

PROJECT ON API Gateway-LAMBDA and DYNAMODB INTEGRATION

OVERVIEW

- User submits a form on a website.
 - The frontend sends a POST request to API Gateway.
 - API Gateway triggers a Lambda function.
 - Lambda:
 - Validates input.
 - Saves user data to DynamoDB.
 - Returns a success or error message.
 - API Gateway sends that response back to the client.
-

1. AWS Lambda

What it is: A compute service that lets you run backend code **without managing servers**.

- You write functions (e.g., in Node.js, Python, Java).
 - Triggered by events (like HTTP requests, DynamoDB changes, S3 uploads, etc.).
-

2. Amazon API Gateway

What it is: A fully managed service to create, publish, and manage **RESTful or WebSocket APIs**.

- Acts as a front door for apps to **securely call Lambda functions**.
 - Converts **HTTP requests into Lambda invocations**.
-

3. Amazon DynamoDB

What it is: A fully managed **NoSQL database** service.

- Fast, scalable, and serverless.
 - Ideal for key-value or document-based storage.
-

-----process start-----

- Open Dynamodb

- Click on **create table**

- Give table name as **veera**
- Partition key as a **email**
- If you change table name you need to change in lambda function

- Click on **create**

- Open IAM
- Click on create role

- Service is select lambda

Step 1
Select trusted entity [Info](#)

Step 2
Add permissions

Step 3
Name, review, and create

Select trusted entity

Trusted entity type

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
Lambda

Choose a use case for the specified service.
Use case

- Lambda

■ Add admin access

■ Click on next

Add permissions [Info](#)

Permissions policies (1/1047) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type		
<input type="text"/> Search	All types	
<input type="checkbox"/> Policy name	Type	Description
<input checked="" type="checkbox"/>  AdministratorAccess	AWS managed - job function	▼
<input type="checkbox"/>  AdministratorAccess-Amplify	AWS managed	▼
<input type="checkbox"/>  AdministratorAccess-AWSElasticBeanstalk	AWS managed	▼
<input type="checkbox"/>  AIOpsAssistantPolicy	AWS managed	▼
<input type="checkbox"/>  AIOpsConsoleAdminPolicy	AWS managed	▼
<input type="checkbox"/>  AIOpsOperatorAccess	AWS managed	▼
<input type="checkbox"/>  AIOpsReadOnlyAccess	AWS managed	▼
<input type="checkbox"/>  AlexaForBusinessDeviceSetup	AWS managed	▼

■ Give the name to role

■ Click on create

Role details

Role name
Enter a meaningful name to identify this role.

lambda-all

Maximum 64 characters. Use alphanumeric and '+,-,@-_' characters.

Description
Add a short explanation for this role.

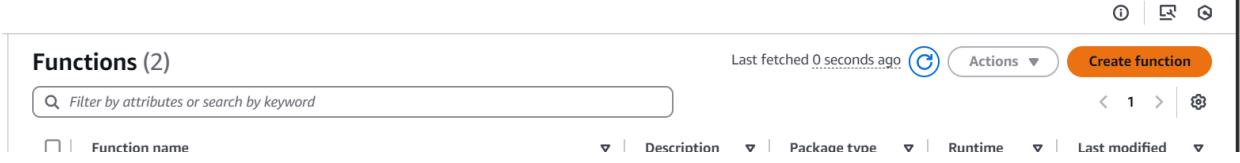
Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[\{\}!#\$%^&()::;"`

Step 1: Select trusted entities

Trust policy

- Open lambda
- Click on **create function**



- Give name
- Select the run time as python

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 application from sample code and configuration presets for common use cases.

Container image
Select a container image to deploy

Basic information

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

lambda-api

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.13

Architecture Info
Choose the instruction set architecture you want for your function code.

- Click on change default execution role
- Select use an existing role
- Select previously created role
- Click on create function

Multicloud With Devops by veera nareshit

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
 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. The role must have permission to upload logs to Amazon CloudWatch Logs.
lambda-all

View the lambda-all role [on the IAM console](#).

▶ Additional Configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

Cancel Create function

- In `lambda_function.py` file paste below code

https://github.com/CloudTechDevOps/project-api-lambda-dynamodb-intigration/blob/main/lambda_function.py

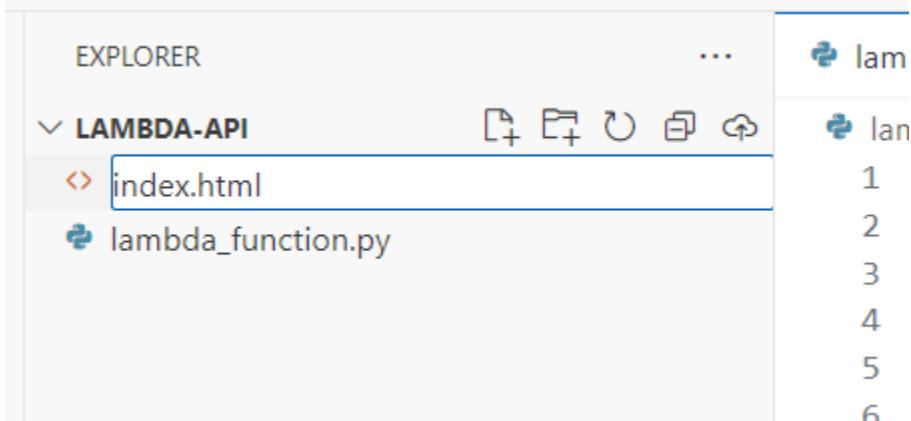
- Code will be on above git link copy and paste it

```
lambda_function.py
1 import json
2 import os
3 import boto3
4
5 def lambda_handler(event, context):
6     try:
7         mypage = page_router(event['httpMethod'], event['queryStringParameters'], event['body'])
8         return mypage
9     except Exception as e:
10         return {
11             'statusCode': 500,
12             'body': json.dumps({'error': str(e)})
13         }
14
15 def page_router(httpmethod, querystring, formbody):
16     if httpmethod == 'GET':
17         try:
18             with open('contactus.html', 'r') as htmlFile:
19                 htmlContent = htmlFile.read()
20             return {
```

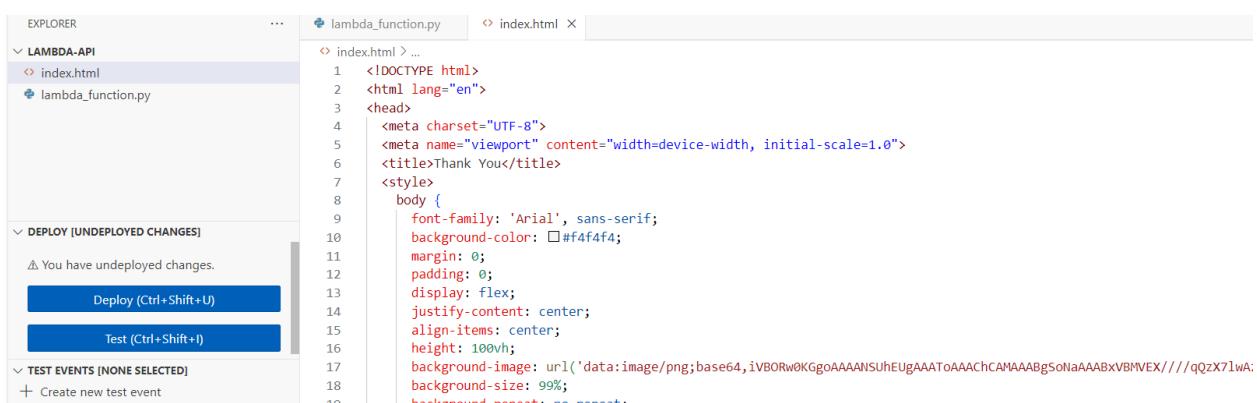
- Create a new file name as `index.html`
- Paste the below code

<https://github.com/CloudTechDevOps/project-api-lambda-dynamodb-intigration/blob/main/index.html>

- Code will be on above git link copy and paste it



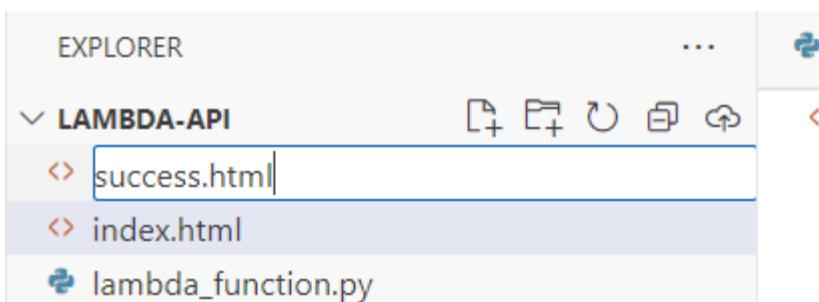
■ Like this



- Create a new file name as **success.html**
- Paste the below code

<https://github.com/CloudTechDevOps/project-api-lambda-dynamodb-intigration/blob/main/success.html>

- Code will be on **above git link** copy and paste it



■ Like this

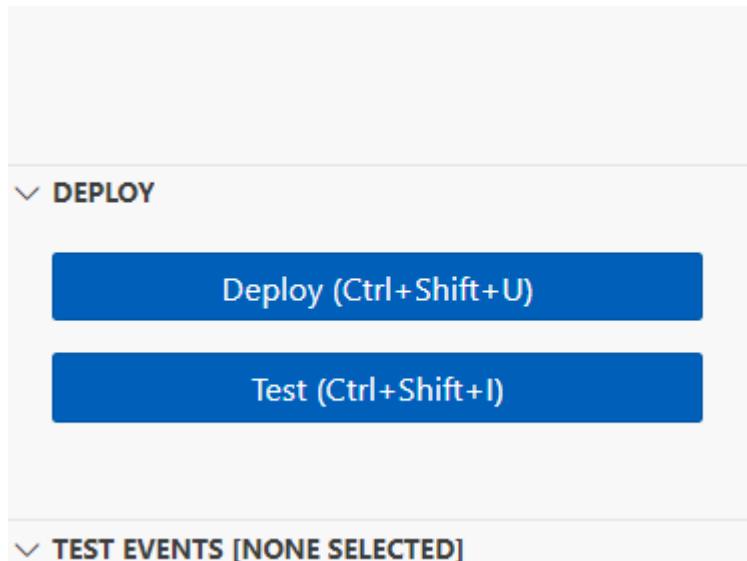
Multicloud With Devops by veera nareshit

The screenshot shows the AWS Lambda function editor interface. On the left, the 'EXPLORER' sidebar lists files: 'LAMBDA-API' (index.html, lambda_function.py), 'DEPLOY [UNDEPLOYED CHANGES]' (with a note about undeployed changes), and 'TEST EVENTS [NONE SELECTED]'. The main area displays the content of 'success.html'.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Thank You</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-image: url('data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAChCAMAAAgSoNaAAABxVBMVEx///qQzX7lwAzsuxC');
            background-size: 99%;
            background-repeat: no-repeat;
        }
    </style>
</head>
<body>
</body>

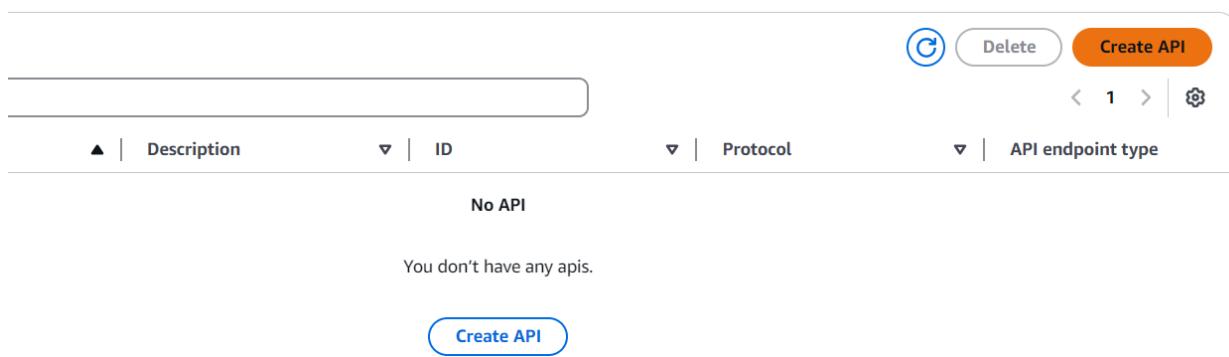
```

- Click on deploy



6
7
8
9
10
11
12
13
14
15
16
17

- Open api gateway page
- Click on create api



- In apis list build the rest api
- Click build on rest api

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:
Lambda, HTTP, AWS Services

[Import](#) [Build](#)

- Give the **api name**

Create REST API [Info](#)

API details

<input checked="" type="radio"/> New API Create a new REST API.	<input type="radio"/> Clone existing API Create a copy of an API in this AWS account.
<input type="radio"/> Import API Import an API from an OpenAPI definition.	<input type="radio"/> Example API Learn about API Gateway with an example API.

API name

lambda

Description - optional

(Empty text area)

- Api endpoint type as **regional**
- Click on create API

API name

lambda

Description - optional

(Empty text area)

API endpoint type

Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

▾

IP address type [Info](#)

Select the type of IP addresses that can invoke the default endpoint for your API.

<input checked="" type="radio"/> IPv4 Supports only edge-optimized and Regional API endpoint types.
<input type="radio"/> Dualstack Supports all API endpoint types.

[Cancel](#) [Create API](#)

- After that open your api
- Click on **create method**

Methods (0)

[Delete](#)

[Create method](#)

Method type



Integration type



Authorization



API key



No methods

No methods defined.

- Method type select **GET**
- Turn on the lambda proxy integration

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 m
- 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.

- Select your **lambda function**
- Click on create method

Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1

Q arn:aws:lambda:us-east-1:637423357373:function:lambda-api X

Grant API Gateway permission
When you save your changes, API Gateway automatically grants permissions to your Lambda function. This allows API Gateway to invoke your Lambda function.

Integration timeout | Info
By default, you can enter an integration timeout value greater than 29,000 ms.

Method request settings

URL query string parameters

HTTP request headers

Request body

arn:aws:lambda:us-east-1:637423357373:function:lambda-api
arn:aws:lambda:us-east-1:637423357373:function:cst-1449-a79d472a600d90b90a5fe44dd8-InitFunction-dFOxK2GUk6IE
arn:aws:lambda:us-east-1:637423357373:function:test
arn:aws:lambda:us-east-1:637423357373:function:test

timeout to greater than 29,000 ms

Cancel Create method

- One method is created we need to create one more method
- Click on the **slash** to create one more method

/

■ Click on **create method**

Methods (1)		Delete	Create method
Method type	Integration type	Authorization	API key
GET	Lambda	None	Not required

- Method type select **POST**

- Turn on the lambda proxy integration

Create method

Method details

Method type: POST

Integration type:

- Lambda function
Integrate your API with a Lambda function.

- HTTP
Integrate with an existing HTTP endpoint.

- Mc
Ge

- 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.

- Select your lambda function
- Click on create method

Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.
us-east-1

Grant API Gateway permission to invoke your Lambda function
When you save your changes, API Gateway updates your Lambda function's resource-based policy to allow this API to invoke it.

Integration timeout Info
By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms.
29000

► Method request settings

► URL query string parameters

► HTTP request headers

► Request body

- Click on Deploy API

API actions ▾ **Deploy API**

Update documentation **Delete**

- Select the stage as new stage
- Give the stage name as dev
- Click on deploy

Deploy API



Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. [Learn more](#)

Stage

New stage



Stage name

VSV

i A new stage will be created with the default settings. Edit your stage settings on the [Stage](#) page.

Deployment description

Cancel

Deploy

- Copy the stage invoke url

Default method-level caching
 Inactive

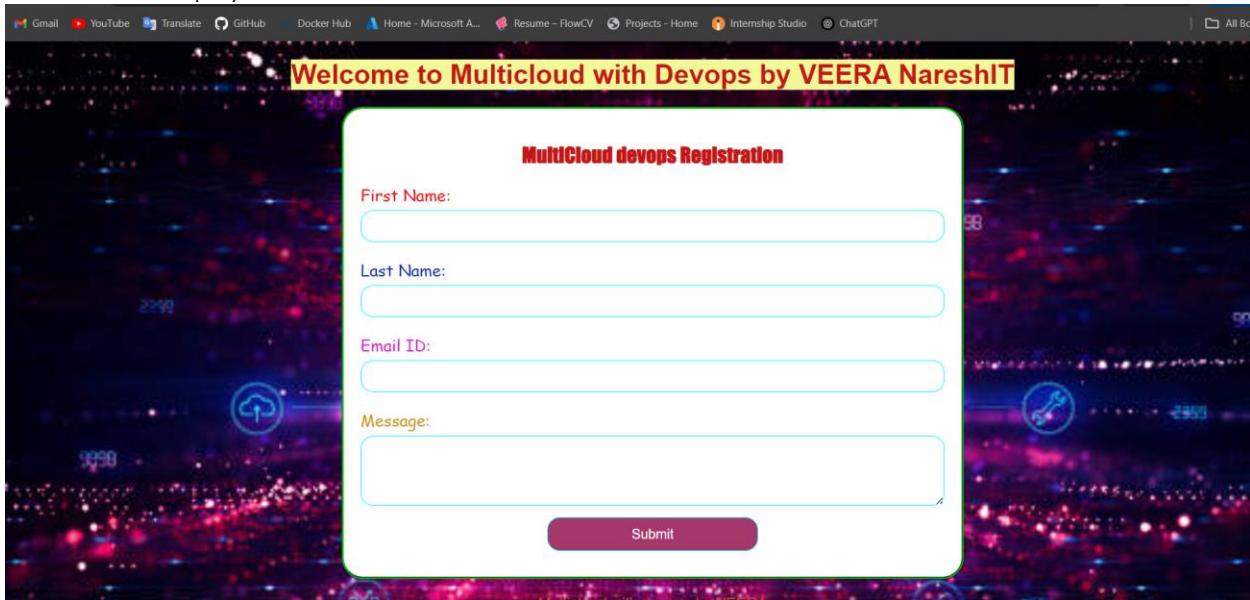
Copied

[Copy URL](https://5023zvxifj.execute-api.us-east-1.amazonaws.com/vsv)

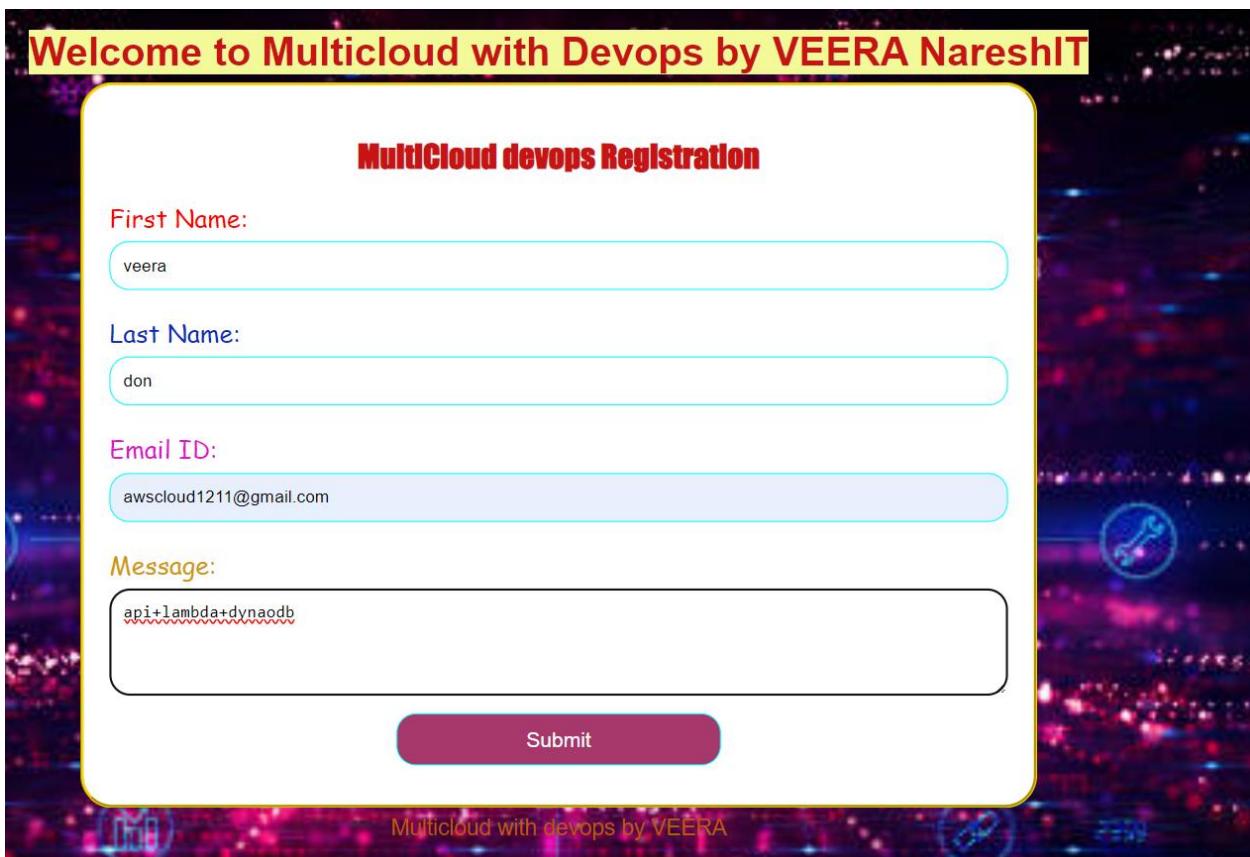
<https://5023zvxifj.execute-api.us-east-1.amazonaws.com/vsv>

Active deployment

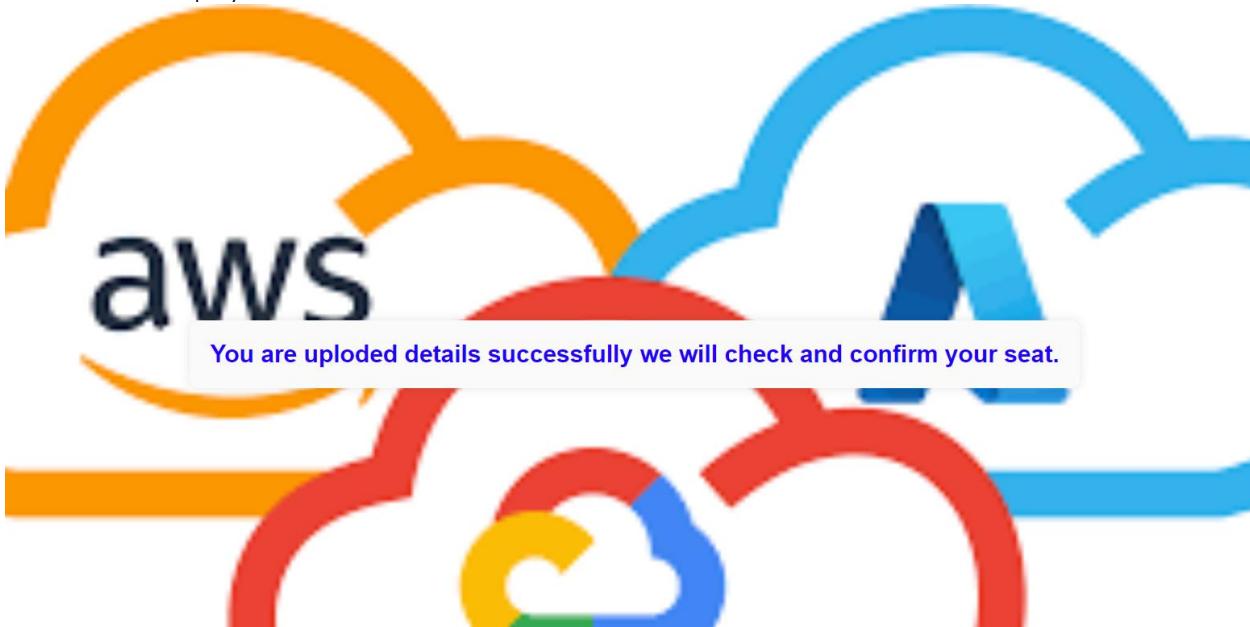
- Search it on web
- You will get the response



- Add the details
- Click on submit



- This is successful submit output



- Open your dynamodb table
- Click on explore items
- In right side corner you Are able to see the records

	email (String)	fname	lname	message
<input type="checkbox"/>	vsy	vsv	vardhan	vsvs
<input type="checkbox"/>	awscloud1211%40g...	veera	don	api%2Blambda%2Bdynamodb

THANK YOU

Benefits of This Integration

- **Serverless:** No server maintenance or provisioning.
- **Scalable:** Automatically handles traffic spikes.
- **Cost-effective:** Pay only for what you use.
- **Decoupled:** Each service handles its specific role.