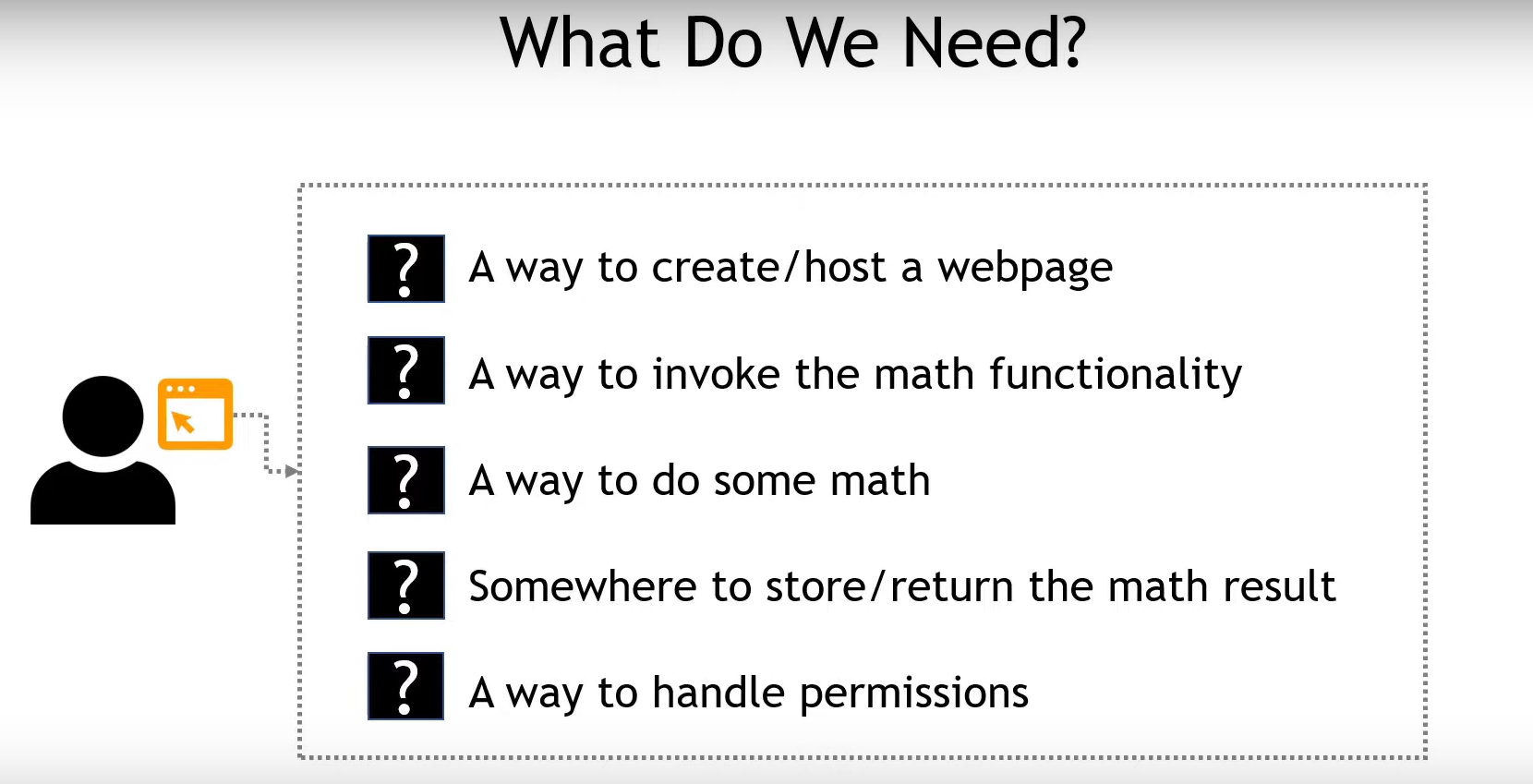
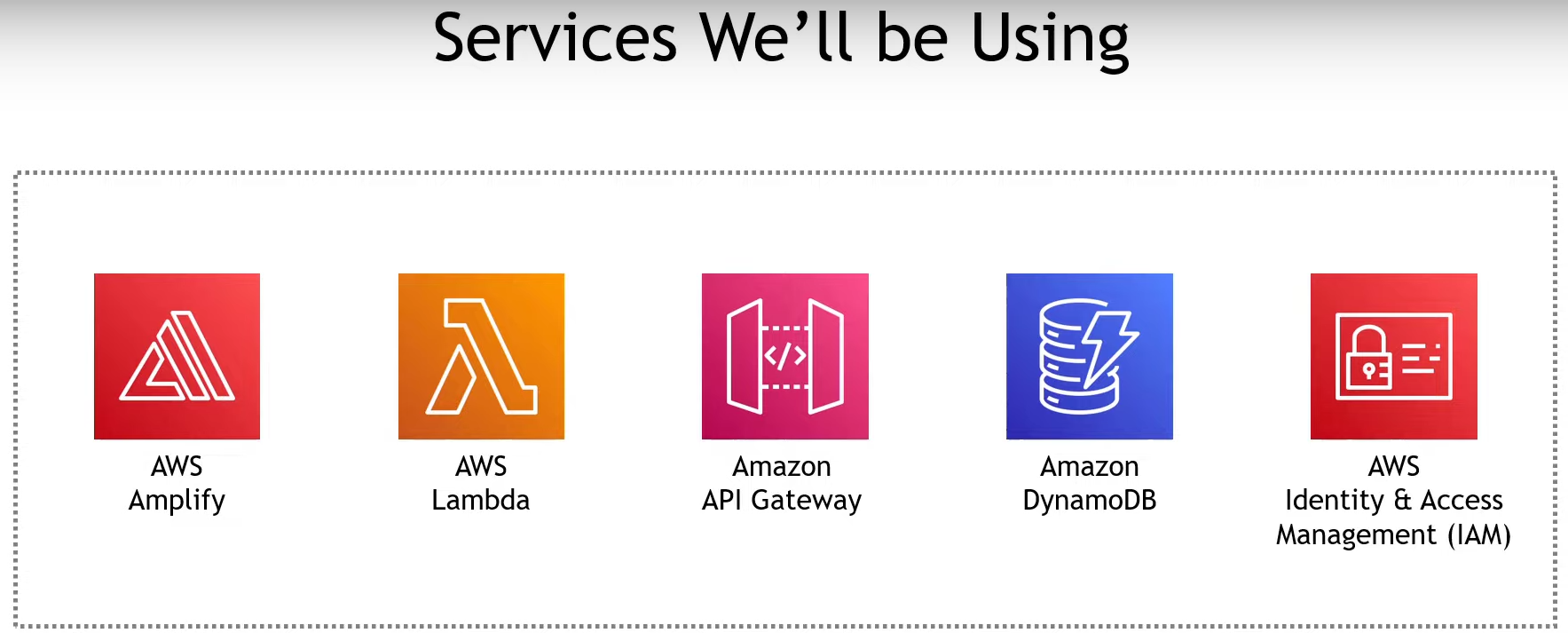
**Deploying End-to-End Serverless AWS Web Application using Amplify & DynamoDB**

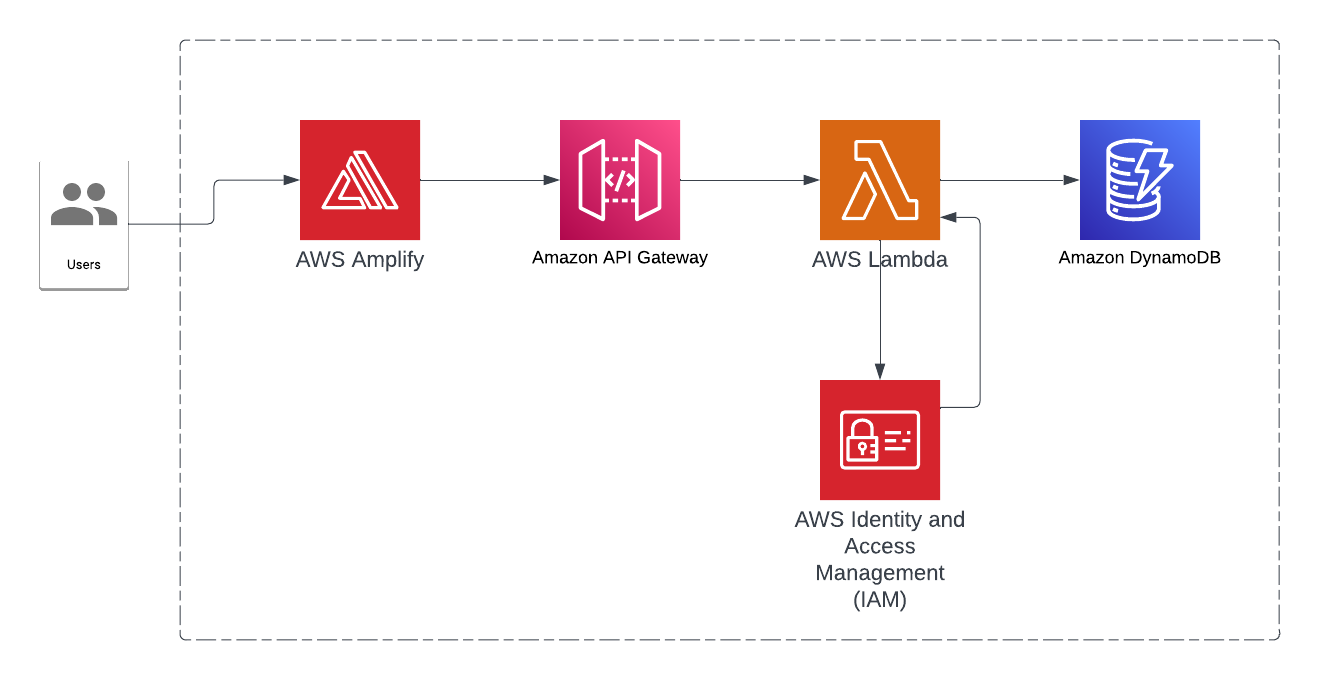
**Objectives:**



**Services we will be using:**

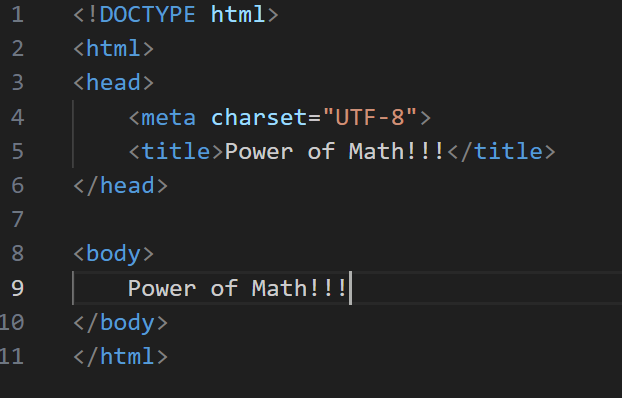


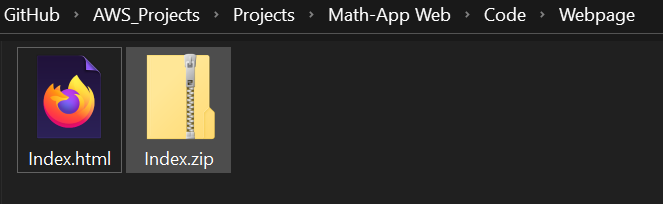
**Architecture of the Project:**

****

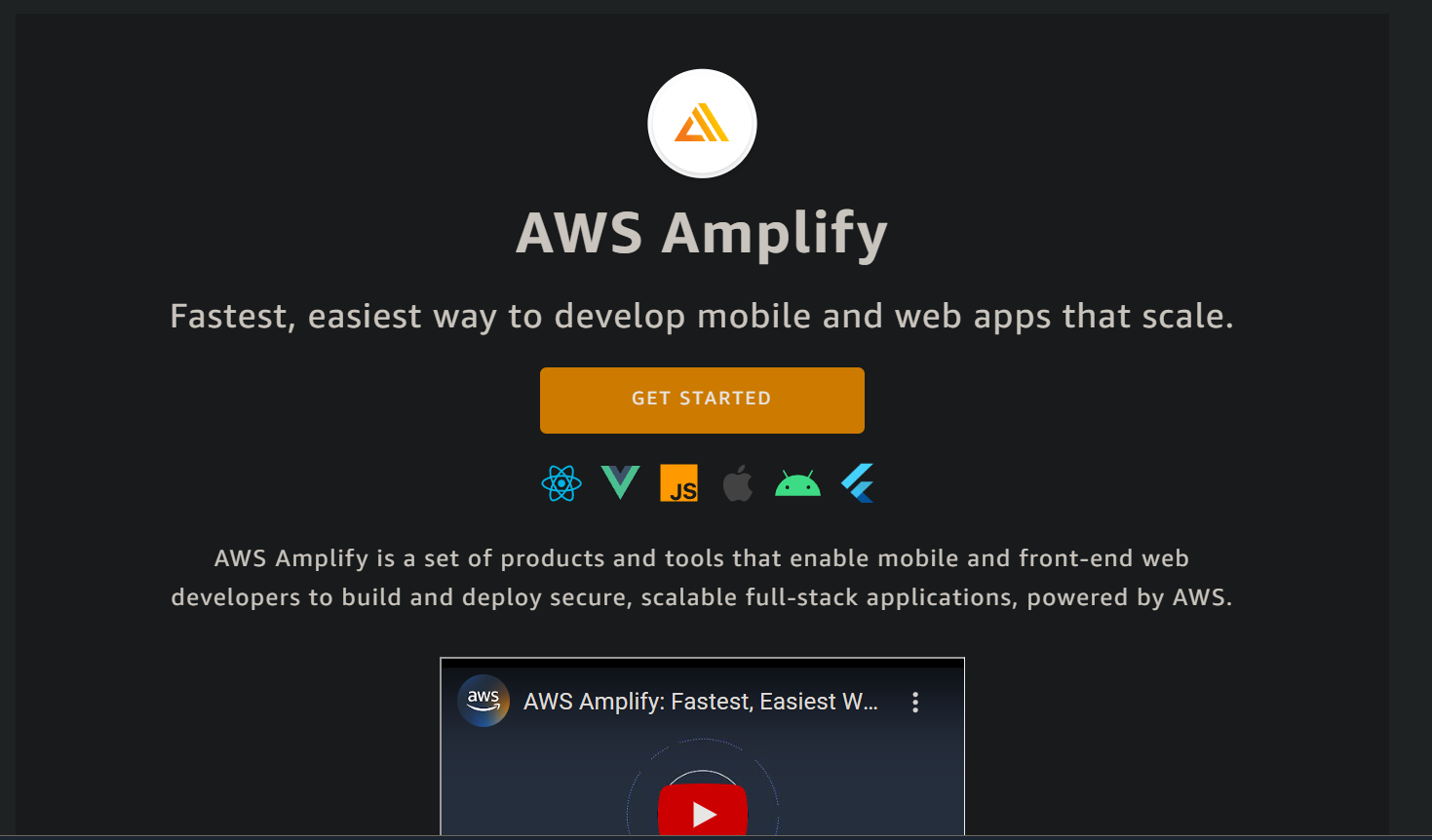
**Step1:** Creating an index.html file for the webpage

Create a index.html file & enter the simple code. Then zip the code (Refer “index-original.html” from code folder).



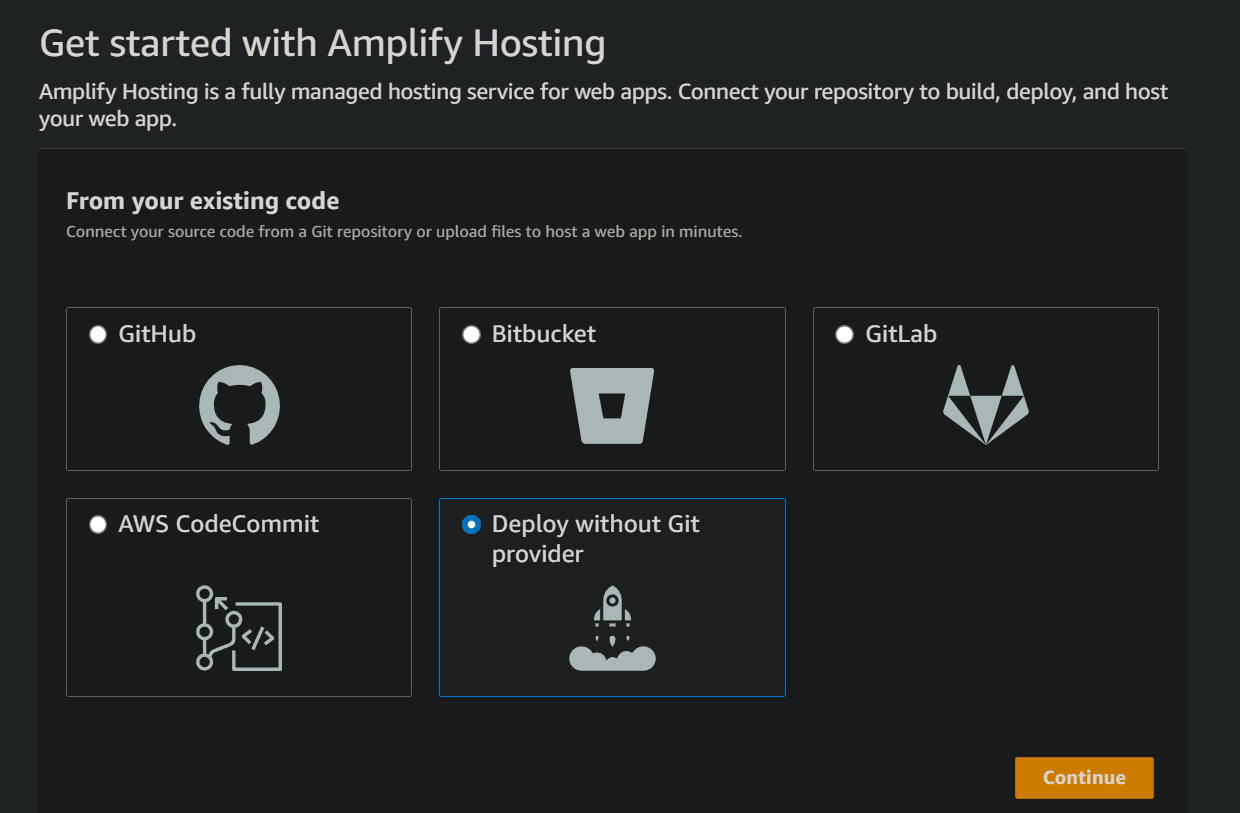


**Step 2:** Use AWS Amplify to deploy the code

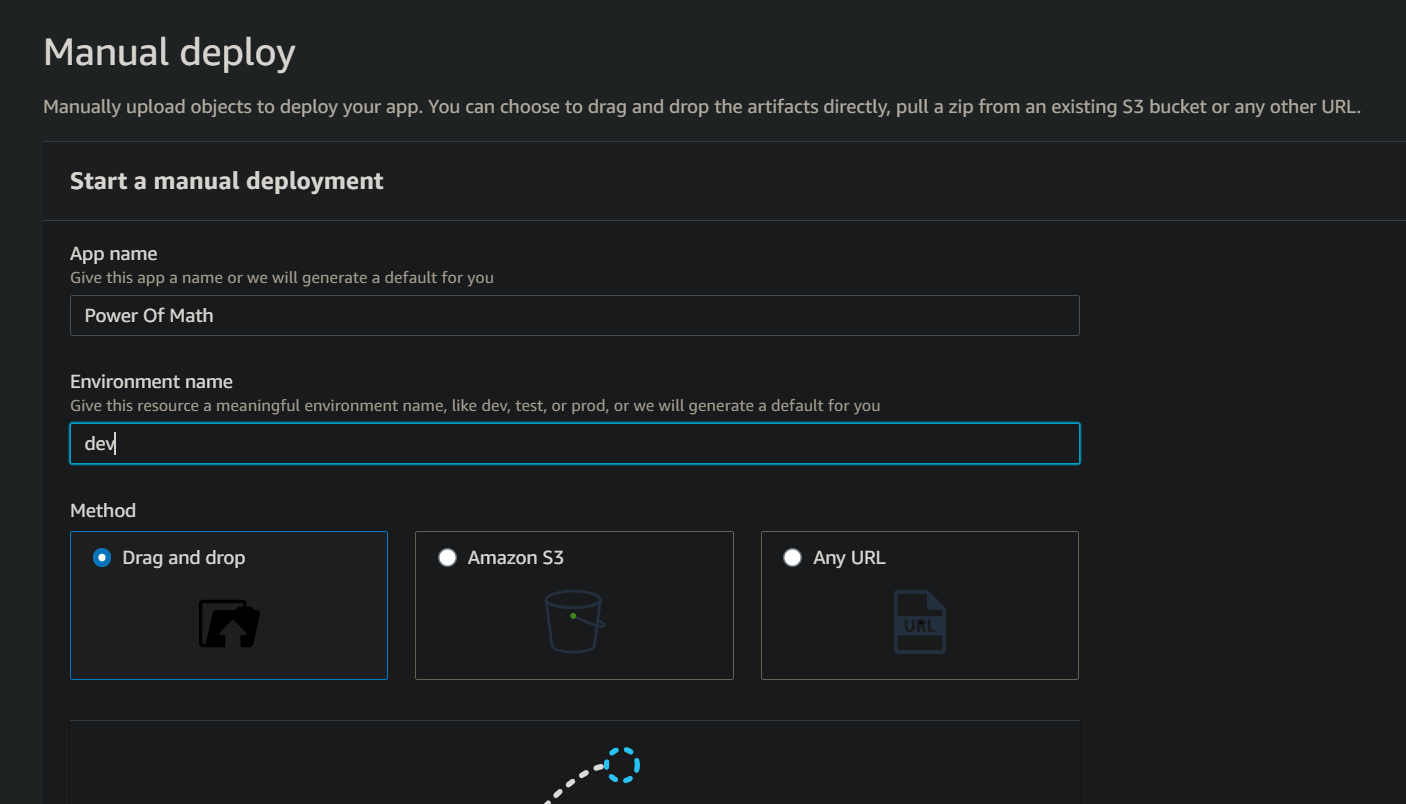


Select Host the web app option.

Then,



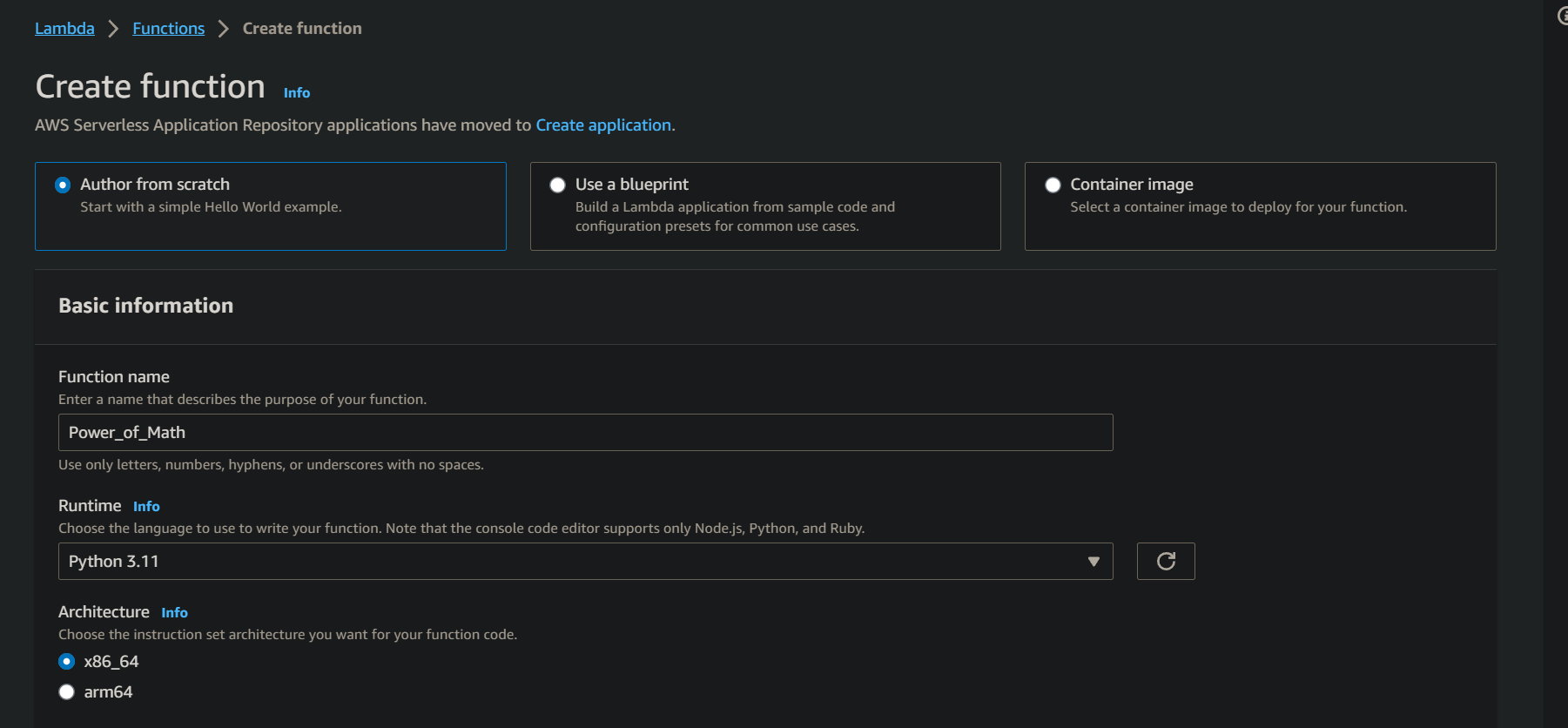
Then,



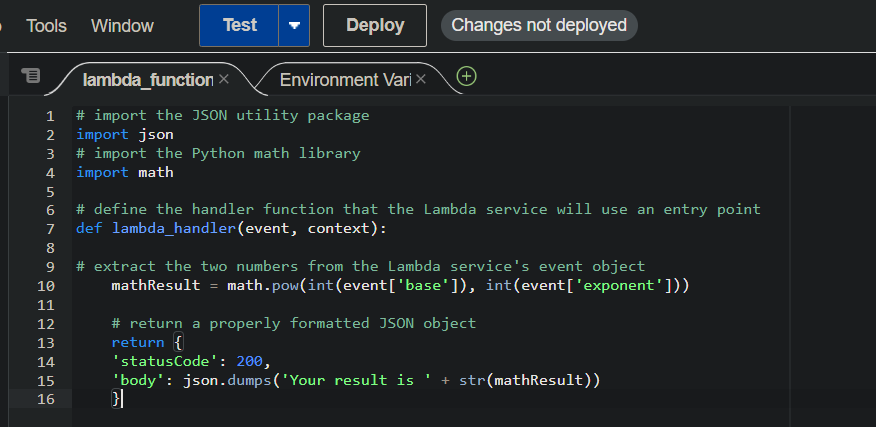
Now the partial frontend hosting of the app is done.

Lets do some maths functions to be triggered upon using Lambda.

**Step 2**: Creating the Lambda function for math calculations backend



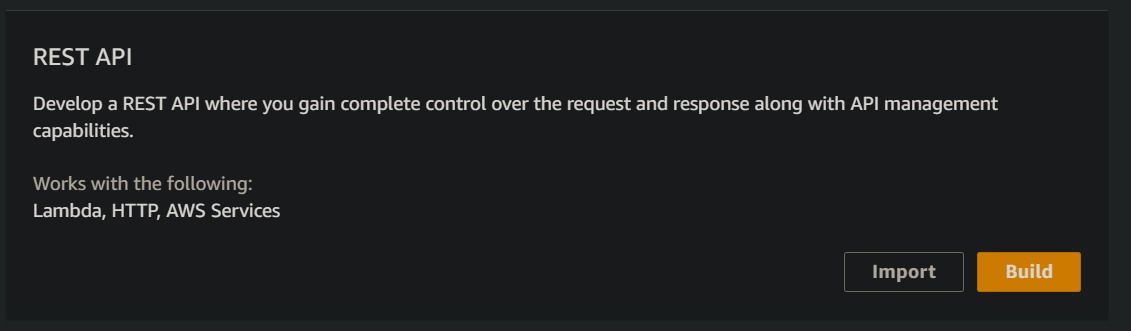
Everything else can be left to default. Then create the function. (Refer “PowerOfMathFunction - Lambda-ORIGINAL.py” from code folder).



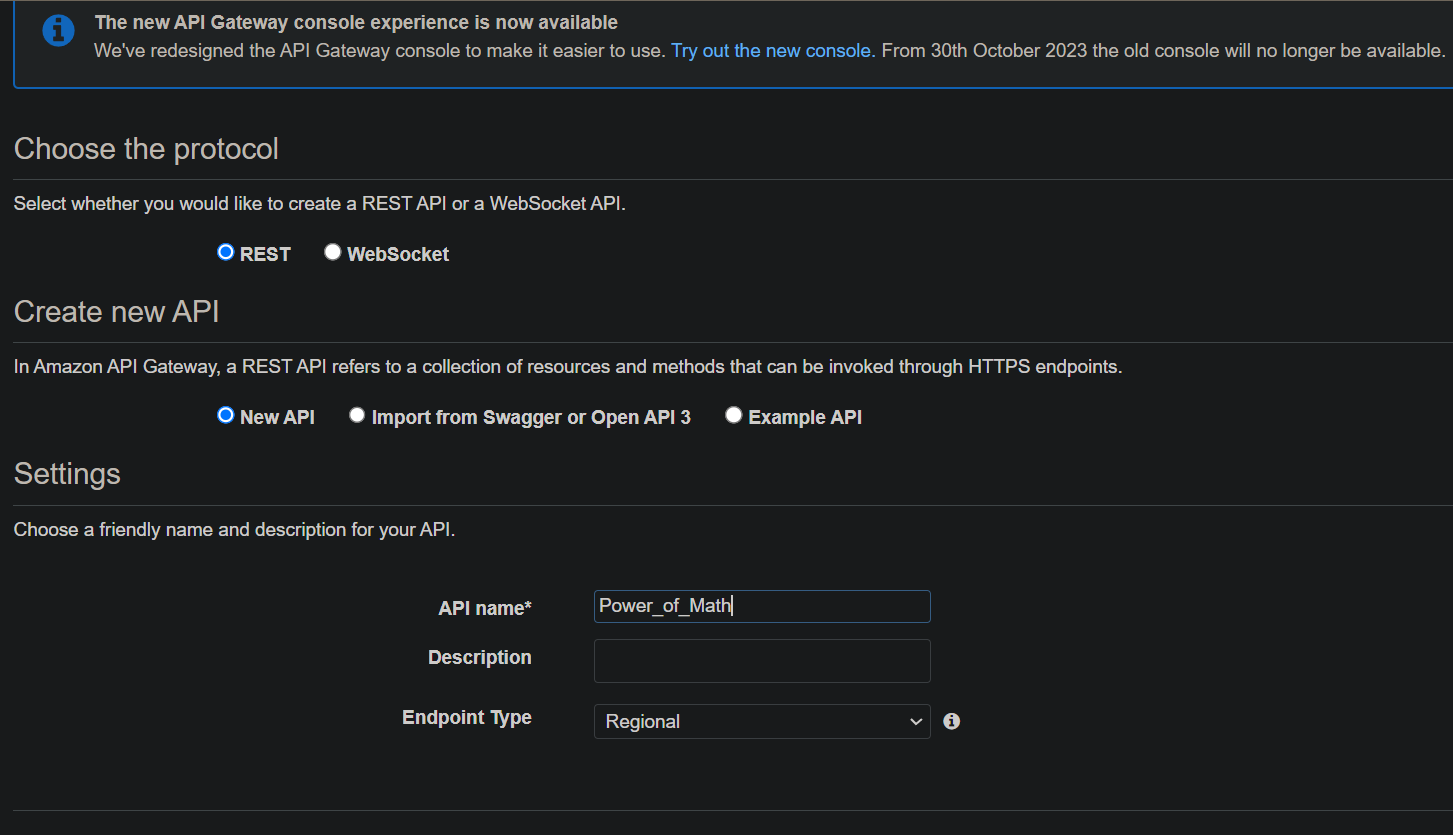
Now use this code & deploy it. Now the Lambda function part is done.

**Step 3**: Now we need a public endpoint/URL to trigger this lambda function when interacting with the website. For this we use AWS API Gateway service.

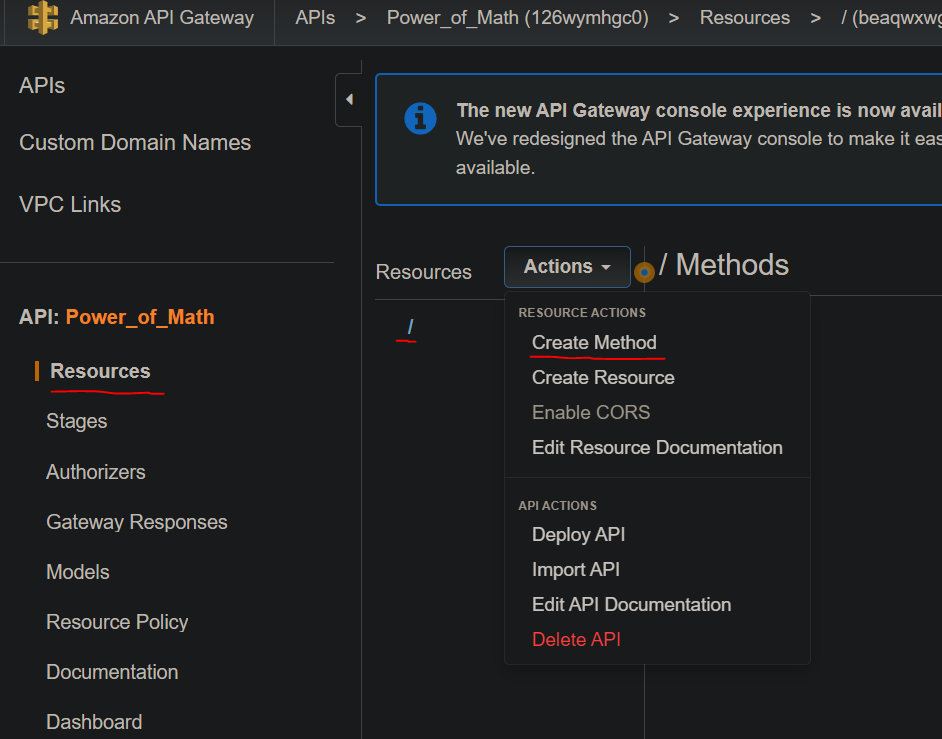
Select REST API option



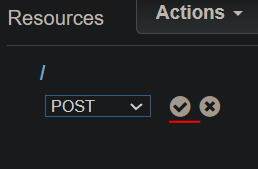
Chose following options



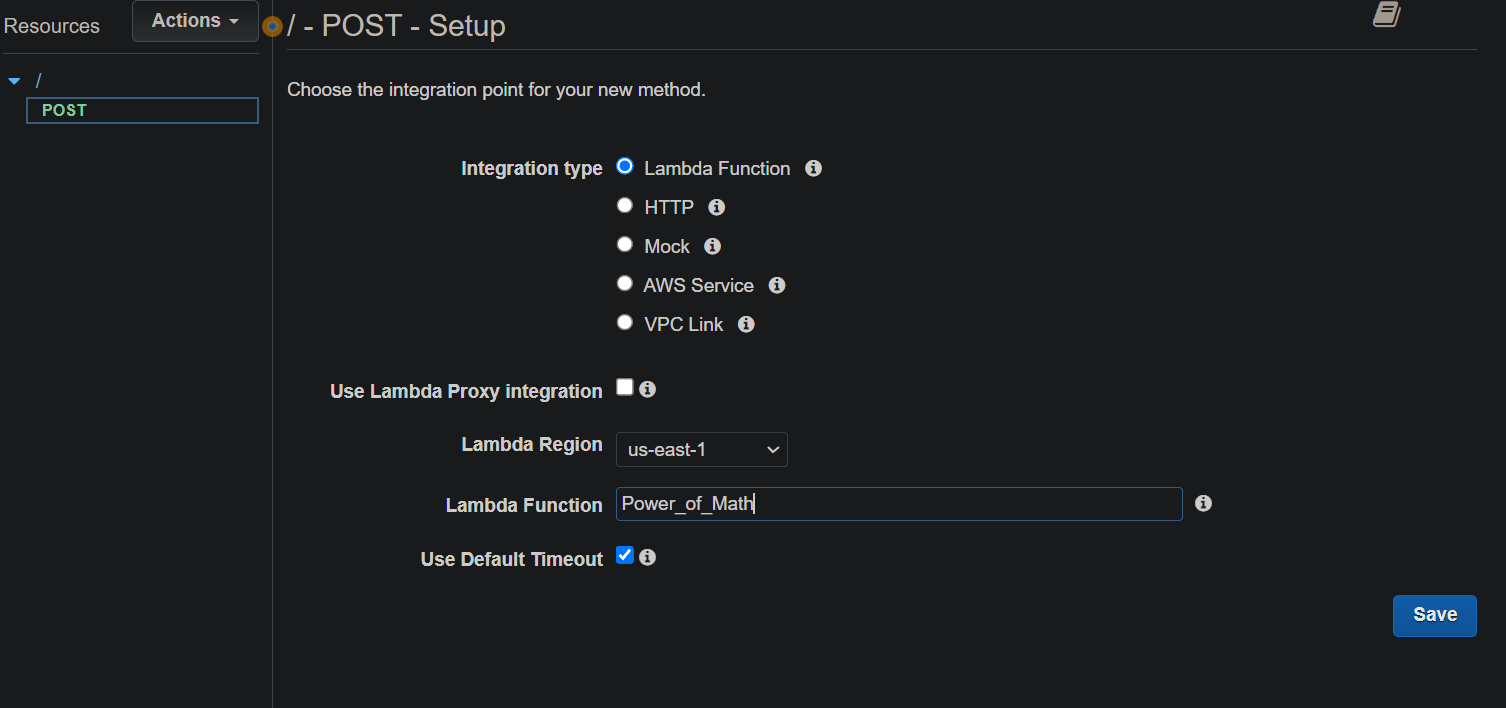
Create method



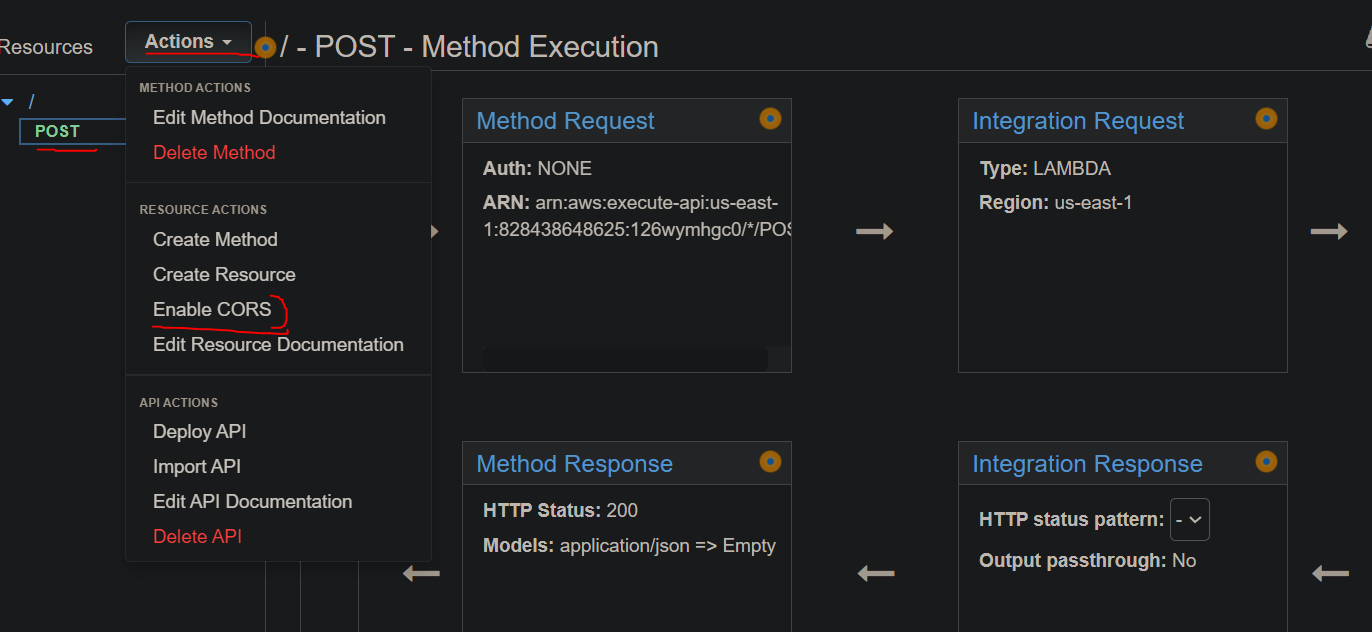
Type of Method – POST . Then select tick mark



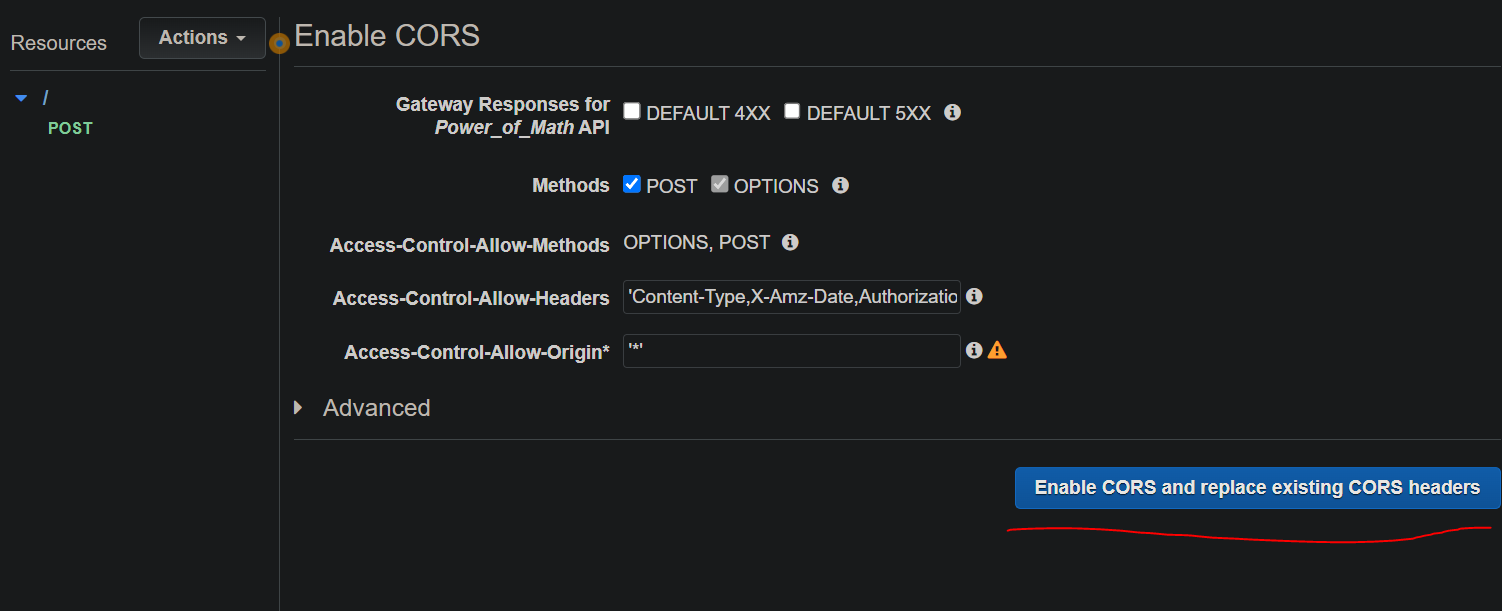
Then,



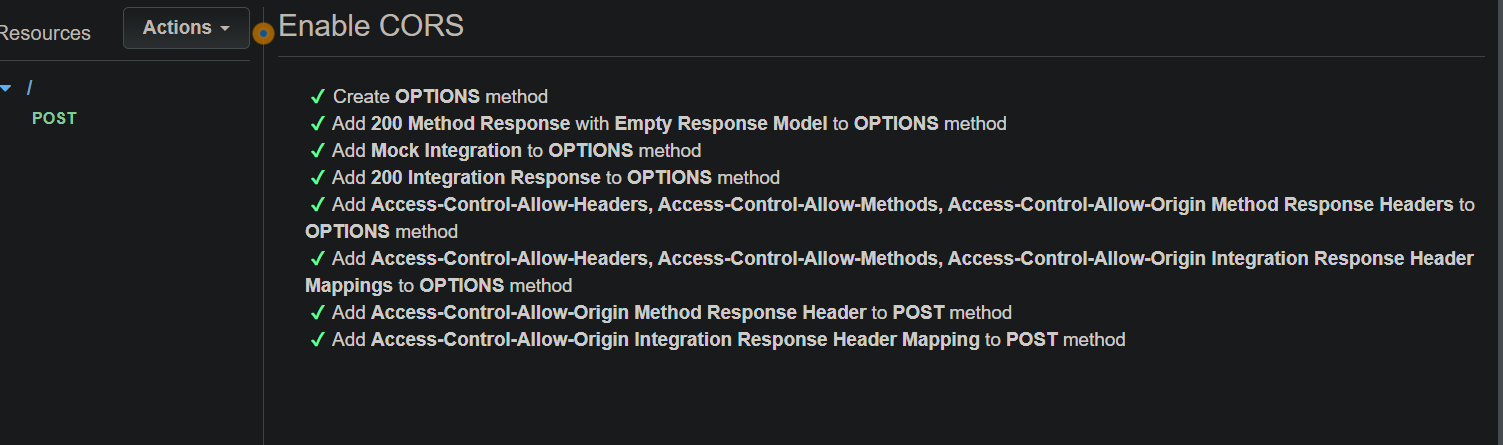
Now we have to make sure we enable **CORS** (Cross Origin Region Sharing). This is to make sure services across domain can work with one another.



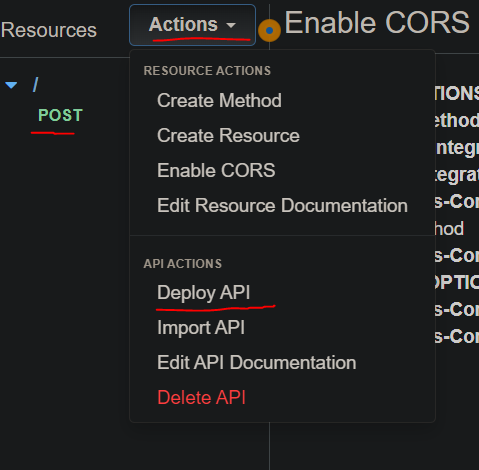
Now,



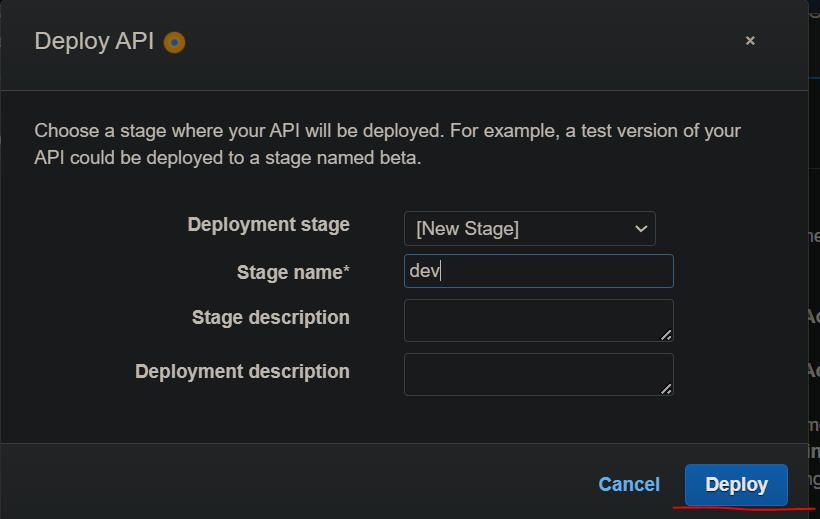
This is the outcome.



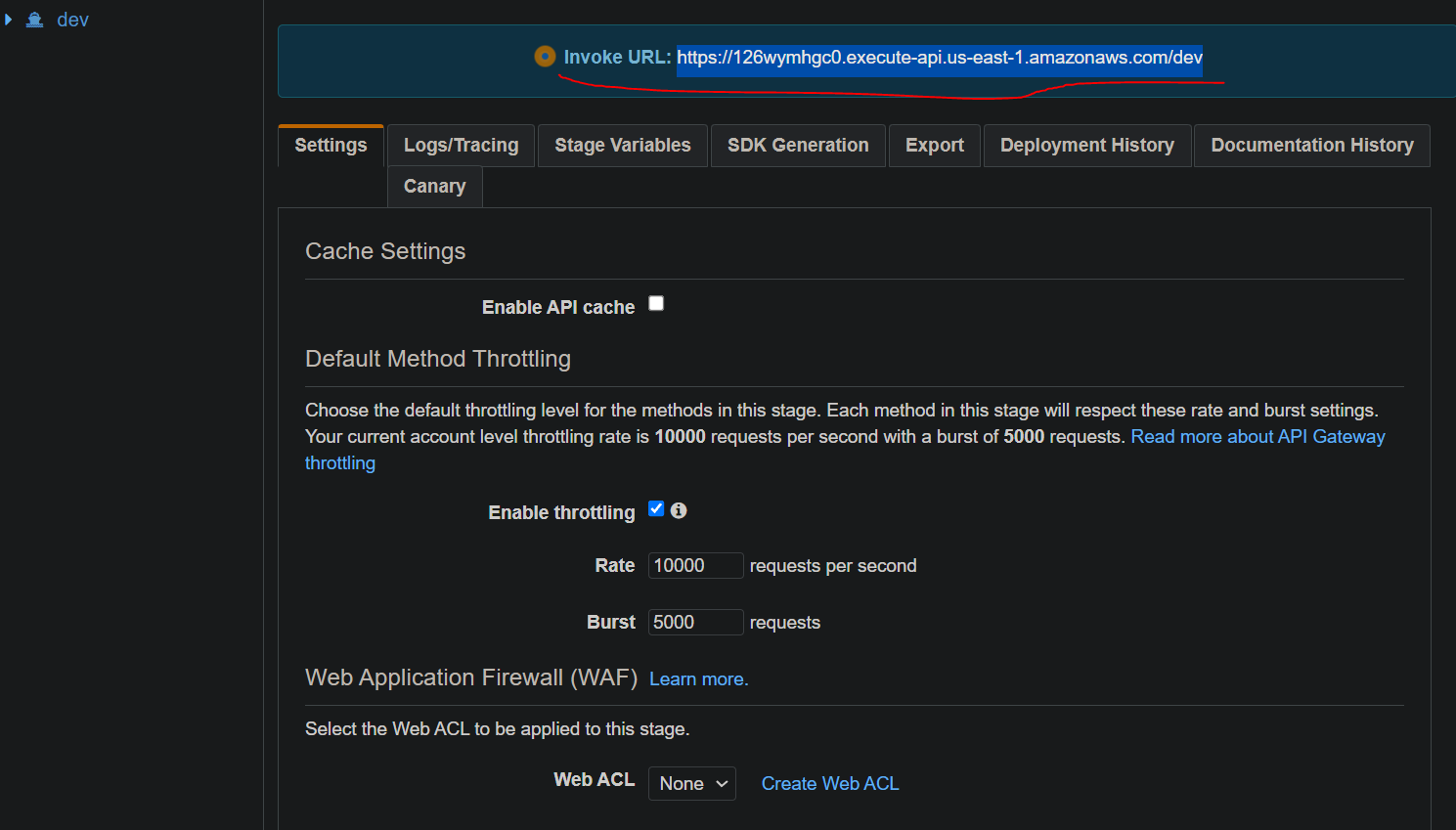
Now we have to deploy our API.



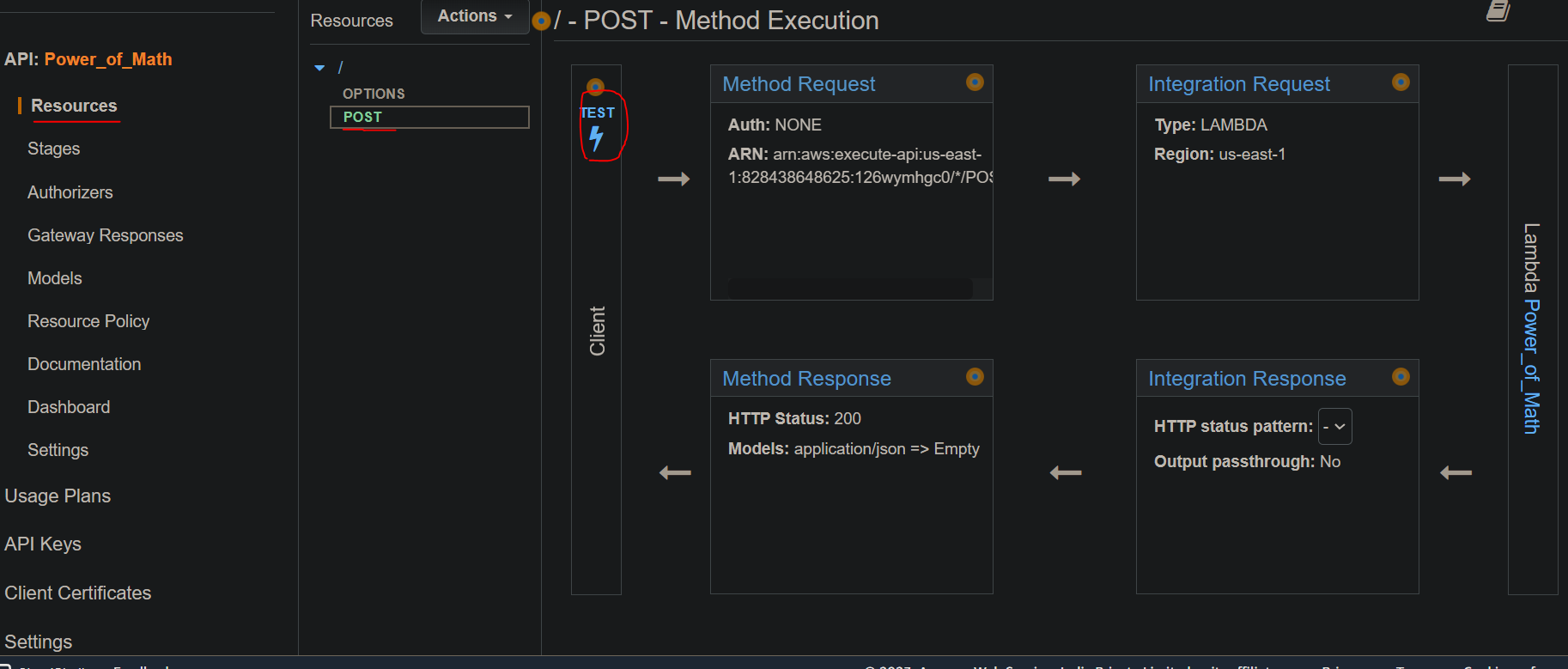
Then,



Now copy & keep the URL generated for you somewhere safe:

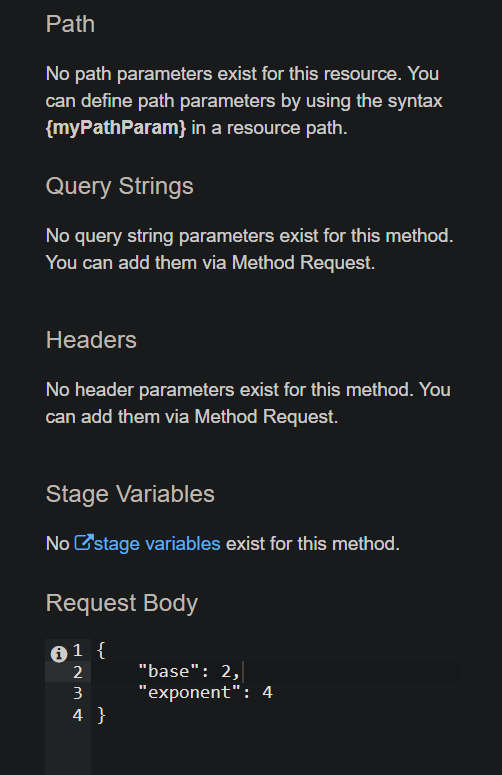
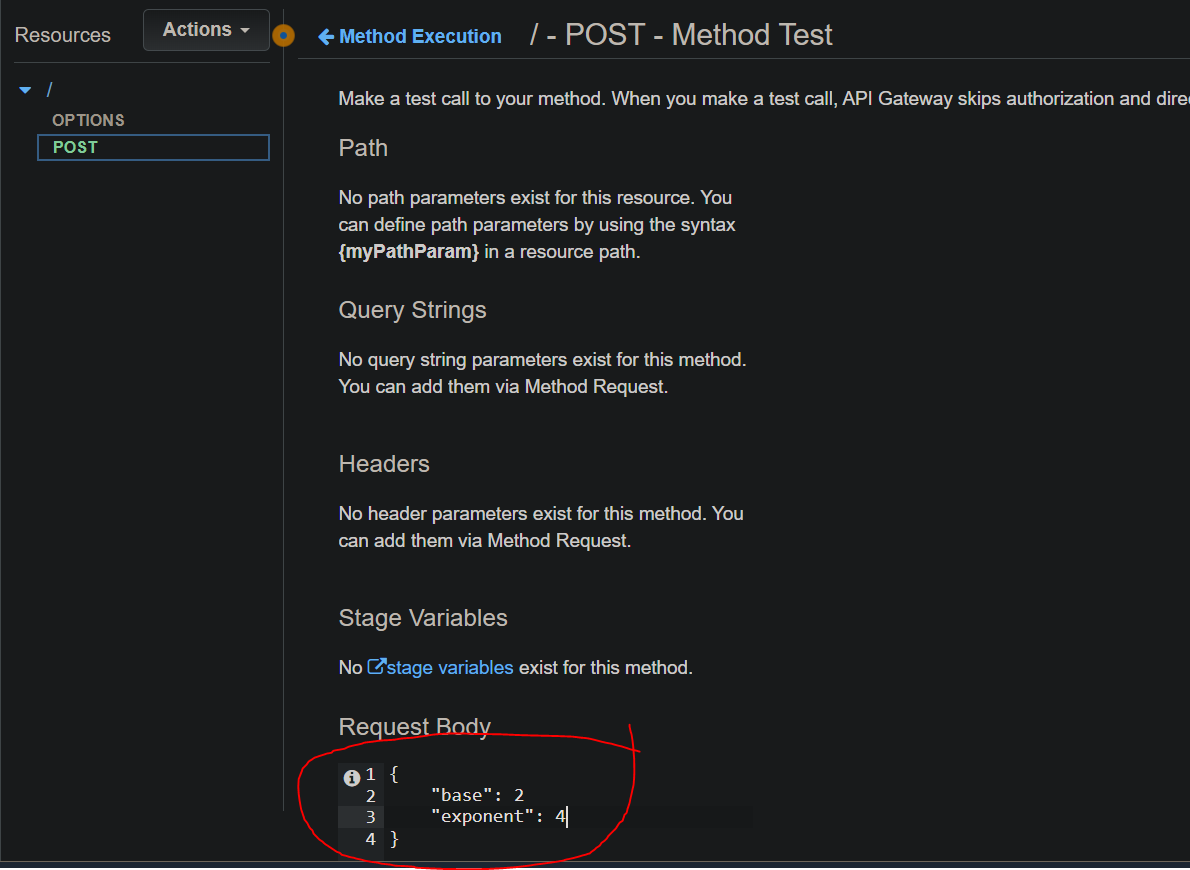


Now lets validate this API. Come to this path. Then click on Test Icon.

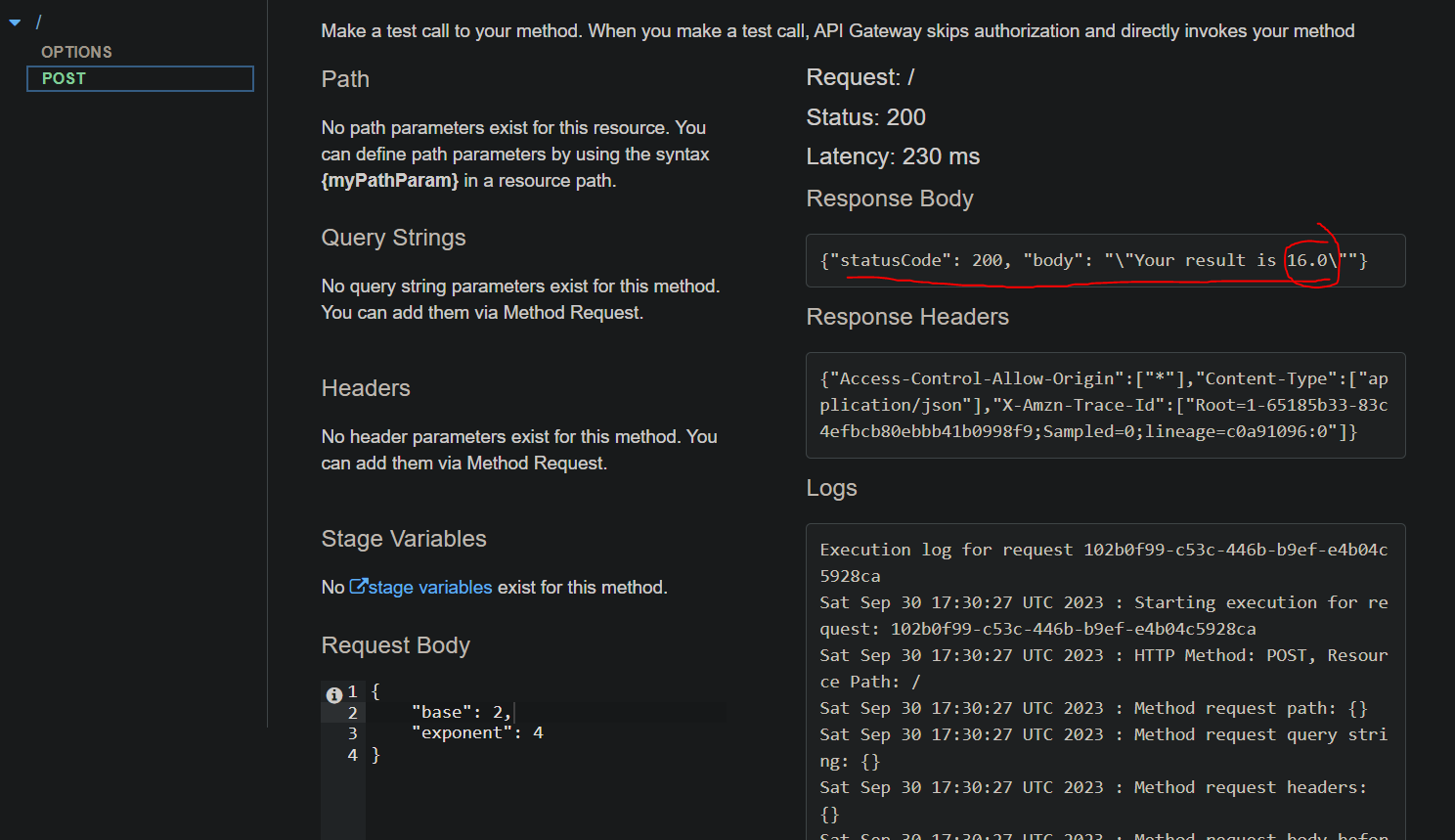


Now what we are doing here is to test if the API is working by passing the parameters that we have set in Lamda function & giving them test values. If the API & Lambda is working as intended, they will give the desired Output.

We write the test values in the request body (use the lambda parameters name) . Then click on test at the bottom.



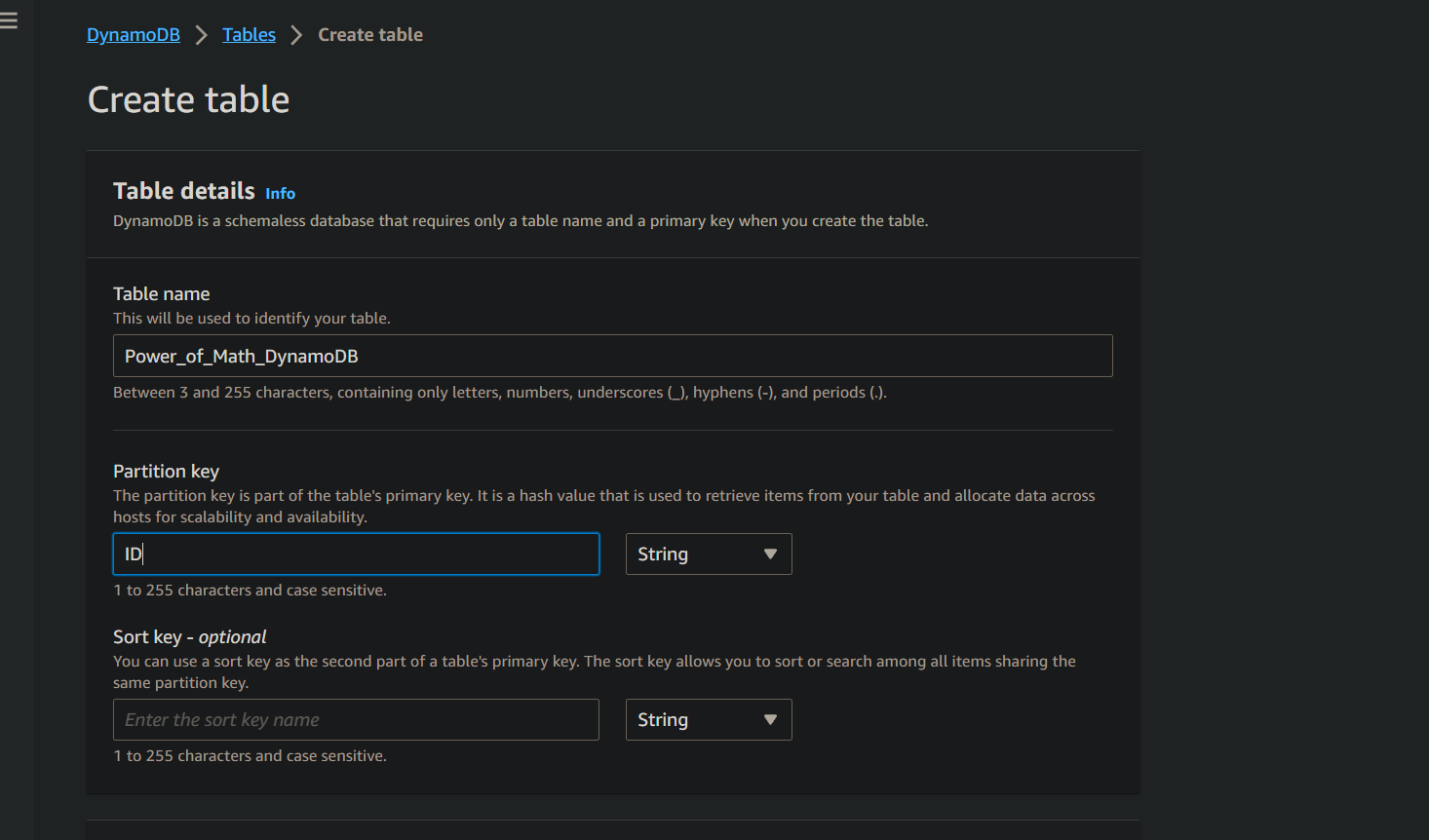
Result is successful. Got the desired output.



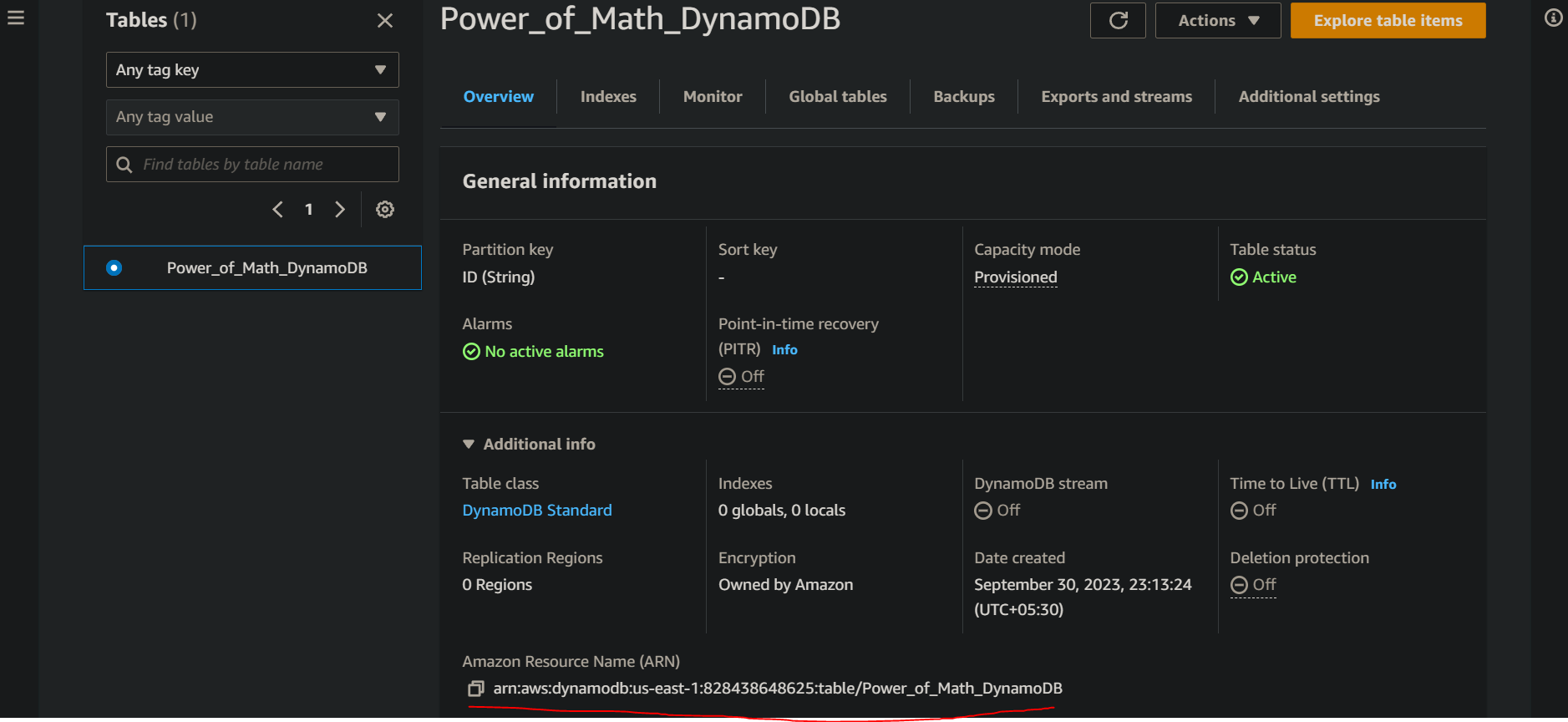
Now the AWS API Gateway part is completed.

**Step 4**:- Now we have to store the returned values somewhere. For this we will use DynamoDB (NoSQL) to store our values.

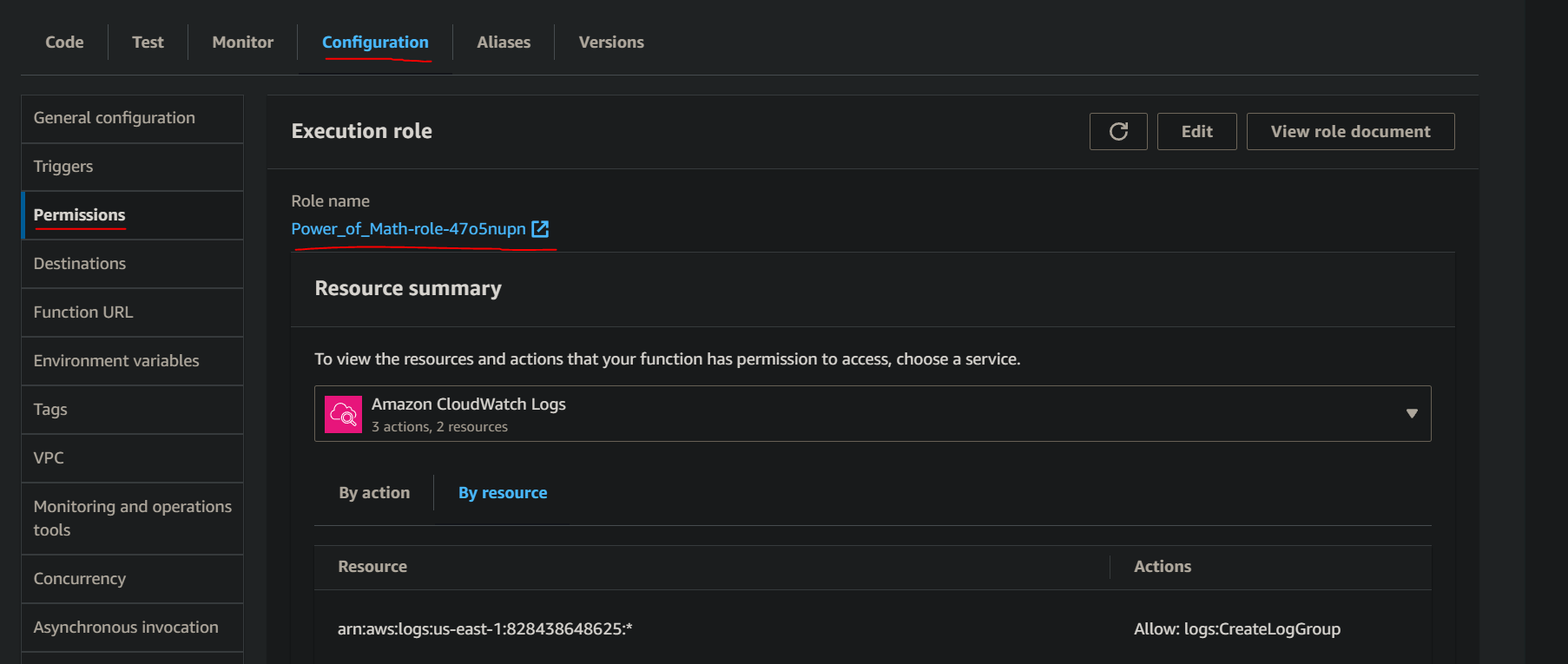
Leave all the other values to Default. Create the table.



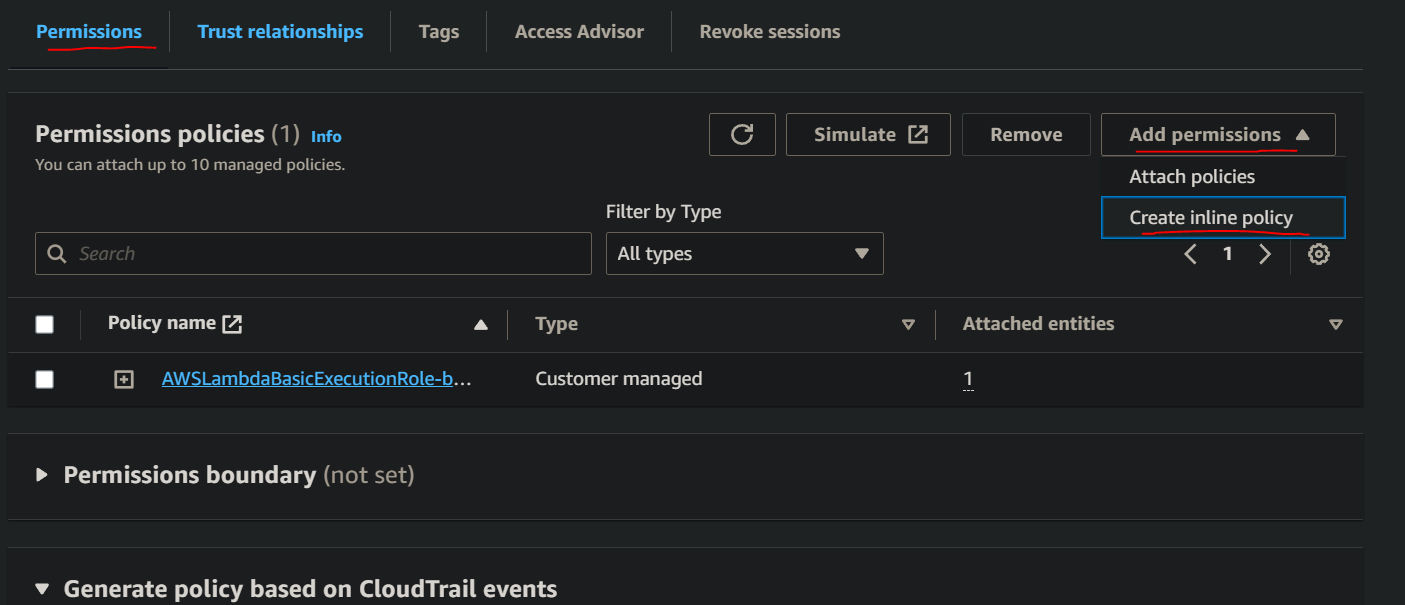
Copy the DynamoDB arn & store it somewhere



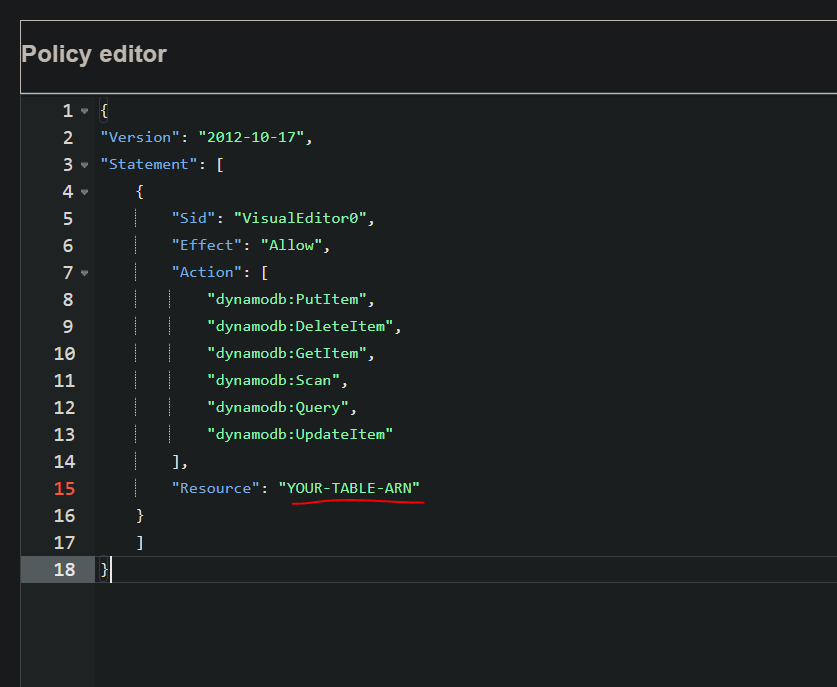
Now we need to have the lambda that we have created to have the permission to write values to this DynamoDB. Go to the lambda section & do this. Click on the Rolename.



Now goto the create inline policy & copy the following JSON code to create DynamoDB Permissions.

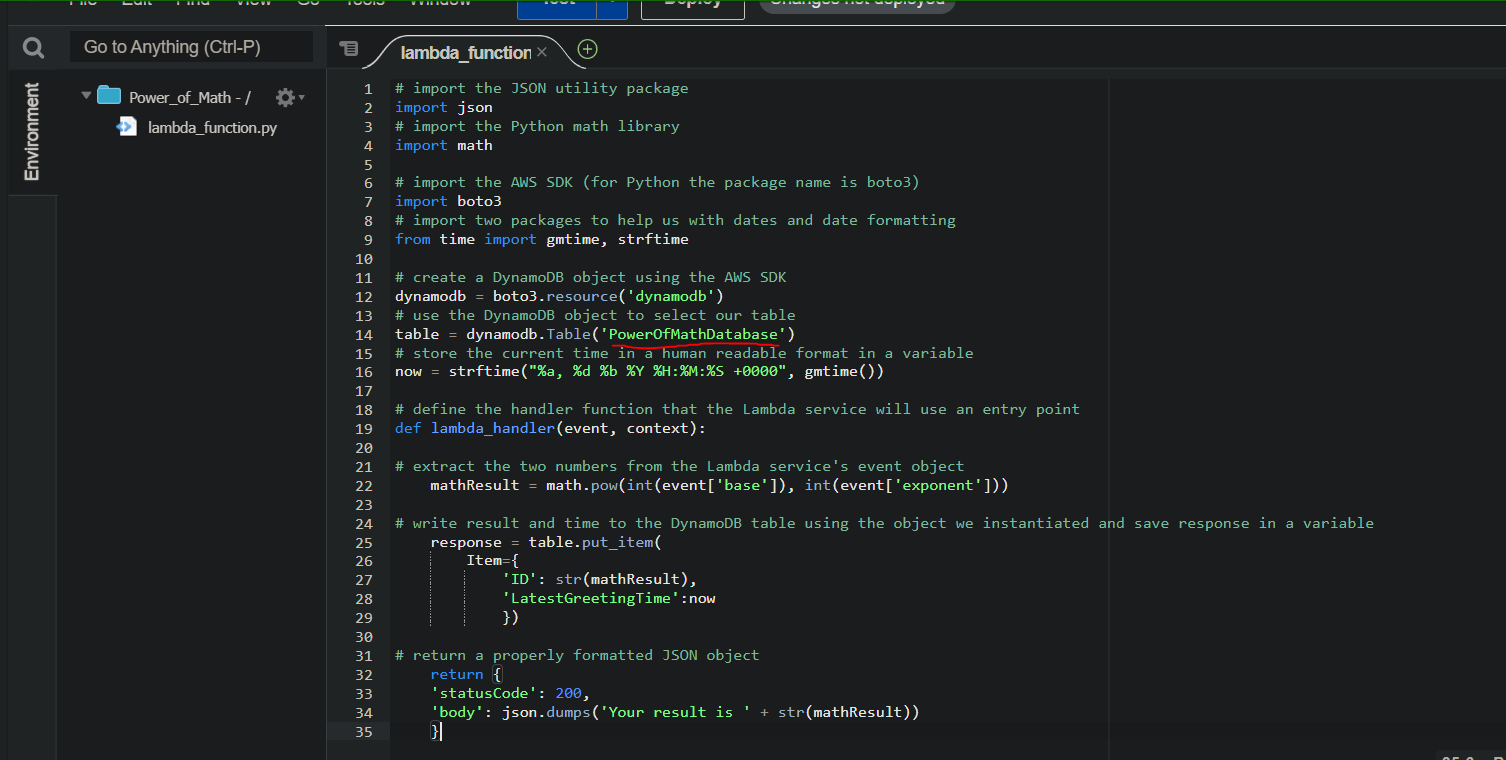


This is the code. Replace your-table-arn with the actual table arn. Basically this policy allows actions of the following things – Put, delete,get,scan,query,update. (Refer “Execution Role Policy JSON.json” from the code folder).



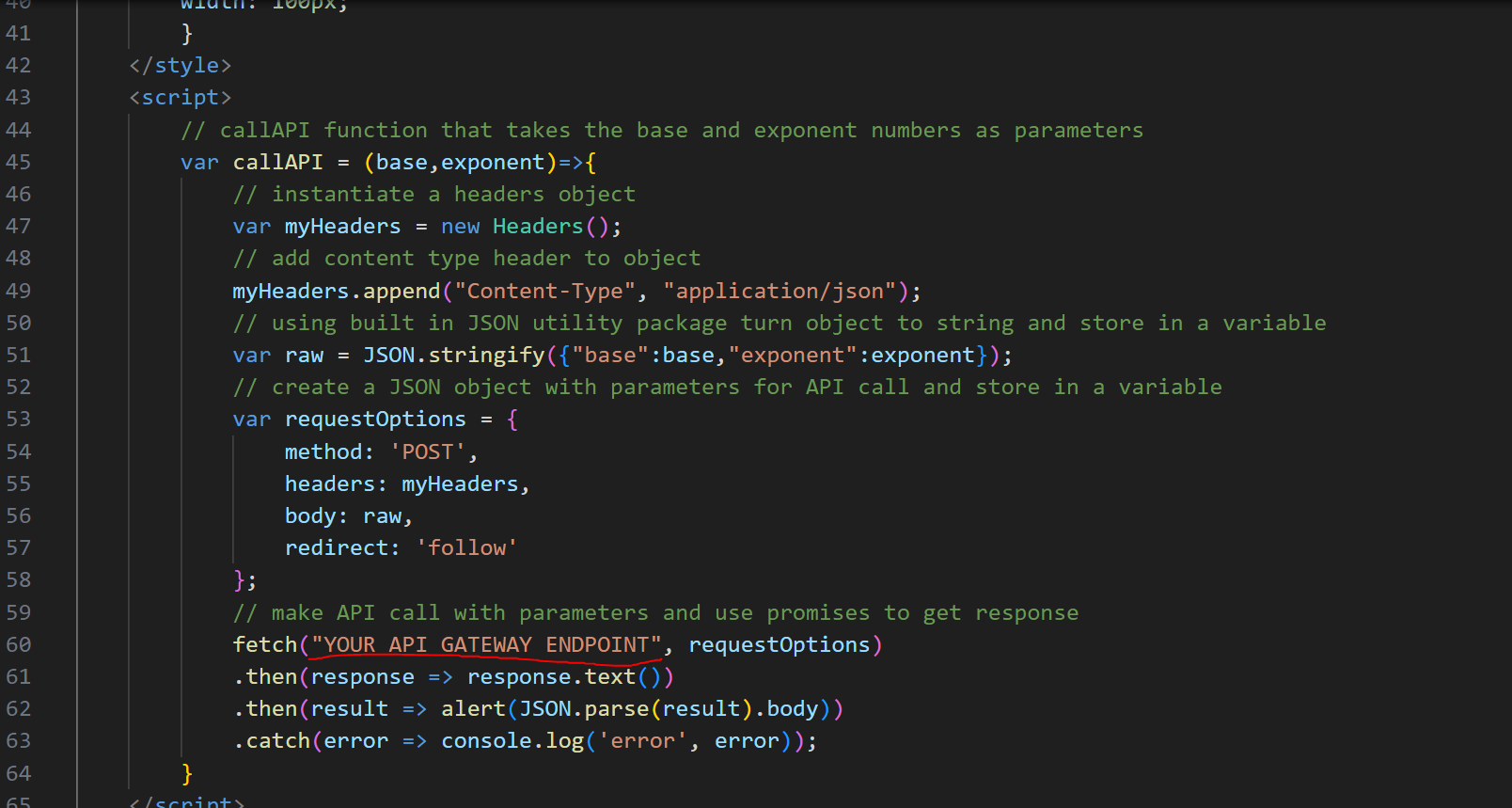
Permissions thing done. Now we have to update the lambda function to actually go write to the database.

**Step 5**: - Updating the lambda function. Change the table variable to your table name. Next save the code & deploy. (Refer “PowerOfMathFunction - Lambda-FINAL.py” from code folder).

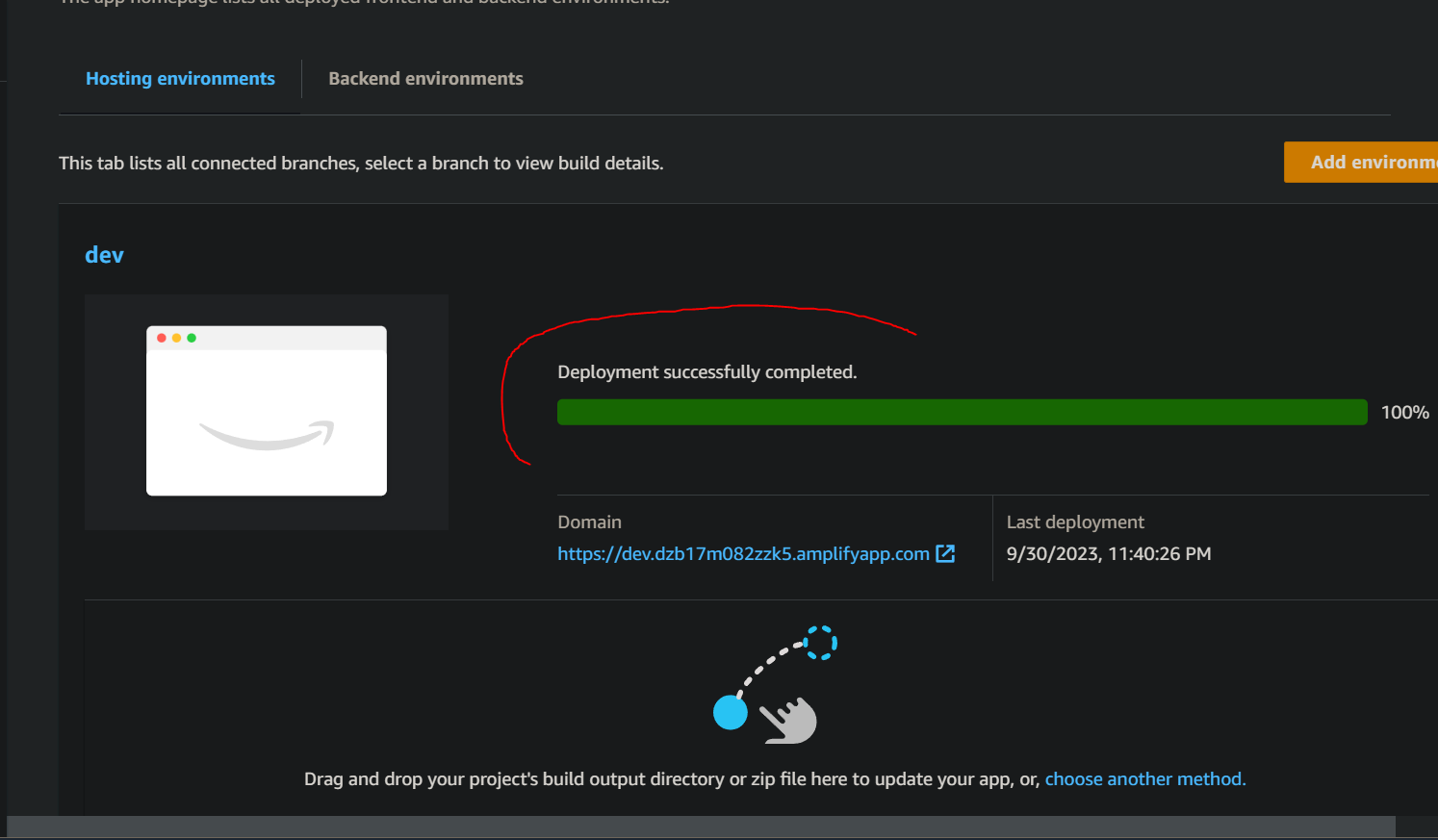


**Step 6**:- Now the final connector. We have to connect our index.html (UI) to the backend process that we have created.

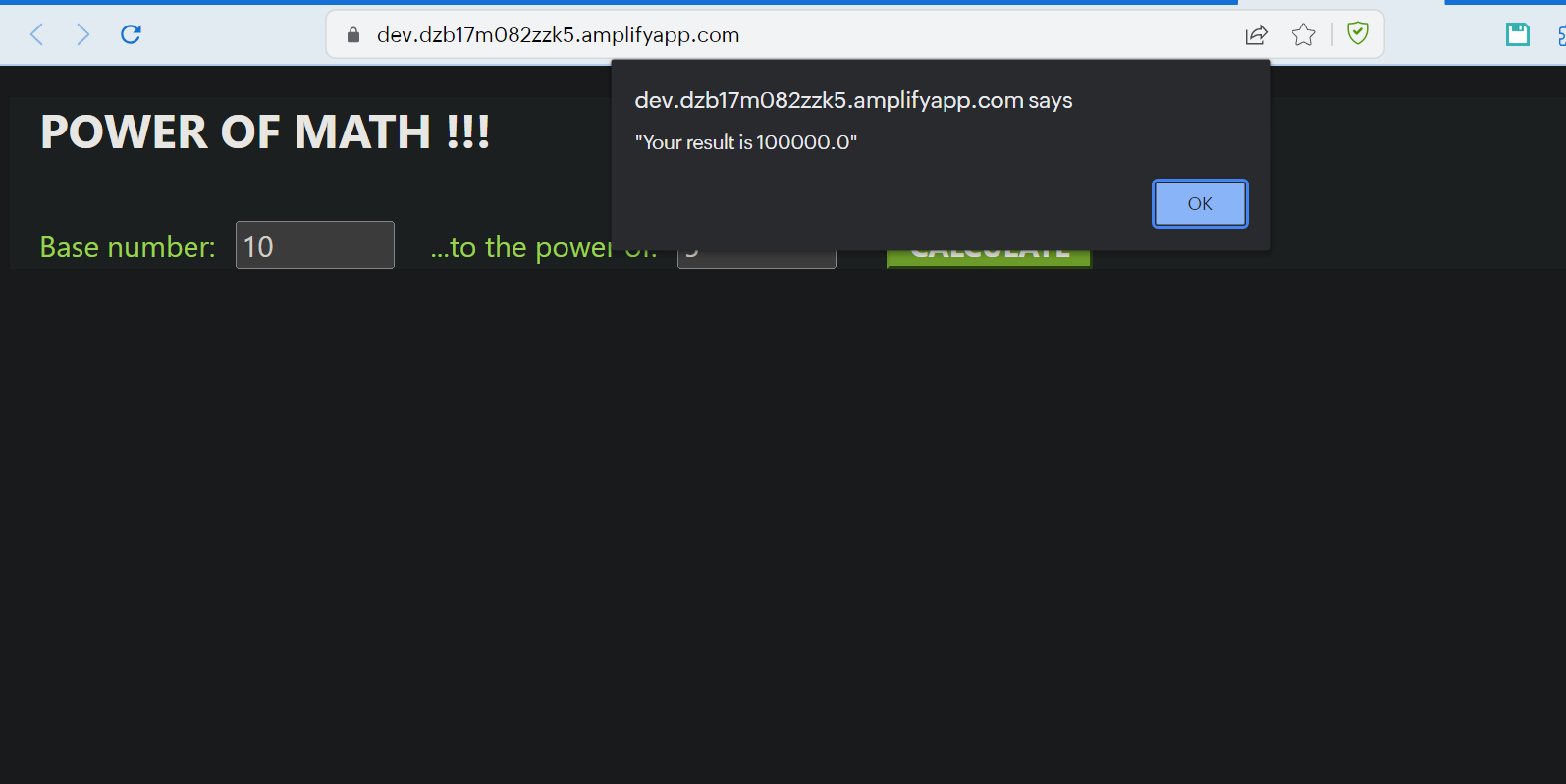
In the final html code, replace this with your actual API arn that we have saved earlier. (Refer “index-original.html” from code folder).



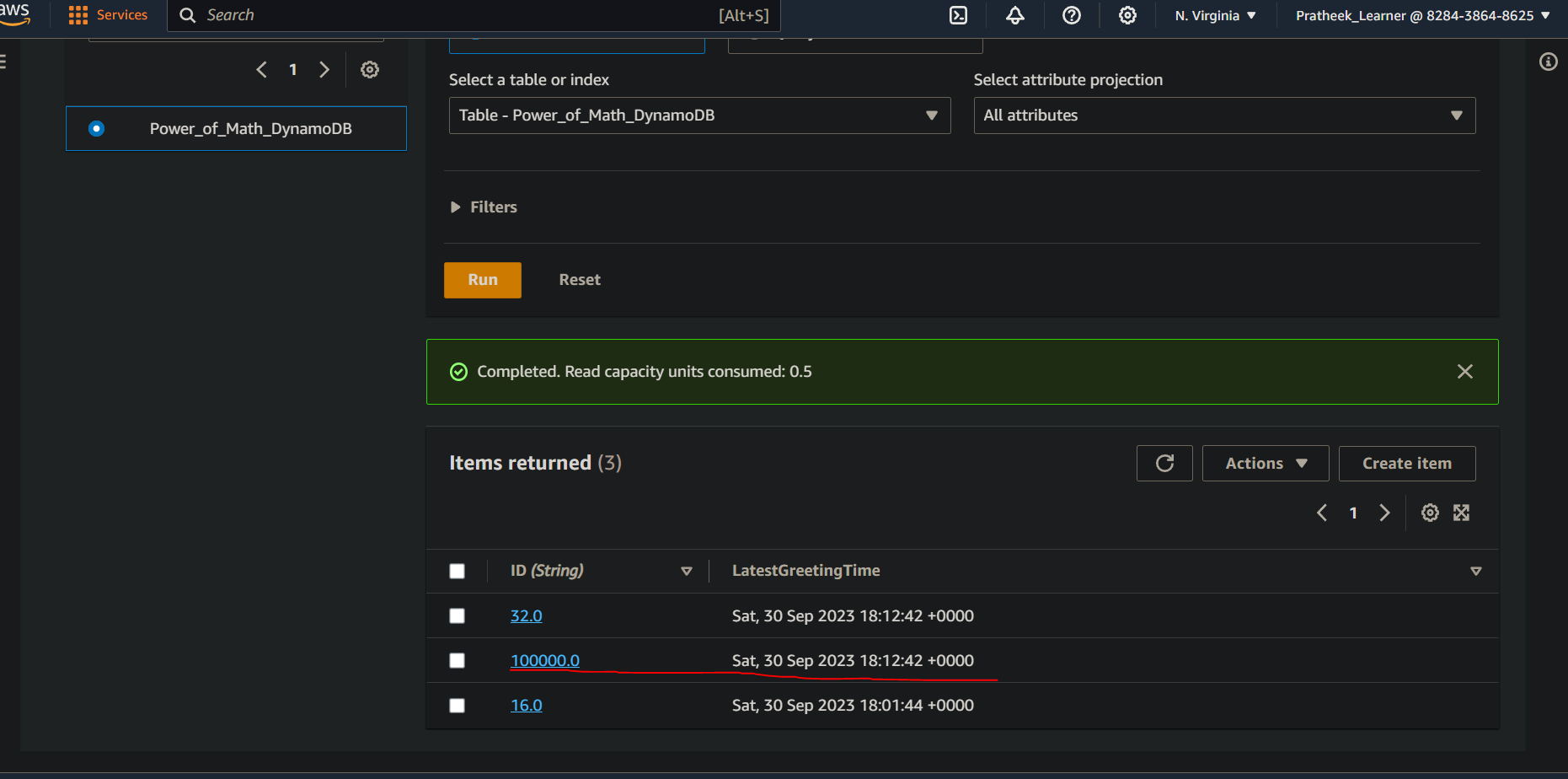
After making the changes, once again create a zip file out of your new index file & redeploy it to Amplify just like before.



Click on the domain & see the results.



Lets check the DynamoDB



Its working. The project has been successfully completed & working as intended.

Resources to clean:

