

# How to build and deploy a REST-API with API Gateway and Lambda in AWS

## Theoretical Part

### What is a REST API?

A REST API (Representational State Transfer Application Programming Interface) is a type of web API that adheres to the principles and constraints of the REST architectural style. It allows systems to communicate over the internet using standard HTTP methods (e.g. GET, POST, PUT, DELETE) and is designed to be stateless and resource-oriented. REST APIs enable the creation, retrieval, updating, and deletion of resources, typically using JSON or XML for data representation. They are widely used for building web services and are known for their simplicity, scalability, and ease of use in web and mobile applications.

### What is AWS API Gateway?

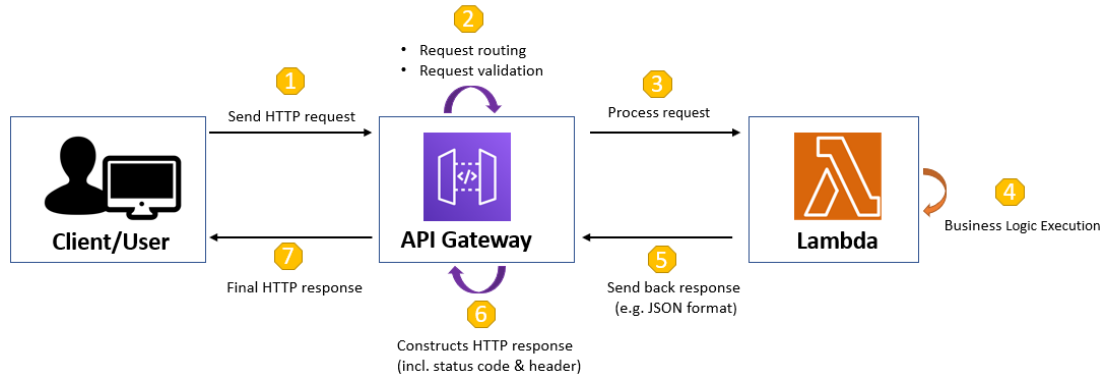
Amazon API Gateway is a fully managed service provided by AWS (Amazon Web Services) that allows you to create, publish, maintain, monitor, and secure APIs (Application Programming Interfaces) for your applications. It acts as a front-end to your backend services and simplifies the process of creating, deploying, and managing APIs.

### What is AWS Lambda?

AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS). It allows you to run code without provisioning or managing servers, making it easy to build scalable and event-driven applications.

### How API Gateway works together with AWS Lambda?

- **API Gateway(Front-end):** API Gateway serves as the front-end for your API. It's responsible for receiving and processing incoming HTTP requests from clients, such as web browsers and mobile apps.
- **Lambda(Backend)** AWS Lambda functions act as the back-end of your API. They are the backend components responsible for executing the business logic, processing data, and generating responses. Lambda functions are triggered by API Gateway when requests come in.



## Hands-on and technical part

- Prerequisite: Your AWS account has to be setup

In [ ]:

### Step 1 - Create a Lambda Function

Lambda > Functions > Create function

#### Create function [Info](#)

AWS Serverless Application Repository applications have moved to [Create application](#).

☒ 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

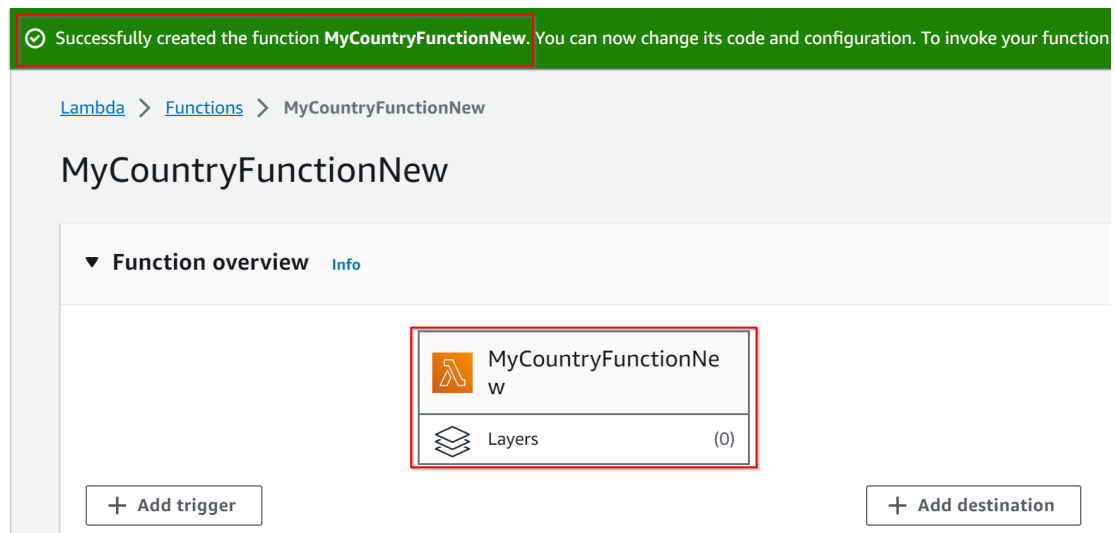
**Function name**  
 Enter a name that describes the purpose of your function.

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

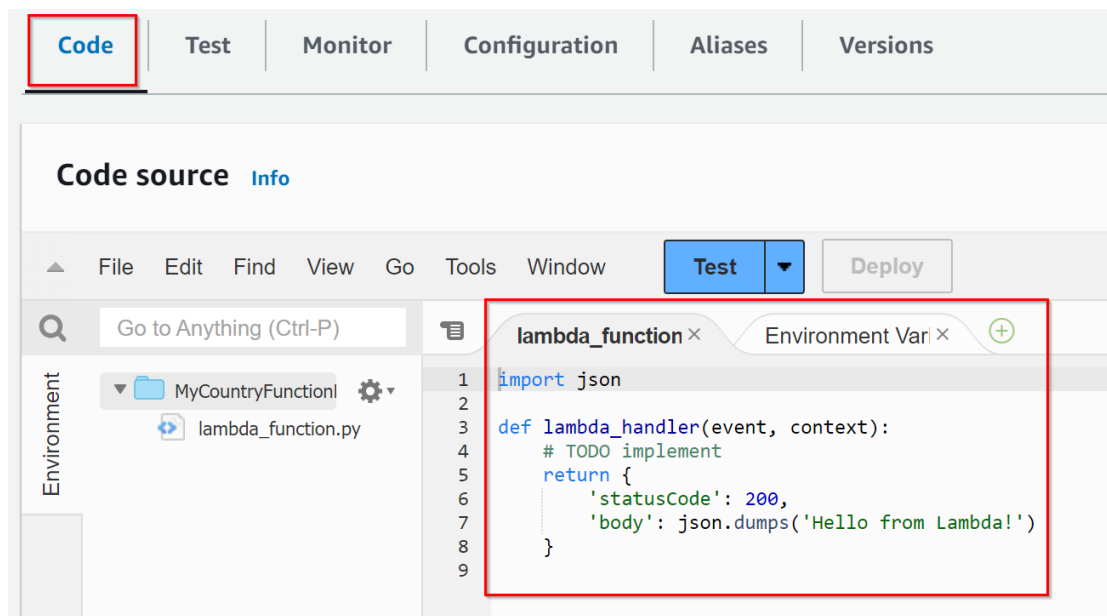
**Runtime** [Info](#)  
 Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

**Architecture** [Info](#)  
 Choose the instruction set architecture you want for your function code.  
☒ x86\_64  
☐ arm64

### Step 2 - Lambda Function successfully created



### Step 3 - Scroll Down in your Lambda Function until you see the Default Code Snippet



### Step 4 - Replace Default Code with your Code (business logic)

AWS Lambda function that processes an incoming HTTP request. This Demo-Code does the following:

- Defines a dictionary called "countries" that contains information about different countries, including their capitals and populations
- Countries provided: Germany, Spain, Italy
- Extracts the value of the "country" query string parameter from the incoming request using the "event" object.
- Checks if the requested country exists in the "countries" dictionary. If it does, it generates an HTTP response with a status code of 200 and provides information

about the requested country in JSON format.

- If the requested country is not found in the dictionary, it generates an HTTP response with a status code of 404 and returns an error message in JSON format, indicating that the requested country is not available.
- The Lambda function returns the HTTP response, which can be sent back to the client that made the request.
- In essence, this code processes requests for country information, providing details if the country is found in the dictionary or returning an error message if it's not.

```
In [ ]: import json

def lambda_handler(event, context):
    # Countries
    countries = {
        "Germany": {
            "capital": "Berlin",
            "population": 83000000
        },
        "Spain": {
            "capital": "Madrid",
            "population": 46000000
        },
        "Italy": {
            "capital": "Rome",
            "population": 60000000
        }
    }

    # Requested country from the query string
    requested_country = event.get("queryStringParameters", {}).get("country")

    if requested_country in countries:
        response = {
            "statusCode": 200,
            "body": json.dumps(countries[requested_country]),
            "headers": {
                "Content-Type": "application/json"
            }
        }
    else:
        response = {
            "statusCode": 404,
            "body": json.dumps({"message": "Country not there"}),
            "headers": {
                "Content-Type": "application/json"
            }
        }

    return response
```

## Step 5 - Create a REST-API in API Gateway

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

## Step 6 - Provide details for your REST 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

Country\_API

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

## Step 7 - Create a GET-method in API Gateway and integrate it with your Lambda-Function

API Gateway > APIs > Resources - Country\_API (21431b9gl)

Resources

API actions Deploy API

Create resource

Resource details

Update documentation Enable CORS

Path / Resource ID 9u5ocpq753

Methods (0)

Delete Create method

Method type	Integration type	Authorization	API key
No methods			
No methods defined.			

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.

eu-central-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


## Step 8 - Make sure API Gateway is connected with Lambda


[Lambda](#) > [Functions](#) > MyCountryFunctionNew

## MyCountryFunctionNew

▼ **Function overview** [Info](#)

 **MyCountryFunctionNew**

 Layers (0)

 **API Gateway**

[+ Add trigger](#)

## Step 9 - Test your function in API Gateway

- Positive Result (provide Germany in your country query string)

Method request | Integration request | Integration response | Method response | **Test**

### Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

**Query strings**  
country=Germany

**Headers**  
Enter a header name and value separated by a colon (:). Use a new line for each header.  
header1:value1  
header2:value2

**Client certificate**  
No client certificates have been generated.

**Test**

**/ - GET method test results**

Request  
/?country=Germany

Latency  
233

Response body  

```
{
  "capital": "Berlin",
  "population": 83000000
}
```

Response headers  

```
{
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-653cdb93-bfa740a9cfa8c09348cbf081;Sampled=0;lineage=7716b283:0"
}
```

## Step 10 - Provide a country which was not(!) provided in Step 4

- we did not provide Greece in the dictionary in Step no.4 - Therefore we receive the message "Country not there"

**/ - GET method test results**

Request  
/?country=Greece

Latency  
42

Response body  

```
{
  "message": "Country not there"
}
```

Response headers  

```
{
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-653cddb5-de0848c9b9cee081360b0804;Sampled=0;lineage=7716b283:0"
}
```

## Step 11 - Deploy your API in the Development-Stage

- When you click on "Deploy API" you can set a new stage or deploy in an existing one. In this example we create a dev stage

API Gateway > APIs > Resources - Country\_API (2i43lbgll)

Resources

Create resource

/ GET

**/ - GET - Method execution**

ARN  
arn:aws:execute-api:eu-central-1:284545156841:2i43lbgll/GET/

Resource ID  
9u5ocpq733

Client

Method request

Integration request

Method response

Integration response  
Proxy integration

Lambda integration

API actions

Deploy API

Update documentation

Delete



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

development

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

Deployment description

This is a simple deployment of our API in the dev stage

Cancel

Deploy

## Step 12 - Invoke the deployed API in your browser

API Gateway

>

APIs

>

Country\_API (2i43lbgli)

>

Stages

Stages

development

Stage details

Stage name

development

API cache

Inactive

Invoke URL

https://2i43lbgli.execute-api.eu-central-1.amazonaws.com/development

- chose "Spain" in your query (part of the dictionary we created in Step no.4)

←

→

↻

2i43lbgli.execute-api.eu-central-1.amazonaws.com/development/?country=Spain

Fruit Shop API

RobotFramework

DemoWebsites

VTG AG

UiPath

JiraAPI

Recsyn

Localhost

Django

```

1 // 20231028120609
2 // https://2i43lbgli.execute-api.eu-central-1.amazonaws.com/development/?country=Spain
3
4 {
5   "capital": "Madrid",
6   "population": 46000000
7 }

```