

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) Create 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 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.

API-serverless

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 and Ruby.

Python 3.11



2) Assign Lambda function permission

▼ **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.

3) Create Table in DynamoDB

[DynamoDB](#) > **Tables**

Tables (2) [Info](#)

 **Actions** ▼ **Delete** **Create table**

Any tag key ▼

Any tag value ▼

4) Provide table-name and donot forget to give partition key.

[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.

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.

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 a items sharing the same partition key.

String ▼

1 to 255 characters and case sensitive.

5)

6) Table has been successfully created.

[DynamoDB](#) > [Tables](#)

Tables (1) [Info](#)

Any tag key ▼ Any tag value ▼ < 1 > [⚙️](#)

<input type="checkbox"/>	Name ▲	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size	Table class
<input type="checkbox"/>	todo-app	Active	id (S)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes	Standard

7) Build RestAPI

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

ImportBuild

8) Create API with the API name.

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

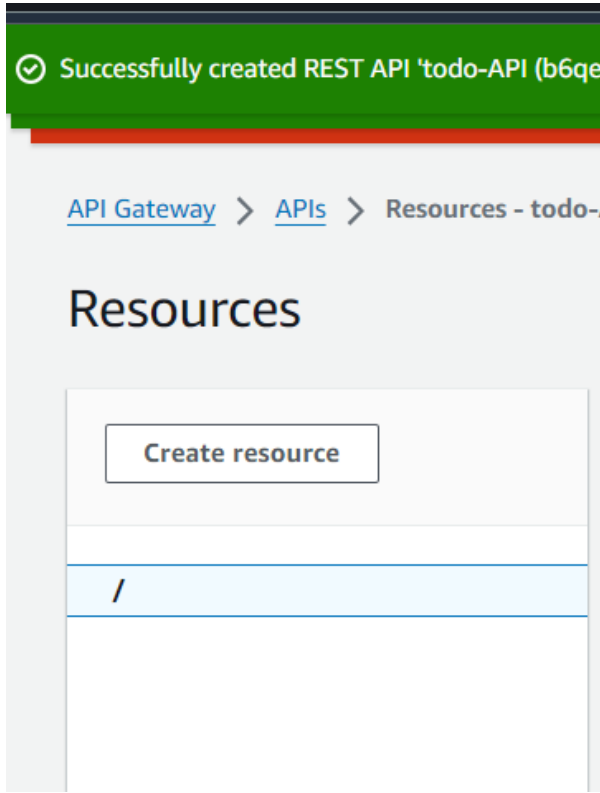
Description - optional

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.

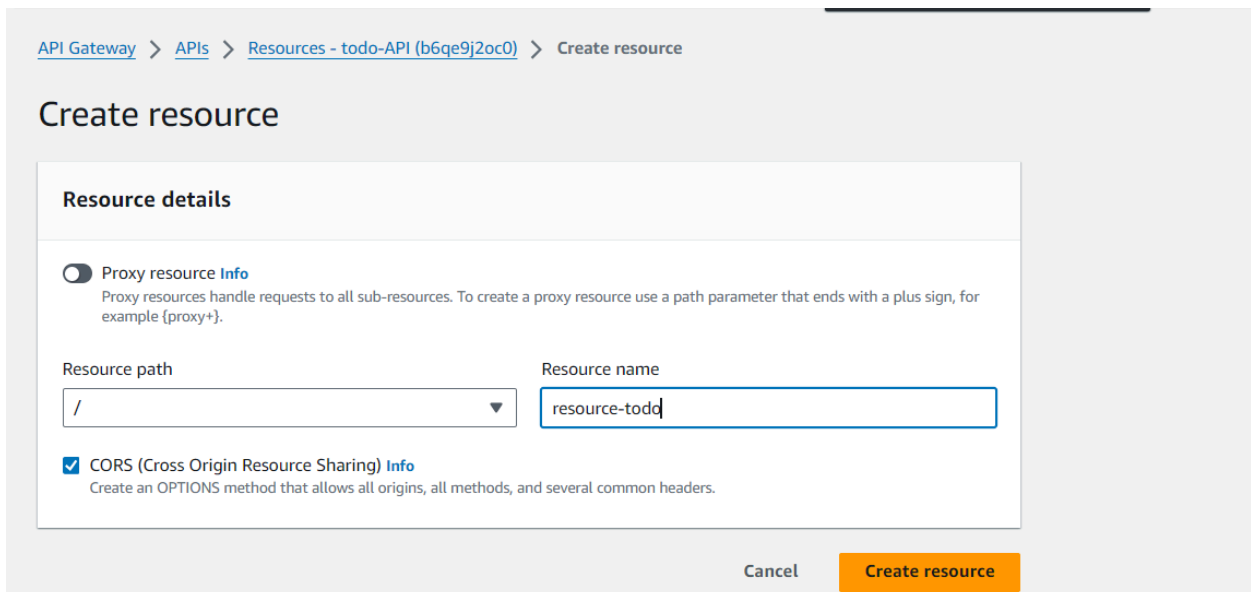
Regional ▼

CancelCreate API

9) After successfully creation of API,add resource.



10) Add resource name and enable CORS.



11) Now after successful creation of resources, methods has to be created.

Path /resource-todo	Resource ID qz5rx1
------------------------	-----------------------

Methods (1)

DeleteCreate method


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


12) Lambda function created in previous steps are used.


For GET method


Method type
GET


Integration type

☒ **Lambda function**
Integrate your API with a Lambda function.


☐ **HTTP**
Integrate with an existing HTTP endpoint.


☐ **Mock**
Generate a n API Gateway transformati


☐ **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-1:304064102356:function:API- X


For POST 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

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.

For PUT method


Create method


Method details


Method type


PUT


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.

For Delete Method

[API Gateway](#) > [APIs](#) > [Resources - todo-API \(b6qe9jzocU\)](#) > [Create method](#)

Create method


Method details

Method type


DELETE

Integration type


☒ **Lambda function**
Integrate your API with a Lambda function.




☐ **HTTP**
Integrate with an external endpoint.



☐ **AWS service**
Integrate with an AWS Service.



☐ **VPC link**
Integrate with a resource that isn't accessible over the internet.



13) Enabling CORS.

API actions ▼

Deploy API

Resource details

Delete

Update documentation

Enable CORS

Path /resource-todo	Resource ID qz5rx1
------------------------	-----------------------

14) For all of the methods, CORS are enabled.

API Gateway > APIs > Resources - todo-API (b6qe9j2oc0) > 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

Cancel

Save

15) Now Click on Deploy API button on right-most side.

todo-API (b6qe9j2oc0)

API actions ▼

Deploy API

Resource details

Delete Update documentation Enable CORS

Path	Resource ID
/resource-todo	qz5rx1

16) A new dialog box appears. Give stage name

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

api-for- todo

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

Deployment description

Cancel

Deploy

17) Create a new stage

API Gateway > APIs > todo-API (b6qe9j2oc0) > Stages

Stages

api-for-todo

Stage details [Info](#)

Stage name	Rate Info	Web ACL
api-for-todo	-	-
Cache cluster Info	Burst Info	Client certificate
<div>Inactive</div>	-	-
Default method-level caching		
<div>Inactive</div>		

Invoke URL

<https://b6qe9j2oc0.execute-api.us-east-1.amazonaws.com/api-for-todo>


18) Testing for POST method.

For pending status(test-1)

Request body

1	▼	{	
2		"httpMethod": "POST",	
3		"id": "2",	
4		"task": "Serverless lab",	
5		"status": "Pending"	
6		}	

Test

 /resource-todo - POST method test results	
Request	Latency
/resource-todo	247
Response body	
{ "statusCode": 200, "message": "New method has been added" }	

For Completed Status(test-2)

Request body

```
1 {  
2   "httpMethod": "POST",  
3   "id": "3",  
4   "task": "Serverless lab",  
5   "status": "Completed"  
6 }
```

Test



/resource-todo - POST method test results

Request

/resource-todo

Latency

24

Response body

{"statusCode": 200, "message": "New method has been added"}

For Empty Status

Request body

```
1 {  
2   "httpMethod": "POST",  
3   "id": "4",  
4   "task": "Server",  
5   "status": "Empty"  
6 }
```

Test



/resource-todo - POST method test results

Request

/resource-todo

Latency

72

Response body

{"statusCode": 200, "message": "New method has been added"}

19) Successful POST method. Its seen under items returned in DynamoDB table.

todo-app Autopreview

▼ Scan or query items

☒ Scan ☐ Query

Select a table or index: Table - todo-app

Select attribute projection: All attributes

► Filters

Run Reset

Completed. Read capacity units consumed: 0.5

Items returned (3) ↻ Actions

<input type="checkbox"/>	id (String)	status	task
<input type="checkbox"/>	2	Pending	Serverless lab
<input type="checkbox"/>	4	Empty	Server
<input type="checkbox"/>	3	Completed	Serverless lab

20) Testing GET method in Postman

GET Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

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

1 {
2 "httpMethod": "GET"
3 }

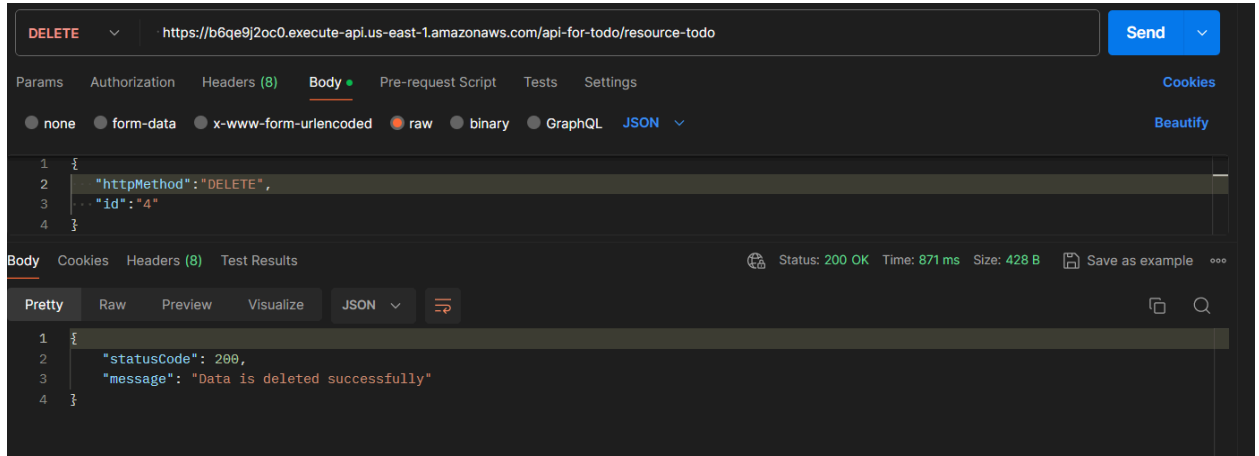
Body Cookies Headers (8) Test Results

Status: 200 OK Time: 898 ms Size: 589 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {  
2   "statusCode": 200,  
3   "message": "Success",  
4   "data": [  
5     {  
6       "task": "Serverless lab",  
7       "id": "2",  
8       "status": "Pending"  
9     },  
10    {  
11      "task": "Server",  
12      "id": "4",  
13      "status": "Empty"  
14    },  
15    {  
16      "task": "Serverless lab",  
17      "id": "3",  
18      "status": "Completed"  
19    }  
20  ]  
}
```

21) Testing DELETE Method



22) After deleting, the results in table is:

Completed. Read capacity units consumed: 0.5

Items returned (2)

<input type="checkbox"/>	id (String)	status	task
<input type="checkbox"/>	2	Pending	Serverless lab
<input type="checkbox"/>	3	Completed	Serverless lab

23) Testing PUT Method

Request body

```
1 {
2   "httpMethod": "PUT",
3   "id": "2",
4   "key": "score",
5   "new_value": "your score is 90"
6 }
```

Test

/resource-todo - PUT method test results

Request	Latency	Status
/resource-todo	77	200

Response body

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

Response headers

The results in DynamoDB table is shown as:

Items returned (1)			
<input type="checkbox"/>	id (String)	score	task
<input type="checkbox"/>	3	your score is 90	Completed

© 2024, Amazon