

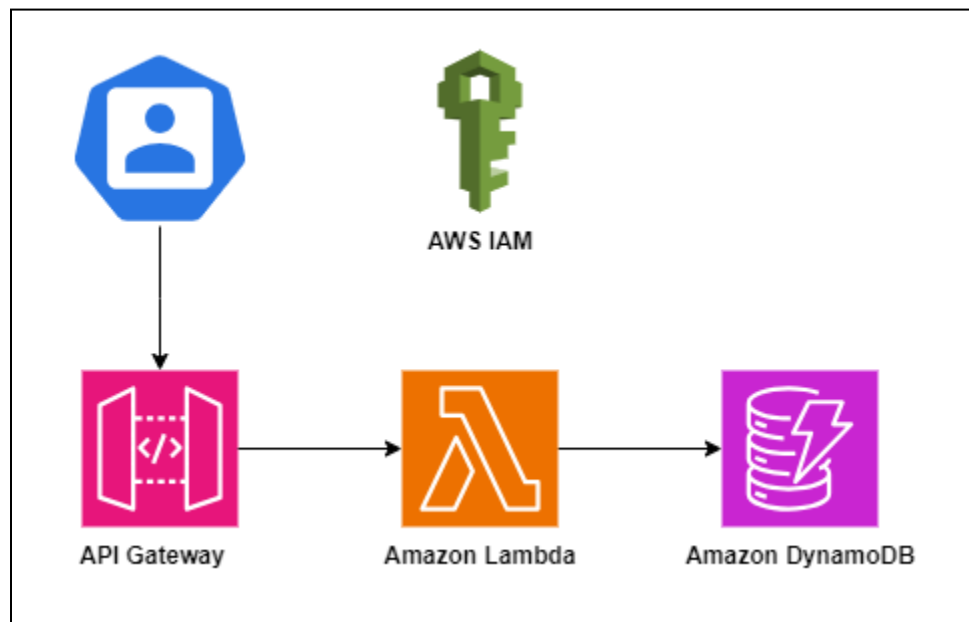
Cloud Project: Building a Serverless Registration WebApp with REST APIs

Table of content

Tech Stack.....	1
1. Create DynamoDB Table.....	1
2.. Create IAM Role for AWS Lambda.....	1
3. Create AWS Lambda Function.....	2
4. Write Lambda Function.....	2
5. Provide HTML, CSS and JavaScript file:.....	4
6. Create items returned in DynamoDB.....	6
7. Create API Gateway and enable CORS.....	7
8. Test the project.....	10

Project flow

This project demonstrates the creation of a scalable, serverless web application for user registration using AWS services. The application allows users to register by submitting a form, which sends data to the API Gateway. The API Gateway triggers a Lambda function that processes the data and stores it in DynamoDB.



Tech Stack

WebApp: HTML, CSS, JavaScript

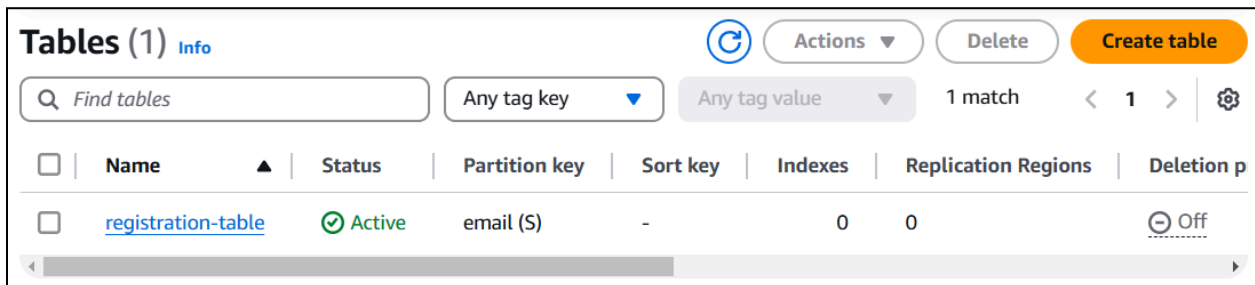
Services: Amazon DynamoDB, IAM, AWS Lambda, Amazon API Gateway

Lambda function written in Python

1. Create DynamoDB Table

Create a new table to start exploring DynamoDB

- Table name: registration-table
- Region: set it as ap-southeast-1
- Partition key: email (String type)
- Keep all table settings as default



The screenshot shows the AWS DynamoDB console interface. At the top, there's a header 'Tables (1)' with an 'Info' link. To the right are buttons for 'Actions', 'Delete', and 'Create table'. Below the header is a search bar labeled 'Find tables' and filters for 'Any tag key' and 'Any tag value'. It shows '1 match'. Below this is a table with columns: Name, Status, Partition key, Sort key, Indexes, Replication Regions, and Deletion protection. The table contains one entry: 'registration-table' with status 'Active', partition key 'email (S)', sort key '-', 0 indexes, 0 replication regions, and deletion protection 'Off'.

	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection
<input type="checkbox"/>	registration-table	Active	email (S)	-	0	0	Off

2.. Create IAM Role for AWS Lambda

Create a new role for Lambda.

Attach permissions:

1. AmazonDynamoDBFullAccess
2. AWSLambdaFullAccess

Name the role RegistrationFormRole and create it.

3. Create AWS Lambda Function

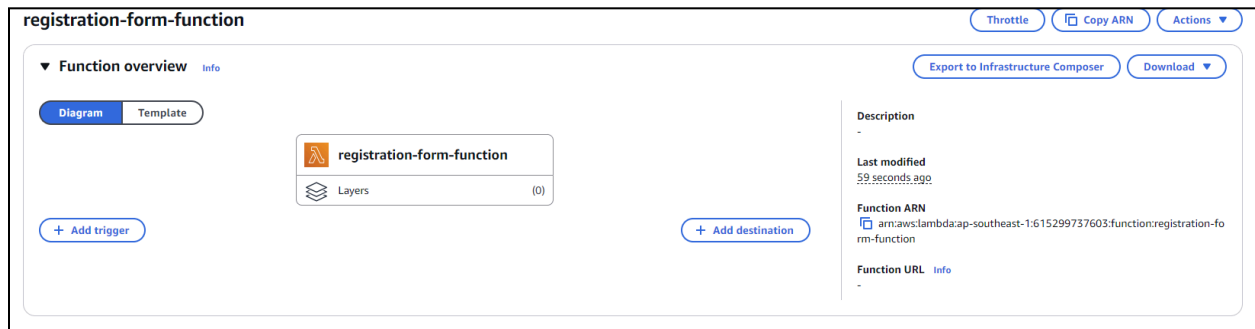
- Function Name: registration-form-function

- Runtime: Python 3.9
- Architecture: x86_64

For the execution role, use an existing role as RegistrationFormRole as created before.

Function ARN:

arn:aws:lambda:ap-southeast-1:615299737603:function:registration-form-function



4. Write Lambda Function

In Lambda function registration-form-function, write the lambda function. Then choose Deploy code.



import json

```

import boto3
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('registration-table')
def lambda_handler(event, context):
    response = table.put_item(
        Item={
            'email': event['email'],
            'name': event['name'],
            'phone': event['phone'],
            'password': event['password']
        }
    )
    return {
        'statusCode': 200,
        'headers': {
            'Content-Type': 'application/json',
            'Access-Control-Allow-Origin': '*'
        },
        'body': json.dumps({'message': 'Registration Successful'})
    }

```

5. Provide HTML, CSS and JavaScript file:

Item.html

```

<!DOCTYPE html>
<html>
<head>
  <title>Registration Form</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1>Registration Form</h1>
    <form>
      <label for="name">Name</label>
      <input type="text" id="name" name="name" required>

```

```

<label for="email">Email</label>
<input type="email" id="email" name="email" required>

<label for="phone">Phone</label>
<input type="tel" id="phone" name="phone" pattern="[0-9]{10}" required>

<label for="password">Password</label>
<input type="password" id="password" name="password" required>

<input type="submit" value="Submit" onclick="submitForm()">
</form>
</div>
<script src="script.js"></script>
</body>
</html>

```

script.js

```

function submitForm() {
  event.preventDefault();

  // Get form data
  const name = document.getElementById('name').value;
  const email = document.getElementById('email').value;
  const phone = document.getElementById('phone').value;
  const password = document.getElementById('password').value;

  // Create request object
  const xhr = new XMLHttpRequest();

  // Set up request
  xhr.open('POST', '<invokeurl>', true);
  xhr.setRequestHeader('Content-Type', 'application/json');

  // Set up response handler
  xhr.onreadystatechange = function () {
    if (xhr.readyState === XMLHttpRequest.DONE) {

```

```

        if (xhr.status === 200) {
            alert('Registration successful!');
            document.getElementById('name').value = "";
            document.getElementById('email').value = "";
            document.getElementById('phone').value = "";
            document.getElementById('password').value = "";
        } else {
            alert('Registration failed: ' + xhr.responseText);
        }
    }
};

// Send request
xhr.send(JSON.stringify({
    name: name,
    email: email,
    phone: phone,
    password: password
})));
}

```

style.css

```

.container {
    max-width: 400px;
    margin: auto;
    padding: 10px;
}

form {
    display: flex;
    flex-direction: column;
}

label {
    margin-top: 10px;
}

```

```
input[type="submit"] {  
  margin-top: 20px;  
}
```

6. Create items returned in DynamoDB

In DynamoDB, edit an item in the table registration-table we created before. We will create 4 attributes for input as email (default value 1), name (default as Empty), phone (default as Empty), password (default as Empty).

Create itemFormJSON view

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

AttributesAdd new attribute

Attribute name	Value	Type	
email - Partition key	1	String	
name	Empty value	String	Remove
phone	Empty value	String	Remove
password	Empty value	String	Remove

CancelCreate item

After create item, it will show format of items returned in format as:

Items returned (1)ActionsCreate item

< 1 > Settings Fullscreen

	email (String)	name	password	phone
<input type="checkbox"/>	1	<empty>	<empty>	<empty>

7. Create API Gateway and enable CORS

Next, reach out to API Gateway, choose to build 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

Create REST API named as registration-api

Create REST API

API details

☒ New API

Create a new REST API.

☐ Clone existing API

Create a copy of an API in this AWS account.

☐ Import API

Import an API from an OpenAPI definition.

☐ Example API

Learn about API Gateway with an example API.

API name

registration-api

Description - optional

registration-api

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.

Regional

Cancel

Create API

Create resource with resource name as register.

Create resource

Resource details

☐ Proxy resource [Info](#)

Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path

/

Resource name

register

☒ CORS (Cross Origin Resource Sharing) [Info](#)

Create an OPTIONS method that allows all origins, all methods, and several common headers.

Cancel

Create resource

We will enable CORS for the resource register. Choose resource /register and choose Enable CORS.

Resources

API actions
Deploy API

Create resource

/
/register
OPTIONS
POST

Resource details
Delete
Update documentation
Enable CORS

Path
/register
Resource ID
botndn

Methods (2)
Delete
Create method

	Method type	Integration type	Authorization	API key
<input type="radio"/>	OPTIONS	Mock	None	Not required
<input type="radio"/>	POST	Lambda	None	Not required

In the Enable CORS console, select two Access-Control-Allow-Method as OPTIONS and POST.

Enable CORS

CORS settings
Info

To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for your API. When you save your configuration, API Gateway replaces any existing CORS settings with your new configuration.

Gateway responses

API Gateway will configure CORS for the selected gateway responses.

☐ Default 4XX
☐ Default 5XX

Access-Control-Allow-Methods

☒ OPTIONS
☒ POST

Access-Control-Allow-Headers

API Gateway will configure CORS for the selected gateway responses.

Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token

Access-Control-Allow-Origin

Enter an origin that can access the resource. Use a wildcard '*' to allow any origin to access the resource.

*

▶ Additional settings

Next, create new stage to deploy API. Name the stage as testing.

Stages

testing

/

/register

OPTIONS

POST

Stage actions

Create stage

Stage details

Info

Edit

Stage name

testing

Rate

10000

Cache cluster

Inactive

Burst

5000

Default method-level caching

Inactive

Web ACL

-

Client certificate

-

Invoke URL

https://vvn2p560fl.execute-api.ap-southeast-1.amazonaws.com/testing

It will then genate an Invoke URL

<https://vvn2p560fl.execute-api.ap-southeast-1.amazonaws.com/testing>

We will then include it in **script.js** so that when we post a request with attributes, it will update records into DynamoDB table.

```
// Set up request
xhr.open('POST', 'https://vvn2p560fl.execute-api.ap-southeast-1.amazonaws.com/testing/register', true);
xhr.setRequestHeader('Content-Type', 'application/json');
```

8. Test the project

We will then access to index.html. We will input information so that it will register into DynamoDB table. Then we click submit. Once it get successful, it will shows alert as Registration successful.

Registration Form

Name

peter

Email

peter@gmail.com

Phone

0477657890

Password

Submit

Then check on registration table in DynamoDB, we will see there is record of the new input. Meaning that we successfully retrieve record from registration form successfully.

registration-table

Autopreview

View table details

► Scan or query items

Expand to query or scan items.

Items returned (2)



Actions ▼

Create item

< 1 > ⚙️ 🔗

<input type="checkbox"/>	email (String) ▼	name ▼	password ▼	phone ▼
<input type="checkbox"/>	1	<empty>	<empty>	<empty>
<input type="checkbox"/>	peter@gmail.com	peter	123456789	0477657890