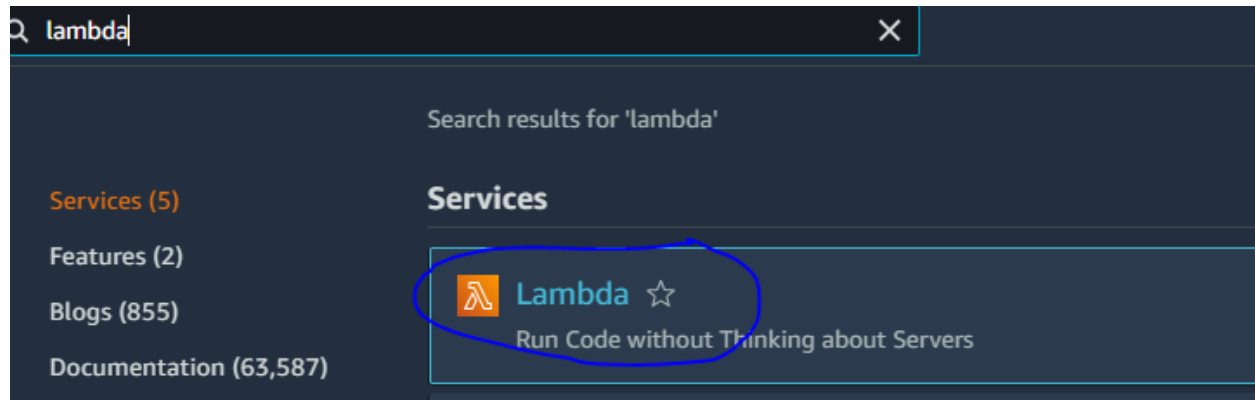
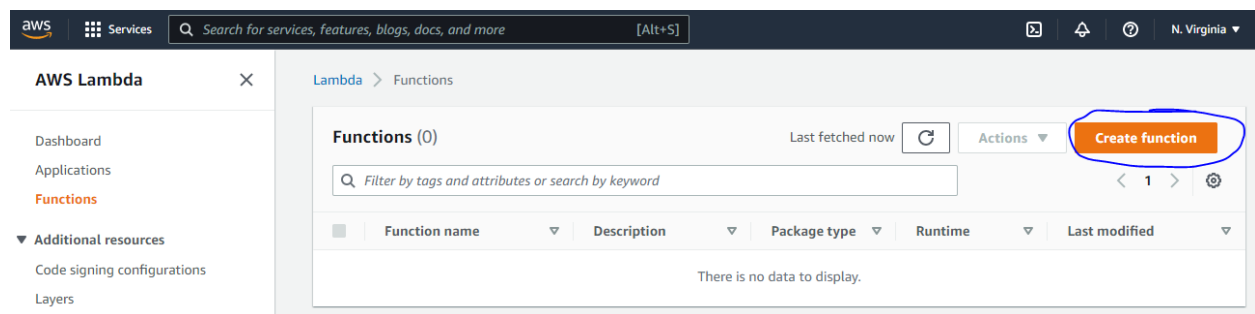


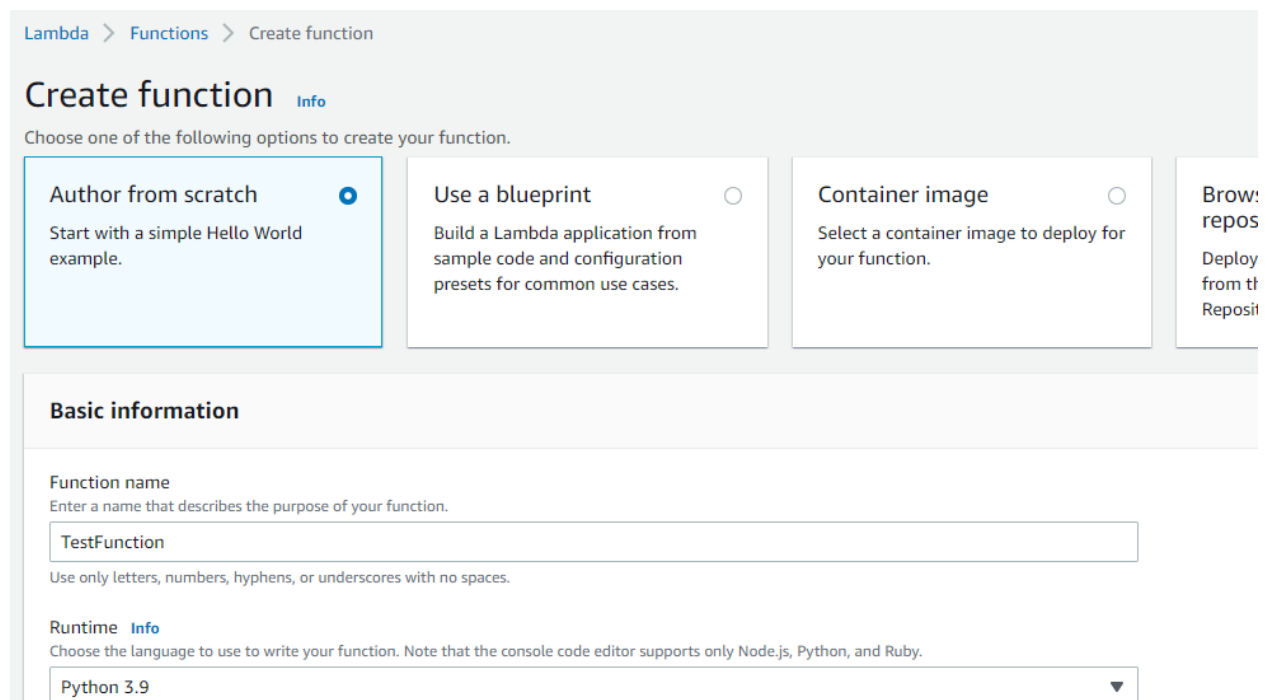
## Creating REST API using Lambda

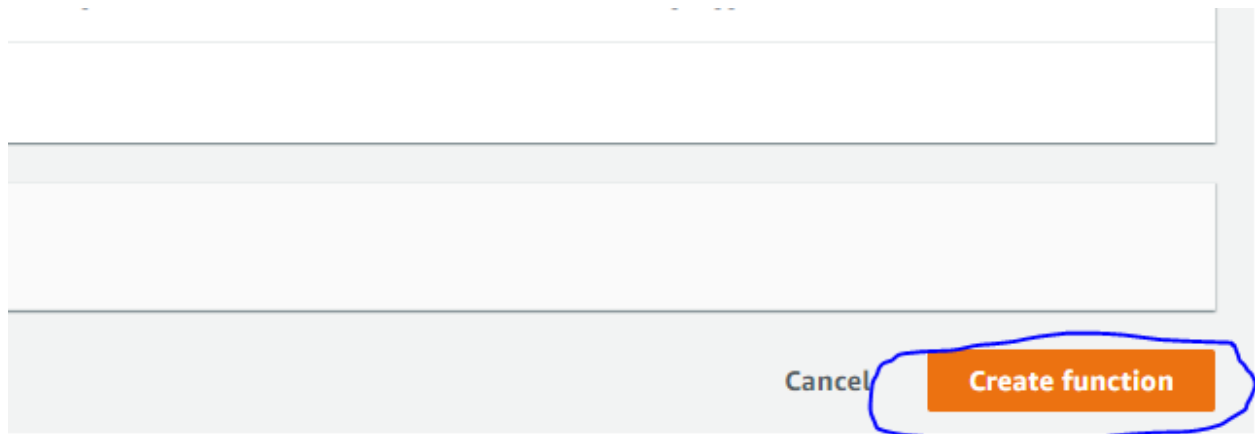


## Click on Lambda



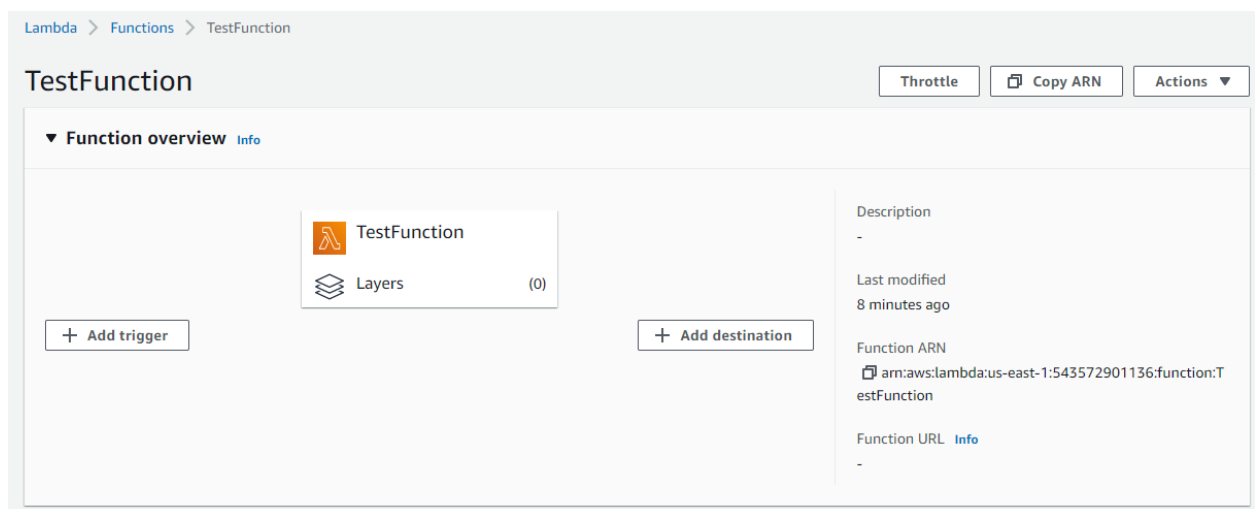
## We are going to create a new function here



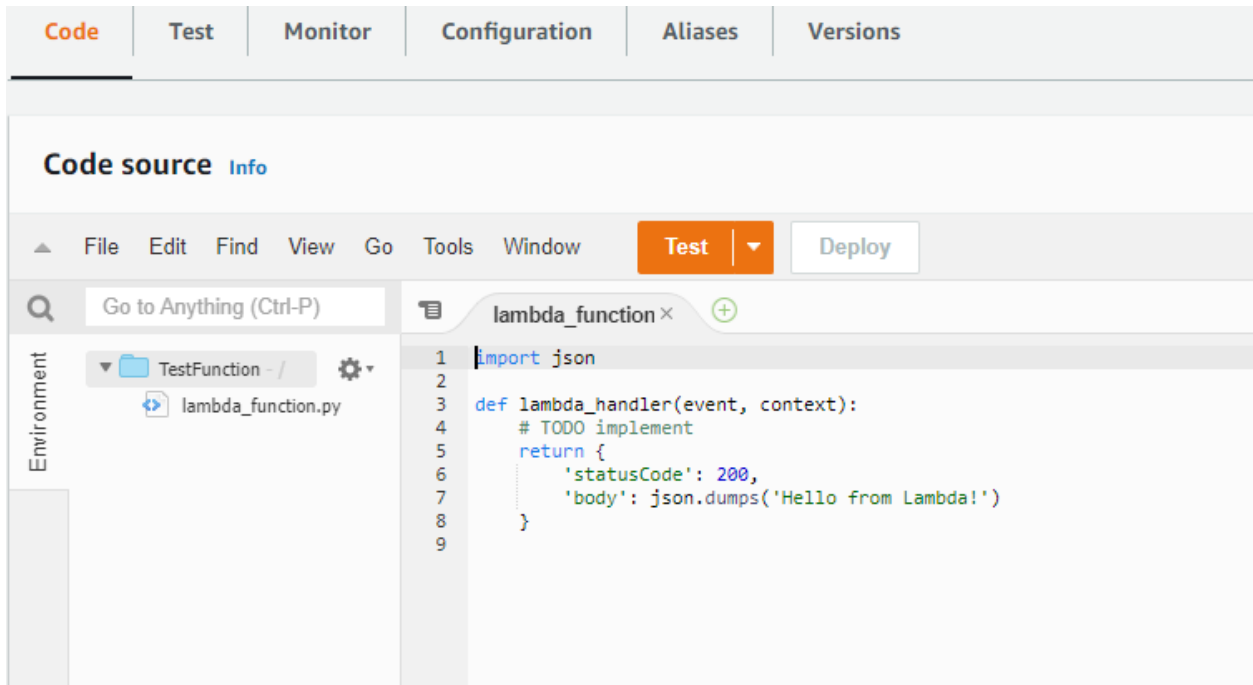


Click on Create function now.

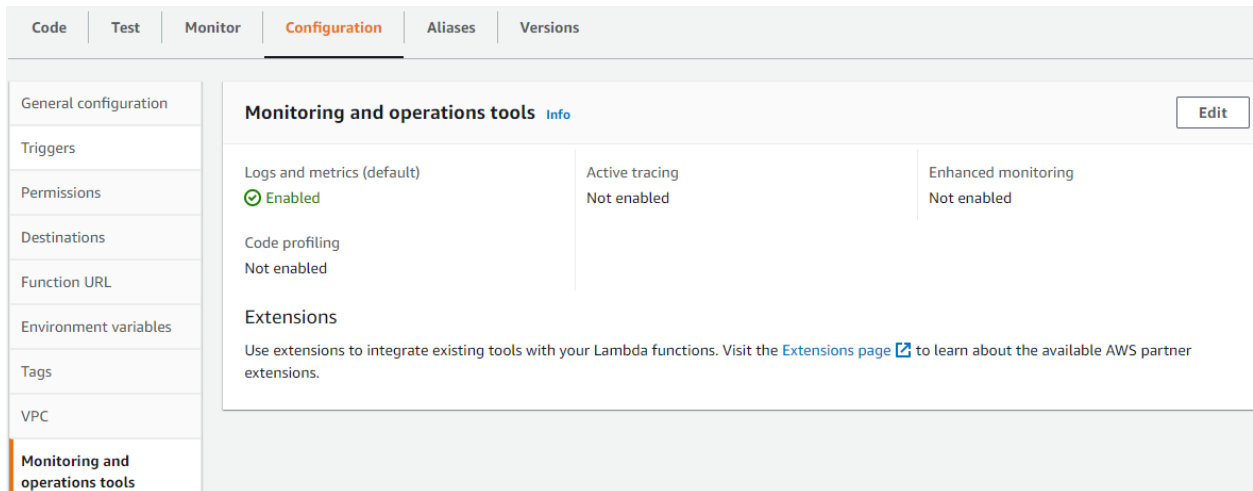
Successfully created the function **TestFunction**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".



See the sample code which is generated by default



## Configuration



Testing the function, for that we need to create an event

► Function overview [Info](#)

Code

Test

Monitor

Configuration

Aliases

Versions

Test event

SaveTest

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event

☐ Edit saved event

Event name

MyEventName

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

## Save it and then click on Test

Event name

TestingFunction

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Event JSON

Format JSON

```
1 {  
2   "key1": "value1",  
3   "key2": "value2",  
4   "key3": "value3"  
5 }
```

► Function overview [Info](#)

Code

Test

Monitor

Configuration

Aliases

Versions

Test event

SaveTest

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event

☐ Edit saved event

🟢 The test event **TestingFunction** was successfully saved.

Lambda > Functions > TestFunction

## TestFunction

Throttle Copy ARN Actions ▼

► **Function overview** Info

Code **Test** Monitor Configuration Aliases Versions

🟢 Execution result: succeeded (logs) ✕

► Details

Test event Delete Save **Test**

🟢 Execution result: succeeded (logs)

▼ Details

The area below shows the last 4 KB of the execution log.

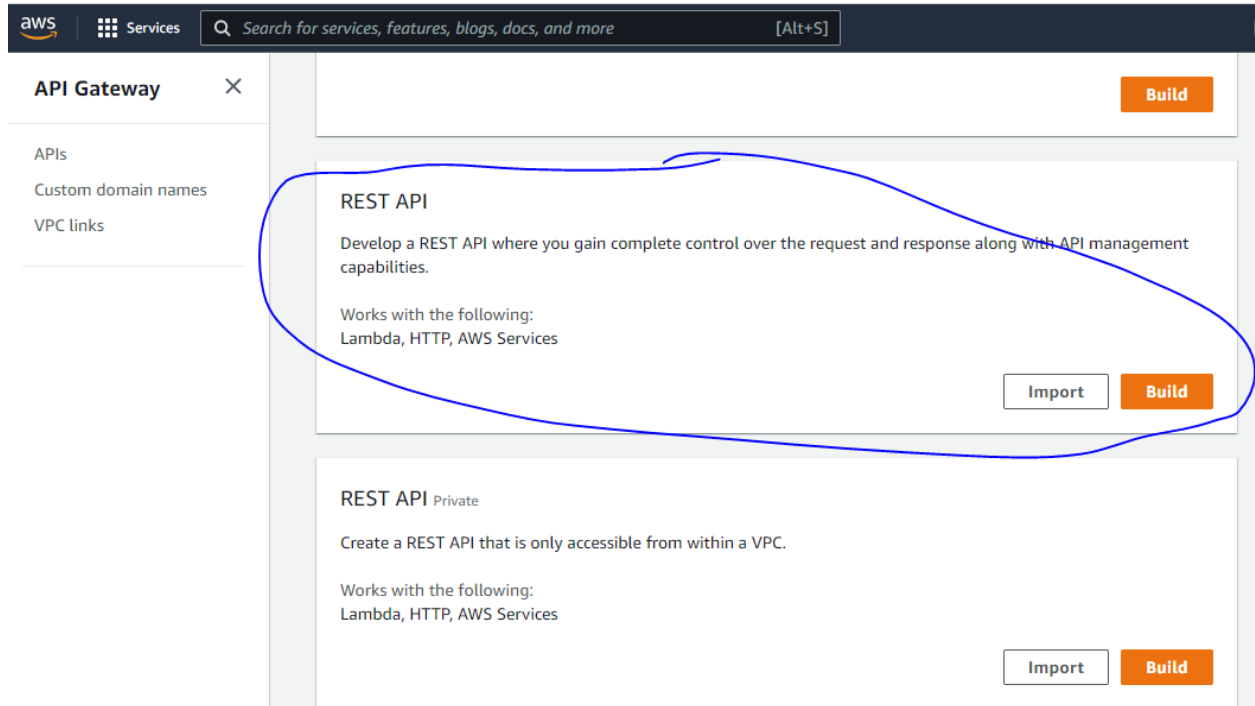
```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

### Summary

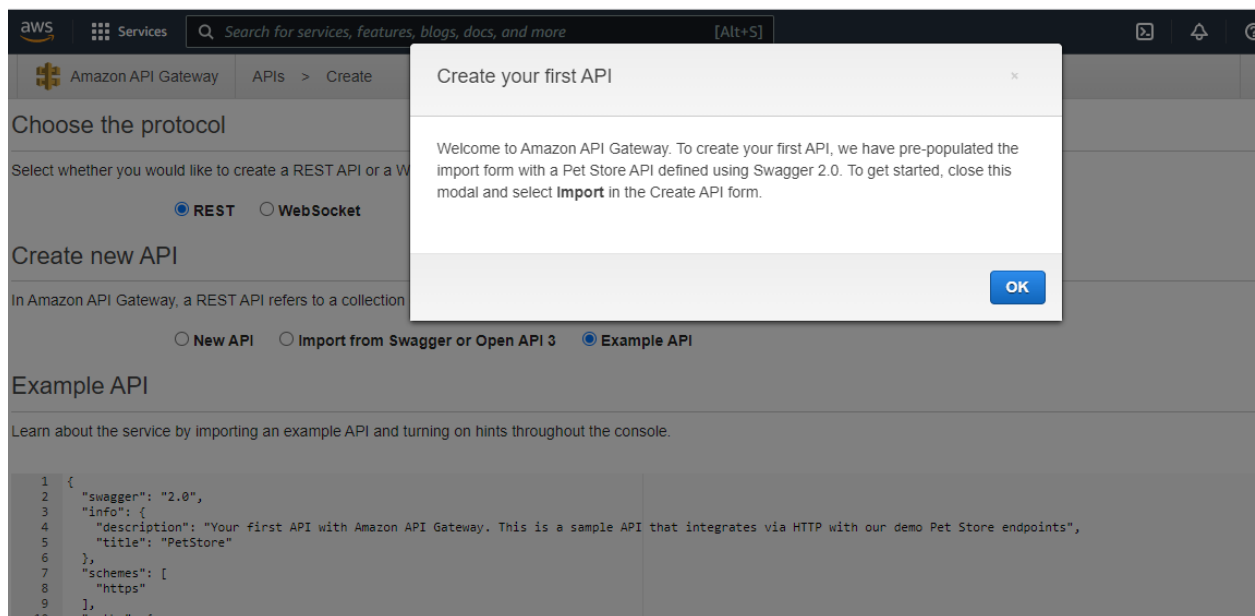
Code SHA-256	Request ID
f106ZIRH/KN6Ra3twvdRIUYaxv182Tjx0qNWNiKIhl=	860621f7-3aa9-4db8-8bb9-97d8990cbdac
Init duration	Duration
104.17 ms	0.93 ms
Billed duration	Resources configured
1 ms	128 MB

**Now the lambda function is ready, we have create an API gateway:**

**Navigate to API gateway service:**



Now we are going to create an API using the REST API, click on build



Use create new options and click on create

## Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

☒ REST ☐ WebSocket

## Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

☒ New API ☐ Import from Swagger or Open API 3 ☐ Example API

## Settings

Choose a friendly name and description for your API.



API name\* newapi  
Description  
Endpoint Type Regional

\* Required


Create API


**Click on Actions and create method and select GET method from dropdown:**


**Select the lambda function which is created before:**


 / - GET - Setup 


Choose the integration point for your new method.


**Integration type** ☒ Lambda Function 

☐ HTTP 


☐ Mock 


☐ AWS Service 

☐ VPC Link 



**Use Lambda Proxy integration** ☐ 

**Lambda Region**


**Lambda Function**  


**Use Default Timeout** ☒ 


**Save**


 / - GET - Setup 


Choose the integration point for your new method.


**Integration type** ☒ Lambda Function 

☐ HTTP 


☐ Mock 


☐ AWS Service 

☐ VPC Link 

**Use Lambda Proxy integration** ☒ 

**Lambda Region**

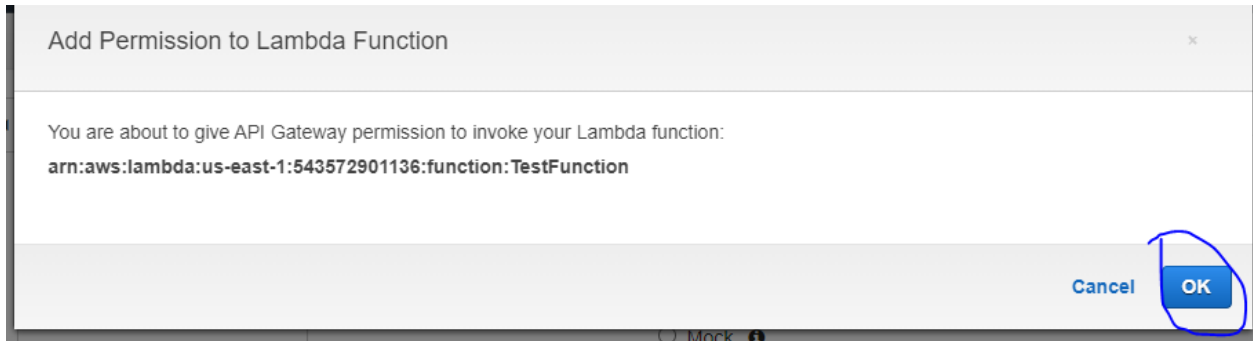
**Lambda Function**  

**Use Default Timeout** ☒ 

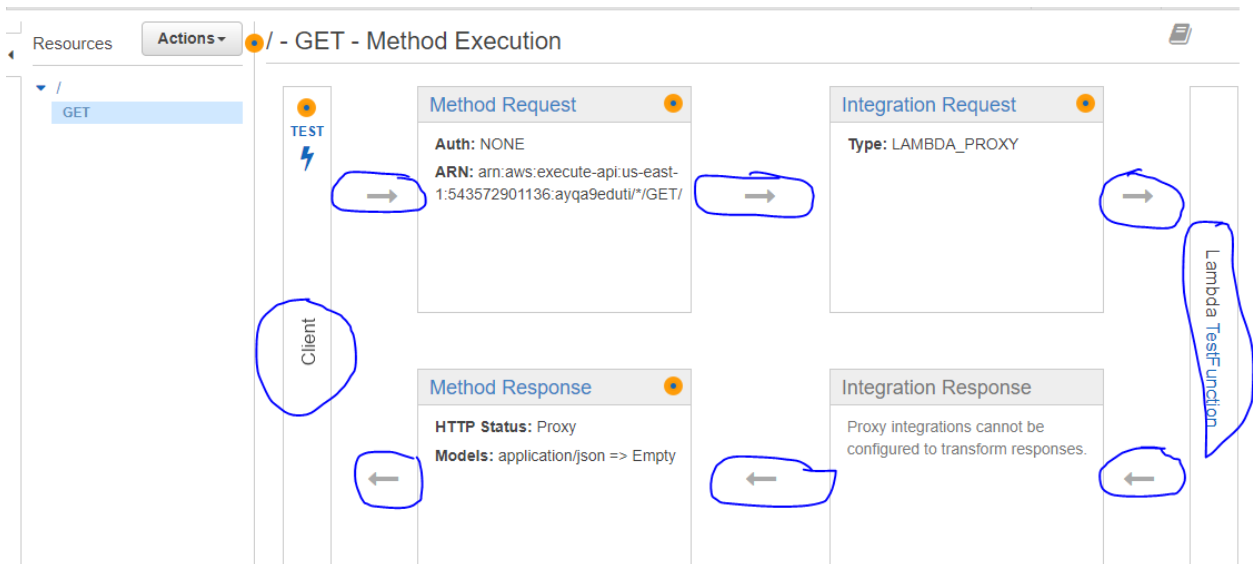
**Save**

You are giving the permission to lamdba function and that is what we needed.



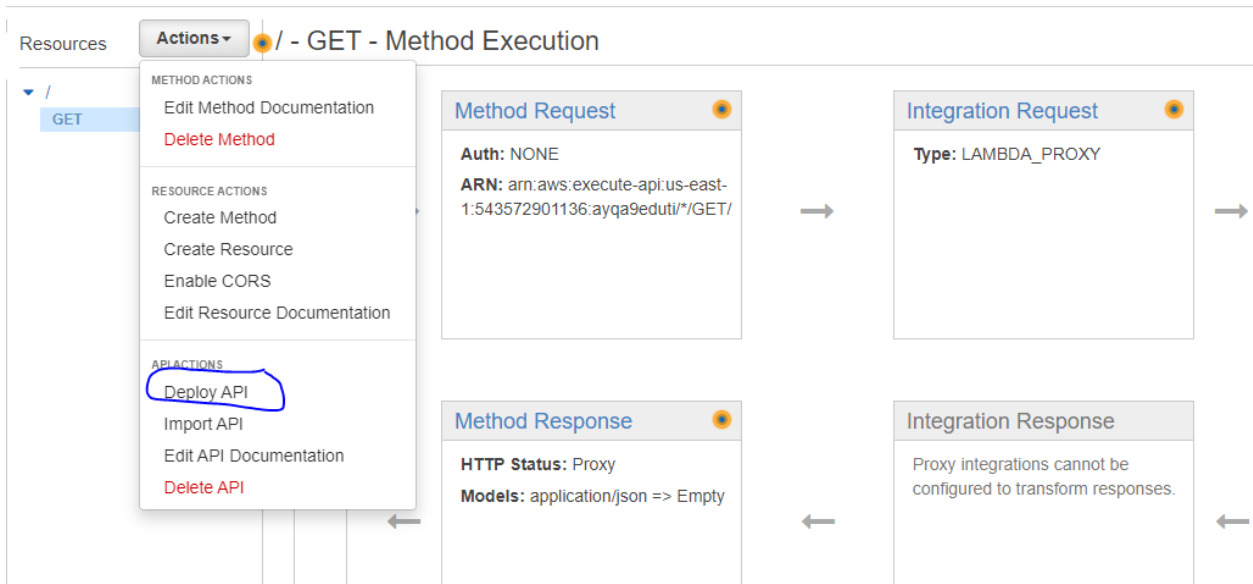


Once it is saved u get a pane opened with all the options



The flow is shown here, when it is tested from client it goes to method request(GET) and Integration Request and hits the lambda function and then it gives the response back : here Integration Response is greyedout and it returns to the Client.

Now click on Deploy 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.

Deployment stage [New Stage] ▾

Stage name\* prod

Stage description

Deployment description

Cancel Deploy

Stages **Create** prod Stage Editor **Delete Stage** **Configure Tags**

► prod

Invoke URL: <https://ayqa9eduti.execute-api.us-east-1.amazonaws.com/prod>

**Settings** Logs/Tracing Stage Variables SDK Generation Export Deployment History Documentation History

Canary

Cache Settings

Enable API cache ☐

Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. [Read more about API Gateway throttling](#)

Enable throttling ☒ ⓘ

Rate  requests per second

← → ↻ 🔒 [ayqa9eduti.execute-api.us-east-1.amazonaws.com/prod](https://ayqa9eduti.execute-api.us-east-1.amazonaws.com/prod)

📁 AWS 📁 Kubernetes 📁 Terraform 📺 (193) How to valida... 📺 How To Integrate O... 📁

"Hello from Lambda!"

<https://ayqa9eduti.execute-api.us-east-1.amazonaws.com/prod>

**Adding a parameter to the lambda function:**

**Create a new lambda function:**

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.

TestFunction2

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 3.9

**Architecture** [Info](#)  
Choose the instruction set architecture you want for your function code.

☒ x86\_64

## Select the existing Role:

**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.

Q |

service-role/Test-role-os1w404o

service-role/TestFunction-role-nwgbdyvy

Cancel **Create function**

## TestFunction2 Created:

## TestFunction2

### ▼ Function overview [Info](#)



TestFunction2



Layers

(0)

+ Add trigger

+ Add destination

### Code source [Info](#)

File Edit Find View Go Tools Window

Test

Deploy

Changes not deployed

Go to Anything (Ctrl-P)

TestFunction2 - /

lambda\_function.py



lambda\_function ×



```
1 import json
2
3 def lambda_handler(event, context):
4     myString = event[myString]
5     # TODO implement
6     return {
7         'statusCode': 200,
8         'body': json.dumps(myString)
9     }
10
```

```
import json
```

```
def lambda_handler(event,context):
```

```
    myString = event[myString]
```

```
    return{
```

```
        'statusCode': 200,
```

```
        'body': json.dumps(myString)
```

```
    }
```

Now, we are going to add resource method and parameter to the REST api in the API gateway:

## Creating Resource:

Resources Actions ▾ New Child Resource

RESOURCE ACTIONS

- Create Method
- Create Resource
- Enable CORS
- Edit Resource Documentation

API ACTIONS

- Deploy API
- Import API
- Edit API Documentation
- Delete API

Create a new child resource for your resource.

☒ proxy resource ⓘ

Resource Name\* My Resource

Resource Path\* / my-resource

You can add path parameters using brackets. For example, the resource path {username} represents a path parameter called 'username'. Configuring /(proxy+) as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

API Gateway CORS ☐ ⓘ

\* Required

Cancel Create Resource

Resources Actions ▾ /myresource - GET - Setup

Choose the integration point for your new method.

Integration type ☒ Lambda Function ⓘ

- ☐ HTTP ⓘ
- ☐ Mock ⓘ
- ☐ AWS Service ⓘ
- ☐ VPC Link ⓘ

Use Lambda Proxy integration ☒ ⓘ

Lambda Region us-east-1 ▾

Lambda Function TestFunction2 ⓘ

Use Default Timeout ☒ ⓘ

Save

### Add Permission to Lambda Function

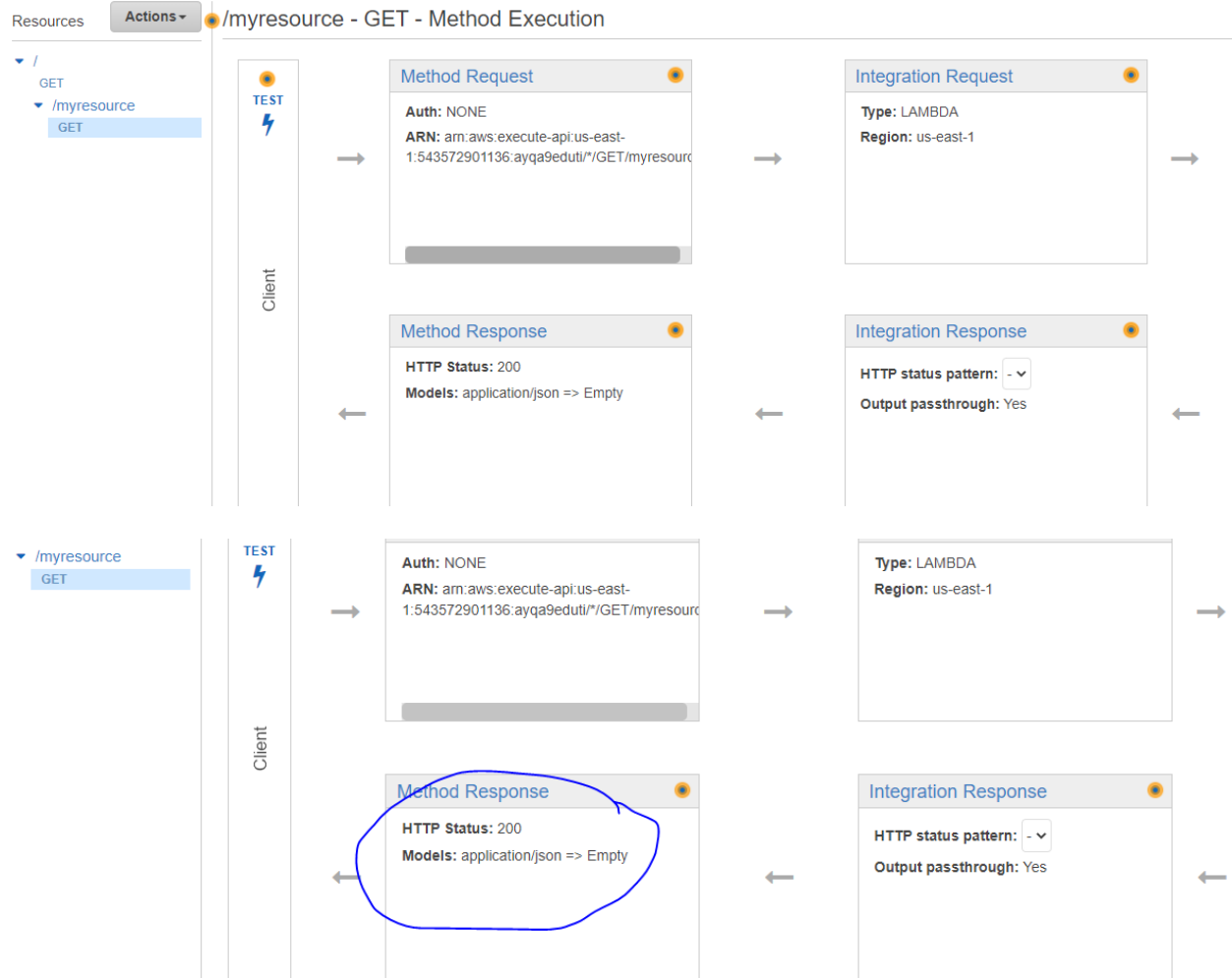
You are about to give API Gateway permission to invoke your Lambda function:

arn:aws:lambda:us-east-1:543572901136:function:TestFunction2

Cancel

OK

Now see the pane where we can see the integration pane is not greyed out:



Then deploy the API.

Now:

Adding the API gateway from the lambda function:

Go to Lambda function, add trigger and select api gateway:

# TestFunction

## ▼ Function overview [Info](#)



TestFunction



Layers

(0)



API Gateway



Add trigger



Add destination



API Gateway

api application-services aws serverless

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. [Learn more](#)

### API

Create a new API or attach an existing one.

Create an API

### API type



HTTP API

Create an HTTP API.



REST API

Create a REST API.

### Security

Configure the security mechanism for your API endpoint.

Open

### ► Additional settings

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger.

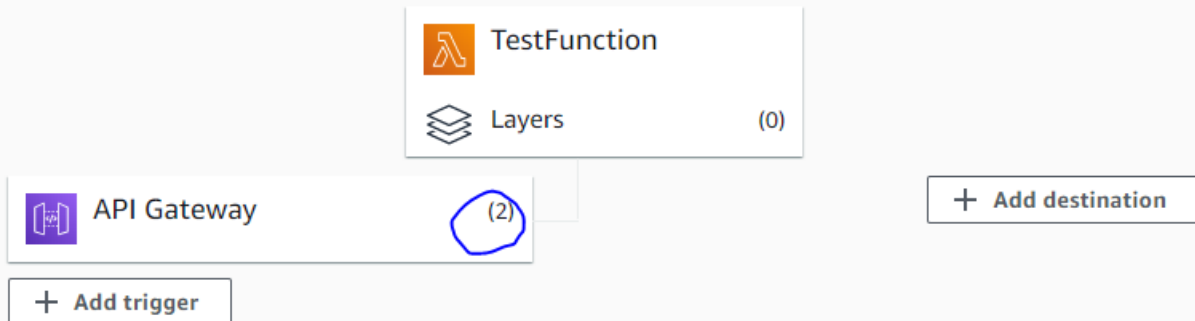
**Now, we have 2 API for the lambda function: TestFunction**



# TestFunction

✓ The trigger TestFunction-API was successfully added to function TestFunction. The function is now receiving events

## ▼ Function overview [Info](#)



### API Gateway: [newapi](#)

arn:aws:execute-api:us-east-1:543572901136:ayqa9eduti/\*/GET/

API endpoint: <https://ayqa9eduti.execute-api.us-east-1.amazonaws.com/prod/>

#### ▼ Details

API type: **REST**

Authorization: **NONE**

Method: **GET**

Resource path: **/**

Stage: **prod**



#### API Gateway: **TestFunction-API**

arn:aws:execute-api:us-east-1:543572901136:tszm3p4268/\*/\*/TestFunction

API endpoint: <https://tszm3p4268.execute-api.us-east-1.amazonaws.com/default/TestFunction>

##### ▼ Details

API: **api-gateway/tszm3p4268/\*/\*/TestFunction**

API name: **TestFunction-API**

API type: **HTTP**

Authorization: **NONE**

Cross-origin resource sharing (CORS): **No**

Enable detailed metrics: **No**

Method: **ANY**

Resource path: **/TestFunction**

Security: **NONE**

Stage: **default**

**Both API Endpoint gives the same output.**

**Last Step is creating API using MOCK:**