

Assignment 1 - Introduction to REST APIs with AWS Lambda and API Gateway

In this assignment, you will learn how to:

- Create an AWS Lambda function
- Create an Amazon API Gateway Endpoint for the Lambda function
- Trigger a Lambda function by making a HTTP request to a URL
- Monitor AWS Lambda functions through the Amazon CloudWatch Log

In the previous lab we saw what Serverless Computing is, and how AWS Lambda is used to run cloud functions without the need for the user to provision servers. In this assignment, let's take a look at what a REST API is, and how AWS Lambda with an API Gateway trigger can be used as an API.

What is a REST API?

A REST API is a way of accessing web services in a simple and flexible way. An API, or application programming interface, is a set of rules that define how applications or devices can connect to and communicate with each other. A REST API is an API that conforms to the design principles of the REST, or representational state transfer architectural style. For this reason, REST APIs are sometimes referred to as RESTful APIs. REST guarantees that every API call from any source for the same resource will look exactly the same. Moreover, these API calls are stateless meaning the server does not store any information about the user session and has no memory of the previous API calls. Every API call is independent and should have enough information for the server to process it.

What is AWS API Gateway?

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads (which is what we will be using for the assignment, with Lambda), as well as web applications.

API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, throttling, monitoring, and API version management. For our assignment, we don't have to worry about most of the features it offers, except setting up an API and connecting it with AWS Lambda to get the appropriate response

Points to note:

1. Qwiklabs will create a temporary AWS account with all the required permissions and access to complete the lab. Do NOT use your personal AWS account. To prevent conflicts with any AWS account that you have already signed into on your browser, use Incognito/Private mode.
2. When using the Qwiklabs created AWS account, DO NOT change the default region/ VPC or any other settings that are automatically created by Qwiklabs.
3. The Qwiklabs lab session is timed. After the time limit is reached/ timer runs out, the AWS account will be removed and you'll have to restart the lab from scratch.
4. The assignments may need you to deviate from the Qwiklabs instructions and use your own code. Instructions will be given.
5. DO NOT try to access or avail any other resources and services that have not been described in the lab session or your account will be blocked.
6. Ensure that you have signed into Qwiklabs from your Google account.

Deliverables:

Create an API that accepts a GET request with a "key" query parameter in the URL from API Gateway and passes it on to a Lambda function. The Lambda function should return the "key" in the format "Your SRN : Key". If any other HTTP Method apart from GET is used, it should return "Only GET is supported"

What is a query parameter?**Sample Input / Output****Input**

GET https://some-api-url.com/default/YOUR_SRN?key=hello

Output

YOUR_SRN : hello

Input

POST https://some-api-url.com/default/YOUR_SRN?key=hello

Output

Only GET is supported

Don't worry, we've explained in detail how to build an API which can do this, in the steps below.

Evaluation of this assignment will be done automatically by our dashboard. You are expected to set up the API as per the instructions in this document, and submit the API URL on the dashboard.

Along with the API submission on the dashboard, the following screenshots are to be submitted:

- a. PESX2019XXXXX_1a.png: Showing the lambda created with your SRN
- b. 1b.png: Showing the API Gateway created, and your unique URL in the configuration tab of the Lambda Function
- c. 1c.png: A screenshot of the code that lets the Lambda function perform as per the requirements listed above.
- d. 1d.png: A screenshot of the dashboard after submission showing all test cases have passed.

Instructions:

Please read **ALL** the instructions carefully before proceeding. We will not be following the Qwiklabs tutorial, but instead will only be using the resources it provides. Actual instructions for the task are below.

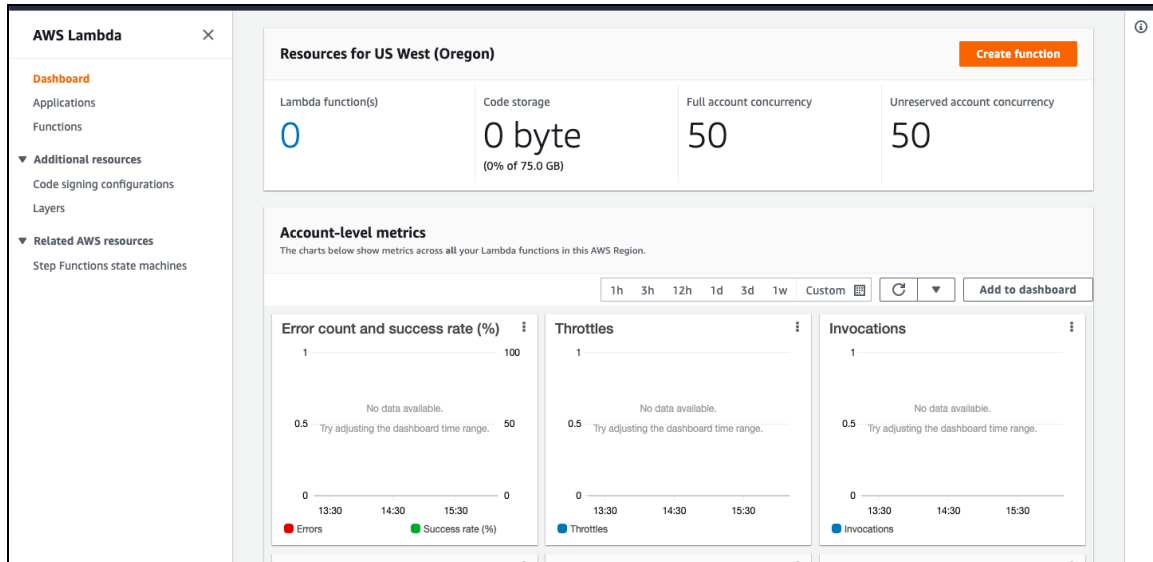
P.S - We have added some hints and suggestions after some steps, and at the end of the document. Please refer to the same. In case the Qwiklabs experiment times out before you can complete this lab, make sure you make a copy of your code / progress so that you can resume faster on the next attempt.

1. Head over to the [Introduction to Amazon API Gateway Qwiklabs](#) by clicking on the link. Sign in with your Google account if you're not already signed in, and click on Start Lab to get started. It might take a minute or two for Qwiklabs to allocate your resources. For this assignment, Qwiklabs gives you access to API Gateway and AWS Lambda only. Do not use resources apart from these two, any attempt could result in a ban of your account.

The screenshot shows the Qwiklabs lab interface for 'Introduction to Amazon API Gateway'. At the top left, there is a red 'End Lab' button and a timer showing 00:54:46. Below the timer is a caution box with the text: 'Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)' and an 'Open Console' button. The main title 'Introduction to Amazon API Gateway' is prominently displayed. Below the title, it says '55 minutes Free' with a 5-star rating and a 'Rate Lab' link. The AWS logo is followed by 'training and certification'. At the bottom, it says 'SPL-58 - Version 2.0.20' and includes a copyright notice: '© 2021 Amazon Web Services, Inc. and its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.' A link for 'AWS Training and Certification' is also present.

Once your resources have been allocated, click on Open Console to head over to your AWS Account. Ensure that you are logged into the student account that Qwiklabs has created for you, and no other account.

2. Once you're on AWS Console, search for Lambda. You should have a lambda page similar to the one below.



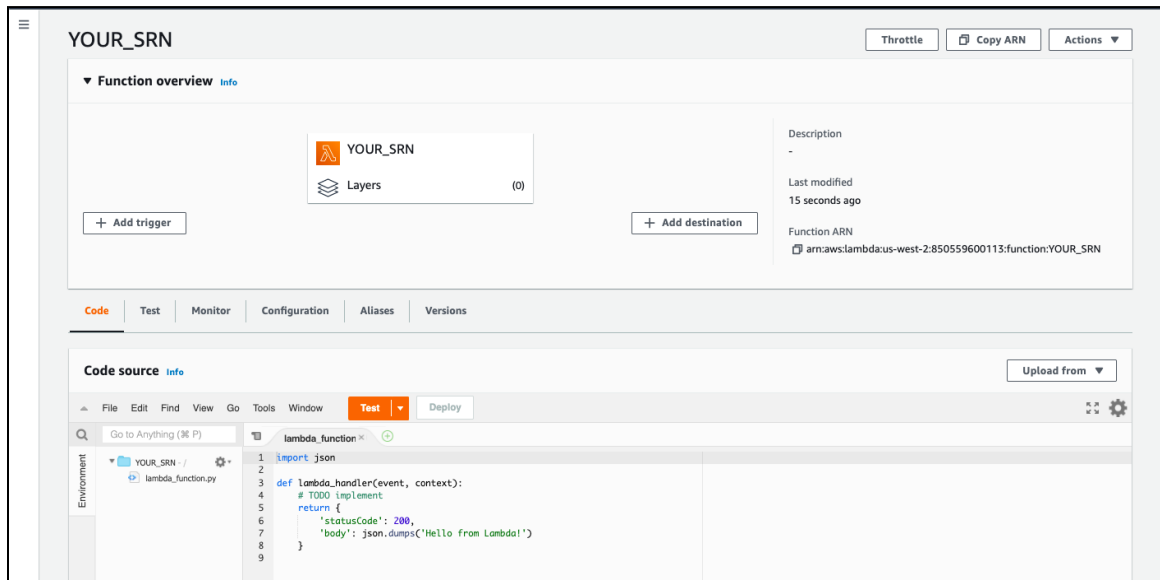
Click on Create Function to create your new function and choose “Author From Scratch”.

3. On the Create Function page, ensure the details are as follows -
 - a. Function Name - Your SRN (We will check whether your final submission is marked with your SRN so please ensure you don't miss this)
 - b. Runtime - Python 3.9
 - c. Architecture - x86_64
 - d. Permissions => Change default execution role => Use an existing role => lambda-basic-execution

You will not be able to proceed without this change in role

Ensure that your Function information looks like the above screenshot.


- Once you've created your function, you should land on the Lambda Function page, similar to the screenshot you see below.



As you can see, it already sets up a basic template for you. You can modify this file for the assignment later on. After any changes, make sure you click on the Deploy button above the code editor to save the changes.

- Let's now enable API Gateway for the Lambda function. On the Lambda Function page, under Function Overview, click on Add Trigger. On the Trigger page, choose API Gateway, and set it up as shown below.

Trigger configuration


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**

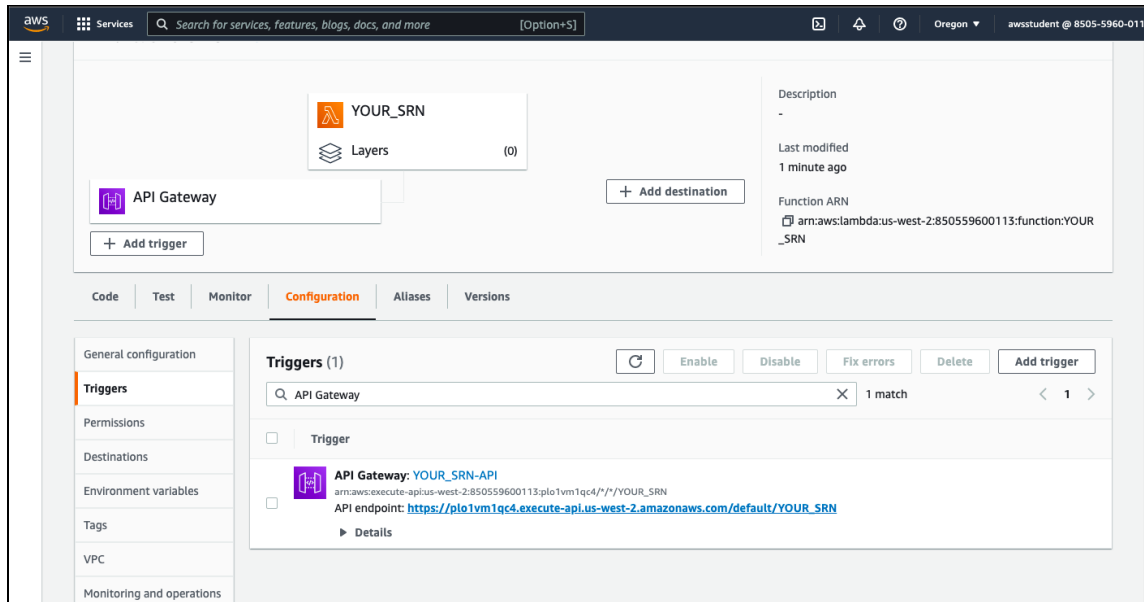
Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel

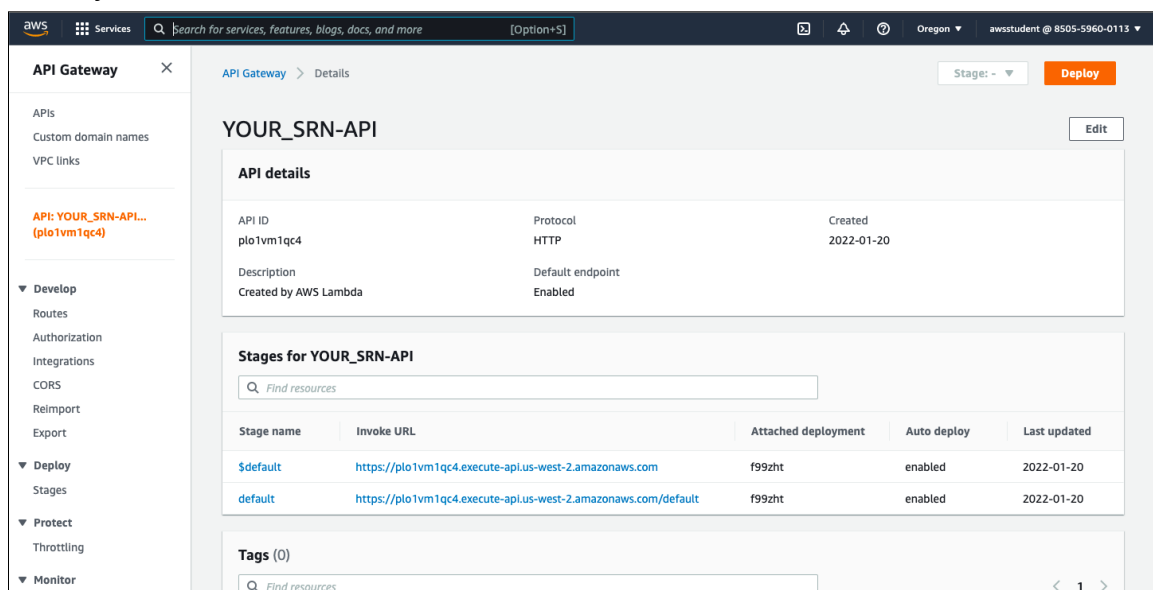
Add

Under Create a new API or Attach an Existing one, select 'Create an API'.
Choose 'HTTP API', and set the Security to 'Open'.

- Once the Trigger is added, you can click on it to view more details. Clicking on the API Gateway trigger will lead to the configuration tab for the Lambda function, where you can see the API Name and API Endpoint. The API Endpoint is the URL which we will use to test out our Lambda Function.

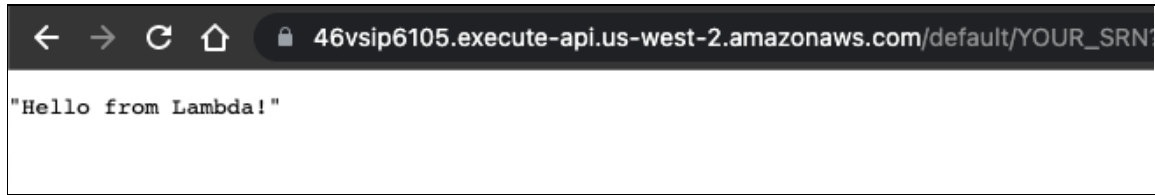


- Click on the API Gateway Name (Your_SRN-API) to view more information about the Gateway.



Here we can configure routes, set authentication and much more. We won't be needing any of that for this experiment, but feel free to look around and figure out what the options do.

- Go back to the Lambda function page. In the configuration tab of the Lambda function, you can see it has already provisioned a URL for us. Click on the API endpoint URL to visit the page, you should see something like this-



As we can see here, the "Hello From Lambda!" message was what was in the lambda_function.py file, which is visible if you click on the 'Code' tab of your Lambda function page. You can change the message and deploy, and you should be able to see the new message when you visit the link.

P.S - Your API Gateway link will be different, use that, not the URL in the screenshot.

9. Now onto the main task!

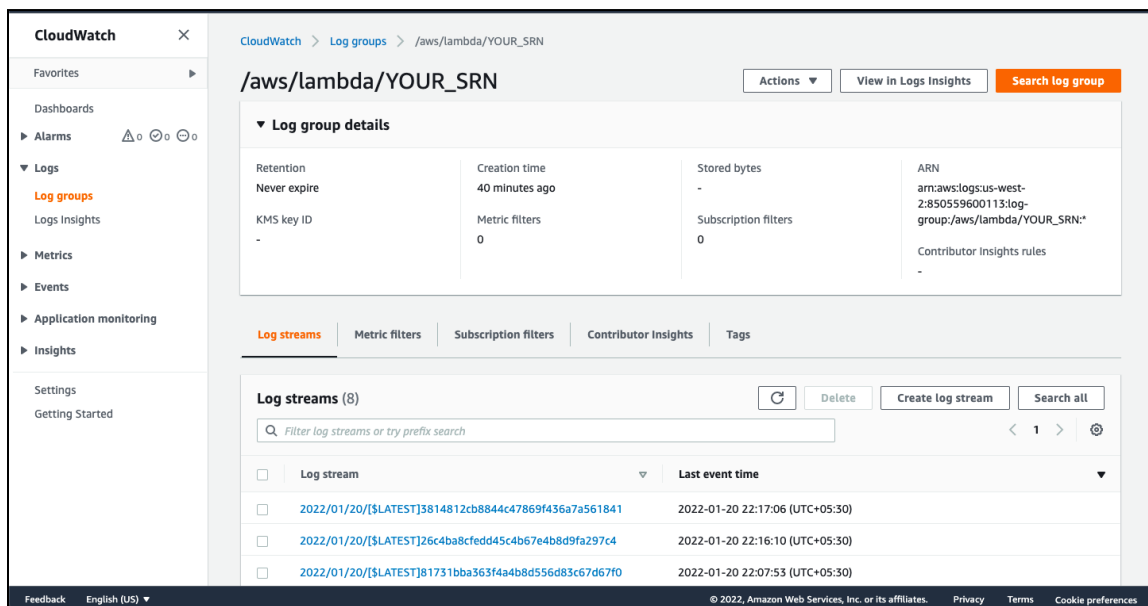
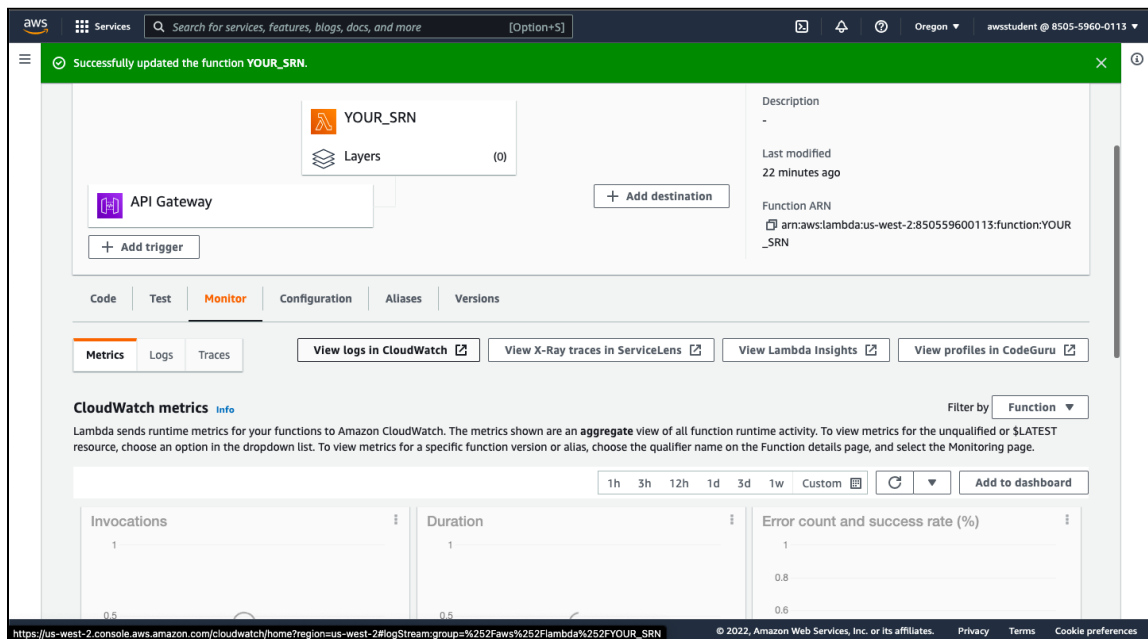
We have a lambda function that returns some text, and an API Gateway to trigger it. Your task is to write the lambda function in such a way that it -

- a. Reads the URL, gets the query parameters, looks for a query parameter called "key" and gets the value of that.
- b. Checks the type of request to the API. If it's a GET request, it should respond with the appropriate response, or else show a message that "Only GET is supported".

HINT : The Lambda function takes a parameter called "event", which contains information about the event that triggered the Lambda function. In our case, that event is the API Gateway, and therefore even contains information about the URL, query parameters etc that triggered the function. Try returning the event as the response from the lambda function, and then go to the API URL to see what the event contains, and which parameters inside the event you require.

Make sure you add a query parameter to the URL before printing the event so you see where exactly the query parameter is inside the event.

10. To check what the event parameter contains, you can print the event parameter in your program, and view the output under the logs. You can create a new log by using the default python print() statement. Then, under the Monitor tab of the Lambda function, click on **View Logs In Cloudwatch** to view the recent logs. On the Cloudwatch page, you can see the logs separated by time they were created, and you can view recent logs to find your print statements.



11. Once you've figured out what fields to use from the event parameter, modify the lambda function to -
 - a. Check if the request made was a GET request, if yes, check for an event parameter called key and get the value. From the Lambda function, return "Your_SRN : Key" (Not this text, but your actual SRN and the key from the URL)
 - b. If it's not a GET request, return "Only GET is supported".

For sample input and output, check page 2 of the document.

12. The final output should look something like this:

```

1 // 20220120223518
2 // https://46vsip6105.execute-api.us-west-2.amazonaws.com/default/YOUR_SRN?key=Some_Text
3
4 "SRN : Some_Text"

```

As you can see in the URL, we've passed a query parameter called key, the value of which was "Some_Text" and the Lambda function returned SRN : Some_Text. Your function should also do the same, and replace SRN with your actual SRN.

13. After testing it out yourself, submit the API to the dashboard for evaluation.

The link to the dashboard is <https://cclabdashboard.netlify.app/>

Assignment 1 Dashboard

[How to submit?](#)

SRN

Make sure you enter your correct SRN before submitting

API Endpoint

Submit your API Endpoint for evaluation

[Submit](#)

Enter your SRN and API Endpoint. The API Endpoint is the one you see in the Configuration Tab of the Lambda function under API Gateway. **Please provide the API Endpoint as it is on the configuration tab of the lambda function. Do not add any query parameters or additional info while submitting.** The dashboard will evaluate your API automatically and assign marks. Make sure you submit only your endpoint, and SRN. We will be checking for plagiarism.

Before submitting on the dashboard, verify locally whether the API is performing as expected. You can use an API Tester like Postman or any online tool like <https://reqbin.com/> or <https://hoppscotch.io/> to test your API.

To test POST response, choose POST as the HTTP method, and enter the API URL as it is, and test.

To test the GET response, choose GET as the HTTP method, and along with API URL, add `?key=testing` to the end of the URL to add the key query parameter and check response.

FAQ and Common Issues

1. The API is returning an “Internal Server Error”

When the Lambda function runs into some issues the API might return an error. There are a few ways to look into this.

- Since the code is in Python, make sure your indentation is correct and there are no syntactic errors
- Since you will be accessing key-value pairs from the event, ensure the key you're accessing is present in the event before getting the value. Defensive coding is always a good practice.
- If your code takes longer than the lambda run time (default of 3 seconds), it will throw this error. You can go to Cloudwatch Logs and see how long your last request took to process.
- Add a try/except block in your code which can catch errors and return the exact error gracefully
- You can go to cloudwatch logs and view if there were any error messages logged. While sometimes a python error might be logged automatically in case of a crash, it's better to write your own print statements so you know exactly where you're going wrong.

2. The GET response is not in expected format

If the dashboard tells you that the GET is not in expected format, this might be because the response is not at all what we were expecting (like the function returning an error) or because of a slight mistake in your return format. Make sure the return statement is `SRN : KEY` i.e **SRN**<space>:<space>**KEY**

Also ensure the SRN your code is returning is yours, and the key is from the query parameter. Key cannot be hard coded.

3. GET Response is correct but POST is failing

If GET works but POST fails, check your code if the message you're returning is as expected. Secondly, check with an API testing platform like Postman whether the response is correct. One issue might be that you're getting the Query Parameter before checking for GET/POST and therefore since POST request does not have a query parameter the function will fail. Always ensure defensive coding.

