Serverless Labs

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).

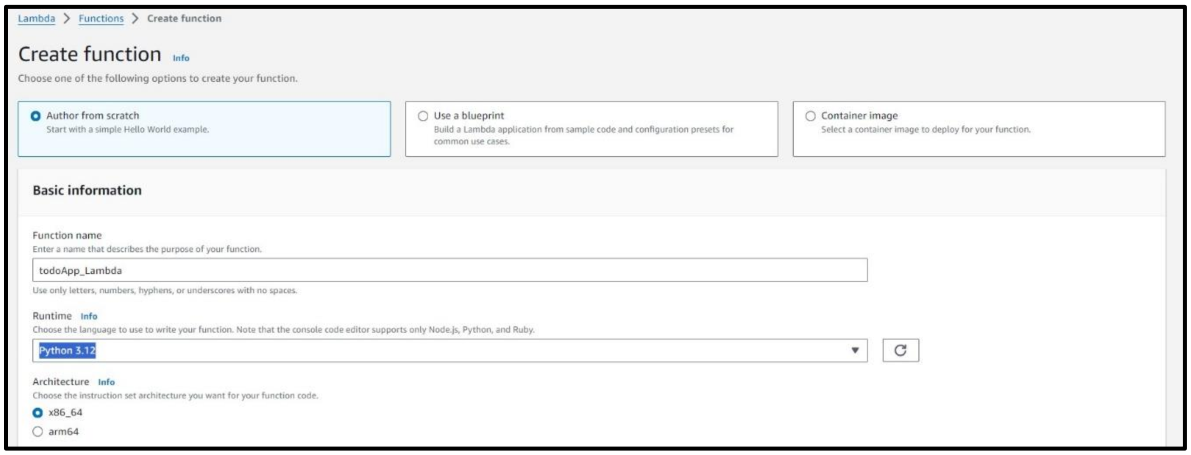
API Gateway Setup: Use API Gateway to set up the API endpoints, connecting each endpoint to the corresponding Lambda function.

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. Creating a Lambda Function



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.
todoApp_Lambda
Use only letters, numbers, hyphens, or underscores with no spaces.

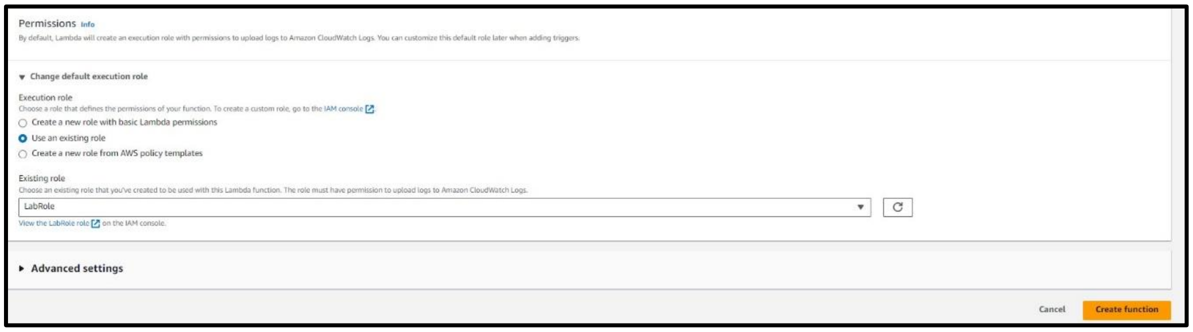
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

Figure 1 Lambda Function Creation

2. Lambda Function Permissions

Lambda function permissions are assigned, and Function is created.



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.

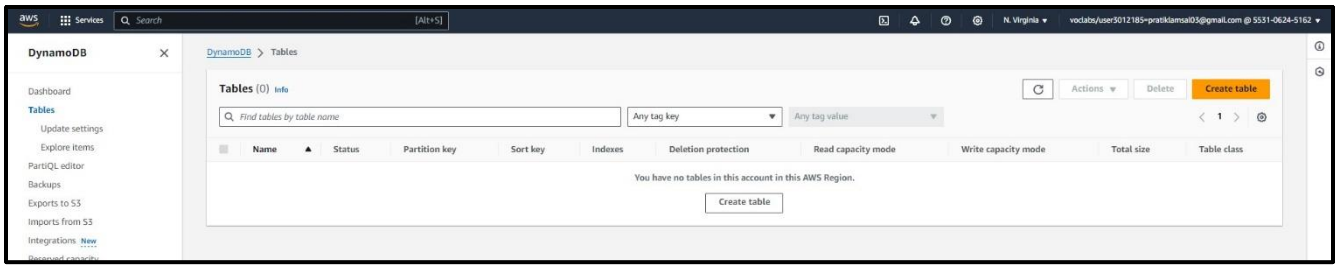
Advanced settings

Cancel Create function

Figure 2 Lambda Function Permissions

3. Creating a Table

A table is to be created in DynamoDB.



Tables (0) Info

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size	Table class

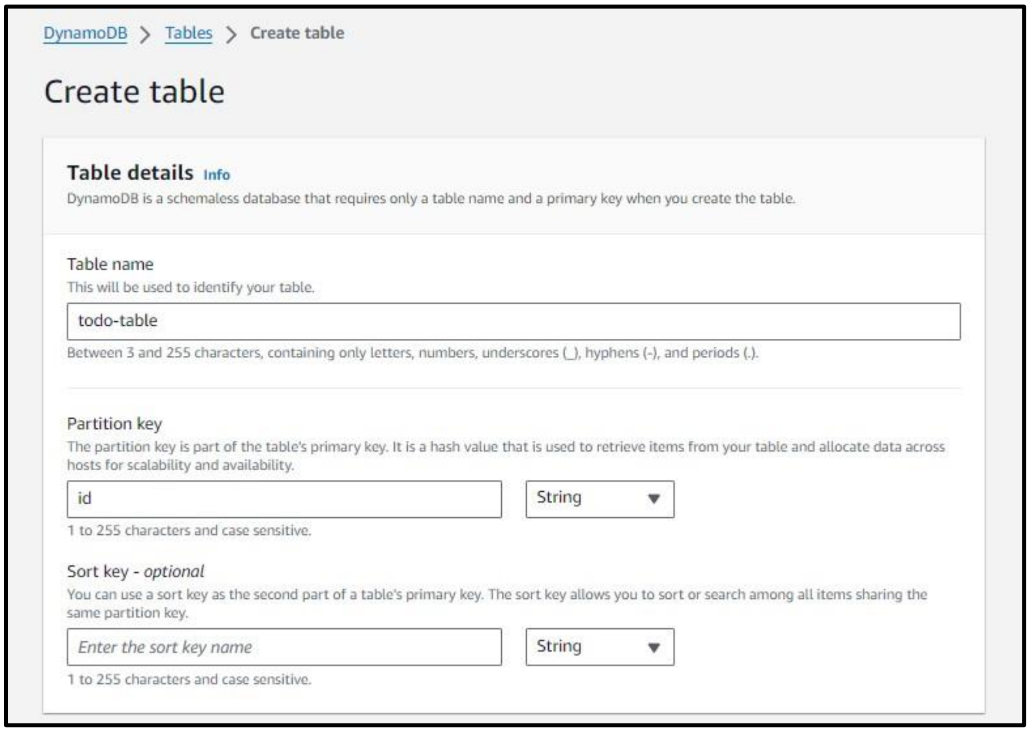
You have no tables in this account in this AWS Region.

Create table

Figure 3 DynamoDB

4. Table Creation

Table with Partition Key is created. It is important to remember the Partition Key as it will be required in later steps.



DynamoDB > Tables > Create table

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.

todo-table

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*

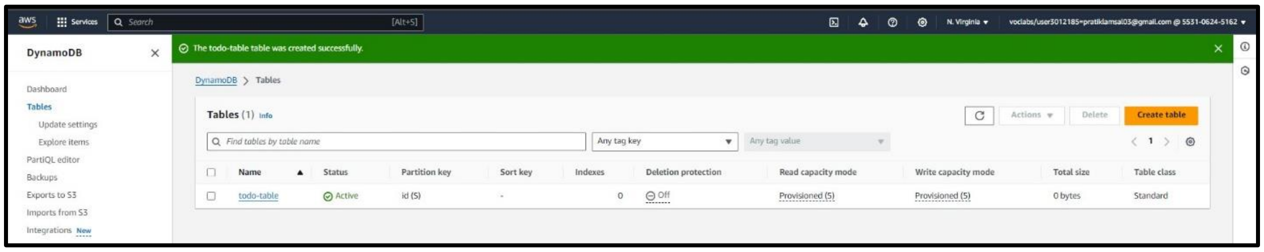
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name | String

1 to 255 characters and case sensitive.

Figure 4 Table Creation

5. Successful Table Creation



vocIabs/user3012185=pratiklamsal03@gmail.com @ 5531-0624-5162

The todo-table table was created successfully.

DynamoDB > Tables

Tables (1) Info

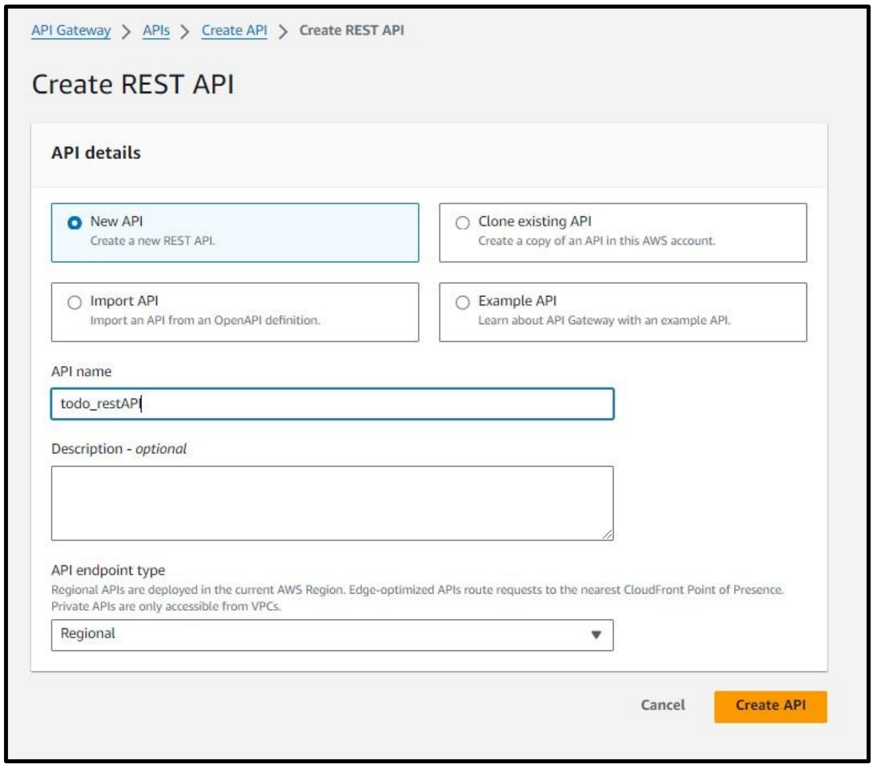
Find tables by table name

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size	Table class
todo-table	Active	id (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes	Standard

Figure 5 Successful Table Creation

6. REST API Creation

API Name may change in further steps while trying to perform the task and facing and error. Rest should be correct.



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: todo_restAPI

Description - *optional*

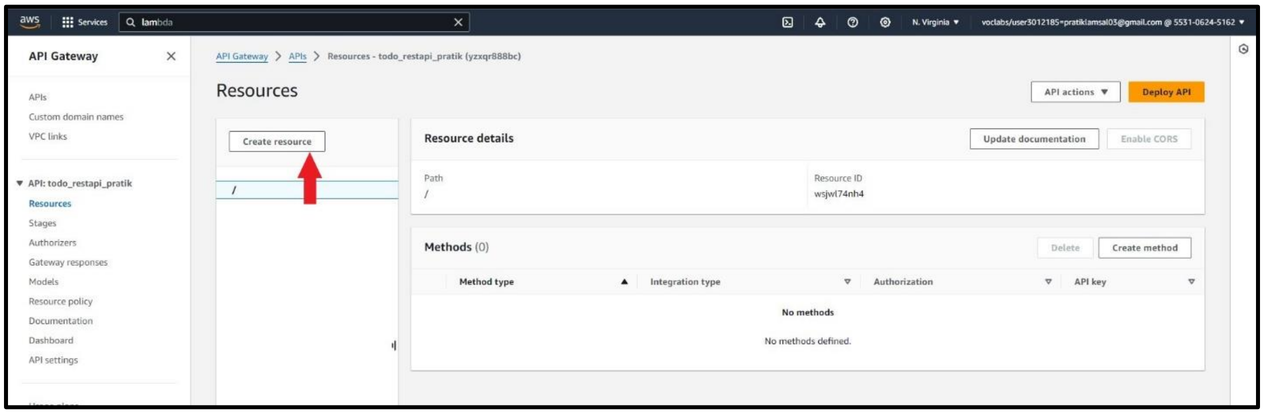
API endpoint type
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

Regional

Figure 6 API Creation

7. Successful API Creation

API is created successfully. Now, resources for it needs to be created.



API Gateway > APIs > Resources - todo_restapi_pratik (yzxqr888bc)

Resources

Resource details

Path: /

Resource ID: wsjwl74nh4

Methods (0)

Method type	Integration type	Authorization	API key

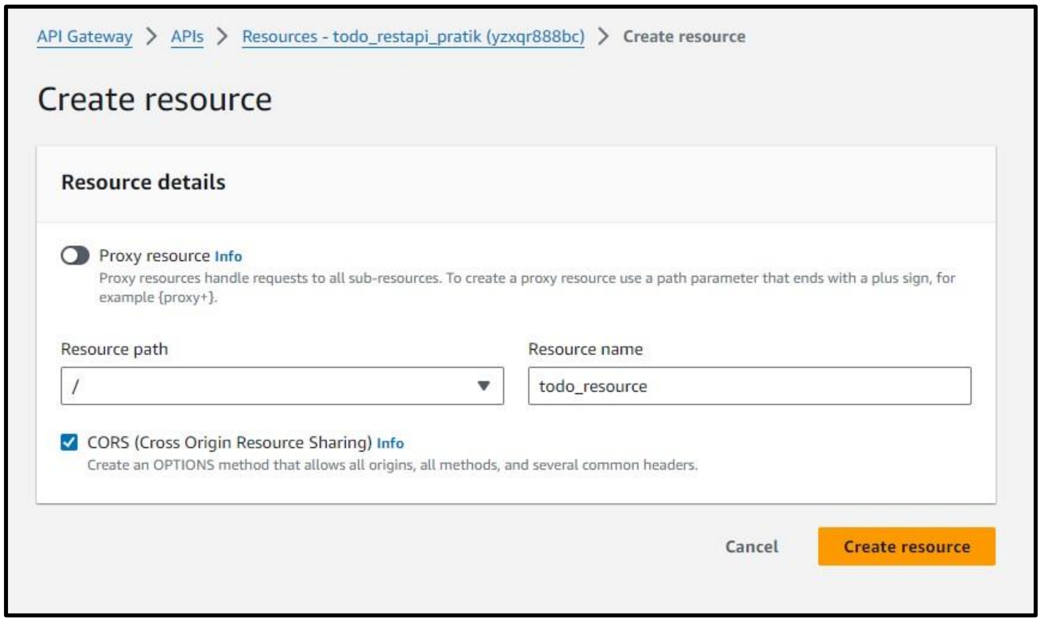
No methods

No methods defined.

Figure 7 Successful API Creation

8. Resource Creation

Resource Name may also change if I encounter errors in further steps. The rest should be correct.



API Gateway > APIs > Resources - todo_restapi_pratik (yzxqr888bc) > 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: /

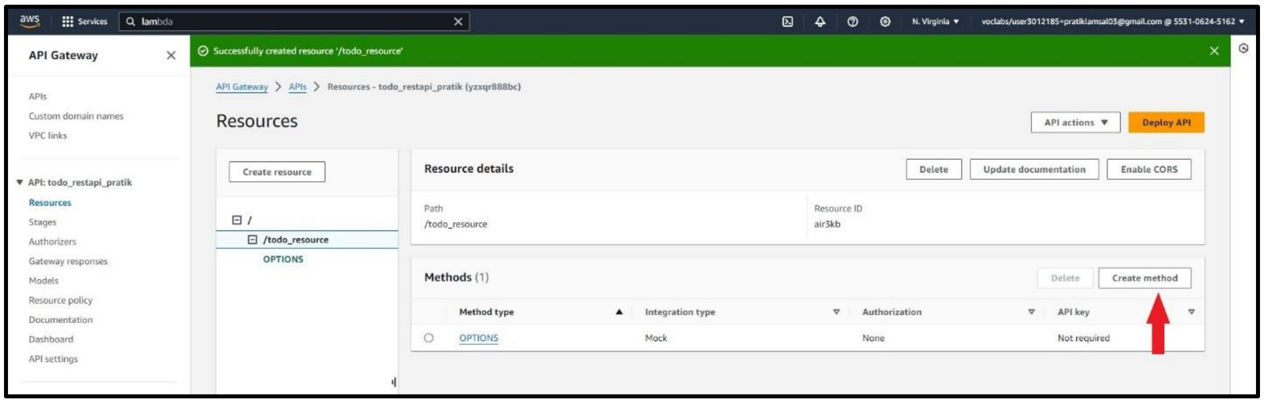
Resource name: todo_resource

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

Figure 8 Resource Creation

9. Successful Resource Creation

Resource is created successfully. Now, Methods are to be created.



Successfully created resource '/todo_resource'

API Gateway > APIs > Resources - todo_restapi_pratik (yzxqr888bc)

Resources

Resource details

Path: /todo_resource

Resource ID: air3kb

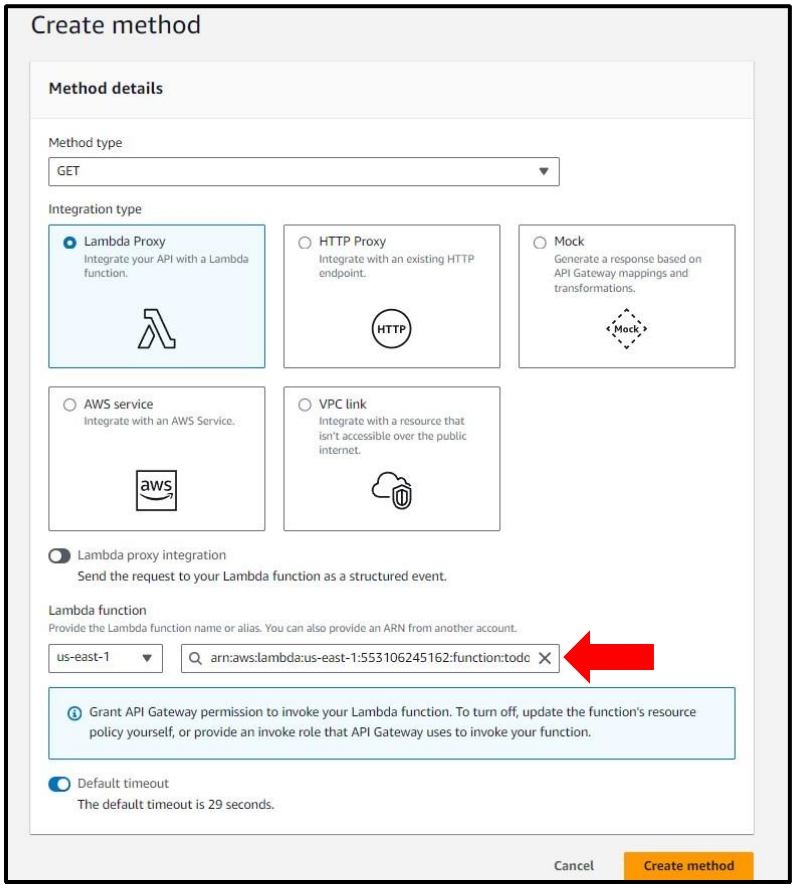
Methods (1)

	Method type	Integration type	Authorization	API key
○	OPTIONS	Mock	None	Not required

Figure 9 Successful Resource Creation

10. Method Creation

All methods required for the task are created one by one. Lambda function created in previous steps is selected for all methods.


Create method

Method details

Method type: GET

Integration type

- (●) Lambda Proxy — Integrate your API with a Lambda function.
- ( ) HTTP Proxy — 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.

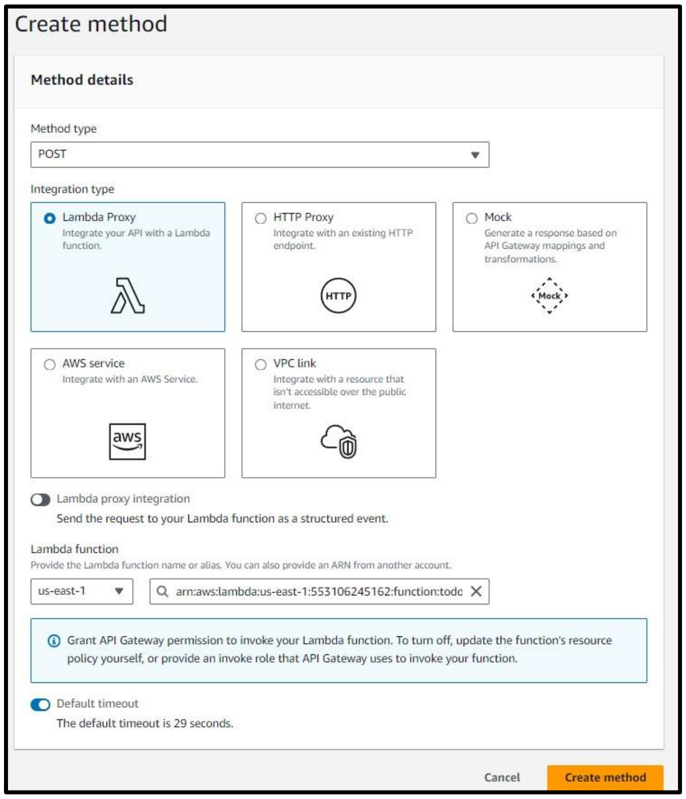
Lambda proxy integration
Send the request to your Lambda function as a structured event.

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-1:553106245162:function:todo

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.

Figure 10 GET Method



Create method

Method details

Method type: POST

Integration type

- [✓ Lambda Proxy] Integrate your API with a Lambda function.
- HTTP Proxy — 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.

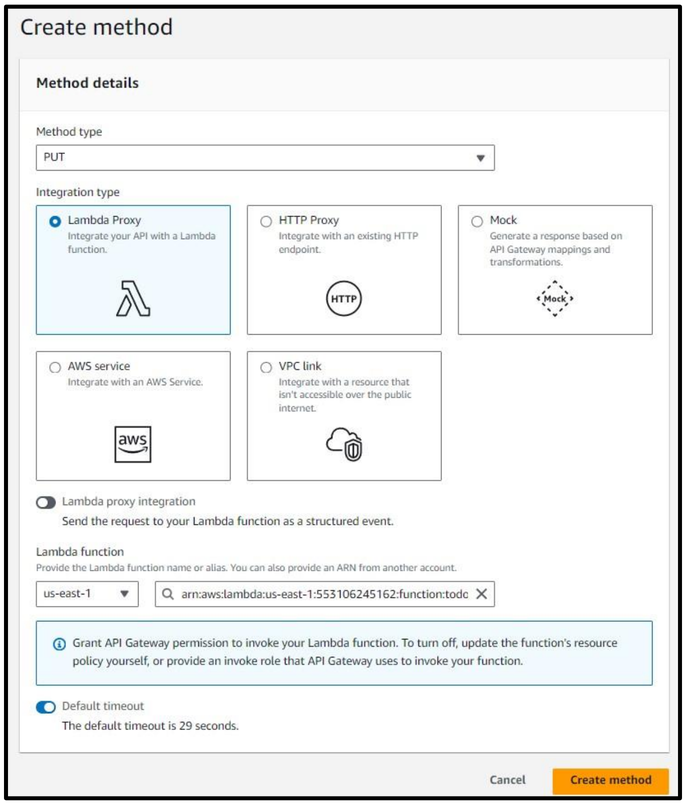
Lambda proxy integration
Send the request to your Lambda function as a structured event.

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-1:553106245162:function:todo

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.

Figure 11 POST Method



Create method

Method details

Method type: PUT

Integration type

- [✓ Lambda Proxy] Integrate your API with a Lambda function.
- HTTP Proxy — 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.

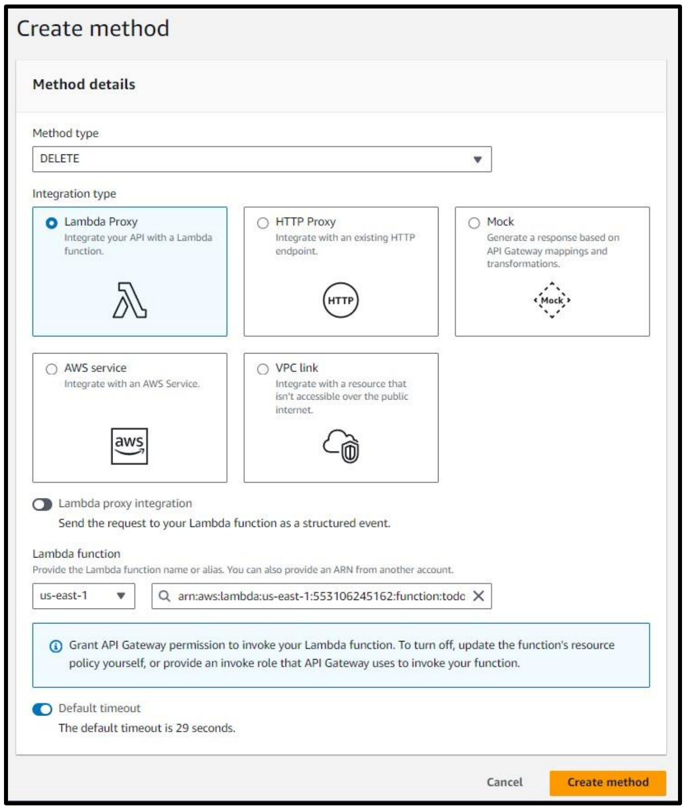
Lambda proxy integration
Send the request to your Lambda function as a structured event.

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-1:553106245162:function:todo

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.

Figure 12 PUT Method



Method type

DELETE

Integration type

- [✓ Lambda Proxy] Integrate your API with a Lambda function.
- HTTP Proxy — 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-1:553106245162:function:todo

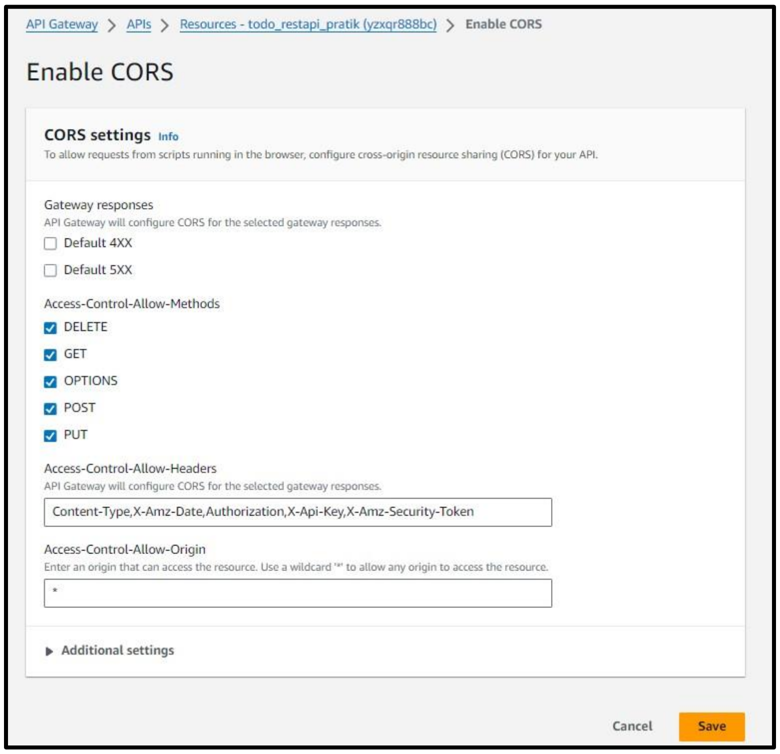
Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

Default timeout
The default timeout is 29 seconds.

Figure 13 DELETE Method

11. Enabling CORS

After method are created, CORS can be enabled for all. The option is available in left side of the screen. On clicking the Enable CORS button, the following window appears. All methods are allowed. The button placement for it is shown in next step.



API Gateway > APIs > Resources - todo_restapi_pratik (yzxqr888bc) > Enable CORS

Enable CORS

CORS settings Info

To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for your API.

Gateway responses

API Gateway will configure CORS for the selected gateway responses.

- ☐ Default 4XX
- ☐ Default 5XX

Access-Control-Allow-Methods

- ☑ DELETE
- ☑ GET
- ☑ OPTIONS
- ☑ POST
- ☑ PUT

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

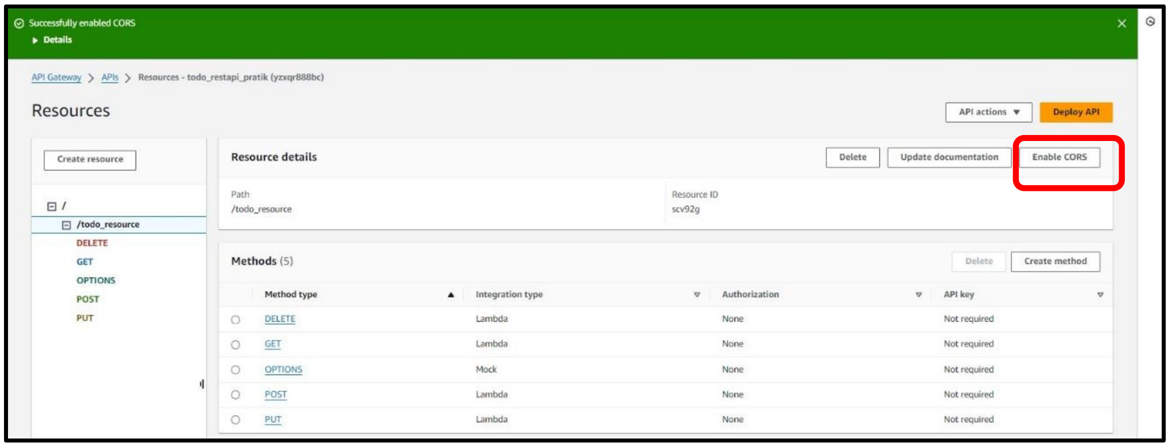
Enter an origin that can access the resource. Use a wildcard '*' to allow any origin to access the resource.

*

▸ Additional settings

Figure 14 Enabling CORS

12. CORS Enabled Successfully



Successfully enabled CORS

▸ Details

API Gateway > APIs > Resources - todo_restapi_pratik (yzxqr888bc)

Resources

/
/todo_resource
DELETE
GET
OPTIONS
POST
PUT

Resource details

Path: /todo_resource

Resource ID: scv92g

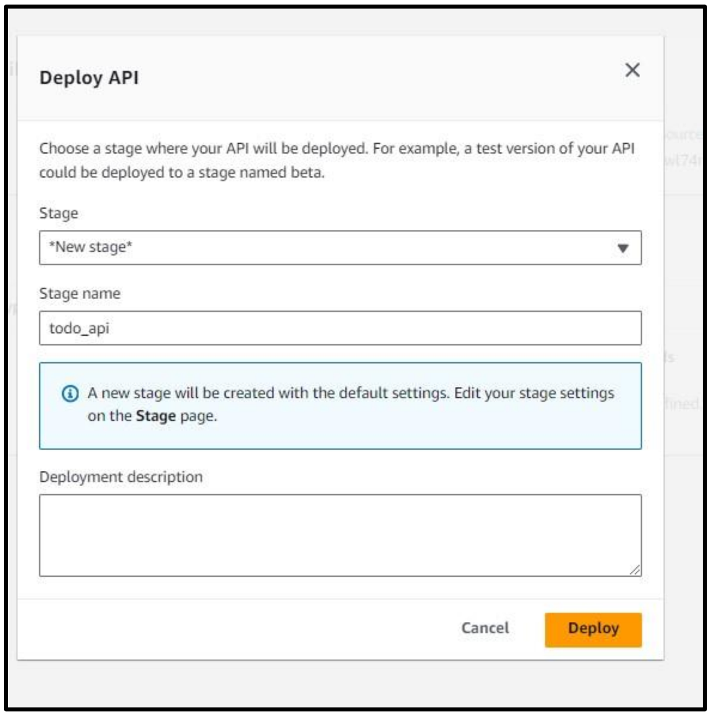
Methods (5)

Method type	Integration type	Authorization	API key
DELETE	Lambda	None	Not required
GET	Lambda	None	Not required
OPTIONS	Mock	None	Not required
POST	Lambda	None	Not required
PUT	Lambda	None	Not required

Figure 15 CORS Enabled Successfully

13. API Deployment

Now, API is to be deployed. On clicking the Deploy button on top-right of the Resources screen, the following window appears. API is deployed in a New Stage.



Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Stage: *New stage*

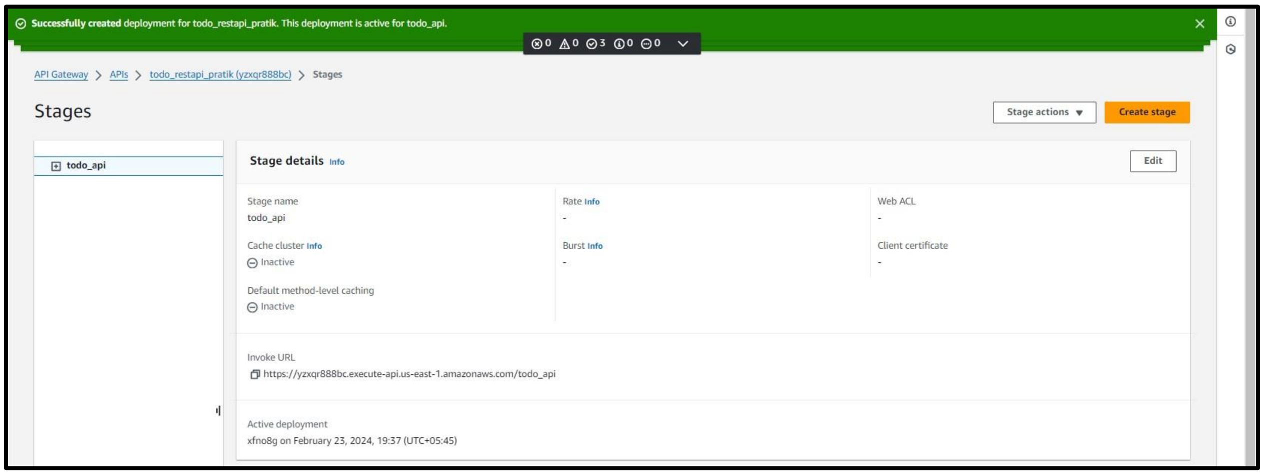
Stage name: todo_api

A new stage will be created with the default settings. Edit your stage settings on the **Stage** page.

Deployment description:

Figure 16 API Deployment

14. Stage Creation



Successfully created deployment for todo_restapi_pratik. This deployment is active for todo_api.

API Gateway > APIs > todo_restapi_pratik (yzxqr888bc) > Stages

Stages

Stage details Info

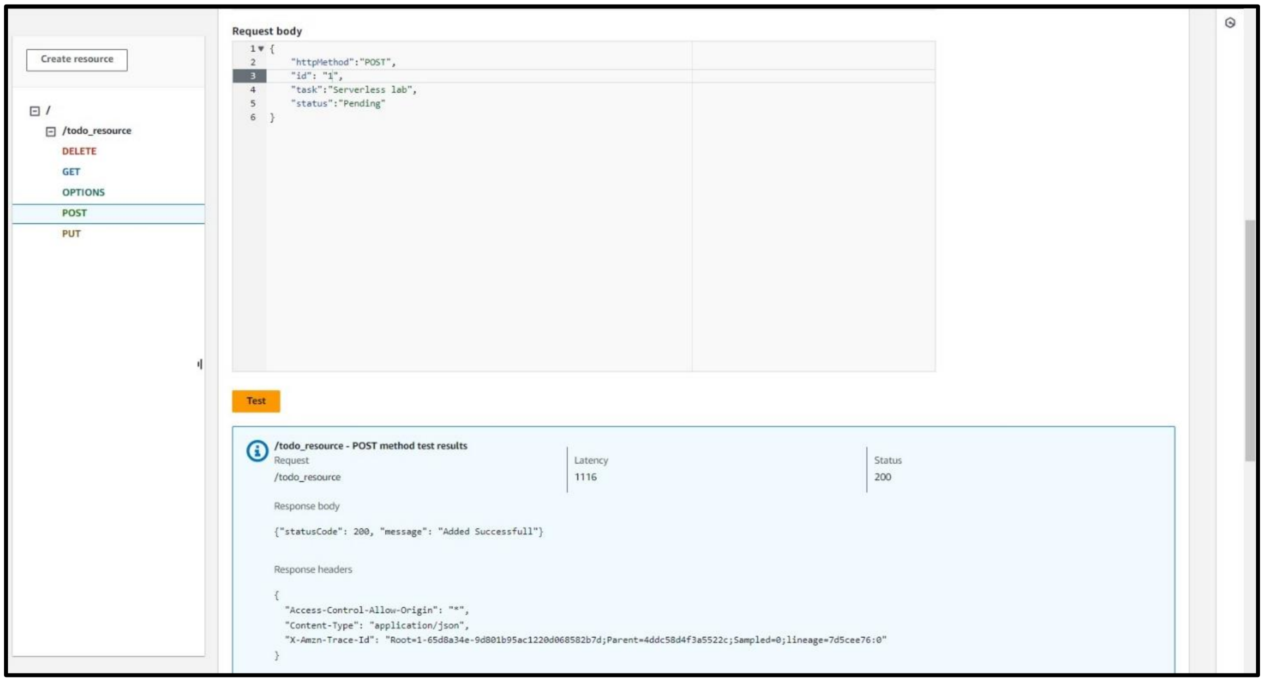
Stage name: todo_api	Rate Info: -	Web ACL: -
Cache cluster Info: Inactive	Burst Info: -	Client certificate: -
Default method-level caching: Inactive		

Invoke URL: https://yzxqr888bc.execute-api.us-east-1.amazonaws.com/todo_api

Active deployment: xfno8g on February 23, 2024, 19:37 (UTC+05:45)

Figure 17 Stage Creation

15. Testing POST Method



Request body

```
{
    "httpMethod":"POST",
    "id": "1",
    "task":"Serverless lab",
    "status":"Pending"
}
```

/todo_resource - POST method test results

Request
/todo_resource

Latency
1116

Status
200

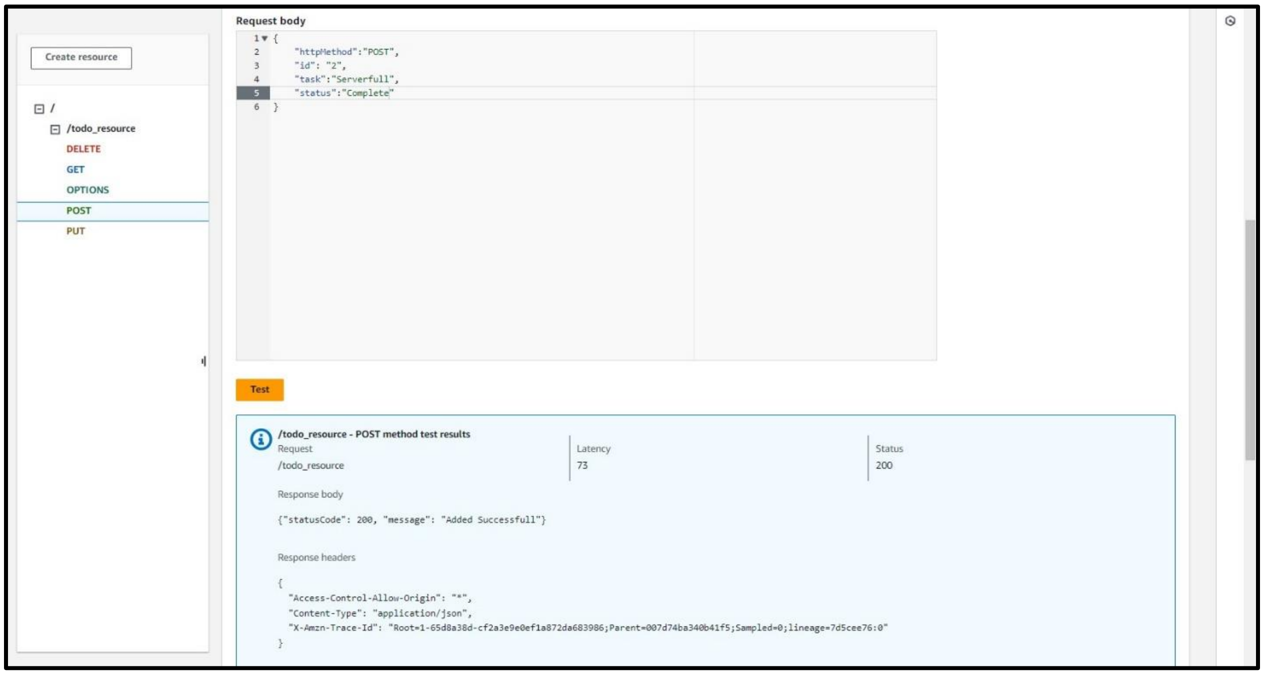
Response body

```
{"statusCode": 200, "message": "Added Successfull"}
```

Response headers

```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-65d8a34e-9d801b95ac1220d068582b7d;Parent=4ddc58d4f3a5522c;Sampled=0;lineage=7d5cee76:0"
}
```

Figure 18 POST Method-1



Request body

```
{
    "httpMethod":"POST",
    "id": "2",
    "task":"Serverfull",
    "status":"Complete"
}
```

/todo_resource - POST method test results

Request
/todo_resource

Latency
73

Status
200

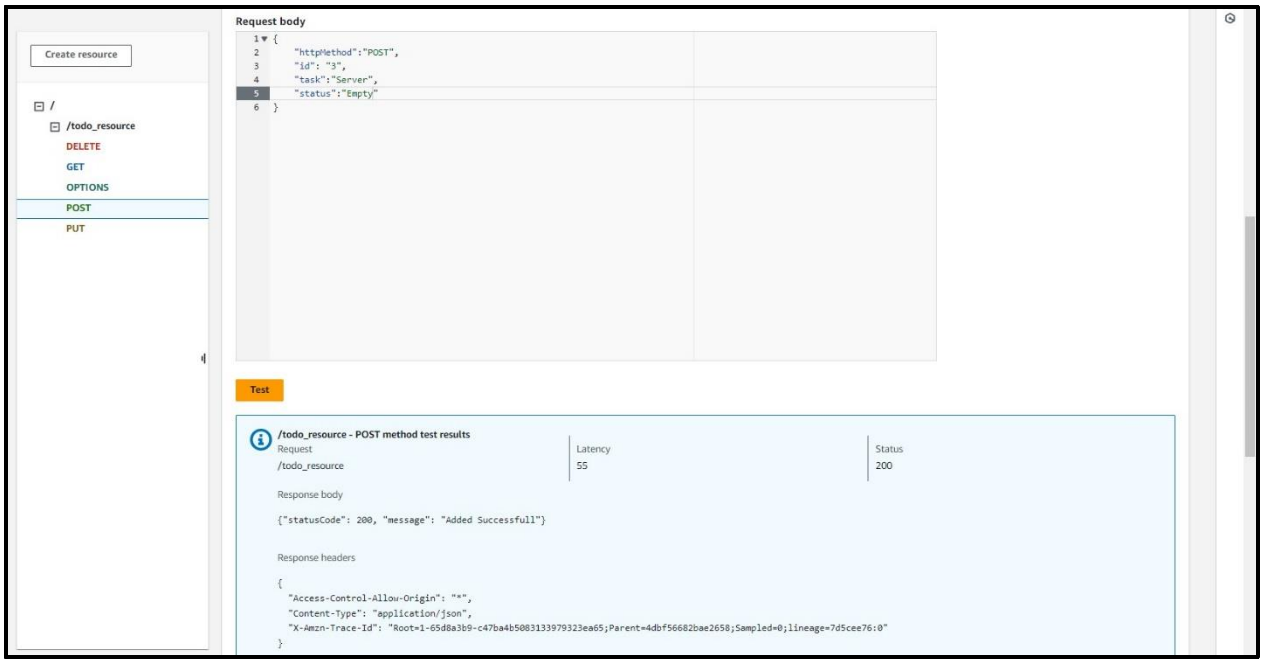
Response body

```
{"statusCode": 200, "message": "Added Successfull"}
```

Response headers

```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-65d8a38d-cf2a3e9e0ef1a872da683986;Parent=007d74ba340b41f5;Sampled=0;lineage=7d5cee76:0"
}
```

Figure 19 POST Method-2



Create resource

/
/todo_resource
DELETE
GET
OPTIONS
POST
PUT

Request body

```
{
    "httpMethod":"POST",
    "id": "3",
    "task":"Server",
    "status":"Empty"
}
```

Test

/todo_resource - POST method test results

Request	Latency	Status
/todo_resource	55	200

Response body

```
{"statusCode": 200, "message": "Added Successfull"}
```

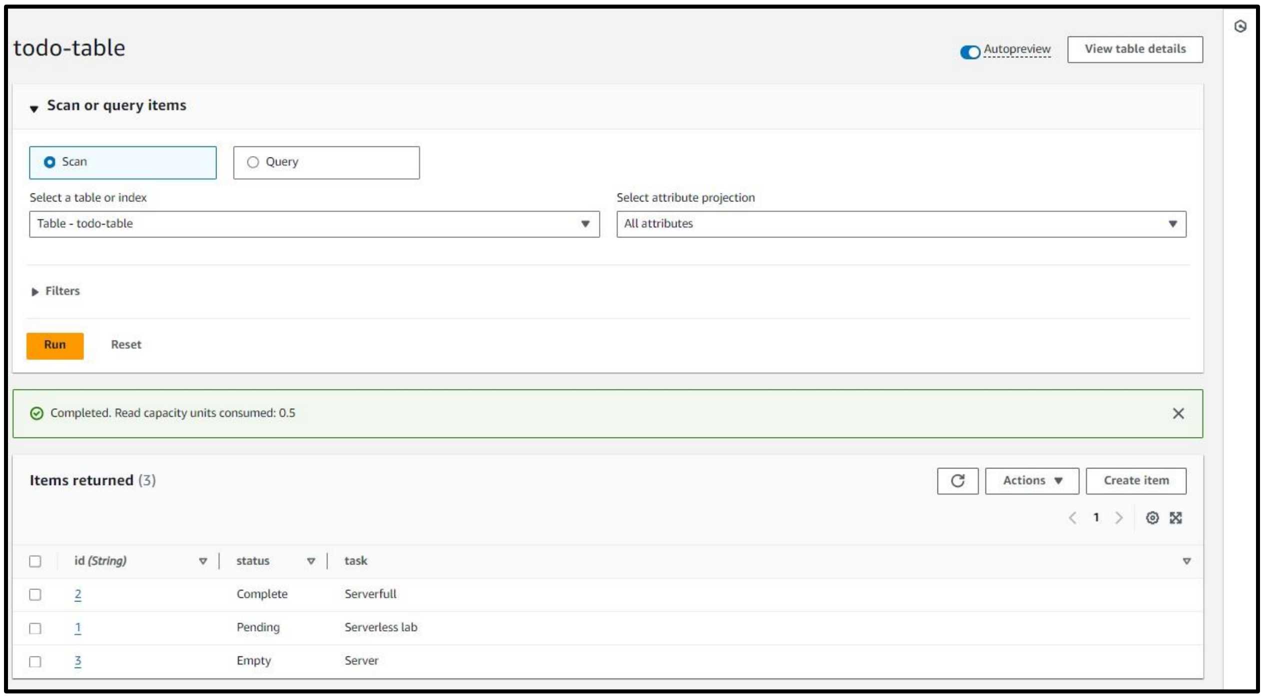
Response headers

```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-65d8a3b9-c47ba4b5083133979323ea65;Parent=4dbf56682bae2658;Sampled=0;lineage=7d5cee76:0"
}
```

Figure 20 POST Method-3

16. POST Method Result

POST is successful.



todo-table

Autopreview | View table details

Scan or query items

[✓ Scan] Query

Select a table or index: Table - todo-table

Select attribute projection: All attributes

Filters

Run | Reset

Completed. Read capacity units consumed: 0.5

Items returned (3)

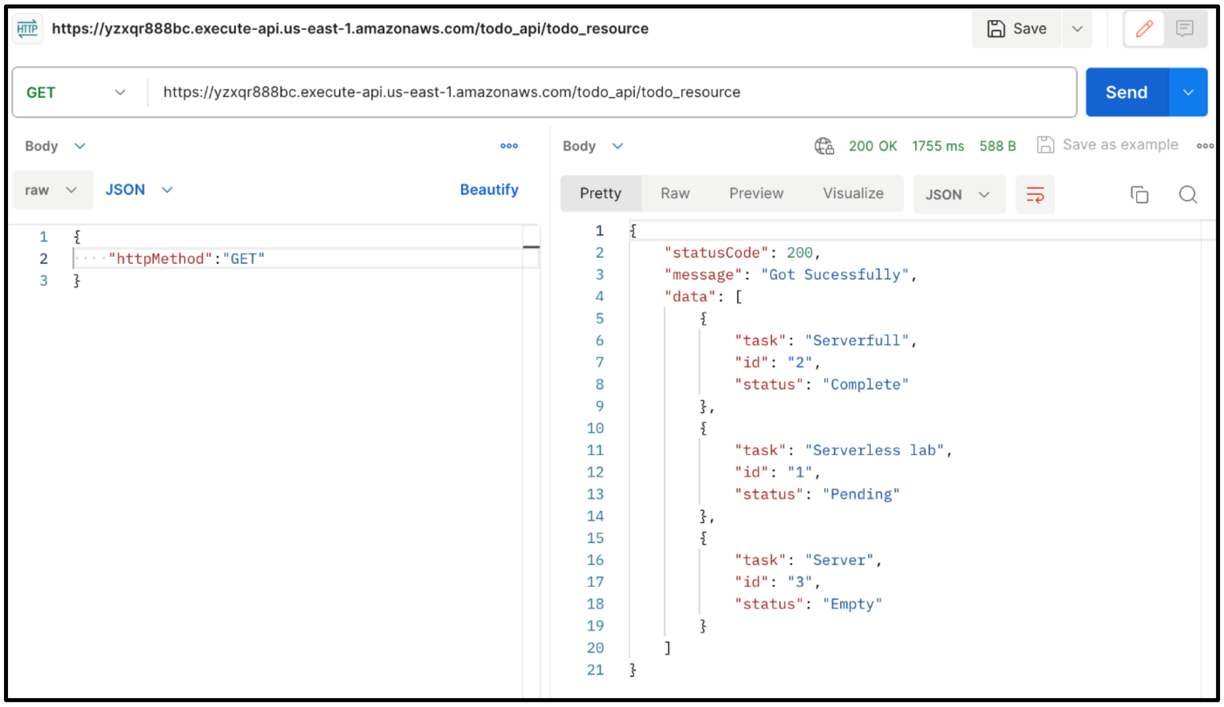
Actions | Create item

id (String)	status	task
2	Complete	Serverfull
1	Pending	Serverless lab
3	Empty	Server

Figure 21 Table with items from POST method

17. Testing GET Method

GET Method tested using Postman. GET is also successful.



https://yzxqr888bc.execute-api.us-east-1.amazonaws.com/todo_api/todo_resource

GET https://yzxqr888bc.execute-api.us-east-1.amazonaws.com/todo_api/todo_resource

Body raw JSON

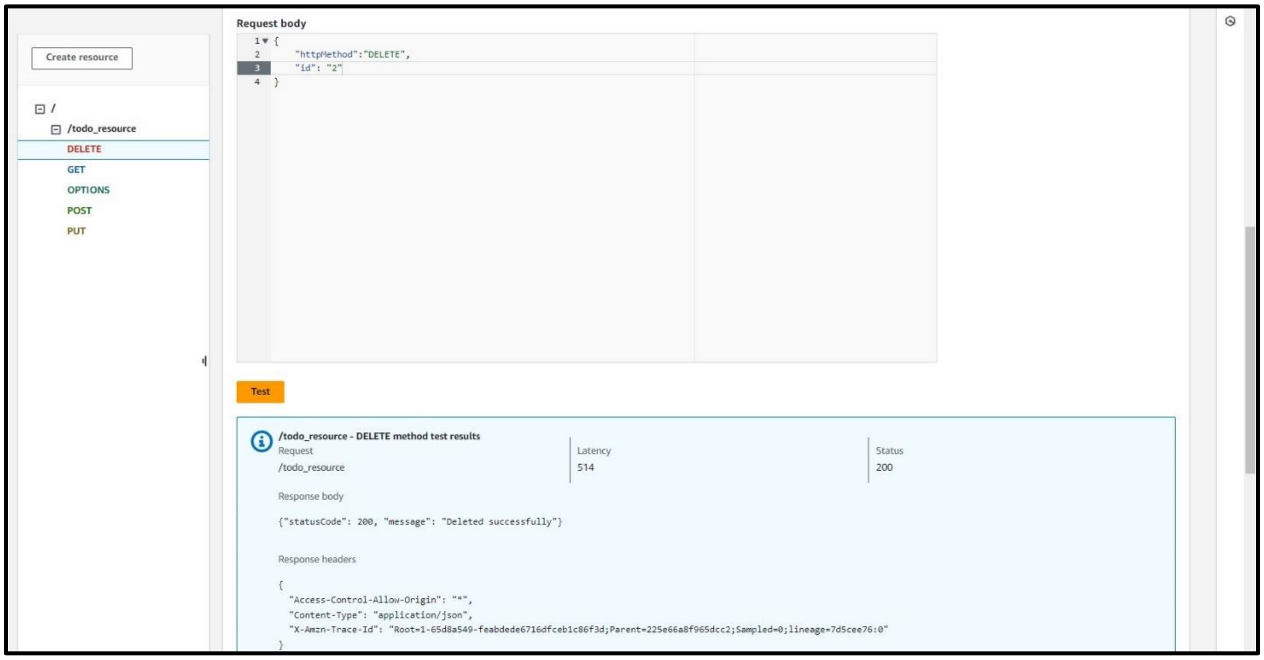
```
{
    "httpMethod":"GET"
}
```

Body 200 OK 1755 ms 588 B

```
{
    "statusCode": 200,
    "message": "Got Sucessfully",
    "data": [
        {
            "task": "Serverfull",
            "id": "2",
            "status": "Complete"
        },
        {
            "task": "Serverless lab",
            "id": "1",
            "status": "Pending"
        },
        {
            "task": "Server",
            "id": "3",
            "status": "Empty"
        }
    ]
}
```

Figure 22 GET Method

18. Testing DELETE Method



Request body

```
{
    "httpMethod":"DELETE",
    "id": "2"
}
```

/todo_resource - DELETE method test results

Request	Latency	Status
/todo_resource	514	200

Response body

```
{"statusCode": 200, "message": "Deleted successfully"}
```

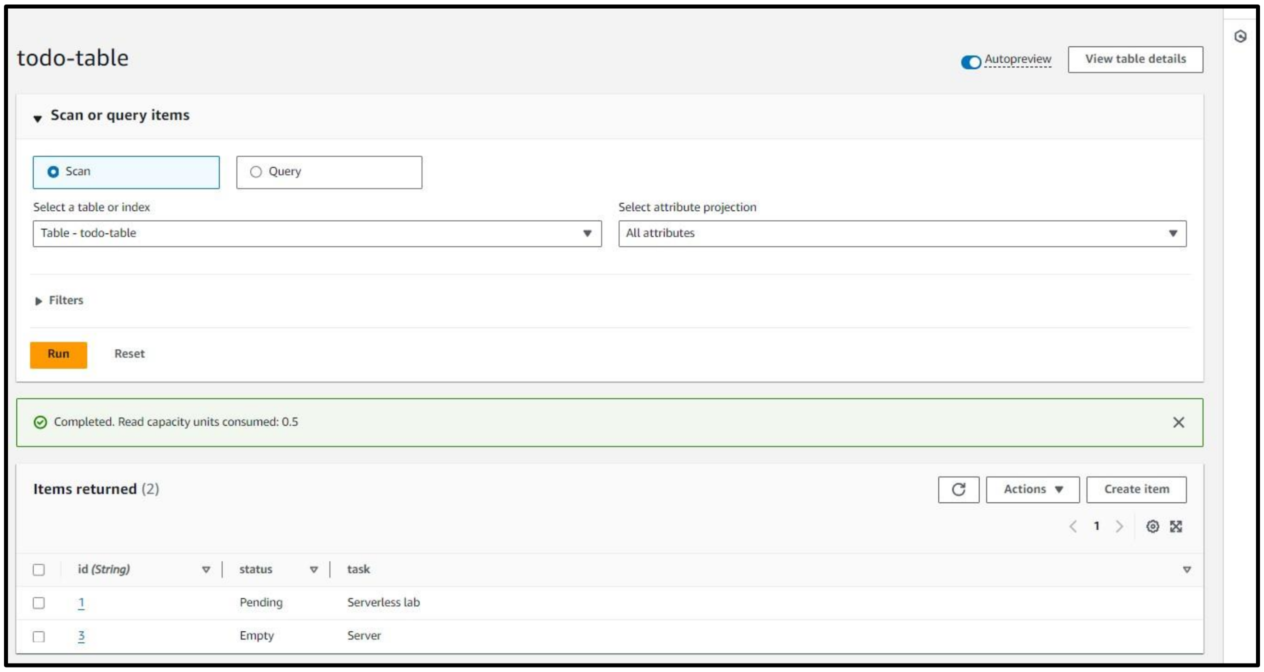
Response headers

```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-65d8a549-feabdede6716dfceb1c86f3d;Parent=225e66a8f965dcc2;Sampled=0;lineage=7d5cee76:0"
}
```

Figure 23 Delete Method

19. DELETE Method Result

DELETE is successful.



todo-table

Scan or query items

Scan [✓] Query

Select a table or index: Table - todo-table

Select attribute projection: All attributes

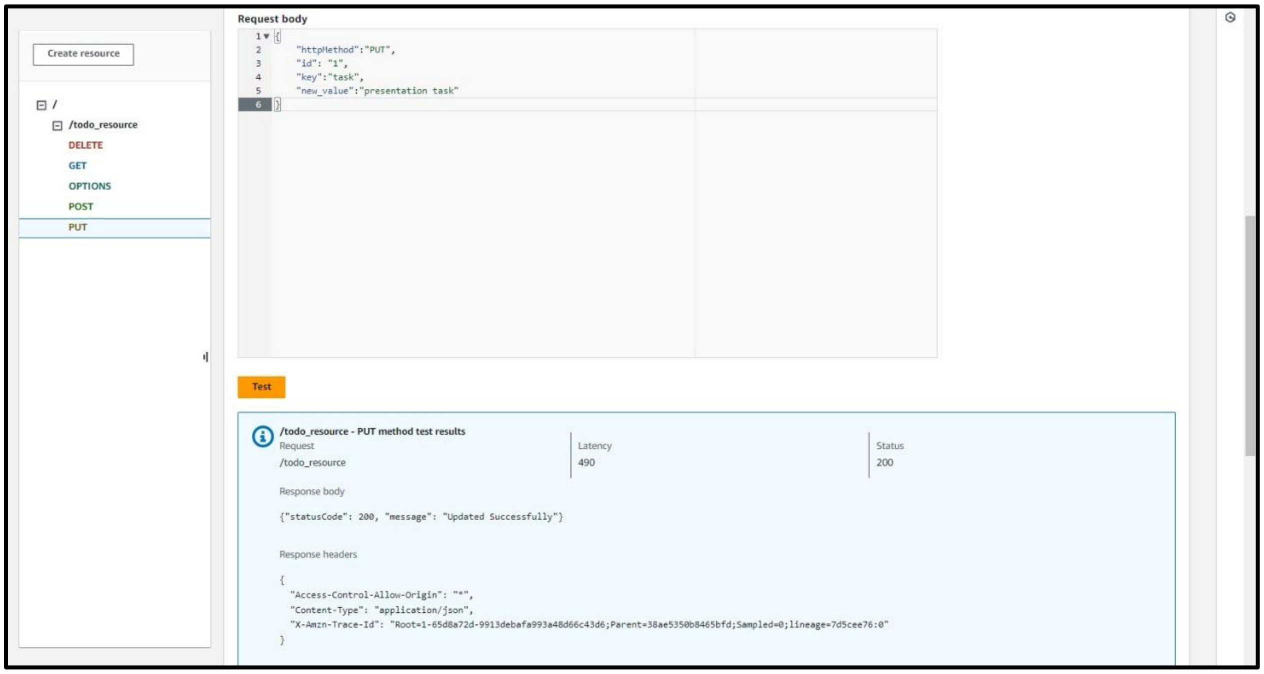
Completed. Read capacity units consumed: 0.5

Items returned (2)

id (String)	status	task
1	Pending	Serverless lab
3	Empty	Server

Figure 24 Table with items delete from DELETE Method

20. Testing PUT Method



Request body

```
{
    "httpMethod":"PUT",
    "id": "1",
    "key":"task",
    "new_value":"presentation task"
}
```

/todo_resource - PUT method test results

Request: /todo_resource
Latency: 490
Status: 200

Response body

```
{"statusCode": 200, "message": "Updated Successfully"}
```

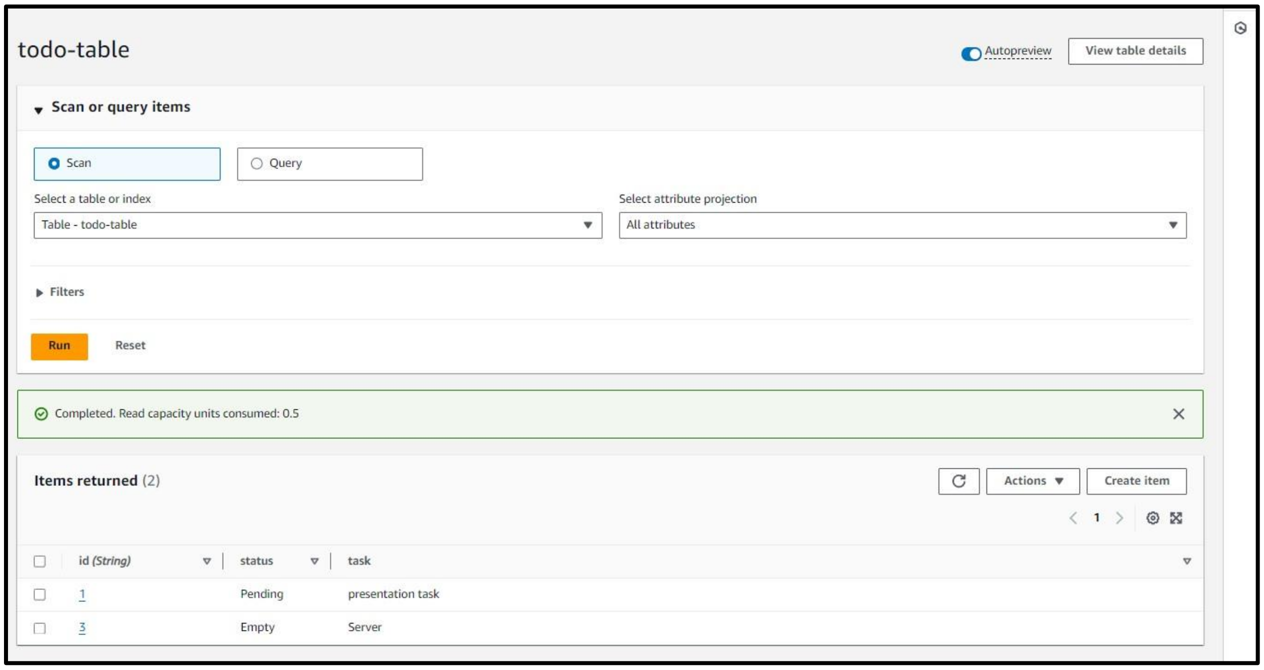
Response headers

```
{
  "Access-Control-Allow-Origin": "*",
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-65d8a72d-9913debafa993a48d66c43d6;Parent=38ae5350b8465bfd;Sampled=0;lineage=7d5cee76:0"
}
```

Figure 25 PUT Method

21. PUT Method Result

PUT is successful.



Scan [✓] Query

Select a table or index: Table - todo-table

Select attribute projection: All attributes

Completed. Read capacity units consumed: 0.5

Items returned (2)

id (String)	status	task
1	Pending	presentation task
3	Empty	Server

Figure 26 Table with items update using PUT Method