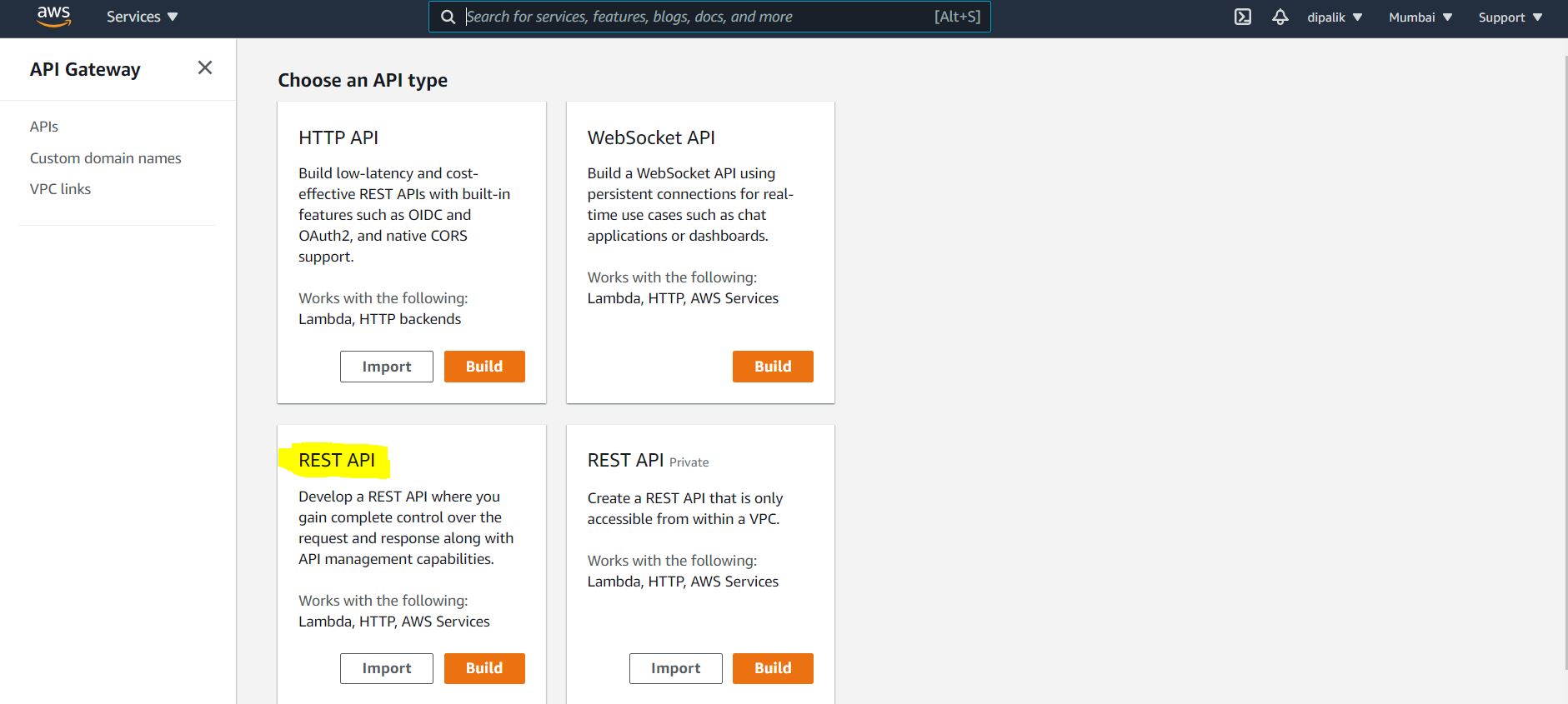
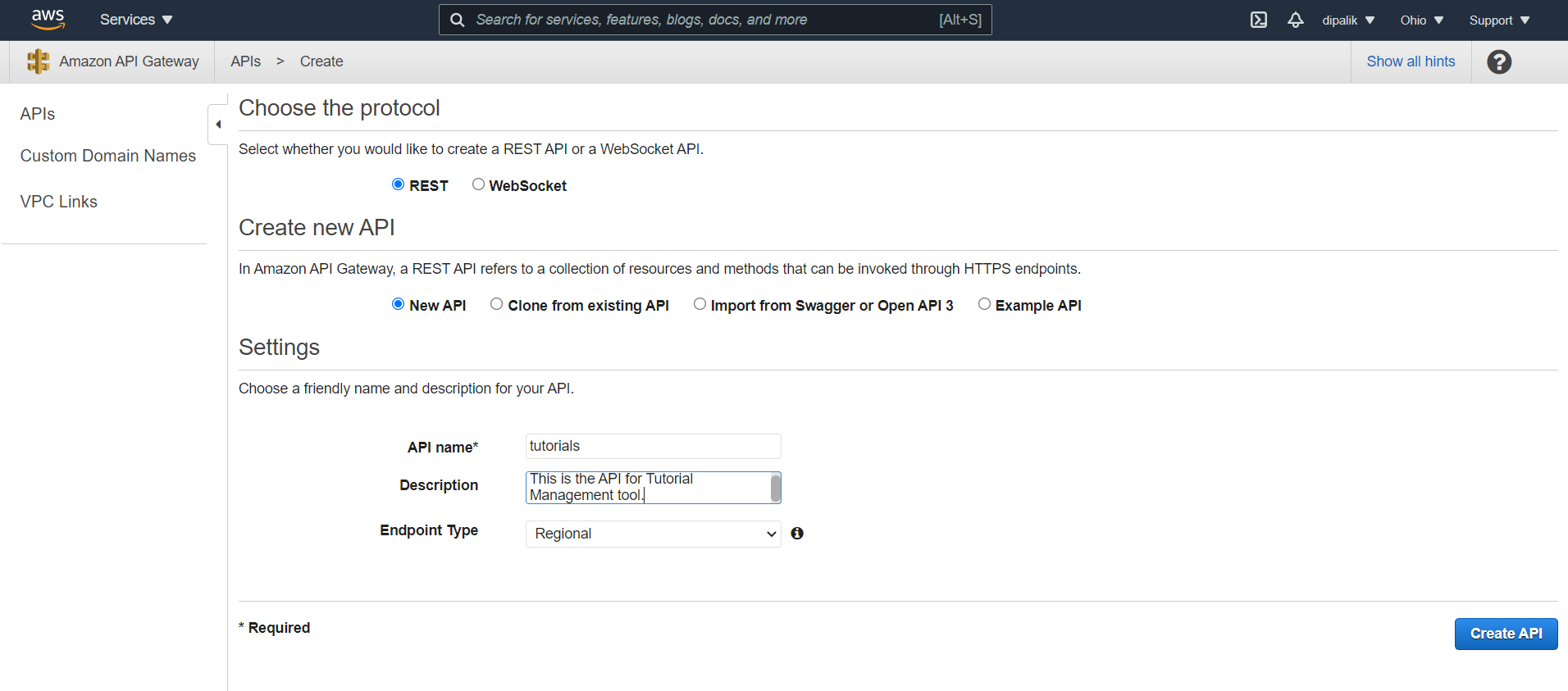
***Amazon API Gateway*** - is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, 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, CORS support, authorization and access control, throttling, monitoring, and API version management. API Gateway has no minimum fees or startup costs. You pay for the API calls you receive and the amount of data transferred out and, with the API Gateway tiered pricing model, you can reduce your cost as your API usage scales.

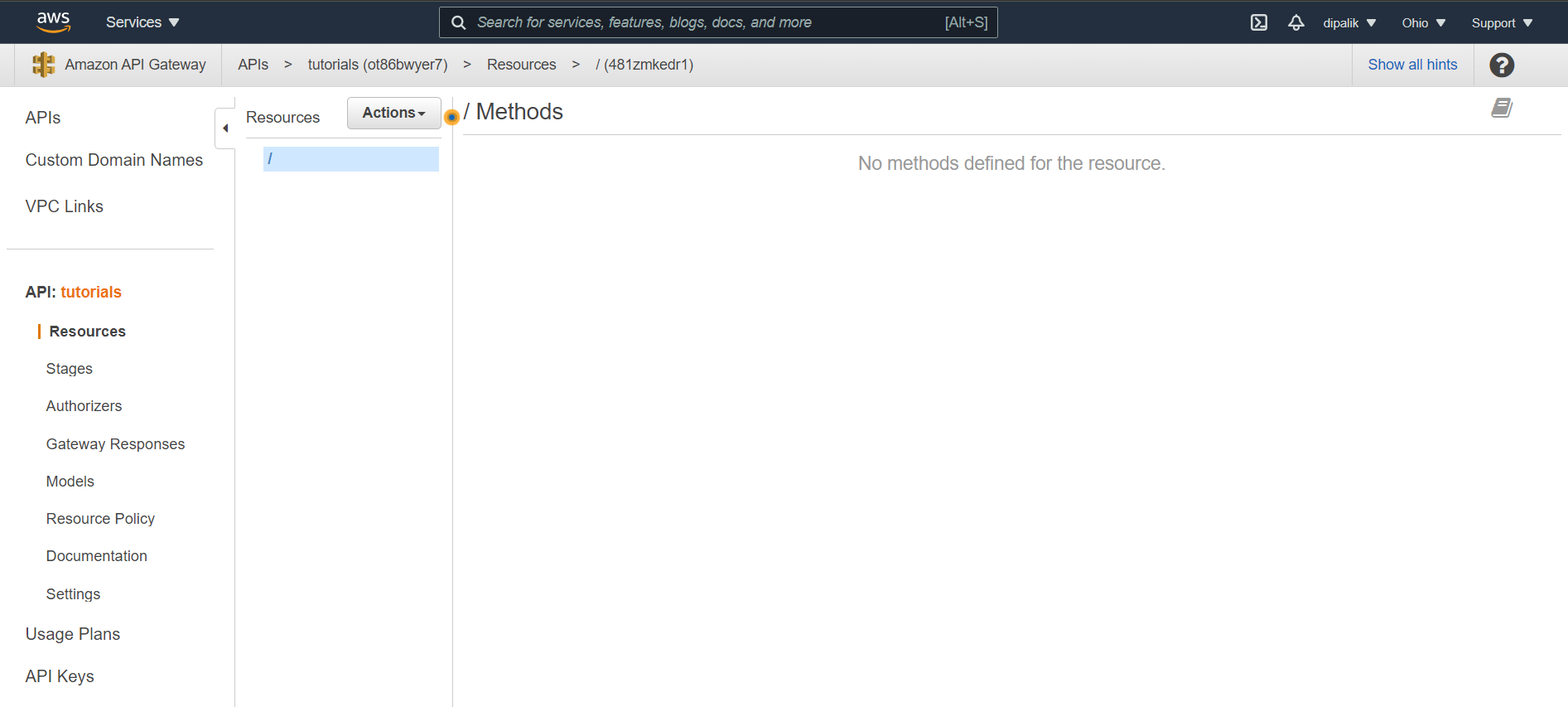
Go to the Amazon Console and Search API Gateway, then click on Create API Button, then Choose the REST API option on the page.



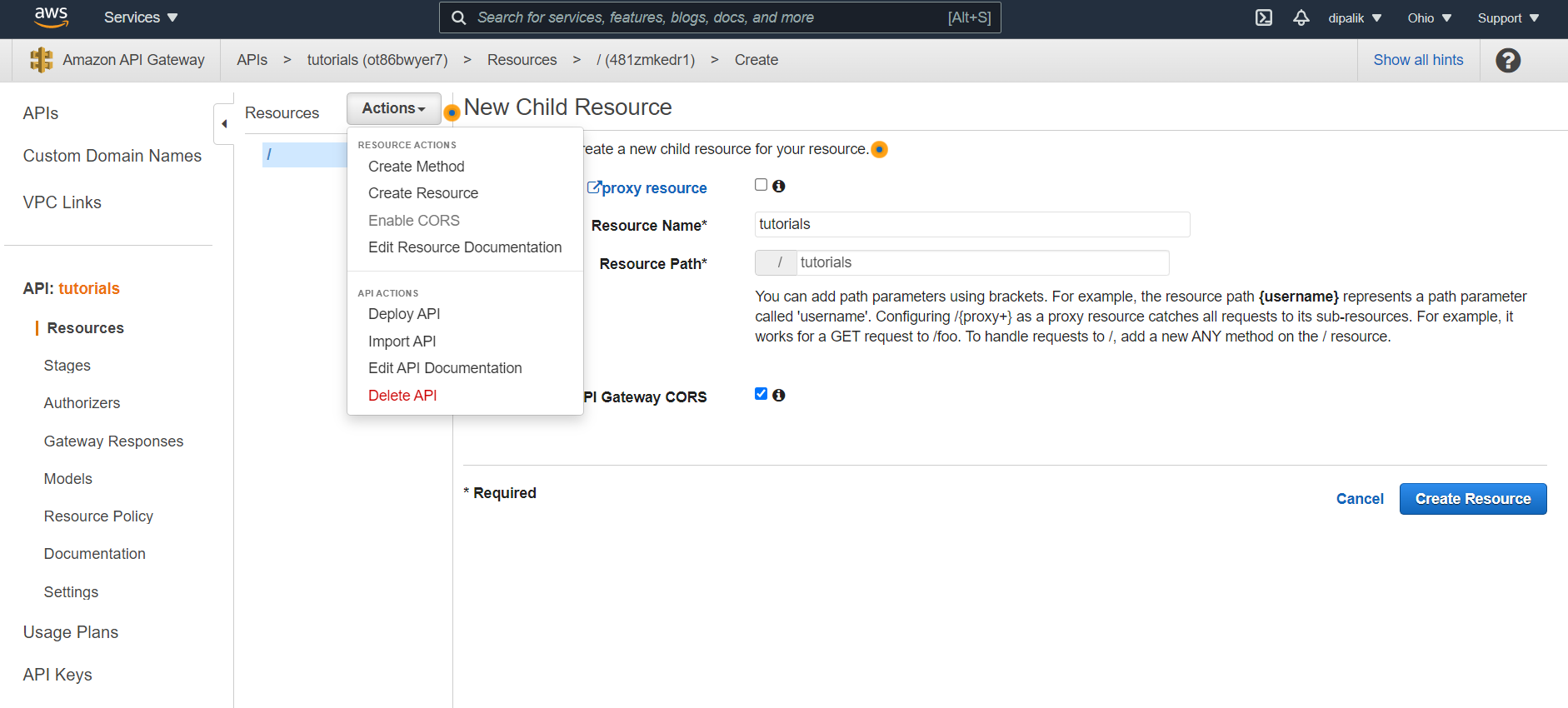
Enter the API Name, Description and Endpoint Type



This will create a new API and you’ll be ready to start adding resources (URL paths) and methods (GET, POST, etc.):



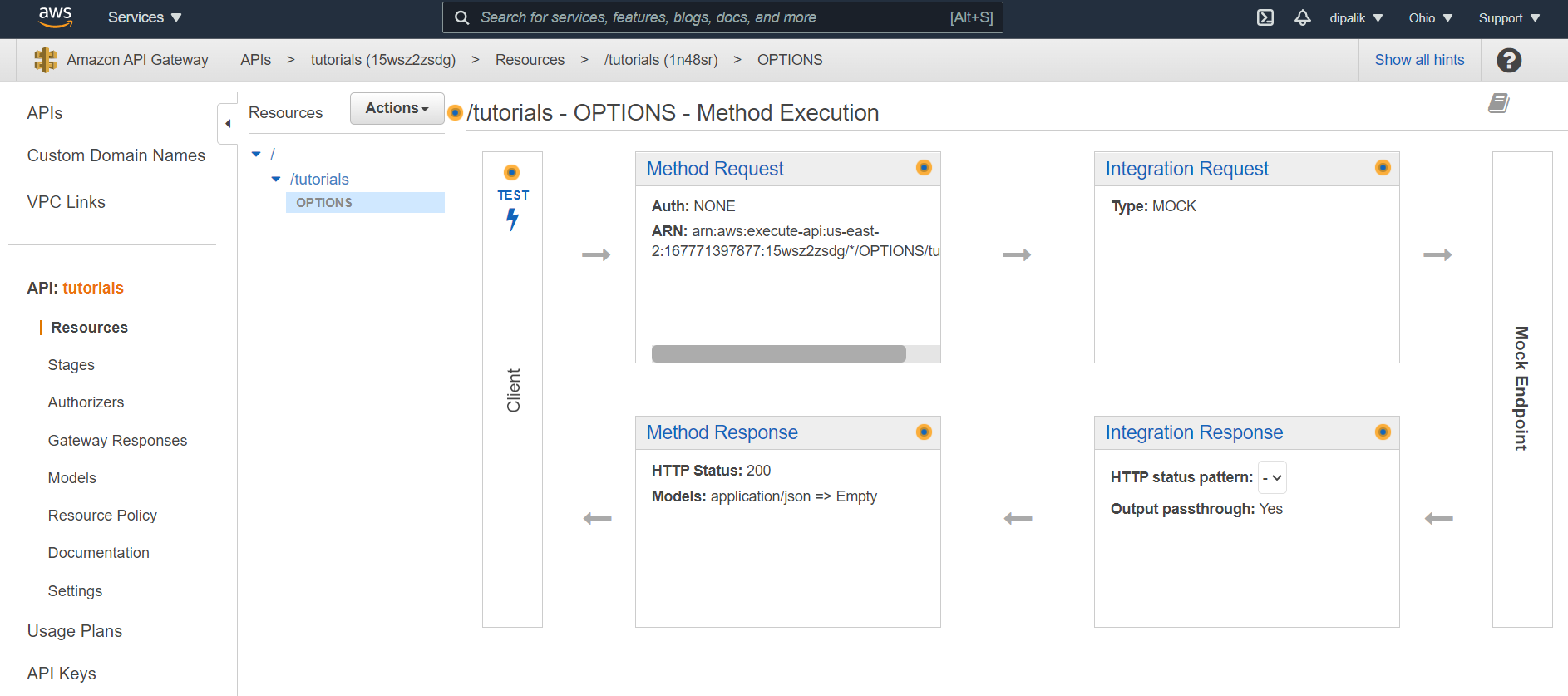
Now open the Actions menu and choose Create Resource. Enter a resource name and check the option Enable API Gateway CORS:



By default, the URL path will be created from the resource name. [Cross-origin resource sharing](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) (CORS) is a mechanism that allows a web page to request resources hosted in another domain (which is going to be our case here). With this mechanism, the server sends some headers to tell the application that is OK to access the resources on that different server.

Next, click on Create Resource:

By default OPTIONS method will be create which will enable the CORS on the API.

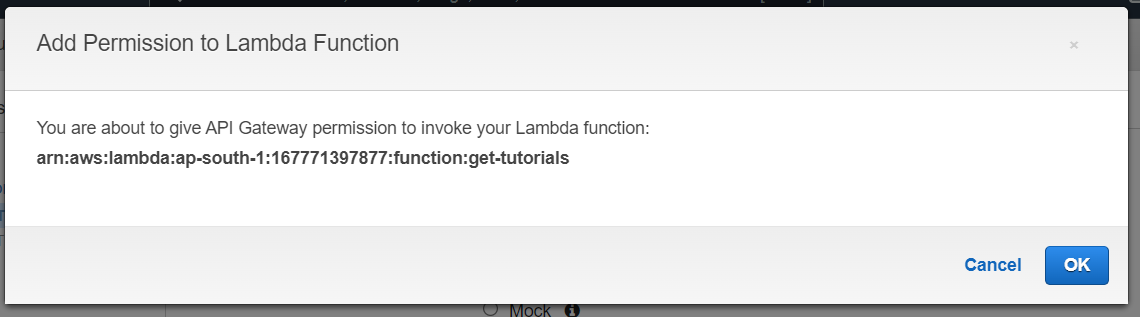


Next, click on Create Method :

Select GET from the drop down menu, and Choose Lambda function for integration type, select the Lambda Region (where you have created your Lambda functions), and then select the name of the Lambda Function which should be called on the execution of this API Method.



When prompted, allow API to add permissions to invoke Lambda Function.

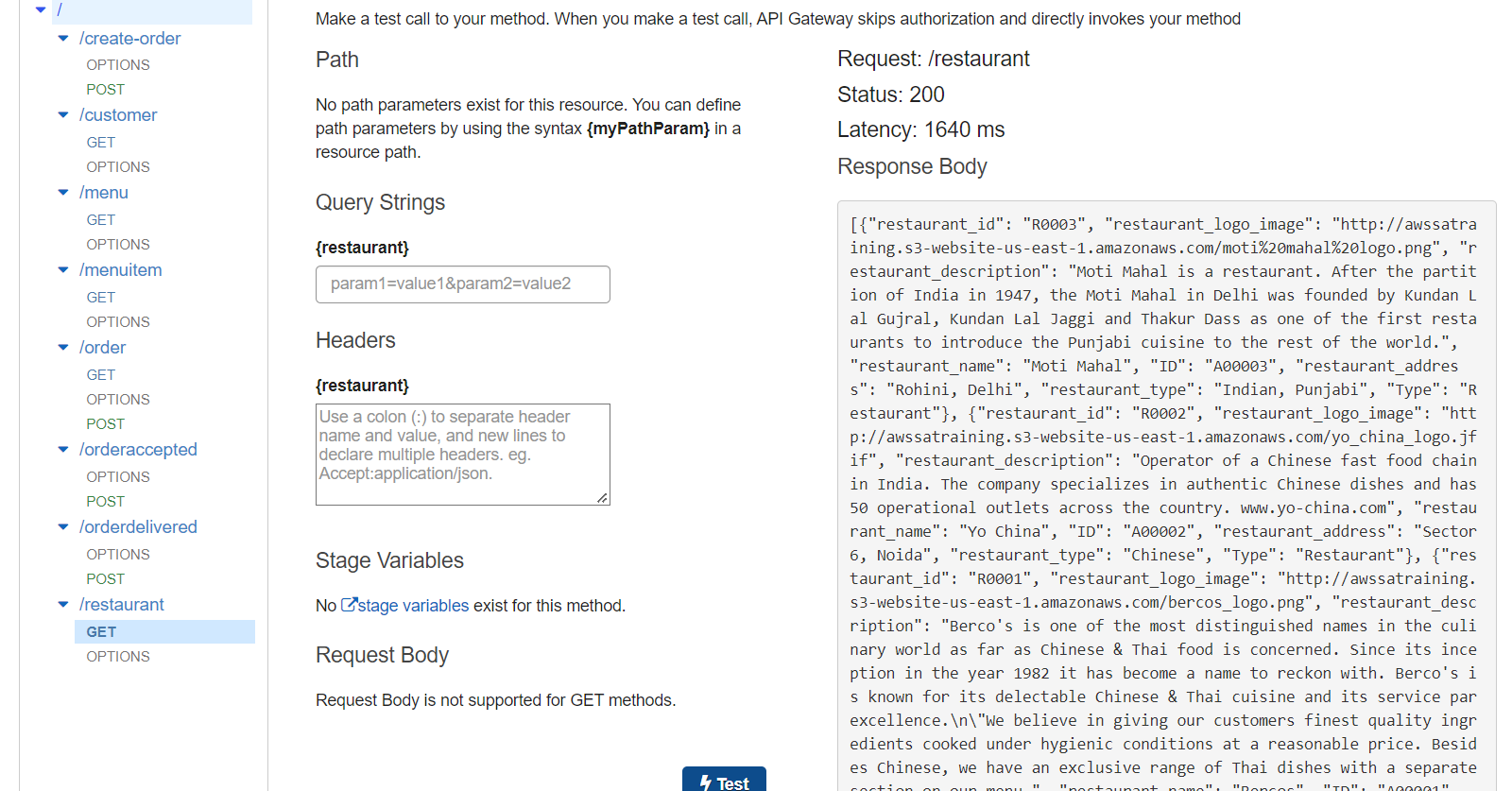


You API is created successfully.



Setup the Integration Mapping for the incoming request to map it to the Lambda parameters.

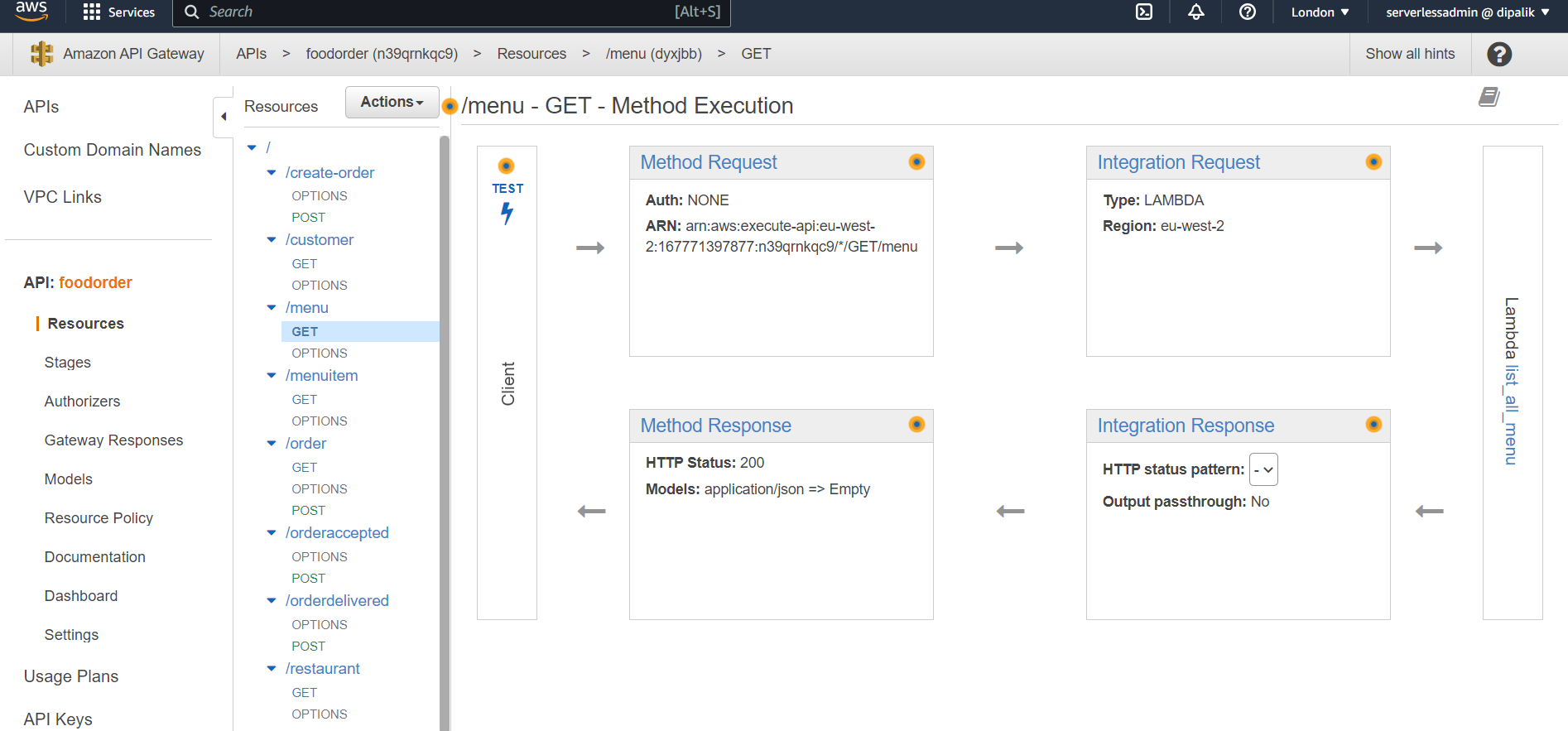
Now Test your api using the Test Feature provided by AWS.



You should see the response of your API in JSON format generated by the Lambda function.

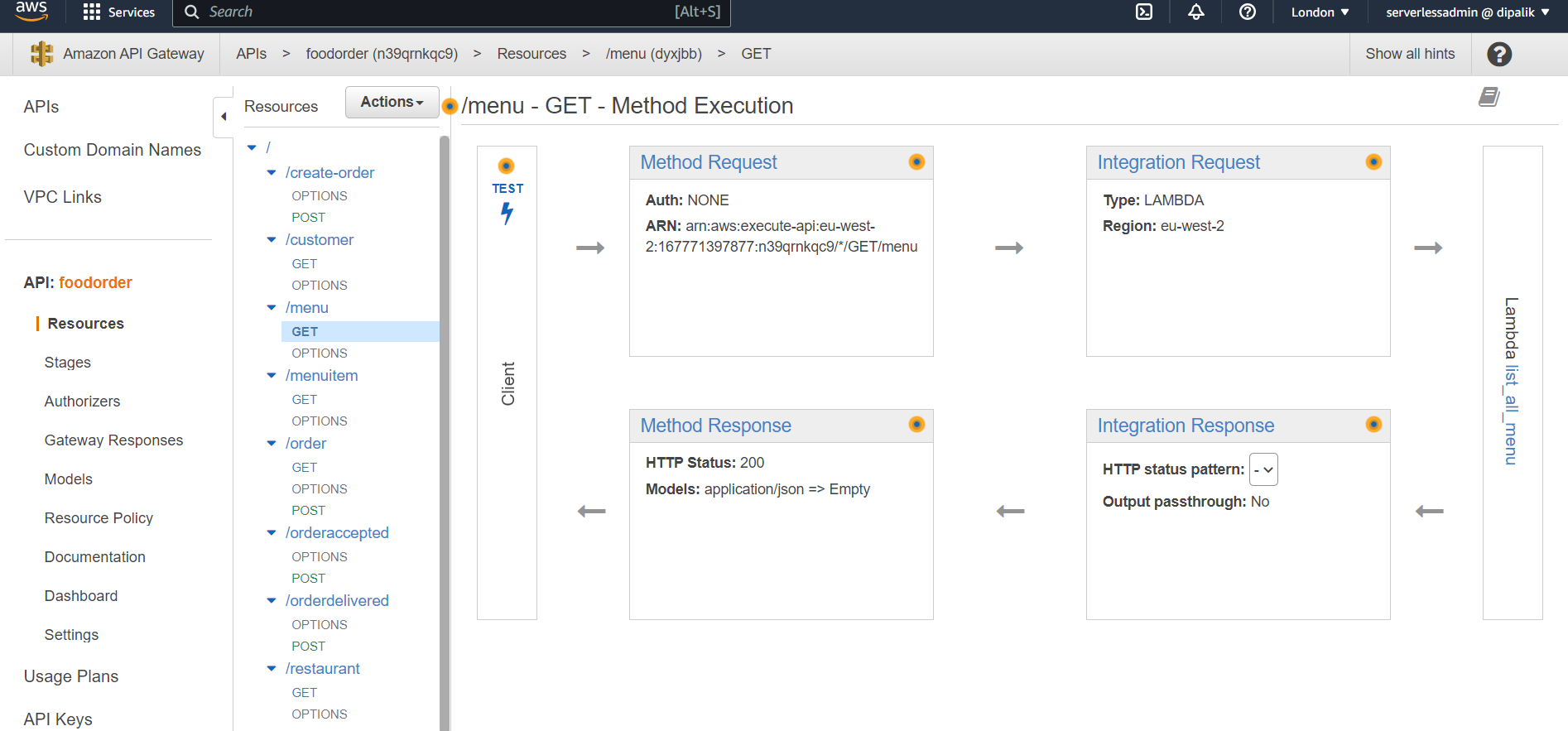
Moving on, we are also going to need a filter ?restaurant\_id=R0001, where R0001 corresponds to the ID of the restaurant for which we want to filter the menu.

Select the menu resource and from the *Actions* menu, create another resource, this time just adding a path variable using brackets (and enabling CORS):



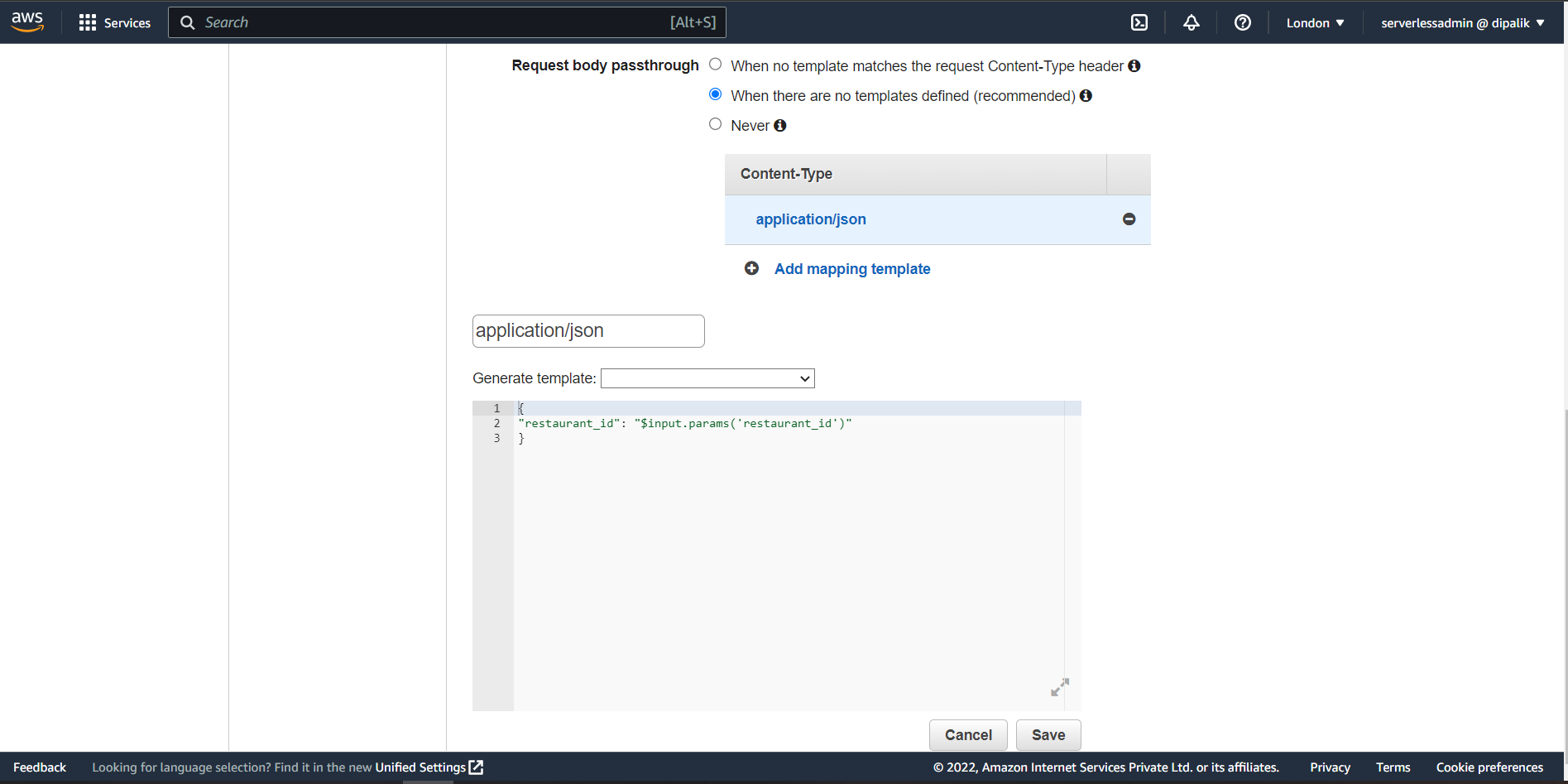
Remember to enable cors for all resources and methods

Click on Create resource to create it:



Define the parameter mapping to map the restaurant\_id element with the api request to Lambda function. Use the following snippet to configure the mapping.

{  
"restaurant\_id": "$input.params('restaurant\_id')"  
}



Now test the API with valid and invalid id value , you should get the response accordingly.

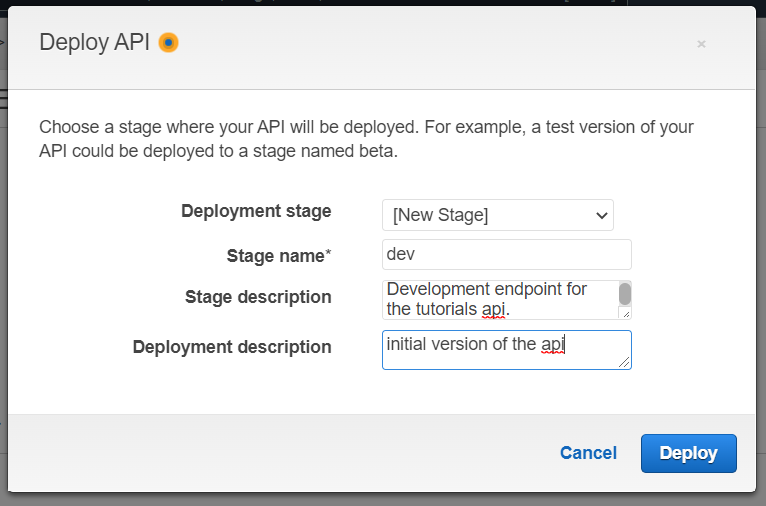
Create all APIs in the same way as above.

Then finally enable CORS on the main resource.

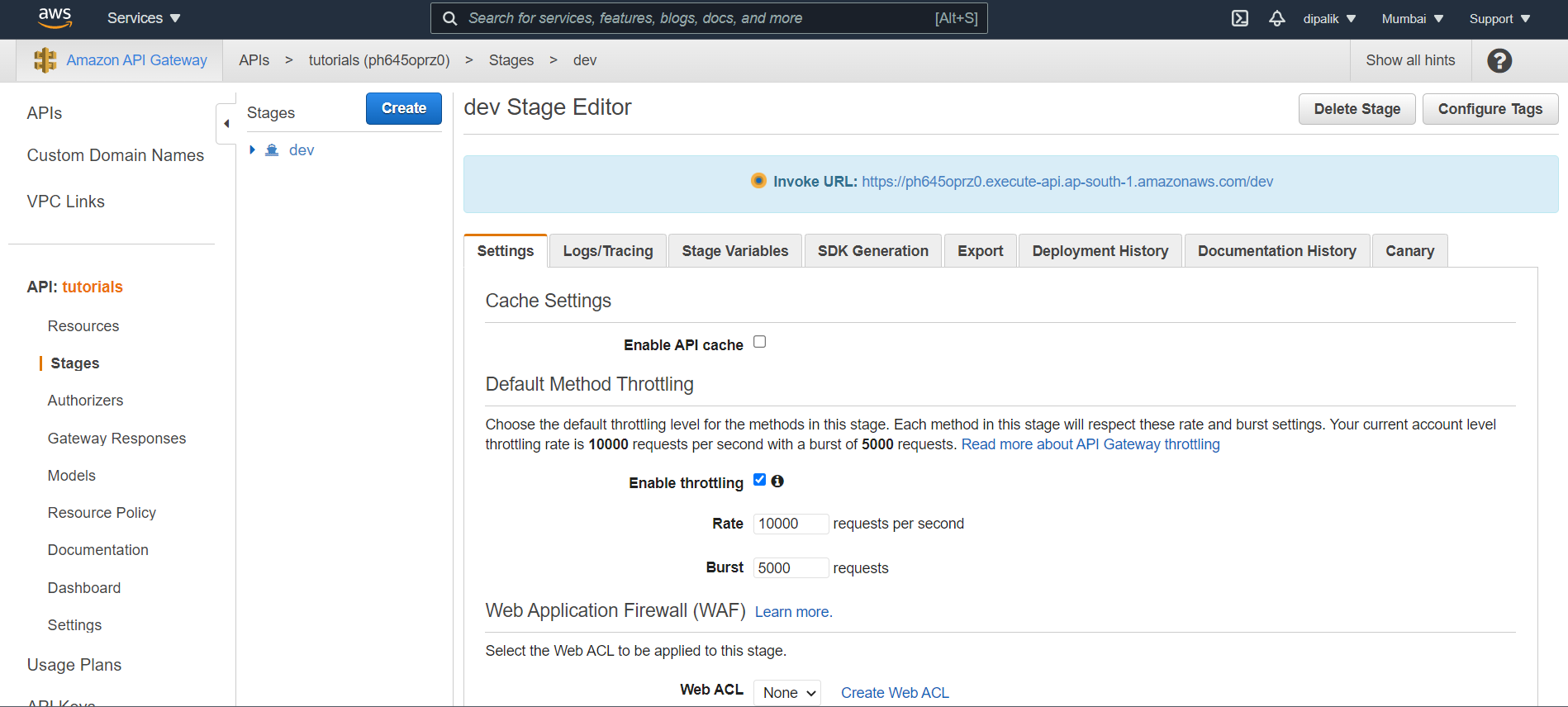
Select the main resource /, click on Actions button and select Enable CORS

Select all the methods DELETE, POST, GET, OPTIONS from the selection.

Now to make the API available on Internet go to Actions and select Deploy API, it will prompt to create a new Stage for Deployment



It will take some time to deploy the API , after successful deployment you will get the API URL to invoke the API



Use POSTMAN tool to call your APIs

