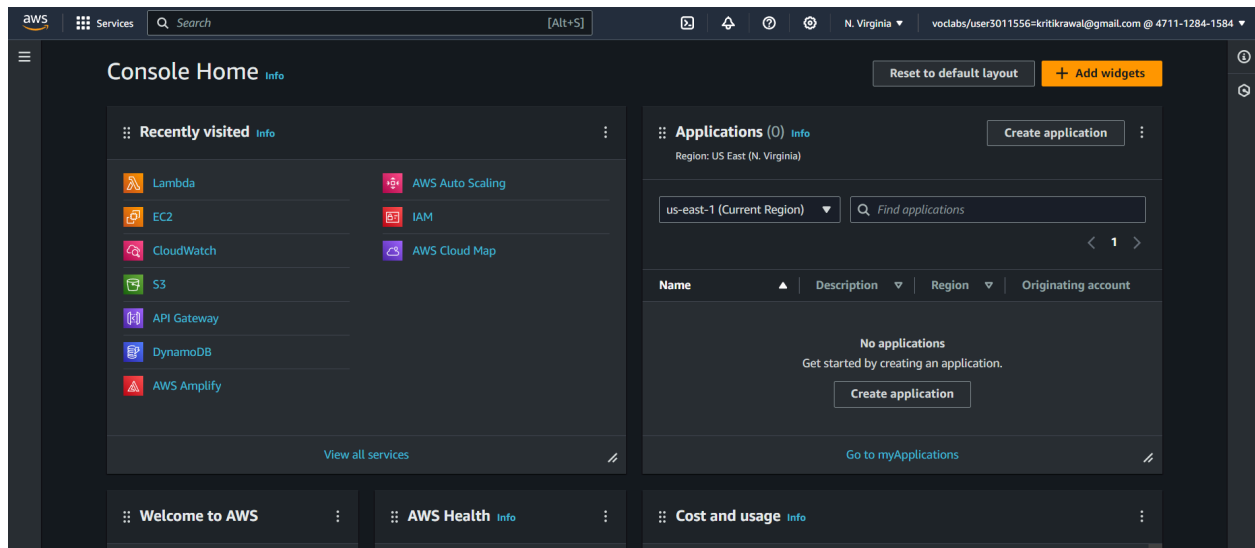
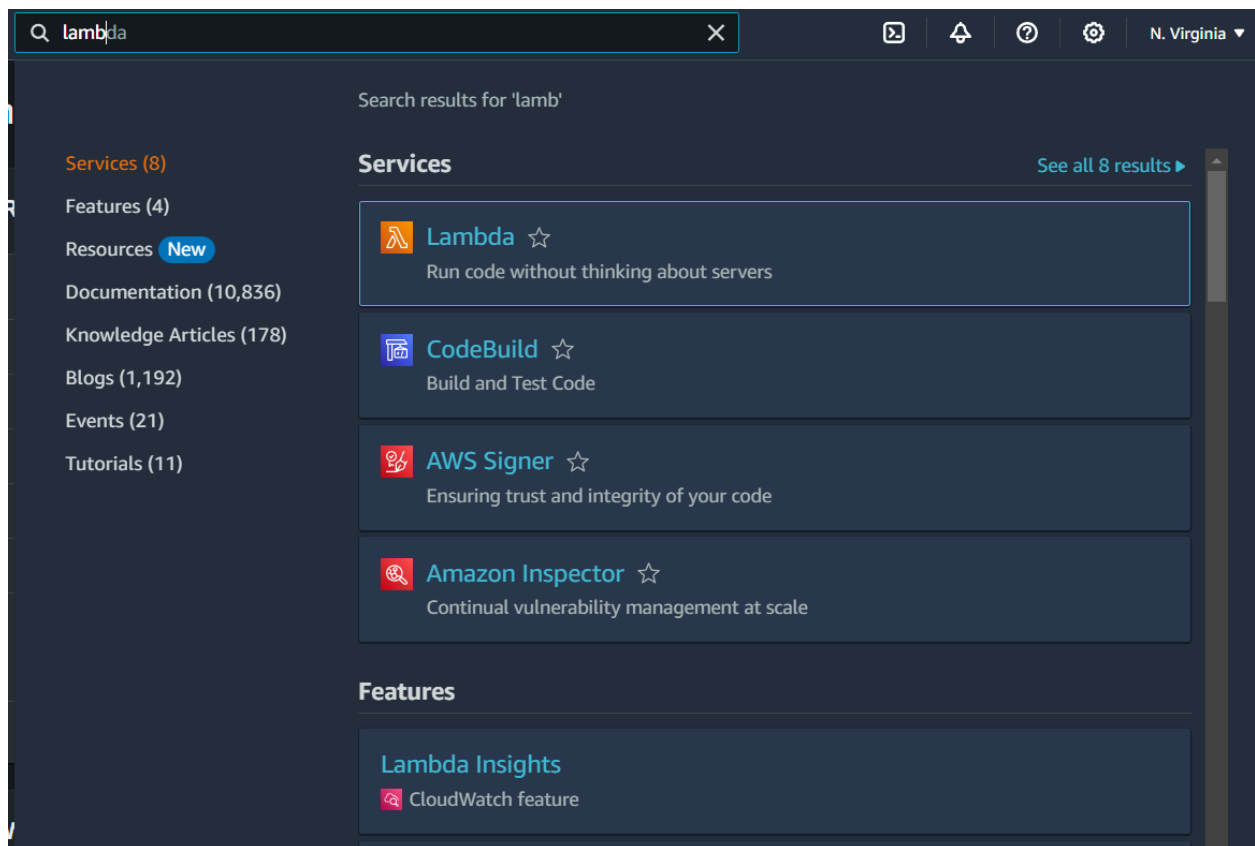


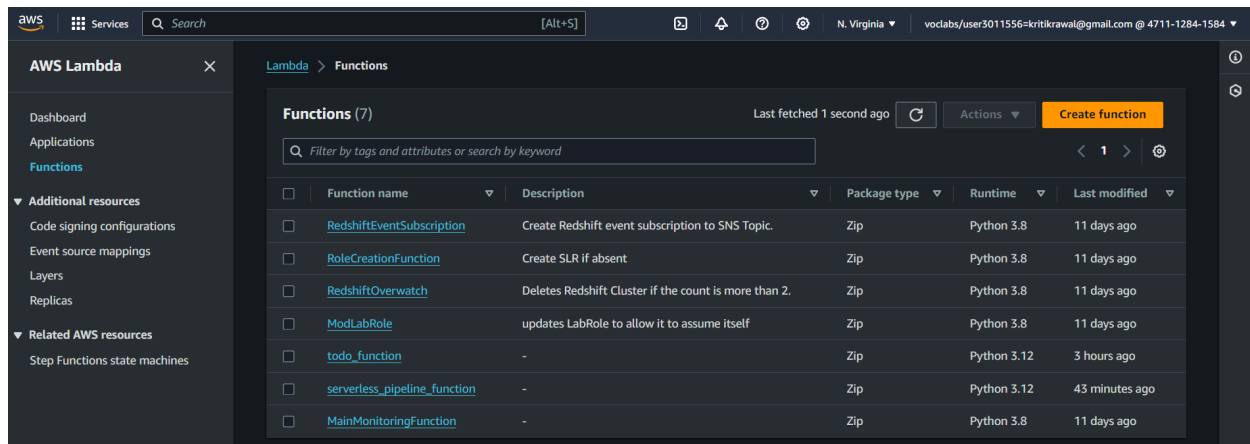
1. Navigate to <https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1>



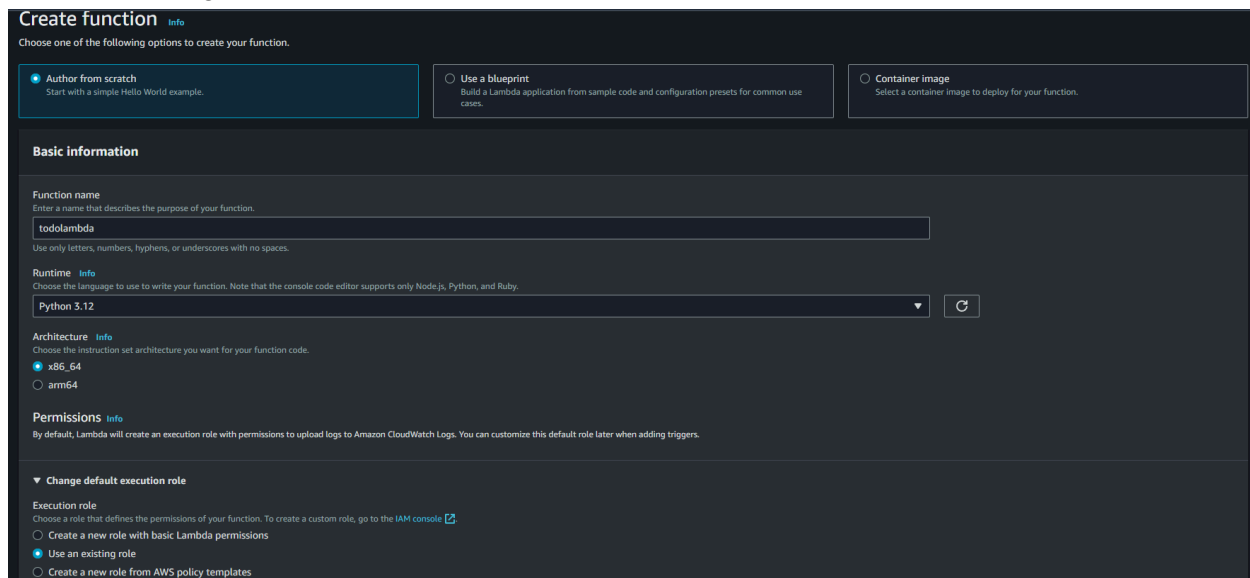
2. Search Lambda



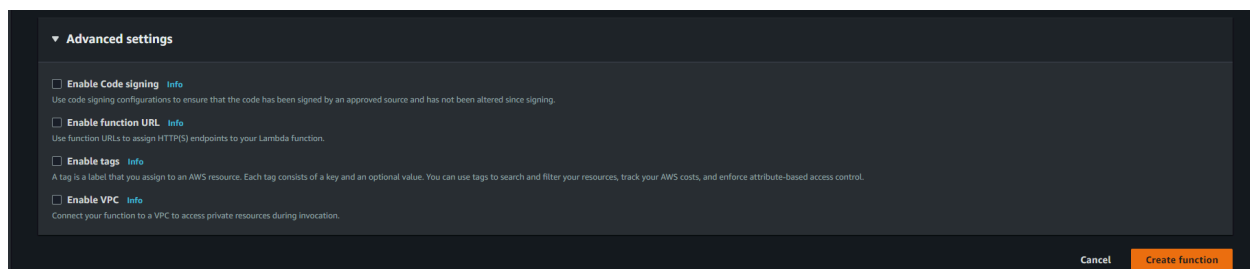
3. Click on the create function.



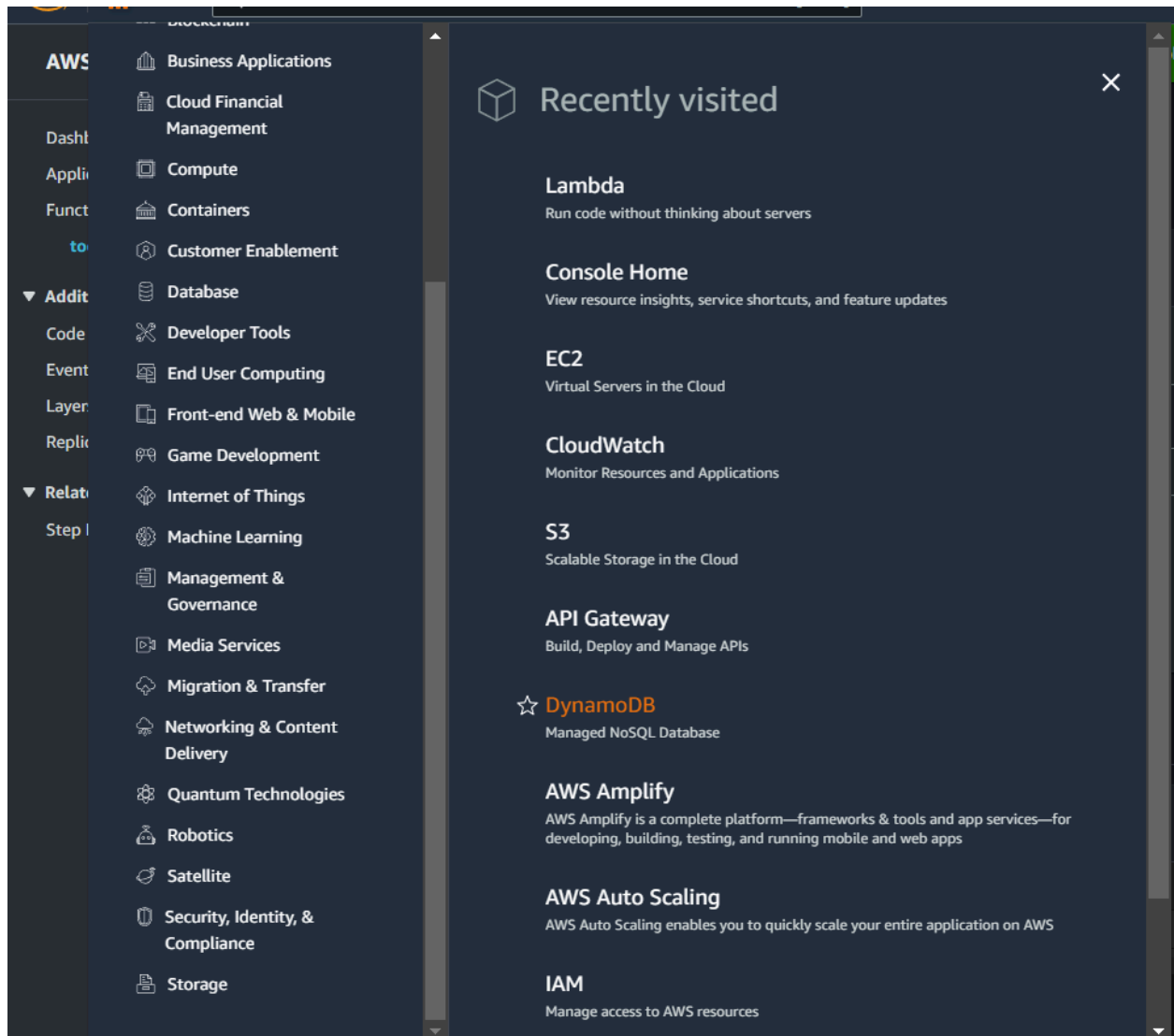
4. Specify the function name, select Python 3.12 in runtime and select Use an existing role in Change default execution role.



5. Create the function.



6. Select DynamoDB to create it.



7. Create the table.

[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 all items sharing the same partition key.

String ▼

1 to 255 characters and case sensitive.

Table settings

☒ **Default settings**

The fastest way to create your table. You can modify these settings now or after your table has been created.

☐ **Customize settings**

Use these advanced features to make DynamoDB work better for your needs.

8. Check the table.

🔔 The mydatabase.db table was created successfully. ✕

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

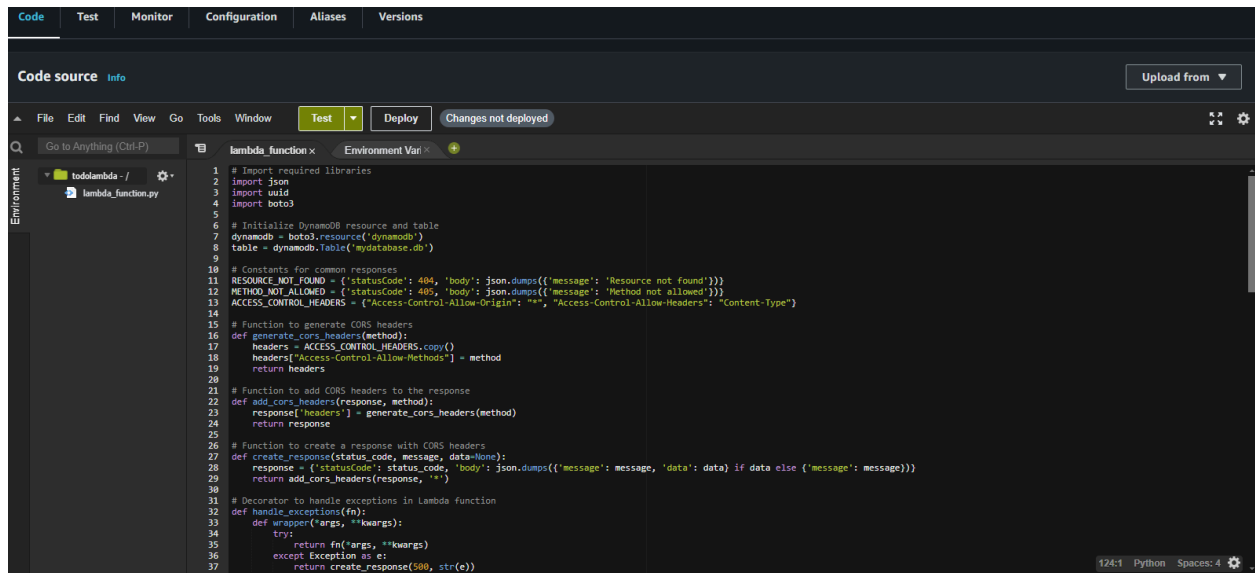
Tables (1/2) [Info](#)

Any tag key ▼Any tag value ▼

[Refresh](#)[Actions ▼](#)[Delete](#)[Create table](#)

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

9. Write the code in the code source.



The screenshot shows the AWS Lambda console's code editor interface. At the top, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is active, displaying the source code for a function named 'lambda_function.py'. The code is written in Python and includes imports for 'json', 'uuid', and 'boto3'. It initializes a DynamoDB resource and a table named 'mydatabase.db'. The code defines several constants for common responses, a function to generate CORS headers, a function to add CORS headers to a response, and a function to create a response with CORS headers. It also includes a decorator to handle exceptions in the Lambda function. The code is as follows:

```
1 # Import required libraries
2 import json
3 import uuid
4 import boto3
5
6 # Initialize DynamoDB resource and table
7 dynamodb = boto3.resource('dynamodb')
8 table = dynamodb.Table('mydatabase.db')
9
10 # Constants for common responses
11 RESOURCE_NOT_FOUND = {'statusCode': 404, 'body': json.dumps({'message': 'Resource not found'})}
12 METHOD_NOT_ALLOWED = {'statusCode': 405, 'body': json.dumps({'message': 'Method not allowed'})}
13 ACCESS_CONTROL_HEADERS = {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Headers': 'Content-Type'}
14
15 # Function to generate CORS headers
16 def generate_cors_headers(method):
17     headers = ACCESS_CONTROL_HEADERS.copy()
18     headers['Access-Control-Allow-Methods'] = method
19     return headers
20
21 # Function to add CORS headers to the response
22 def add_cors_headers(response, method):
23     response['headers'] = generate_cors_headers(method)
24     return response
25
26 # Function to create a response with CORS headers
27 def create_response(status_code, message, data=None):
28     response = {'statusCode': status_code, 'body': json.dumps({'message': message, 'data': data if data else {'message': message}})}
29     return add_cors_headers(response, '')
30
31 # Decorator to handle exceptions in Lambda function
32 def handle_exceptions(fn):
33     def wrapper(*args, **kwargs):
34         try:
35             return fn(*args, **kwargs)
36         except Exception as e:
37             return create_response(500, str(e))
```

10. Go to choose an API gateway and choose REST API.

[API Gateway](#) > [APIs](#) > [Create API](#)

Choose an API type

HTTP API

Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

Works with the following:
Lambda, HTTP backends

[Import](#)[Build](#)

WebSocket API

Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.

Works with the following:
Lambda, HTTP, AWS Services

[Build](#)

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](#)

11. Create the 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

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 ▼

[Cancel](#) [Create API](#)

12. Choose on create resource and create the resource.

[API Gateway](#) > [APIs](#) > [Resources - todo-api \(ljytx3d7sc\)](#) > **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

todos

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

Cancel

Create resource

13. Go to create method and choose GET in method type and also choose the todo function.

Create method


Method details

Method type


GET ▼

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-1:471112841584:function:todo ✕

i 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.

Cancel Create method

14. Go to create resource and enter the resource name.

[API Gateway](#) > [APIs](#) > [Resources - todo-api \(ljytx3d7sc\)](#) > **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
todd

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

[Cancel](#) [Create resource](#)

15. Create GET in todo.

[API Gateway](#) > [APIs](#) > [Resources - todo-api \(ljytx3d7sc\)](#) > **Create method**

Create method


Method details

Method type


GET ▼

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-1:471112841584: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.

[Cancel](#) [Create method](#)

16. In the similar way create PUT, POST and DELETE.

API Gateway > APIs > Resources - todo-api (jytx3d7sc) > Create 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.

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.

☒ **Default timeout**
The default timeout is 29 seconds.

Cancel Create method

API Gateway > APIs > Resources - todo-api (jytx3d7sc) > 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

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.

Cancel Create method

API Gateway > APIs > Resources - todo-api (jytx3d7sc) > Create method

Create method

Method details

Method type
DELETE

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.

☒ **Default timeout**
The default timeout is 29 seconds.

Cancel Create method

17. Deploy the API

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

dev

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

Deployment description

Cancel Deploy

18. Go to the lambda function.

Lambda > Functions > todolambda

todolambda

Throttle Copy ARN Actions

Function overview info

Export to Application Composer Download

Diagram Template

todolambda

Layers (0)

API Gateway (5)

+ Add destination

Description

Last modified 3 minutes ago

Function ARN arn:aws:lambda:us-east-1:471112841584:function:todolambda

Function URL info

Code Test Monitor Configuration Aliases Versions

Code source info

Upload from

File Edit Find View Go Tools Window Test Deploy Changes not deployed

Environment

Go to Anything (Ctrl-P)

todolambda /

lambda_function.py

```
1 # Import required libraries
2 import json
3 import uuid
4 import boto3
5
6 # Initialize DynamoDB resource and table
7 dynamodb = boto3.resource('dynamodb')
8 table = dynamodb.Table('todos-todoapp-dev')
9
10 # Constants for common responses
11 RESOURCE_NOT_FOUND = {'statusCode': 404, 'body': json.dumps({'message': 'Resource not found'})}
12 METHOD_NOT_ALLOWED = {'statusCode': 405, 'body': json.dumps({'message': 'Method not allowed'})}
13 ACCESS_CONTROL_HEADERS = {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Headers': 'Content-Type'}
14
15 # Function to generate CORS headers
16 def generate_cors_headers(method):
17     headers = ACCESS_CONTROL_HEADERS.copy()
18     headers['Access-Control-Allow-Methods'] = method
19     return headers
20
21 # Function to get todos from the database
```

19. Click on deploy

The screenshot shows the AWS Lambda console for the 'todolambda' function. The 'Function overview' tab is active, displaying a diagram with an API Gateway trigger connected to the 'todolambda' function. The function has 0 layers and 5 destinations. The 'Code source' tab is also visible, showing the Python code for the function. The code imports libraries like json, uuid, and boto3, initializes a DynamoDB resource and table, and defines a handler function that generates CORS headers and returns a response.

Function overview

Diagram | Template

API Gateway (5) | + Add trigger

+ Add destination

Code source

```
1 # Import required libraries
2 import json
3 import uuid
4 import boto3
5
6 # Initialize DynamoDB resource and table
7 dynamodb = boto3.resource('dynamodb')
8 table = dynamodb.Table('mydatabase.db')
9
10 # Constants for common responses
11 RESOURCE_NOT_FOUND = {'statusCode': 404, 'body': json.dumps({'message': 'Resource not found'})}
12 METHOD_NOT_ALLOWED = {'statusCode': 405, 'body': json.dumps({'message': 'Method not allowed'})}
13 ACCESS_CONTROL_HEADERS = {'Access-Control-Allow-Origin': '*', 'Access-Control-Allow-Headers': 'Content-Type'}
14
15 # Function to generate CORS headers
16 def generate_cors_headers(method):
17     headers = ACCESS_CONTROL_HEADERS.copy()
18     headers['Access-Control-Allow-Methods'] = method
19     return headers
20
21 # Function to handle GET requests to the resource
```

20. Go to API gateway and select stages.

The screenshot shows the AWS API Gateway console for the 'todolambda' API. The 'Stages' tab is active, displaying the 'dev' stage details. The stage is currently inactive. The 'Stage details' section shows the stage name, rate limit, burst limit, and default method-level caching. The 'Logs and tracing' section shows the status of CloudWatch logs, Detailed metrics, and X-Ray tracing.

API Gateway

APIs | Stages | Resources | Authorizers | Gateway responses | Models | Resource policy | Documentation | Dashboard | API settings

Stages

dev

Stage details

Stage name: dev

Rate info: -

Web ACL: -

Cache cluster info: Inactive

Burst info: -

Client certificate: -

Default method-level caching: Inactive

Inliner URL: https://1ytx3d7ic.execute-api.us-east-1.amazonaws.com/dev

Active deployment: bdepmc on February 24, 2024, 23:59 (UTC+05:45)

Logs and tracing

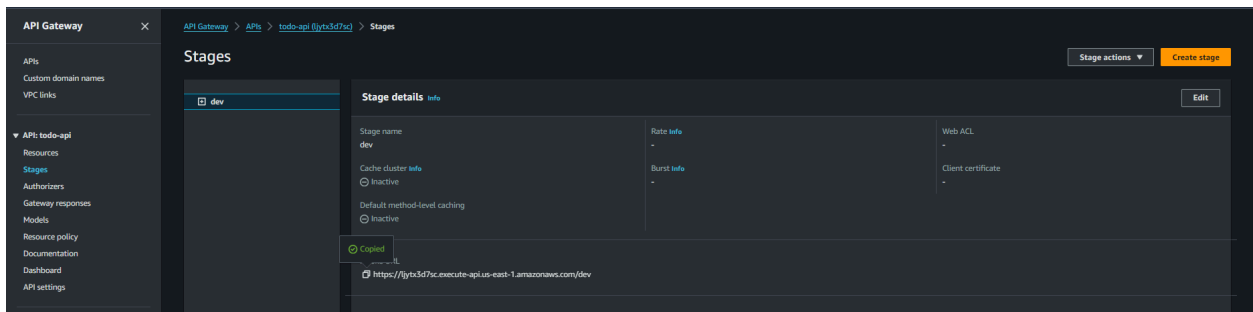
CloudWatch logs: Inactive

Detailed metrics: Inactive

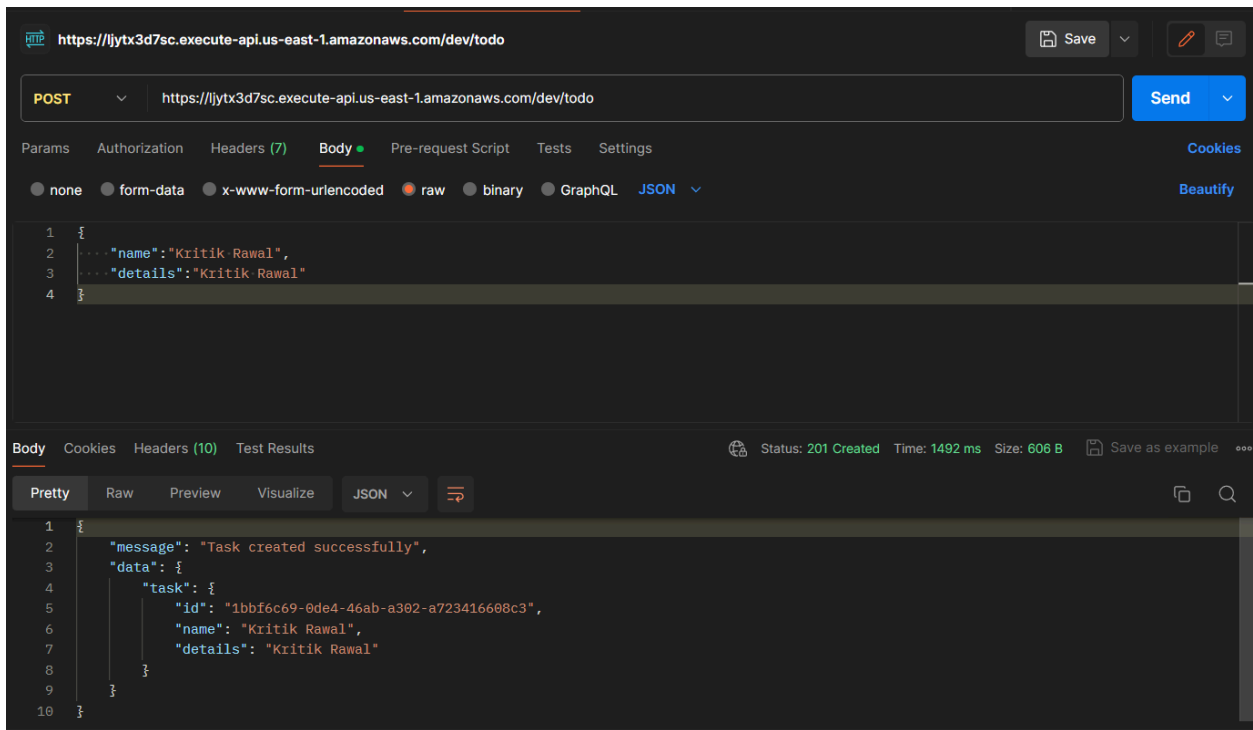
X-Ray tracing: Inactive

Custom access logging: Inactive

21. Copy the invoke URL.



22. Go to postman and check.



23. In the dynamodb check the items returned.

The screenshot shows the AWS DynamoDB console interface. On the left, a sidebar lists tables: 'mydatabase.db' (selected) and 'todos-db'. The main panel is titled 'mydatabase.db' and includes an 'Autopreview' toggle and a 'View table details' button. Below this, the 'Scan or query items' section is active, with the 'Scan' radio button selected. It shows 'Table - mydatabase.db' and 'All attributes' selected for the attribute projection. A 'Run' button is visible. Below the scan options, the 'Items returned (1)' section shows a single item with the following details:

id (String)	details	name
1bbf6c69-0de4-46ab...	Kritik Rawal	Kritik Rawal

24. Send a GET request.

The screenshot shows a REST client interface with a GET request to 'https://jytx3d7sc.execute-api.us-east-1.amazonaws.com/dev/todos'. The request body is a JSON object:

```
{  "name": "Kritik Rawal",  "details": "Kritik Rawal"}
```

. The response status is '200 OK' with a time of '949 ms' and a size of '591 B'. The response body is a JSON object:

```
{  "message": "Tasks retrieved successfully",  "data": [    {      "details": "Kritik Rawal",      "id": "1bbf6c69-0de4-46ab-a302-a723416608c3",      "name": "Kritik Rawal"    }  ]}
```


25. Send a POST request.

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** `https://ijytx3d7sc.execute-api.us-east-1.amazonaws.com/dev/todo?id=1bbf6c69-0de4-46ab-a302-a723416608c3`
- Body:** A JSON object:

```
{
  "name": "Created the new task",
  "details": "Detail is updated"
}
```
- Status:** 200 OK
- Time:** 945 ms
- Size:** 483 B
- Response Body:** A JSON object:

```
{
  "message": "Task updated successfully"
}
```

26. DELETE the request.

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** `https://ijytx3d7sc.execute-api.us-east-1.amazonaws.com/dev/todo?id=1bbf6c69-0de4-46ab-a302-a723416608c3`
- Status:** 200 OK
- Time:** 494 ms
- Size:** 483 B
- Response Body:** A JSON object:

```
{
  "message": "Task deleted successfully"
}
```

27. Refresh the page.

DynamoDB > Explore items > mydatabase.db

Tables (2) ×

Any tag key ▾

Any tag value ▾

🔍 Find tables by table name

< 1 > ⚙️

● mydatabase.db

○ todos-db

mydatabase.db

Autopreview View table details

▶ Scan or query items
Expand to query or scan items.

✔ Completed. Read capacity units consumed: 0.5 ×

Items returned (0) ↻ Actions ▾ Create item

< 1 > ⚙️ 🗨

The query did not return any results.