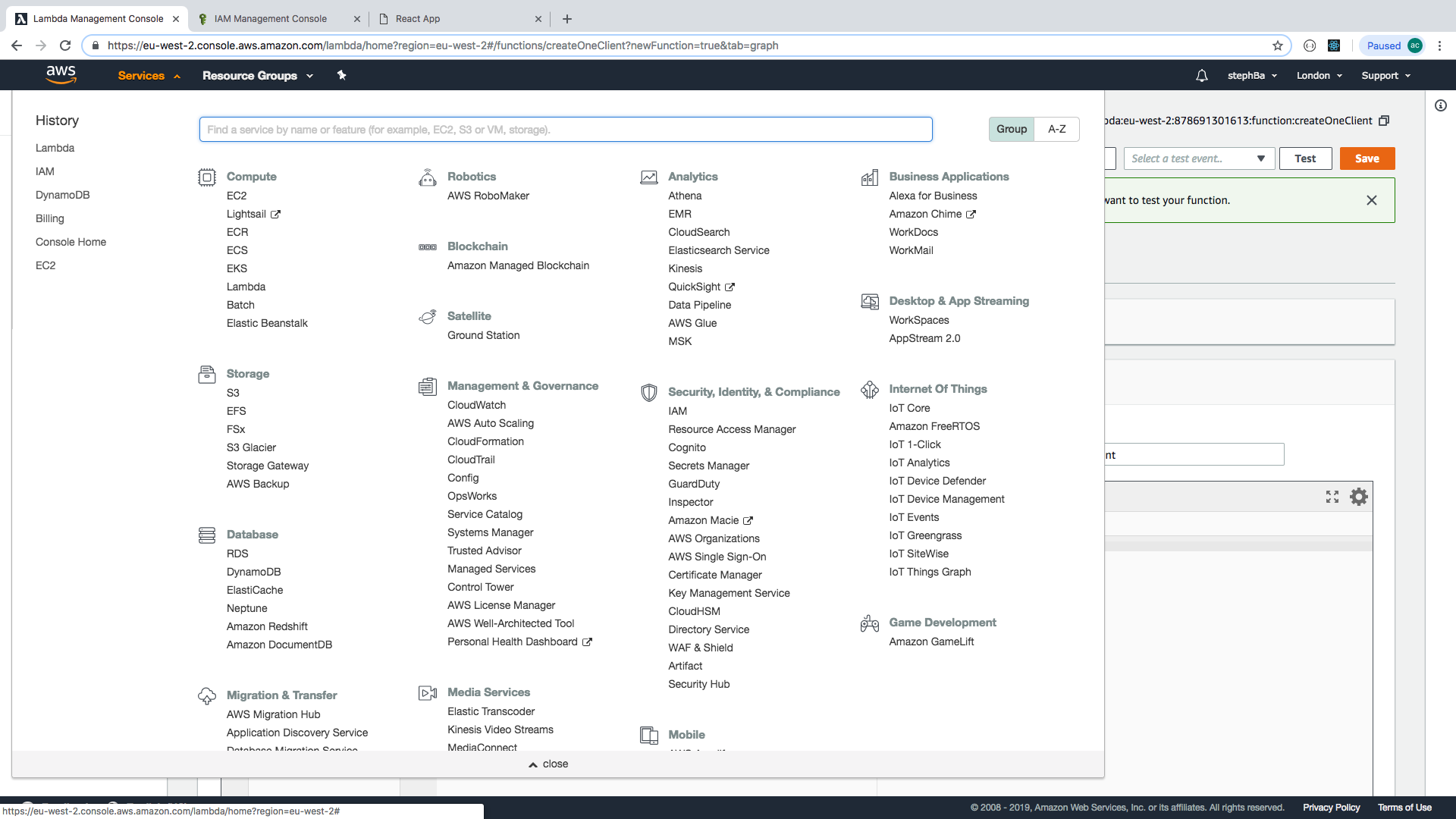
# **AWS Setup**

Sign in to AWS and select your region (Ireland or London - just remember to keep the same region at all times for all team members as Lambda functions cannot be moved between regions).

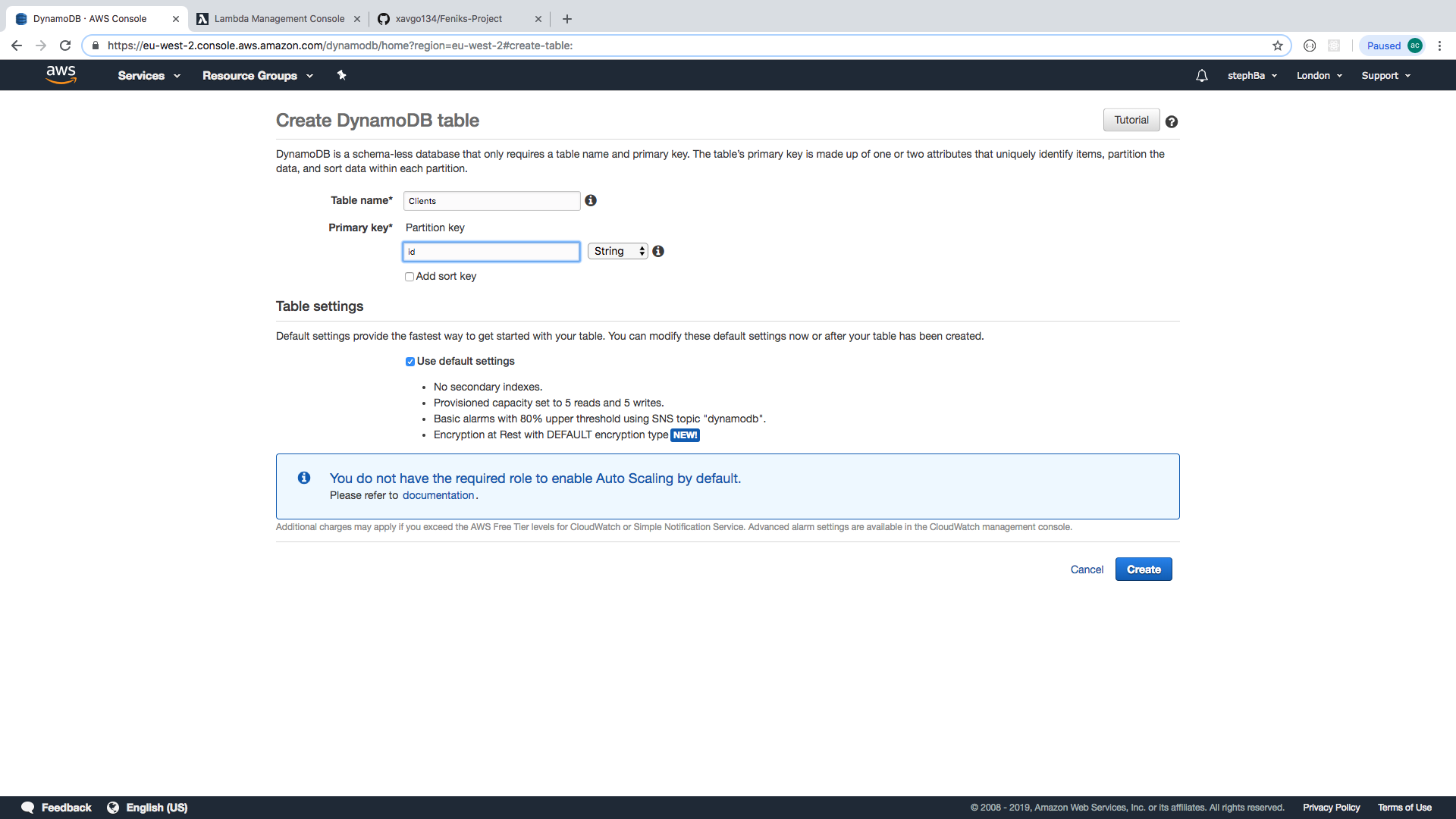


Select DynamoDB form services menu (under Database):

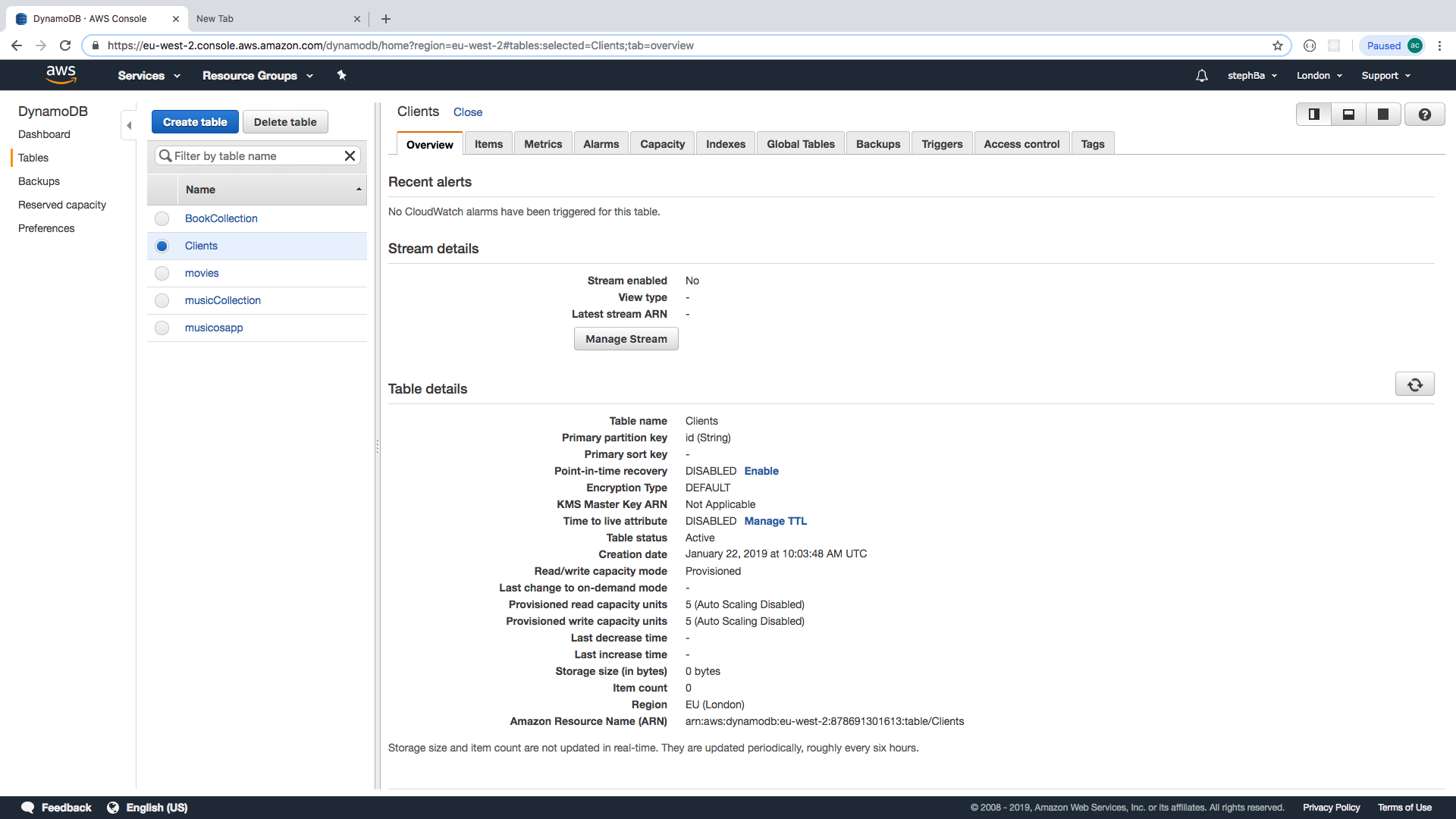


**Set Up dynamoDB**

Create Table:



For example, this creates a non-relational database called ‘Clients’ with one primary key ‘id’



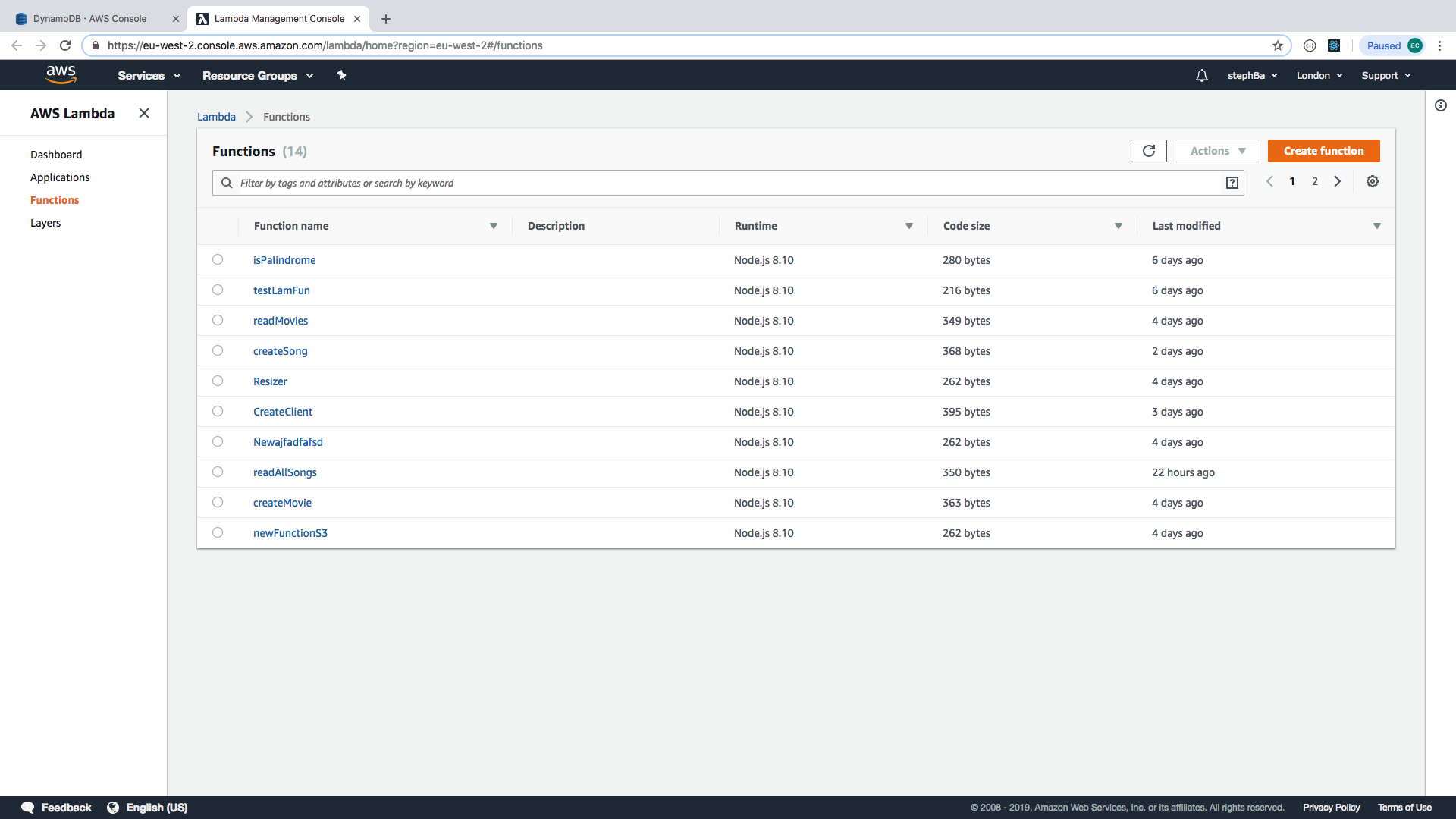
### **Create Lambda Function**

NOTE: the code you’ve pulled from GitHub contains the Lambda functions, but they aren’t used in the code, they are only used on AWS. They are saved in the code just so you have a copy of them!

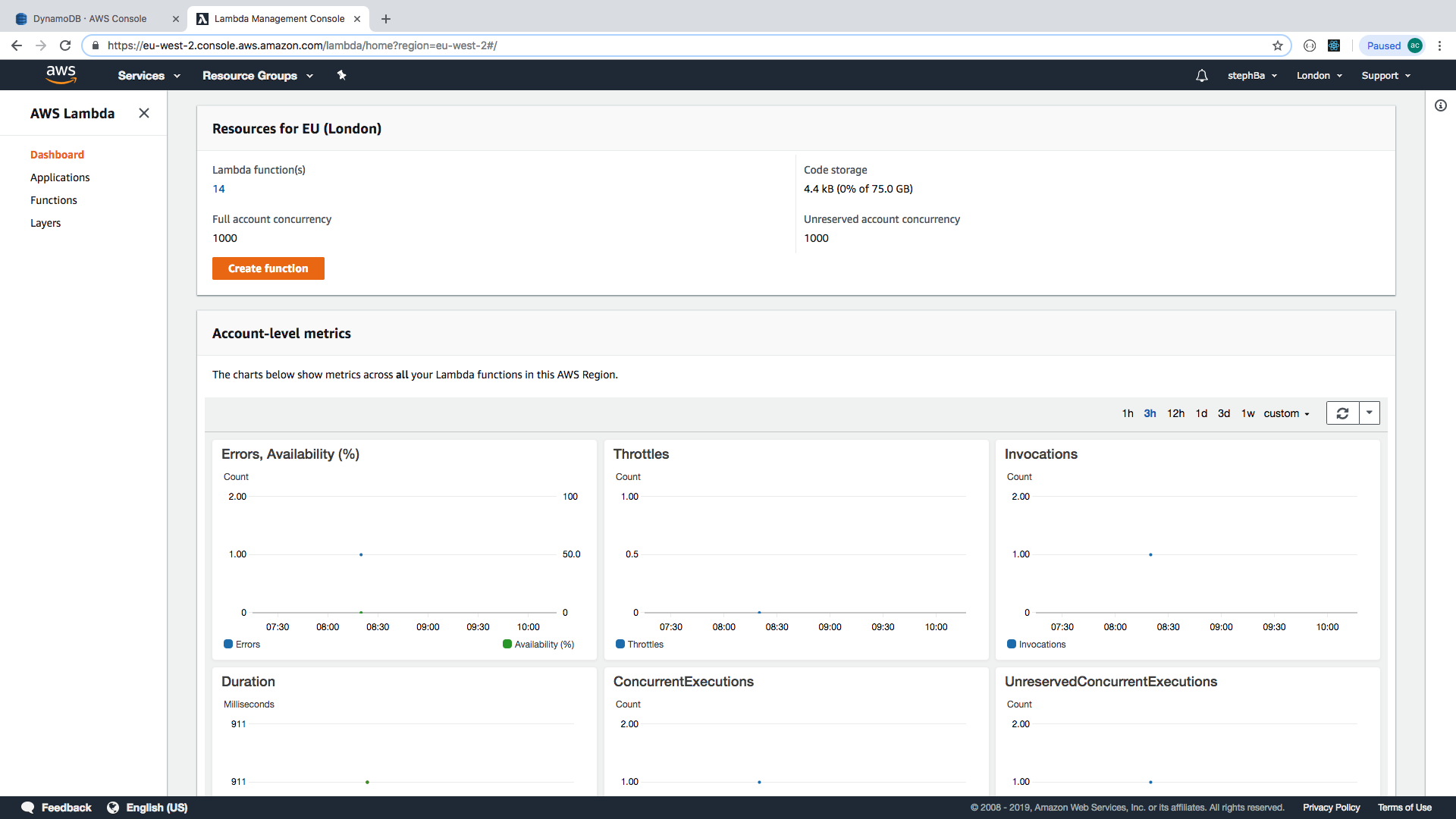
Go here for some example JS for lambdas using dynamoDB if you get stuck:

<https://github.com/awsdocs/aws-doc-sdk-examples/blob/master/javascript/example_code/dynamodb/ddb_batchgetitem.js>

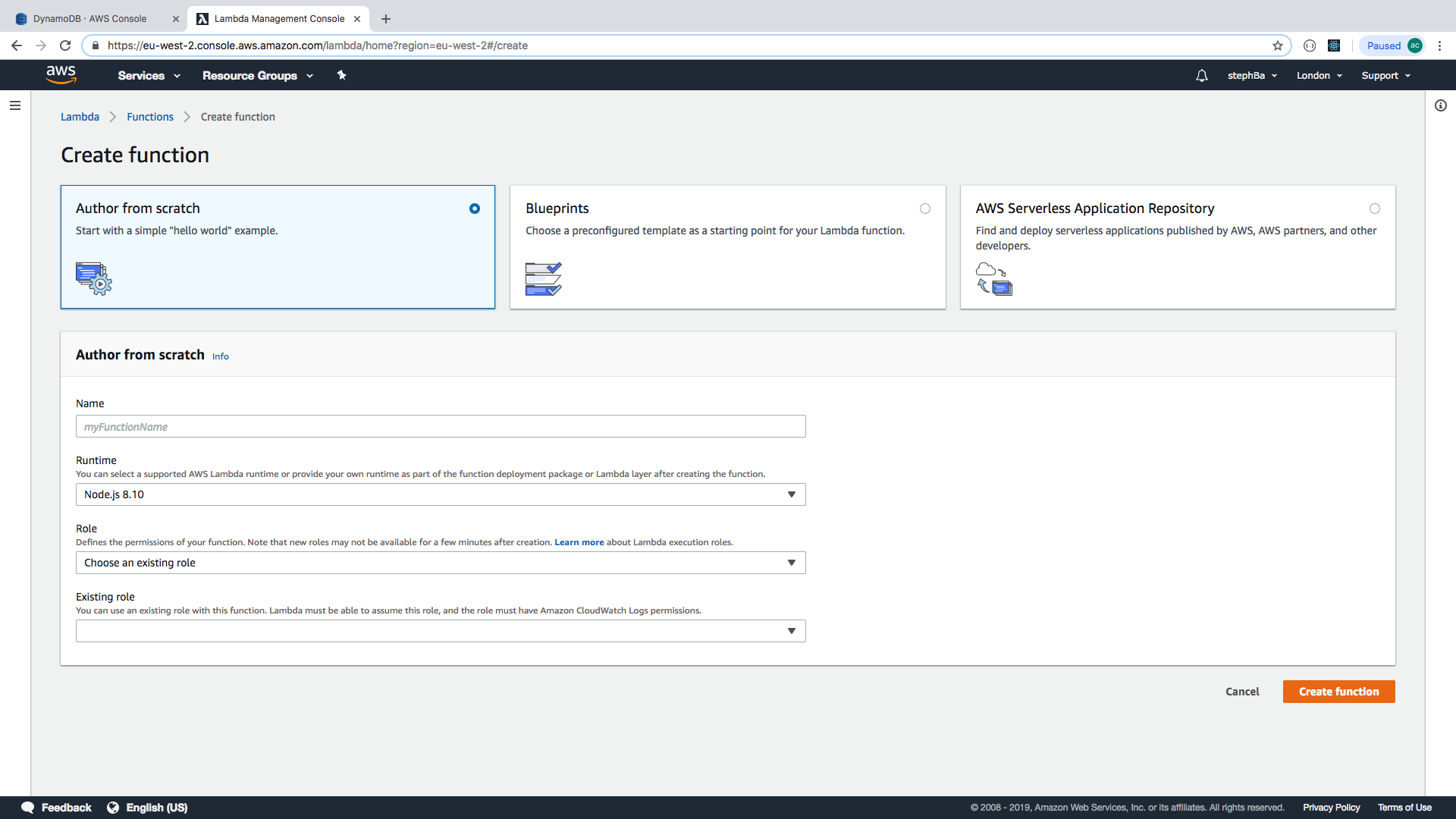
Select Lambda from the services menu (under “Compute”). Can be created from ‘functions’ or ‘dashboard’:



Select ‘create function:

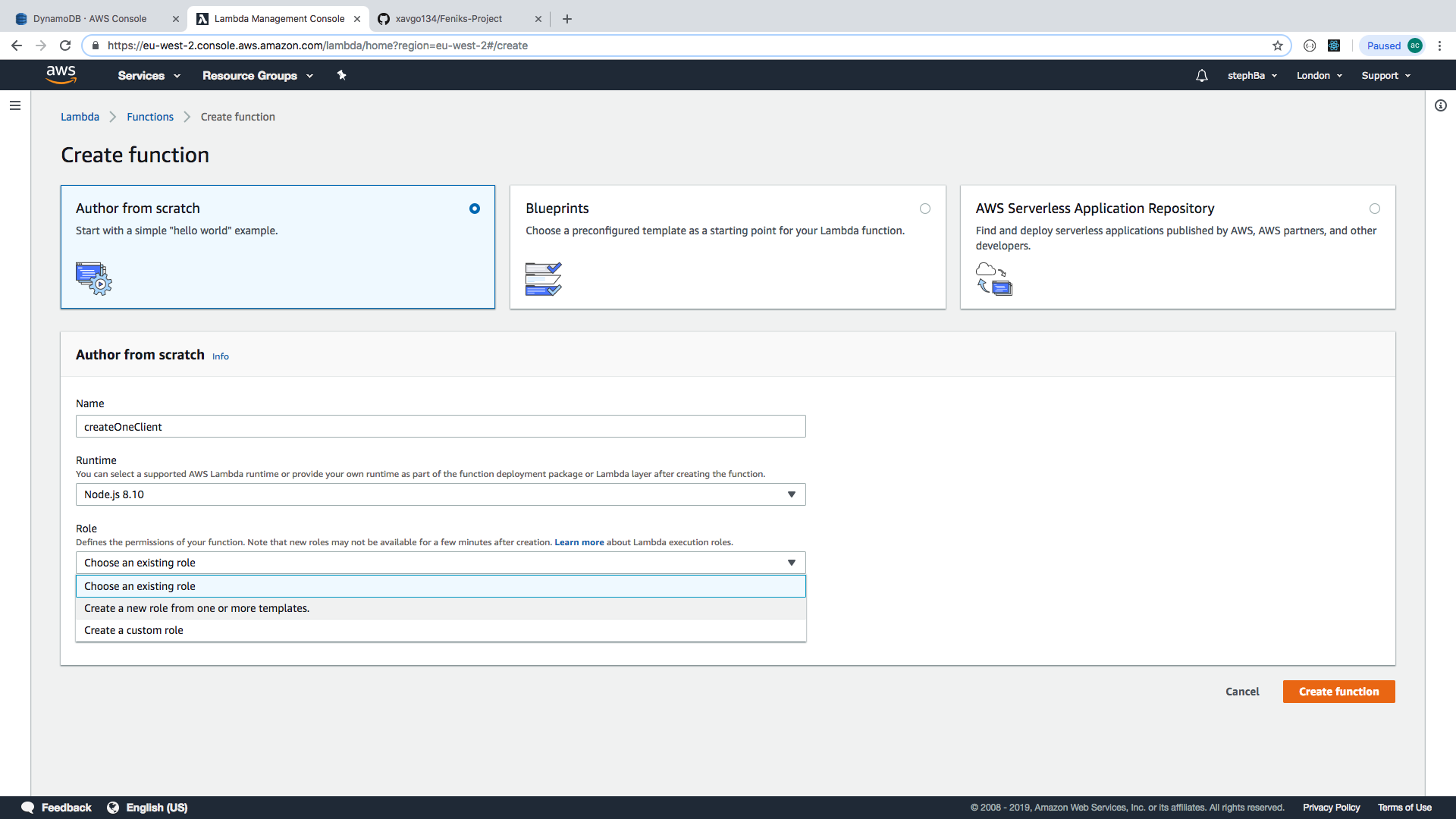


Then complete the details of the function using “Author from scratch”:



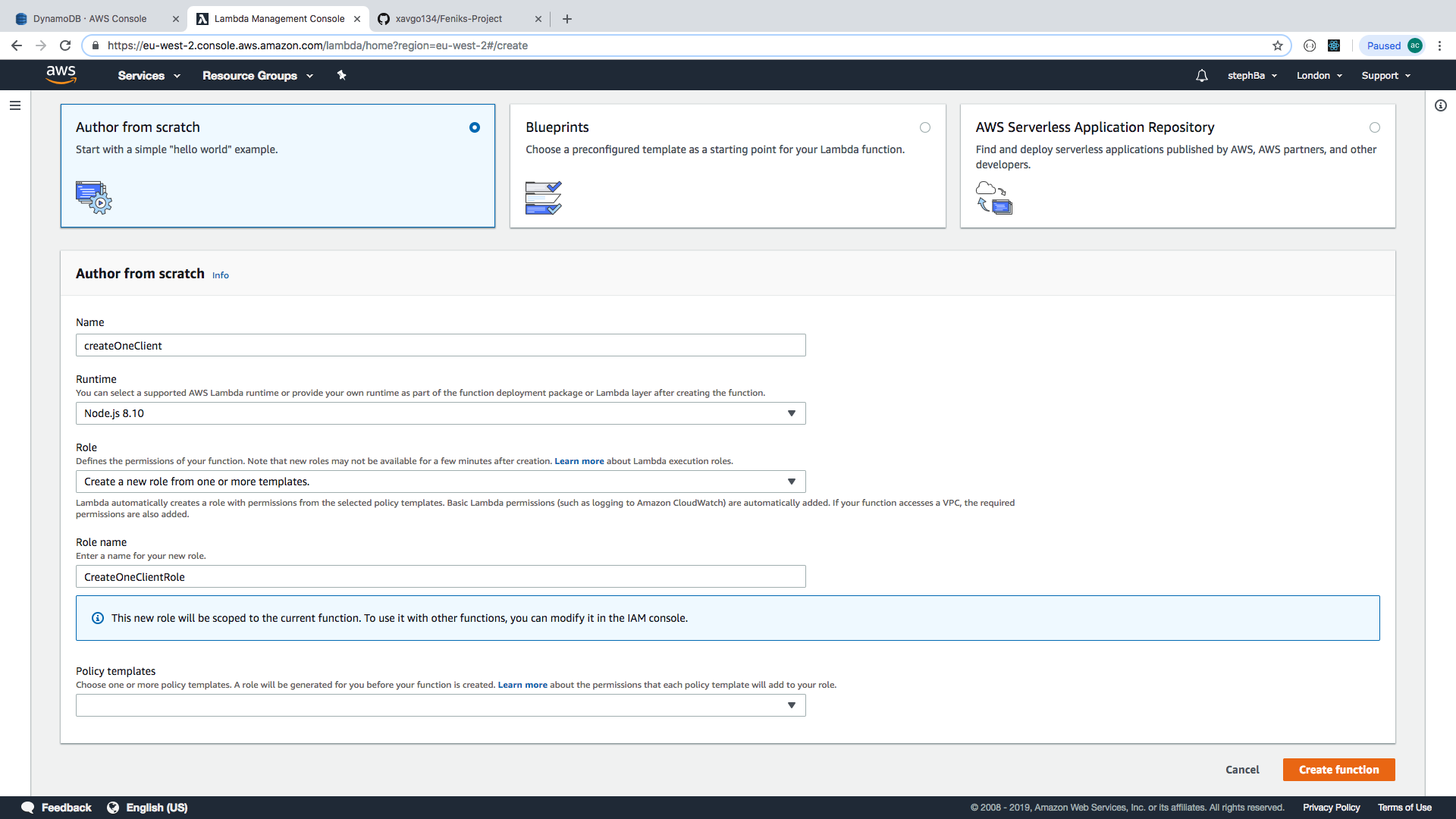
We are going to create the function to create one client.

Select the runtime, it defaults to node.js 8.10, which is what we want.

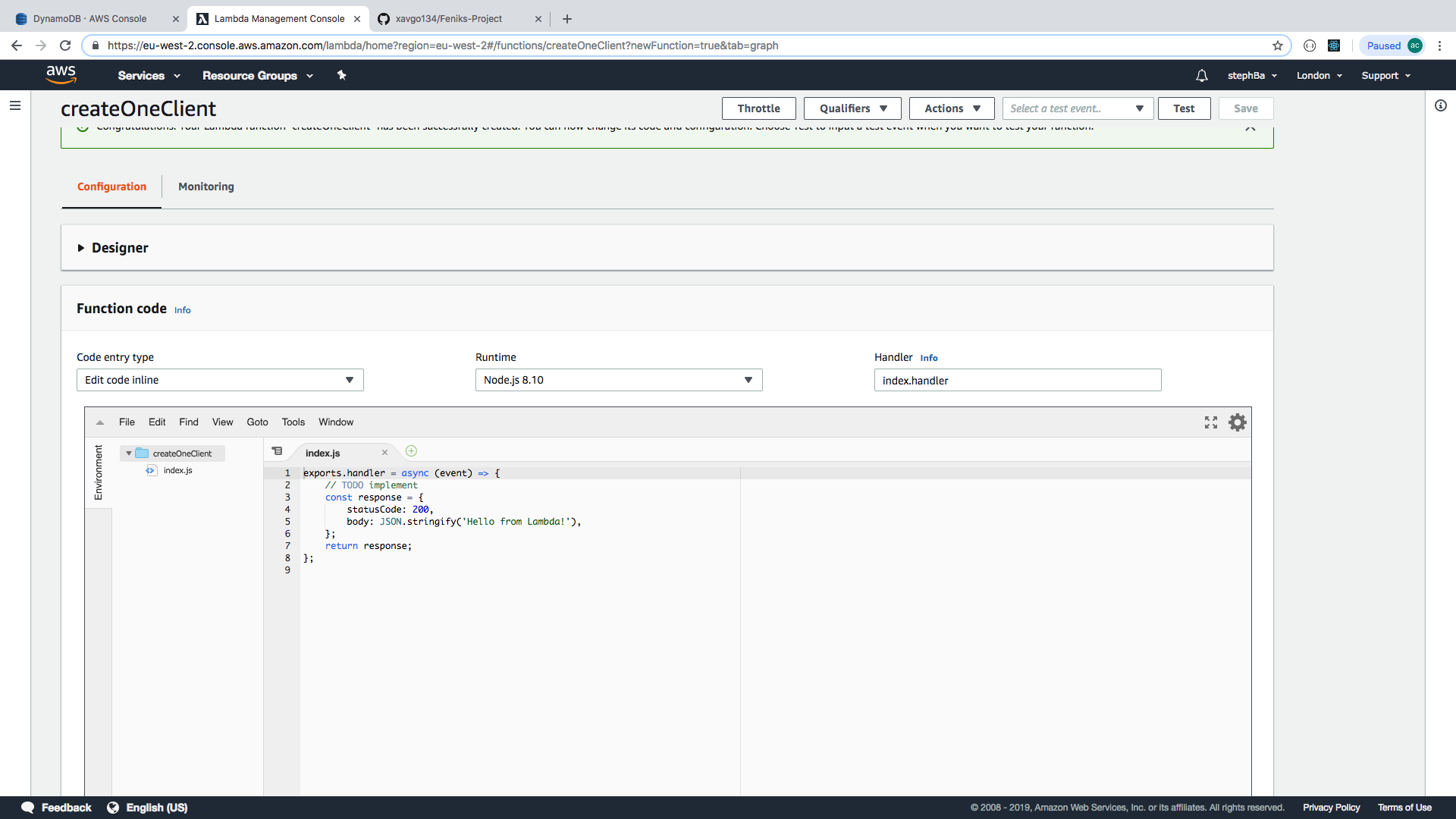


Select create a new role from one or more templates.

We have called it CreateOneClientRole:

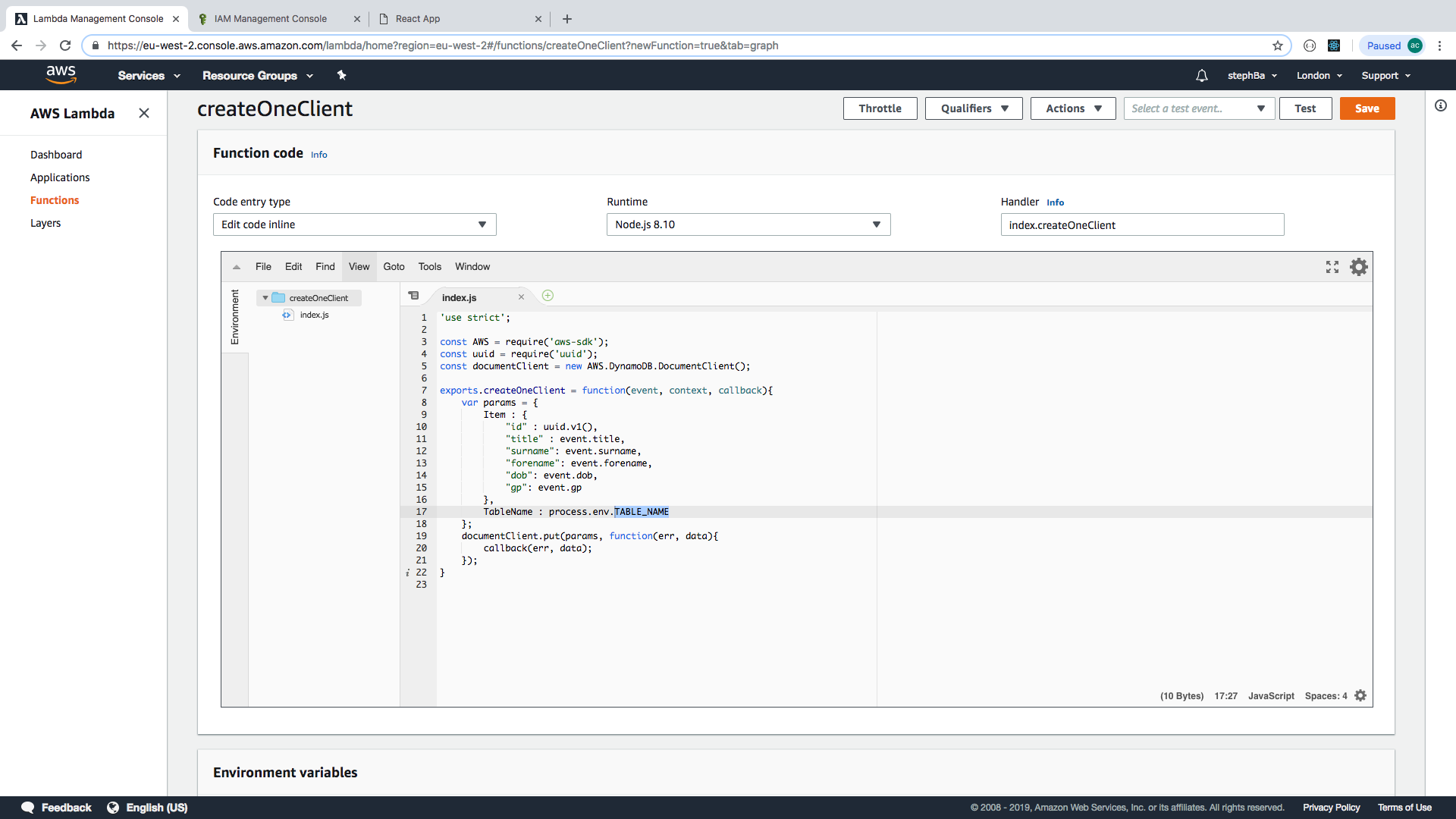


Hit create function. You get a template of “exports.handler” as a start point for writing your own function.

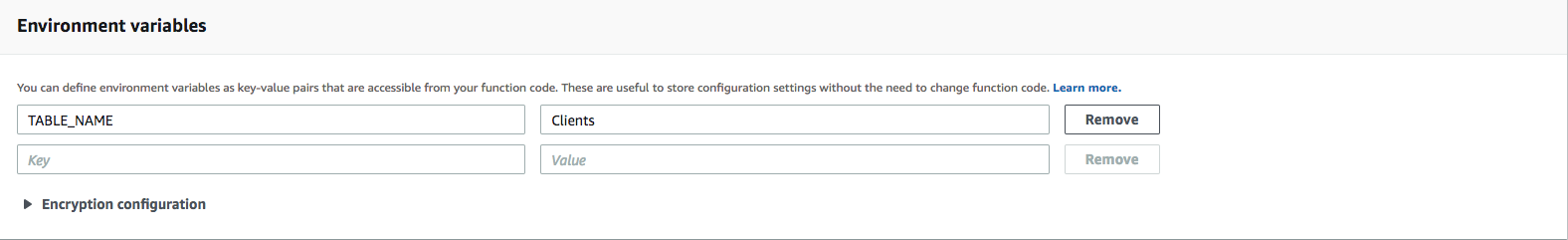


Below we have entered the code for our createOneClient function. Note that this requires “aws-sdk” (aws software development kit), uuid (universally unique identifier) and dynamoDB.

Important! The Handler suffix when first created defaults to index.handler. This must be changed match the export suffix, so below we have “**export.createOneClient**” and “**index.createOneClient**”.

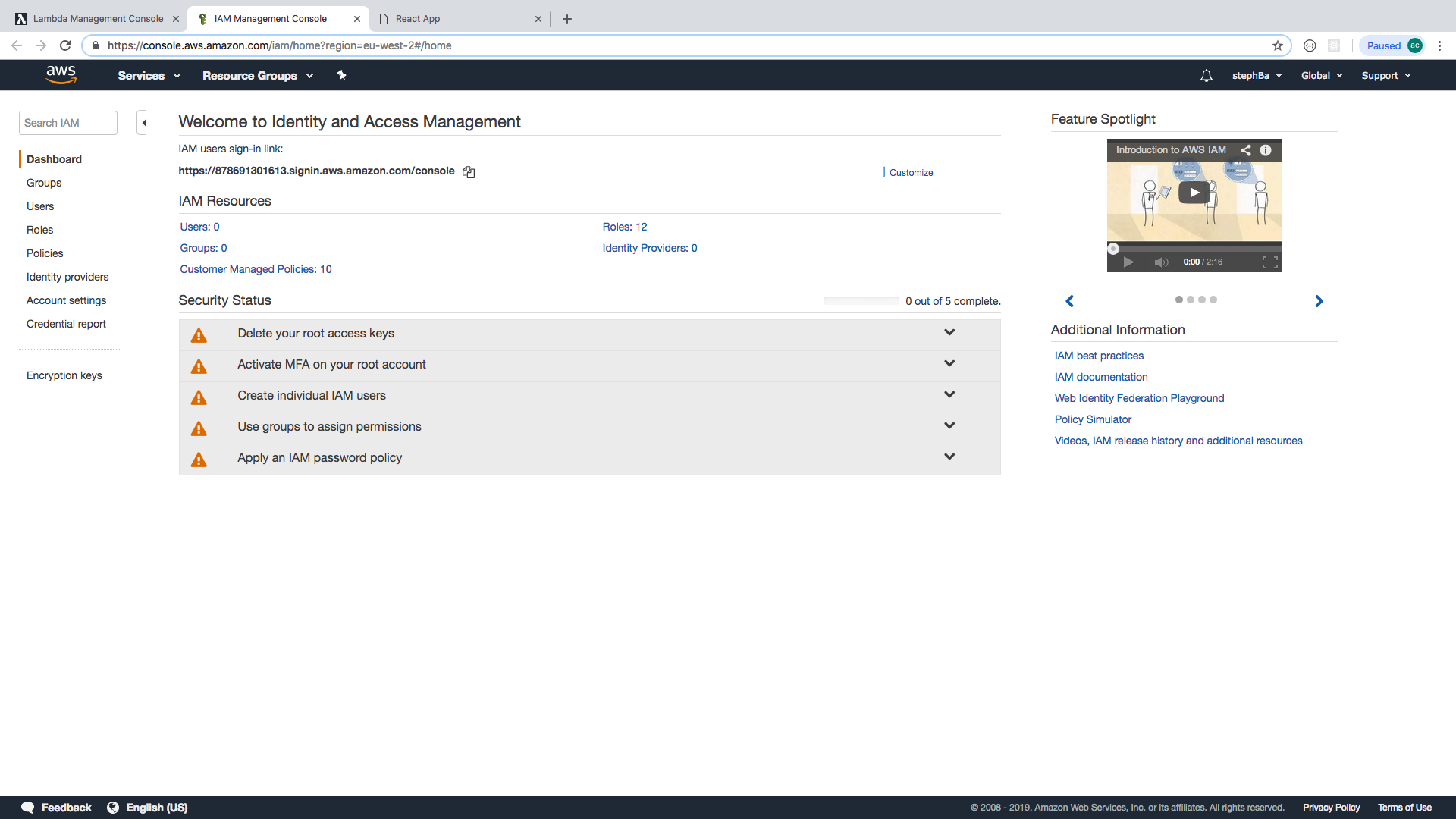


Now set up the environment variable for TABLE\_NAME to be the name of the table (Clients)



### **Set up IAM (Identity Access Management)**

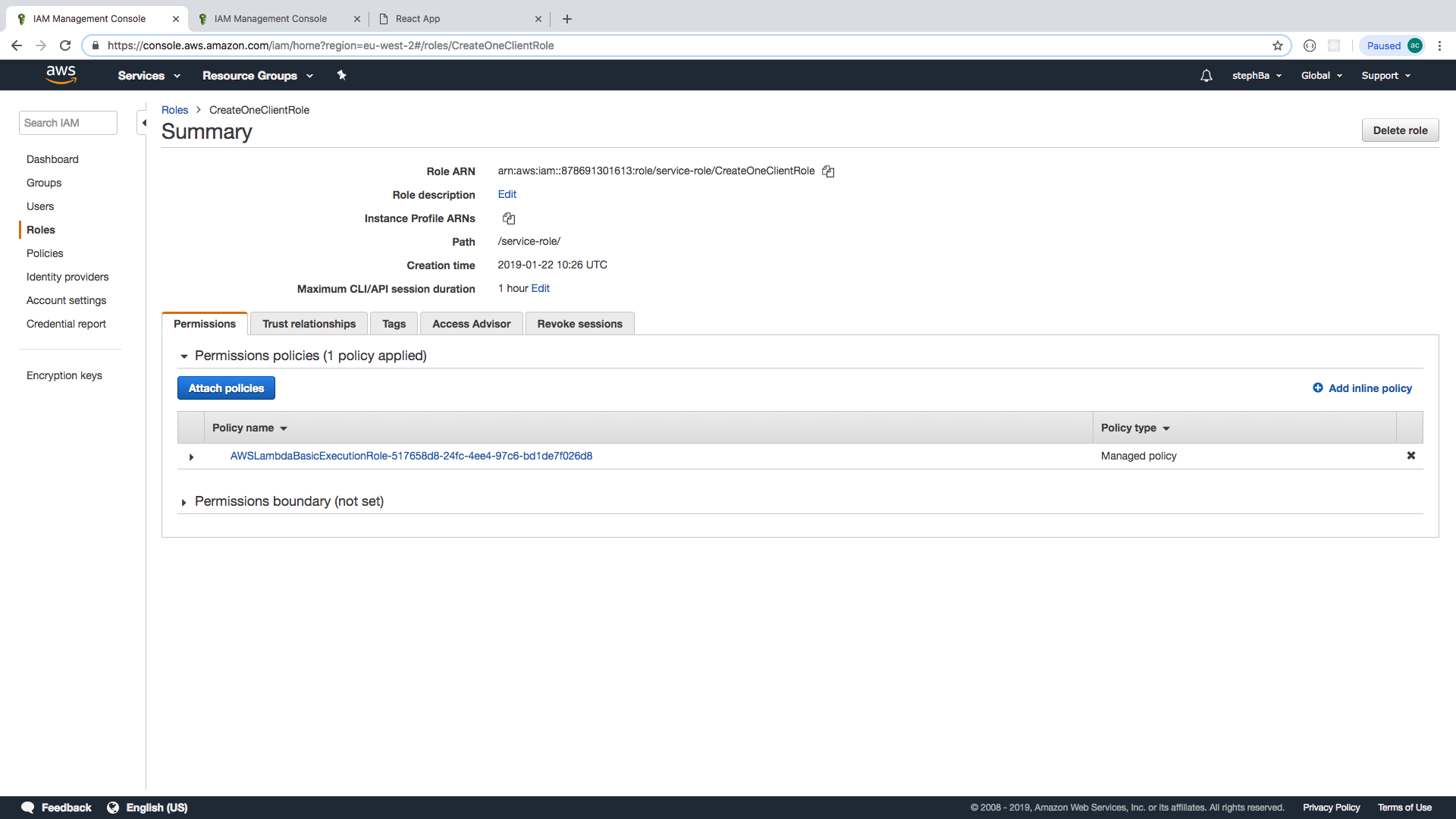
Select IAM from services menu (under Security):



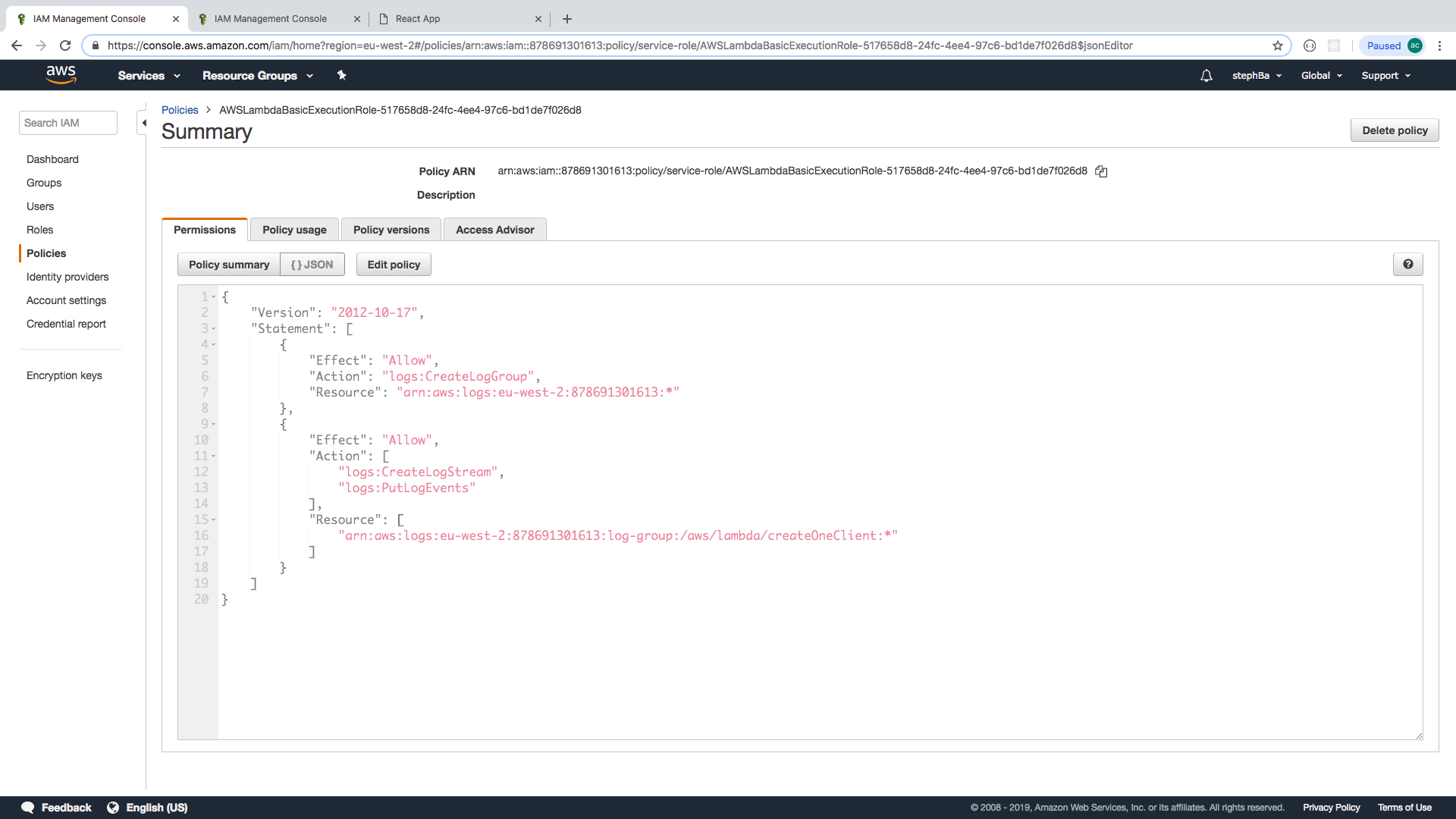
Select roles



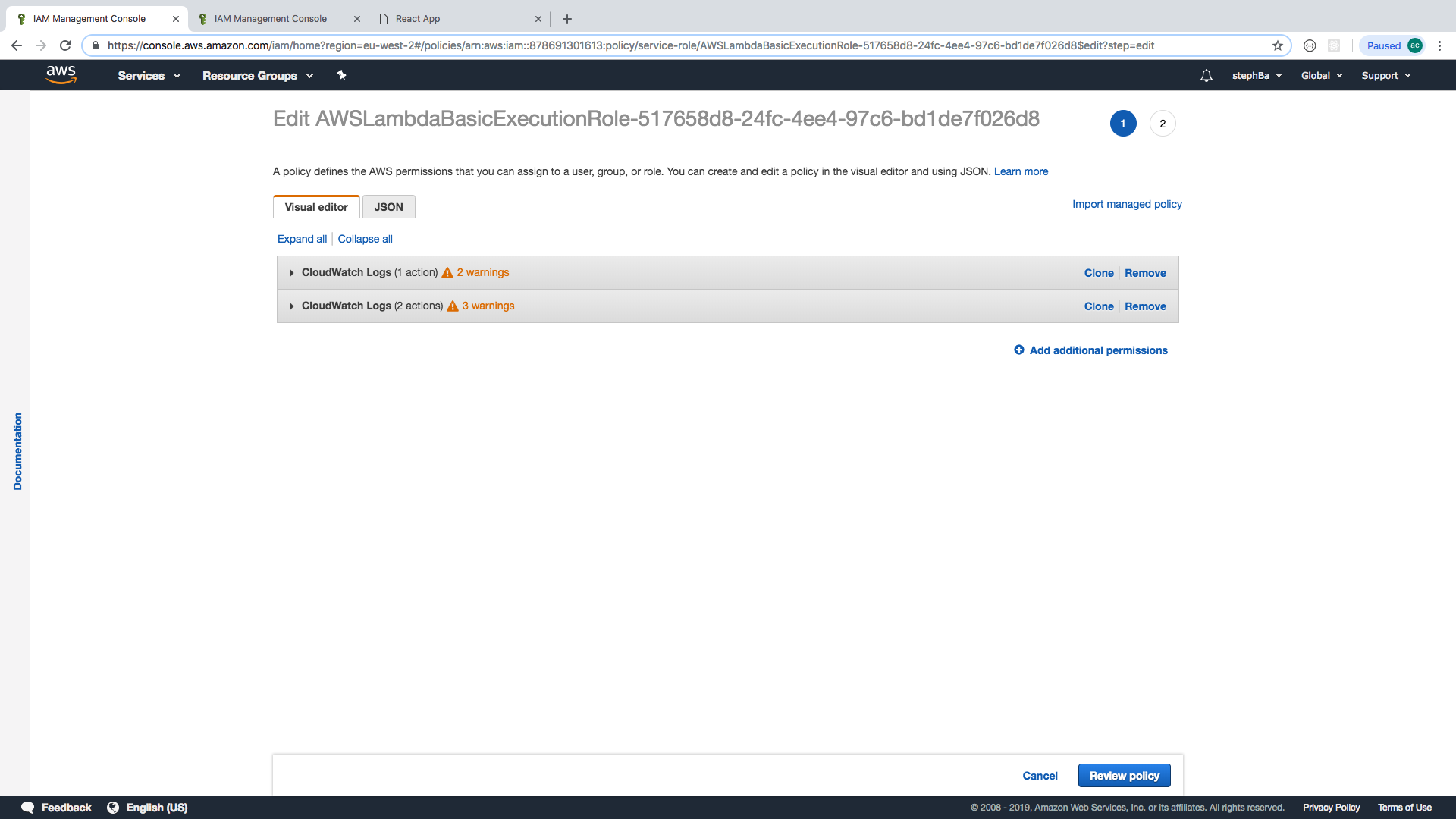
Click on the role you created for the function:



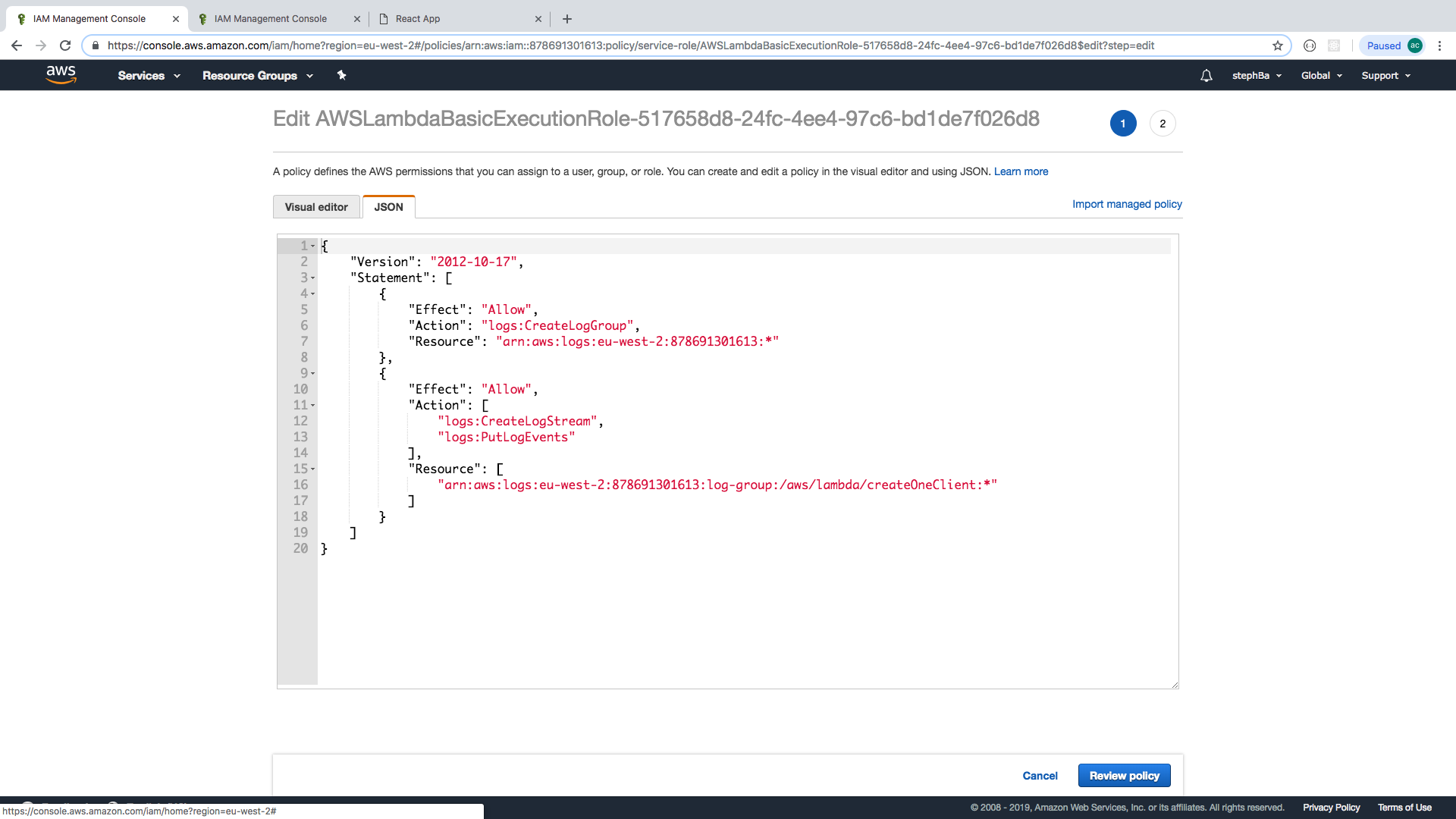
Select the policy name:



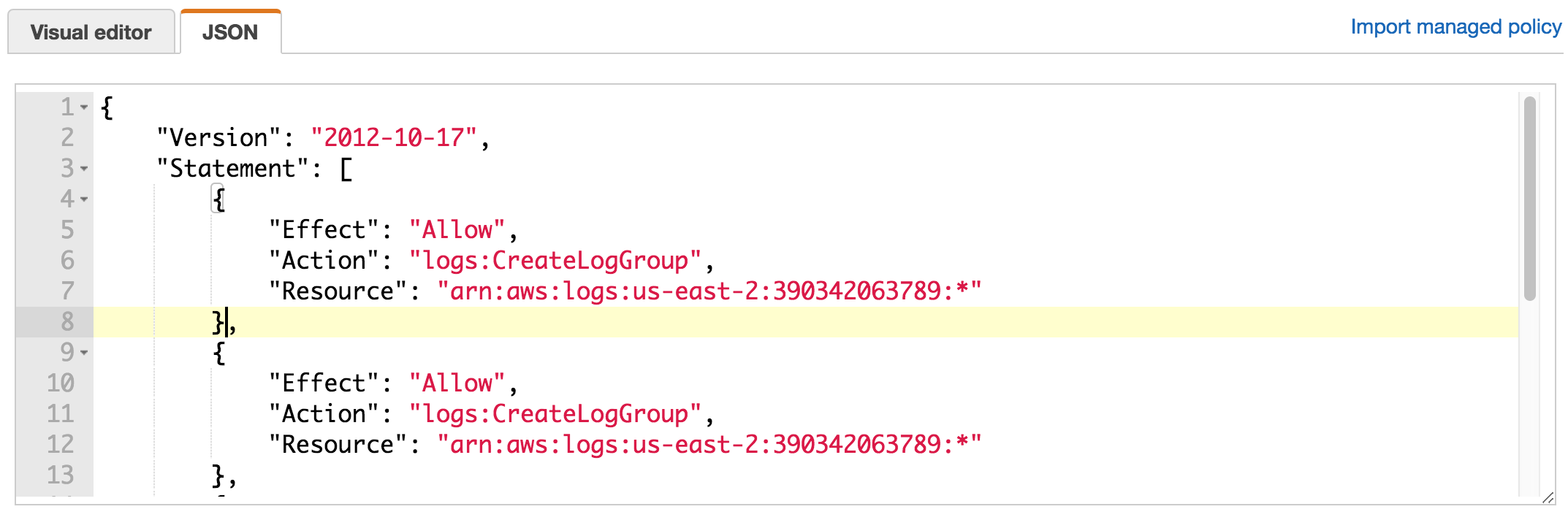
Select {}JSON to see the policy in a JSON format then click on ‘edit’ policy:



Then click on JSON to edit the code:

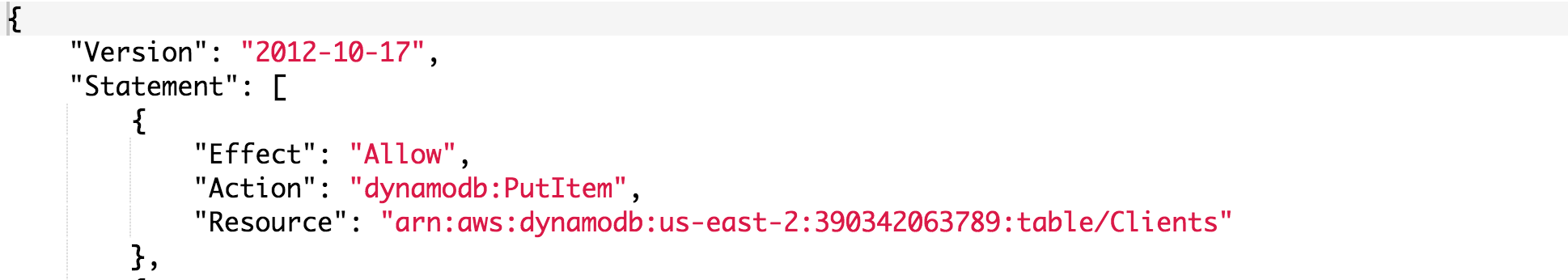


Now you need to edit the permissions. Copy the object that includes “Action”:”logs:CreateLogGroup” and paste the copy above that object, as shown:



In this object, change the “Action” to reference dynamoDB and the method we want to use (in this example it’s a PutItem, but for info on other methods check the [docs](https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/Welcome.html)).

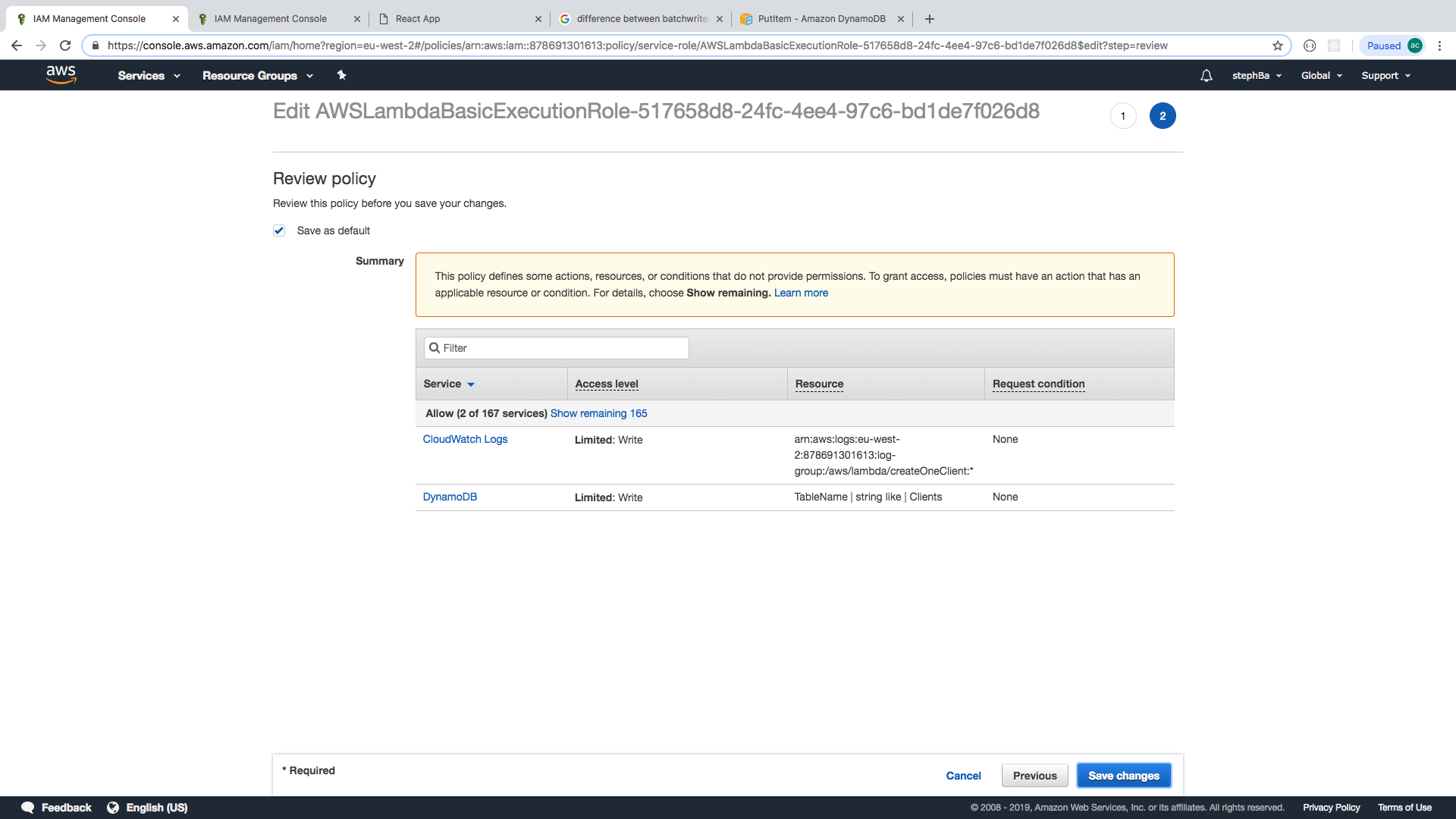
Change the “Resource” to reference dynamoDB, and remember to swap the asterisk at the end with your table and route (in this example it’s “Clients”).



This adds:

1. Permission for the function to PutItem
2. Defines the resource based on the Amazon Resource Names.

Click on review policy then save changes:



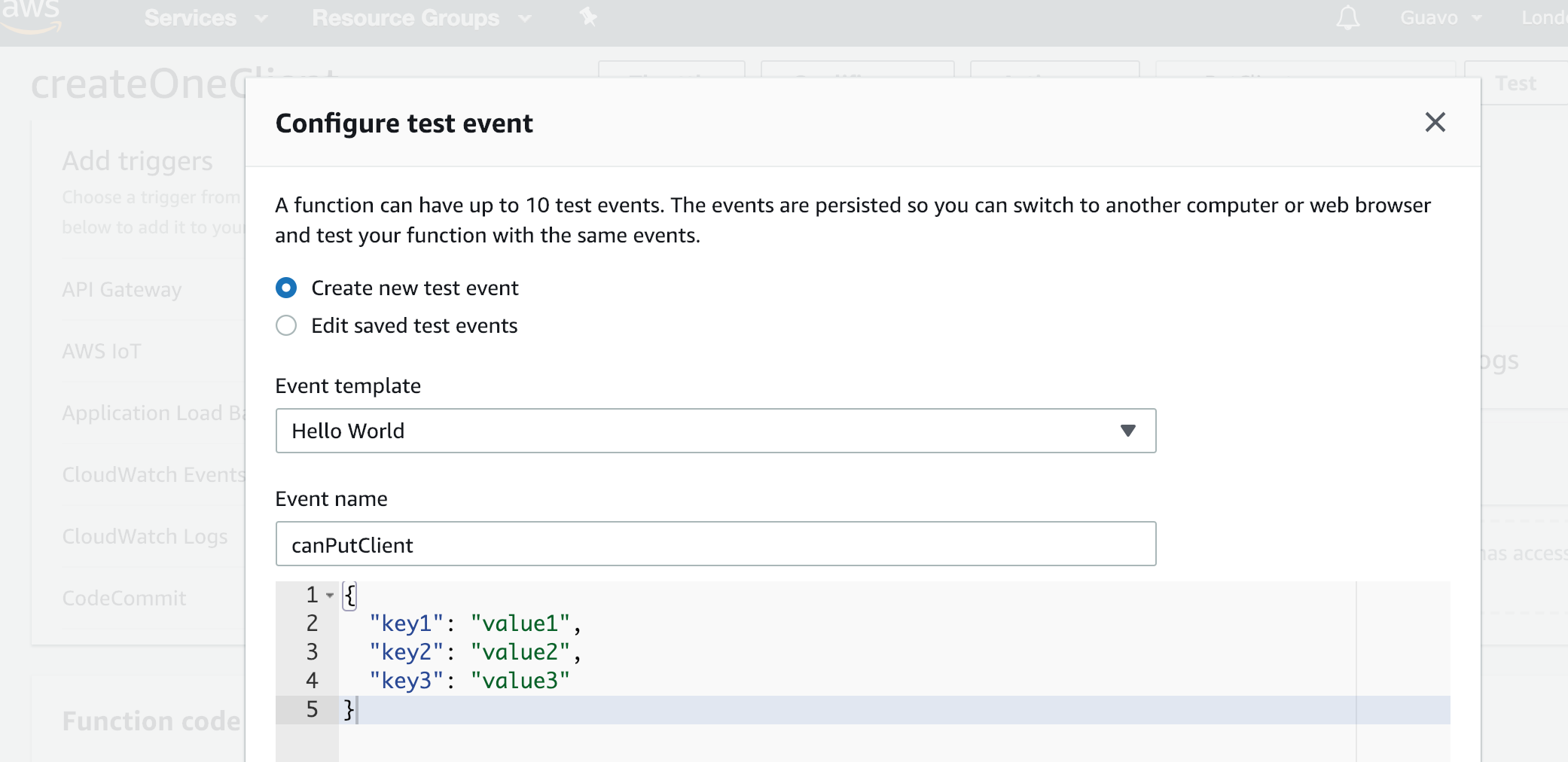
Go back to Lambda using the services menu. Select your function.



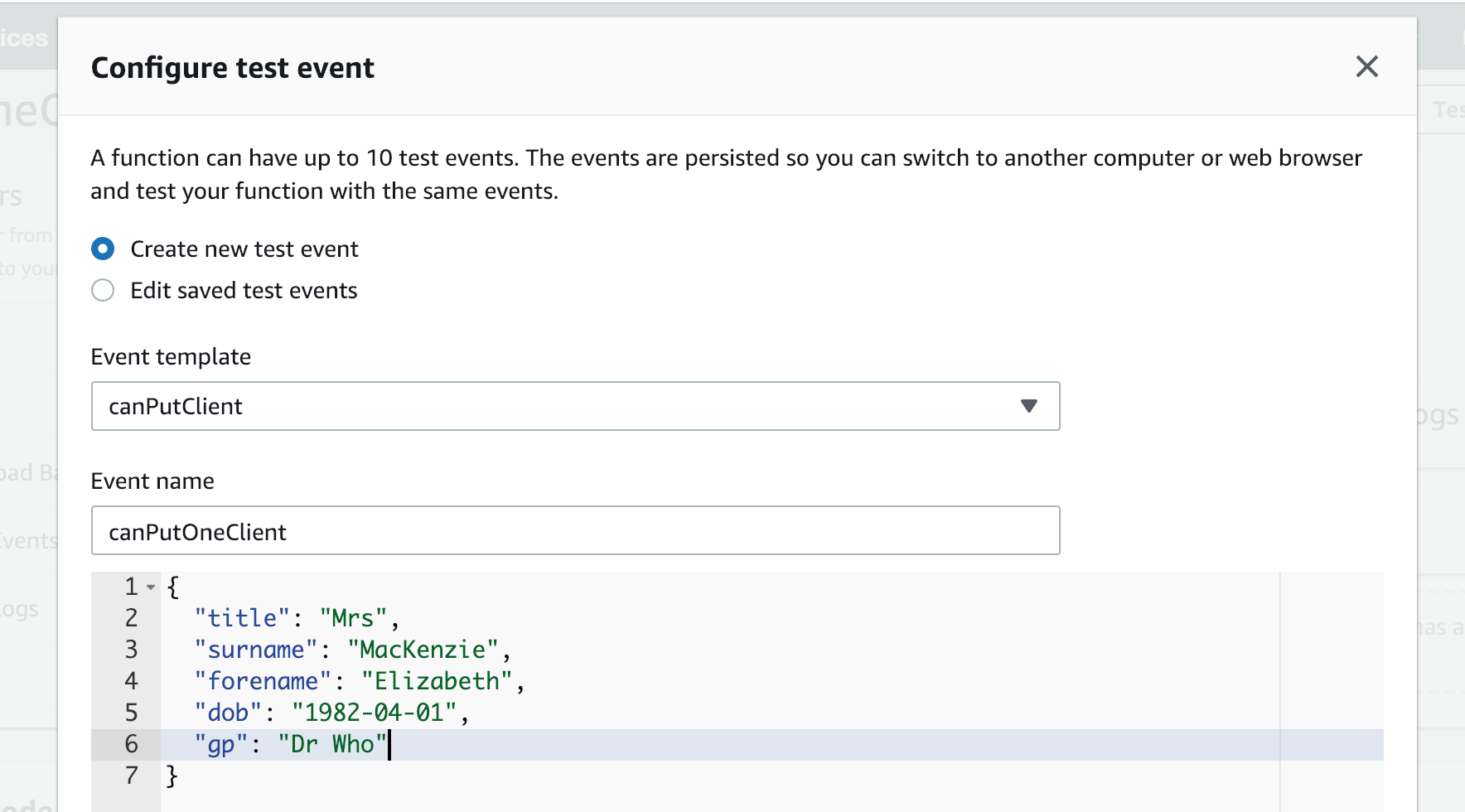
Select ‘configure test events’ from the ‘select a test event’ dropdown.

Note - For example, in the app, the code refers to “title” : event.target.title.value, but for Lambda to access this information, the format is just:

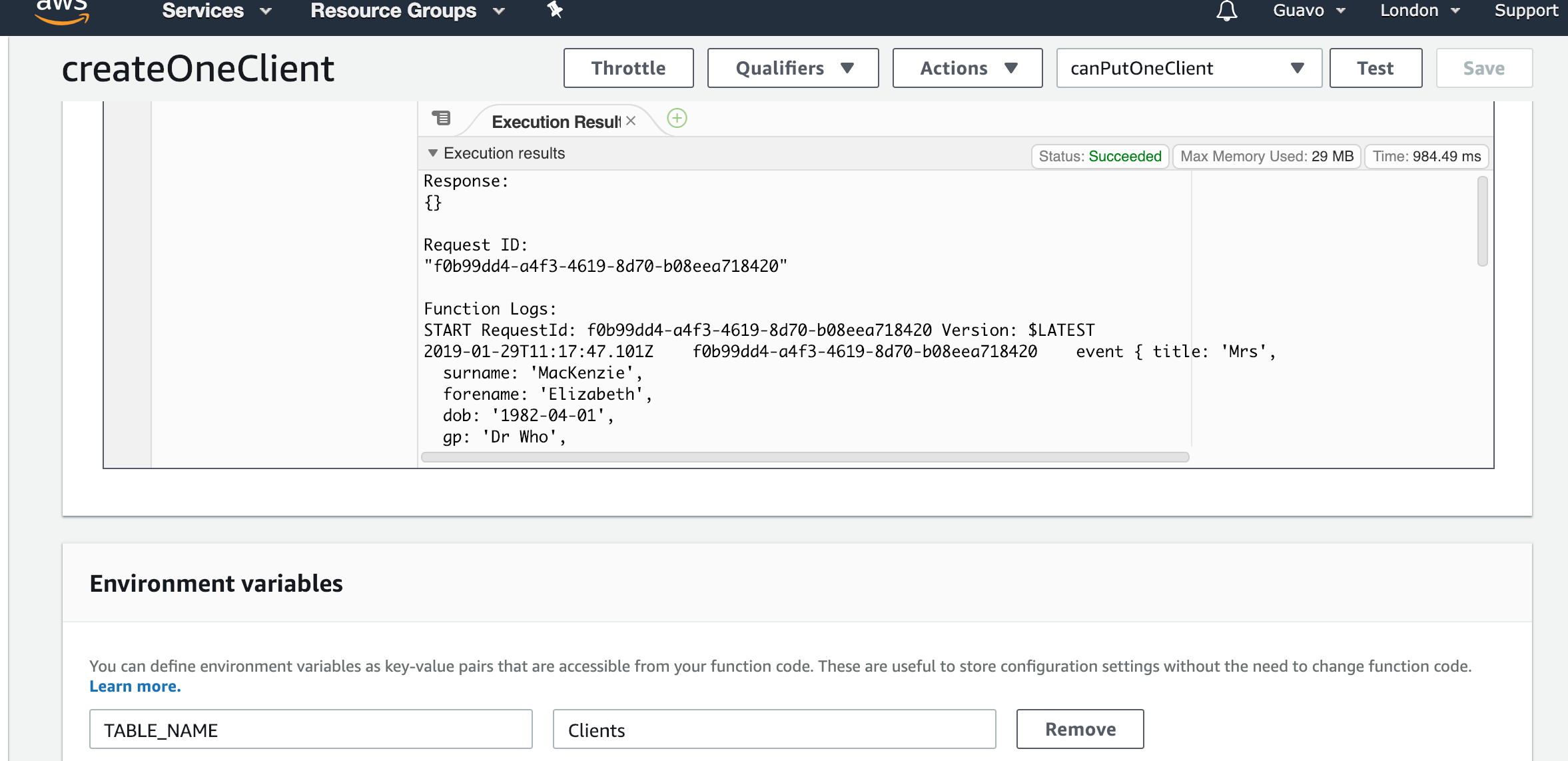
“Title” : event.title,



Replace the template with your own test:



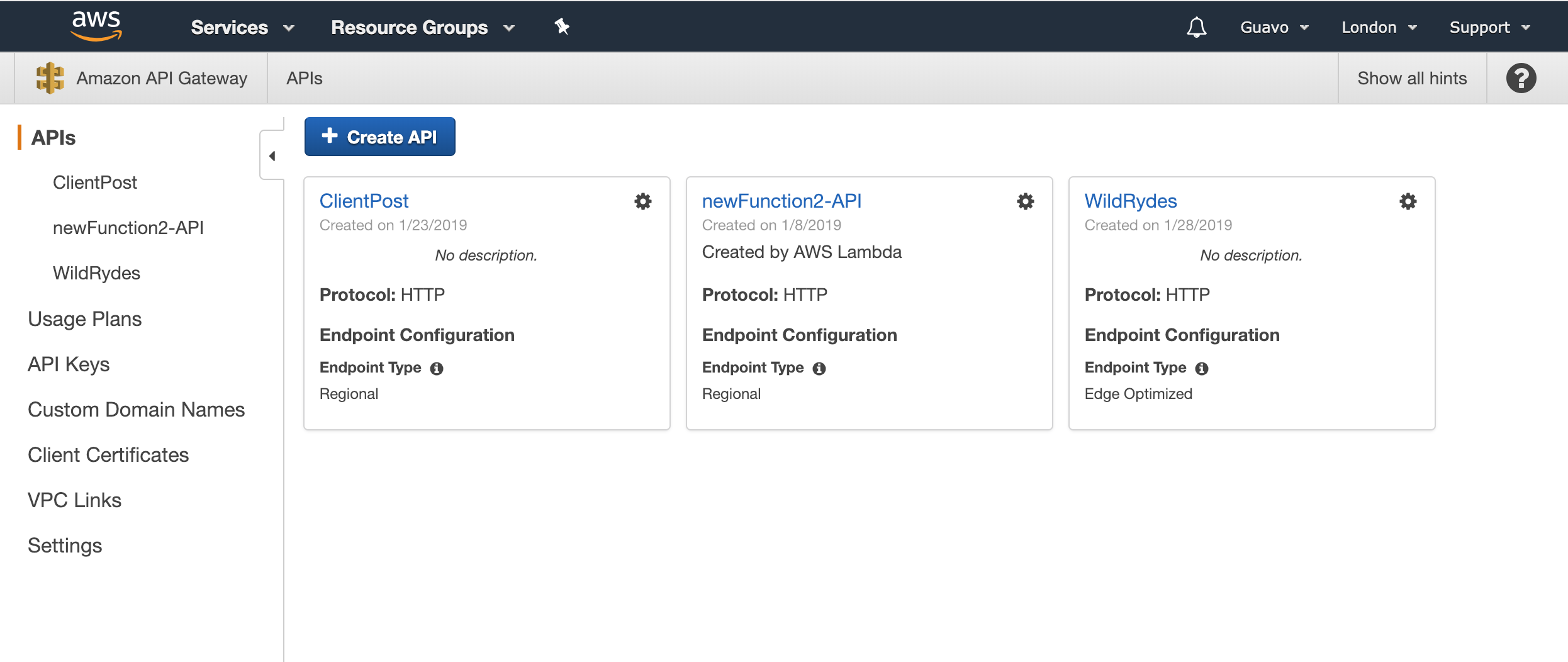
Then run the test, which will give the following if successful:



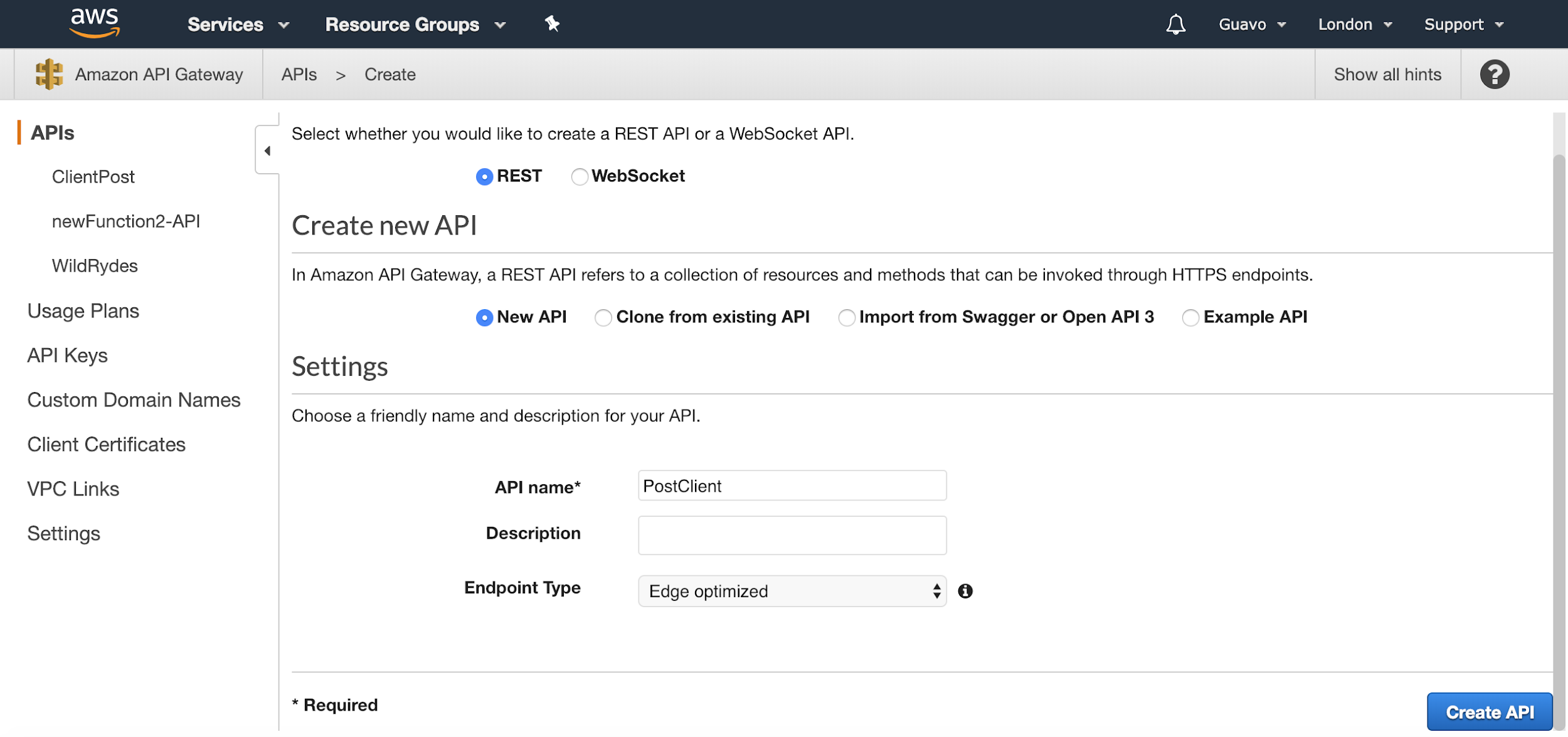
If the test is unsuccessful, check the resulting error message for more info. It may be that the test itself is wrong, or it could be that the role you’ve just created isn’t correct.

### **API Gateway**

Next up is the API Gateway. Go to services and select API Gateway:

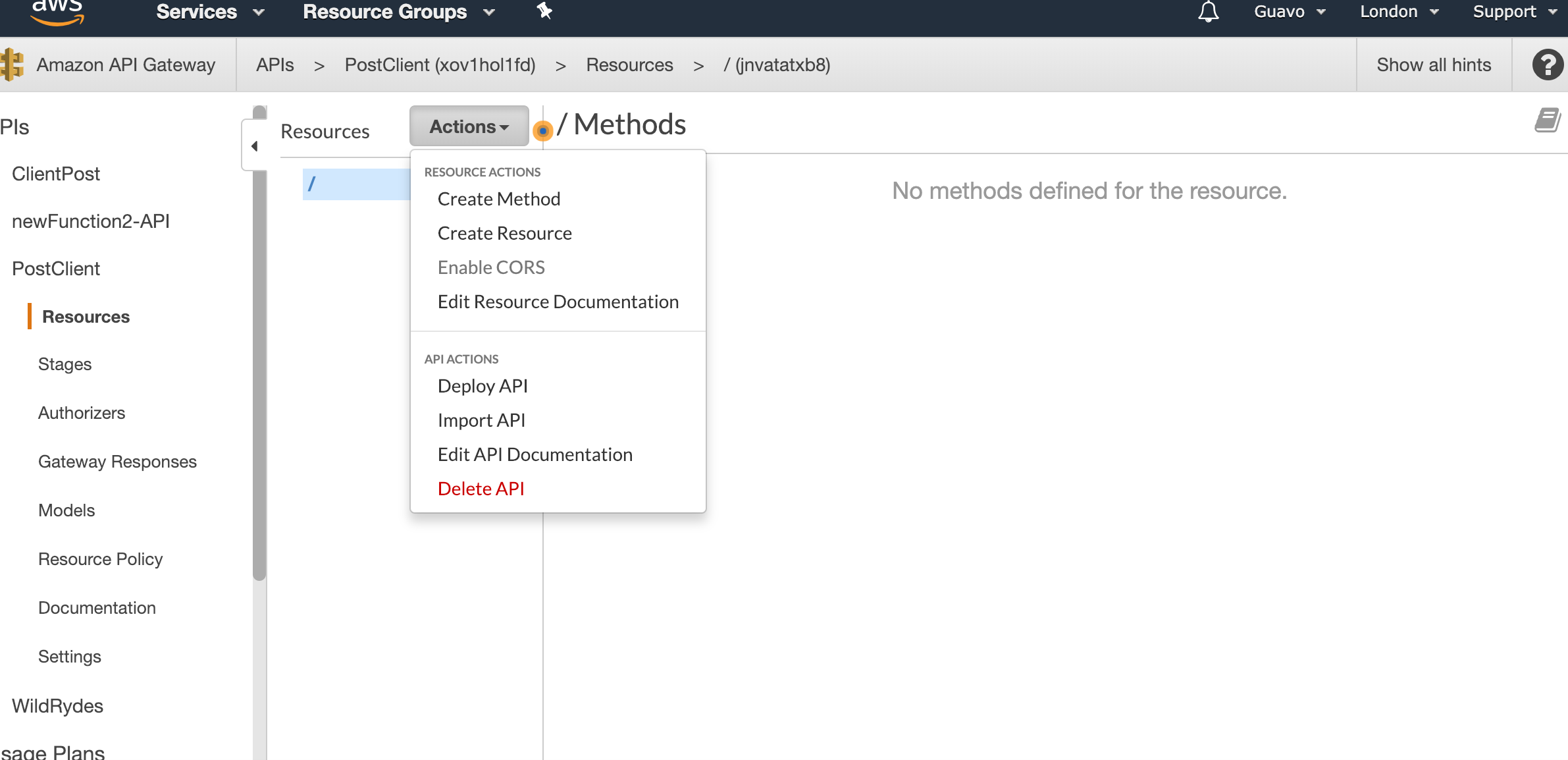


Select Create API, give it a suitable name (we called ours Feniks), and select ‘Edge Optimized’, then ‘Create API’:

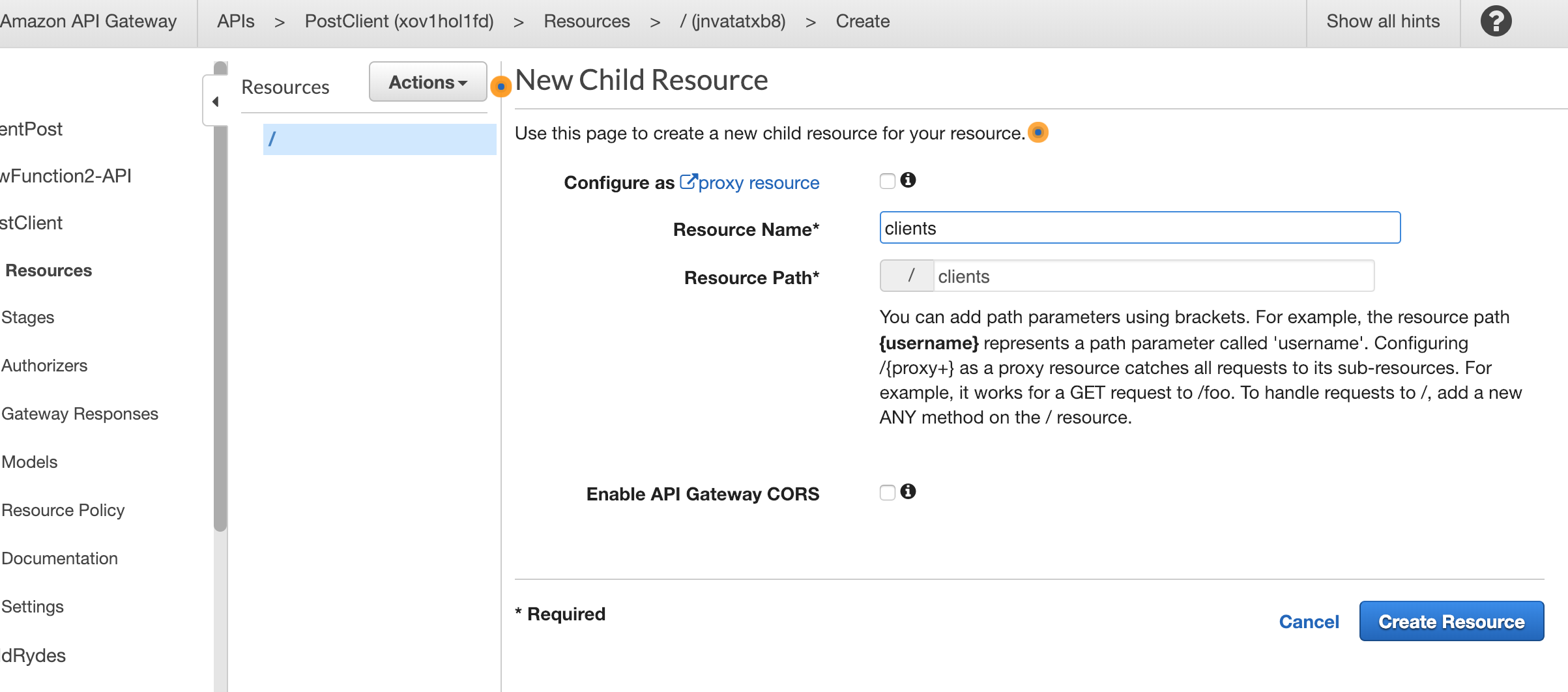


Click on Action to select the Resource and the method or any actions in API.

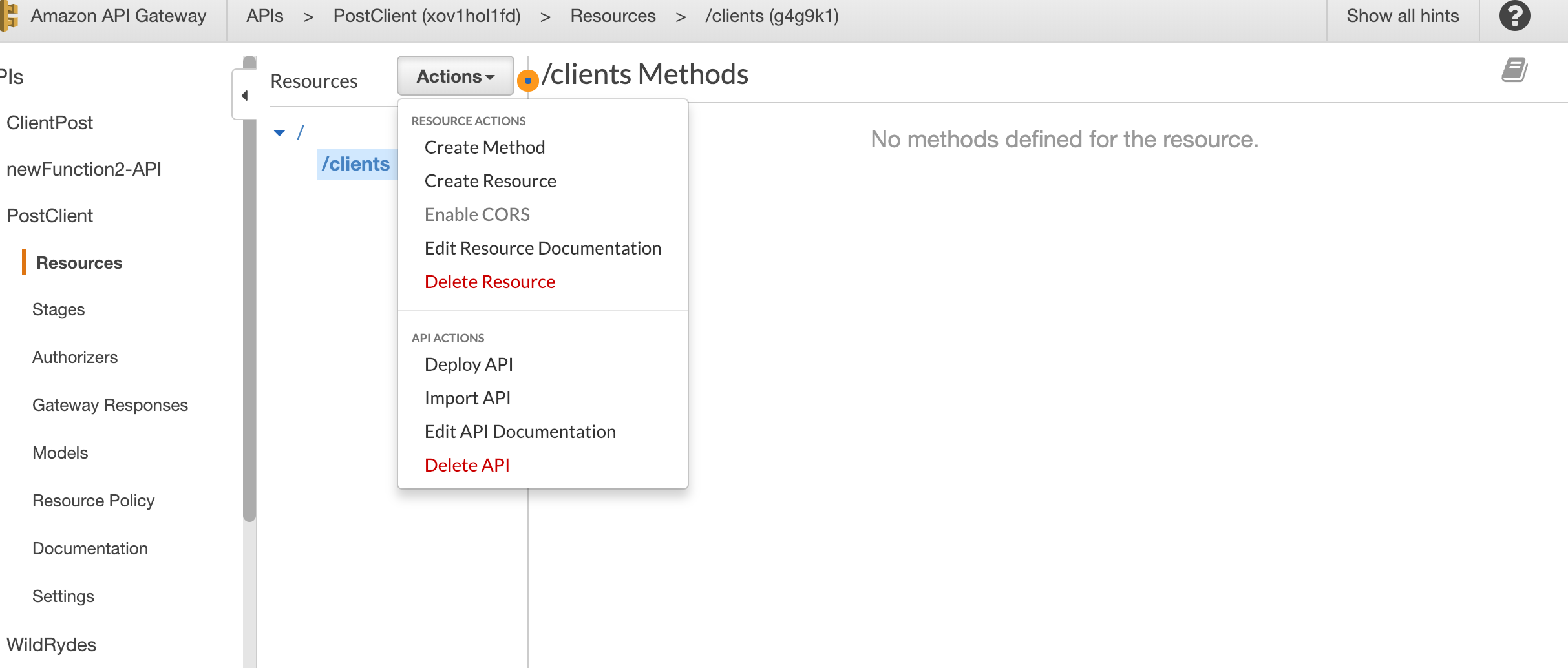
First we need to create a Resource:



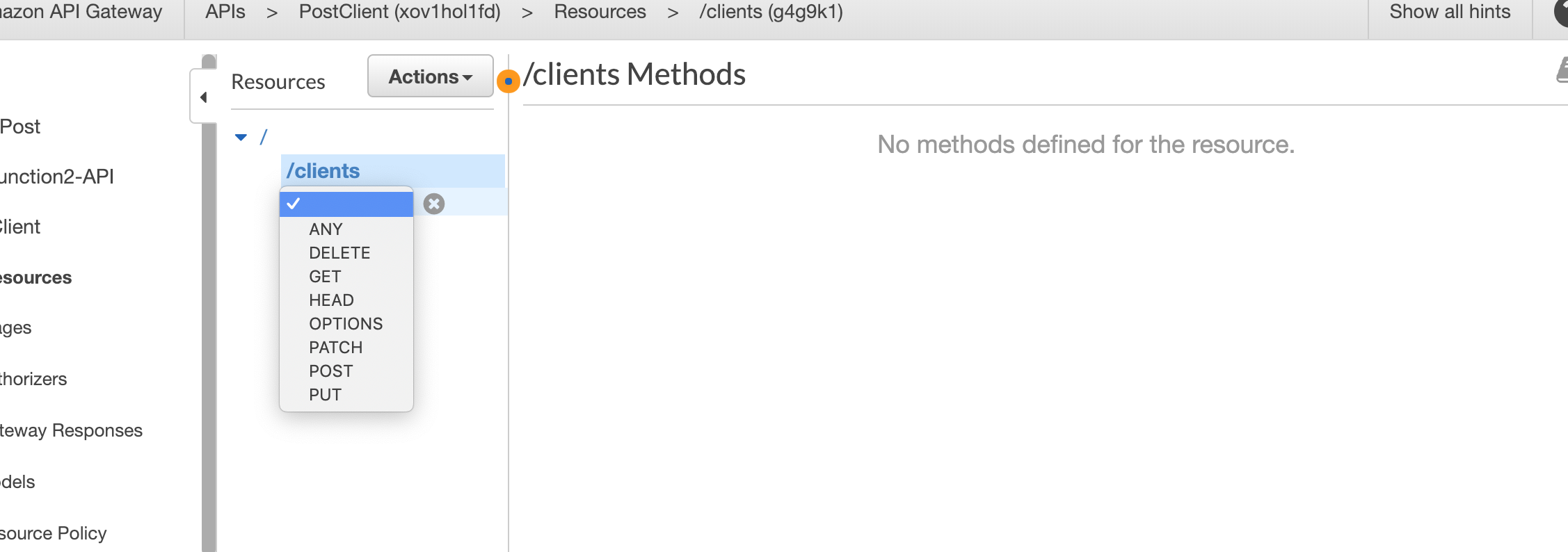
Enter the name of the resource and click on create Resource:



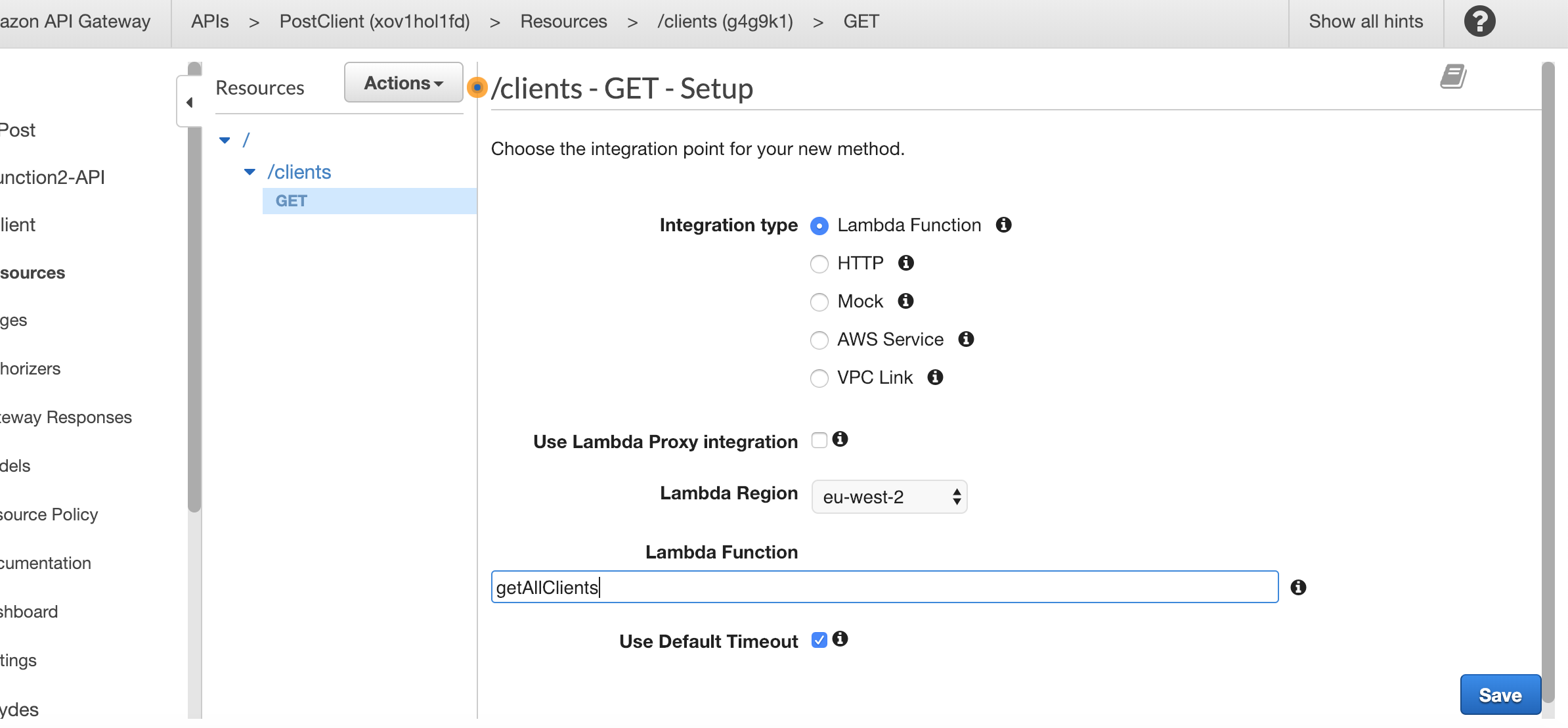
The route will be created. You will then need to add method to the resource by clicking Create Method.



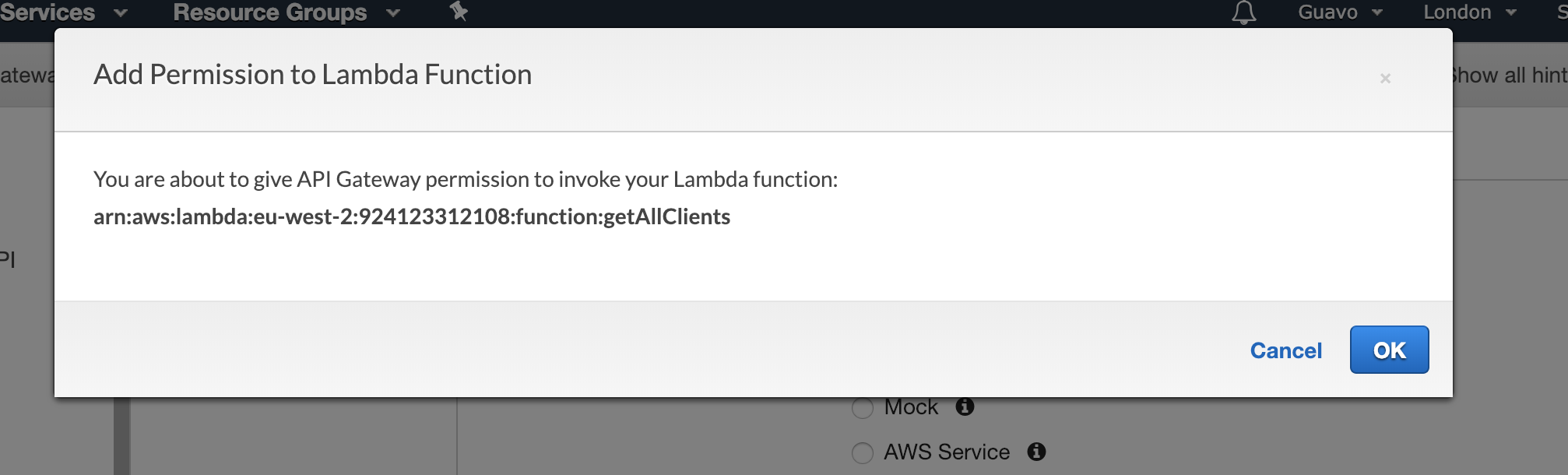
Select the method you want to add in the dropdown menu and click on the tick to confirm.



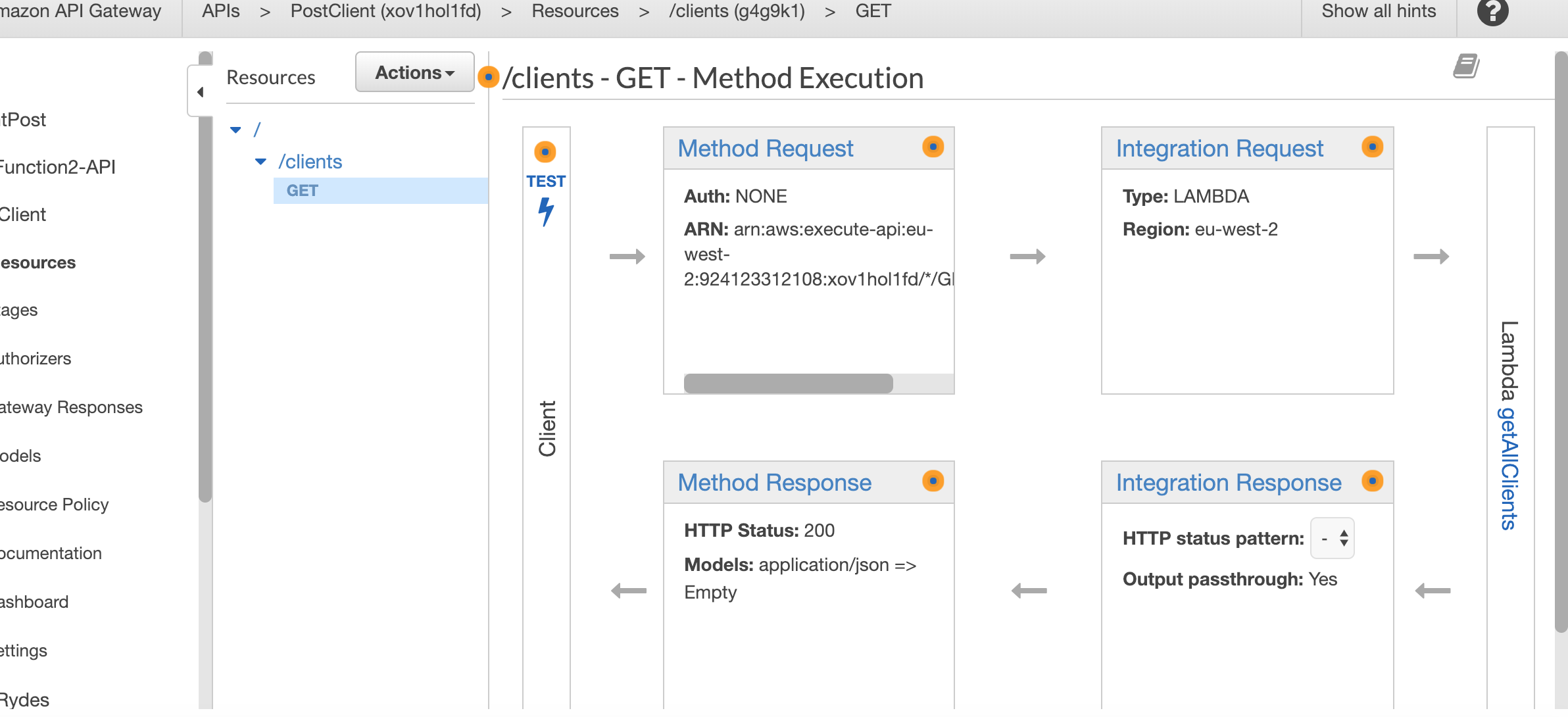
Leave the option by default and enter the name of the lambda function that you want to use for this method, then click on save.



A pop up will ask you to confirm, click on OK

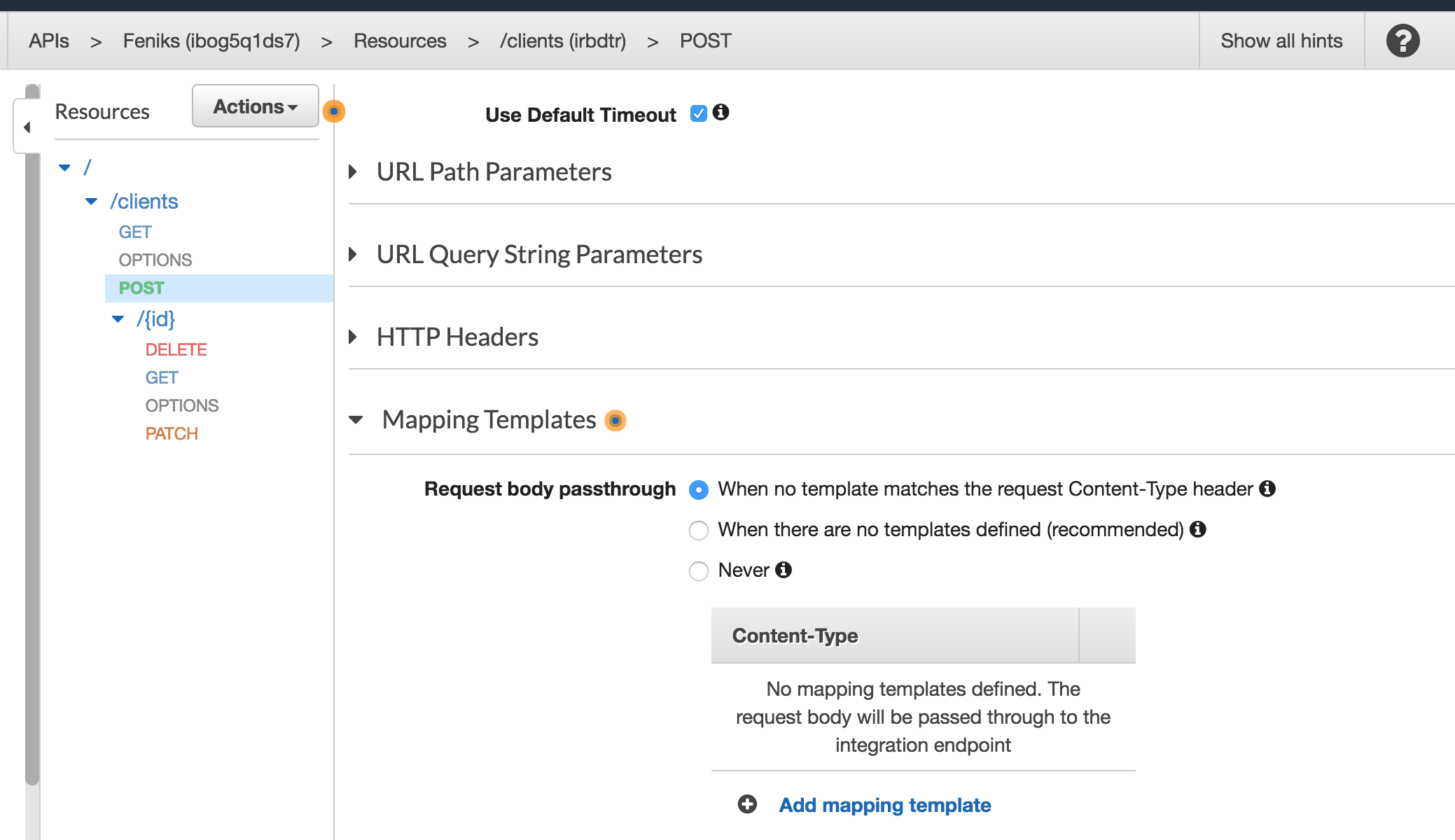


You will be redirected to this page:

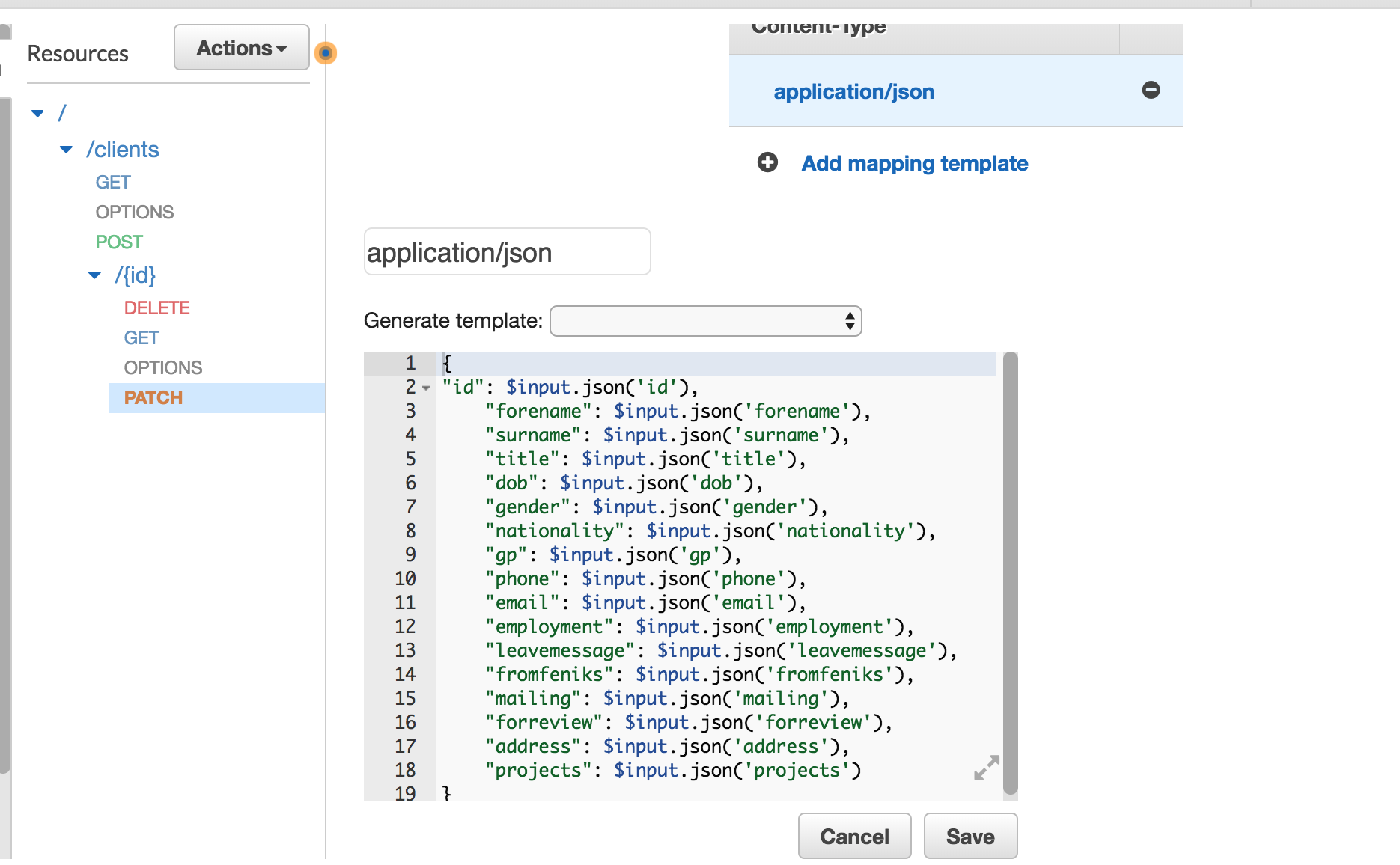


If the method is writing to the DB, then follow the next steps. If the method is only reading from the DB, then skip this part and go to “Select the Resource”.

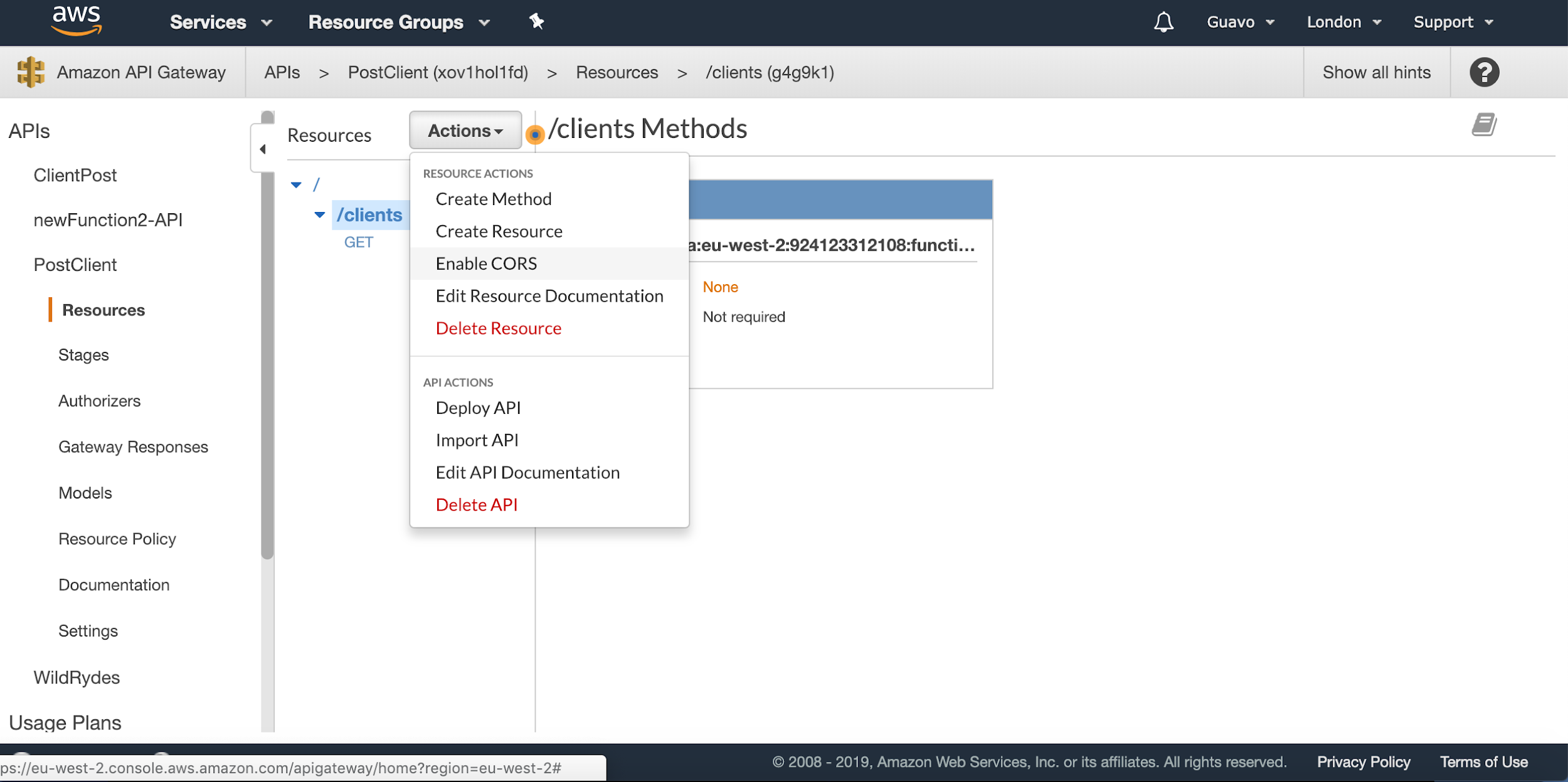
Click on Integration Request. Scroll down to the bottom of the screen find Mapping Templates. Expand this section and click on “Add Mapping Template”.



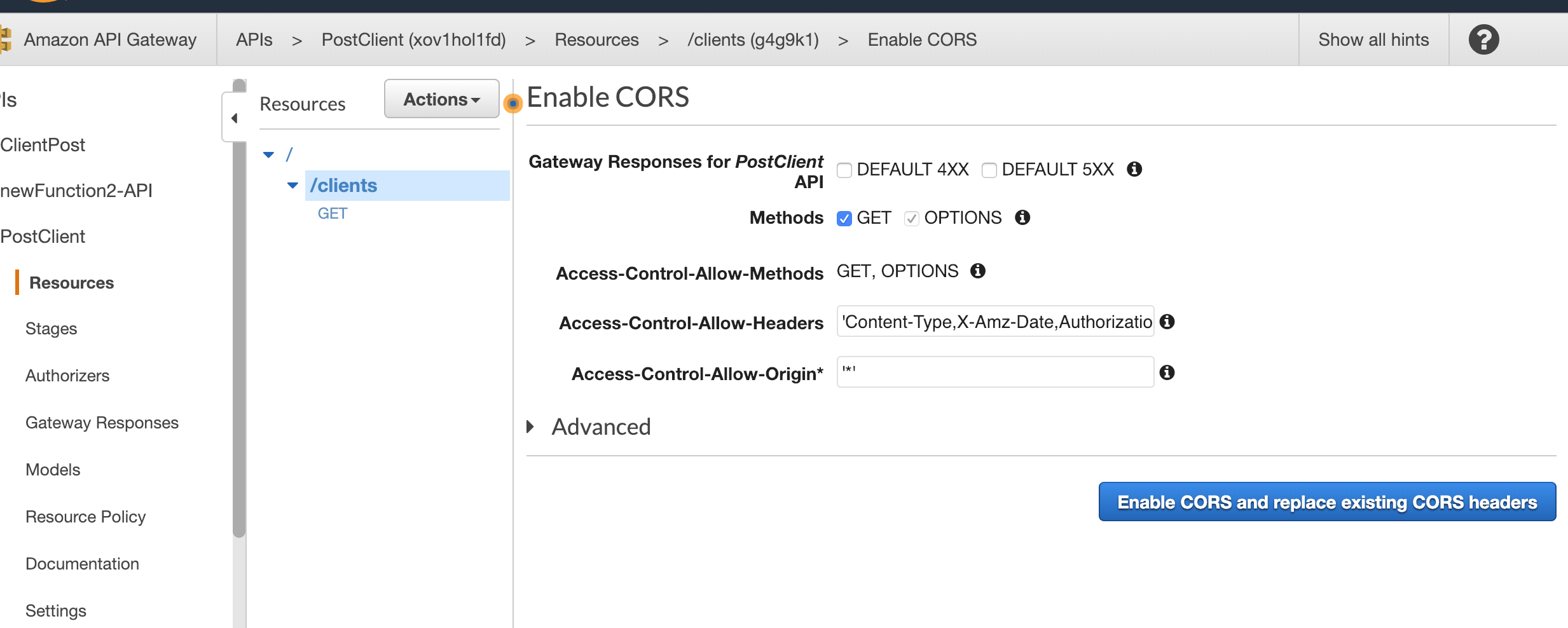
Click on the resulting JSON and enter an object in the following style, and press Save:



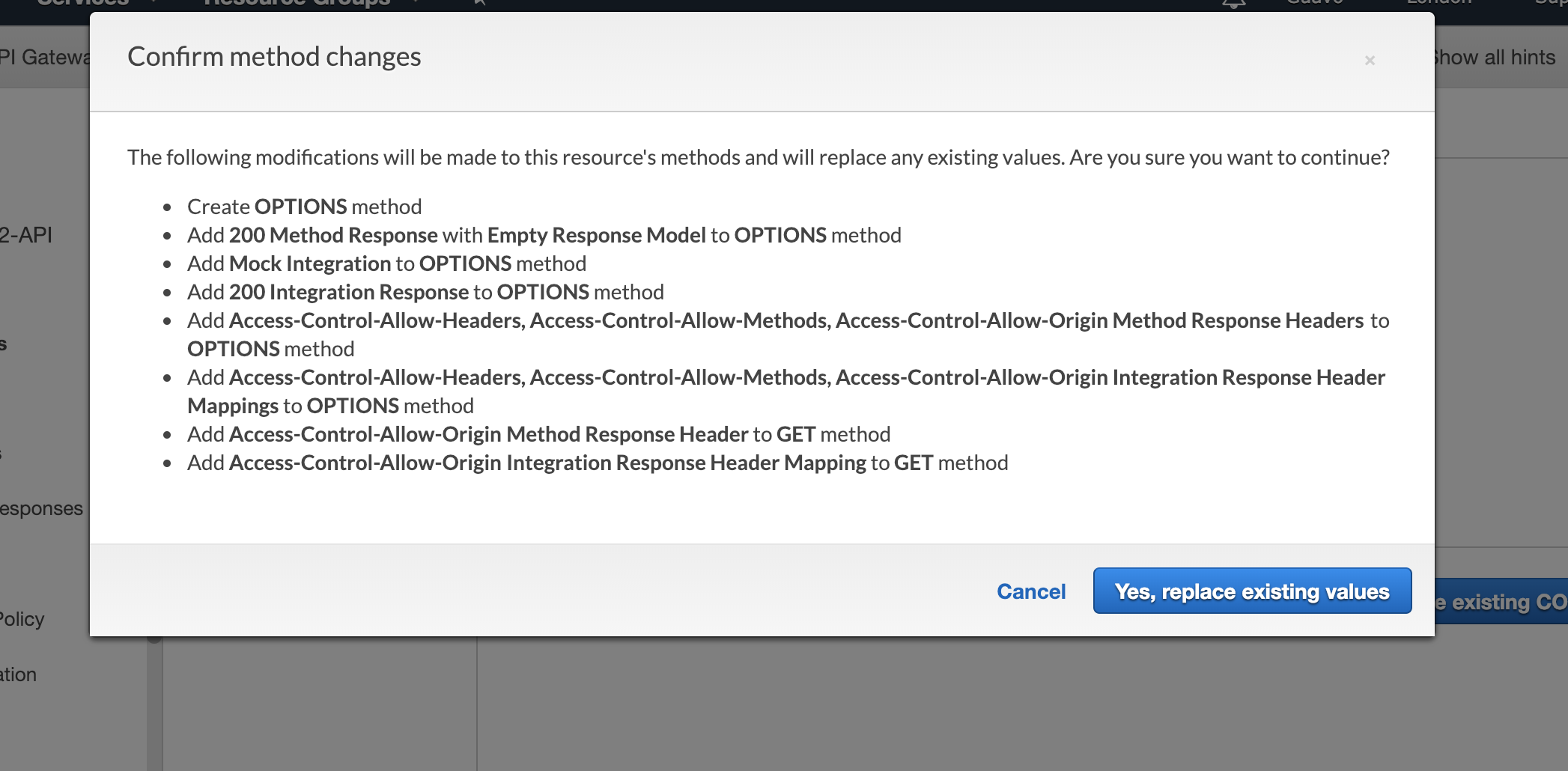
**Select the Resource** that you created and select Enable CORS

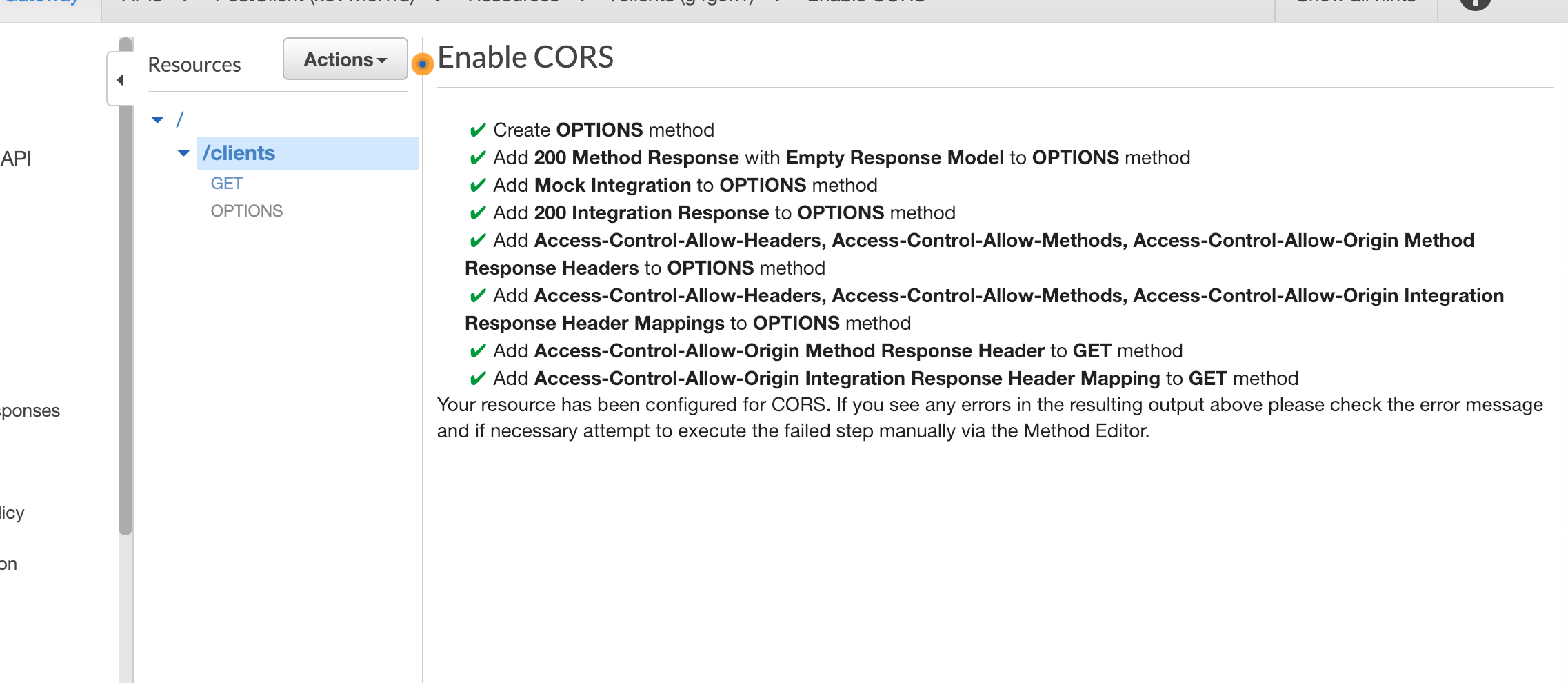


Leave everything by default, and click on Enable CORS



Click on Yes, replace existing values

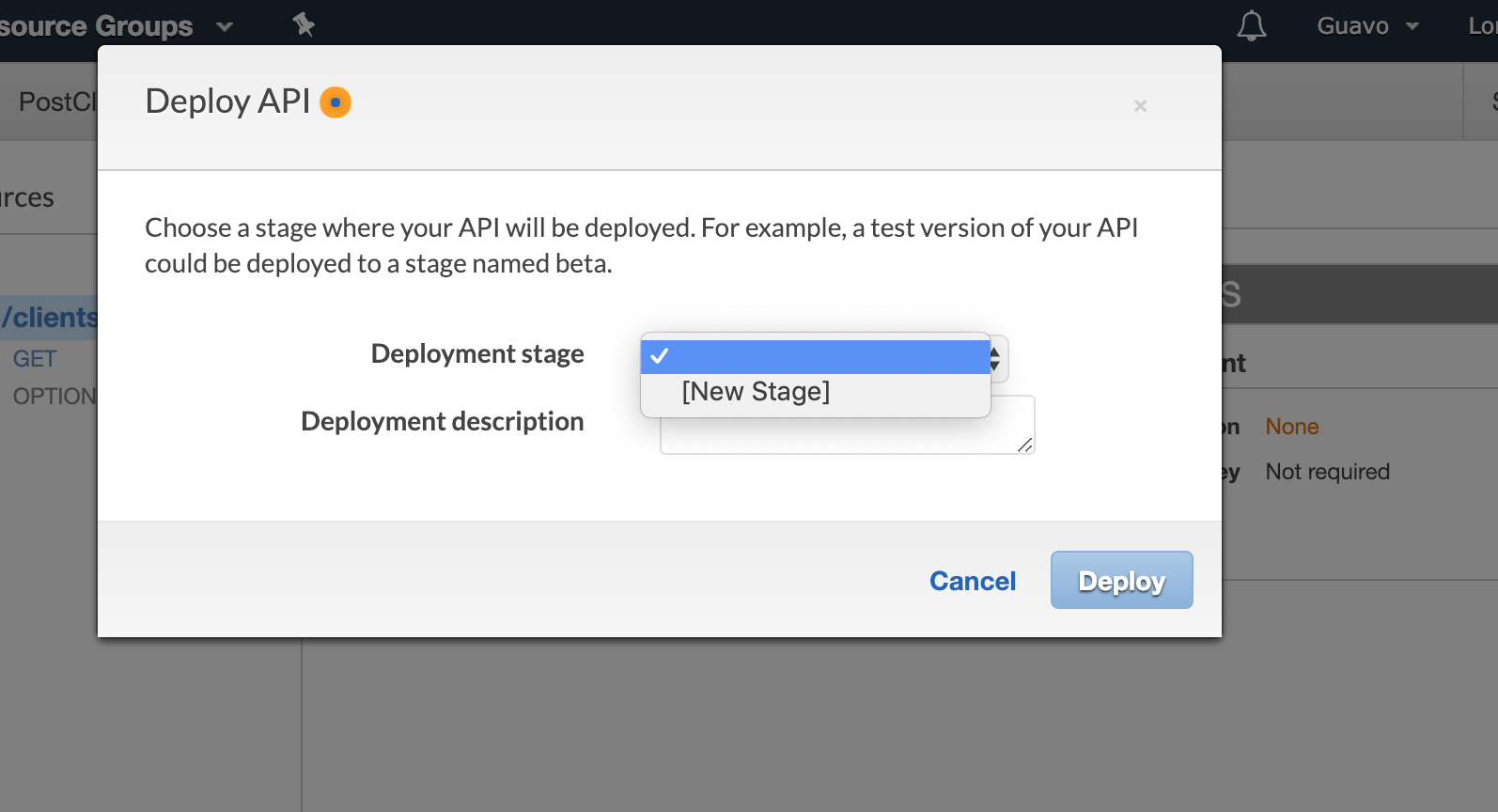




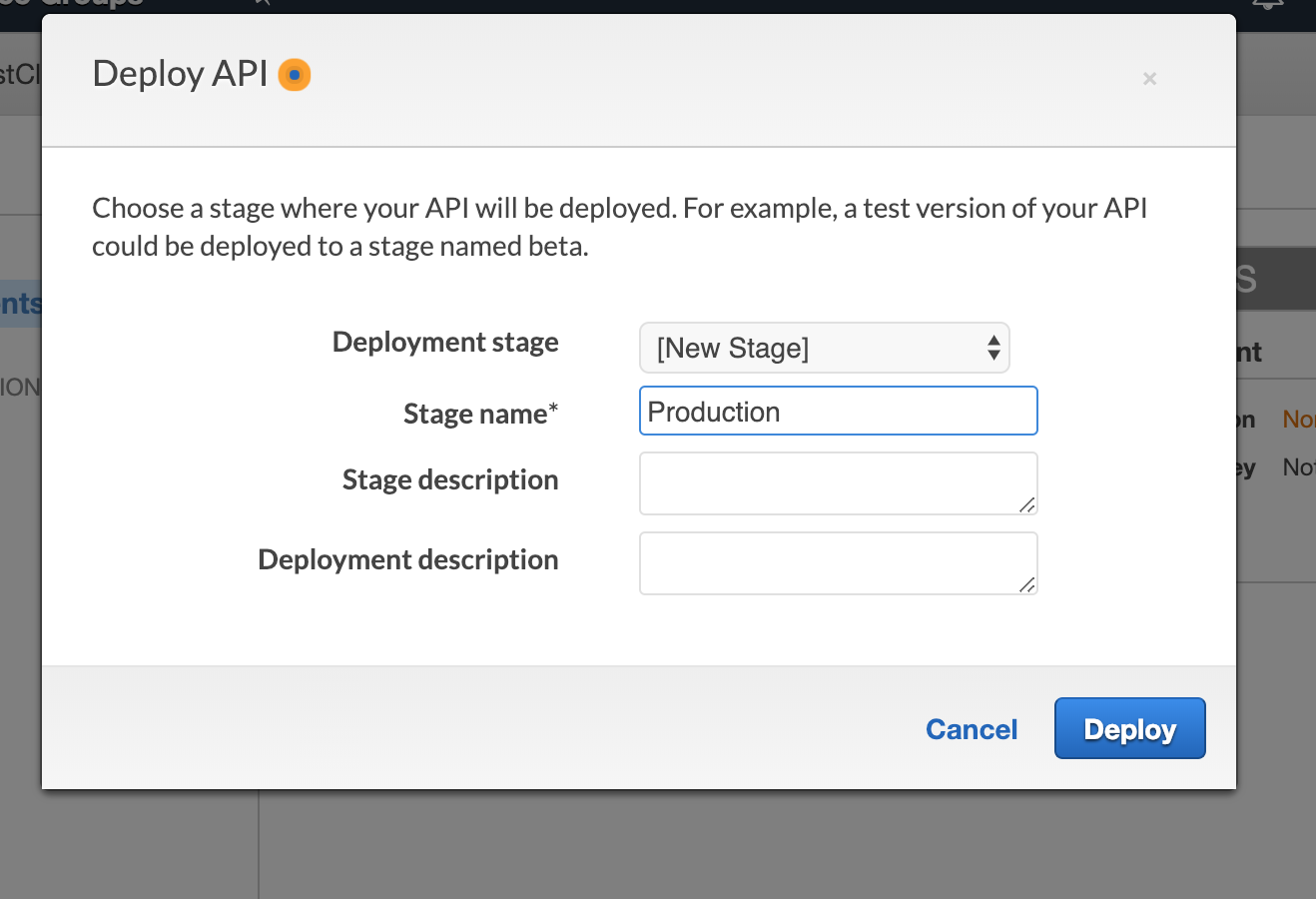
Then under Action select deploy API



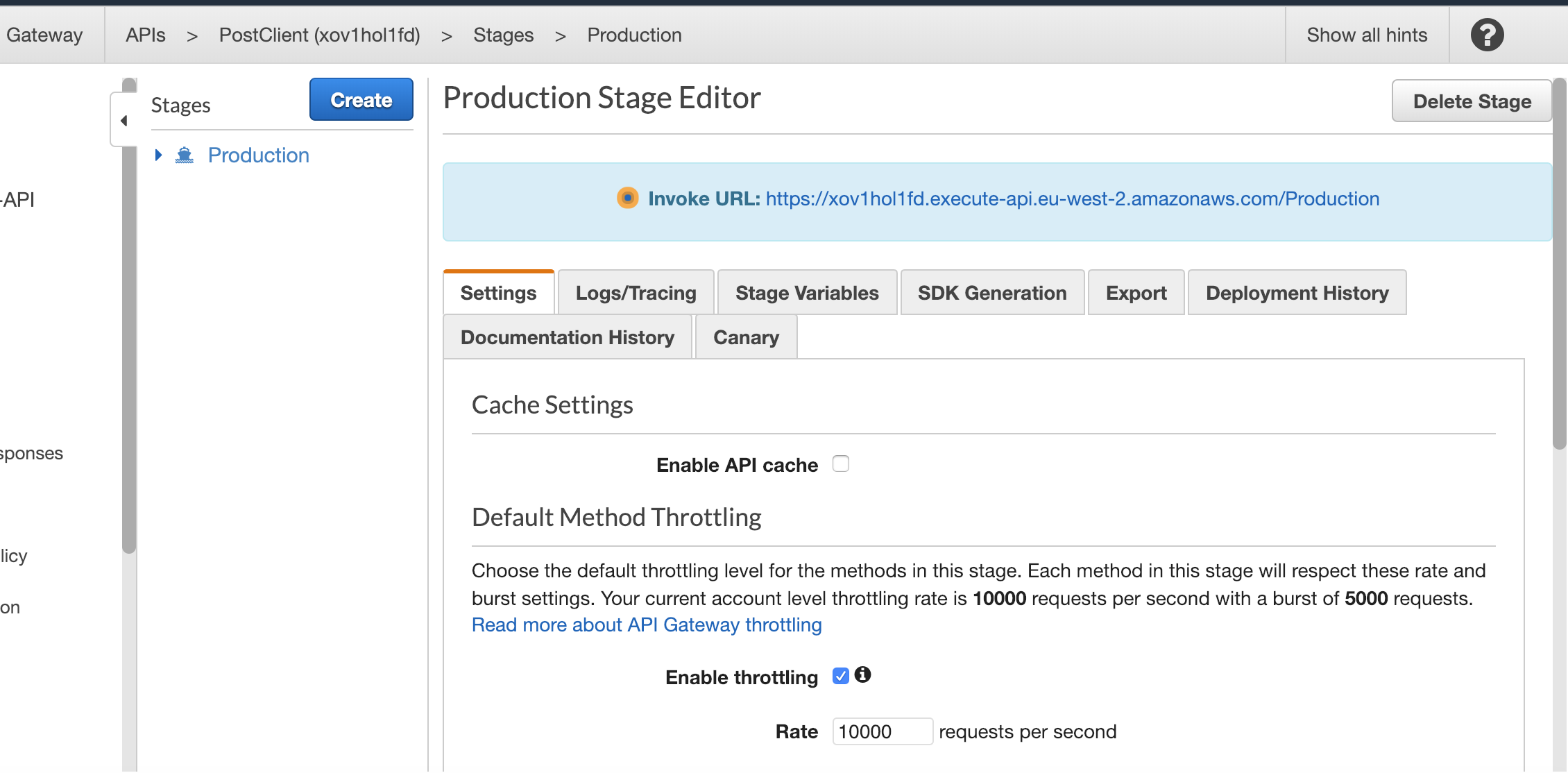
Select New Stage from the dropdown menu



Then enter the name of the stage and click on deploy

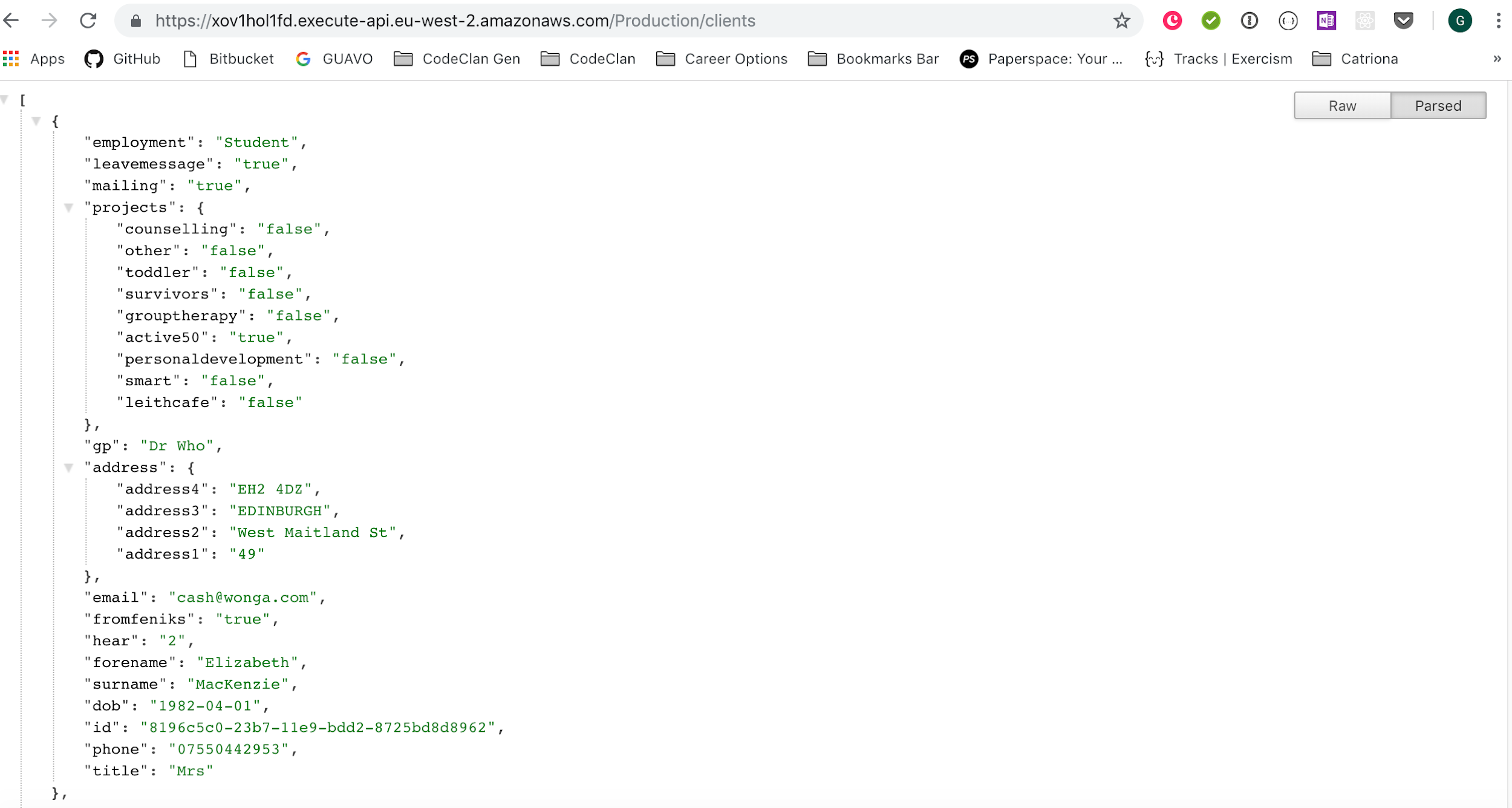


Copy the Invoke URL. This is the endpoint of the API that you will use to access the backend.

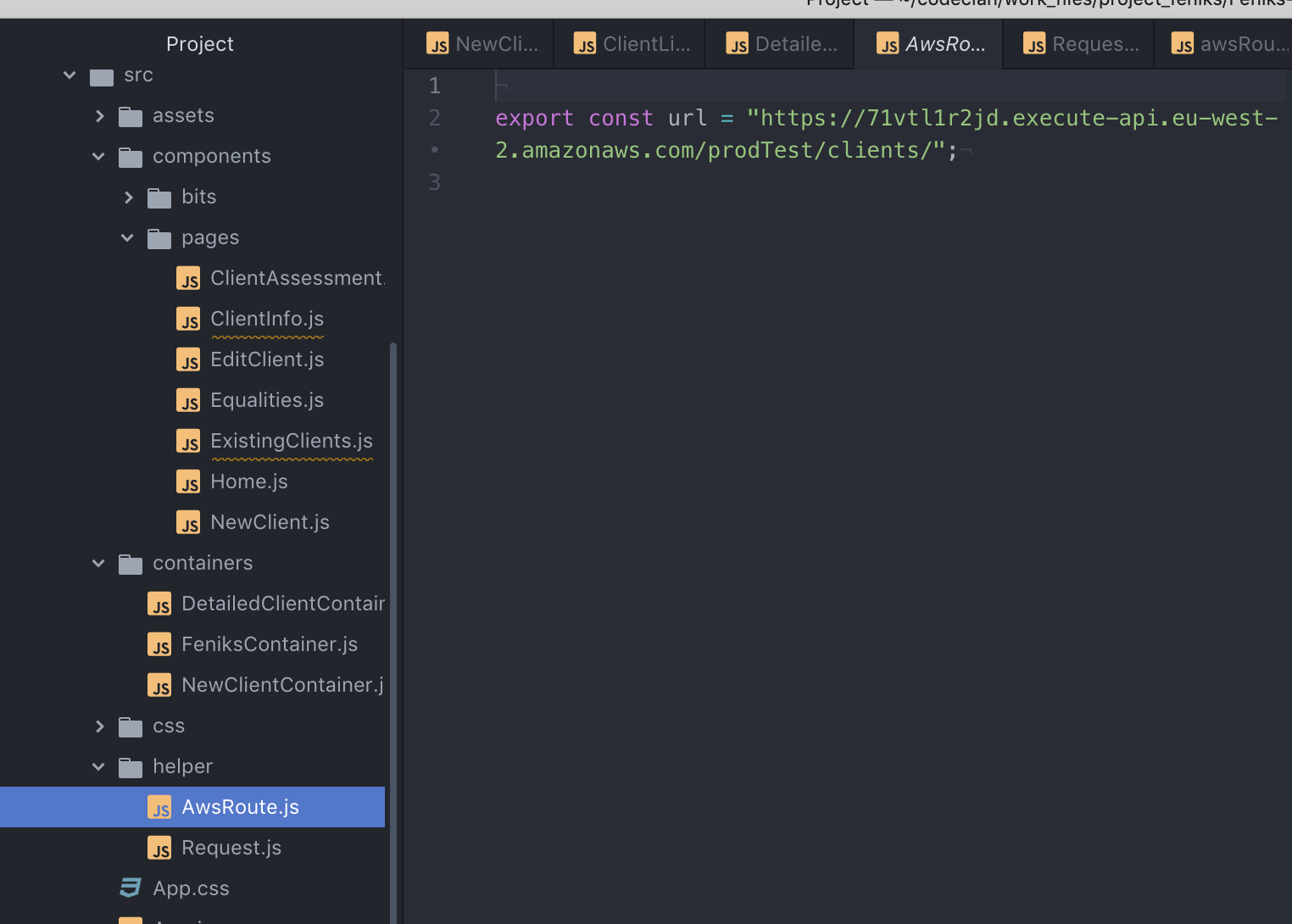


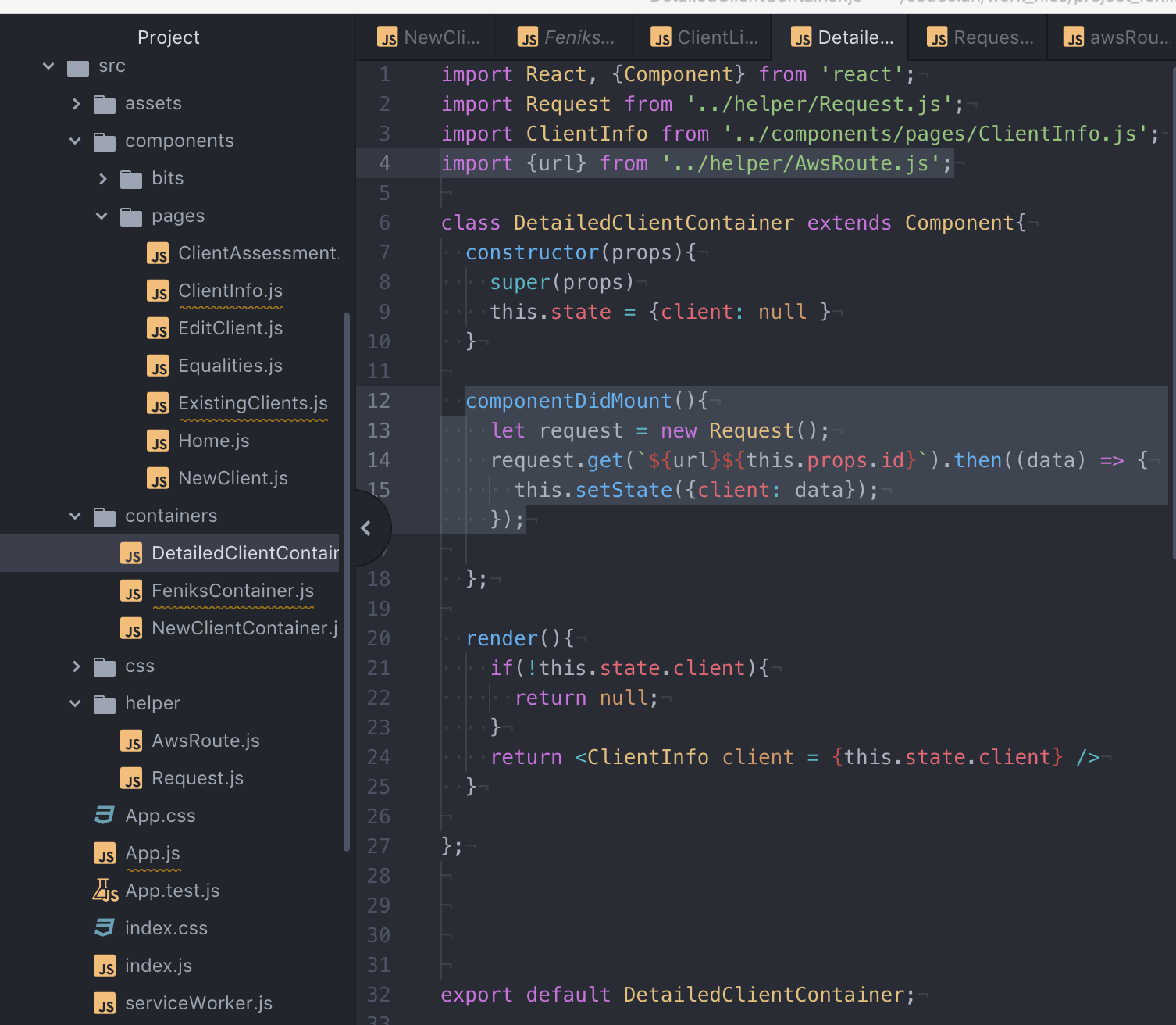
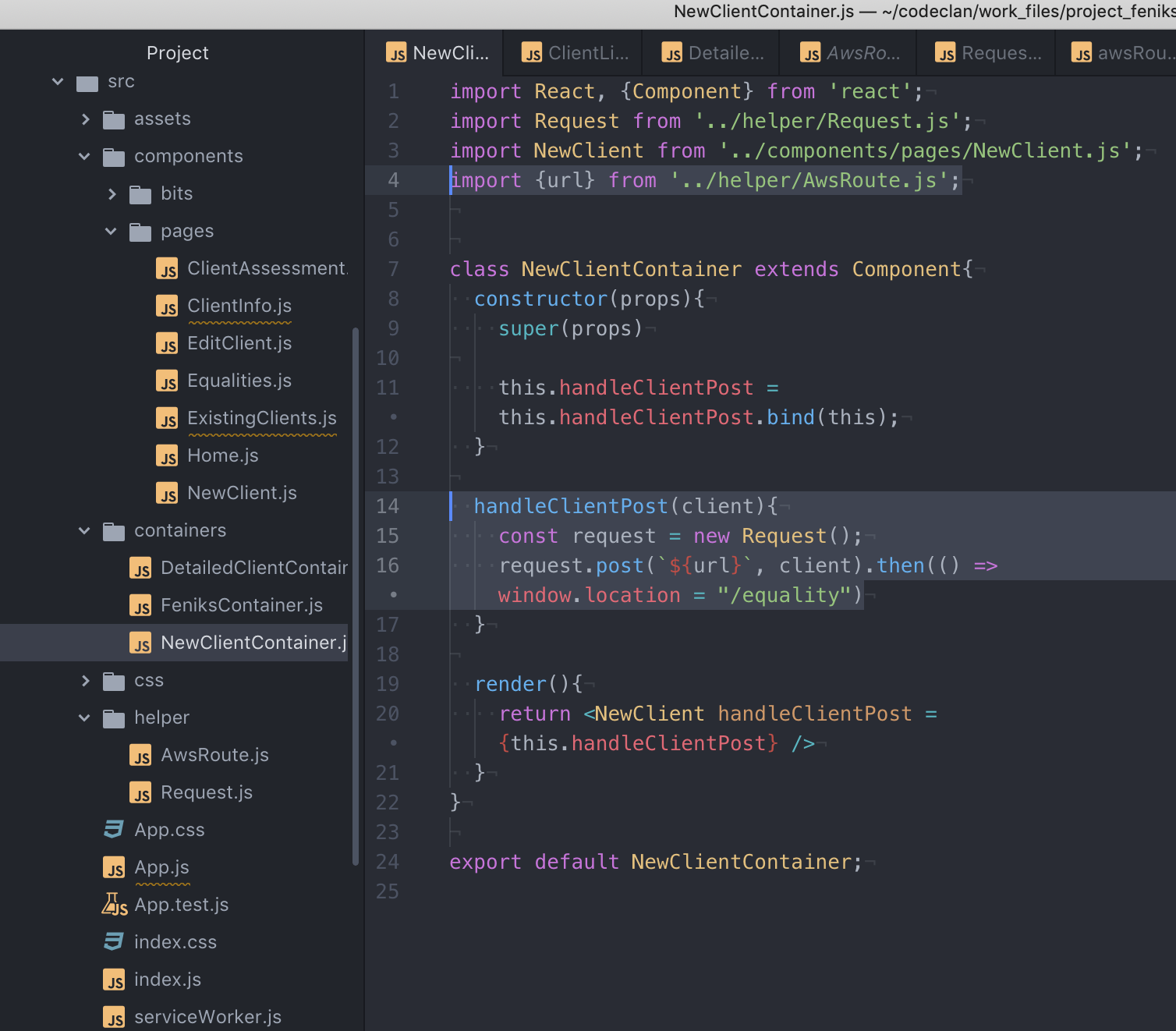
To access the database you need to add the resource to the endpoint:

https://xov1hol1fd.execute-api.eu-west-2.amazonaws.com/Production/clients



In the React JS Project, the url is only changed in the AwsRoute.js file, where it is exported as a constant and is imported in the files that need to make requests through API Gateway:





Remember that you can also check the post functionality by looking directly at dynamoDB, or using Insomnia.