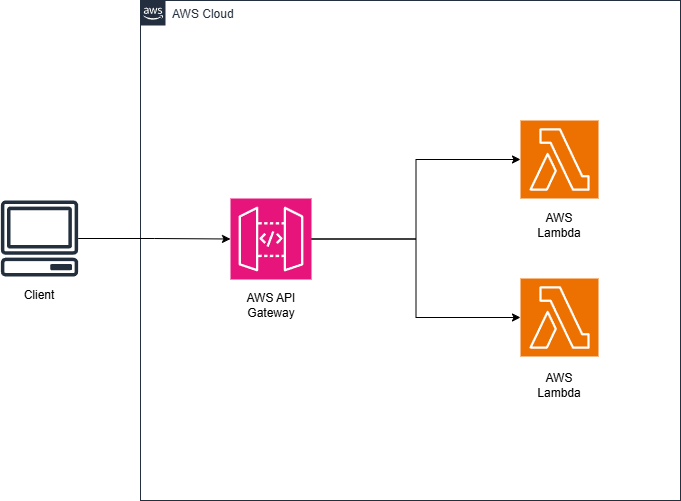
Integrate AWS with Lambda and API Gateway and Monitoring

Architecture :



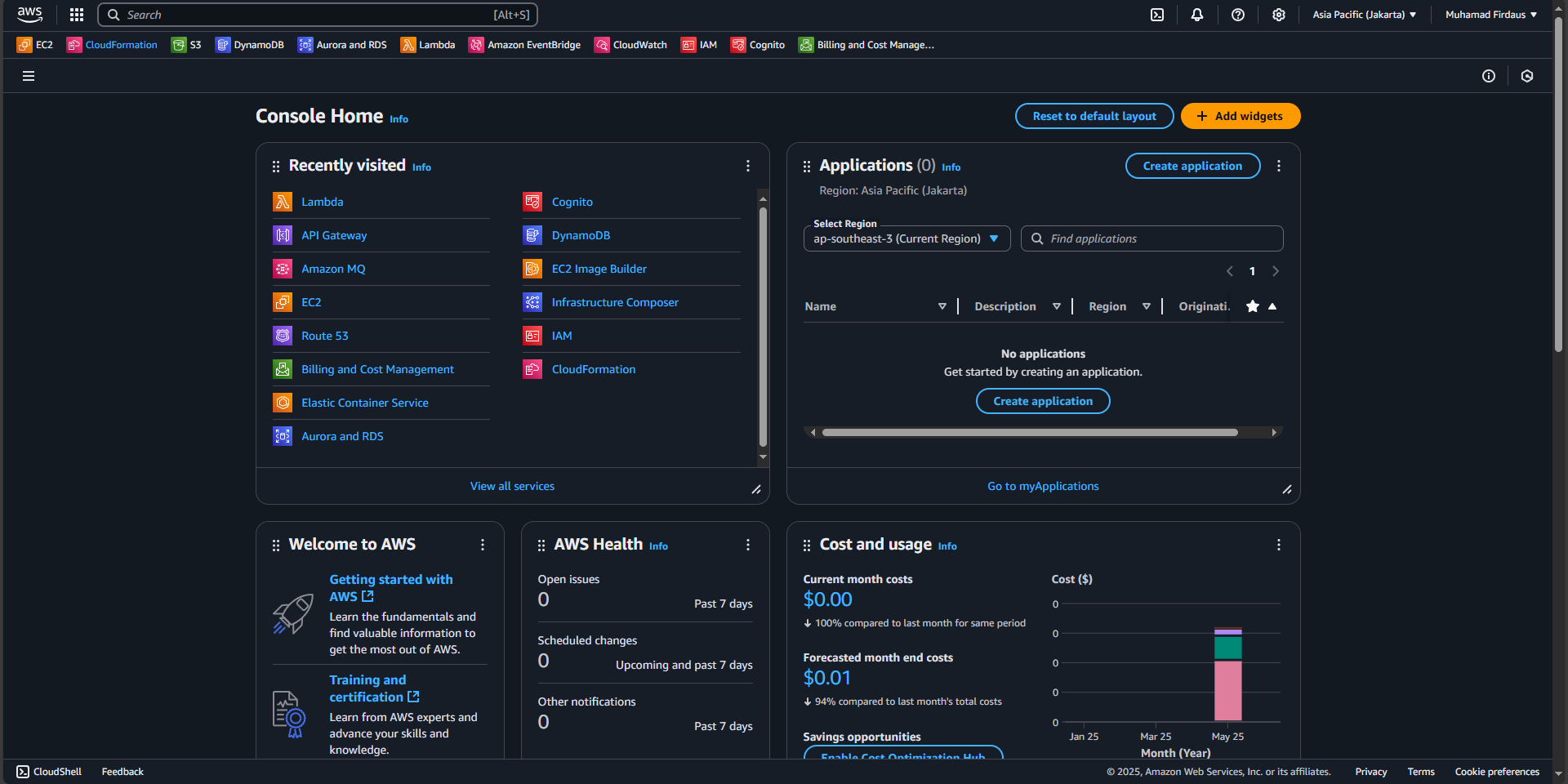
Service used :

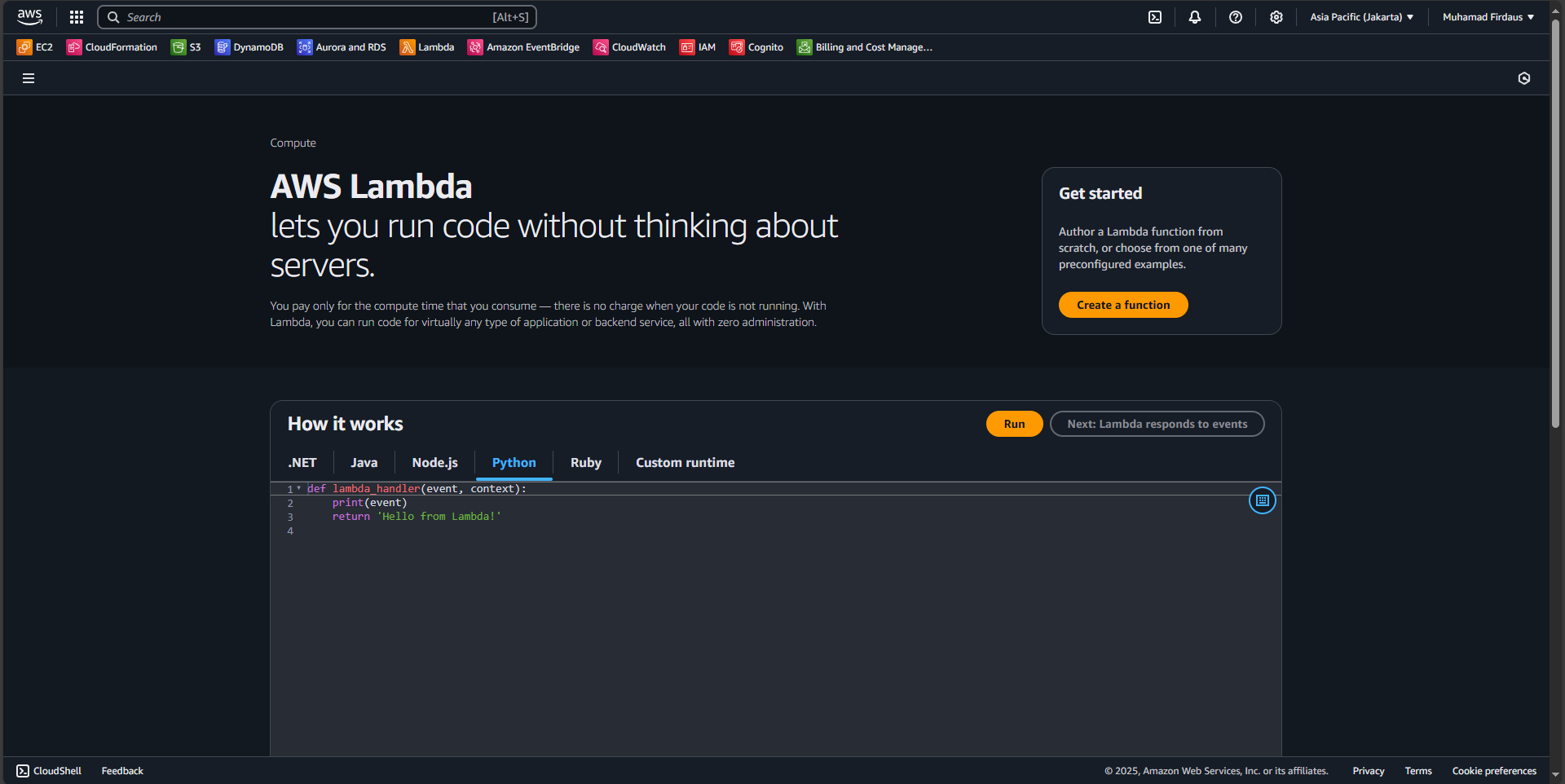
* AWS Lambda
* AWS API Gateway

Walkthrough:

**PART 1 : Creating the Lambda and monitor the function logs**

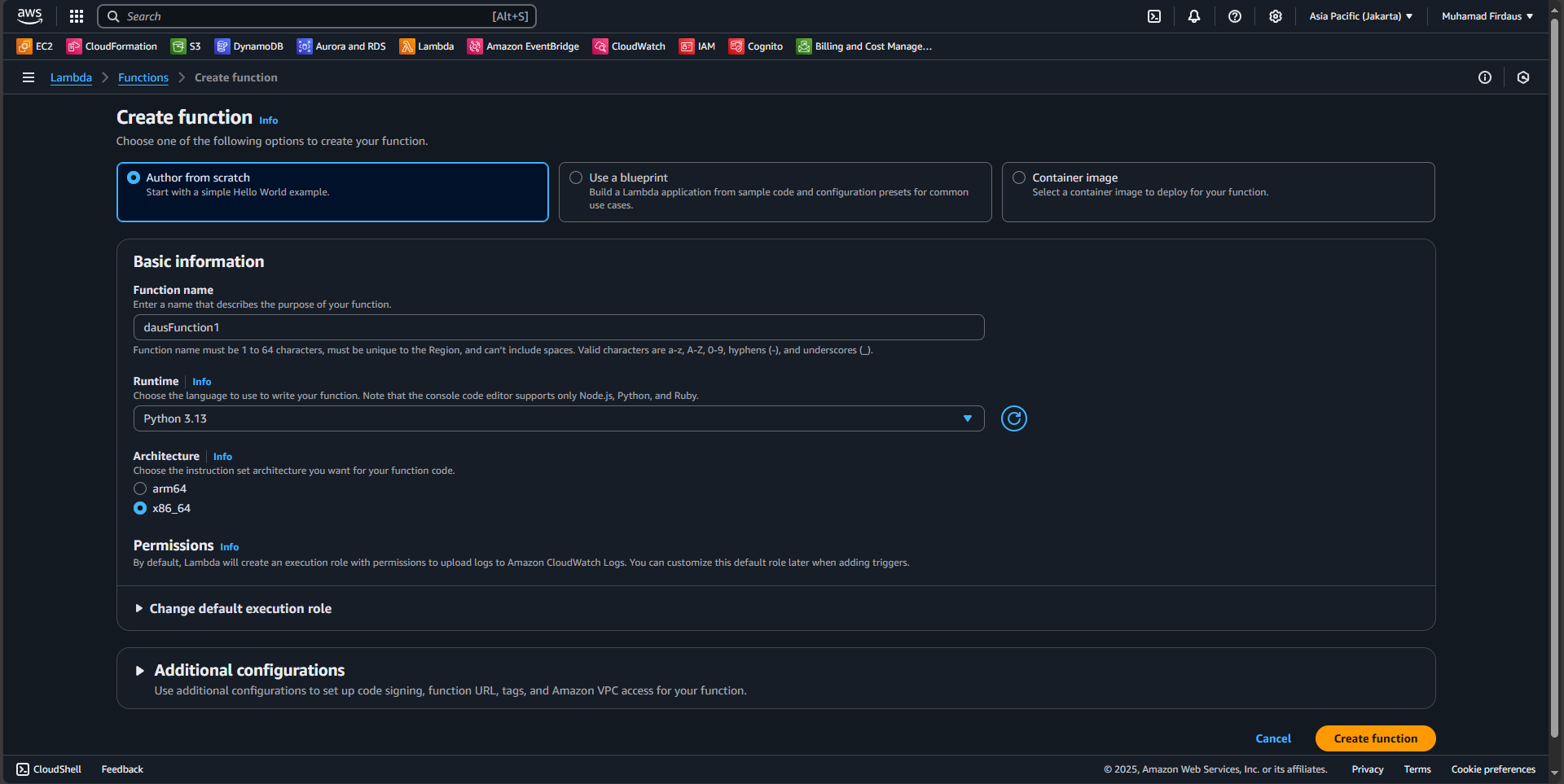
1. Open up the AWS Console and open the Lambda services, choose to create a new Function.



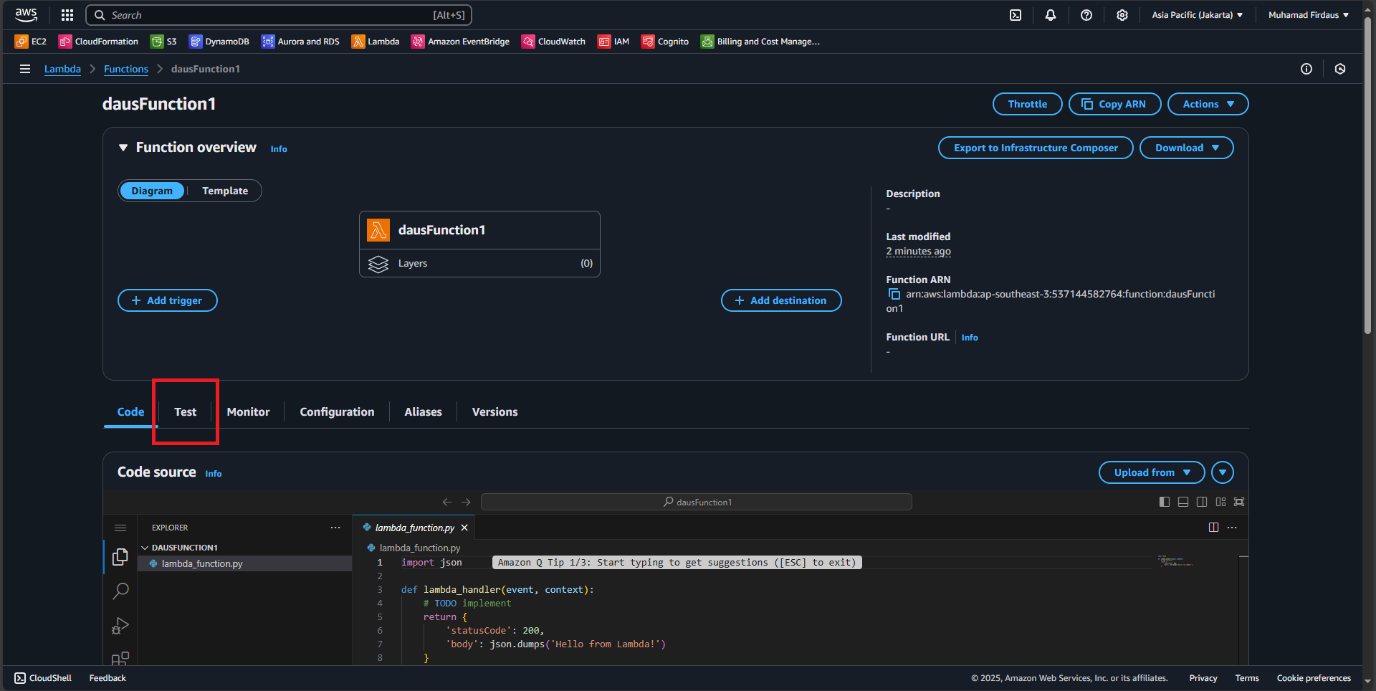


1. Fill the function details with information that you want, in this case I am using below options to fill the details and after that choose to create the function :

|  |
| --- |
| Option : Author from scratch  Function name : dausFunction1  Runtime : Python  Architecture : x64 |

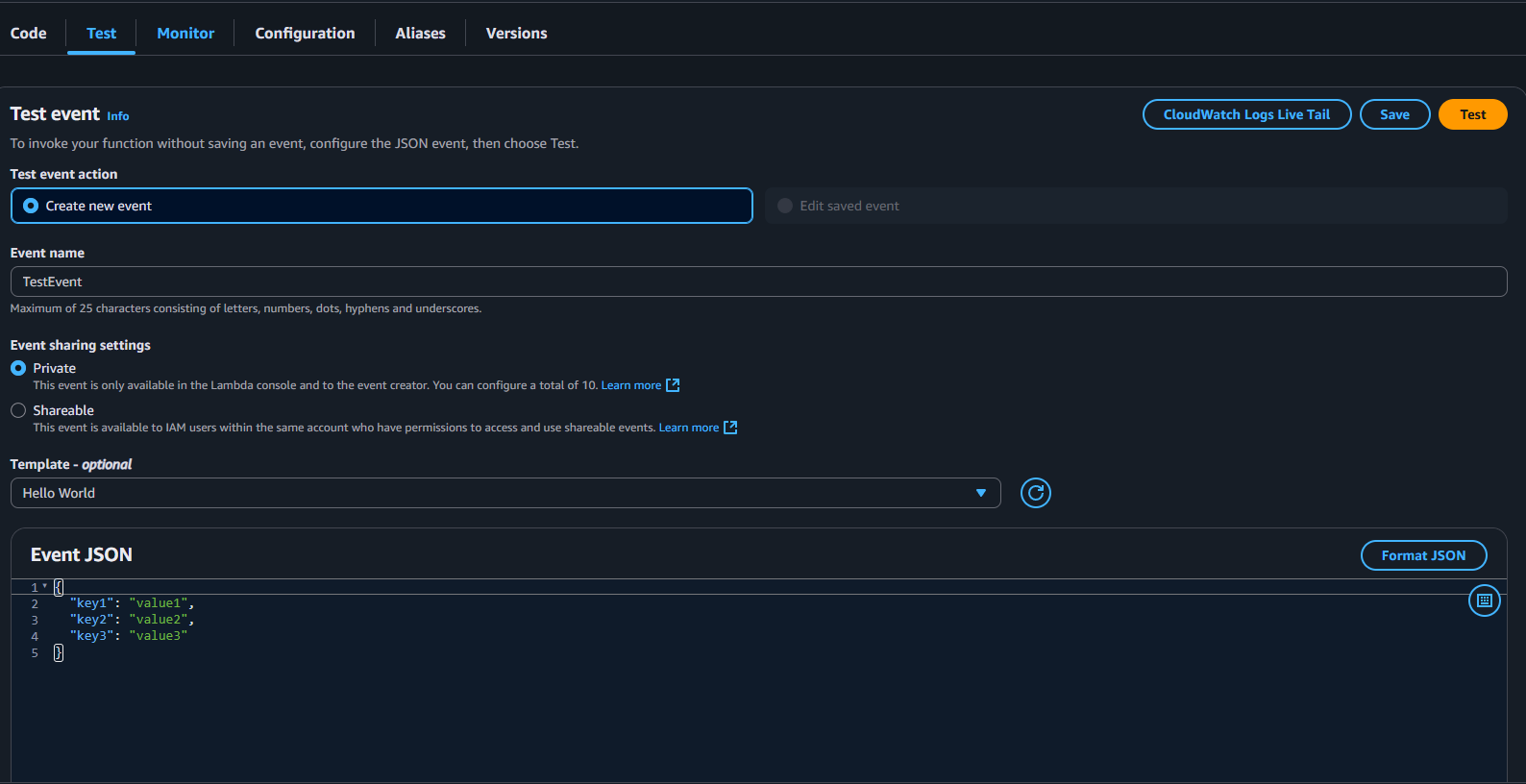


1. A pop up will show after the service created the new function, after the pop up choose the “Test” button on the bottom screen to open the section.

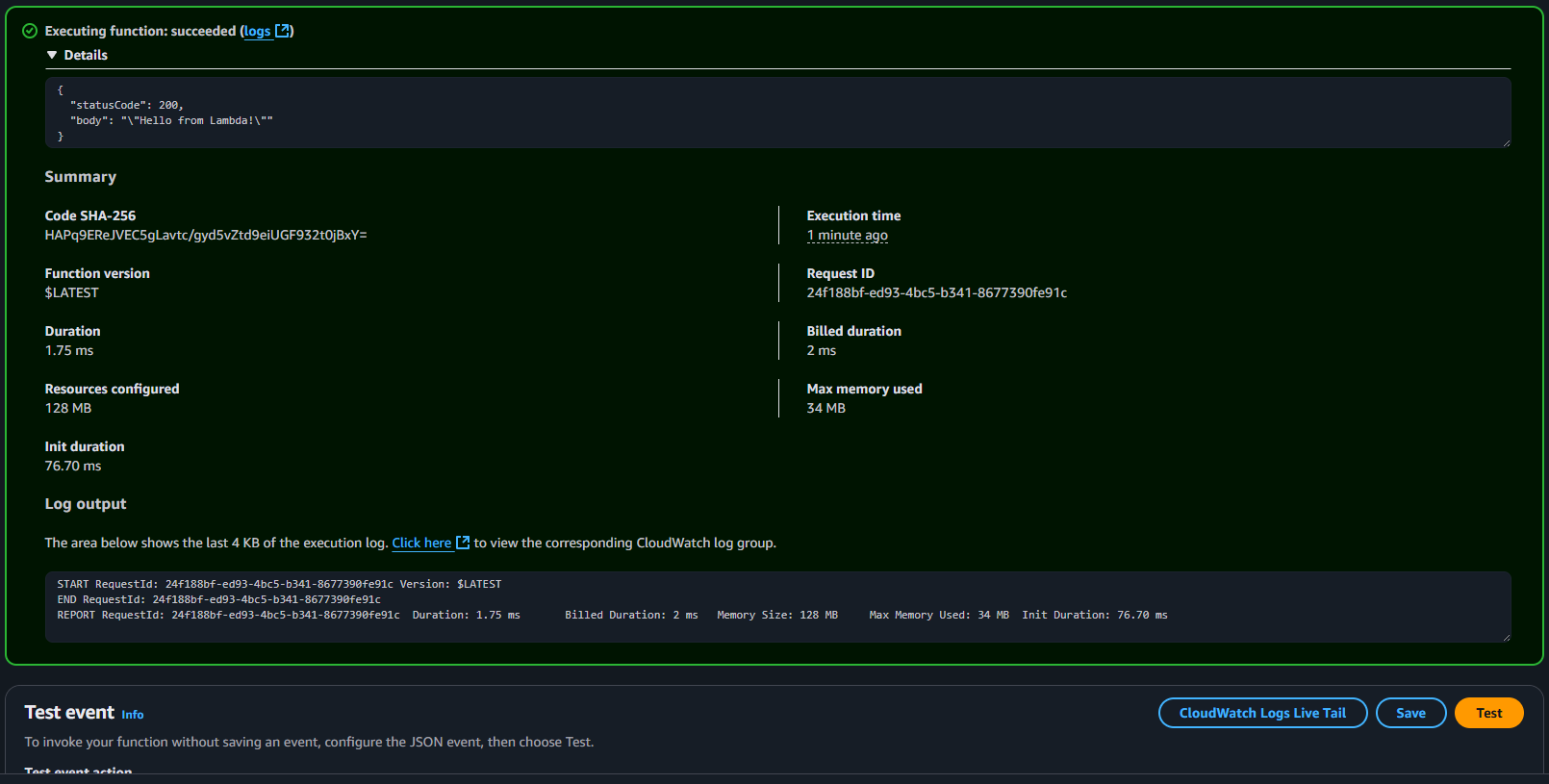


1. On the “Test” section, edit the details with the following details so we can use it to test the function and after that click on the orange “Test” button on the right above screen.

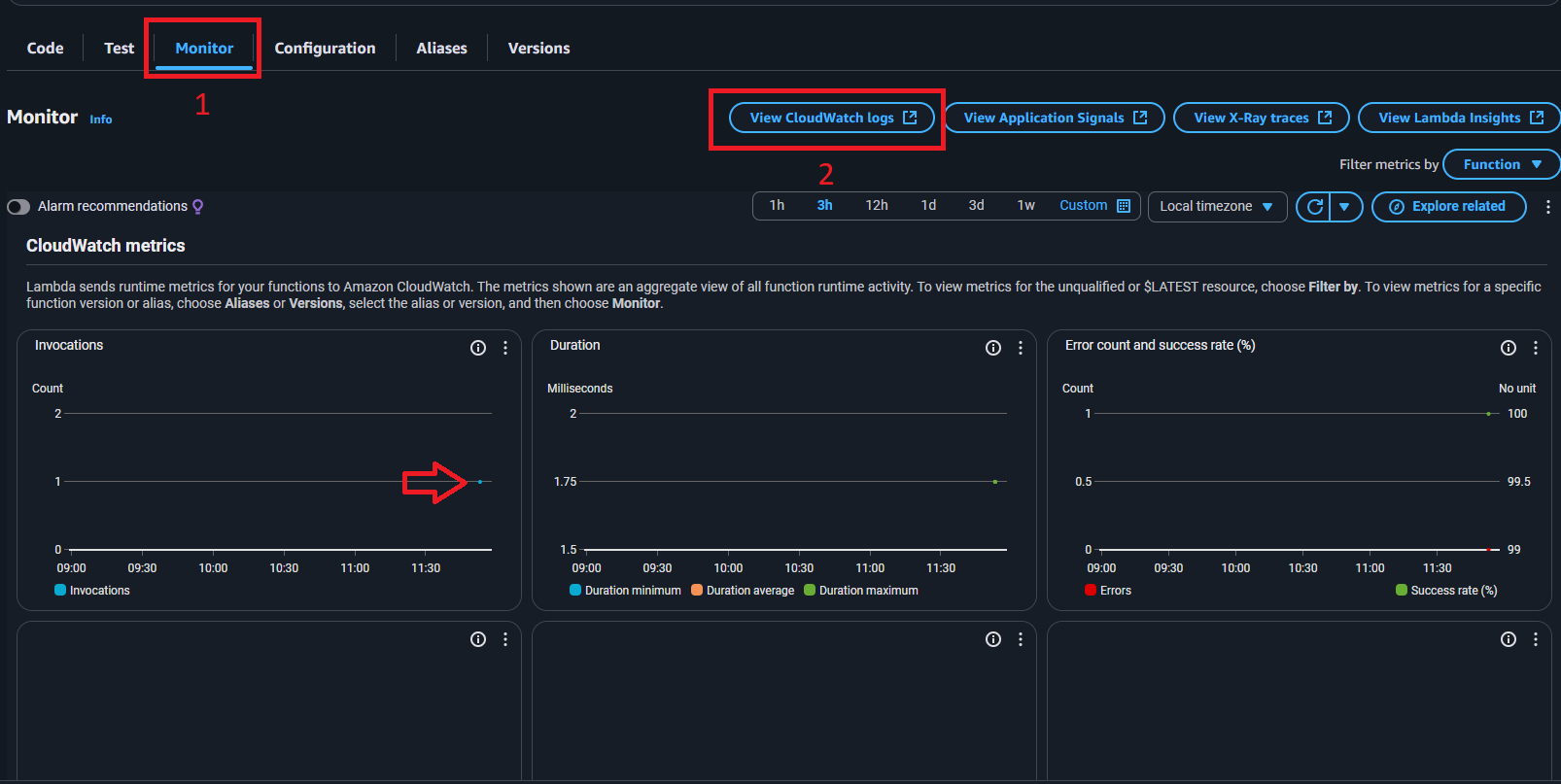
|  |
| --- |
| Test event action : Create new event  Event name : TestEvent  Event sharing settings : Private  Template : Hello World |



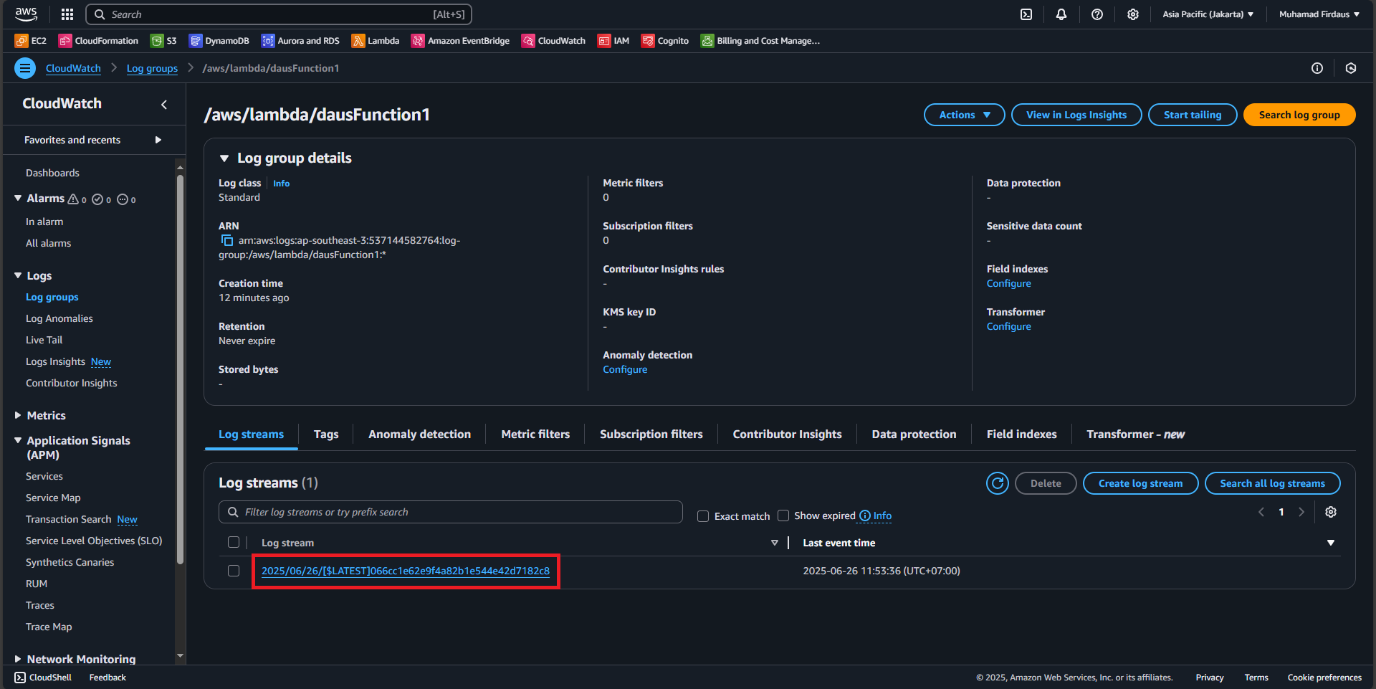
1. The result of the function test will then pop up to give us the details regarding the function testing.



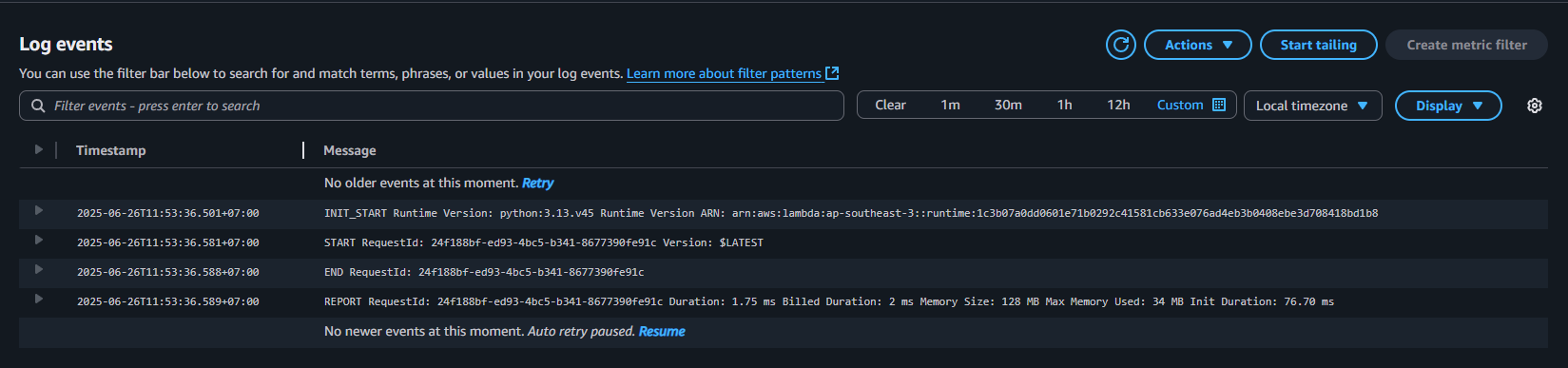
1. Now we can see the CloudWatch logs regarding the function testing on the CloudWatch section, click on “Monitoring” and choose “View CloudWatch Logs”. On the monitoring tab you can also see a dot that list our testing history activity.



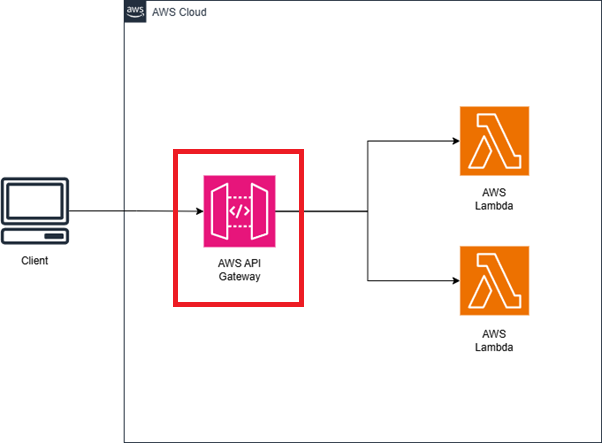
1. On the CloudWatch pop up, choose the newest log name on the log stream (the blue one) and click it to get redirected to the logs menu.



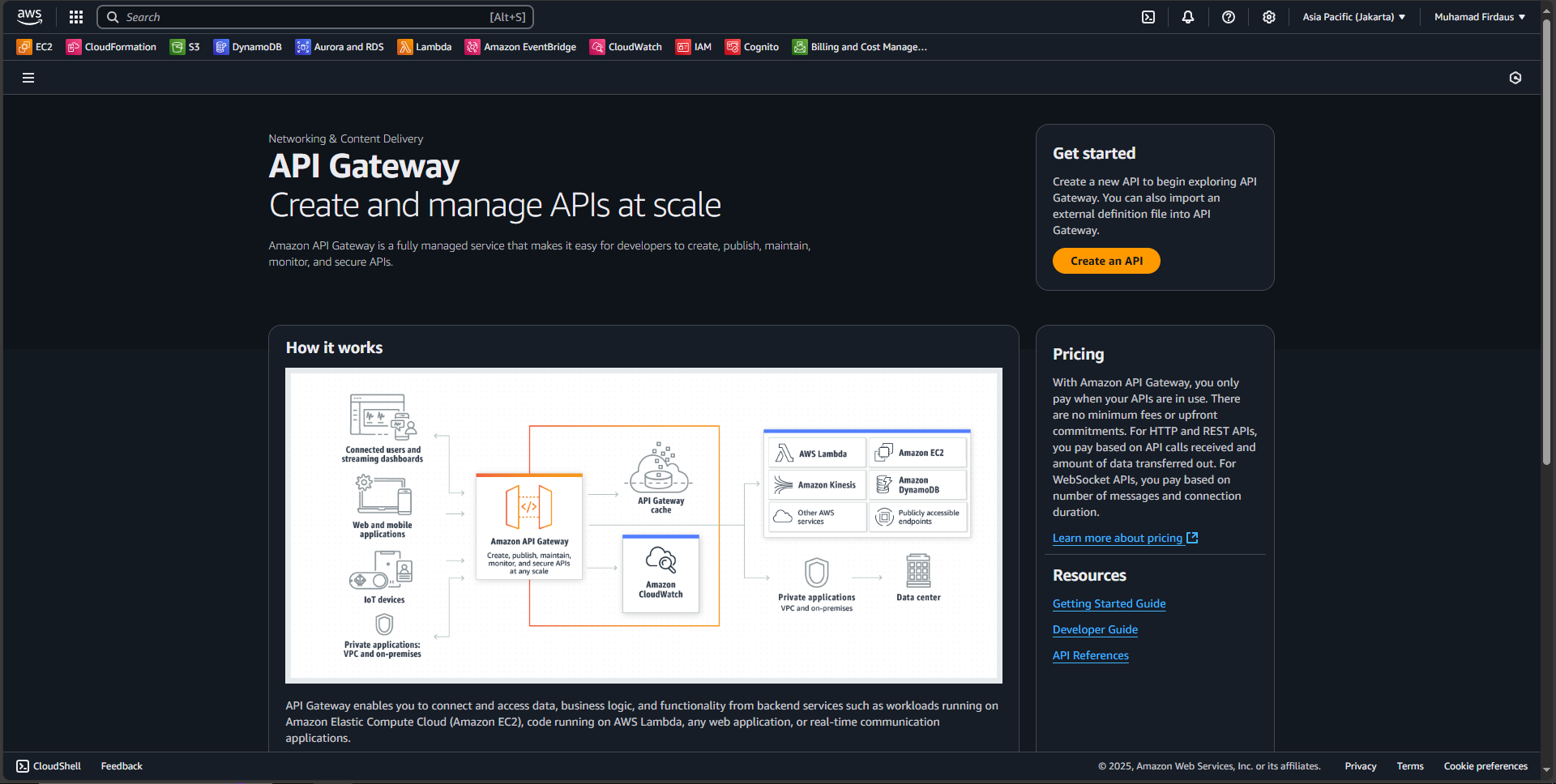
1. The log event then will show the detail regarding the running function process that we can see.



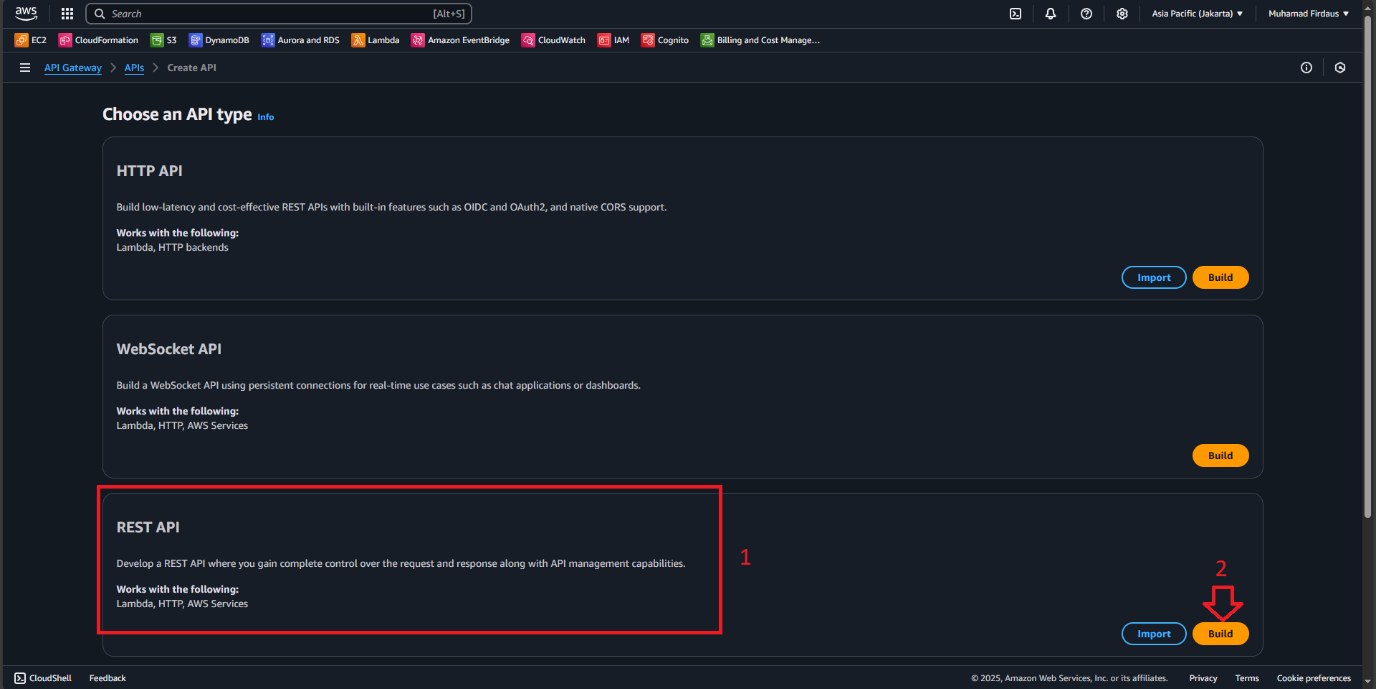
**PART 2 : Connecting Lambda function with the API Gateway**

****

1. On the AWS Console, search for API Gateway service and enter the service. We will create the API Gateway (I will mention it further as APIGW) to connect to the lambda function. Choose to create a new API.

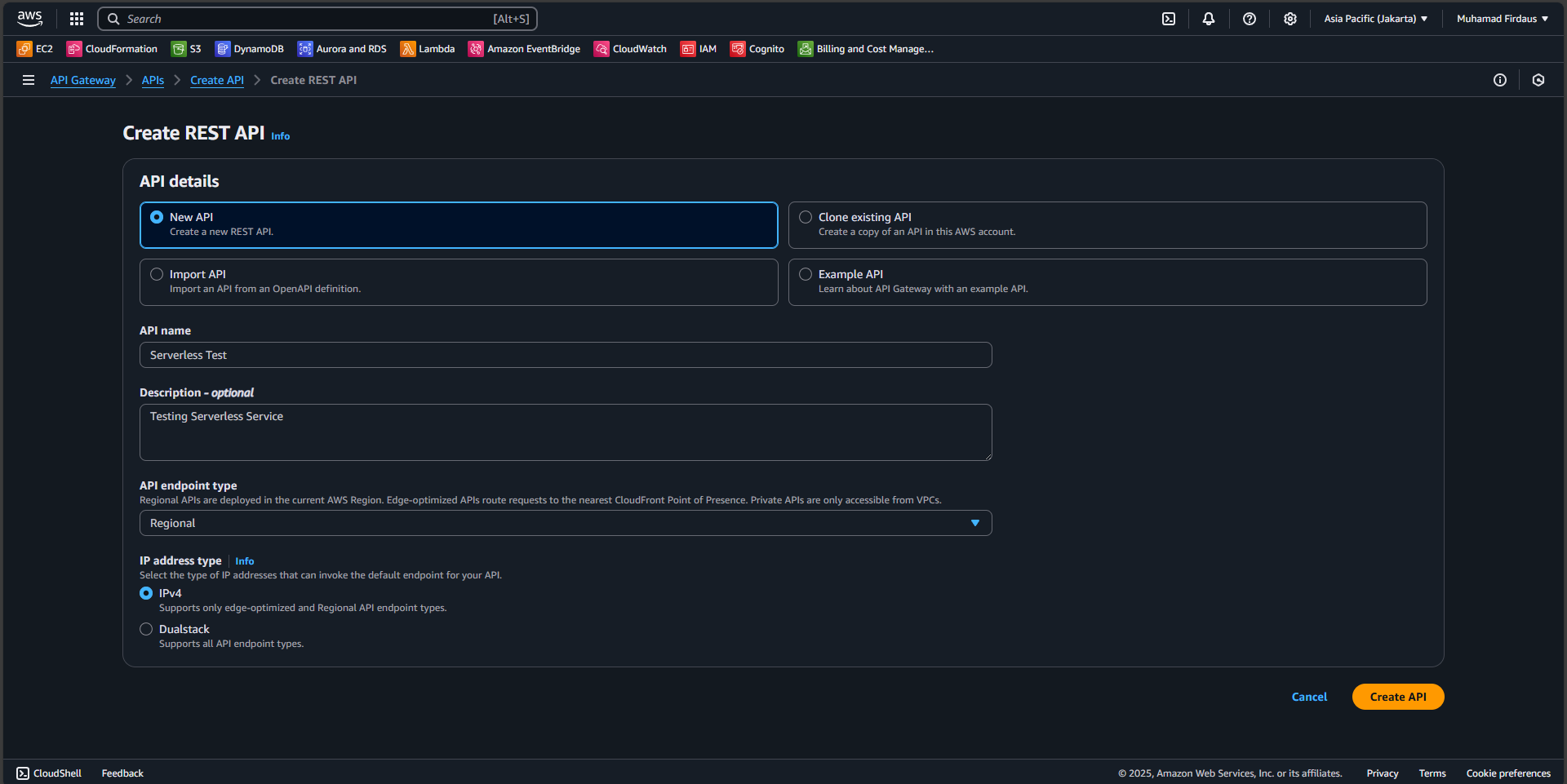


1. On the creation page, you will be asked with what kind of API you want to create. We will continue by using REST API (not the private one).

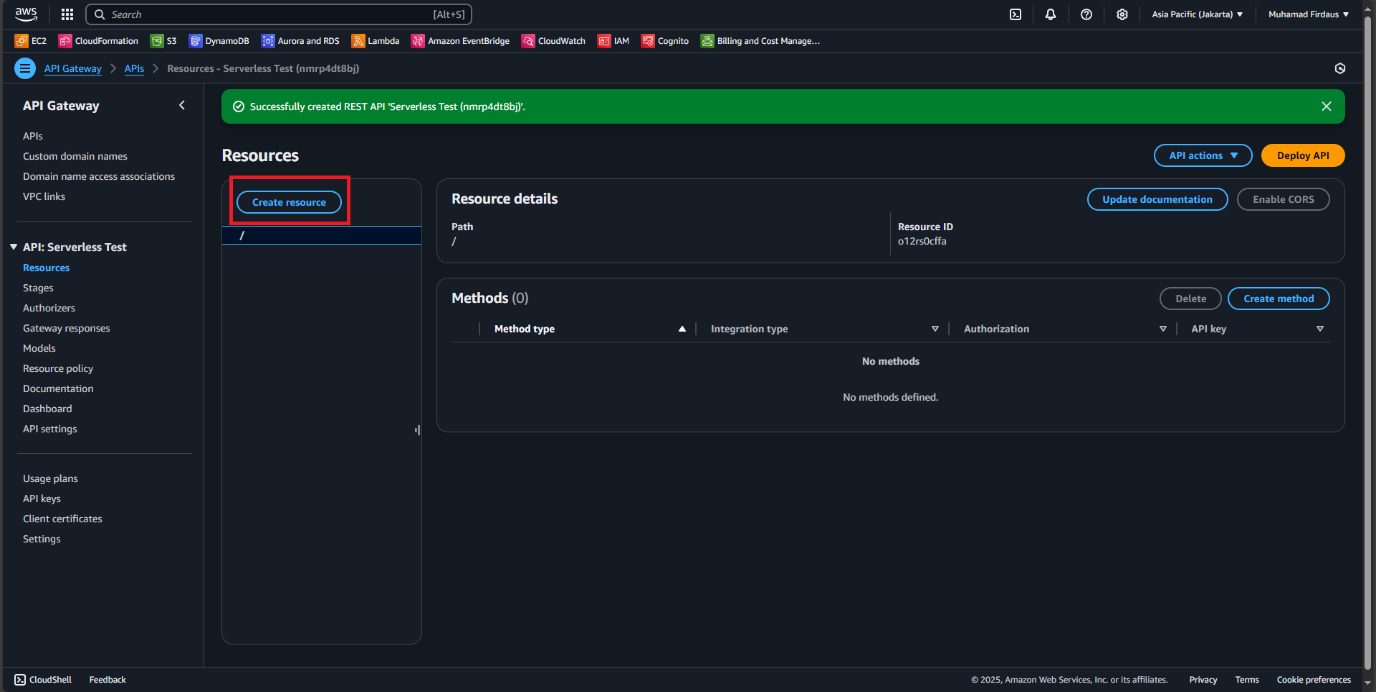


1. On the API creation page, choose to fill the details as you see fit. In this case I will be using below details to continue and choose to create the API :

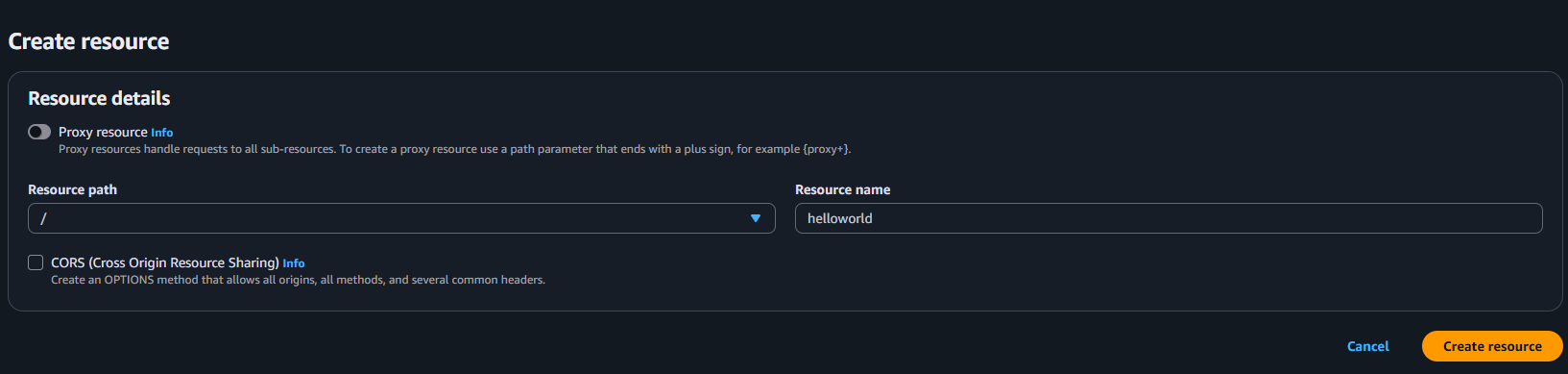
|  |
| --- |
| API Details : New API  API name : Serverless Test  Description : Testing Serverless Service  API endpoint type : Regional  IP address type : IPv4 |



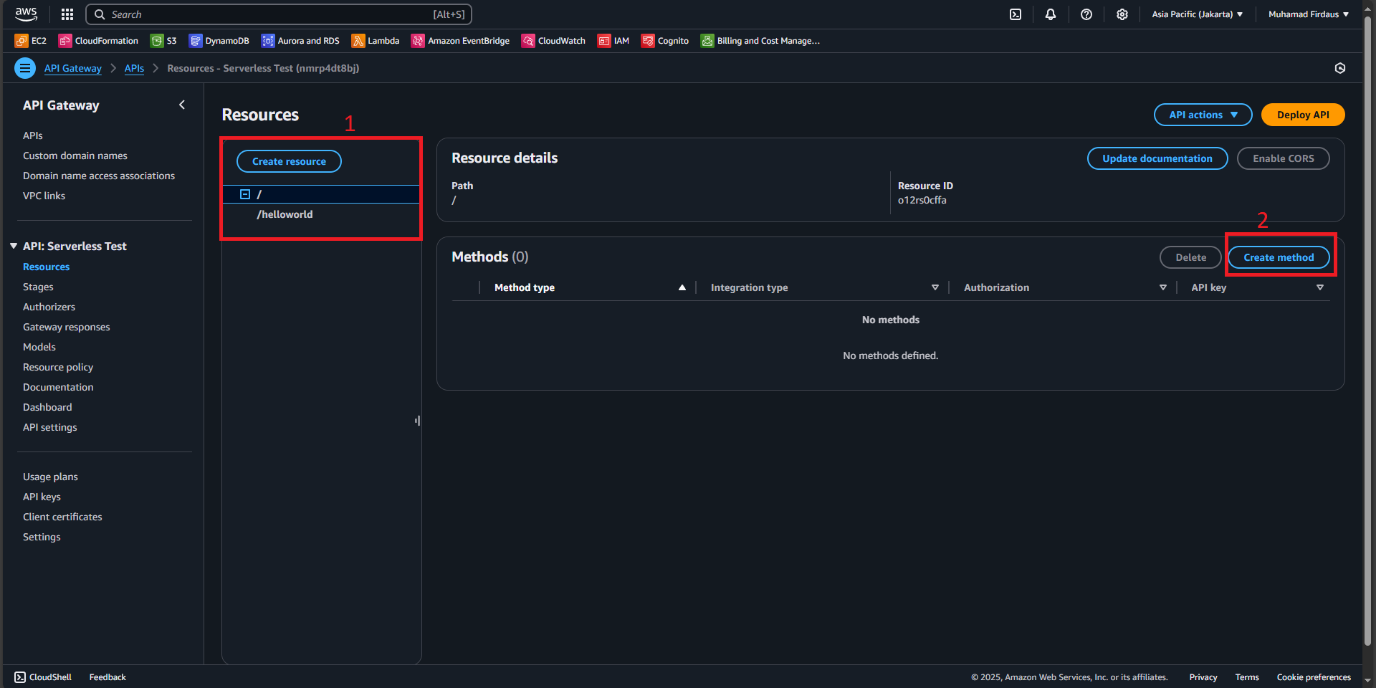
1. A page will then popup showing you the finished process of creating the new API, now on that popup page we will choose the “Create Resource” option to create a new resources.



1. On the create resource section, fill in the details with default setting while adding the name of the resource as “helloworld”.

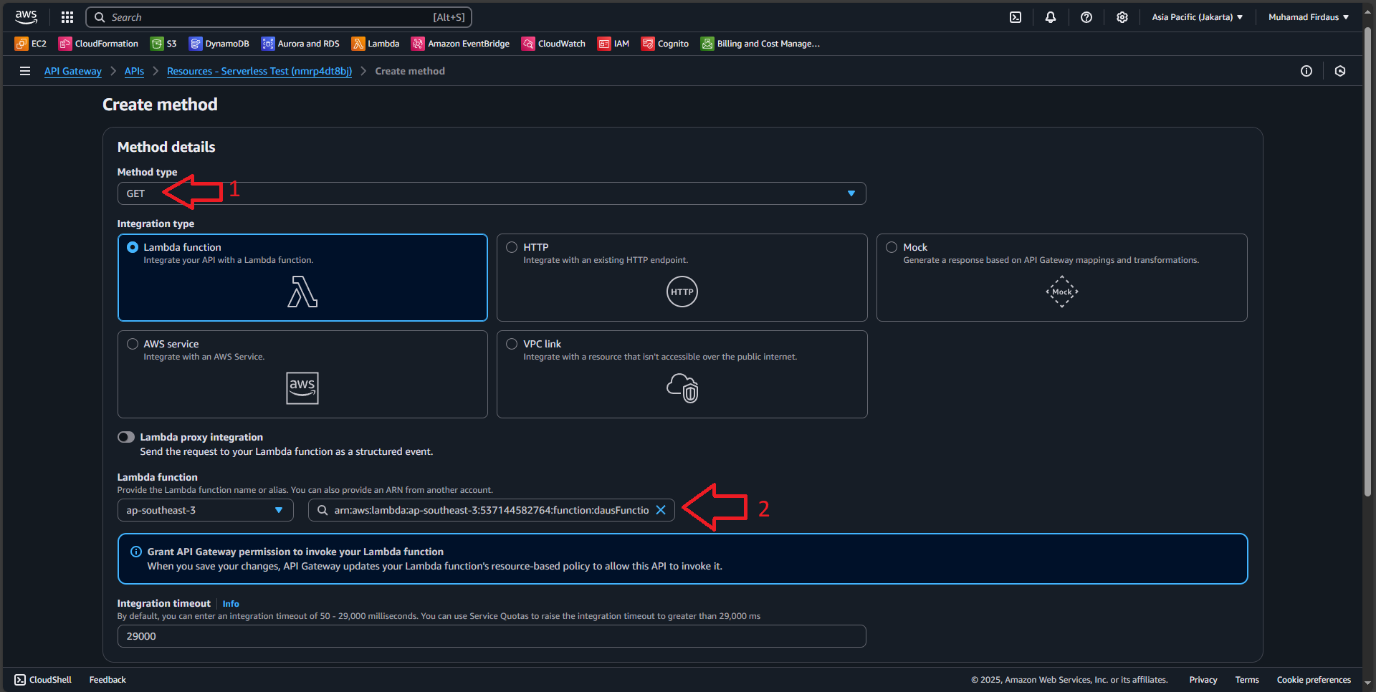


1. After creating the resource, a new pop up will show us that the resource is successfully created. And then we choose the “create method” option AFTER we click on the /helloworld to create a new method for our resource under the /helloworld resource.

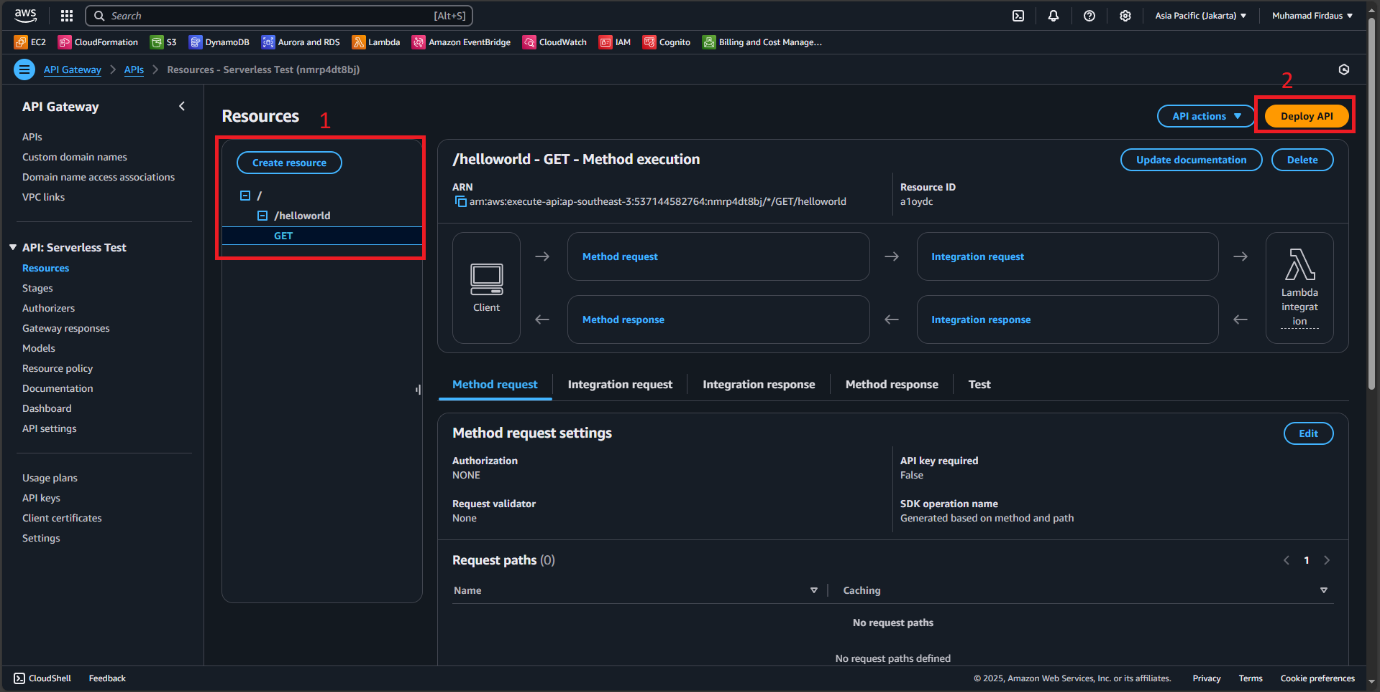


1. On the create method page, we fill the details with the below information to proceed with our method creation and then choose create method.

|  |
| --- |
| Method type : GET  Integration type : Lambda function  Lambda proxy integration : off  Lambda fucntion : our created function earlier in my case it is dausFunction1  Integration timeout : left it default |

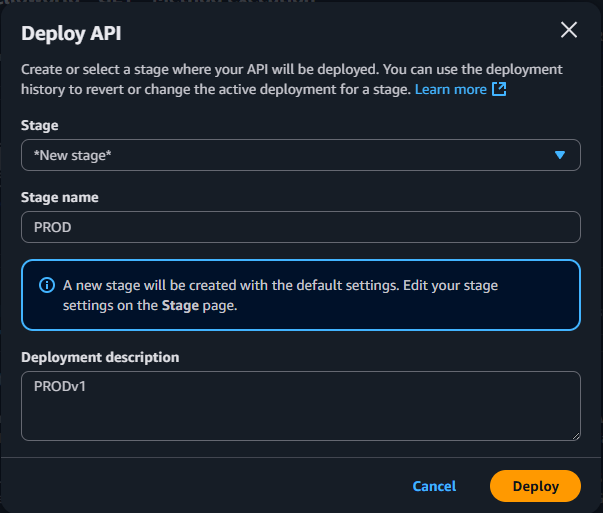


1. After the method was created, the GET method will appear under the /helloworld on the resource tab, confirm this to proceed. Then click on the “Deploy API” so we can begin to prepare to deploy our API

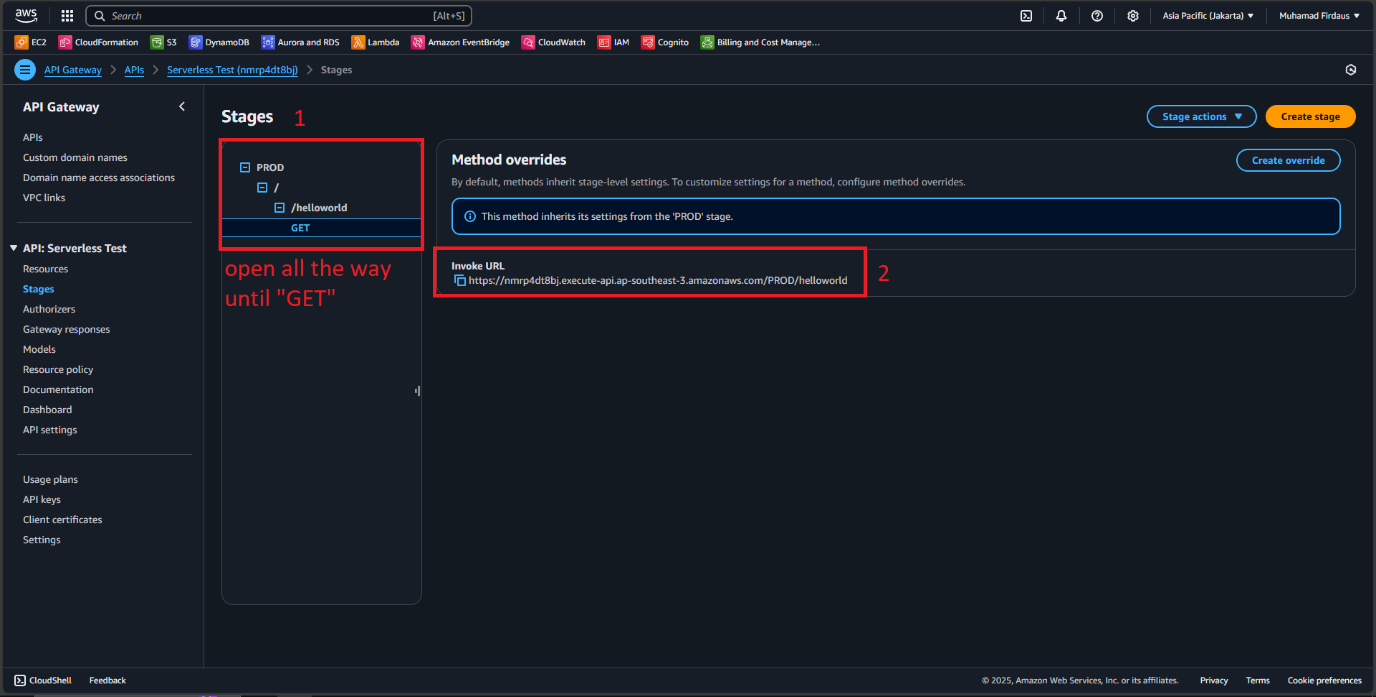


1. A popup will appear for us to configure the API deployment, use below option to fill the required field and then “Deploy”.

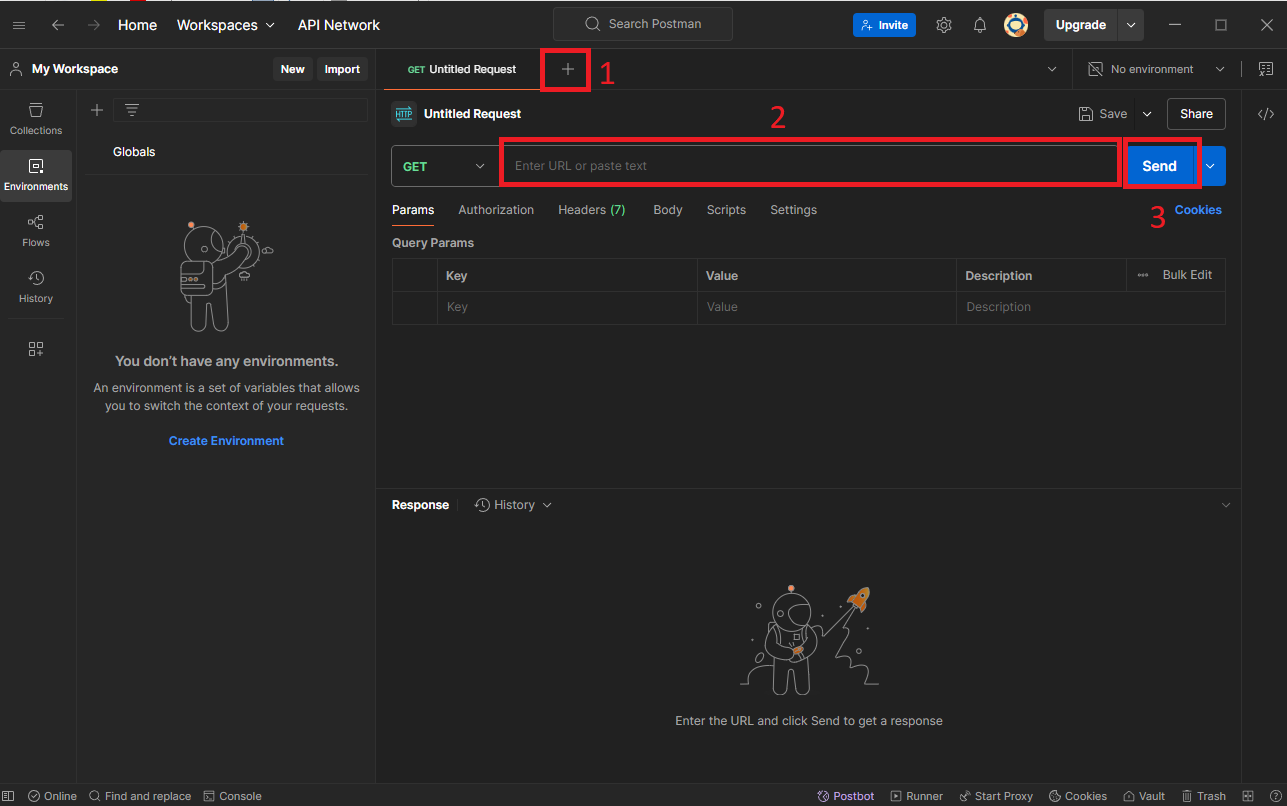
|  |
| --- |
| 1. Stage : \*New stage\* (Its a dropdown menu) 2. Stage name : PROD 3. Description : PRODv1 |



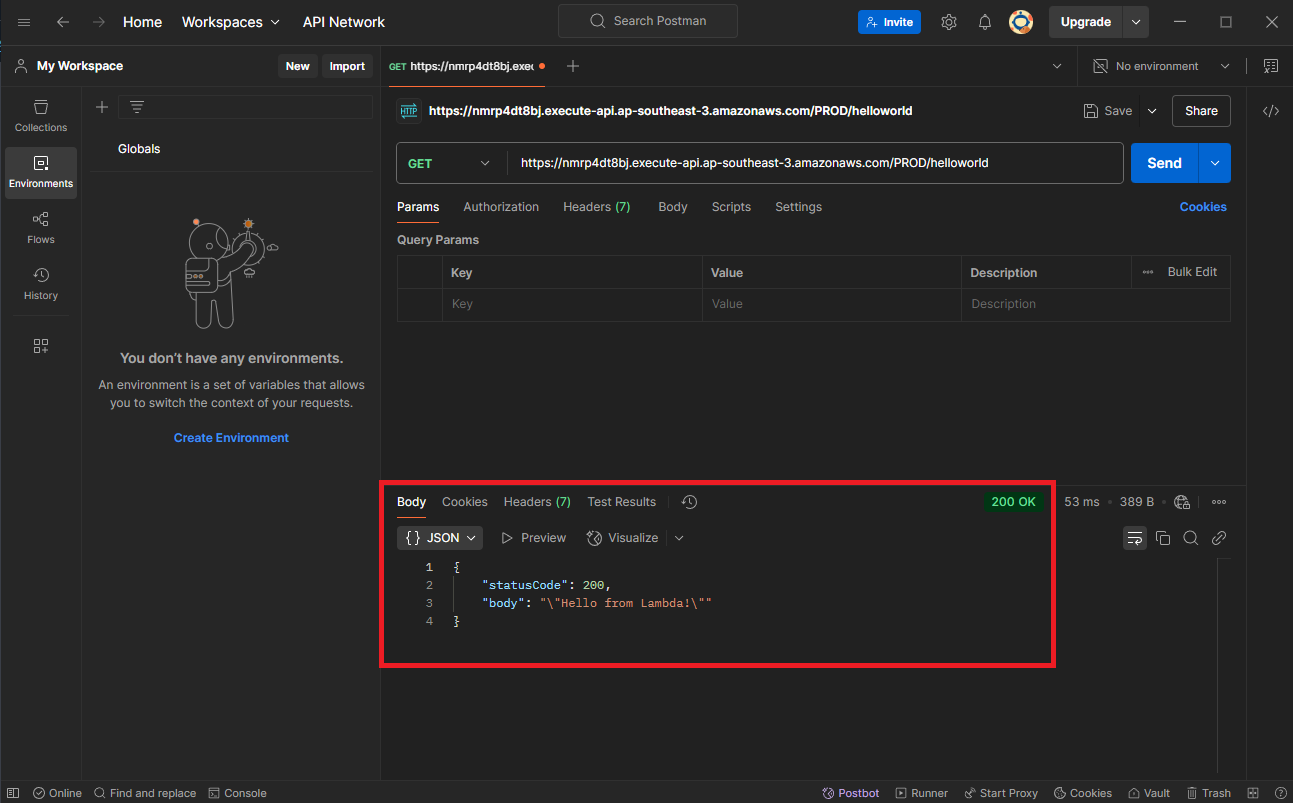
1. After we clicked the deploy button, a new page pop up showing us the stages of the method, open all the stages until you get to the GET stages, click on it to reveal the method content and then copy the “Invoke URL” for us to be able to use it on API testing tools.



1. Open up your API testing tools such as Postman API or Insomnia, then click on a new testing environment, use the GET method and then paste the Invoke URL we previously copied on the AWS API Gateway and then click on send request.

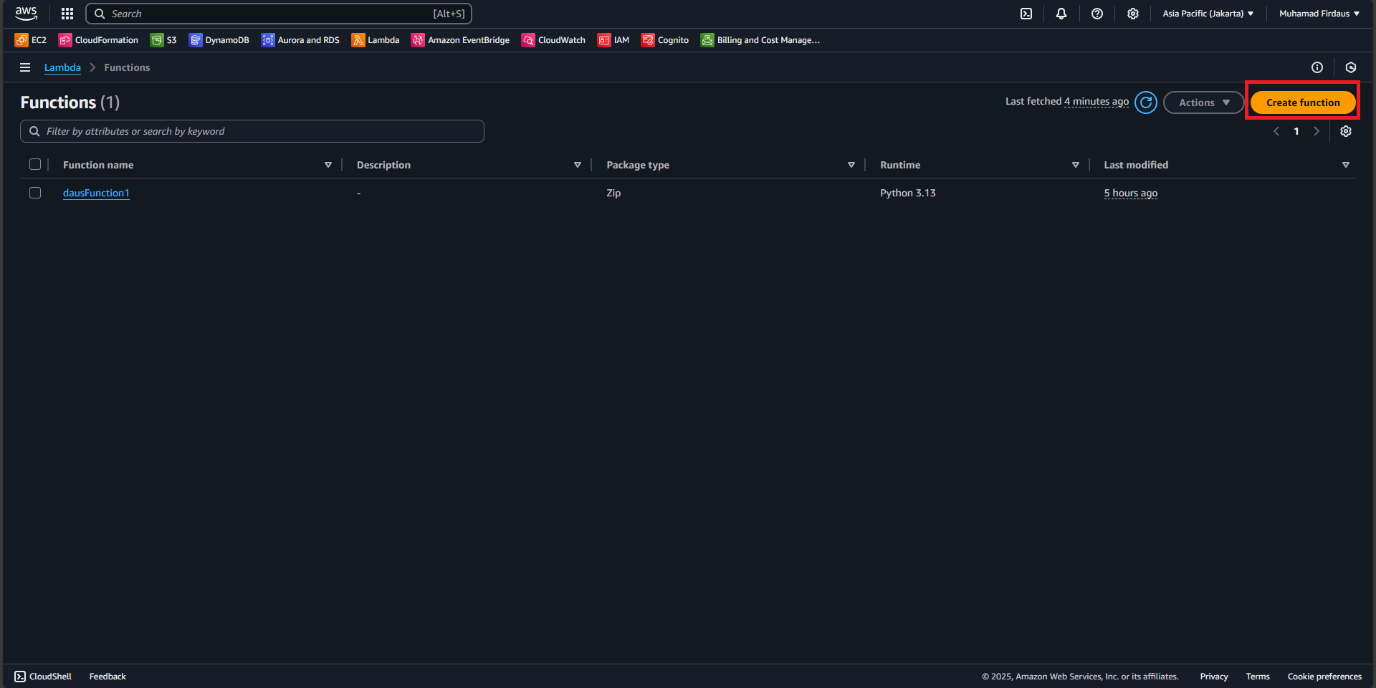


1. If the configuration is correct, we will get the response from the AWS API Gateway with status 200 OK and the Hello from lambda message.



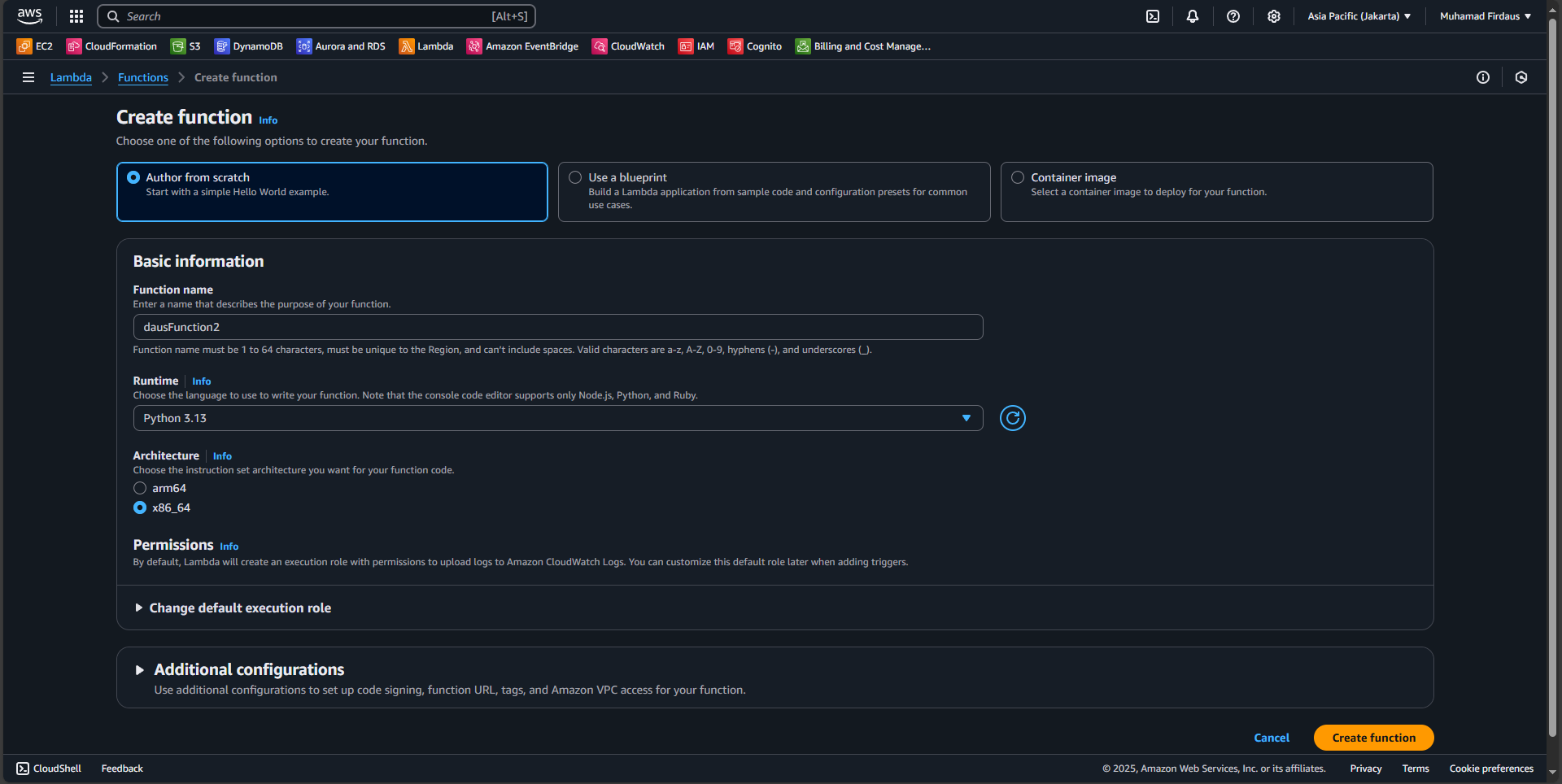
**PART 3 : Implement API Canary on the API Gateway**

1. A



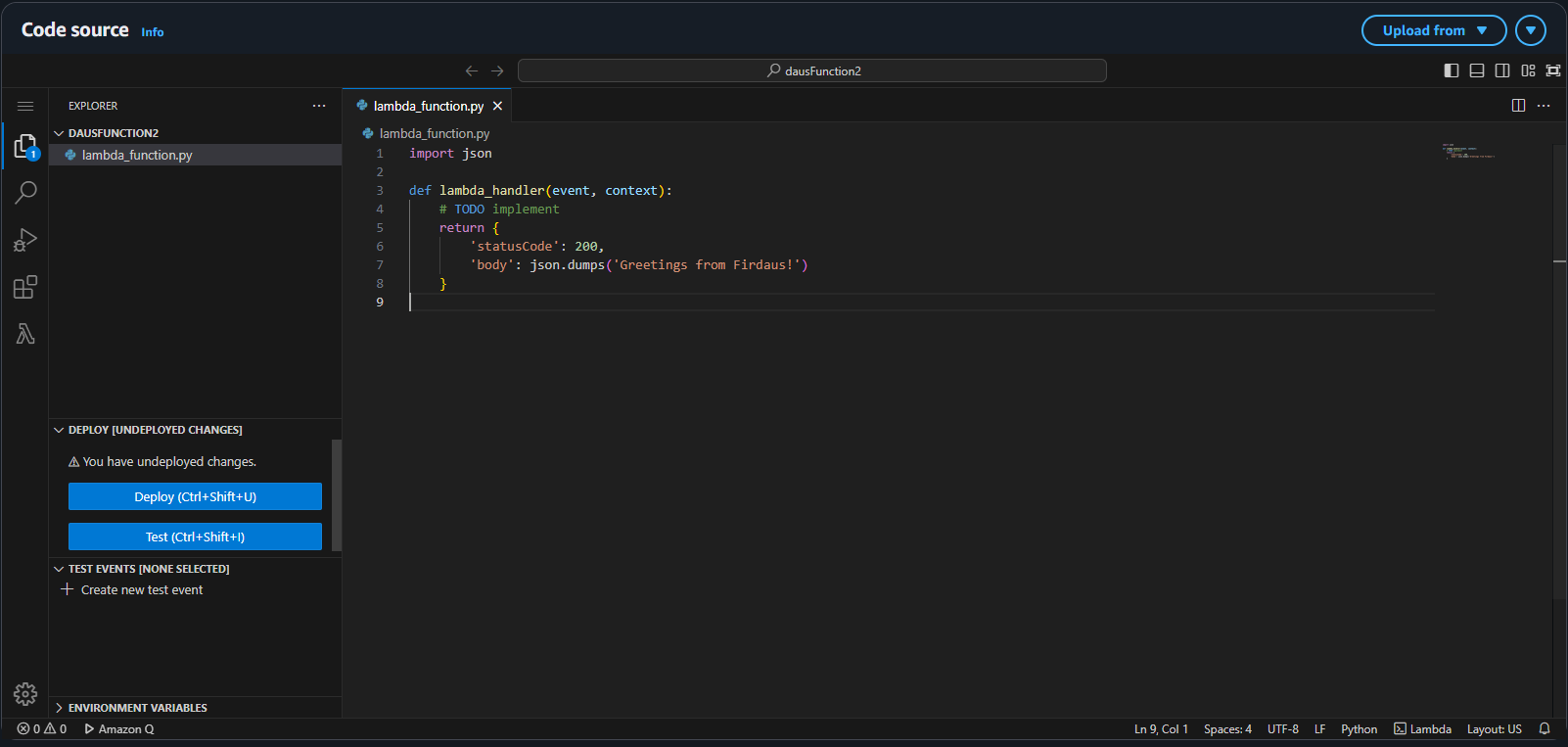
1. A

|  |
| --- |
| Create function : Author from scratch  Function name : dausFunction2  Runtime : Python  Architecture : x64 |



1. A

|  |
| --- |
| import json  def lambda\_handler(event, context):  # TODO implement  return {  'statusCode': 200,  'body': json.dumps('Greetings from Firdaus!')  } |



1. A
2. A
3. A
4. A
5. A
6. A
7. A
8. A
9. A
10. A
11. A
12. A
13. A
14. A
15. A
16. A
17. A