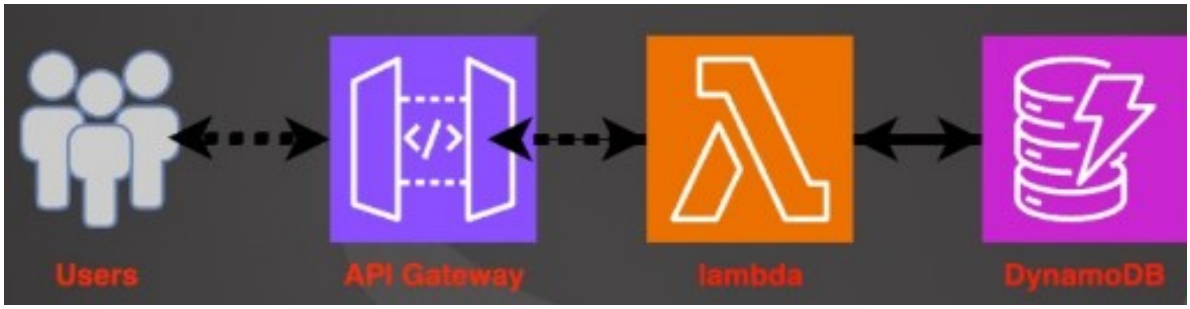
Author: Darshan Bhandari https://linkedin.com/in/darshan-bhandari-128793241

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.



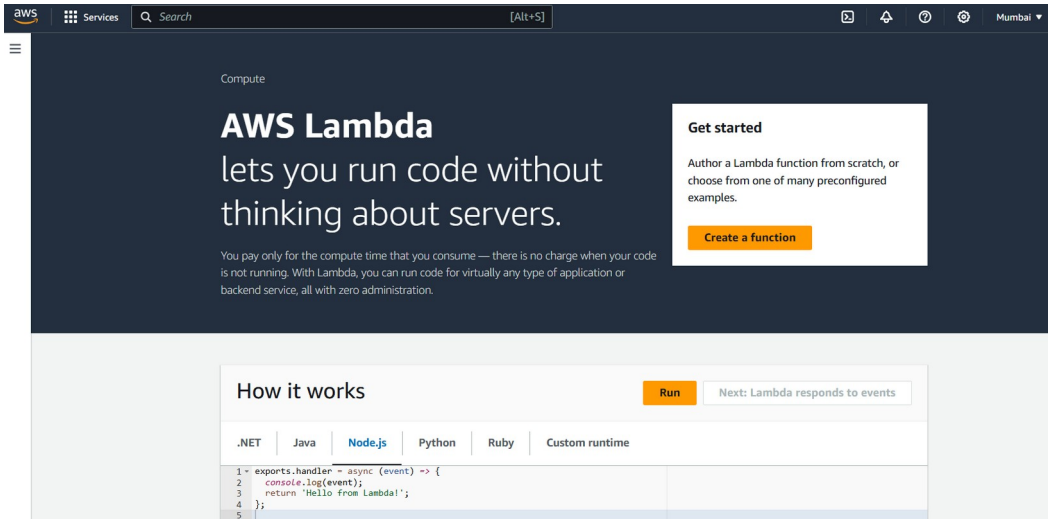
Step-by-Step Guide

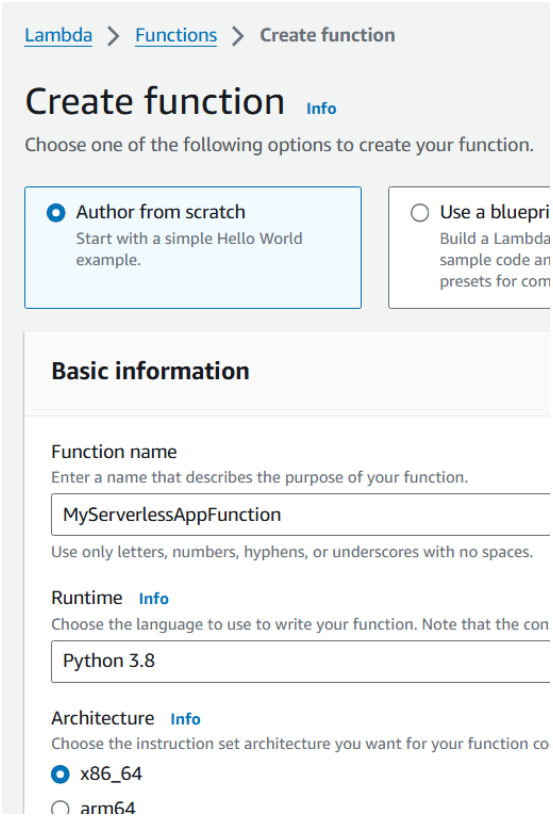
Prerequisites

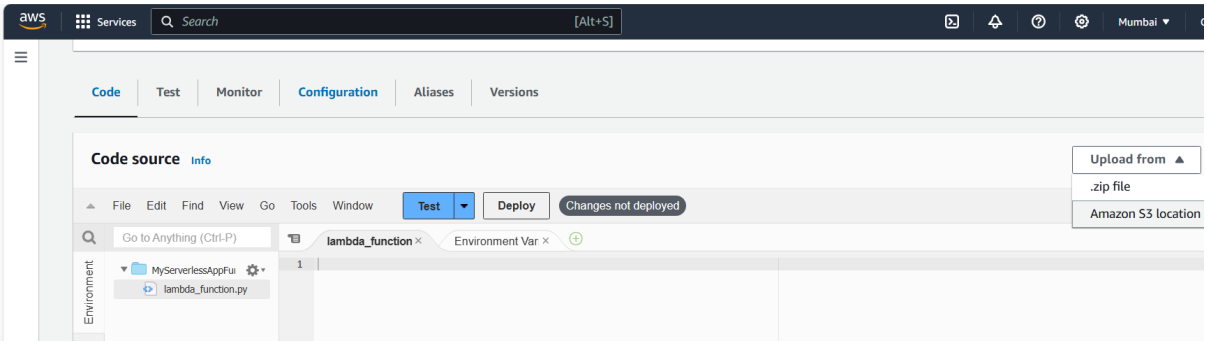
- AWS account
- Basic knowledge of programming (Python, Node.js, or another supported Lambda language)
- Basic understanding of RESTful APIs

Step 1: Set Up AWS Lambda

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


aws
Services
Search
[Alt+S]
Mumbai
Compute
AWS Lambda
lets you run code without thinking about servers.
You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.
Get started
Author a Lambda function from scratch, or choose from one of many preconfigured examples.
Create a function
How it works
Run
Next: Lambda responds to events
.NET
Java
Node.js
Python
Ruby
Custom runtime
exports.handler = async (event) => {
console.log(event);
return 'Hello from Lambda!';
};

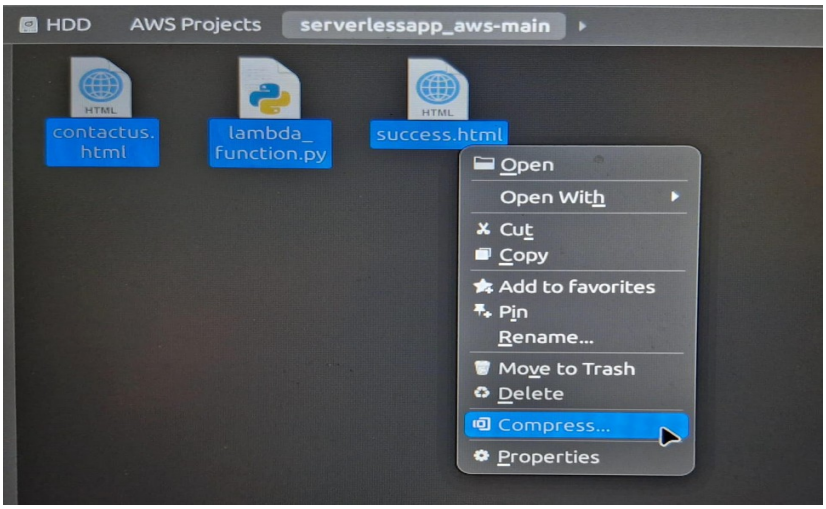

Lambda
Functions
Create function
Create function
Info
Choose one of the following options to create your function.
Author from scratch
Start with a simple Hello World example.
Use a bluepri
Build a Lambda
sample code an
presets for com
Basic information
Function name
Enter a name that describes the purpose of your function.
MyServerlessAppFunction
Use only letters, numbers, hyphens, or underscores with no spaces.
Runtime
Info
Choose the language to use to write your function. Note that the con
Python 3.8
Architecture
Info
Choose the instruction set architecture you want for your function co
x86_64
arm64


aws
Services
Search
[Alt+S]
Mumbai
Code
Test
Monitor
Configuration
Aliases
Versions
Code source
Info
Upload from
.zip file
Amazon S3 location
File
Edit
Find
View
Go
Tools
Window
Test
Deploy
Changes not deployed
Go to Anything (Ctrl-P)
lambda_function
Environment Var
MyServerlessAppFui
lambda_function.py
Environment

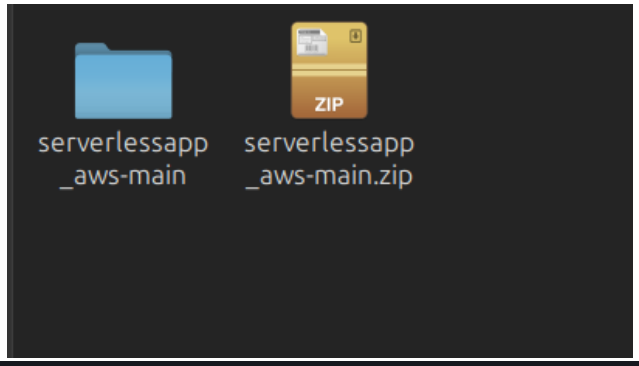
Now, upload a zip file of your project containing-frontend and backend files that are available in your local machine.

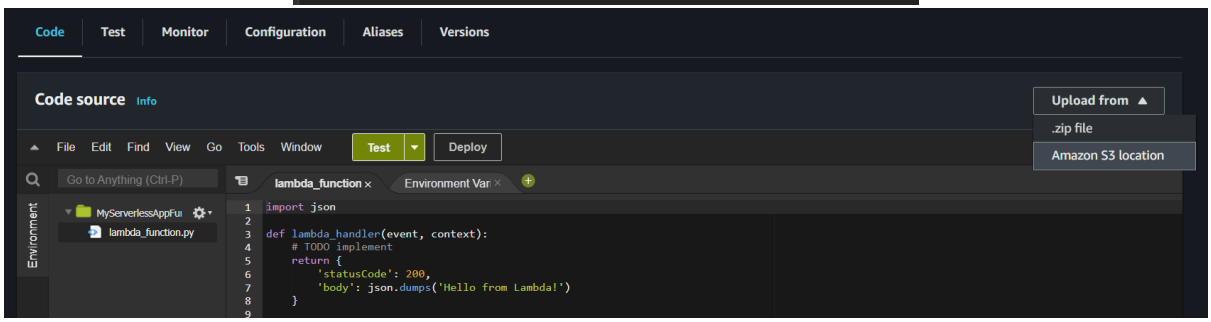
Source code from my GitHub: https://github.com/thedarshanbhandari/serverless-app

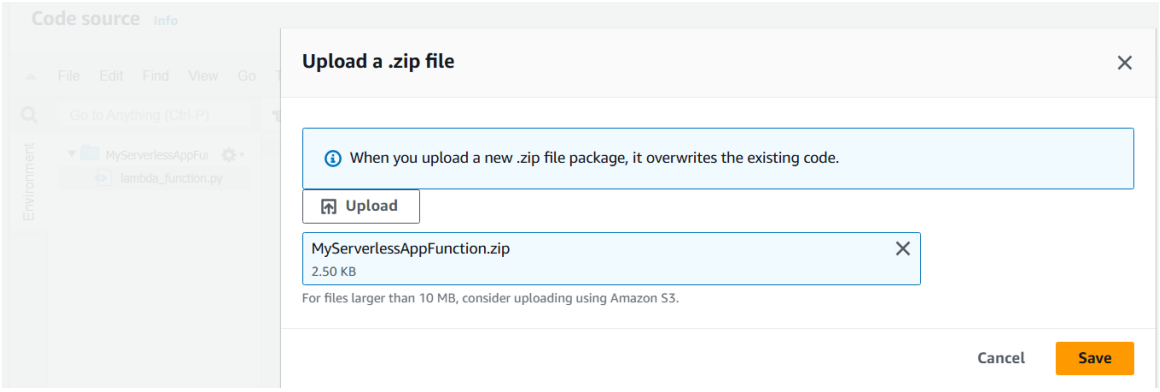
Compressed into the zip file:



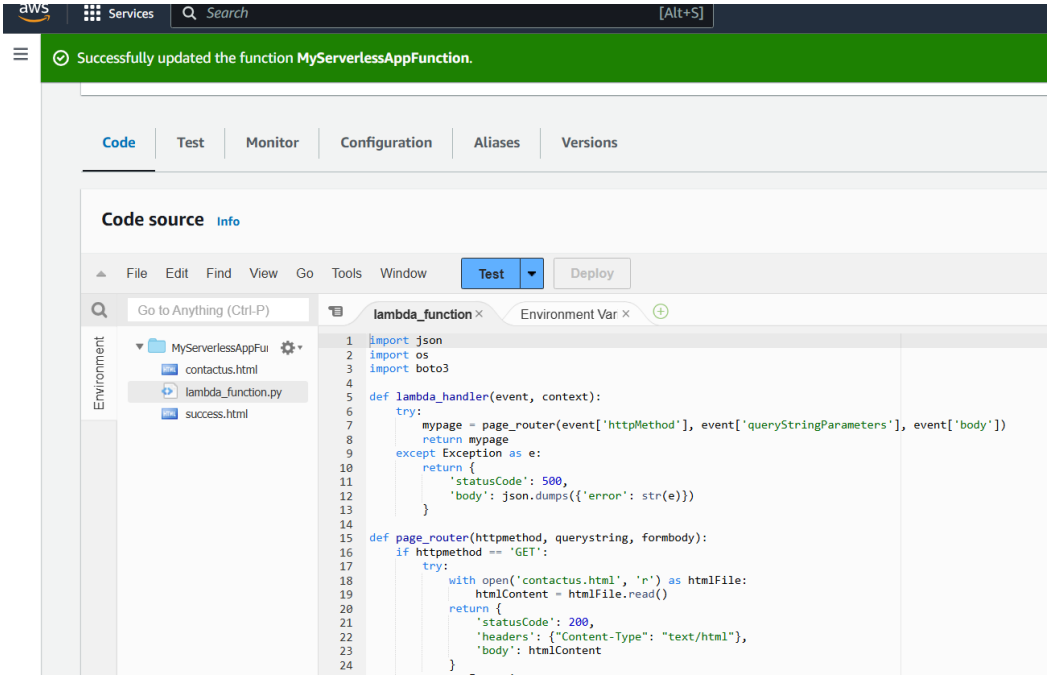
In Lambda, uploading as a zip file.



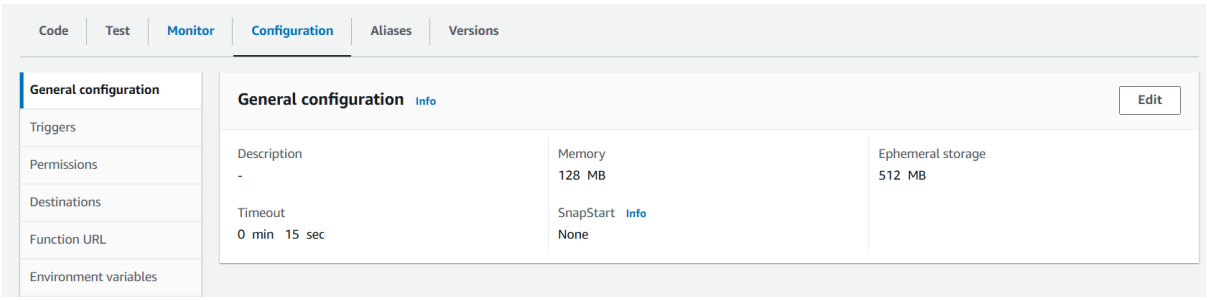




Then it looks like this in your lambda function page:



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



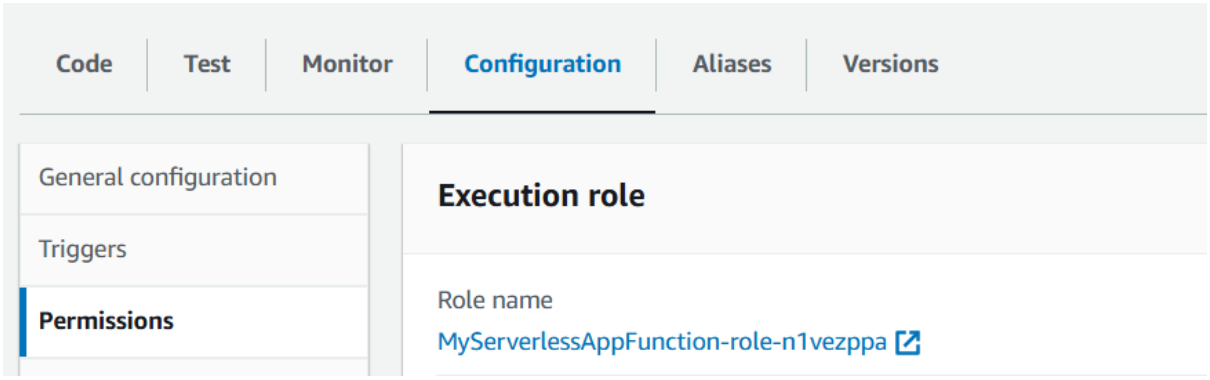
Add IAM Role under Permissions in the same tab:

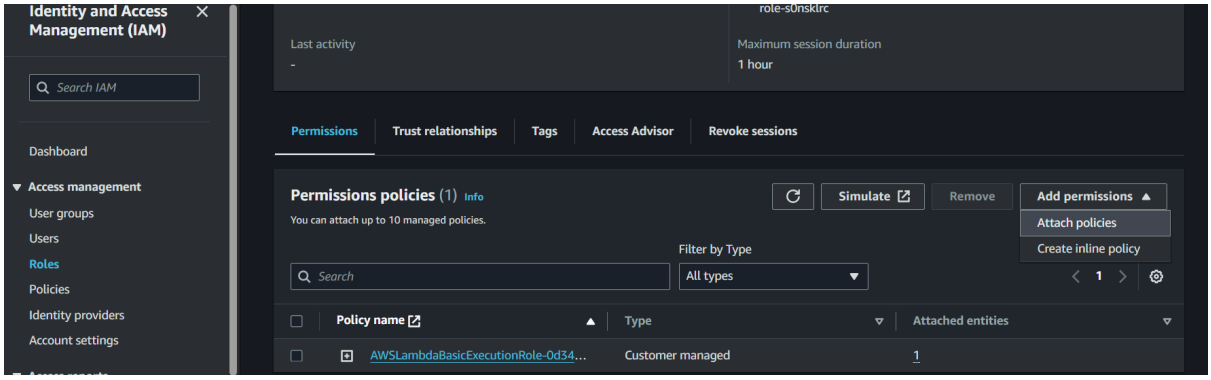
- Ensure your Lambda execution role has the necessary permissions to interact with DynamoDB.

- 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.

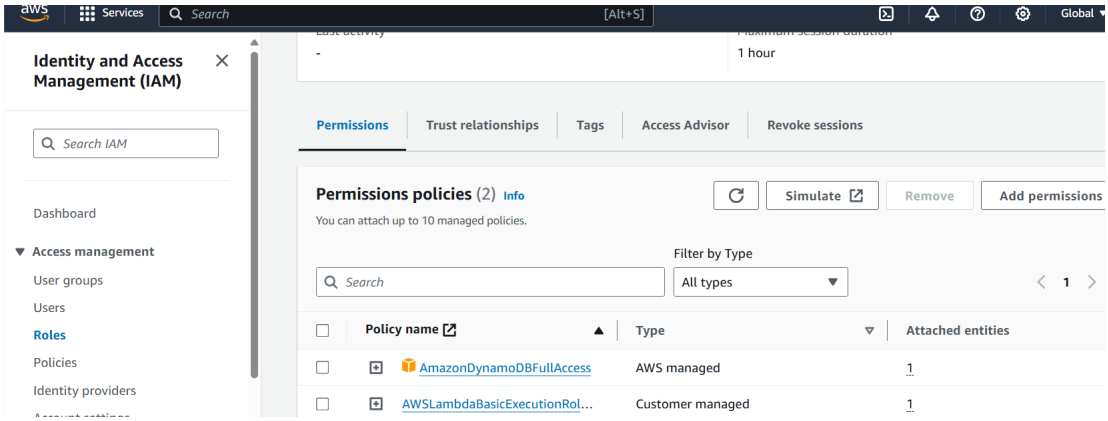
Or else,

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



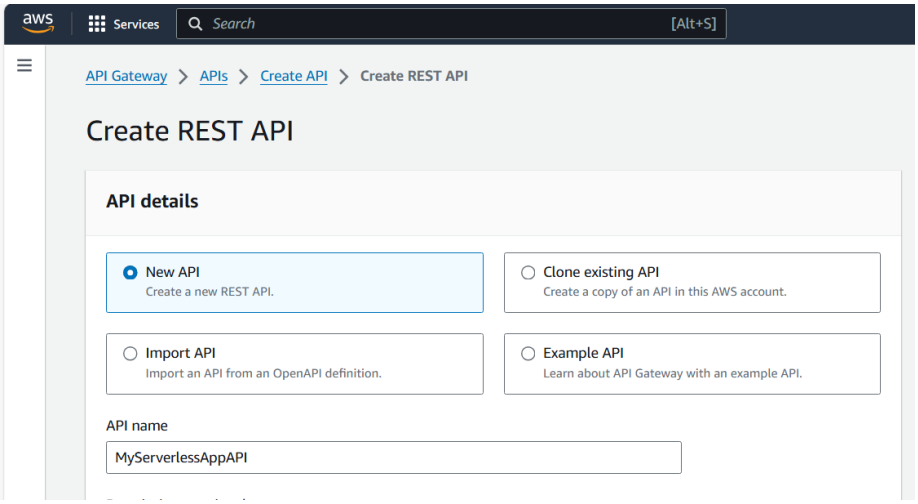


Attach respective dynamodb permissions and you can see these attached policies:

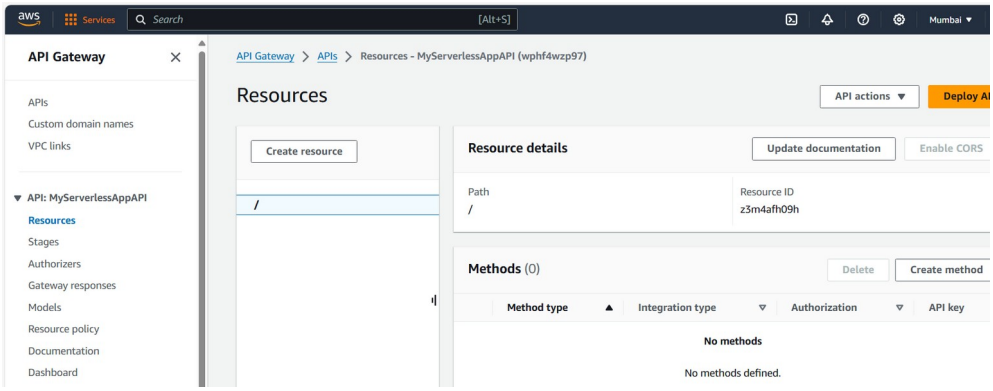


Step 2: Configure API Gateway

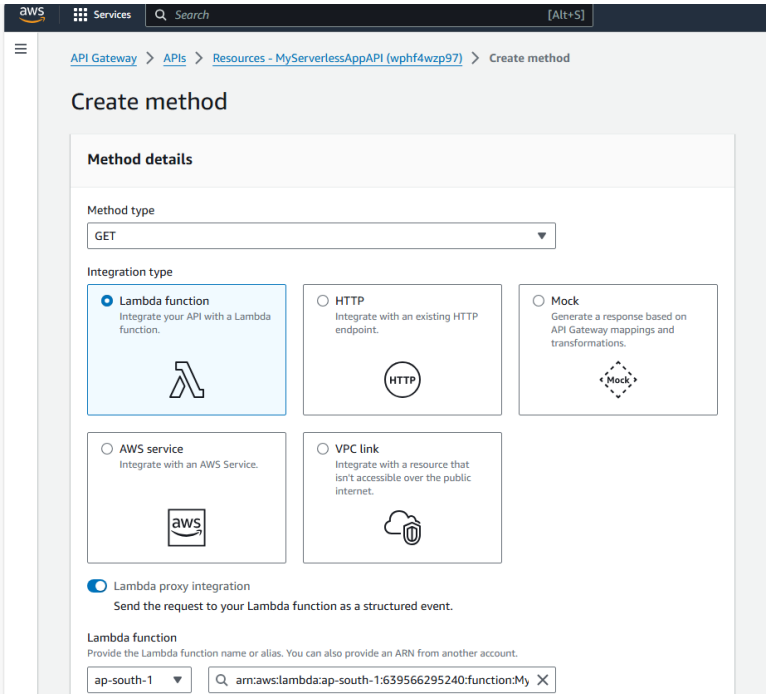
1. **Create an API**:
    - Open the API Gateway console.
    - Click "Create API".
    - Choose "REST API" (recommended for simplicity) and click "Build".
    - Enter an API name (e.g., `MyServerlessAppAPI`).
    - Click "Create".



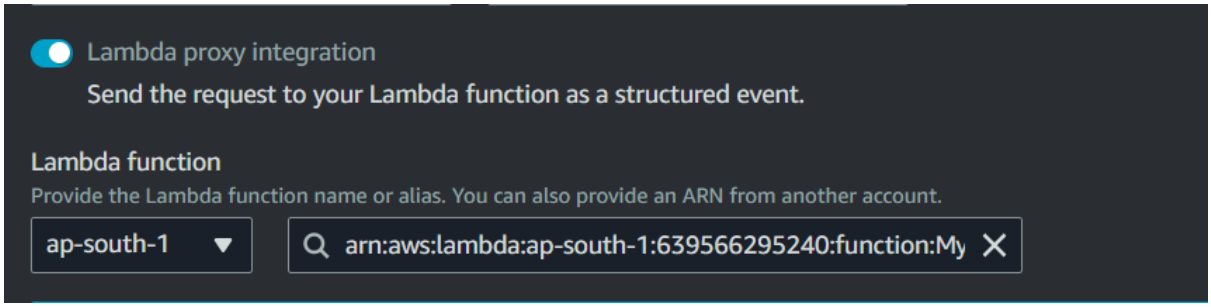
2. **Create Resource and Integrate with Lambda**:
    - In the "Resource" section, right side you can see “Create Method”



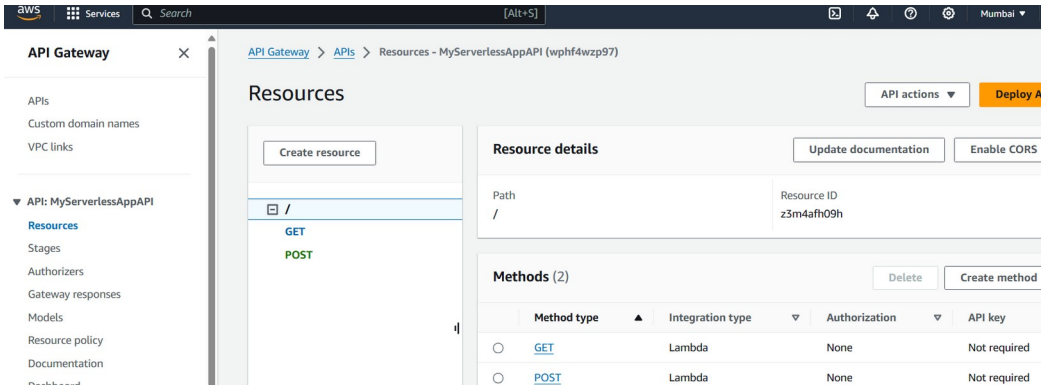
    - Choose the method (e.g., `GET`) ,
    - Click "Next".
    - Choose "Lambda" as the integration type.
    - Select the Lambda function you created earlier.
    - Click "Create".



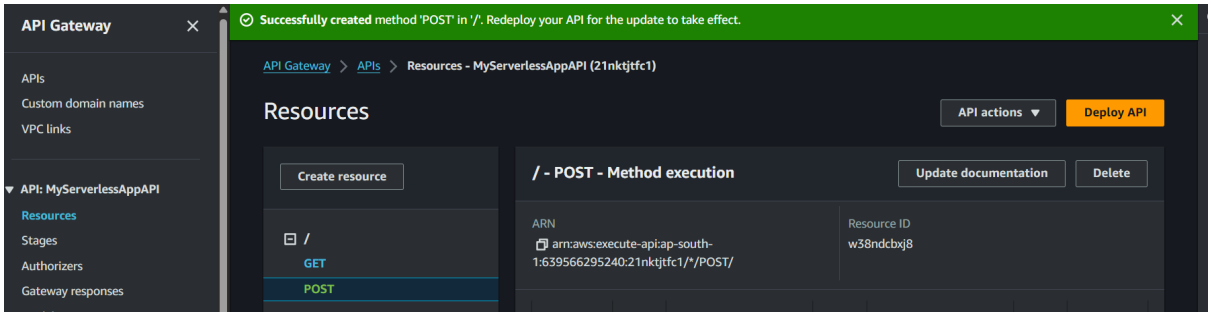
Note: make sure you are enabling lambda proxy integration

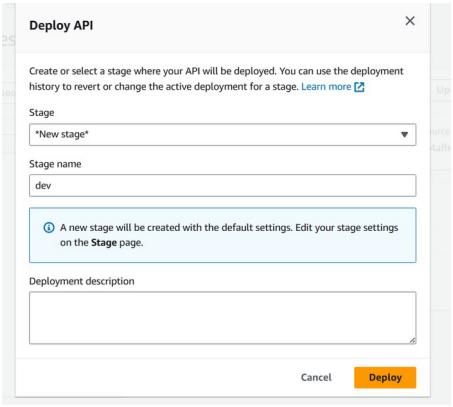


Repeat the same process to add more routes, such as a `POST` method to create new items. Go to API that you created and open resources to create the POST method.

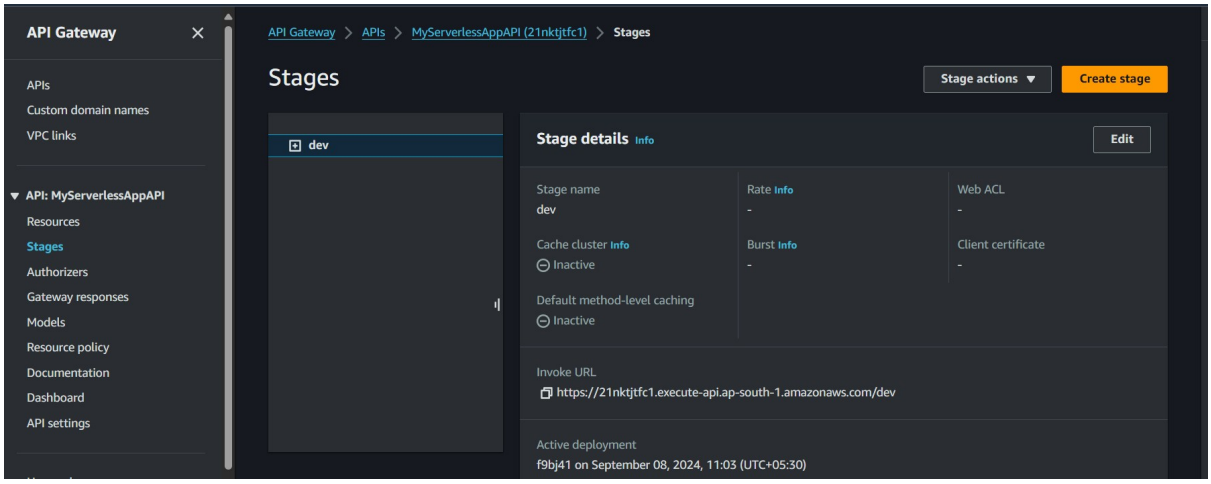


3. **Deploy the API**:
   - After creating the routes, click "Deploy on top right"→create a new stage as dev→deploy.



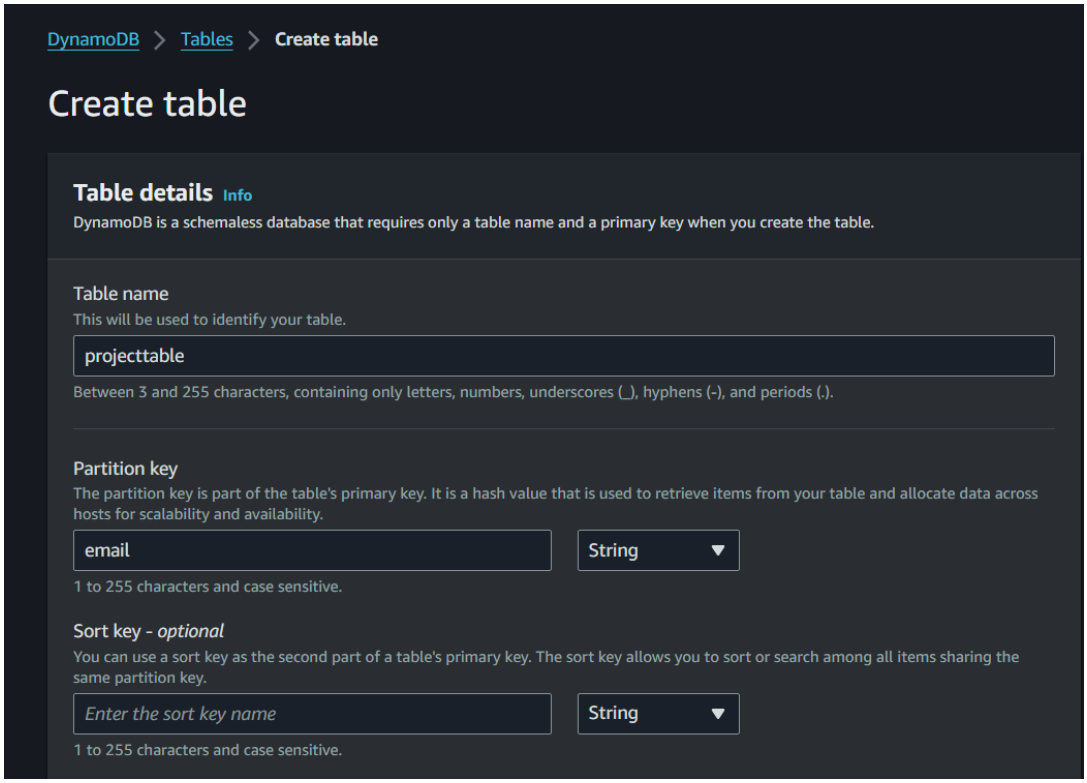


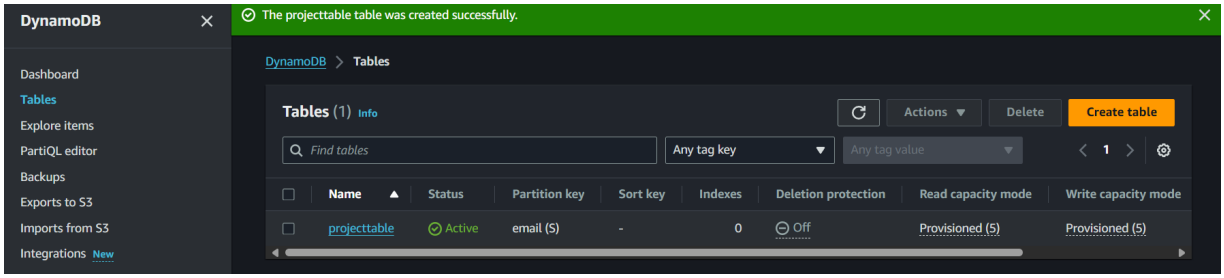
- Note the API endpoint invoke URL.



Step 3: Integrate DynamoDB

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

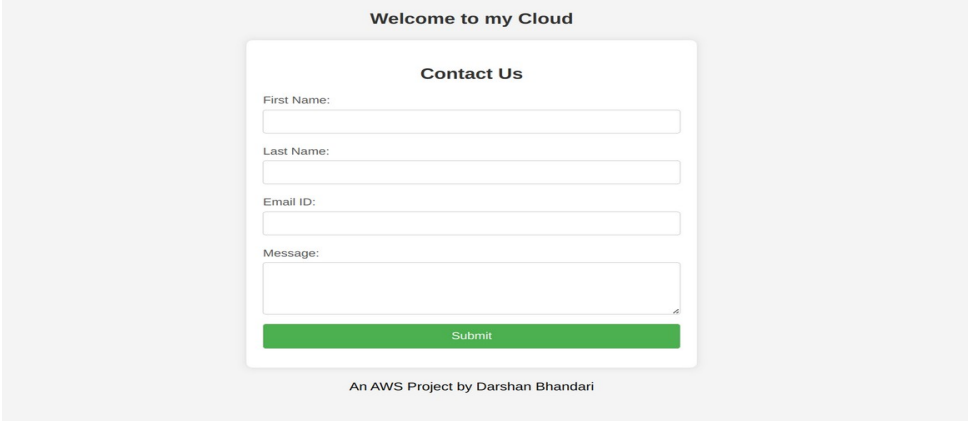




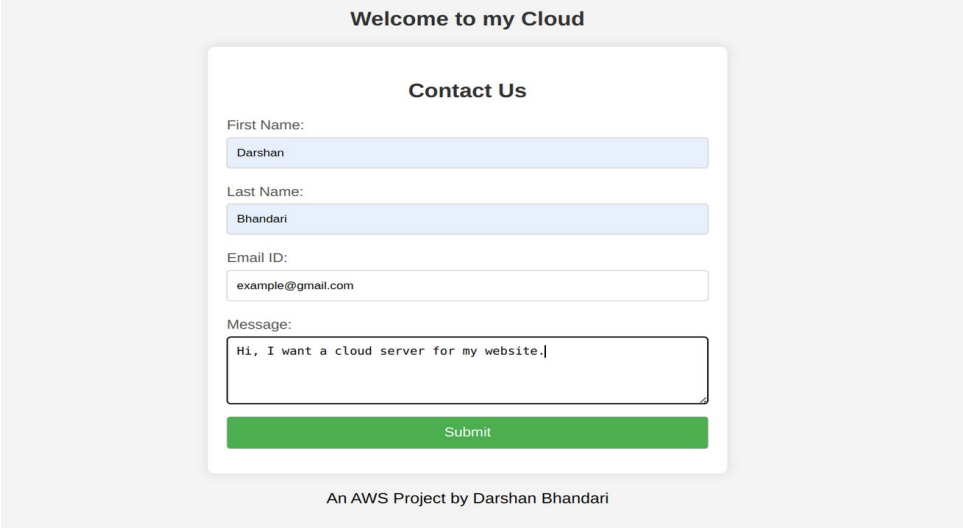
Test Your Serverless Application

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

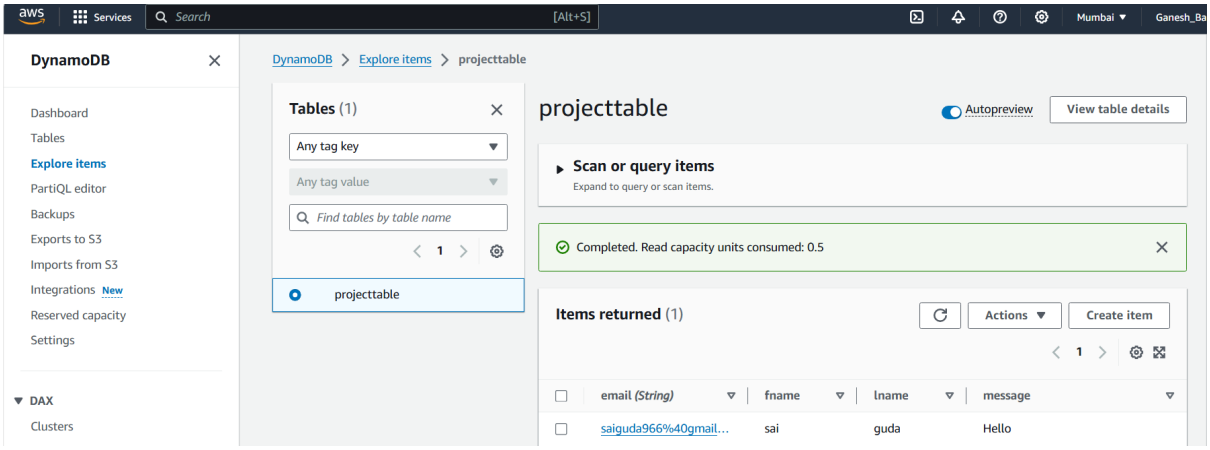
Add the data:



Output message as:



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



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