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?

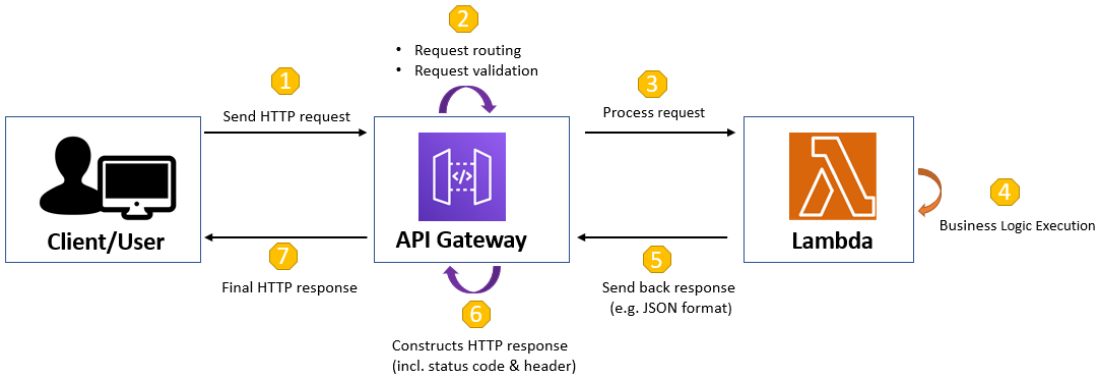
Amazon API Gateway is a fully managed service provided by AWS (Amazon Web Services) that allows you to create, publish, maintain, monitor, and secure APIs (Application Programming Interfaces) for your applications. It acts as a front-end to your backend services and simplifies the process of creating,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

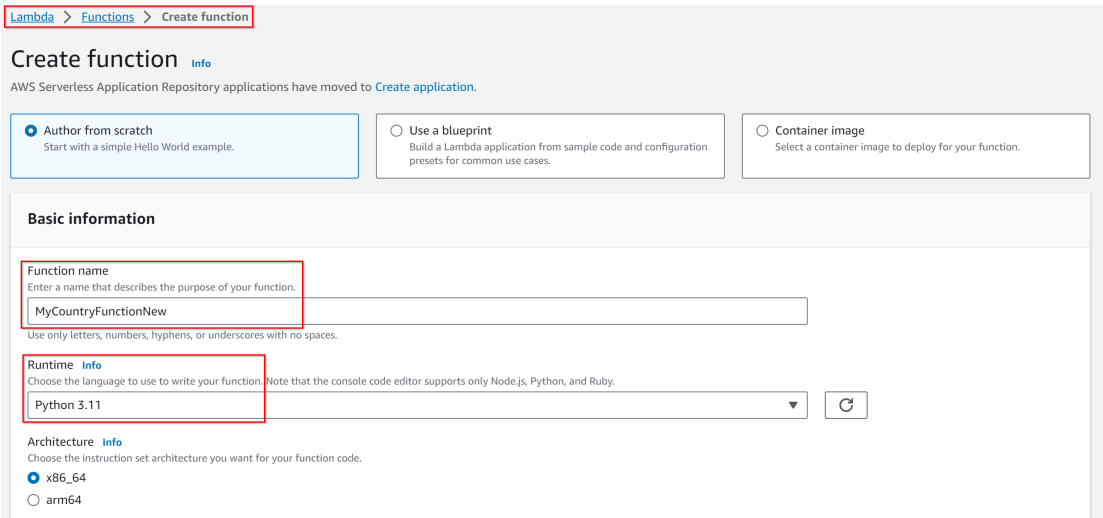


Hands-on and technical part

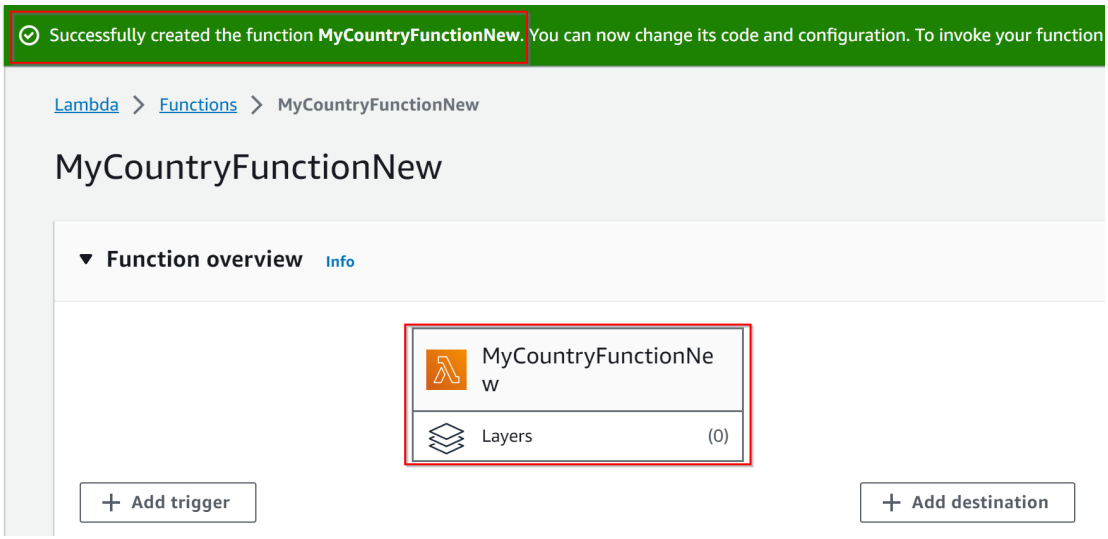
- Prerequisite: Your AWS account has to be setup

In [ ]:

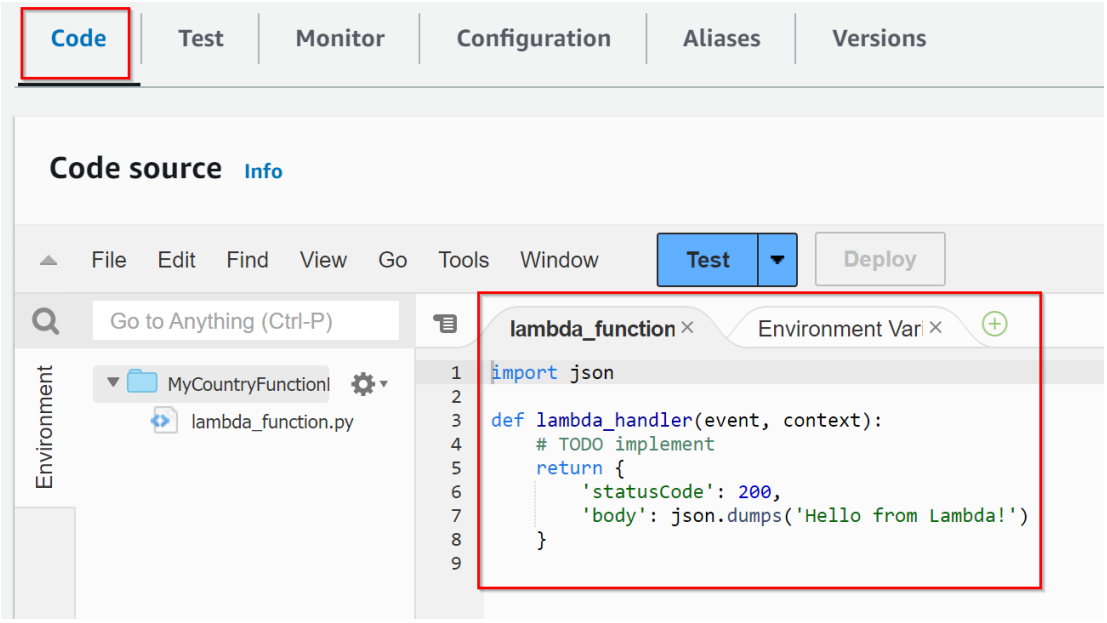
Step 1 - Create a Lambda Function



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:

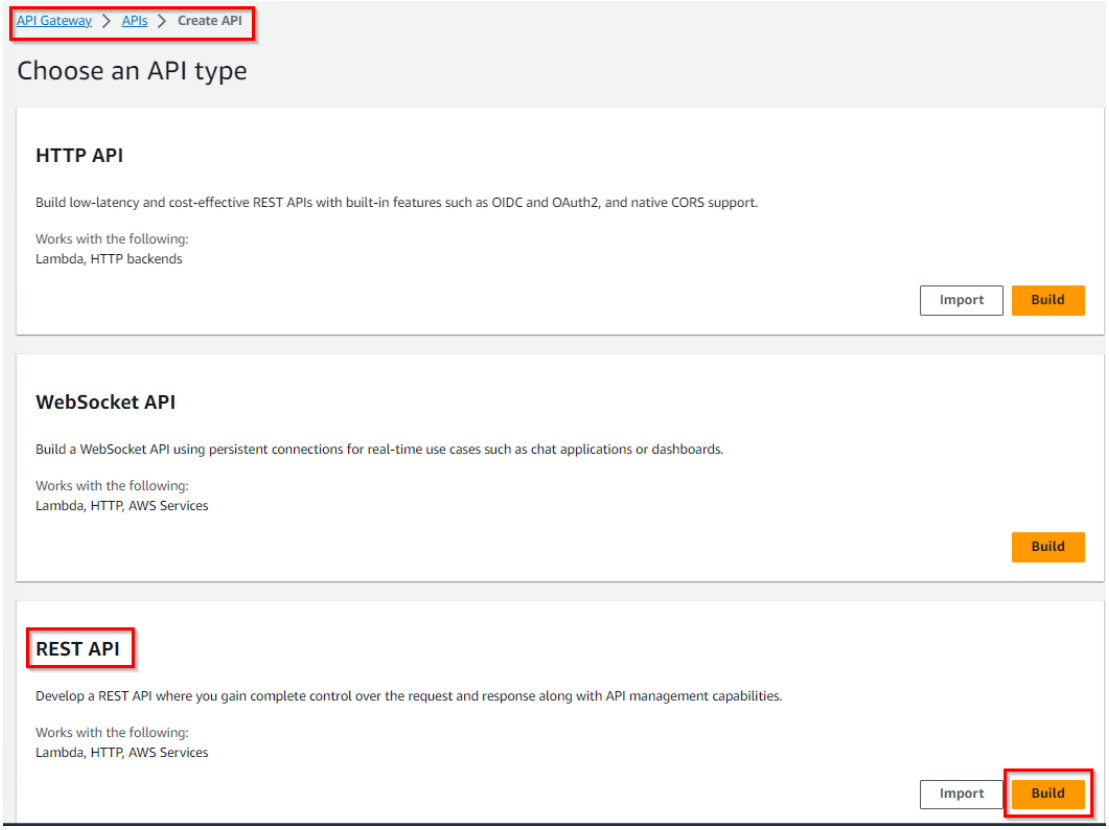
- Defines a dictionary called "countries" that contains information about different countries, including their capitals and populations
- Countries provided: Germany, Spain, Italy
- Extracts the value of the "country" query string parameter from the incoming request using the "event" object.
- Checks 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.

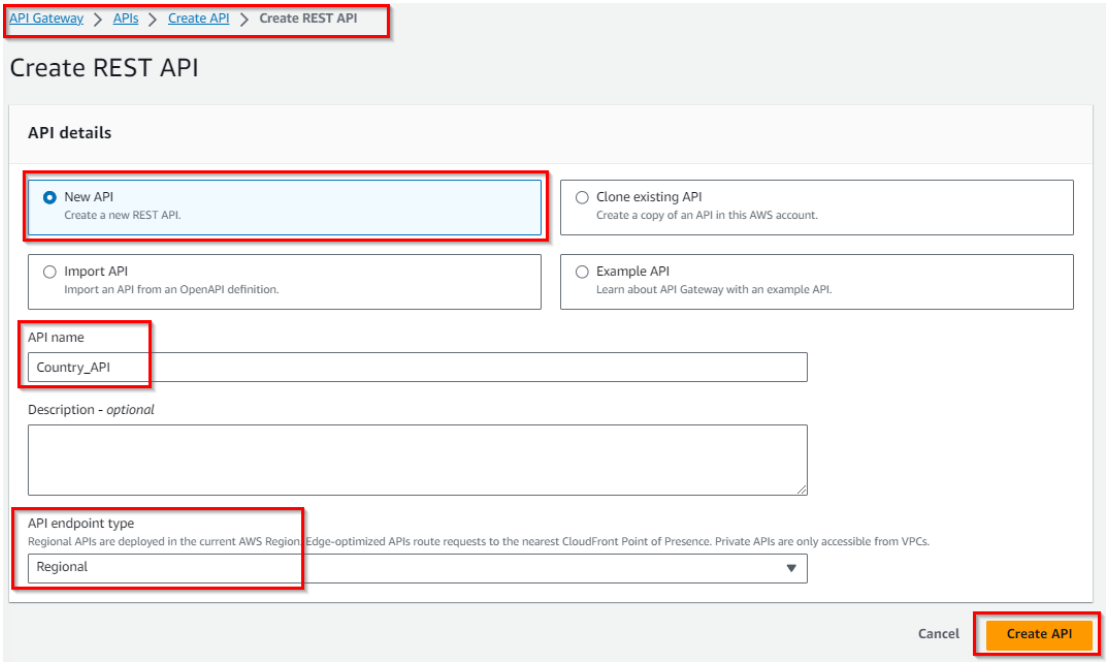
- If the requested country is not found in the dictionary, it generates an HTTP response with a status code of 404 and returns an error message in JSON format, indicating that the requested country is not available.
- The Lambda function returns the HTTP response, which can be sent back to the client 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
```

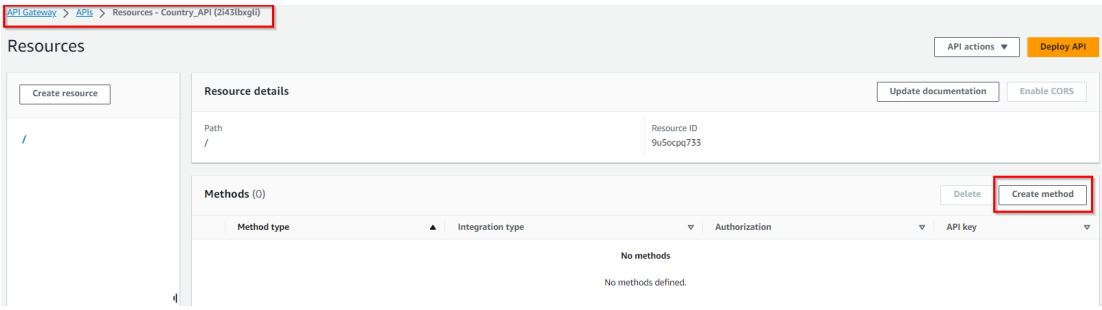
Step 5 - Create a REST-API in API Gateway

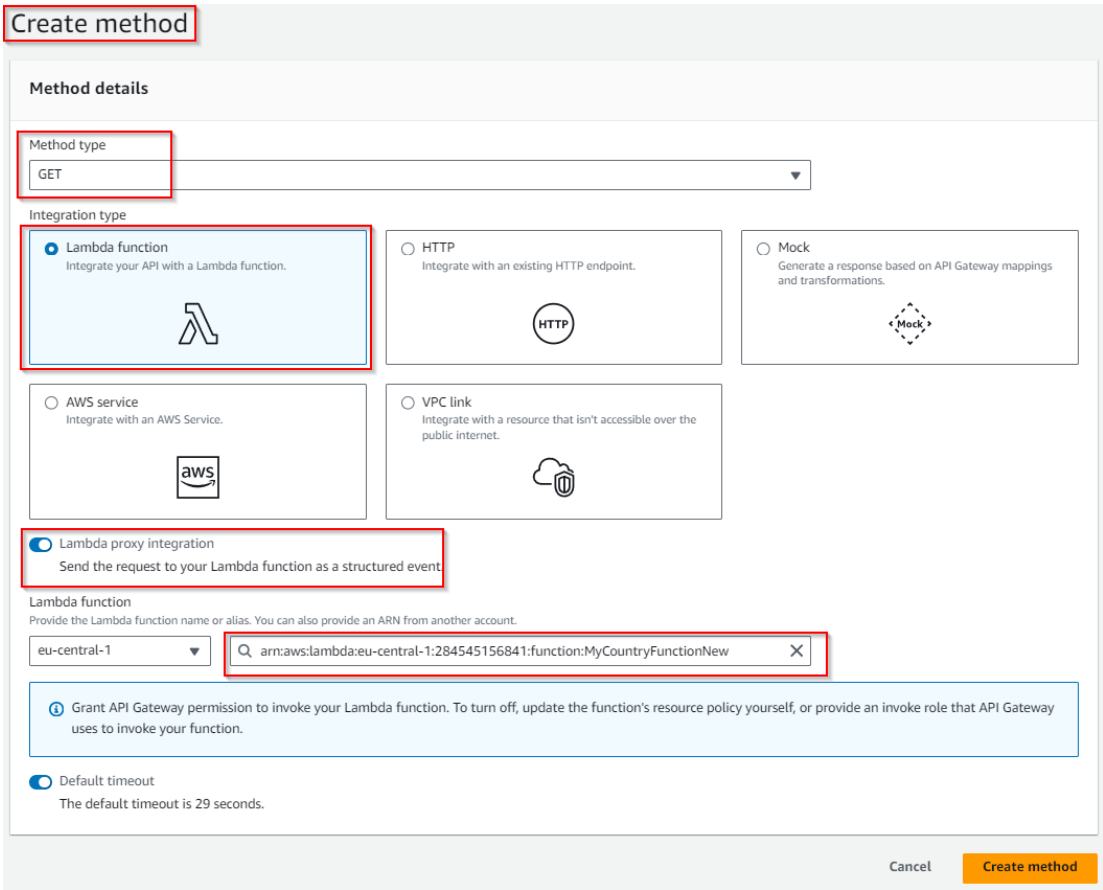


Step 6 - Provide details for your REST API

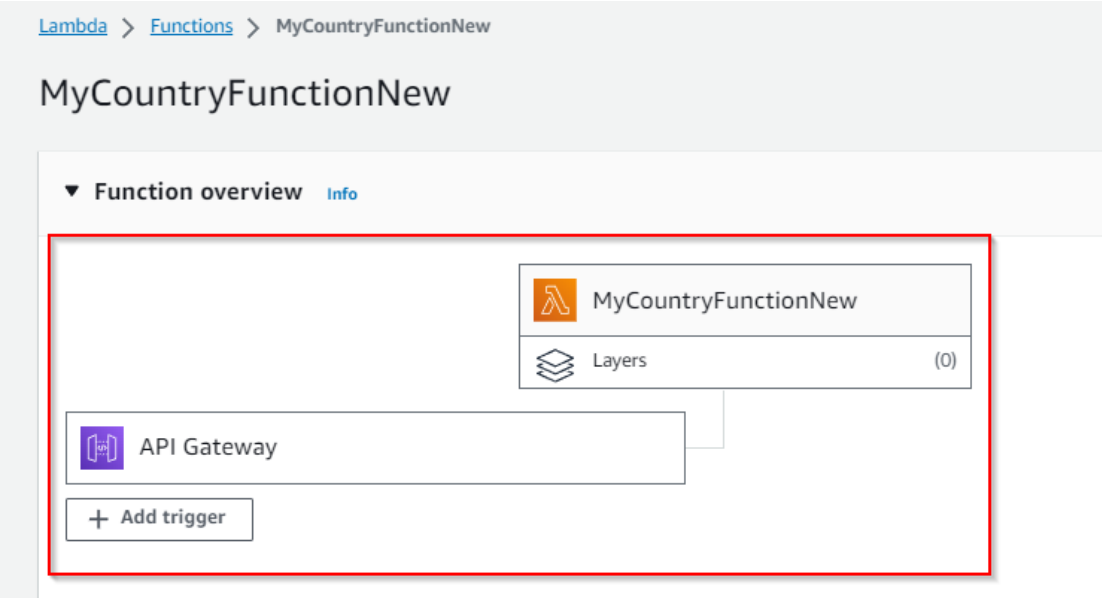


Step 7 - Create a GET-method in API Gateway and integrate it with your Lambda-Function



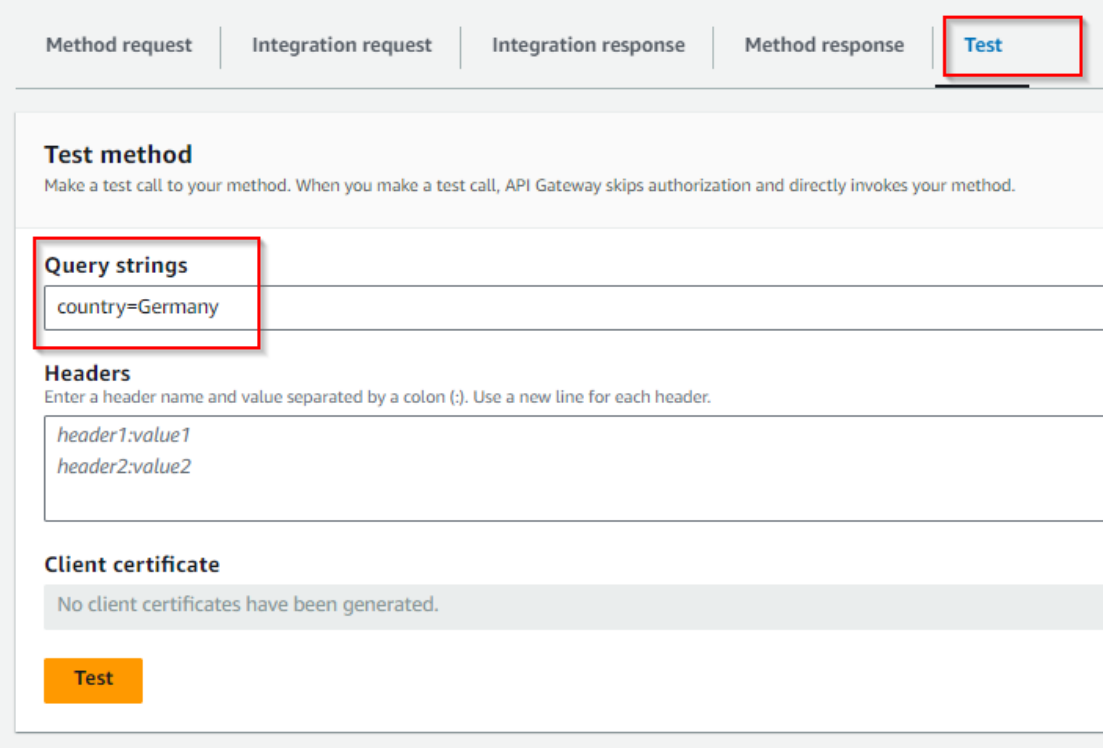


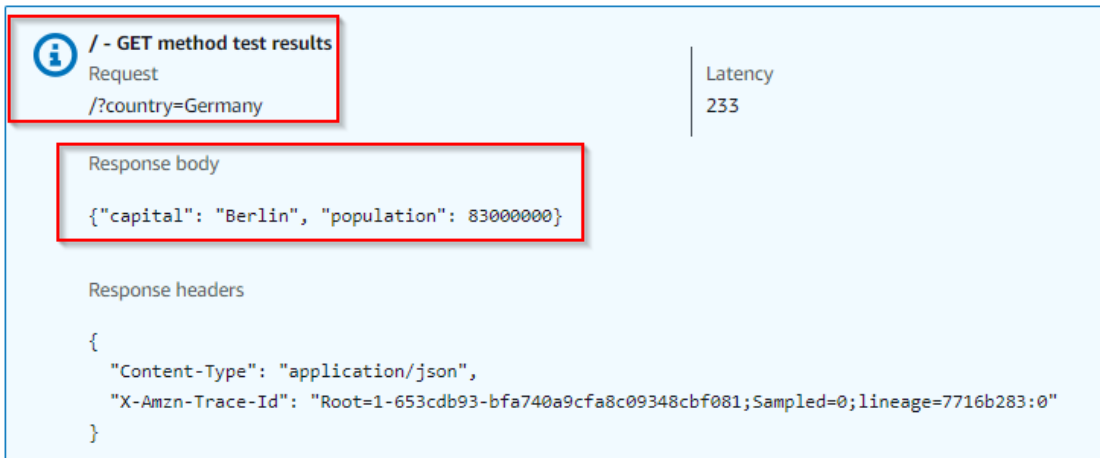
Step 8 - Make sure API Gateway is connected with Lambda



Step 9 - Test your function in API Gateway

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



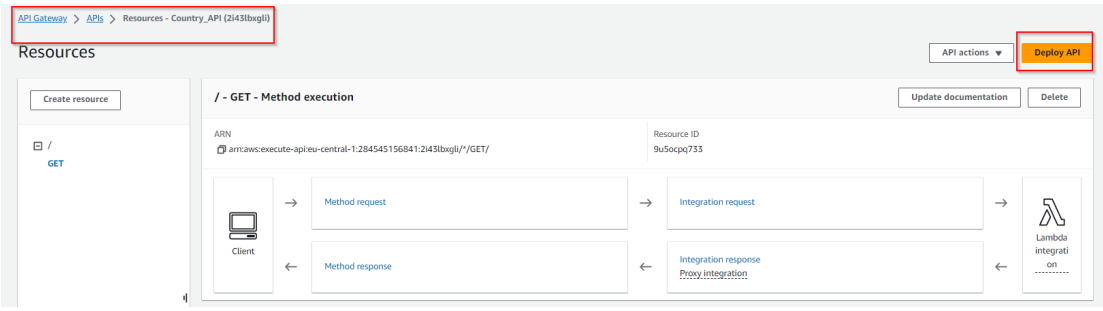


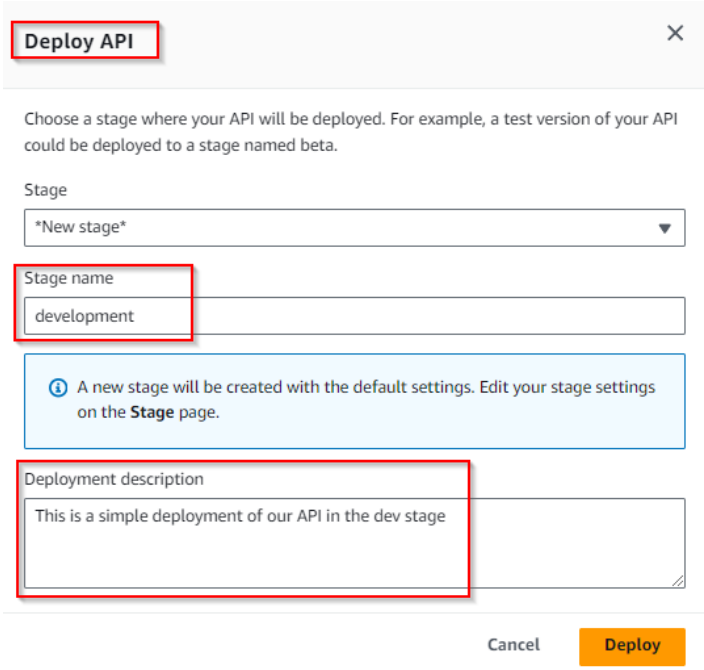
Step 10 - Provide a country which was not(!) provided in Step 4

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

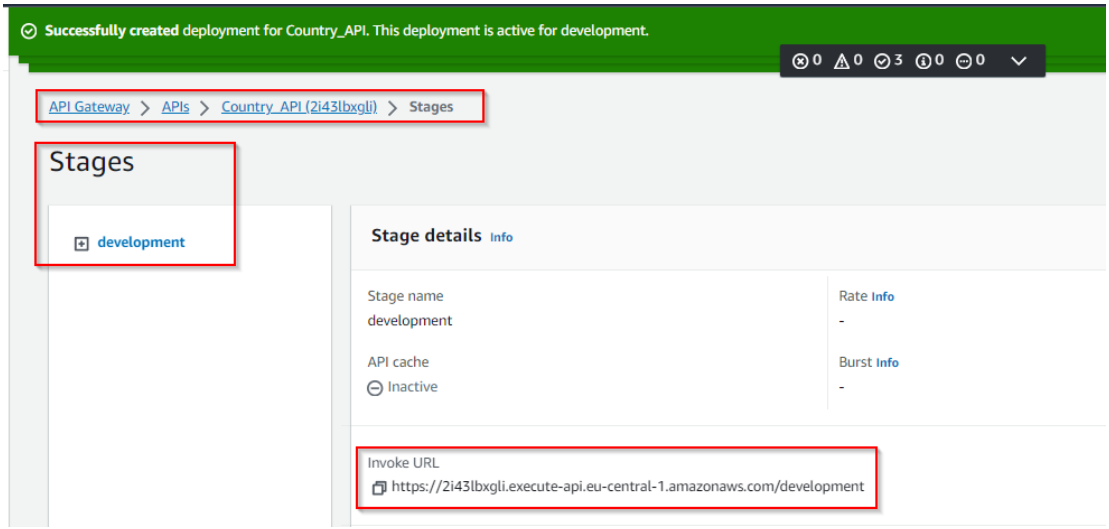
Step 11 - Deploy your API in the Development-Stage

- 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





Step 12 - Invoke the deployed API in your browser



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