

Aim-

Create a REST API with the Serverless Framework.

Procedure-

1. Create a new lambda function.

The screenshot displays the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and user information (N. Virginia, Mikil Lalwani). Below the navigation bar, the 'Lambda' service is selected, and the 'Functions' page is shown. The 'Functions (0)' section indicates no functions are currently listed. A 'Create function' button is visible. Below this, four deployment methods are presented: 'Author from scratch' (selected), 'Use a blueprint', 'Container image', and 'Browse serverless app repository'. The 'Basic information' section is expanded, showing fields for 'Function name' (rest_api), 'Runtime' (Python 3.9), and 'Architecture' (x86_64). The 'Permissions' section is also expanded, showing the 'Execution role' with the option 'Create a new role with basic Lambda permissions' selected. The footer contains a feedback link, copyright information (© 2022, Amazon Internet Services Private Ltd. or its affiliates), and links for Privacy, Terms, and Cookie preferences.

2. Write the code as needed and click on deploy.

Tools Window **Test** **Deploy** Changes not deployed

lambda_function x

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     first_name = event["queryStringParameters"]["first_name"]
6     last_name = event["queryStringParameters"]["last_name"]
7
8     app_response = {}
9
10    app_response['message'] = f'The details are {first_name} and {last_name}'
11    app_response['profession'] = 'cricket'
12    app_response['age'] = 40
13
14
15    responseObject = {}
16    responseObject['statusCode'] = 200
17    responseObject['headers'] = {}
18    responseObject['headers']['Content-Type'] = 'application/json'
19    responseObject['body'] = json.dumps(app_response)
20
21    return responseObject
```

3. Goto API Gateway and create a new Rest API.

rch for services, features, blogs, docs, and more [Alt+S]

Elastic Beanstalk AWS Amplify Lambda S3 API Gateway

Networking & Content Delivery

Amazon API Gateway

create, maintain, and secure APIs at any scale

Amazon API Gateway helps developers to create and manage APIs to back-end systems running on Amazon EC2, AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can generate custom client SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or services.

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

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*

Description

Endpoint Type ⓘ

* Required

4. Select create the resource and create the 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*

Resource Path*

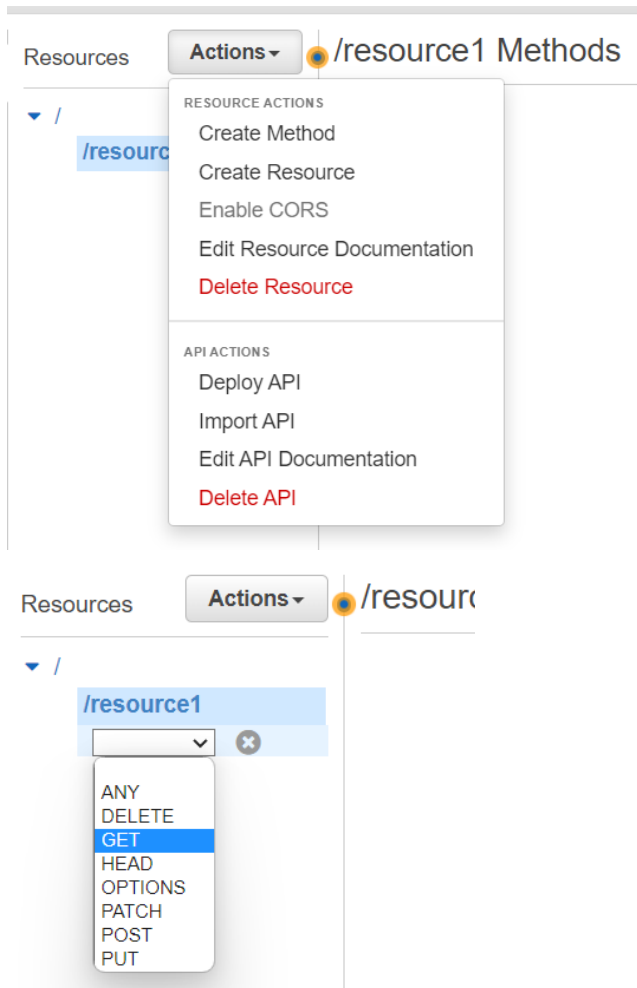
You can add path parameters using brackets. For example, the resource path {username} represents a path parameter called 'username'. Configuring /(proxy+) as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new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

API Gateway CORS ⓘ

* Required

[Cancel](#) [Create Resource](#)

5. Create a Get method.



/resource1 - 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 ☒ ⓘ

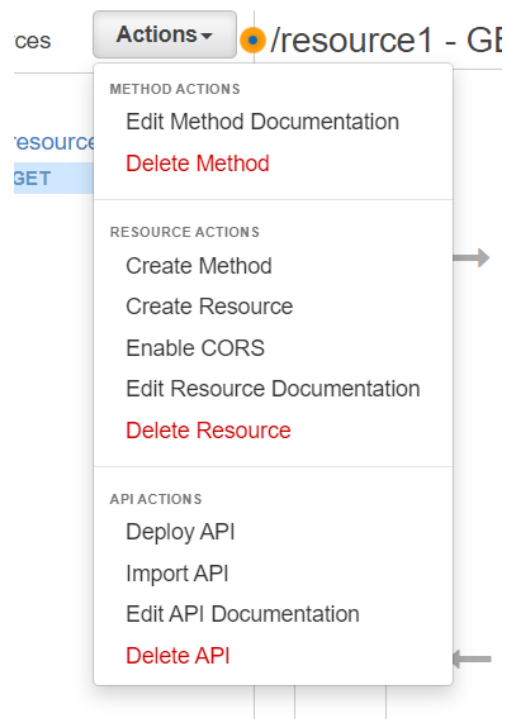
Add Permission to Lambda Function

You are about to give API Gateway permission to invoke your Lambda function:
`arn:aws:lambda:us-east-1:689179663899:function:rest_api`

Cancel

OK

6. Deploy the API now.



Deploy API

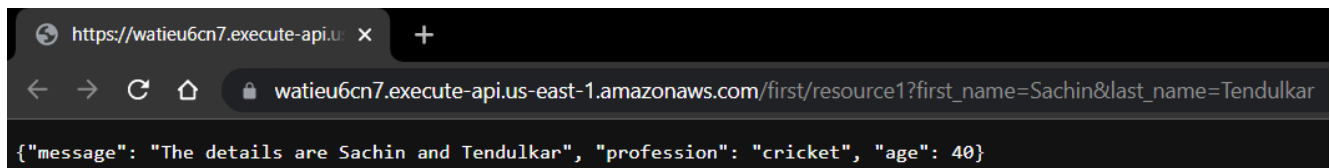
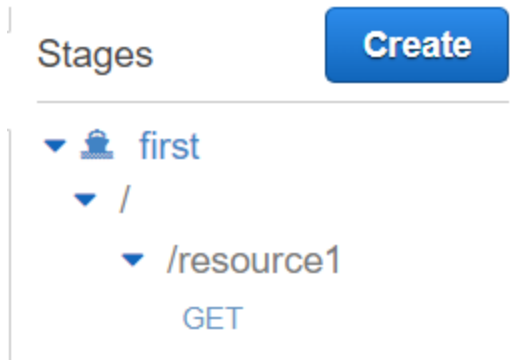
Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage	<input type="text" value="[New Stage]"/>
Stage name*	<input type="text" value="first"/>
Stage description	<input type="text"/>
Deployment description	<input type="text"/>

Cancel

Deploy

7. Now we can call the API.



We have successfully created a REST API.