AWS Lambda - AWS API Gateway Integration Project

First we will create a Function on AWS Lambda and we will test that function on AWS Lambda itself , then we will do api integration then we will test it on postman.

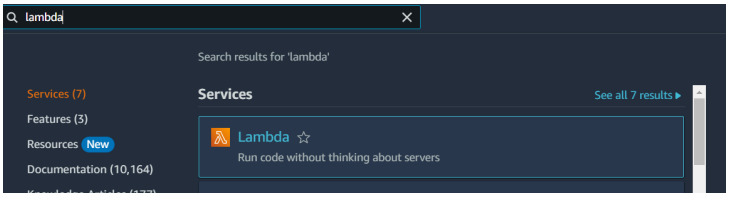
What is AWS Lambda ?

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers.

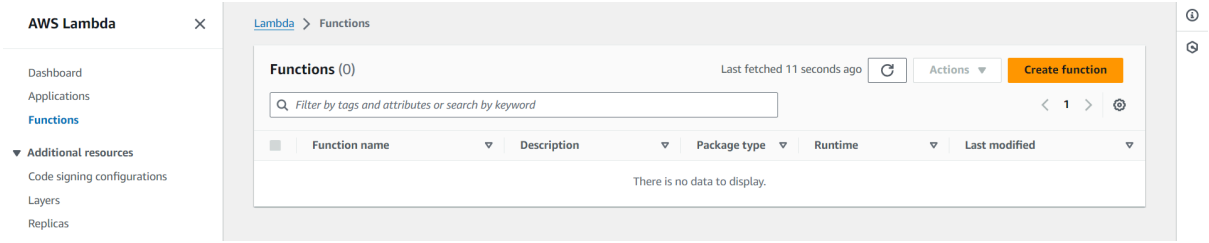
PART 1

STEPS

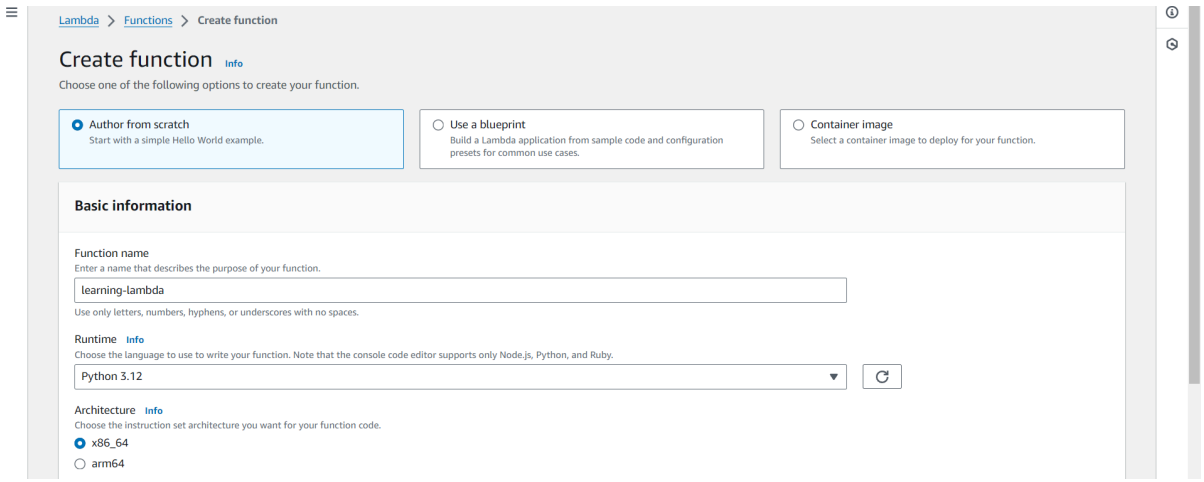
Step 1 :- Search Lambda -> Click on Lambda



Step 2 :- Click create function

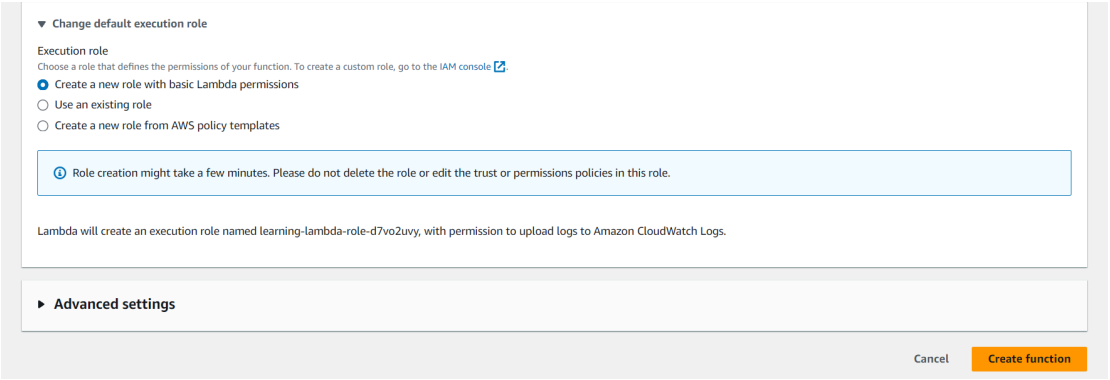


Step 3 :-
Give function name -> I'm giving learning-lambda
Select Runtime -> I'm selecting Python
Select Architecture -> I'm selecting x86_64

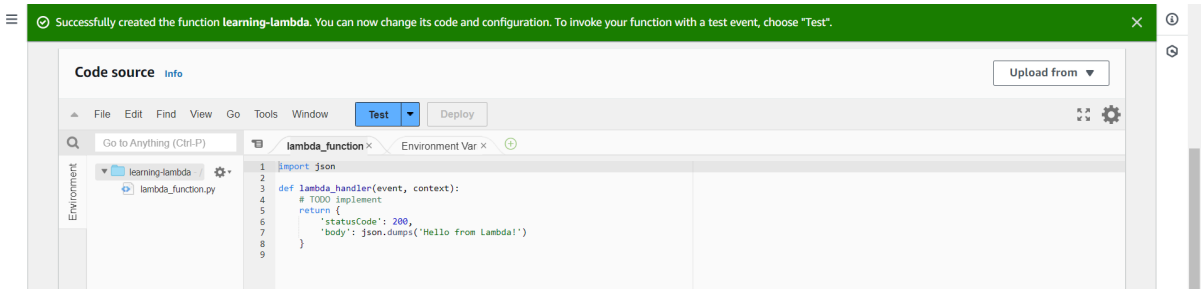


Permissions -> Select Create a new role with basic Lambda permissions

Press Create Function



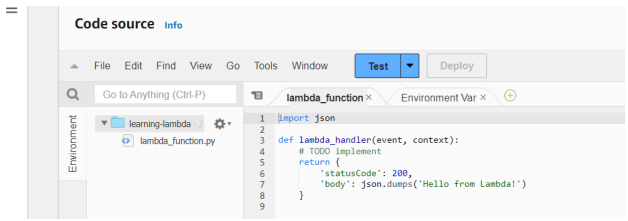
The function created successfully



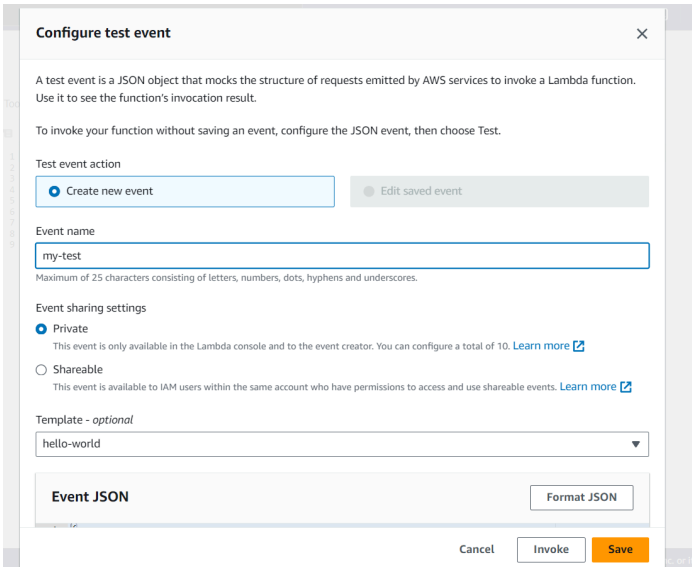
As you can see the function name is ‘lambda_handler’ which takes 2 parameters. One is the event and second is context.

event :- The data from the event that triggered the function.
context : The data about the execution environment of the function.

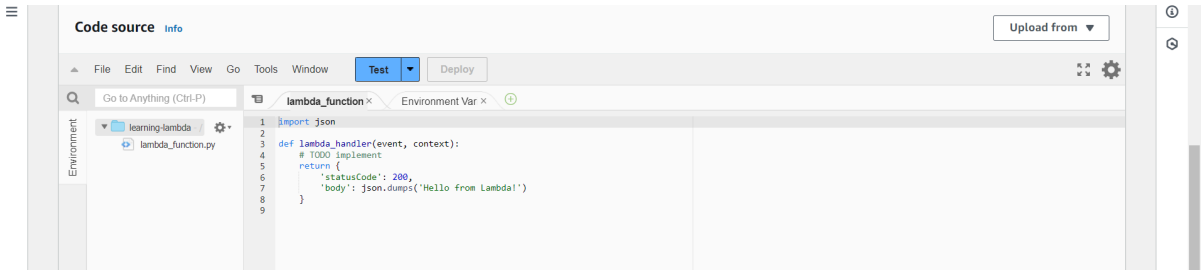
Step 4 :- Testing the function
Click on the test button



Give name to your test -> I'm giving my-test
Click save



The status code is 200

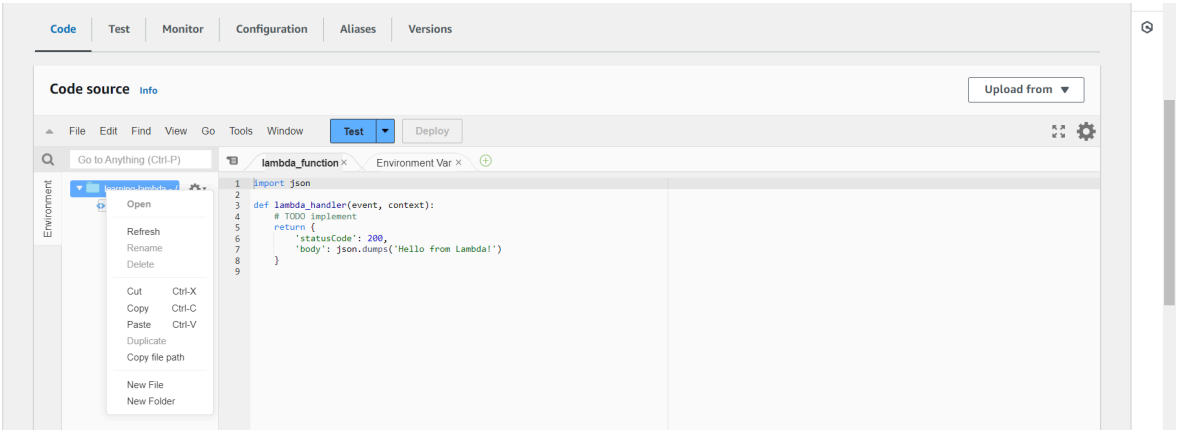


PART 2

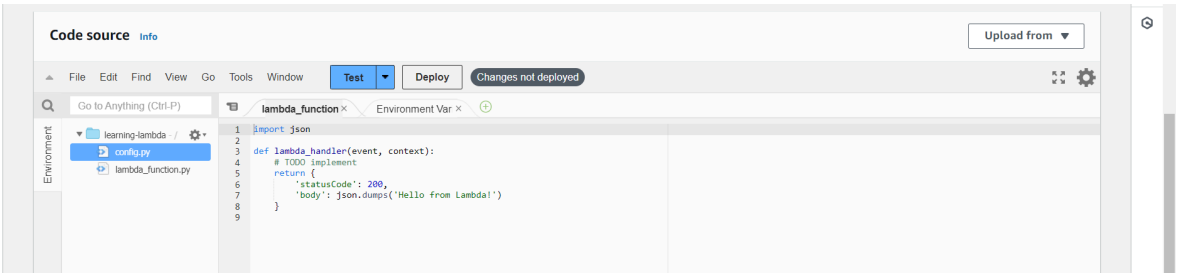
Go to
https://jsonplaceholder.typicode.com/
Free fake API for testing and prototyping.

STEPS

Step 1 :- Create a new file -> Right click on folder -> Click new file

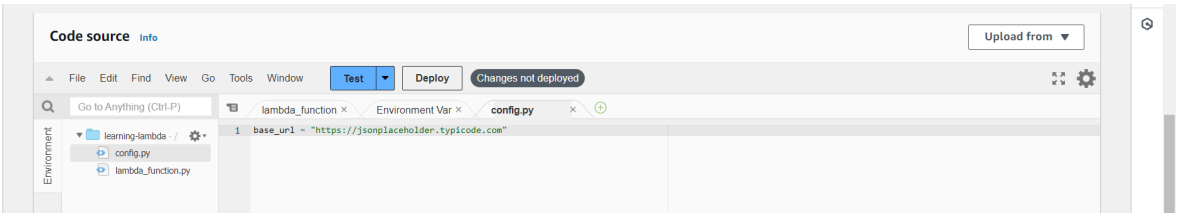


Give name to your file -> I'm giving config.py



Step 2 :- Copy the base url from jsonplaceholder website.

Open config.py and paste the base url.

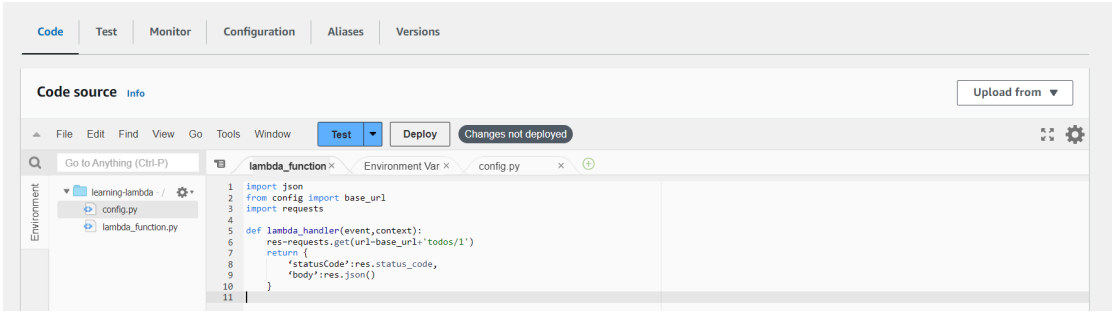


Open lambda_function.py -> Copy the below code

```
import json
from config import base_url
import requests

def lambda_handler(event,context):
        res=requests.get(url=base_url+'todos/1')
```

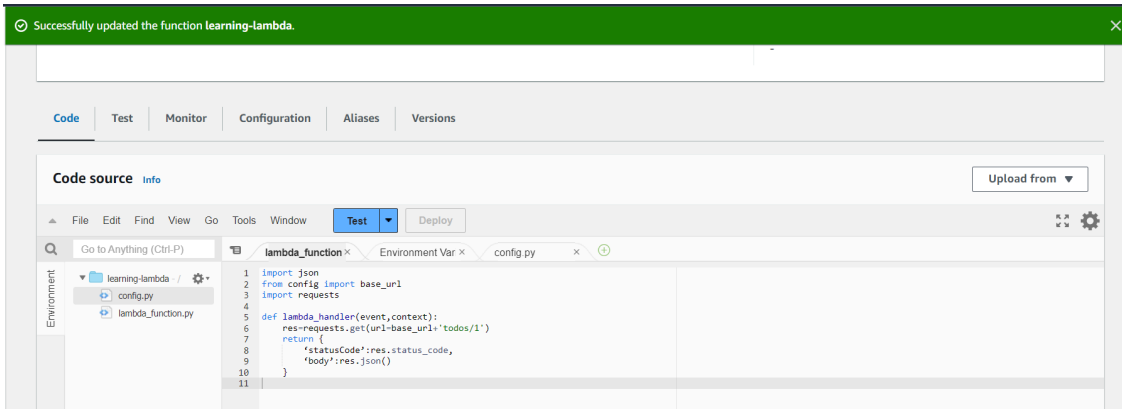
```
return {
        'statusCode':res.status_code,
        'body':res.json()
}
```



Step 3 :- Deploying the function

Now press Deploy button

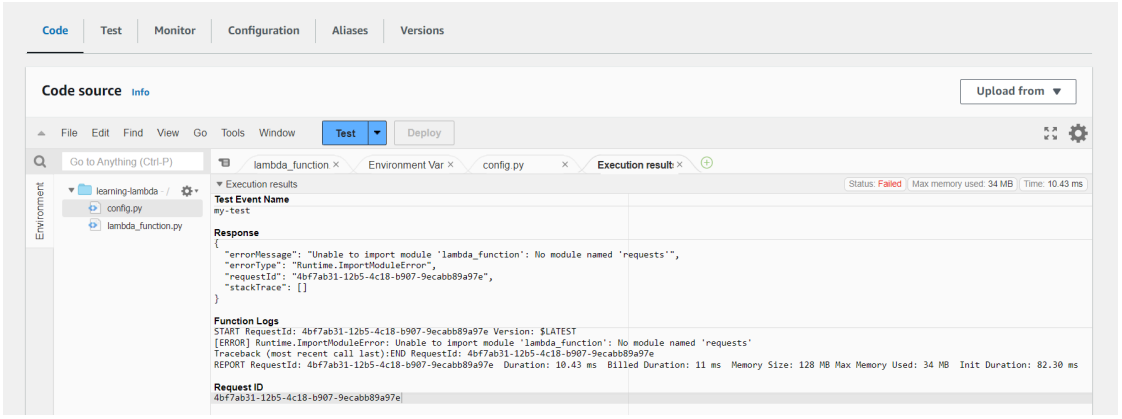
Successfully updated the function learning-lambda.



Step 4 :- Testing the function

Press Test button

Error :- "Unable to import module 'lambda_function': No module named 'requests'"



If the same error occurs on the server we would install the request library using pip. But as we are using lambda the serverless architecture so we are going to use the “Layers” concept.

A Lambda layer is a .zip file archive that contains supplementary code or data. Layers usually contain library dependencies, a custom runtime, or configuration files.

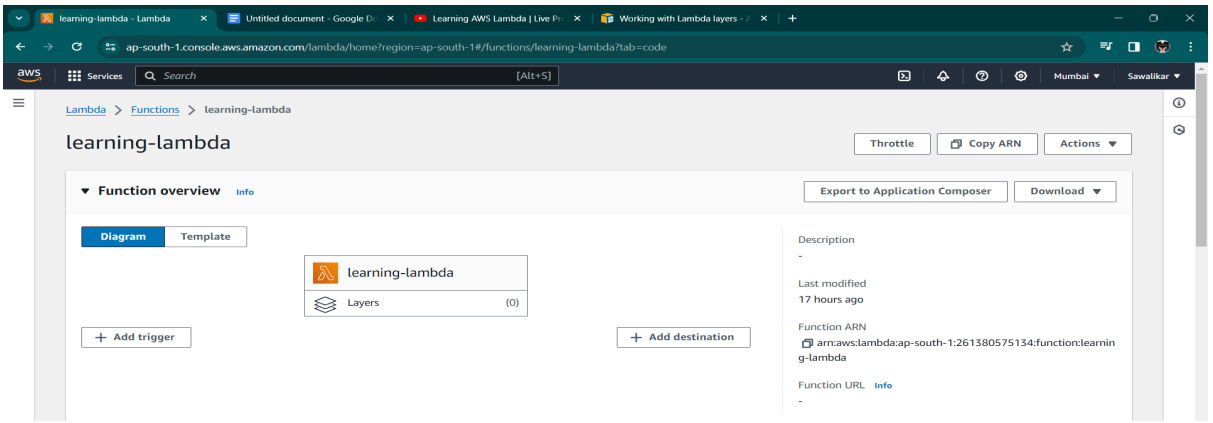
There are multiple reasons why you might consider using layers:

- To reduce the size of your deployment packages. Instead of including all of your function dependencies along with your function code in your deployment package, put them in a layer. This keeps deployment packages small and organised.
- To separate core function logic from dependencies. With layers, you can update your function dependencies independent of your function code, and vice versa. This promotes separation of concerns and helps you focus on your function logic.
- To share dependencies across multiple functions. After you create a layer, you can apply it to any number of functions in your account. Without layers, you need to include the same dependencies in each individual deployment package.
- To use the Lambda console code editor. The code editor is a useful tool for testing minor function code updates quickly. However, you can’t use the editor if your deployment package size is too large. Using layers reduces your package size and can unlock usage of the code editor.

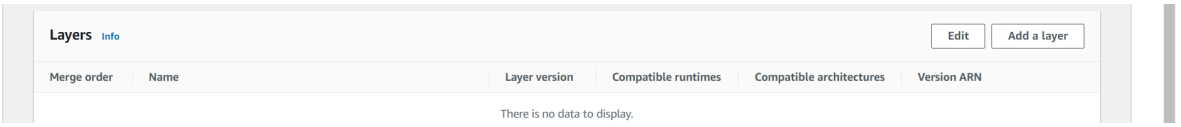
PART 3

Adding Layers

Step 1:- Click on Layers present below learning-lambda

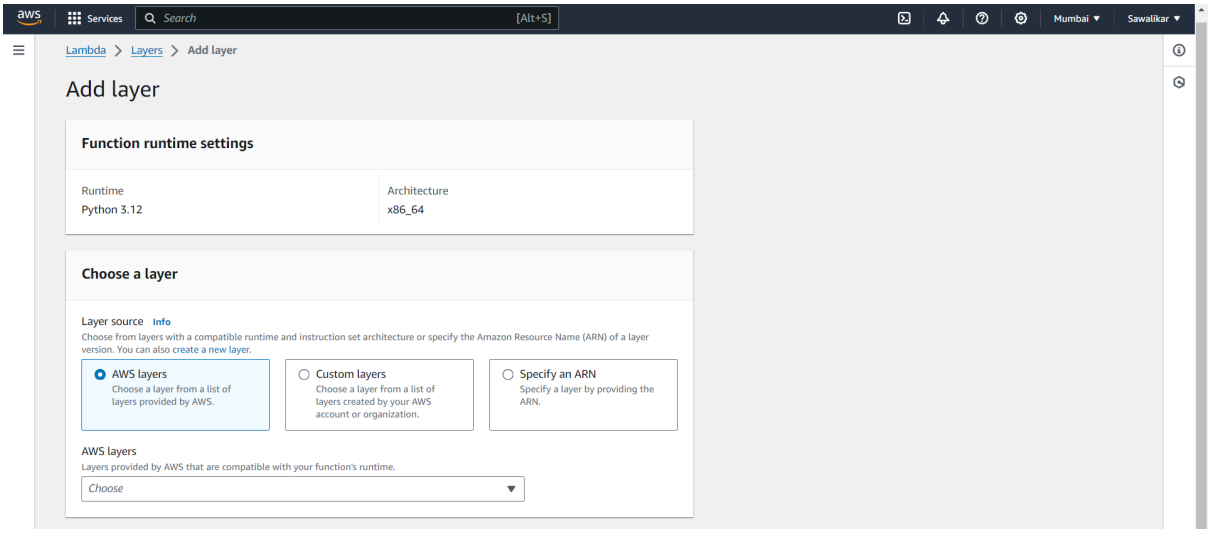


Step 2 :- click add a layer

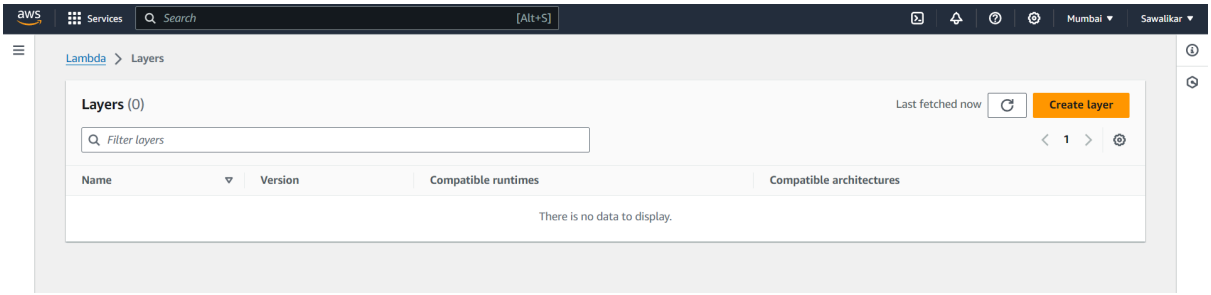


Right now we don't have layers so we will create a new layer first then we will add it.

Step 3 :- Click on Layers at the top



Press Create Layer

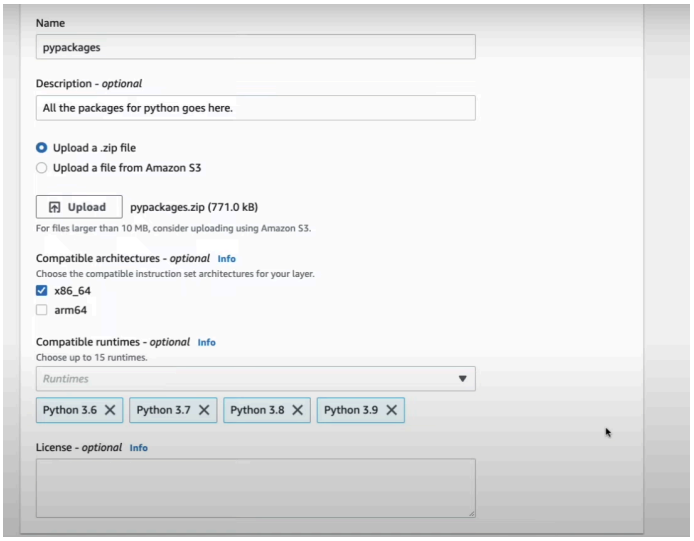


Now open a server and create a zip file (Use vs code to create this zip file by doing ssh)

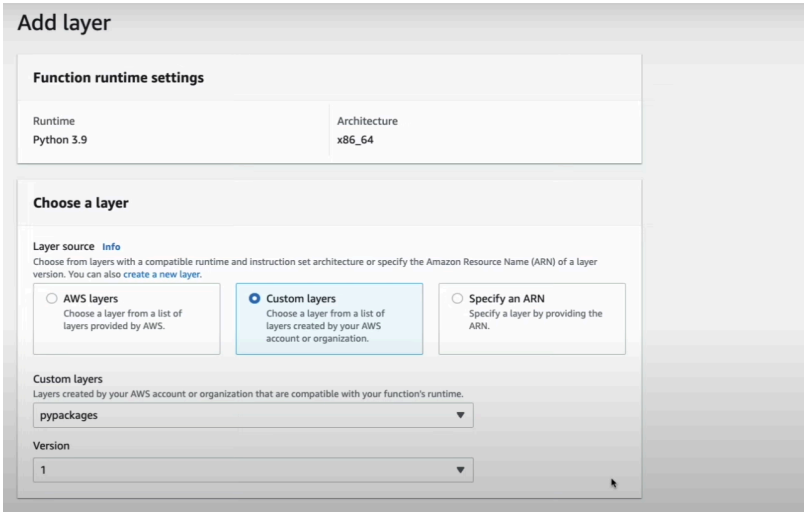
$ mkdir pypackages
$ ls
$ cd pypackages
$pwd
(Copy the path)
$ cd ..
$ pip3 install requests -t path_of_pypackages
pip3 install requests -t /home/ubuntu/pypackages
(In my case its /home/ubuntu/pypackages)
$ ls

Now creating zip
$ sudo apt install zip
$ zip -r pypackages.zip pypackages/
$ ls

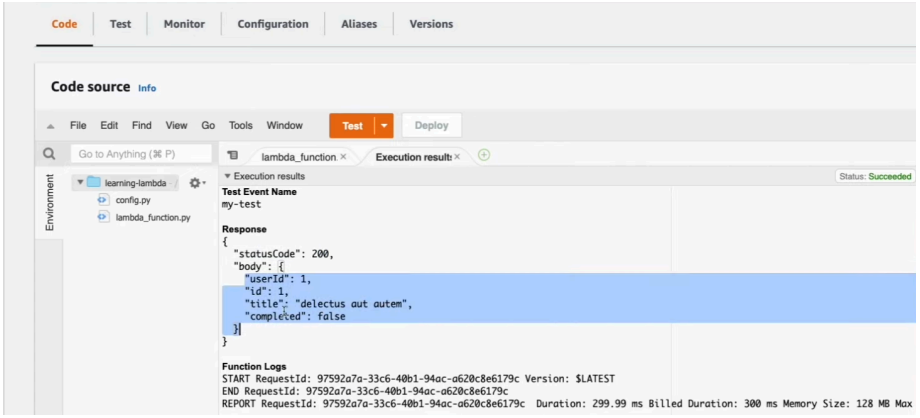
Step 4 :- Now Upload the zip file -> Select x86_64 ->Select all versions of Python



Step 5 :-Adding layer ->Choose custom layer (pypackages) ->Give version



Step 6 :- Now test the code again

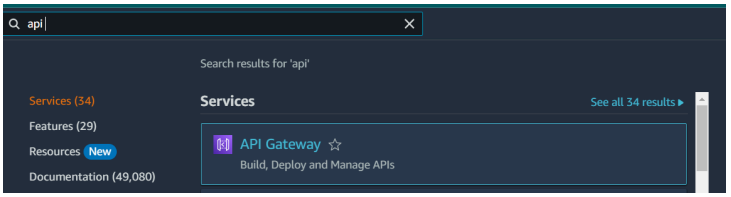


PART 4

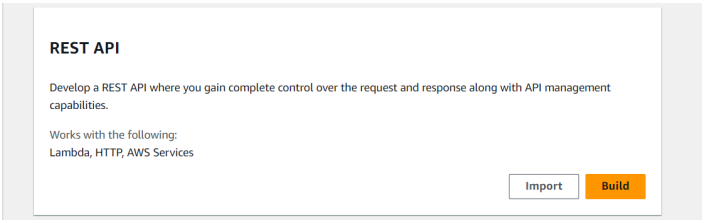
Now we are creating a rest api that will power our function.

What is an API gateway ?
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.

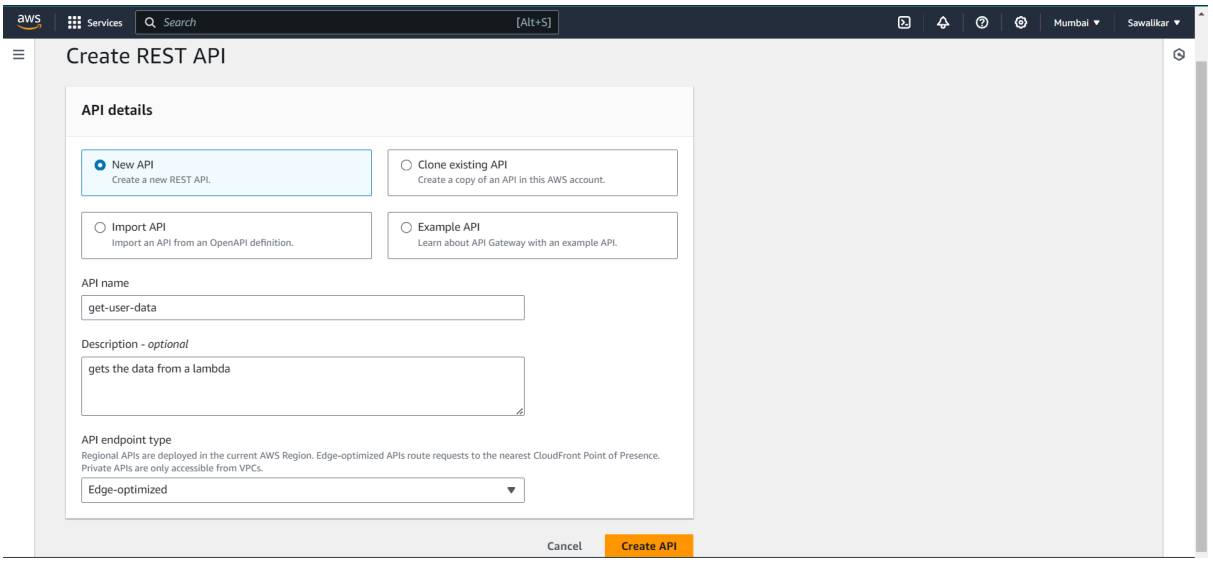
Step 1 :- Search api gateway -> Select api gateway



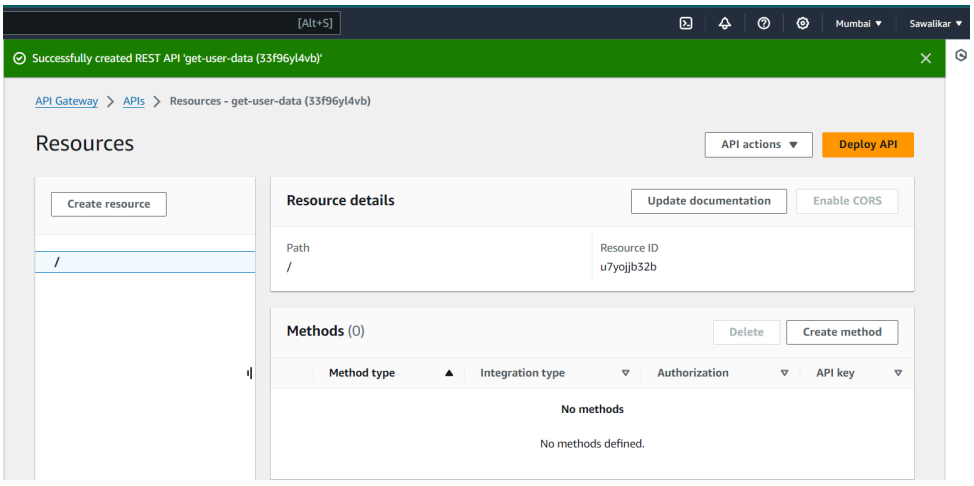
Step 2 :- Select REST Api -> Click Build



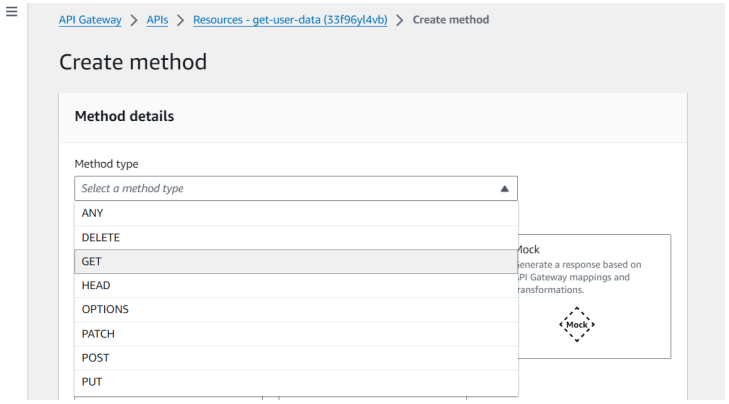
Step 3 :-



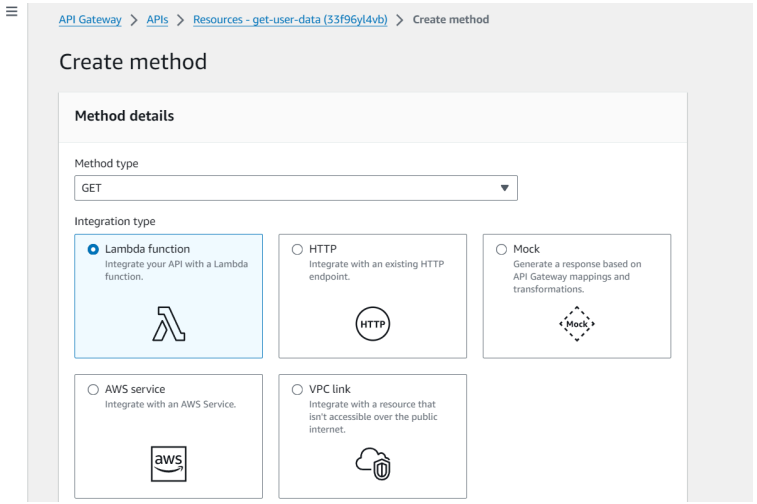
Now select Create Method



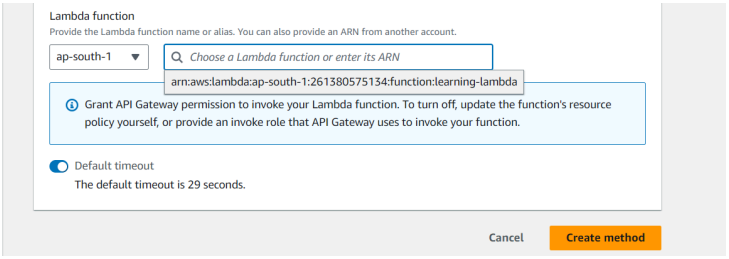
Select get method

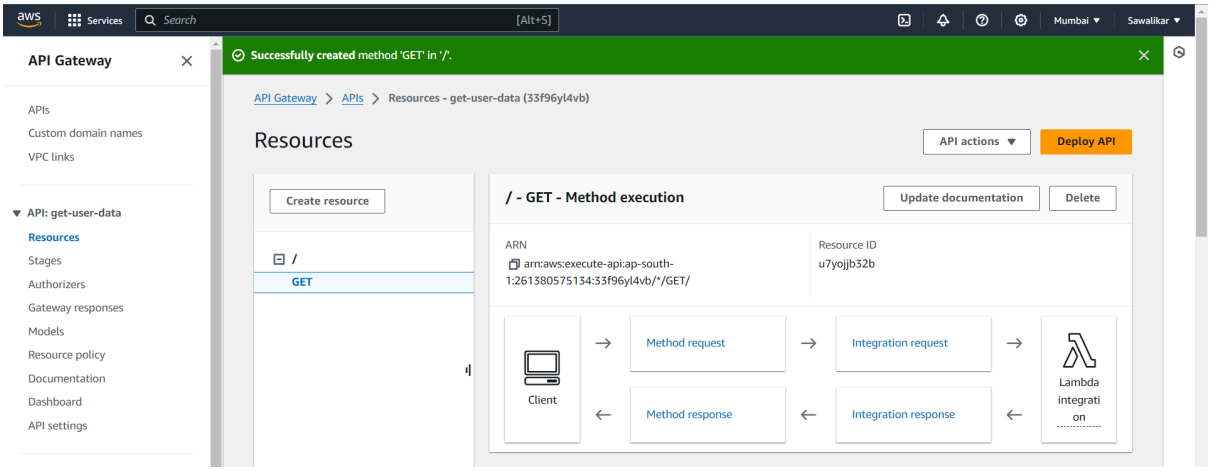


Integration type -> Lambda function

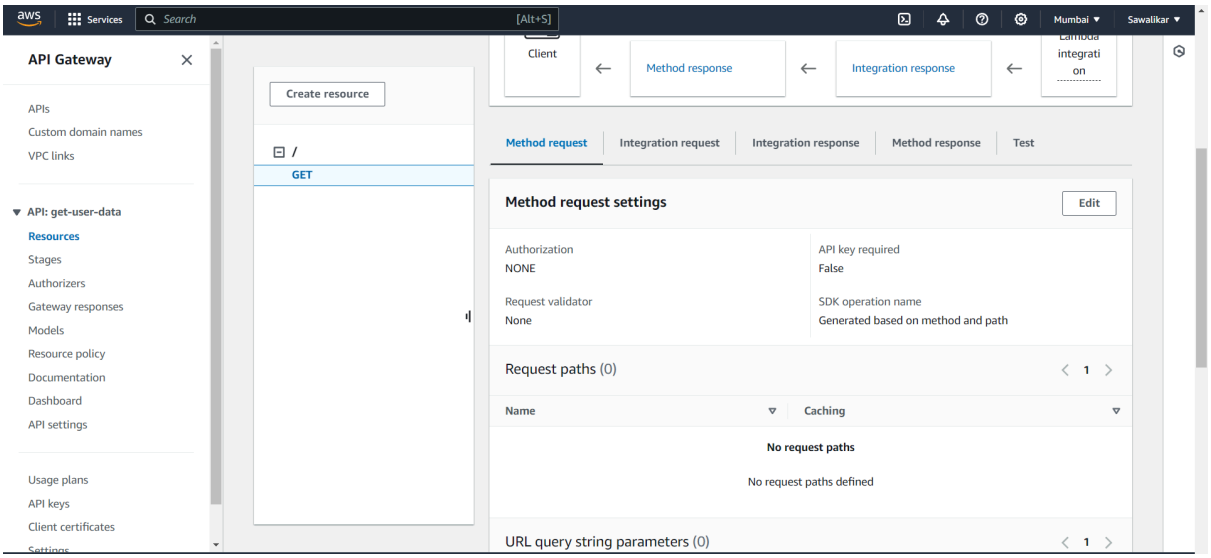


Select above created function



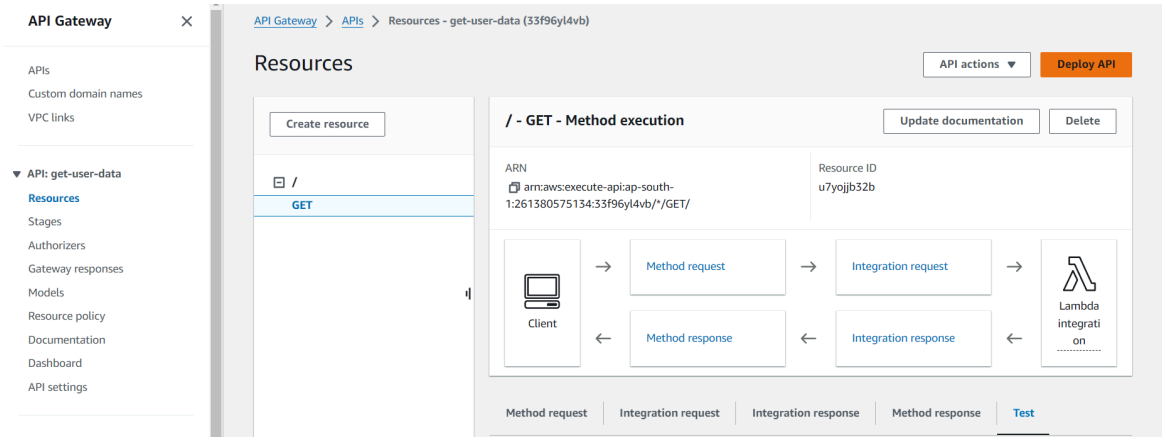


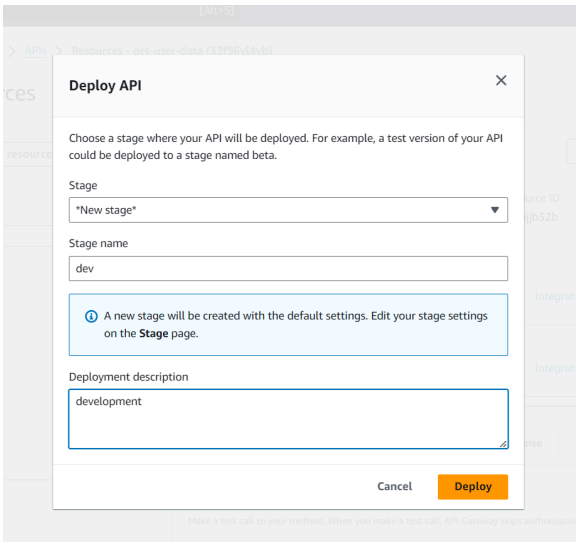
Testing the api
Click on Test



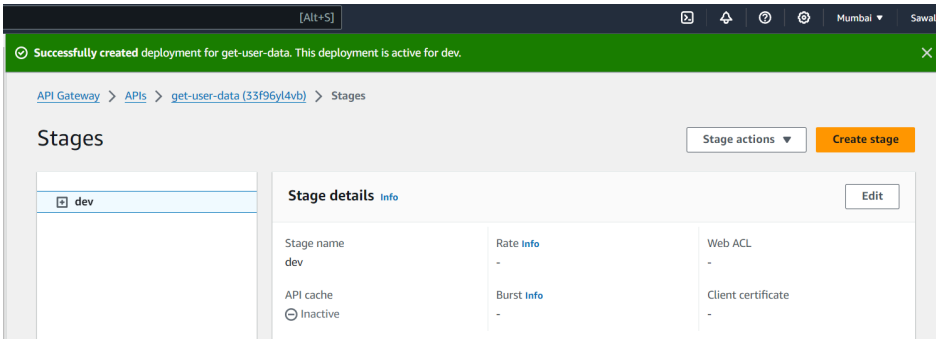
Deploying the api

Press Deploy API button





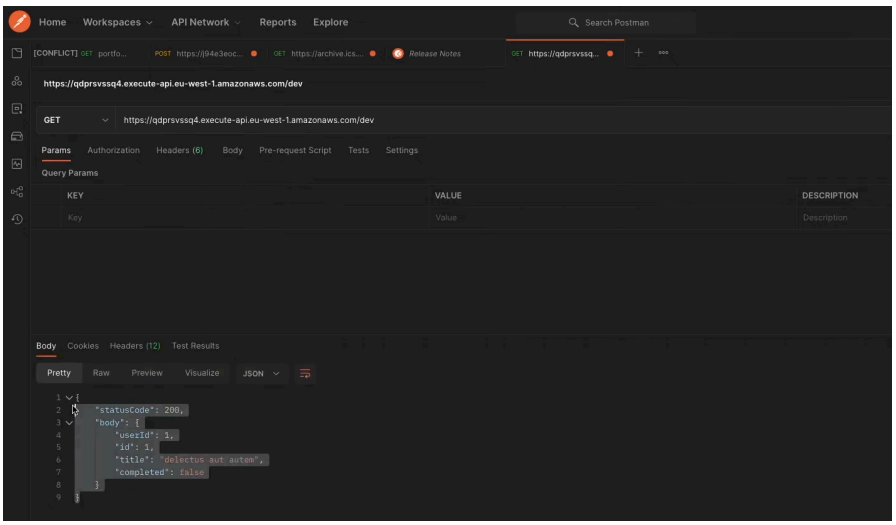
Successfully created deployment for get-user-data



PART 5

Open postman

Selecy get -> Copy the API url



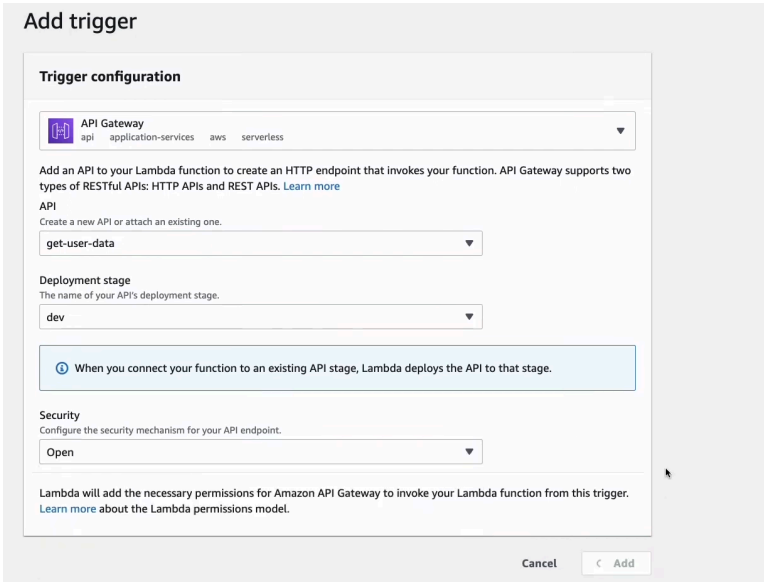
PART 6

Adding trigger

Means whenever my api will get trigger it will call my lambda function

Click on add trigger

Select api gateway -> select our created API ->Select deployment stage (dev) -> Select security (open) -> Click add


Add trigger
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.
get-user-data
Deployment stage
The name of your API's deployment stage.
dev
When you connect your function to an existing API stage, Lambda deploys the API to that stage.
Security
Configure the security mechanism for your API endpoint.
Open
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