

Name: Syam Krishna Reddy P

Date: 31st oct, 2025

Emp.id: ENC/16801

Track: Java

Question:

Create a serverless web application that does CRUD operation on data through an HTML form hosted on EC2, processes it via Lambda functions, and stores it in DynamoDB.

Architecture Components

1. **EC2 Instance:** Hosts a static HTML form
2. **API Gateway:** REST API endpoint
3. **Lambda Functions:** Process form data
4. **DynamoDB:** Stores submitted form data
5. **IAM Roles:** Secure permissions between services

STEP 1: Create DynamoDB Table

1. Open the DynamoDB console

- Go to AWS Console → DynamoDB → Tables → Create table

2. Table details

- Table name: UserSubmissions
- Partition key: submissionId (String)
- Leave Sort key empty
- Choose Provisioned or On-demand capacity (On-demand is fine for simplicity)

3. Click Create table

The screenshot shows the AWS DynamoDB console with the 'UserSubmissions' table selected. The left sidebar shows navigation options like Dashboard, Tables, and DAX. The main area displays the 'UserSubmissions' table details under the 'Settings' tab. Key information includes:

- General information:** Partition key (submissionId), Capacity mode (On-demand), Item count (0), Average item size (0 bytes).
- Sort key:** -
- Table status:** Active
- Point-in-time recovery (PITR):** Off
- Table size:** 0 bytes
- Resource-based policy:** Not active

At the bottom, there are links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

STEP 2: Create IAM Role for Lambda

1. Go to IAM → Roles → Create role

- Select AWS Service
- Use case: Lambda
- Click Next

2. Attach permissions

- AmazonDynamoDBFullAccess (for simplicity — can later restrict to the table)

The screenshot shows the AWS IAM Roles page. A role named 'LambdaDynamoDBRole' is selected. The 'Permissions' tab is active, showing one managed policy attached: 'AmazonDynamoDBFullAccess'. Other tabs include 'Trust relationships', 'Tags', 'Last Accessed', and 'Revoke sessions'. The ARN of the role is listed as arn:aws:iam::187681507880:role/LambdaDynamoDBRole. The maximum session duration is set to 1 hour. The last activity was 22 minutes ago.

STEP 3: Create Lambda Functions

You'll create **two** functions:

- ◆ A. Submission Lambda (POST /submit)
 1. Go to AWS Lambda → Create function
 - a. Name: **SubmissionFunction**
 - b. Runtime: Python 3.9
 - c. Execution role: Use existing role → LambdaDynamoDBRole
 2. Click Create function

The screenshot shows the AWS Lambda Functions page. A function named 'SubmissionFunction' is selected. The 'Test' tab is active. The function overview shows it is triggered by an API Gateway. The function ARN is arn:aws:lambda:ap-south-1:187681507880:function:SubmissionFunction. The function URL is also listed.

◆ B. Query Lambda (GET /submissions)

1. Create another Lambda:
 - a. Name: QueryFunction
 - b. Runtime: Python 3.9
 - c. Role: Use existing role → LambdaDynamoDBRole

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and account information (Account ID: 1876-8150-7880, Region: Asia Pacific (Mumbai), User: Syam Krishna Reddy Pulagam). Below the navigation is a breadcrumb trail: Lambda > Functions > QueryFunction. The main area is titled 'QueryFunction' and contains a 'Function overview' section. It shows the function name 'QueryFunction' with a 'Layers' count of 0. There are tabs for 'Diagram' (selected) and 'Template'. A diagram shows the function connected to an 'API Gateway' trigger. Buttons for '+ Add destination' and '+ Add trigger' are visible. To the right, there are sections for 'Description' (empty), 'Last modified' (3 hours ago), 'Function ARN' (arn:aws:lambda:ap-south-1:187681507880:function:QueryFunction), and 'Function URL' (empty). Below the overview are tabs for 'Code' (selected), 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. On the far right, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'.

STEP 4: Create API Gateway

1. Go to Amazon API Gateway → Create API
 - a. Choose REST API → Build
 - b. Name: UserSubmissionAPI
 - c. Endpoint Type: Regional
 - d. Click Create API

The screenshot shows the AWS API Gateway interface. On the left, there's a sidebar with 'APIs' and 'API: UserSubmissionAPI' sections. The main area is titled 'Resources' and shows a tree view of resources under the path '/'. It lists 'OPTIONS' and 'POST' methods under '/submissions'. To the right, 'Resource details' show the path '/' and a resource ID 'd3vbwqfz7c'. Below that, the 'Methods (1)' section shows an 'OPTIONS' method with 'Mock' integration type, 'None' authorization, and 'Not required' API key. Buttons for 'Update documentation' and 'Enable CORS' are also present.

4.1 Create Resource /submit

1. Click Actions → Create Resource
 - a. Resource name: submit
 - b. Resource path: /submit
 - c. Click Create Resource
2. With /submit selected → Click Create Method → POST
3. Integration type: Lambda Function
 - a. Select your region
 - b. Lambda Function: SubmissionFunction
 - c. Save → Allow permission prompt

This screenshot shows the same API Gateway interface after the '/submit' resource was created. The 'Resources' tree now includes '/submissions' under '/'. The 'Methods (2)' section now shows both 'GET' and 'OPTIONS' methods for the '/submissions' resource. The 'Integration type' for both methods is set to 'Lambda', and 'Authorization' and 'API key' are both 'None'.

4.2 Create Resource /submissions

1. Create another resource /submissions
2. Add a **GET** method
 - a. Integration: **Lambda Function**
 - b. Lambda: **QueryFunction**
 - c. Save → Allow permission prompt

The screenshot shows the AWS API Gateway interface. On the left, the navigation pane is visible with sections like API Gateway, APIs, and the current API: UserSubmissionAPI. Under the API section, there are links for Resources, Stages, Authorizers, etc. The main content area is titled 'Resources' and shows a tree structure for resources. A 'Create resource' button is at the top. Below it, under the root resource, there are two entries: '/OPTIONS' and '/submissions'. Under '/submissions', there are two methods: 'GET' and 'OPTIONS'. The 'OPTIONS' method is currently selected. To the right of the tree, there is a 'Resource details' panel with the path '/submit' and a resource ID '888ad4'. Below that is a 'Methods (2)' table:

| Method type | Integration type | Authorization | API key |
|-------------|------------------|---------------|--------------|
| OPTIONS | Mock | None | Not required |
| POST | Lambda | None | Not required |

4.3 Enable CORS

1. For each resource (/submit and /submissions):
 - a. Click **Actions** → **Enable CORS**
 - b. Accept defaults and confirm

4.4 Deploy the API

1. Click **Actions** → **Deploy API**
 - a. Deployment stage: **[New Stage]**
 - b. Stage name: prod
2. Click **Deploy**

Now you'll get URLs like:

POST <https://cssjz6x5ok.execute-api.ap-south-1.amazonaws.com/prod/submit>
GET <https://cssjz6x5ok.execute-api.ap-south-1.amazonaws.com/prod/submissions>

STEP 5: Launch EC2 Instance and Host HTML

5.1 Launch EC2

1. Go to **EC2 → Launch Instance**
 - a. Name: WebFormServer
 - b. AMI: **Amazon Linux 2**
 - c. Instance type: **t2.micro**
 - d. Key pair: (Create one if needed)
 - e. Security Group: Allow **HTTP (port 80)** and **SSH (port 22)**
2. Click **Launch Instance**

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with 'EC2' selected. The main area displays an 'Instance summary' for a new instance named 'i-00213c34f41dc096f' (WebFormServer). The summary includes:

- Connect**, **Instance state** (Running), **Actions**
- Updated less than a minute ago.
- Instance ID**: i-00213c34f41dc096f
- IPv6 address**: -
- Hostname type**: IP name: ip-172-31-2-163.ap-south-1.compute.internal
- Answer private resource DNS name**: IPv4 (A)
- Auto-assigned IP address**: 13.126.5.48 [Public IP]
- Public IPv4 address**: 13.126.5.48 | [open address](#)
- Private IP DNS name (IPv4 only)**: ip-172-31-2-163.ap-south-1.compute.internal
- Instance state**: Running
- Instance type**: t3.micro
- VPC ID**: vpc-03ba0d20965ab76c7
- Private IPv4 addresses**: 172.31.2.163
- Public DNS**: ec2-13-126-5-48.ap-south-1.compute.amazonaws.com | [open address](#)
- Elastic IP addresses**: -
- AWS Compute Optimizer finding**: Opt-in to AWS Compute Optimizer for recommendations.

At the bottom, there are links for CloudShell, Feedback, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

5.2 Connect via SSH and install apache and create html form

```
C:\Users\SymKrishnaReddy>ssh -i "C:\Users\SymKrishnaReddy\Downloads\New_pair.pem" ec2-user@13.126.5.48
usage: ssh [-i key] [-l login_name] [-p port] [-t timeout] [-W widthxheight]
           [-b bind_interface] [-b bind_address] [-E log_file]
           [-c cipher_spec] [-D [bind_address:]port] [-F configfiles] [-I pkcs11] [-I identity_file]
           [-e escape_char] [-f configfile] [-l login_name] [-m mac_spec]
           [-j destination] [-o address] [-p login_name] [-w local_tun[remote_tun]]
           [-O ctl_cmd] [-o option] [-p tag] [-p port] [-Q query_option]
           [-R address] [-S ctl_path] [-W host:port] [-w local_tun[remote_tun]]
           destination [command [argument ...]]]

C:\Users\SymKrishnaReddy>ssh -i "C:\Users\SymKrishnaReddy\Downloads\New_pair.pem" ec2-user@13.126.5.48
The authenticity of host '13.126.5.48' (13.126.5.48) can't be established.
ED25519 key fingerprint is SHA256:E7bTX3Wn2TfesE8Xrpm8rVzNpQib980cUKEOMTIk3Cc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.126.5.48' (ED25519) to the list of known hosts.

      _#
     ~_\###_
    ~~ \###\
   ~~\###|
  ~~ \#/_ https://aws.amazon.com/linux/amazon-linux-2023
  ~~ \#/_/ \
  ~~~ \_/_ \
  ~~~ \_/_ \
  ~~~ \_/_ \
  [ec2-user@ip-172-31-2-163 ~]$ sudo yum update -y

Amazon Linux 2023 Kernel Livepatch repository                               212 kB/s |  28 kB   00:00
Last metadata expiration check: 0:00:01 ago on Fri Oct 31 06:45:47 2025.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-2-163 ~]$ sudo yum install -y httpd
Last metadata expiration check: 0:00:13 ago on Fri Oct 31 06:45:47 2025.
Dependencies resolved.

=====
          Repository          Size          Package          Architecture          Version
=====
httpd           x86_64       2.4.65-1.amzn2023.0.2    amazonlinux        47 k
Installing dependencies:
apr             x86_64       1.7.5-1.amzn2023.0.4    amazonlinux        129 k
apr-util        x86_64       1.6.3-1.amzn2023.0.2    amazonlinux        97 k
apr-util-lmdb   x86_64       1.6.3-1.amzn2023.0.2    amazonlinux        13 k
generic-logos-httd noarch      18.0.0-12.amzn2023.0.3  amazonlinux        19 k

[ec2-user@ip-172-31-2-163 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-2-163 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-2-163 ~]$ sudo nano /var/www/html/index.html
[ec2-user@ip-172-31-2-163 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-10-31 06:46:09 UTC; 2min 59s ago
       Docs: man:httpd.service(8)
 Main PID: 14975 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec: 0 B/sec"
      Tasks: 177 (limit: 1053)
     Memory: 13.3M
        CPU: 254ms
       CGroup: /system.slice/httpd.service
               ├─14975 /usr/sbin/httpd -DFOREGROUND
               ├─15081 /usr/sbin/httpd -DFOREGROUND
               ├─15098 /usr/sbin/httpd -DFOREGROUND
               ├─15106 /usr/sbin/httpd -DFOREGROUND
               └─15118 /usr/sbin/httpd -DFOREGROUND

Oct 31 06:46:09 ip-172-31-2-163.ap-south-1.compute.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Oct 31 06:46:09 ip-172-31-2-163.ap-south-1.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Oct 31 06:46:09 ip-172-31-2-163.ap-south-1.compute.internal httpd[14975]: Server configured, listening on: port 80
```

5.5 Test it!

1. Open browser → enter **EC2 Public IP**
2. Fill out the form → Submit
3. Go to **DynamoDB** → **Tables** → **UserSubmissions** → **Explore table items**
 - a. You'll see your submission!

User Submission Form

| | |
|---------|---------------|
| Name | Syam |
| Email | syam@test.com |
| Message | hi |

Submit

All Submissions

Refresh

| Submission ID | Name | Email | Message | Submission Date | Status |
|--------------------------------------|------|---------------|---------|----------------------------|----------|
| 1e2a87dc-8756-4c37-9e59-659514811b53 | Syam | syam@test.com | Hi | 2025-10-31T08:24:30.722610 | Received |
| 00cf8fcf-4dc5-4186-a015-73bdfb5d44f9 | John | john@test.com | Hello | 2025-10-31T07:18:29.634124 | Received |

Table: UserSubmissions - Items returned (2)



Actions ▾

Create item

Scan started on October 31, 2025, 14:28:06

< 1 > |

| <input type="checkbox"/> submissionId (String) | email | message | name | status |
|------------------------------------------------------|----------------|---------|------|----------|
| 1e2a87dc-8756-4c37-9e59-659514811b53 | syam@test.... | Hi | Syam | Received |
| 00cf8fcf-4dc5-4186-a015-73bdfb5d44f9 | john@test.c... | Hello | John | Received |