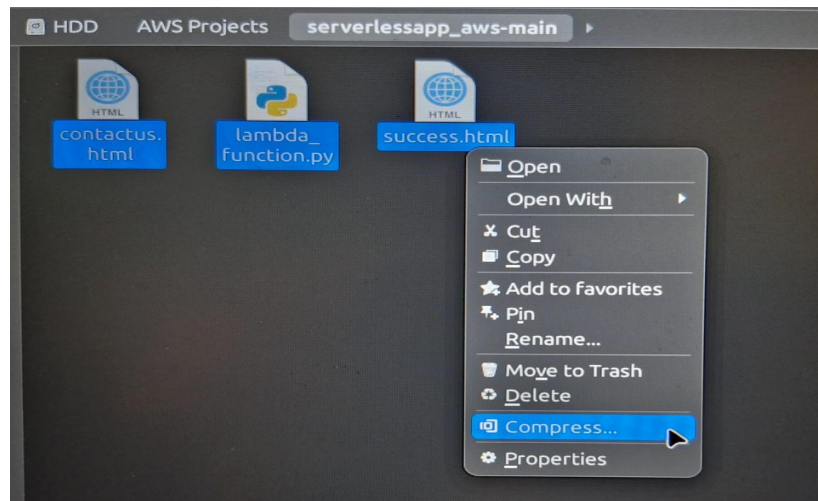
☐ arm64



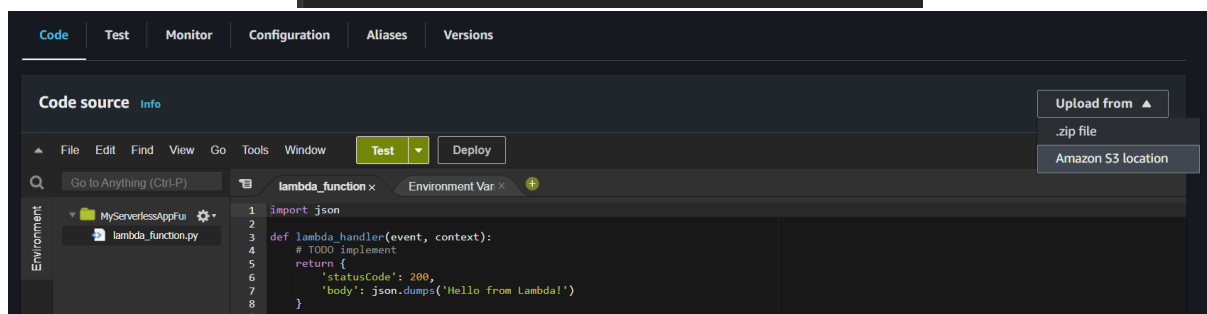
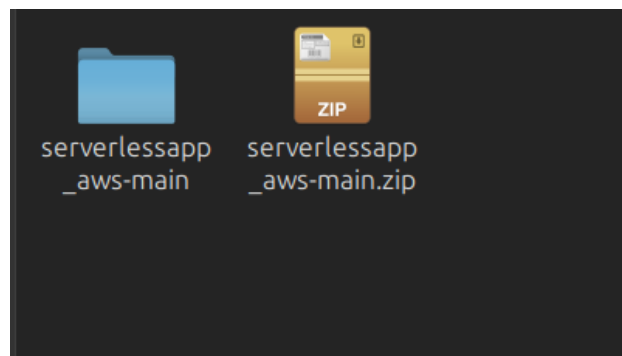
Now, upload a zip file of your project containing-frontend and backend files that are available in your local machine.

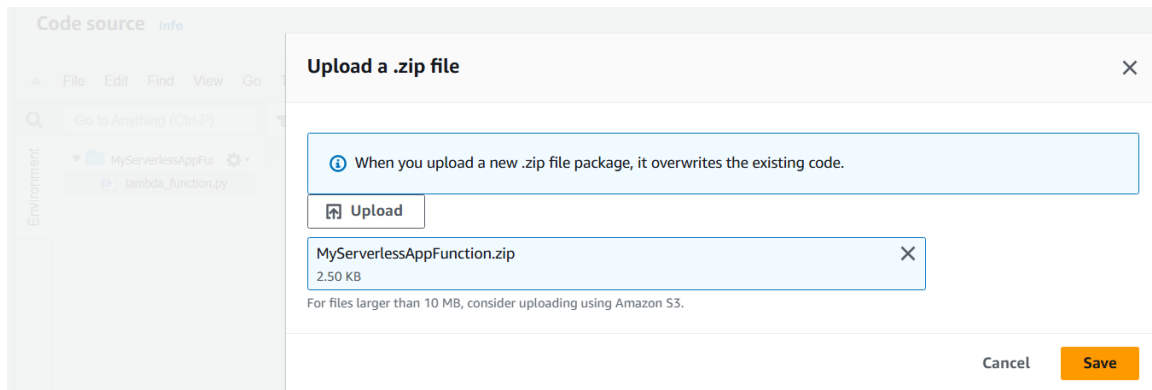
Source code from my GitHub: <https://github.com/thedarshanbhandari/serverless-app>

Compressed into the zip file:

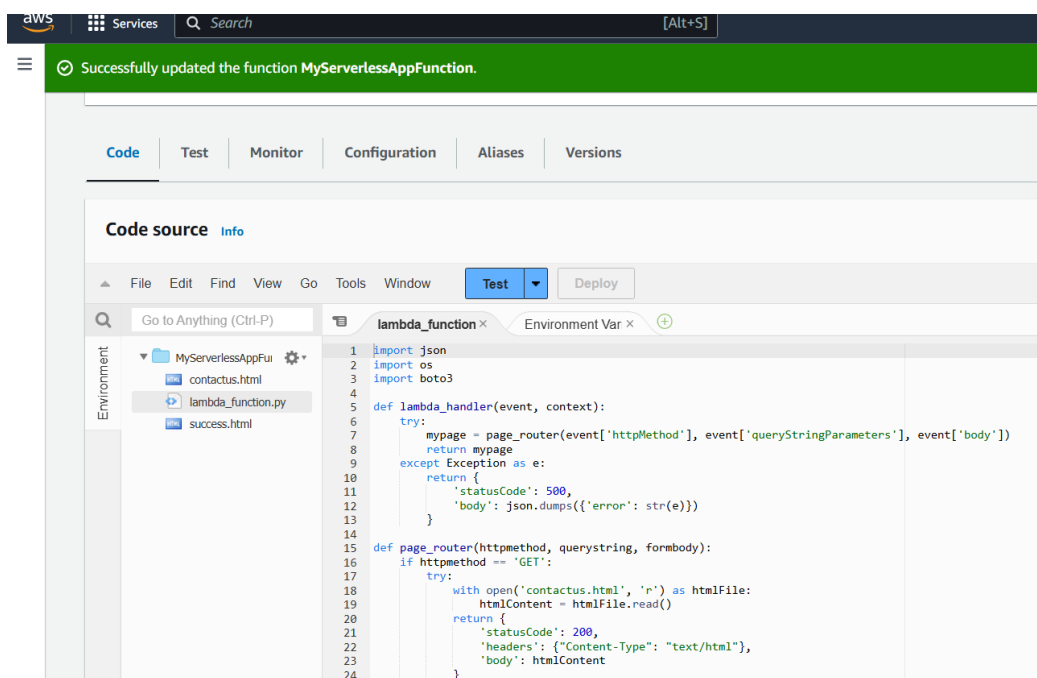


In Lambda, uploading as a zip file.

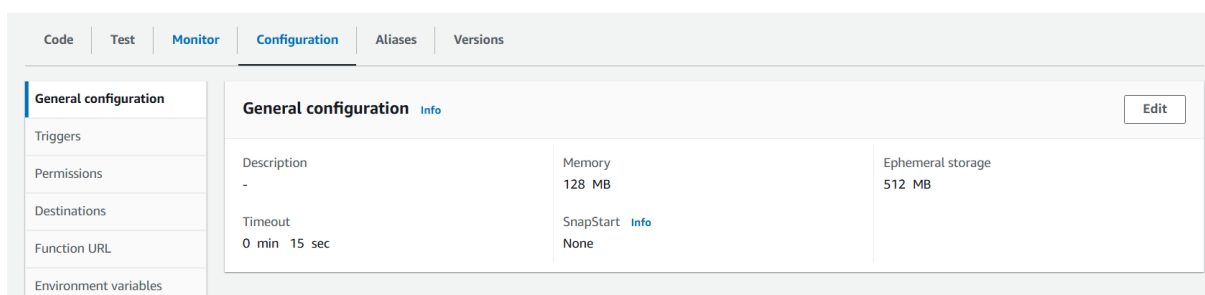




Then it looks like this in your lambda function page:



Edit the general configuration in lambda function, set timeout to 15sec.



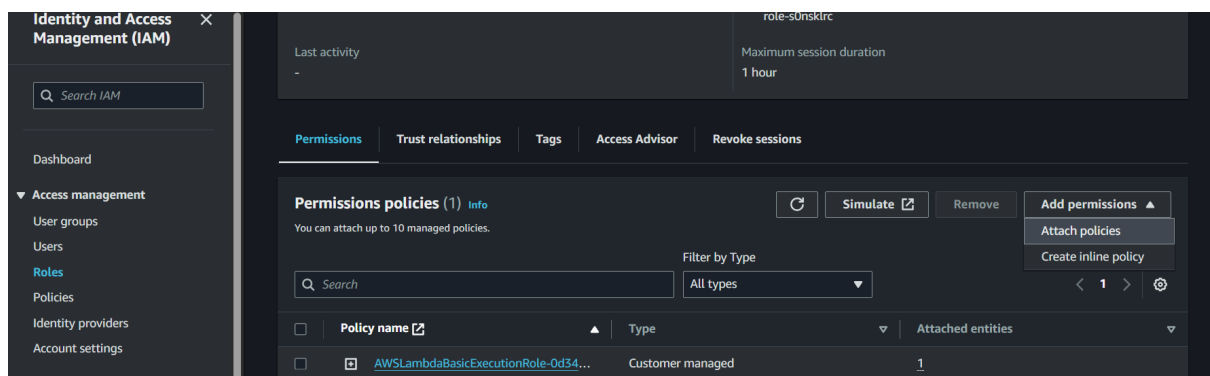
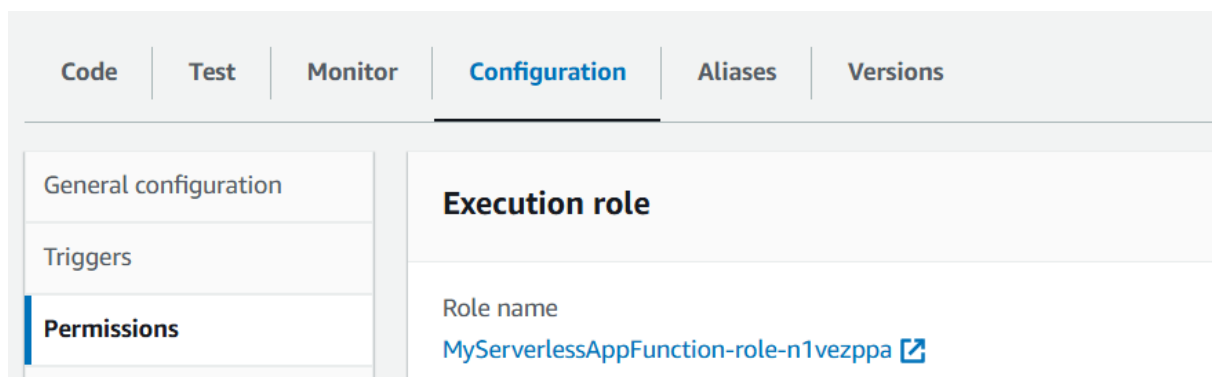
Add IAM Role under Permissions in the same tab:

- Ensure your Lambda execution role has the necessary permissions to interact with DynamoDB.

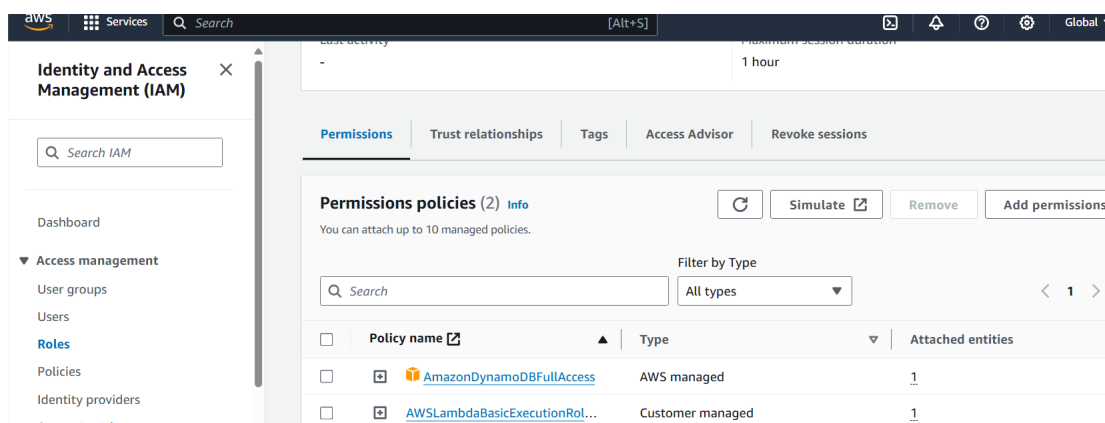
- Open the IAM console.
- Find and select your Lambda execution role.
- Attach the AmazonDynamoDBFullAccess policy or a custom policy with the required permissions.
- Also, attach S3 Get object permission(Optional) if you are trying to upload your zip file using AWS S3 bucket into lambda function.

Or else,

Under Permission tab in lambda general configuration a link is attached, Open the link in new tab:



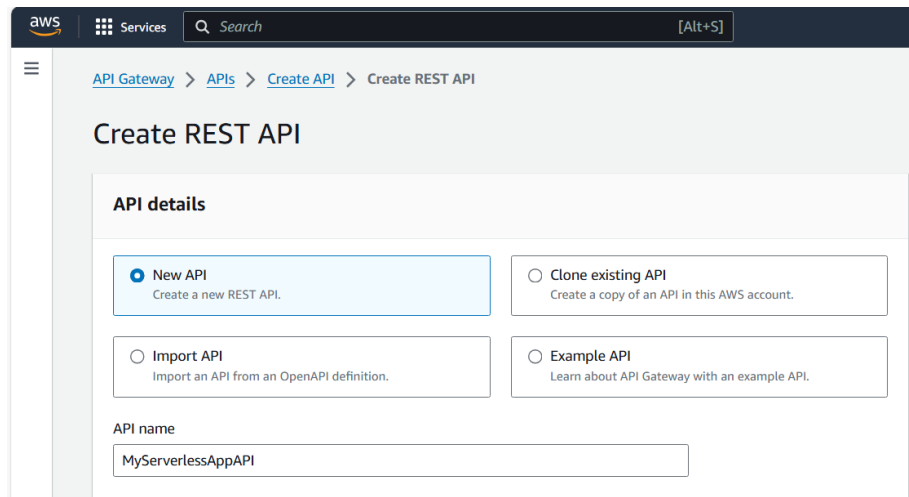
Attach respective dynamodb permissions and you can see these attached policies:



Step 2: Configure API Gateway

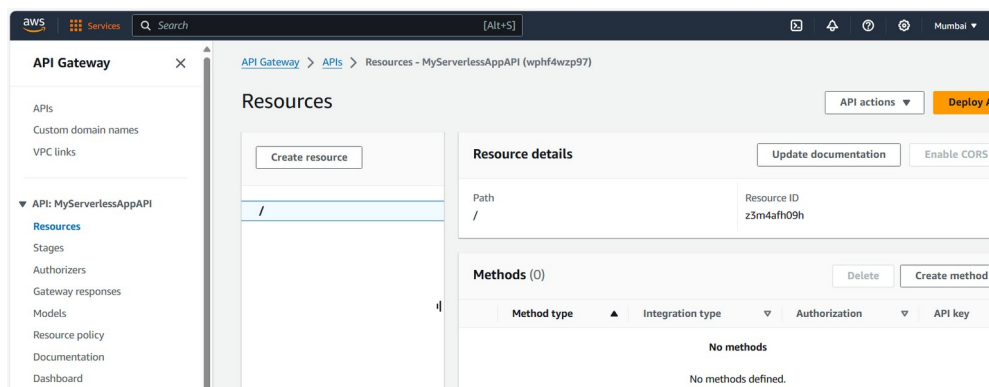
1. Create an API:

- o Open the API Gateway console.
- o Click "Create API".
- o Choose "REST API" (recommended for simplicity) and click "Build".
- o Enter an API name (e.g., MyServerlessAppAPI).
- o Click "Create".



2. Create Resource and Integrate with Lambda:

- o In the "Resource" section, right side you can see “Create Method”



- o Choose the method (e.g., GET) ,
- o Click "Next".
- o Choose "Lambda" as the integration type.
- o Select the Lambda function you created earlier.
- o Click "Create".

Create method

Method details

Method type: GET

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.

☒ **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.

ap-south-1

Note: make sure you are enabling lambda proxy integration

☒ **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.

ap-south-1

Repeat the same process to add more routes, such as a POST method to create new items. Go to API that you created and open resources to create the POST method.

API Gateway

APIs
Custom domain names
VPC links

▼ API: MyServerlessAppAPI

Resources
Stages
Authorizers
Gateway responses
Models
Resource policy
Documentation

Resources

Create resource

Path: /

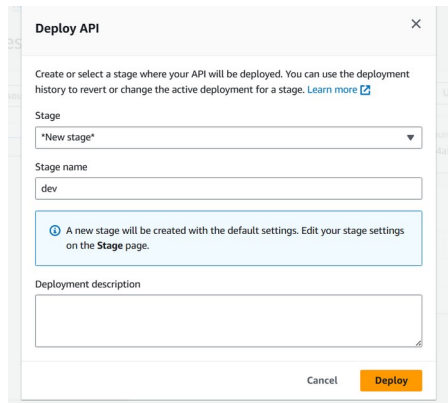
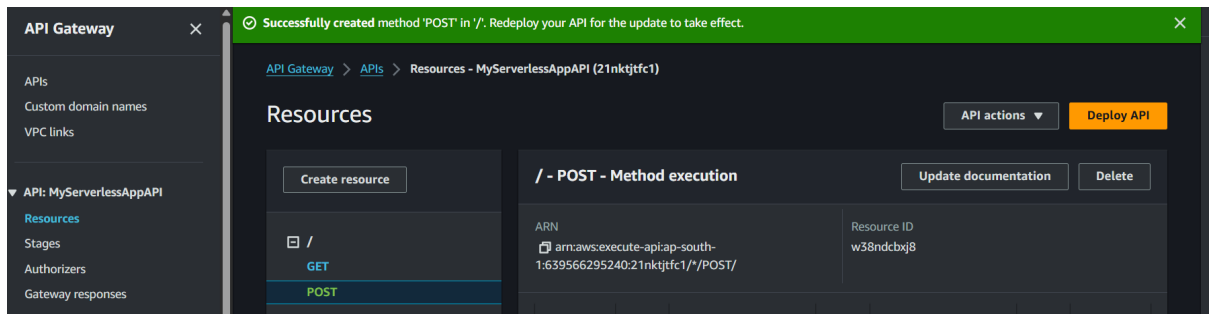
Resource ID: z3m4afh09h

Methods (2)

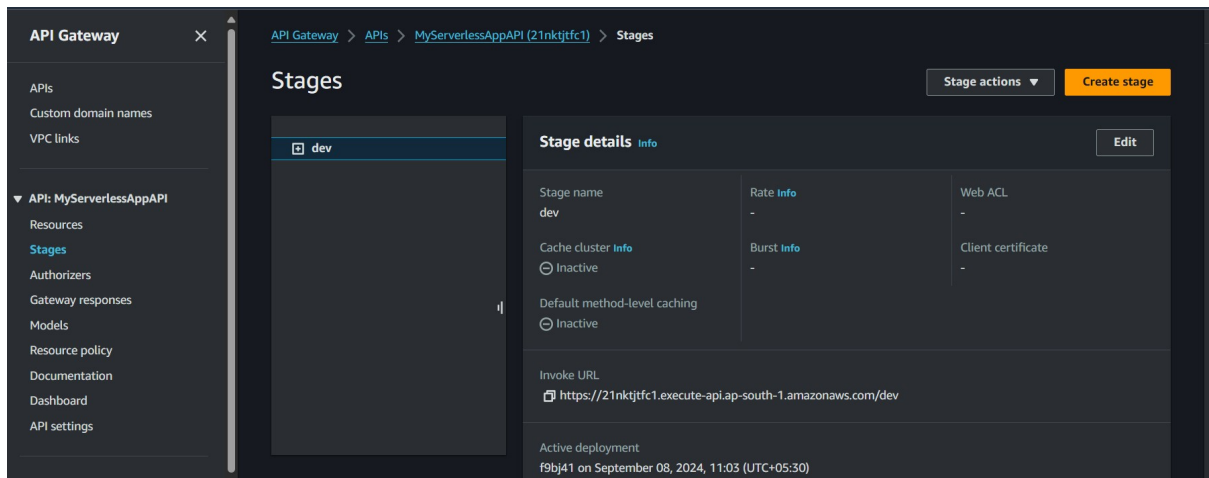
Method type	Integration type	Authorization	API key
GET	Lambda	None	Not required
POST	Lambda	None	Not required

3. Deploy the API:

- o After creating the routes, click "Deploy on top right" → create a new stage as dev → deploy.



- o Note the API endpoint invoke URL.



Step 3: Integrate DynamoDB

1. Create a DynamoDB Table:

- o Open the DynamoDB console.
- o Click "Create table".
- o Enter a table name (e.g., `project table`) as defined in your python code deployed in lambda.
- o Define the primary key (e.g., `email` as a string).
- o Click "Create table".

[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.

String ▼

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.

String ▼

1 to 255 characters and case sensitive.

DynamoDB × 🟢 The projecttable table was created successfully. ×

[DynamoDB](#) > [Tables](#)

Tables (1) [Info](#) 🔄 Actions ▼ Delete Create table

Any tag key ▼ Any tag value ▼ < 1 > ⚙️

<input type="checkbox"/>	Name ▲	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode
<input type="checkbox"/>	projecttable	🟢 Active	email (S)	-	0	🛑 Off	Provisioned (5)	Provisioned (5)

Test Your Serverless Application

1. **Invoke the API** using web-browser.
 - o Test the GET and POST methods to ensure they interact correctly with your Lambda function and DynamoDB table.

Add the data:

Welcome to my Cloud

Contact Us

First Name:

Last Name:

Email ID:

Message:

Submit

An AWS Project by Darshan Bhandari

Output message as:

Welcome to my Cloud

Contact Us

First Name:

Last Name:

Email ID:

Message:

An AWS Project by Darshan Bhandari

Thanks for trying this Project. "Innovative Solutions for a Brighter Future".

Now, I can explore the data from my DynamoDB table under explore items:

The screenshot shows the AWS Management Console interface for a DynamoDB table named 'projecttable'. The left sidebar contains navigation options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. The main content area shows the 'projecttable' details, including a 'Scan or query items' button and a status message: 'Completed. Read capacity units consumed: 0.5'. Below this, the 'Items returned (1)' section displays a table with one item:

	email (String)	fname	lname	message
<input type="checkbox"/>	saiguda966%40gmail...	sai	guda	Hello

Conclusion

By following these steps, you'll have a fully functional serverless web application using AWS Lambda, API Gateway, DynamoDB, and IAM Role. This project provides a solid foundation in serverless architecture and prepares you for more advanced AWS projects.

Thank you.

Regards:

Darshan Bhandari

<https://linkedin.com/in/darshan-bhandari-128793241>