2. Creating a Serverless API

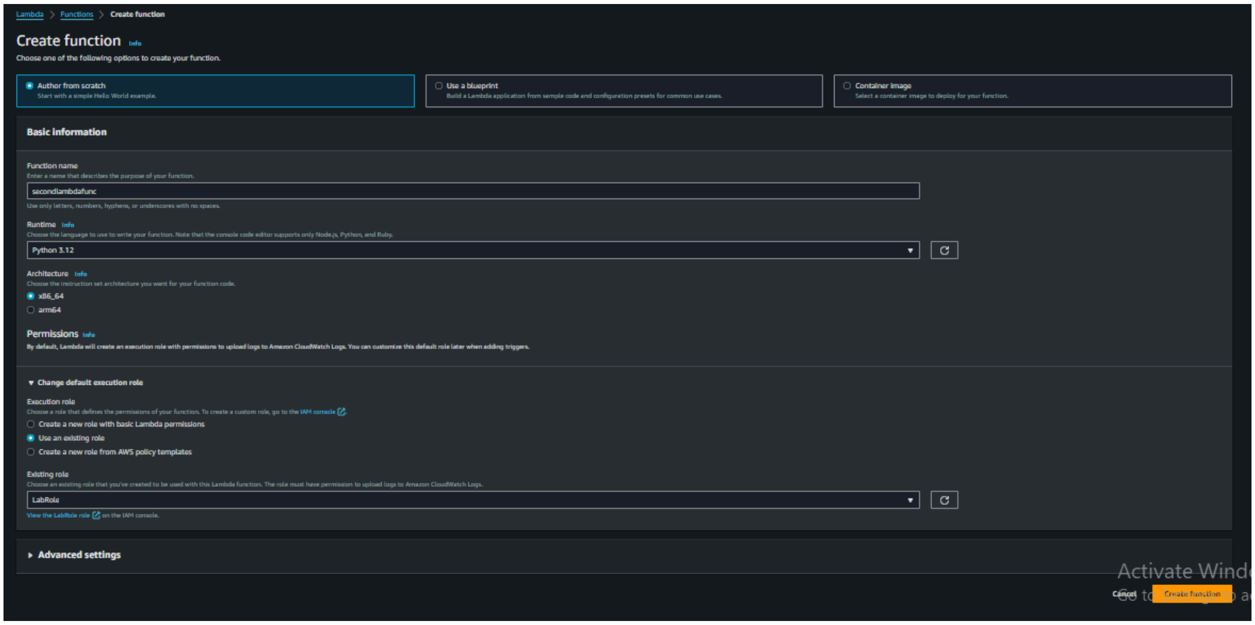
Objective: Develop a serverless API using AWS Lambda and API Gateway.

Approach:

- **Define API**: Design a simple RESTful API (e.g., for a todo list application).
- **Lambda Functions**: Create Lambda functions for each API method (GET, POST, PUT, DELETE).
- **API Gateway Setup**: Use API Gateway to set up the API endpoints, connecting each endpoint to the corresponding Lambda function.
- **Testing**: Test the API using tools like Postman or AWS API Gateway test functionality.

Goal: Gain hands-on experience in building and deploying a serverless API, understanding the integration between Lambda and API Gateway.

1. Created the lambda function and assigned the LabRole



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

○ Container image
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
secondlambdafunc
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 code editor supports only Node.js, Python, and Ruby.
Python 3.12

Architecture Info
Choose the instruction set architecture you want for your function code.
● x86_64
○ arm64

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.

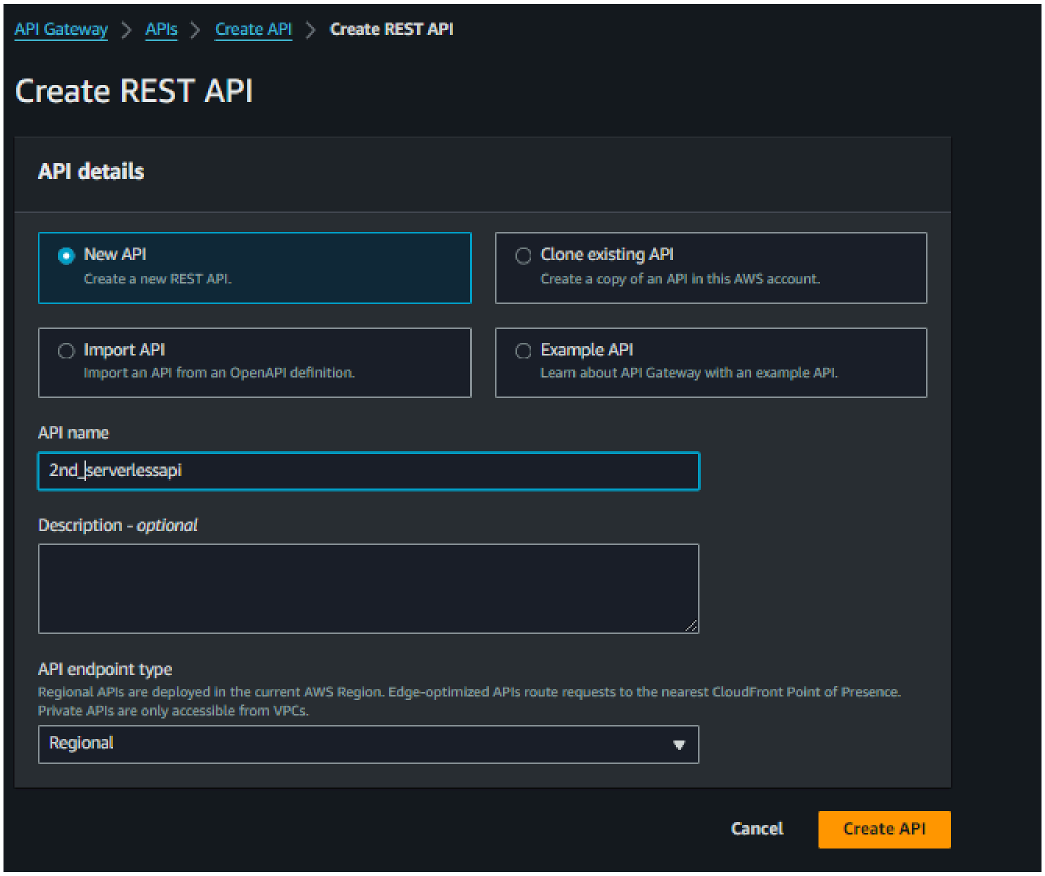
▼ Change default execution role

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.
LabRole
View the LabRole role on the IAM console.

▸ Advanced settings

2. Created the REST API



API Gateway > APIs > Create API > Create REST 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.

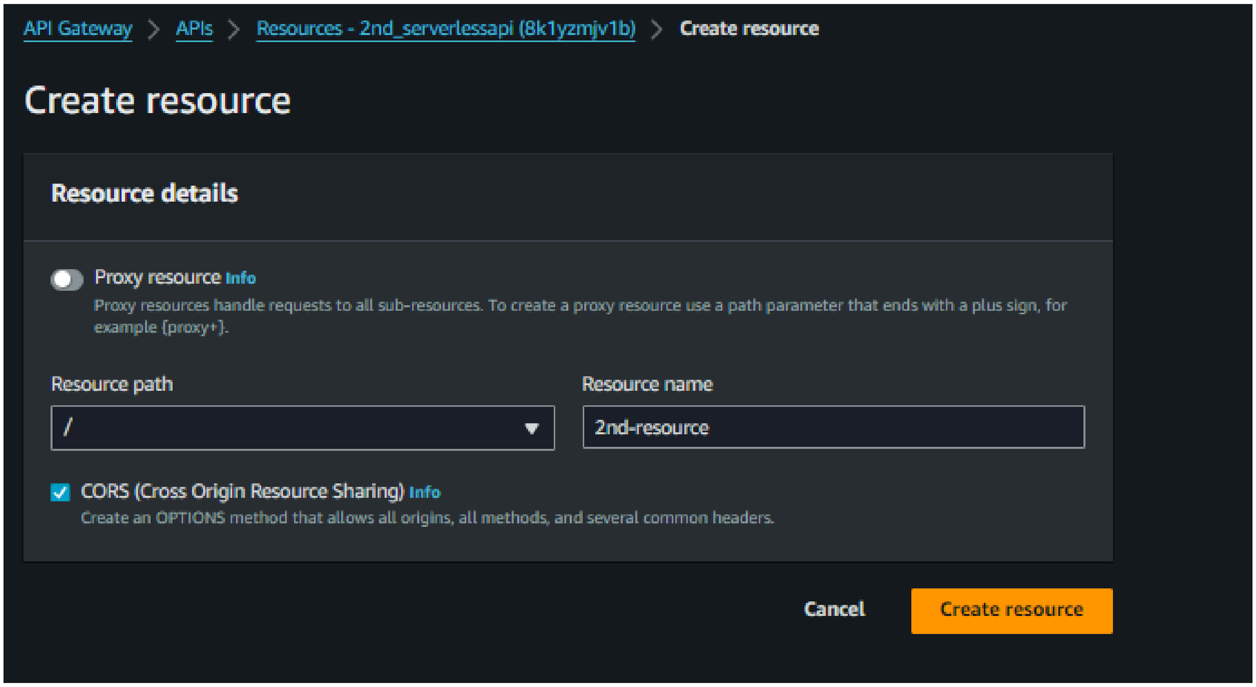
API name: 2nd_serverlessapi

Description - *optional*

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

3. Created the resources



API Gateway > APIs > Resources - 2nd_serverlessapi (8k1yzmjv1b) > Create resource

Create resource

Resource details

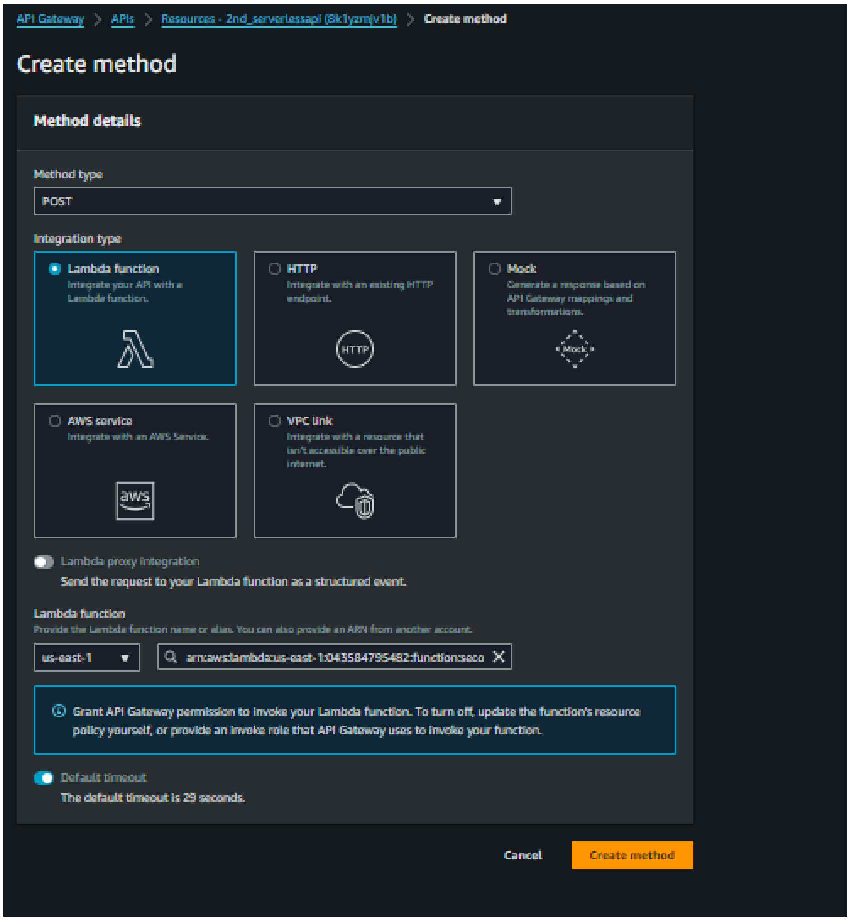
Proxy resource Info
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path: /

Resource name: 2nd-resource

☑ CORS (Cross Origin Resource Sharing) Info
Create an OPTIONS method that allows all origins, all methods, and several common headers.

4. Created method inside the resources



API Gateway > APIs > Resources - 2nd_serverlessapi (8k1yzmjv1b) > Create method

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.

○ 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.

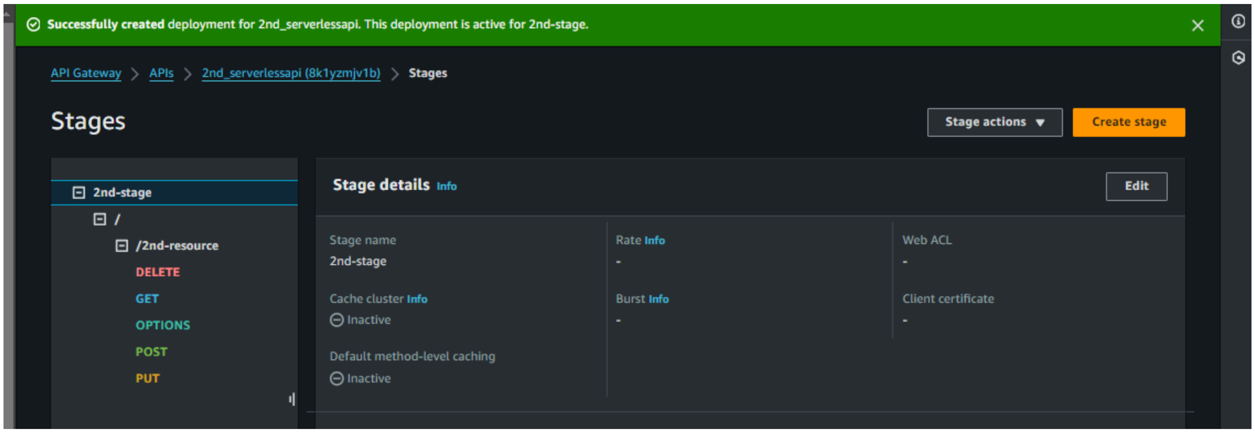
Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1 | arn:aws:lambda:us-east-1:043584795482:function:seco

Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

Default timeout
The default timeout is 29 seconds.

5. In similar way all four methods were created: PUT, POST, DELETE, GET



Successfully created deployment for 2nd_serverlessapi. This deployment is active for 2nd-stage.

API Gateway > APIs > 2nd_serverlessapi (8k1yzmjv1b) > Stages

Stages

- 2nd-stage
  - /
    - /2nd-resource
      - DELETE
      - GET
      - OPTIONS
      - POST
      - PUT

Stage details Info

Stage name: 2nd-stage

Rate Info: -

Web ACL: -

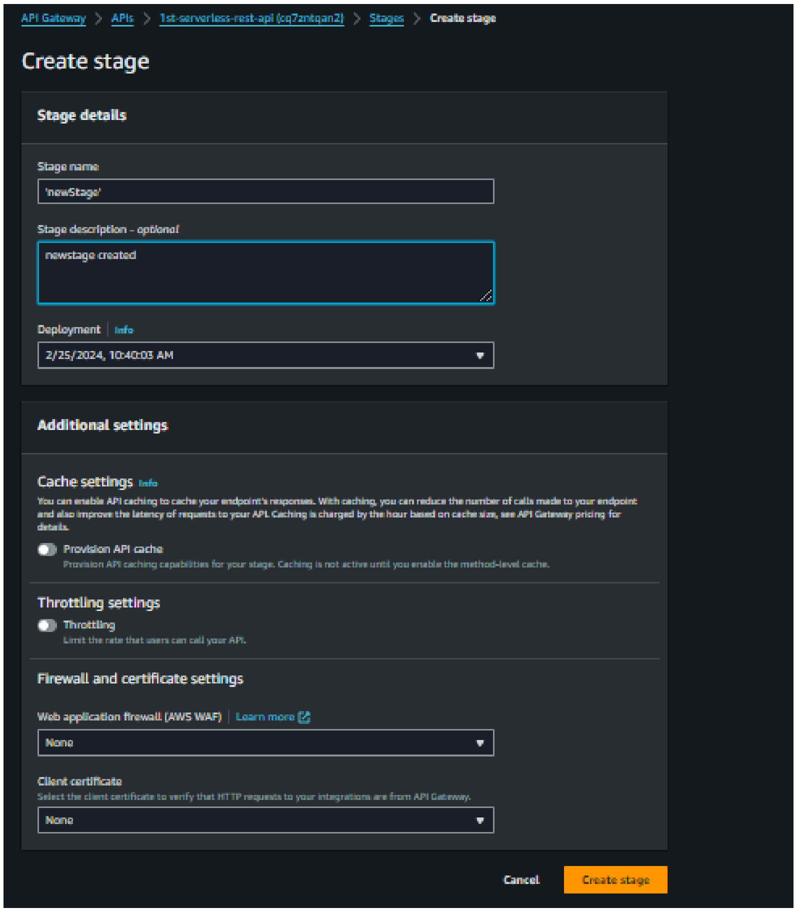
Cache cluster Info: Inactive

Burst Info: -

Client certificate: -

Default method-level caching: Inactive

6.Stage Created and deployed API


API Gateway > APIs > 1st-serverless-rest-api (cq7zntqan2) > Stages > Create stage

Create stage

Stage details

Stage name

'newStage'

Stage description - *optional*

newstage created

Deployment | Info

2/25/2024, 10:40:03 AM

Additional settings

Cache settings Info

You can enable API caching to cache your endpoint's responses. With caching, you can reduce the number of calls made to your endpoint and also improve the latency of requests to your API. Caching is charged by the hour based on cache size, see API Gateway pricing for details.

○ Provision API cache

Provision API caching capabilities for your stage. Caching is not active until you enable the method-level cache.

Throttling settings

○ Throttling

Limit the rate that users can call your API.

Firewall and certificate settings

Web application firewall (AWS WAF) | Learn more

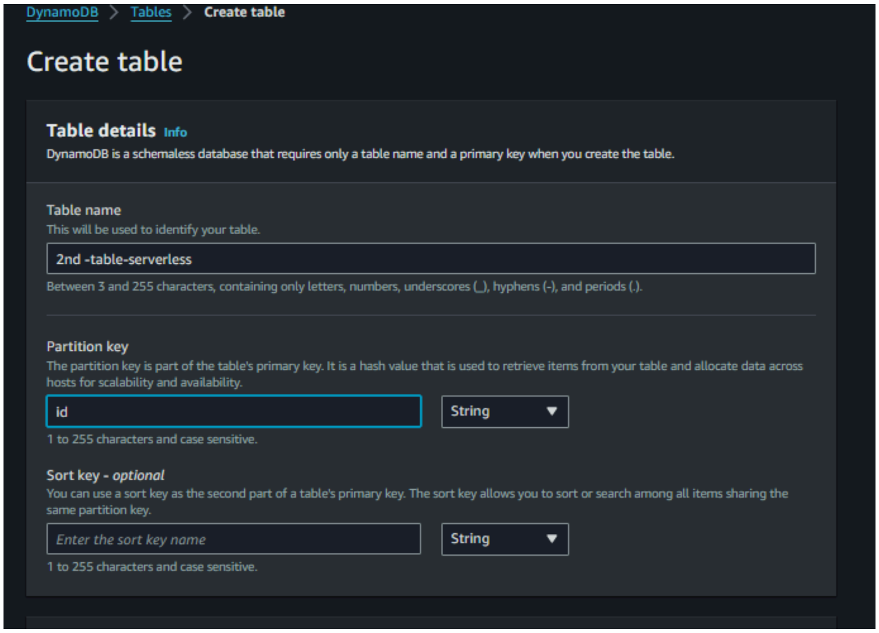
None

Client certificate

Select the client certificate to verify that HTTP requests to your integrations are from API Gateway.

None

6.DynamoDb table created:



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.

2nd -table-serverless

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.

id | String

1 to 255 characters and case sensitive.

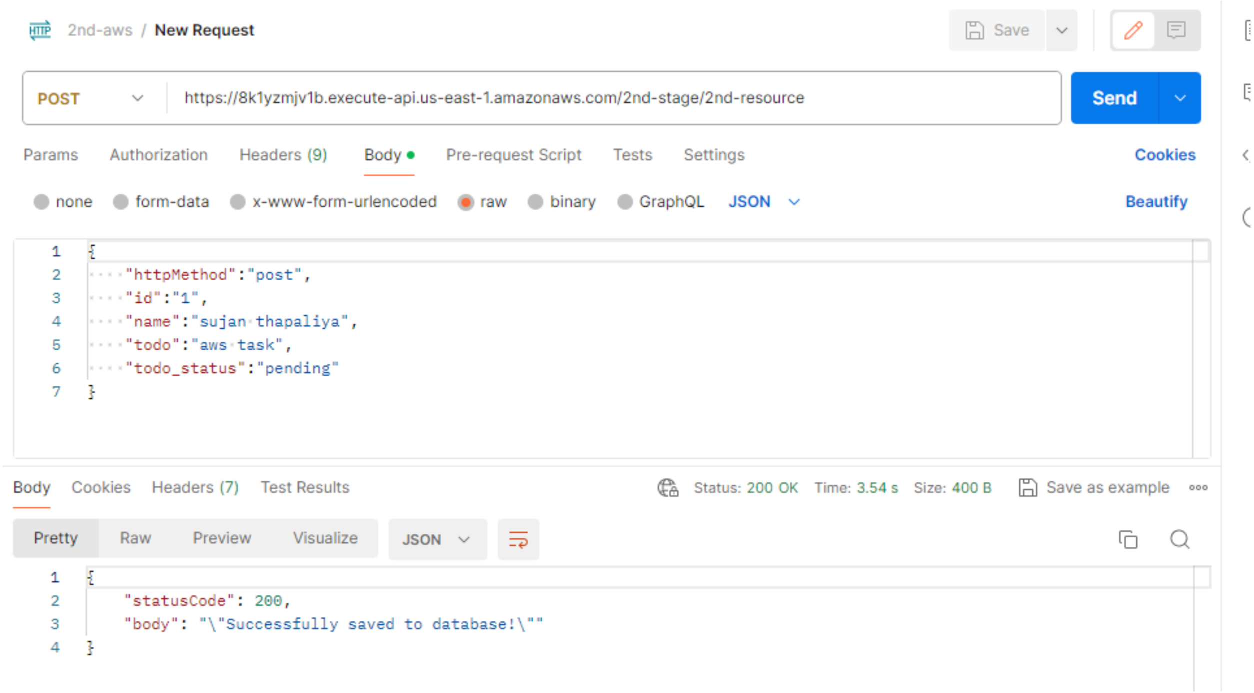
Sort key - *optional*

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.

Enter the sort key name | String

1 to 255 characters and case sensitive.

7.API Tested on POSTMAN



POST https://8k1yzmjv1b.execute-api.us-east-1.amazonaws.com/2nd-stage/2nd-resource

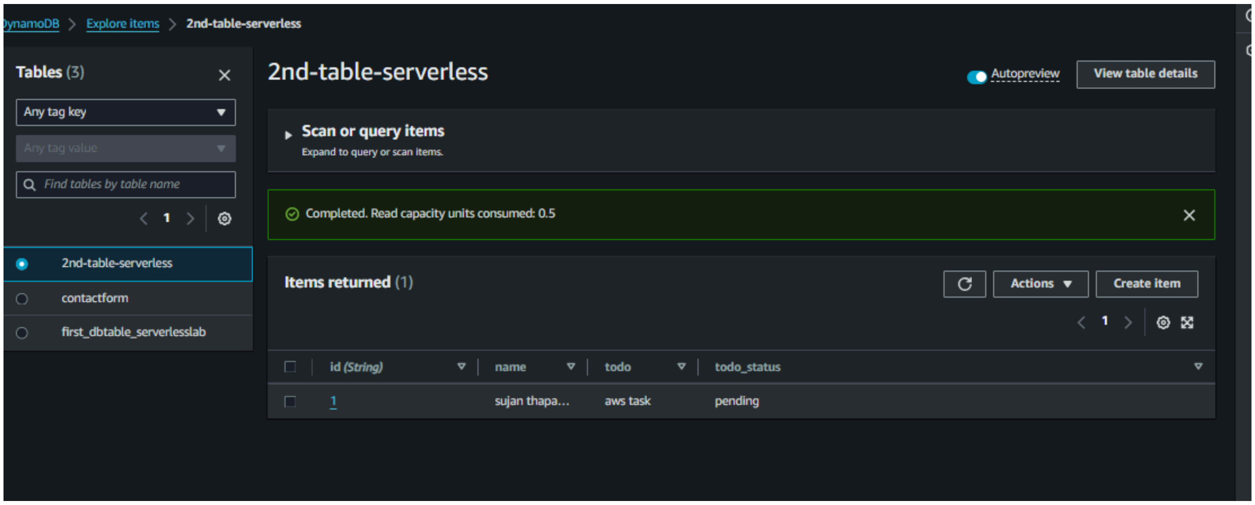
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "httpMethod":"post",
    "id":"1",
    "name":"sujan thapaliya",
    "todo":"aws task",
    "todo_status":"pending"
}
```

Status: 200 OK Time: 3.54 s Size: 400 B

JSON

```
{
    "statusCode": 200,
    "body": "\"Successfully saved to database!\""
}
```


DynamoDB > Explore items > 2nd-table-serverless
Tables (3)
Any tag key
Any tag value
Find tables by table name
Autopreview
View table details
Scan or query items
Expand to query or scan items.
Actions
Create item

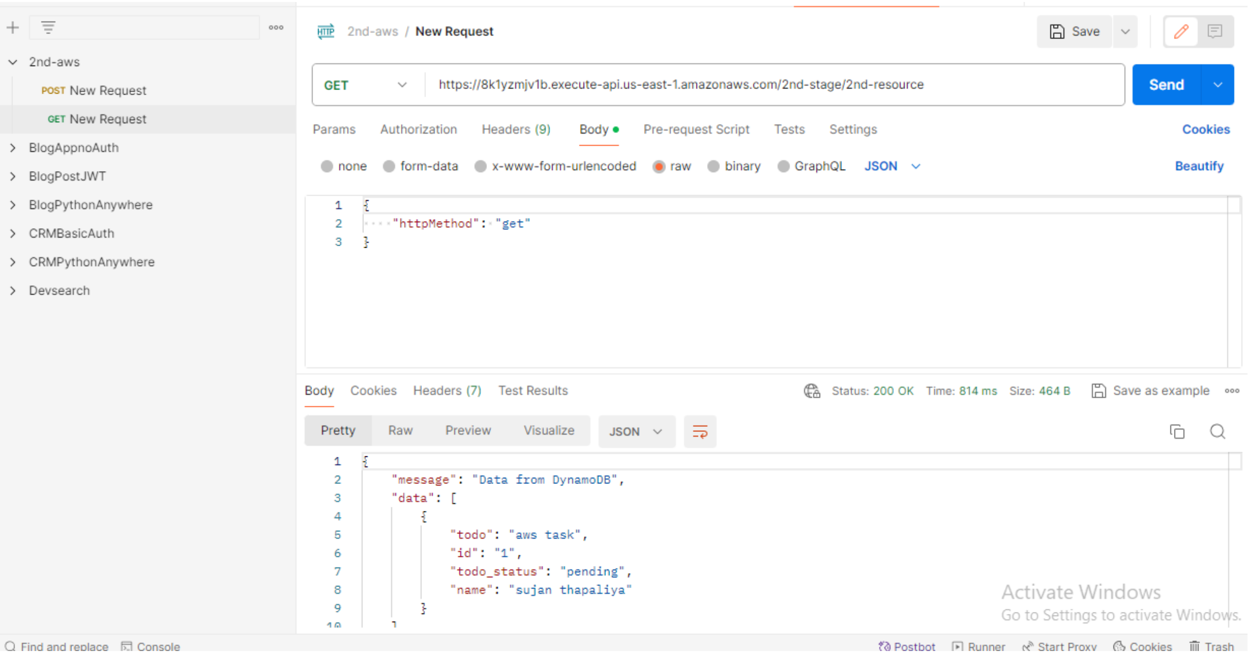
● 2nd-table-serverless
○ contactform
○ first_dbtable_serverlesslab

2nd-table-serverless

Completed. Read capacity units consumed: 0.5

Items returned (1)

	id (String)	name	todo	todo_status
□	1	sujan thapa...	aws task	pending



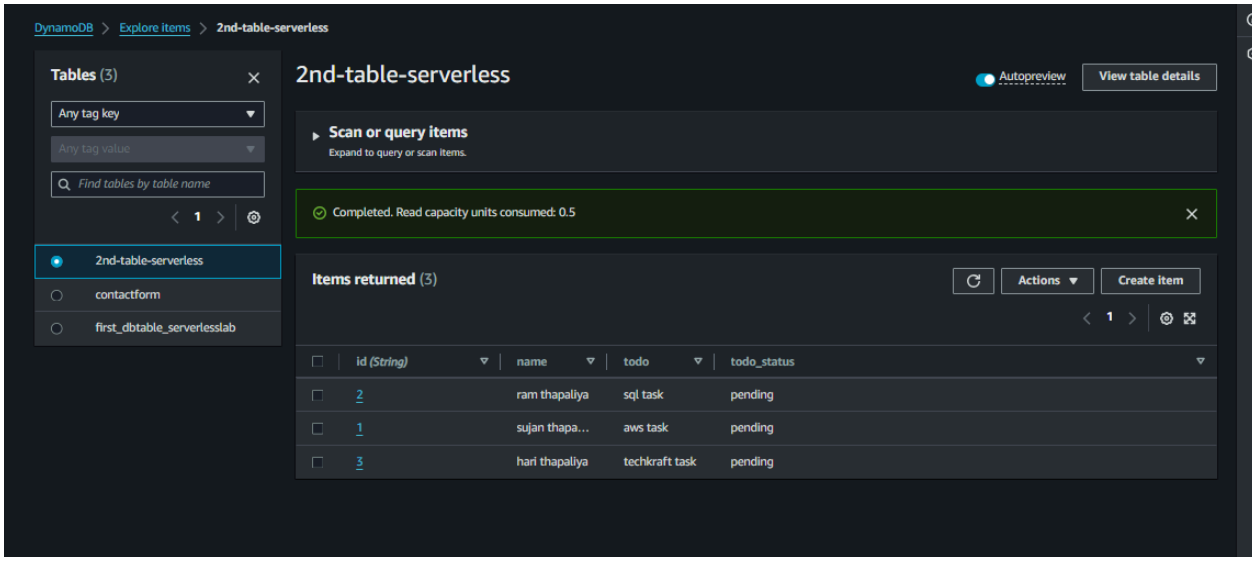
GET https://8k1yzmjv1b.execute-api.us-east-1.amazonaws.com/2nd-stage/2nd-resource

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "httpMethod": "get"
}
```

Status: 200 OK Time: 814 ms Size: 464 B

```
{
    "message": "Data from DynamoDB",
    "data": [
        {
            "todo": "aws task",
            "id": "1",
            "todo_status": "pending",
            "name": "sujan thapaliya"
        }
    ]
```



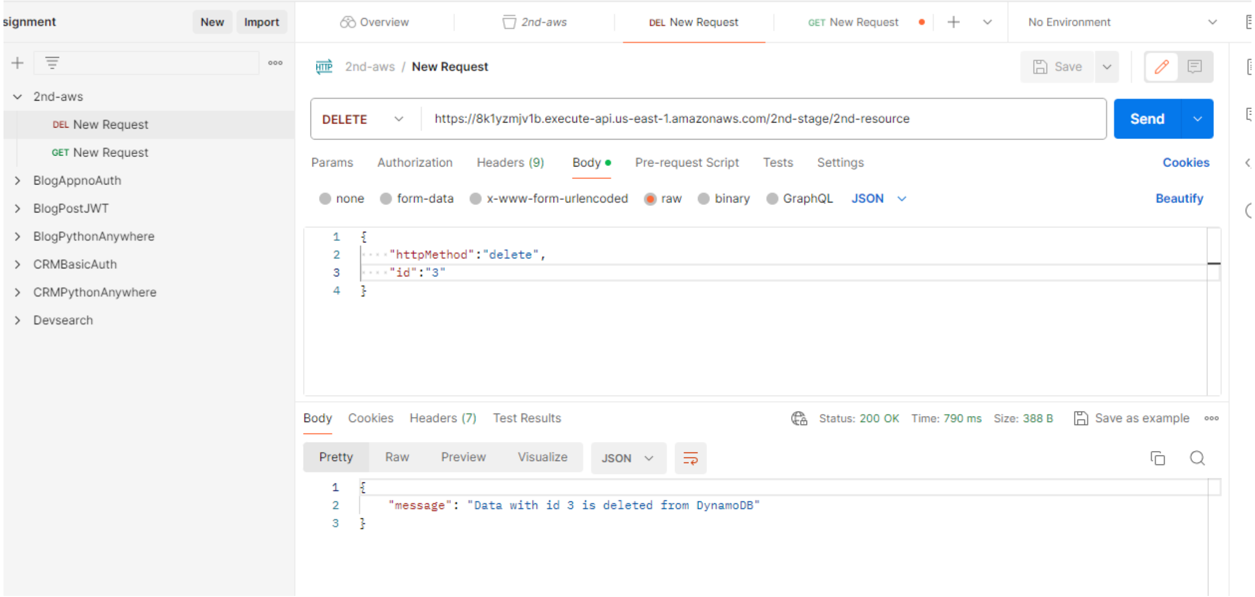
● 2nd-table-serverless
○ contactform
○ first_dbtable_serverlesslab

2nd-table-serverless

Completed. Read capacity units consumed: 0.5

Items returned (3)

□	id (String)	name	todo	todo_status
□	2	ram thapaliya	sql task	pending
□	1	sujan thapa...	aws task	pending
□	3	hari thapaliya	techkraft task	pending



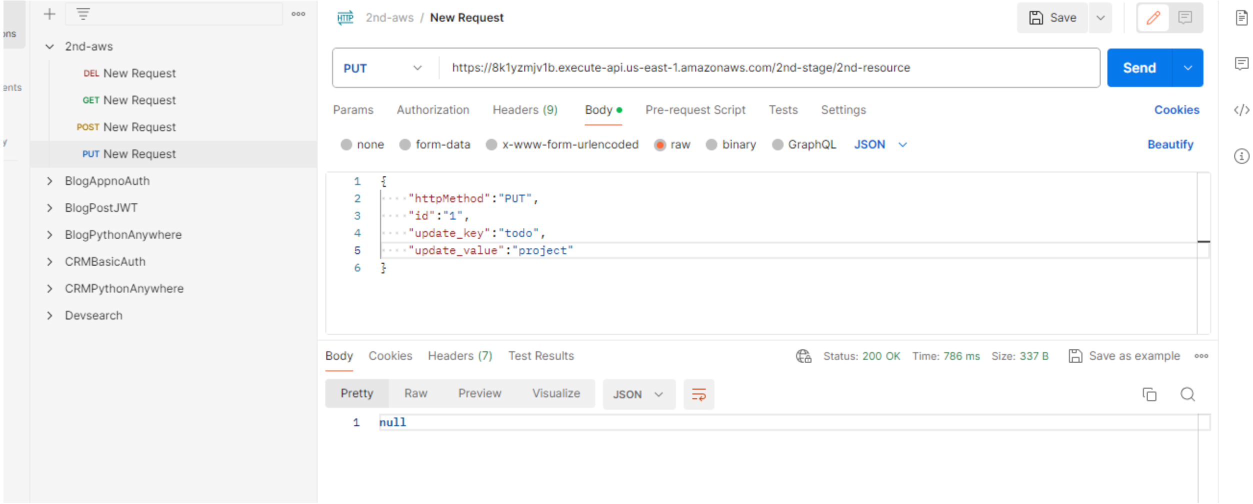
DELETE https://8k1yzmjv1b.execute-api.us-east-1.amazonaws.com/2nd-stage/2nd-resource

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "httpMethod":"delete",
    "id":"3"
}
```

Status: 200 OK Time: 790 ms Size: 388 B

```
{
    "message": "Data with id 3 is deleted from DynamoDB"
}
```



PUT https://8k1yzmjv1b.execute-api.us-east-1.amazonaws.com/2nd-stage/2nd-resource

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "httpMethod":"PUT",
    "id":"1",
    "update_key":"todo",
    "update_value":"project"
}
```

Status: 200 OK Time: 786 ms Size: 337 B

```
null
```