Assignment 1 - Introduction to AWS Lambda with 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

What is Serverless Computing?

In simple terms, Serverless computing allows you to build and run applications and services without thinking about servers.

More formally, serverless computing is a method of providing backend services on an as-used basis. It allows users to write and deploy code without the hassle of worrying about the underlying infrastructure. A company is charged based on their computation and does not have to reserve and pay for a fixed amount of bandwidth or number of servers, as the service is auto-scaling. Despite the name serverless, physical servers are still used but developers do not need to be aware of them.

What is AWS Lambda?

AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you.
These events may include changes in state or an update, such as a user placing an item in a shopping cart on an e-commerce website. AWS Lambda automatically runs code in response to multiple events, including but not limited to HTTP requests via Amazon API Gateway, modifications to objects in Amazon Simple Storage Service (Amazon S3) buckets, table updates in Amazon DynamoDB, and state transitions in AWS Step Functions.

Lambda runs your code on high availability compute infrastructure and performs all the administration of your compute resources. This includes server and operating system maintenance, capacity provisioning and automatic scaling, code and security patch deployment, and code monitoring and logging. All you need to do is supply the code.

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

Introduction to Qwiklabs:

Qwiklabs is an online platform which provides end to end training in Cloud Services. This is a platform where you can learn in a live environment anywhere, anytime and on any device. Qwiklabs offers training through various Labs which are specially designed to get you trained in Google Cloud Platform (GCP) as well as Amazon Web Services (AWS). This Lab in specific will be done using Qwiklabs.

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".

It must also accept a POST request with the form-data as a csv file (provided to you by your professor), which details a set of 50,000 values for 5 columns, and return the average of each column as a json.

If any other HTTP Method is used, it must return "Only GET and POST is supported". Test your API out using Postman.

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
(File Upload, for eg: times.csv)

A	B	C	D	E
1	1	5	4	3
2	3	1	4	1

Output
{A:1.5, B:2, C:3, D:4, E:2}

Input
PUT https://some-api-url.com/default/YOUR_SRN?key=hello
Output
"Only GET and POST are supported"

Neither of these functions are in **any way** complicated; The POST request deliverable can be done in Excel in a couple of clicks! However, they serve to demonstrate the **basic usage** of serverless computing. Much more complex computation and request serving can be done with the same basic

set of steps.

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

The following screenshots are to be submitted in a PDF file:

a. 1a: Showing the lambda created with your SRN
b. 1b: Showing the API Gateway created, and your unique URL in the configuration tab of the Lambda Function under Trigger
c. 1c: A screenshot of the code that lets the Lambda function perform as per the requirements listed above.
d. 1d: A screenshot of a successful GET request with query parameter "key" having a value of your choosing (for example, "Hello") with the response body being "YOUR_SRN: Hello" (for example) on Postman
e. 1e: A screenshot of a successful POST request with the form-data being the csv file provided to you by your professor, with the returned value being a dict of the column names with their average values.
f. 1f: A screenshot of a failed HEAD/PUT request with the response body being "Only GET and POST is supported" on Postman

Name the file **<SECTION>_<SRN>_<NAME>_A1** (eg: *E_PES1UG19CS999_Rumali_A1.pdf*").

Instructions:

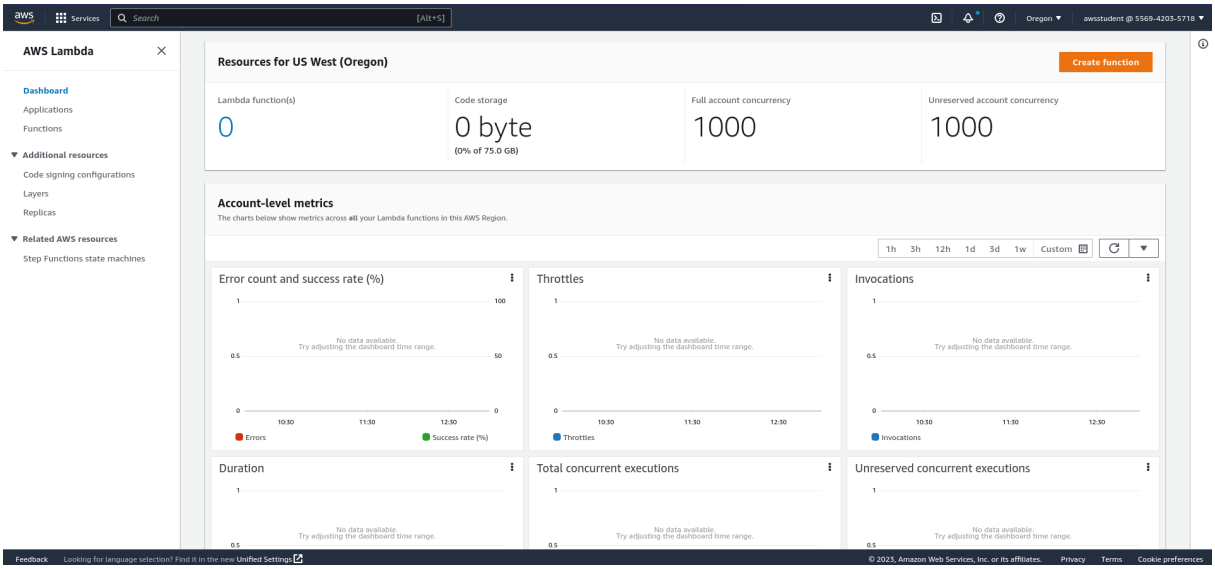
Please read **ALL the instructions** carefully before proceeding. **We will not be following the Qwiklabs tutorial**, but instead will **only be using the resources it provides**. Actual instructions for the task are below.

P.S - We've added some hints and suggestions after some steps, and at the end of the document. This assignment is **likely** to take more than the **1 hour** allotted by Qwiklabs. In the 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.**

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; if you don't, click on the hamburger menu on the left and navigate to the 'dashboard' section.

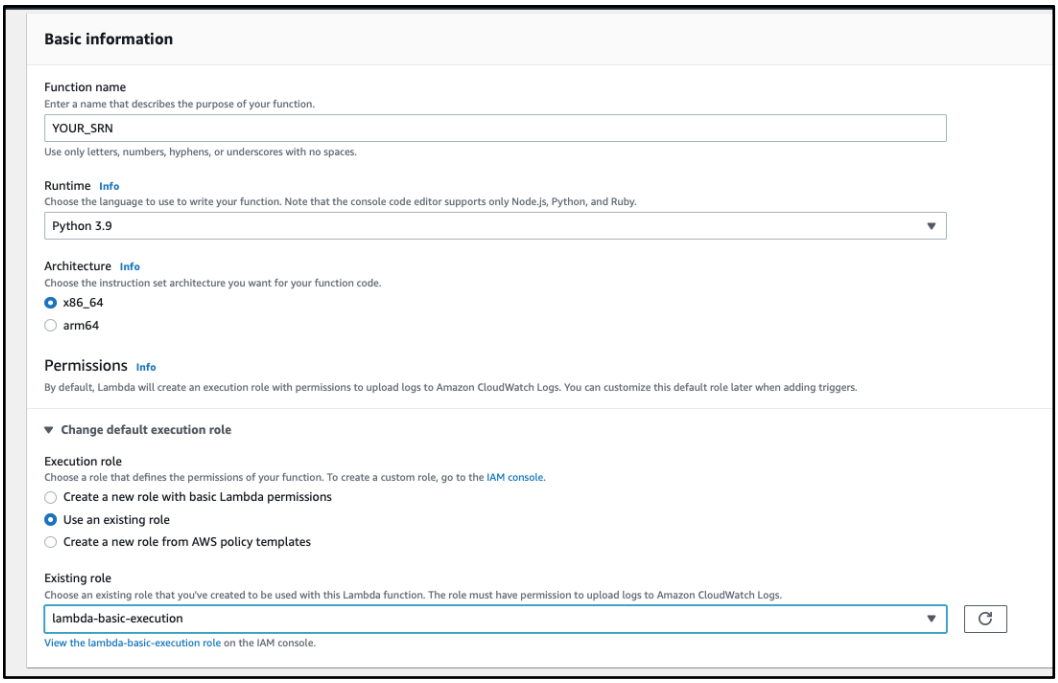


Click on Create Function to create your new function and choose **"Author From Scratch".**

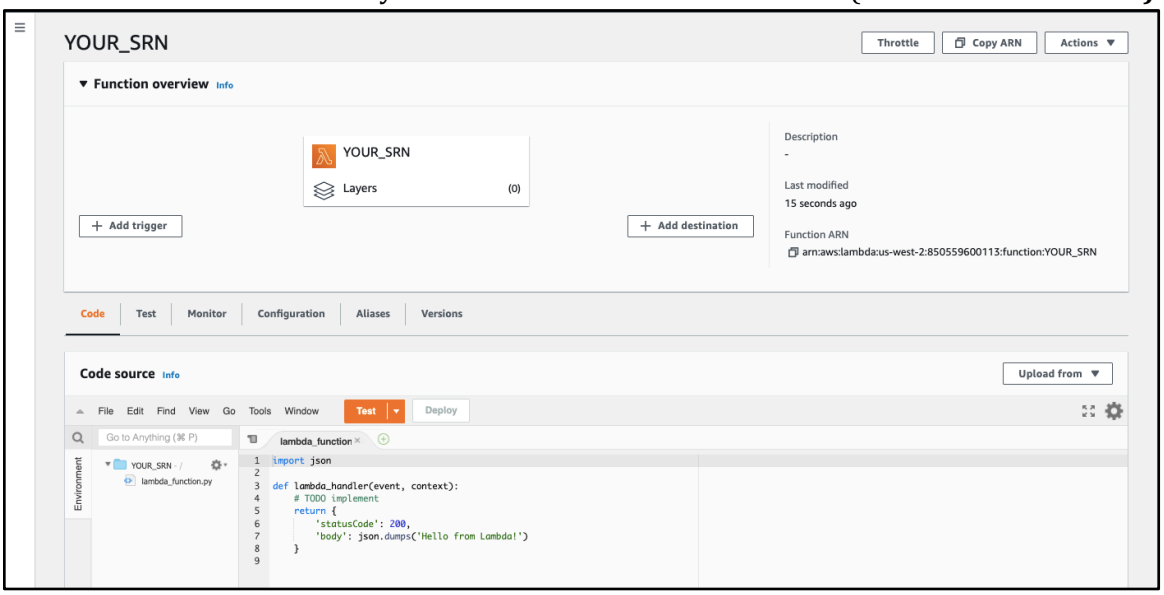
It's possible to use serverless with a **container image**! However, we shall stick to a simple function in this assignment, because unfortunately, the concept of containers and images have not been covered yet. Feel free to experiment with this in your own time, though!

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 below screenshot. You will learn more about roles in the **IAM Experiment (Experiment 5).**

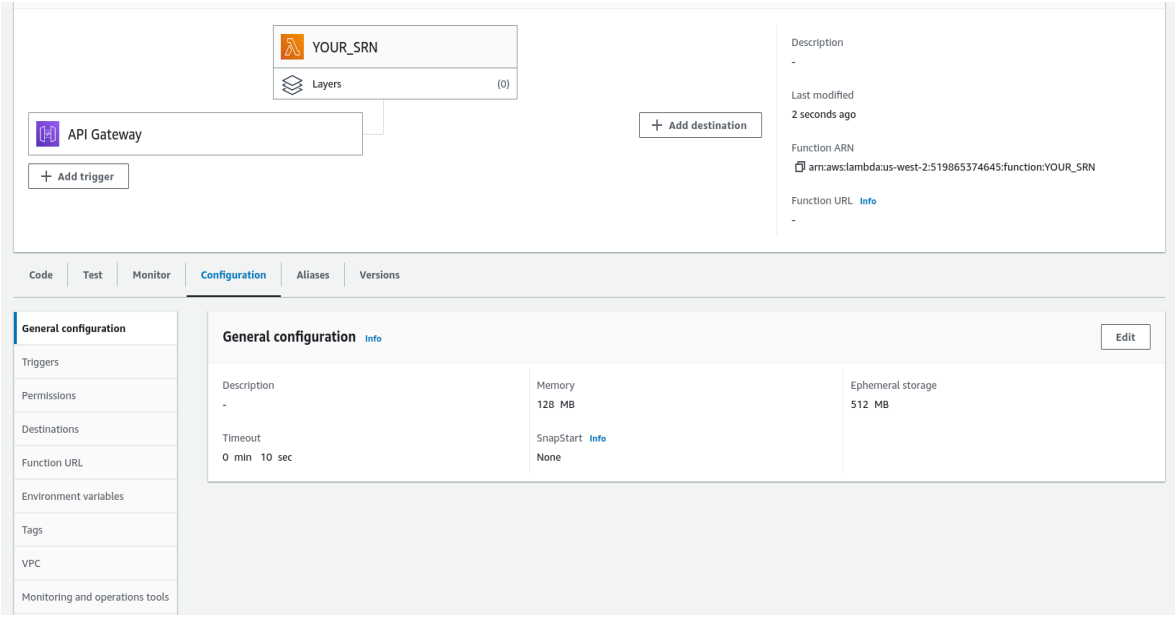


4. Once you've created your function, you should land on the Lambda Function page, similar to the screenshot you see below. (**Screenshot 1a)**

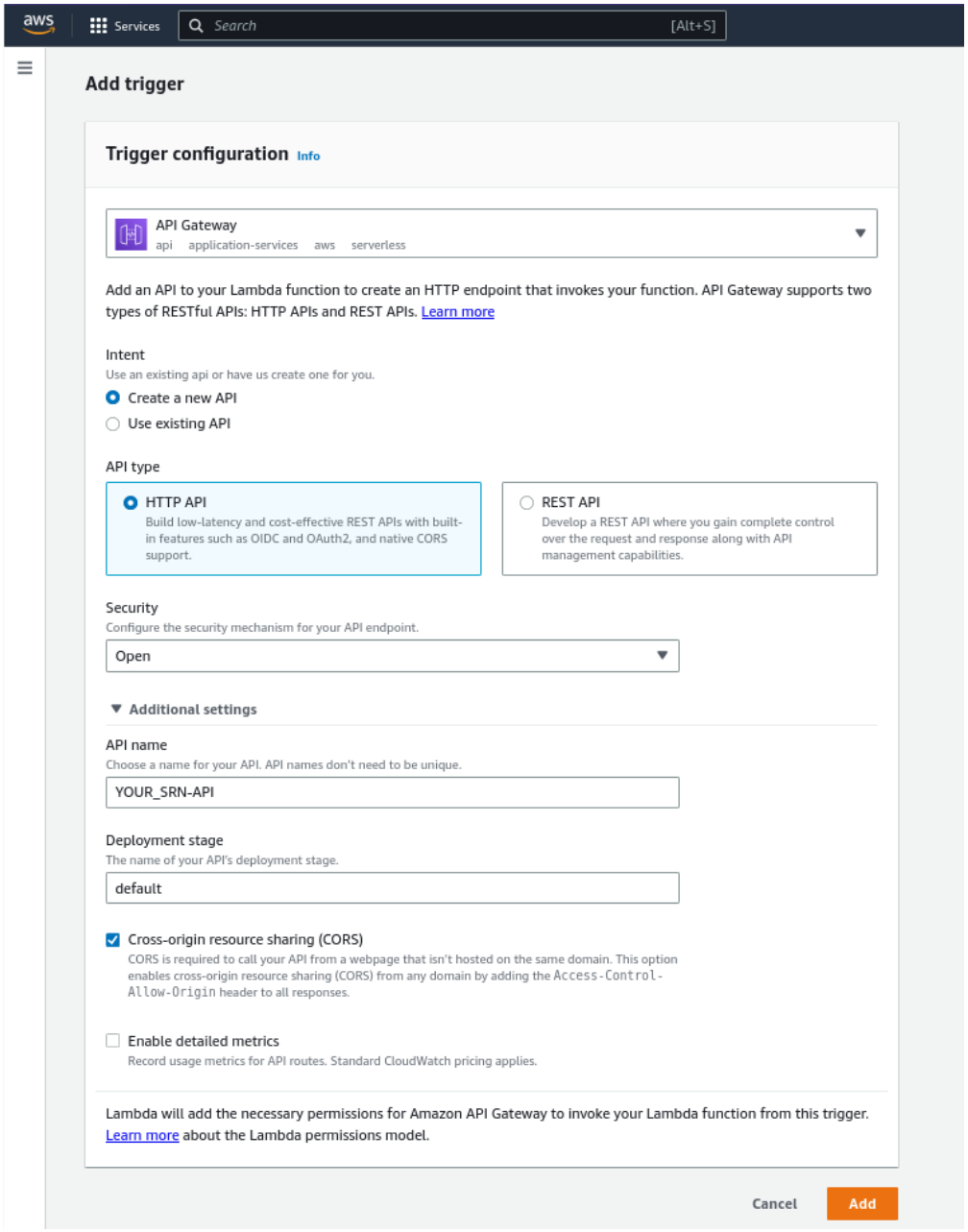


As you can see, it already sets up a basic template for you. You'll be **modifying this** file for the assignment later on. After any changes, make sure you click on the **Deploy** button above the code editor to save the changes. For now however, let's keep the function as is.

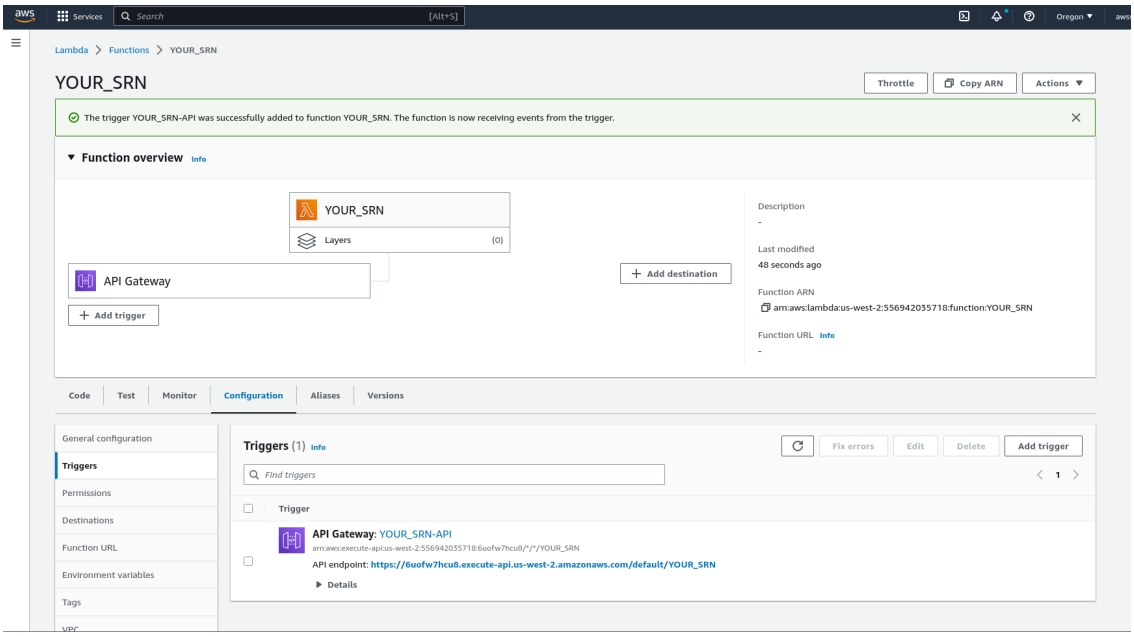
5. Go to the “Configuration” tab, select “General Configuration”, and set the timeout to at least **10 seconds**; this is because reading/processing the csv for the POST request takes longer than the default timeout of 3 seconds. Vary this to find the optimal number!



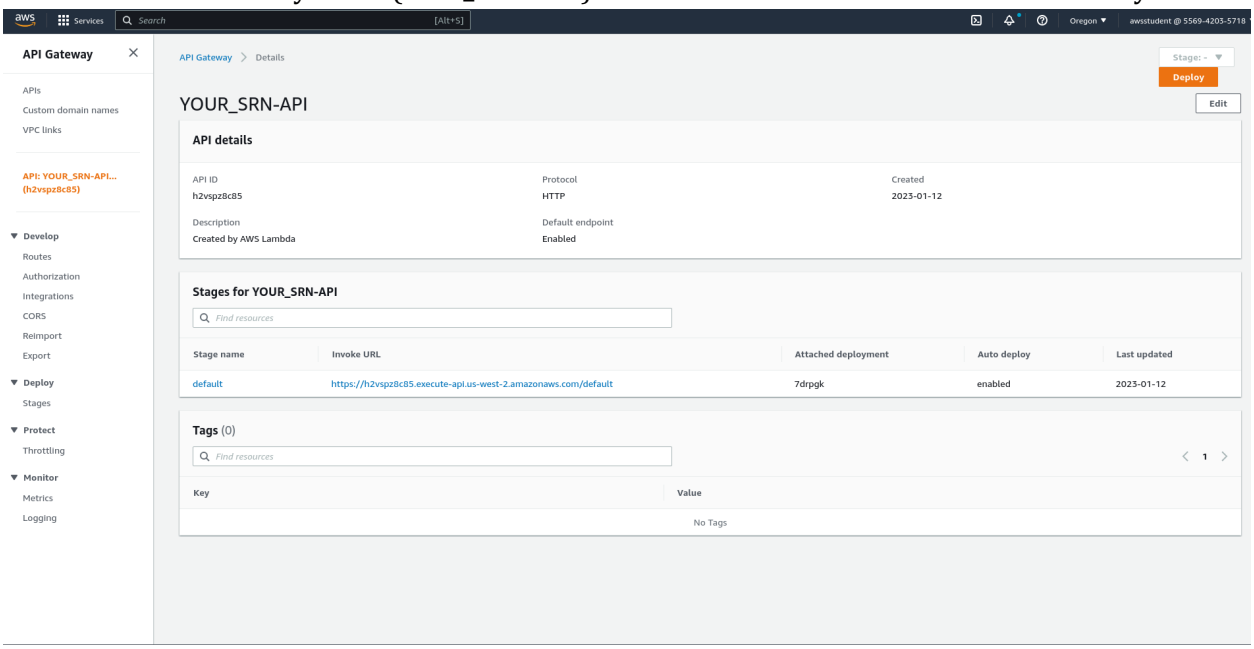
6. Let's now enable API Gateway for the Lambda function. On the Lambda Function page, under **Function Overview,** click on **Add Trigger**. On the Trigger page, choose **API Gateway,** and set it up as shown below. Under Create a new API or Attach an Existing one, select 'Create an API'. Choose 'HTTP API', and set the Security to 'Open'. Under additional settings, **enable CORS.**



7. 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'll use to test out our Lambda Function. (**Screenshot 1b)**

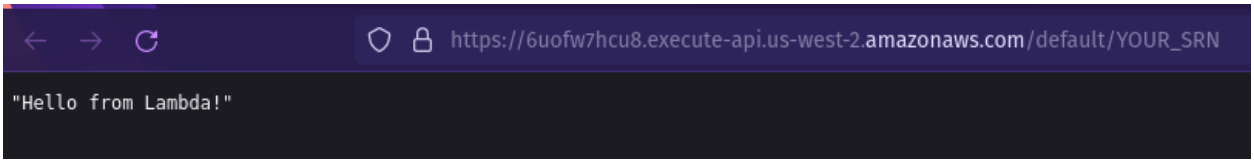


8. Click on the API Gateway Name (Your_SRN-API) to view more information about the Gateway.



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.

9. 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.

10. 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. **Checks the type of request to the API;**
    b. **If it's a GET request, it must read the URL, get the query parameters, look for a query parameter called "key" and get the value of that. It must then use it to return a response of "YOUR_SRN: value".**
    c. **If it's a POST request and contains a csv file, it must calculate the average values of each column and return it as a json, with the key-value pairs being (column_header: avg).**
    d. **Rejects all other request types.**

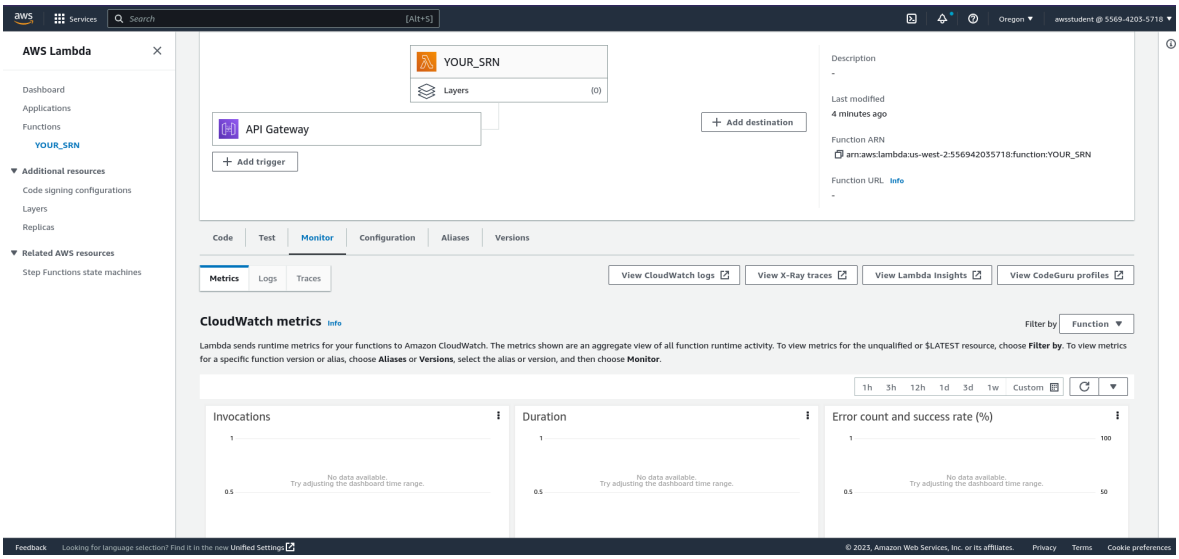
    *For GET requests, make sure you **add a query parameter** to the URL before printing the event so you see where exactly the query parameter is inside the event. For example:*

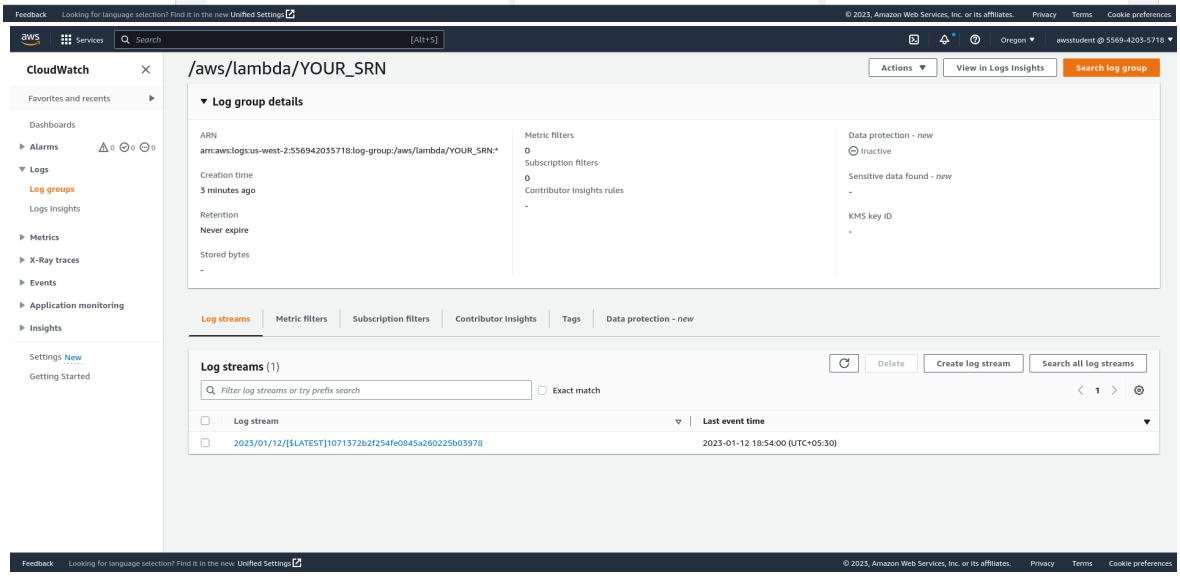
    https://some-api-url.com/default/YOUR_SRN**?key=hello**

    The part highlighted in red is the query parameter for the sample (fake) url.

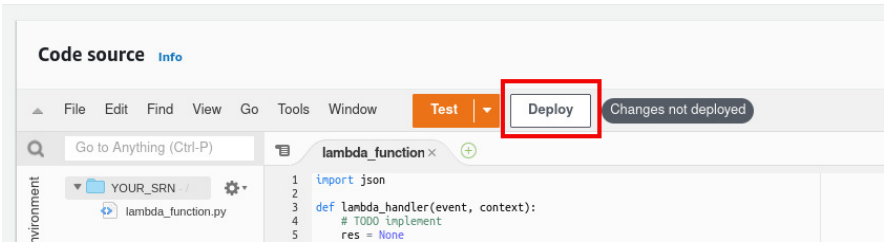
    *For POST requests, use **Postman** to test out your API. Steps for using Postman for uploading a csv as part of your POST request can be found in **Step 14**.*

11. 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. To access this, click on the 'Monitor' section on the Function page and click on 'View CloudWatch logs'. This will help you debug your lambda function.

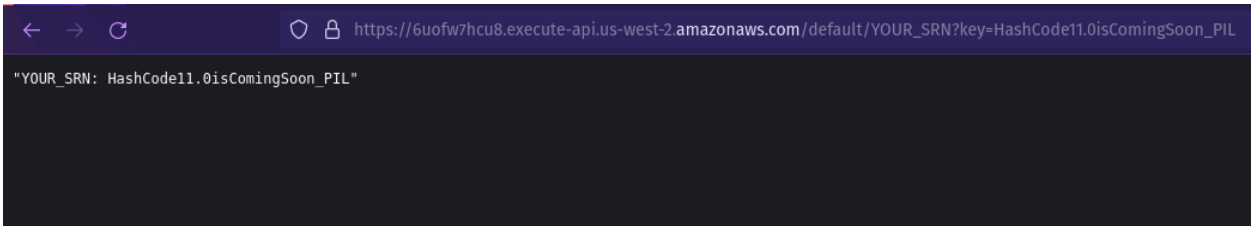




12. Once you've figured out what fields to use from the event parameter, modify the lambda function to satisfy the above requirements. Make sure to **deploy** the changes you make for it to reflect on your API. **(Screenshot 1c)**



13. The final output for a GET request could look something like this:



As you can see in the URL, we've passed a query parameter called key, the value of which was "HashCode11.0isComingSoon_PIL" and the Lambda function returned *"SRN: HashCode11.0isComingSoon_PIL"*
Your function should also do the same, and replace YOUR_SRN with your actual SRN; use any parameter value you'd like.

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** itself 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.*

*For your **POST** request, the csv file will likely be sent in **base64** as part of the **request body**, which should be evident once you return the event itself; Look up ways to **decode** the base64 string and **read** it as a csv!*

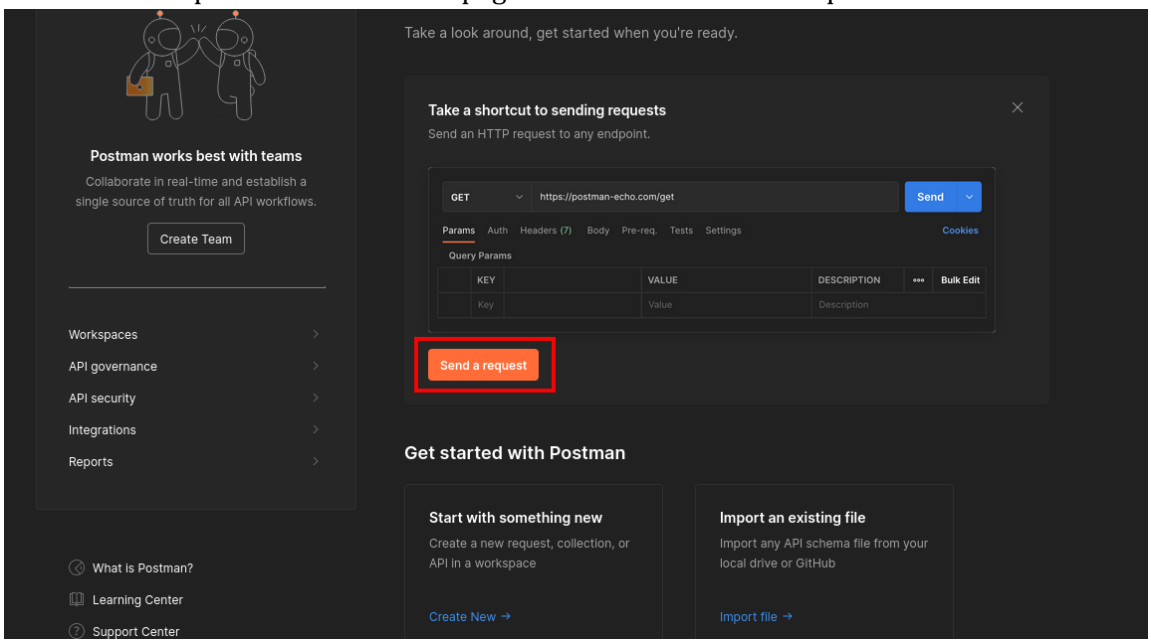
*While it is possible to test out your **GET** request handling without Postman by just entering the url in the address bar, you will **need to use Postman (or cURL)** to test out your POST request handling, due to needing to send the csv file along with it.*

*For your **GET** request, 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. For example:*

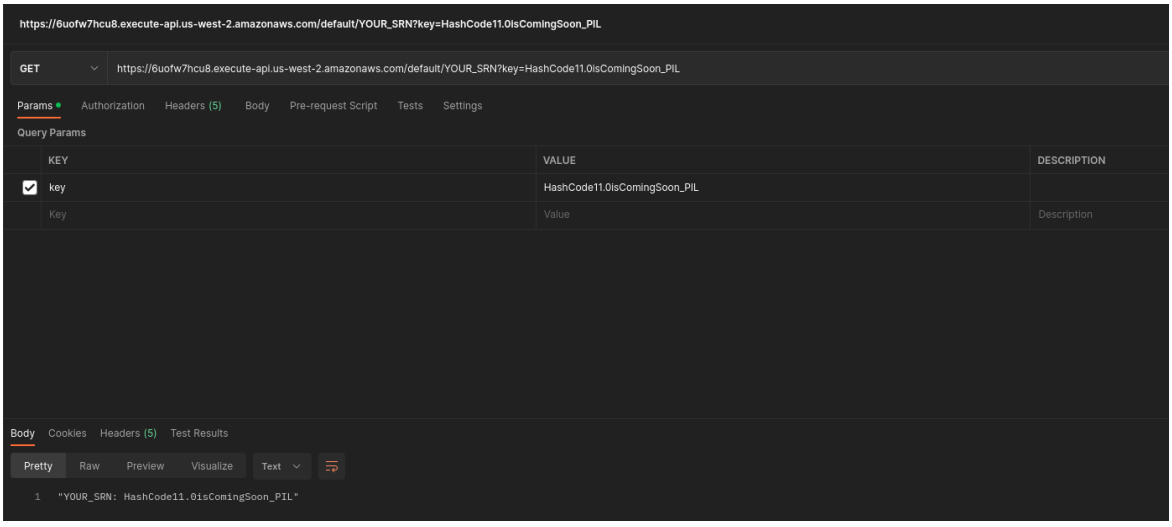
https://some-api-url.com/default/YOUR_SRN**?key=hello**

The part highlighted in red is the query parameter for the sample (fake) url.

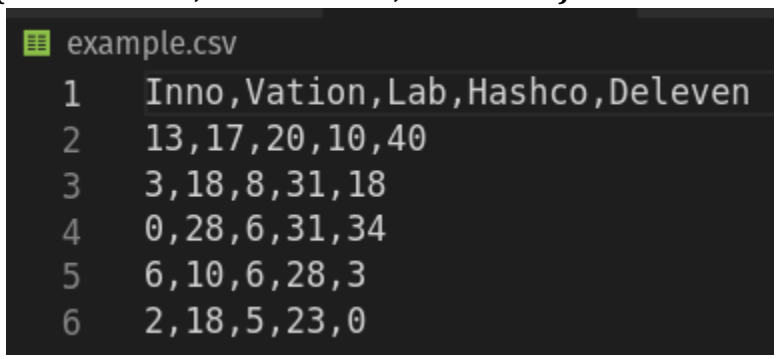
14. Go to https://web.postman.co and sign in with your Google account. Complete the initial steps to reach the home page. Click on the “Send a Request” button.

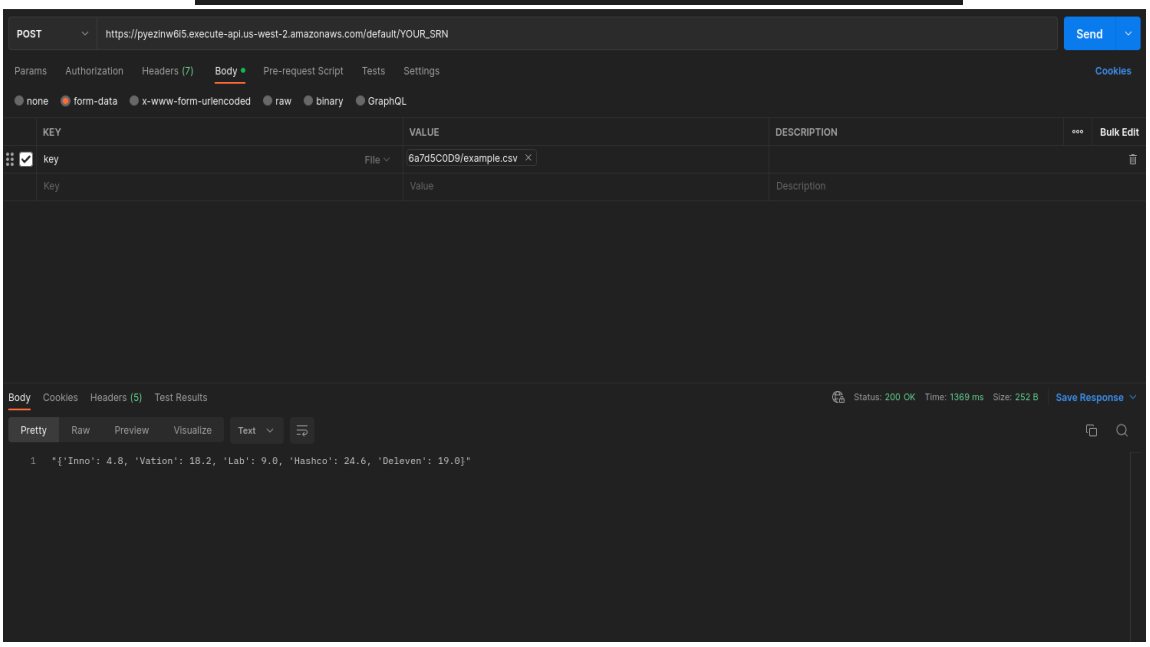


Paste your API in the “Enter request URL” section; Send a GET request similar to the one shown below. **(Screenshot 1d; Make sure the response body is visible, and is of the form YOUR_SRN: Value”)**

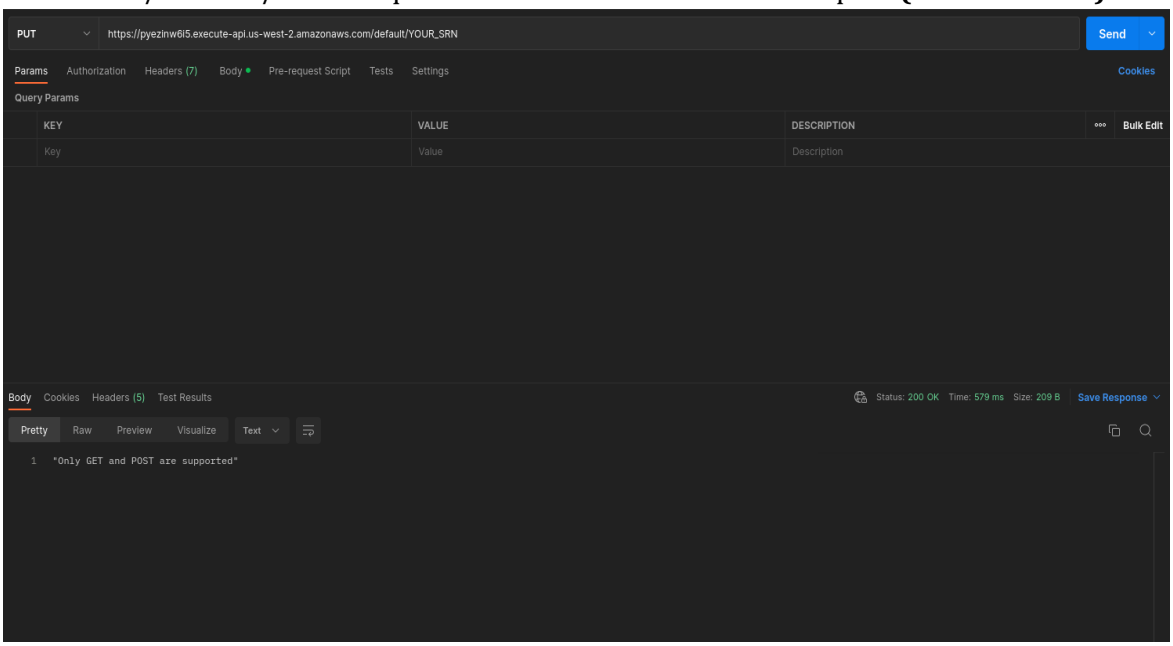


To send a POST request with the csv, click on the Body tab and select form-data. Add the key name as 'key', change its type from 'text' to 'file' and upload your csv file, similar to the one shown below. **(Screenshot 1e; not of the CSV, of Postman)**





Send a PUT/DELETE/HEAD request and show that it does not accept it. **(Screenshot 1f)**



FAQ and Common Issues

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.

a. Since the code is in Python, make sure your indentation is correct and there are no syntactic errors
b. 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.
c. 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.
d. Add a try/except block in your code which can catch errors and return the exact error gracefully
e. 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