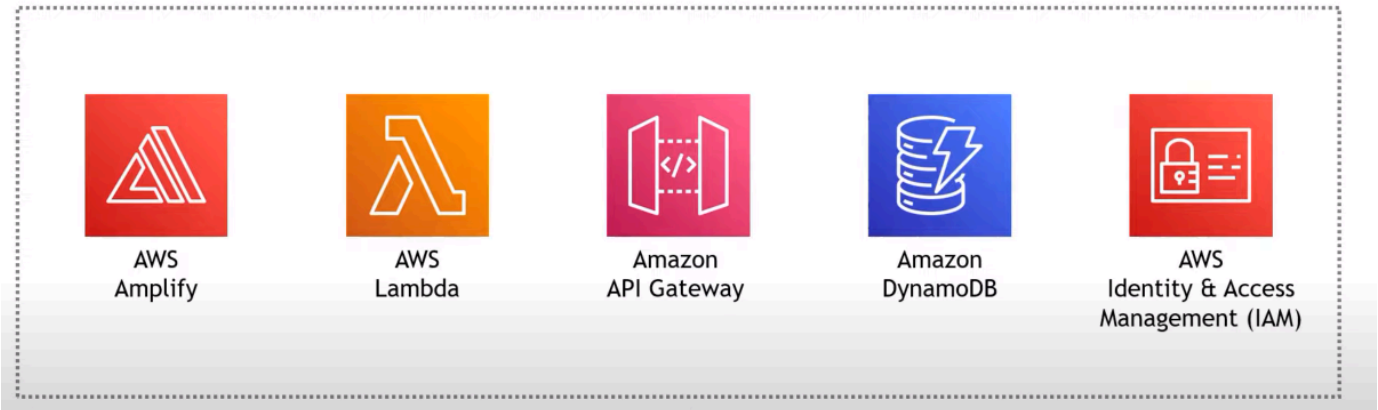
AWS Web Application using 5 services in AWS

AWS Project - Architect and Build an End-to-End AWS Web Application from Scratch, Step by Step
https://www.youtube.com/watch?v=7m_q1ldzw0U



In this demo, we used 5 services in AWS. We created a simple website calculator to calculate the exponent of any number. We use the following services in AWS:

1. AWS Amplify
2. AWS Lambda
3. Amazon API Gateway
4. Amazon DynamoDB
5. AWS Identity & Access Management (IAM)

Why did we pick this mini project?

We picked this mini calculator website because it's using five different AWS services, each of which played a crucial role in ensuring that the application functioned smoothly and efficiently. These services include AWS Amplify, AWS Lambda, AWS API Gateway, Amazon DynamoDB, and AWS Identity & Access Management.

Out of these five services, AWS API Gateway and AWS Lambda were particularly important for the successful development of the calculator. AWS API Gateway provided a secure and scalable way for the calculator to communicate with other services, acting as a bridge between the frontend and backend. AWS Lambda, on the other hand, served as the compute service that performed the calculations requested by users, ensuring that the calculator functioned efficiently.

Here's how each service contributed to the end result of this mini calculator project.

- AWS Amplify was used to develop the frontend of the calculator, which provided a user-friendly interface for users to input their numbers and receive results.
- AWS Lambda played a crucial role in the backend, as it served as the compute service that performed the calculations requested by users.
- AWS API Gateway acted as the bridge between the frontend and backend, providing a secure and scalable way for the calculator to communicate with other services.

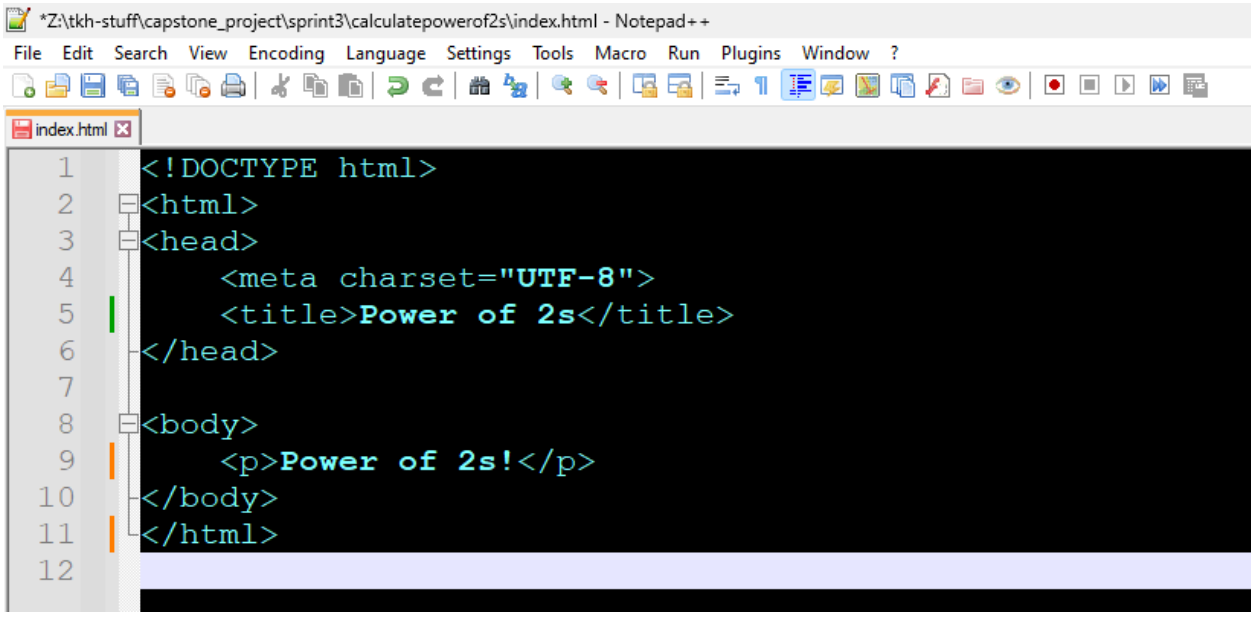
Submitted by: Emilie Dionisio

- Amazon DynamoDB was used to store and manage the data associated with each calculation request, providing fast and reliable access to information.
- AWS Identity & Access Management ensured that only authorized users were able to access and interact with the calculator, enhancing the overall security and privacy of the application.

Need the following:

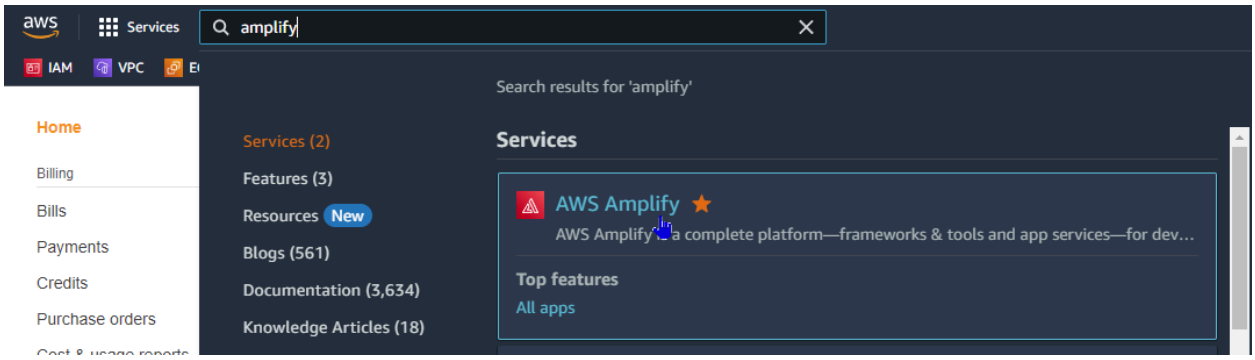
- Create a webpage host
    - Amplify - is used to build and host a websites
- Invoke a math function
- Calculate the math
- Store and return the math result to the user
- Handle permissions
- Text Editor (notepad++, visual studio)
- AWS account
- Basic knowledge of AWS

In the local machine, create a workspace, create a folder called “calculatepowerof2s” within the folder and create an index.html (Just a plain index.html with a title and body “Power of 2s”) and zip it.

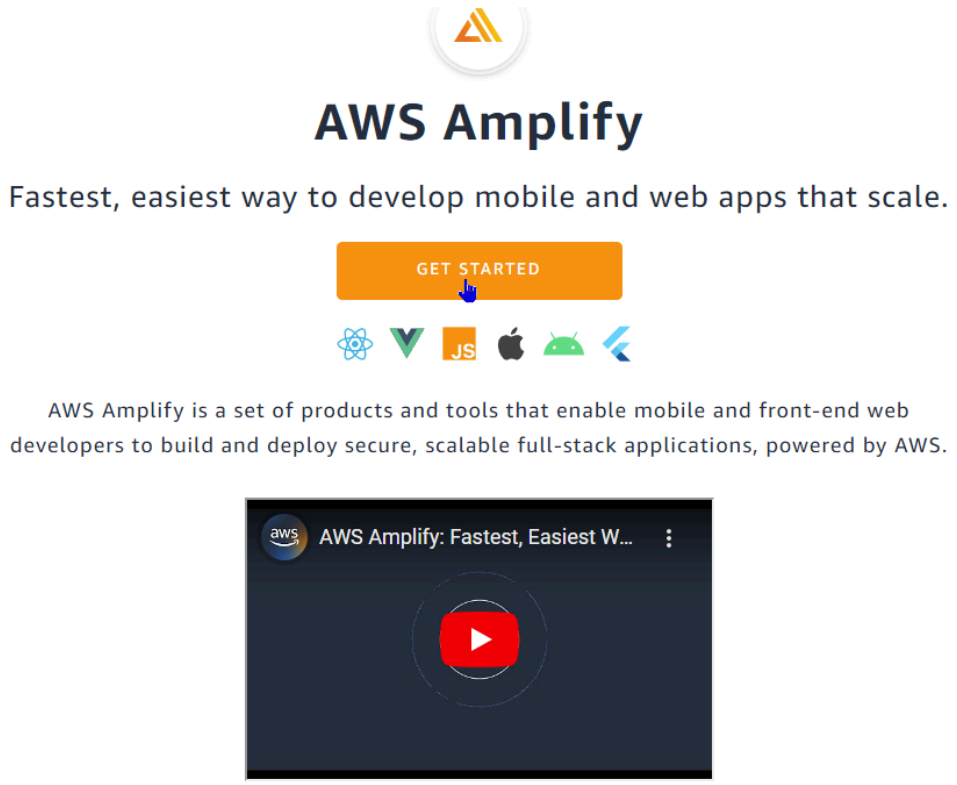


Log in to AWS and search for AWS Amplify in the AWS Console search.

Submitted by: Emilie Dionisio

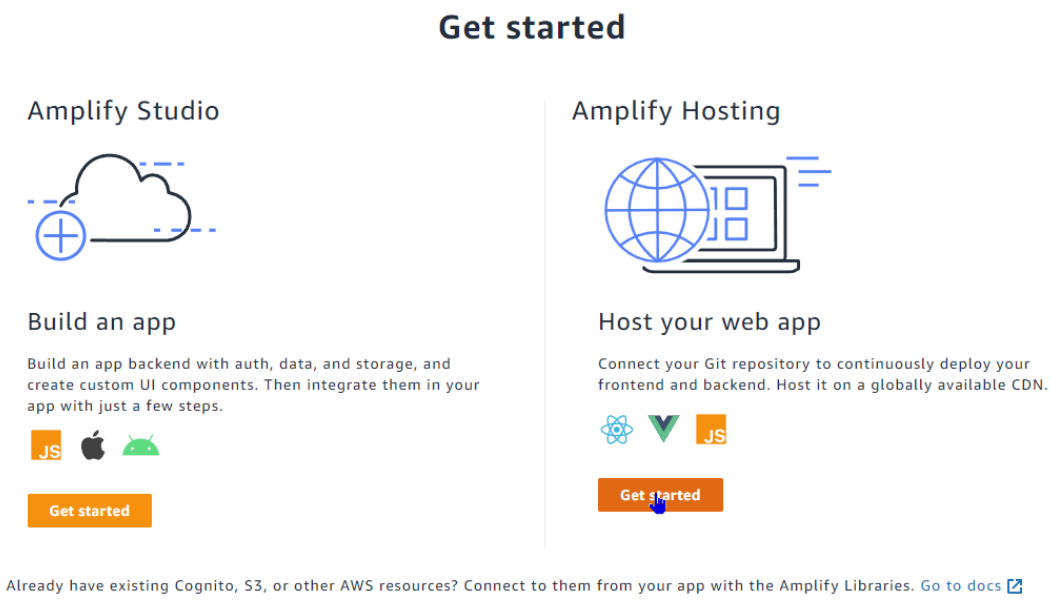


In AWS Amplify page, select “Get Started” button

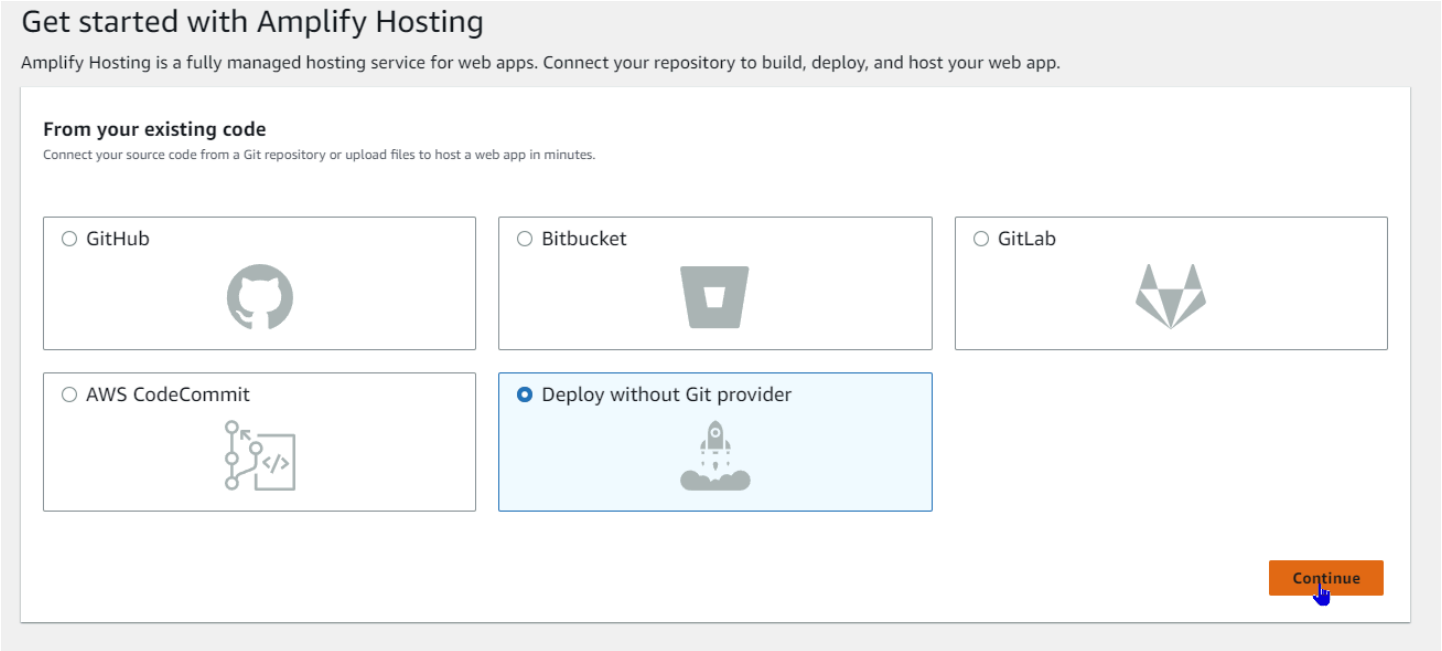


In Get Started under Amplify Hosting, select the “Get Started” button.

Submitted by: Emilie Dionisio

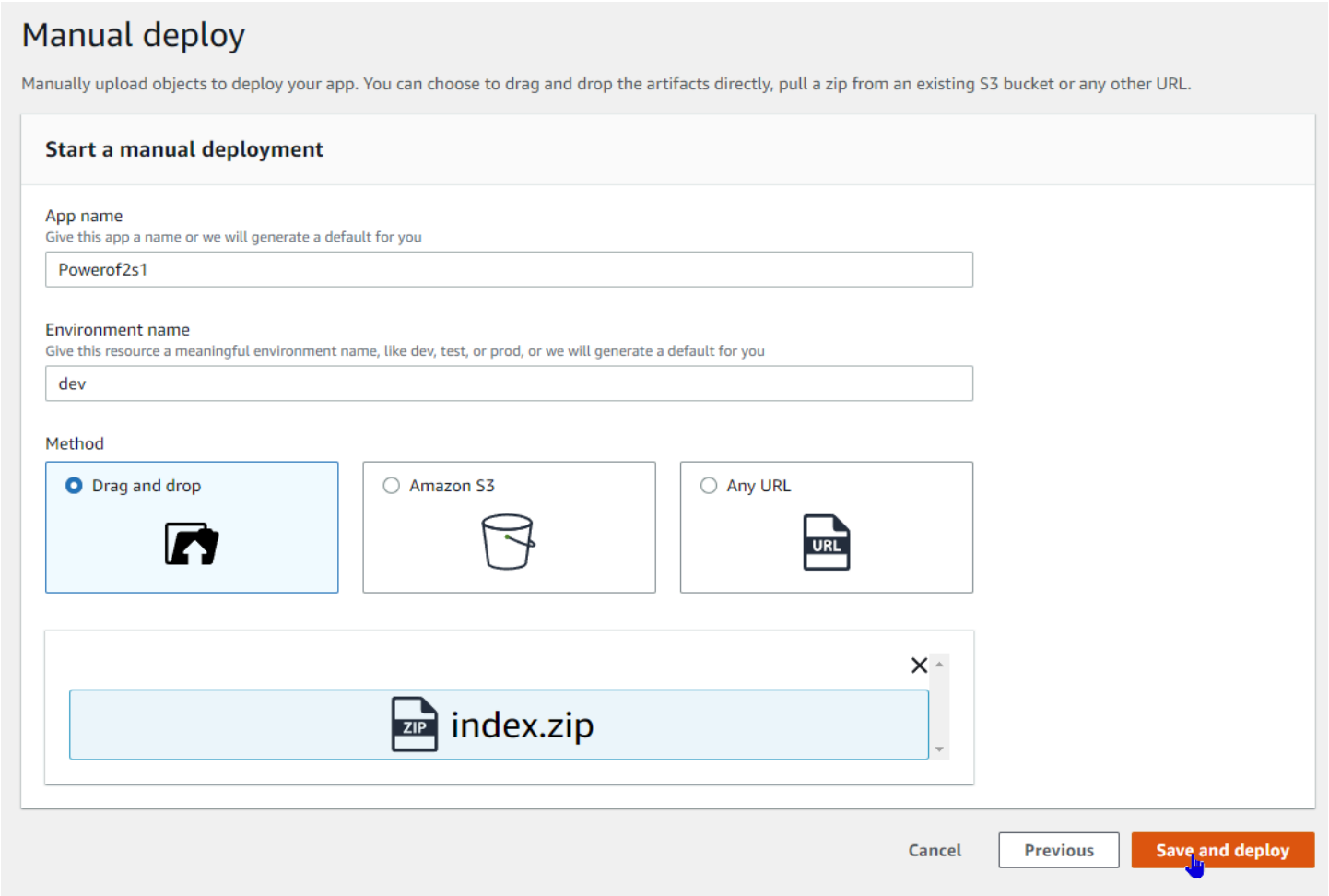


In Get started with Amplify Hosting, select Select “Deploy without Git provider”, and click “Continue” button.

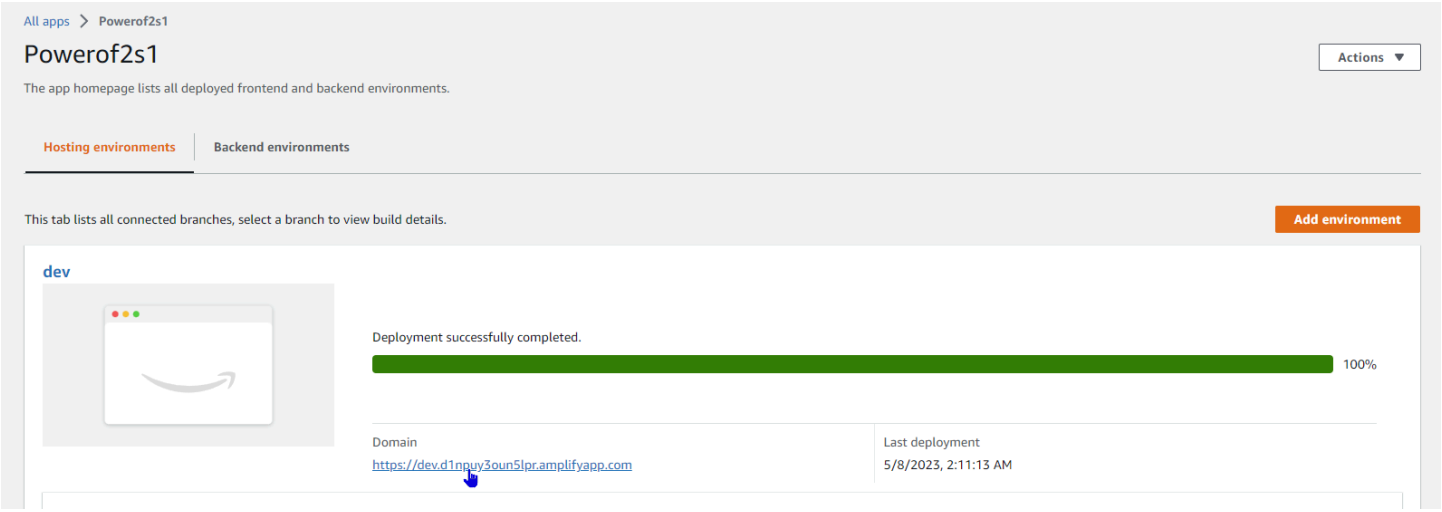


In Manual Deploy page, create an App Name, call it “Powerof2s1”. Type “dev” as Environment name and select Drag and drop, click choose file and select “index.zip” and click “Save and deploy” button.

Submitted by: Emilie Dionisio

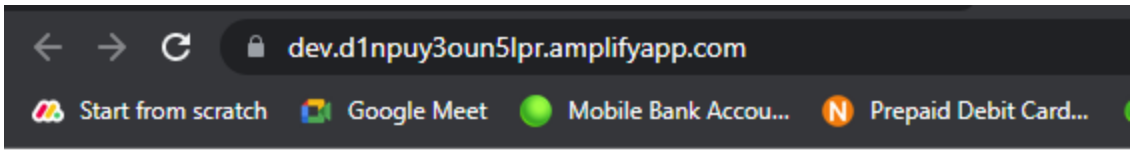


You will see “Deployment successfully completed” and the link has been created.



When you click the link another browser tab will open and you will see the index.html page with the title “Power of 2s on the page.

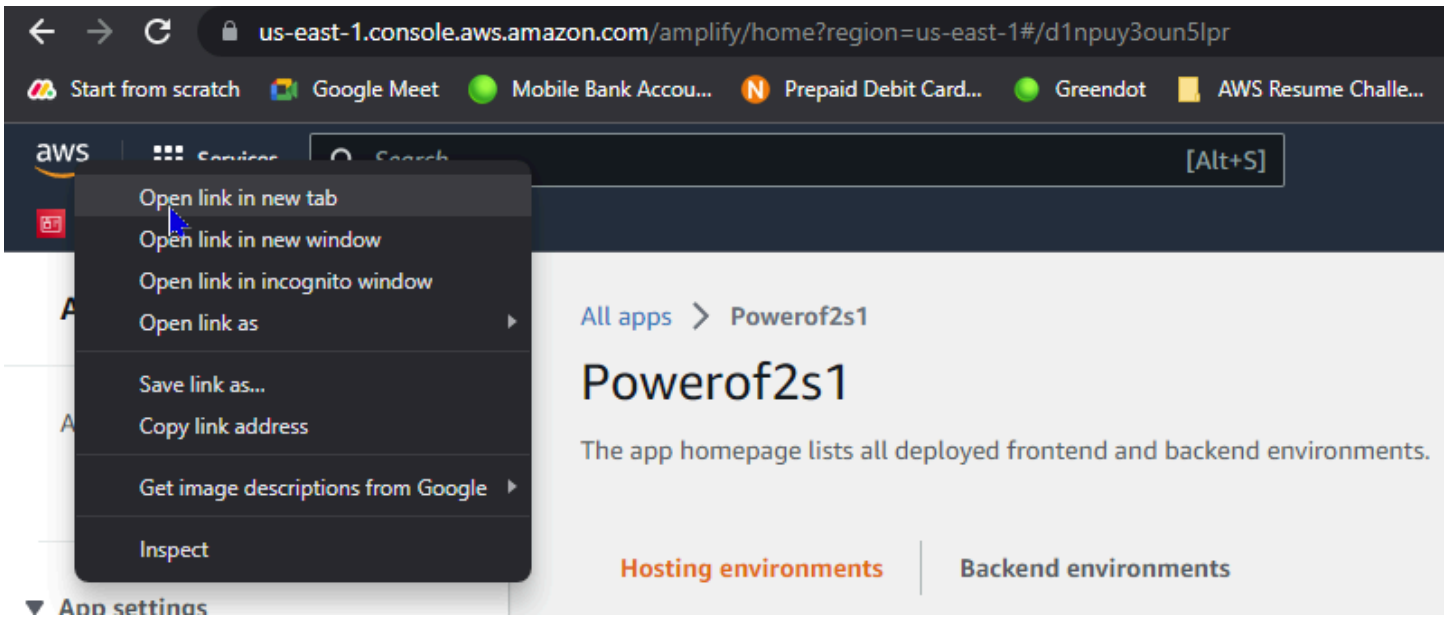
Submitted by: Emilie Dionisio



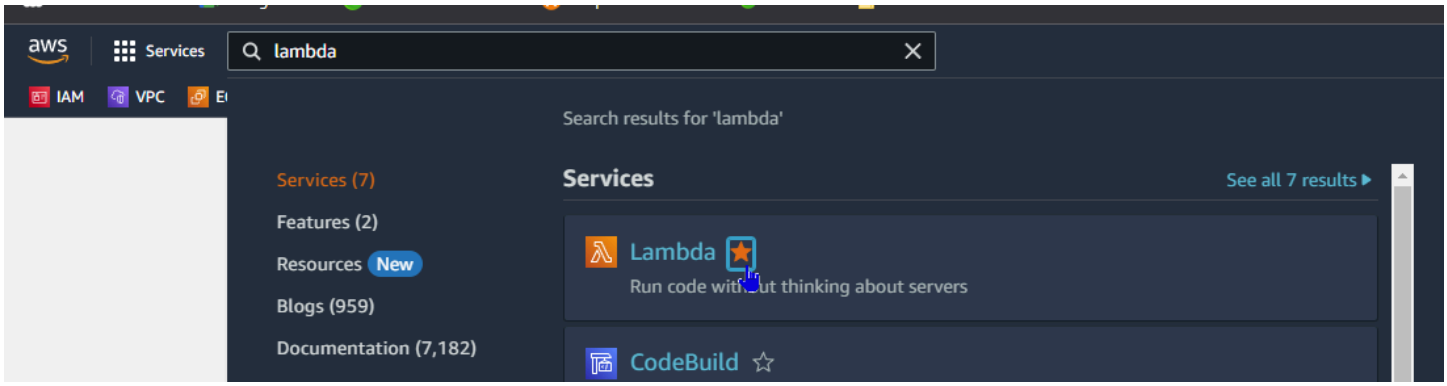
Power of 2s!

To do the math, a Lambda function will be used. In simple terms, Lambda is a code function that responds to a trigger and it's serverless so it doesn't need to manage a server to store the code. Python code will be used for this exercise and will use some Python math library to use the calculations that we need.

On AWS Console, right click the AWS logo and click on "Open link in new tab" to keep Amplify handy.

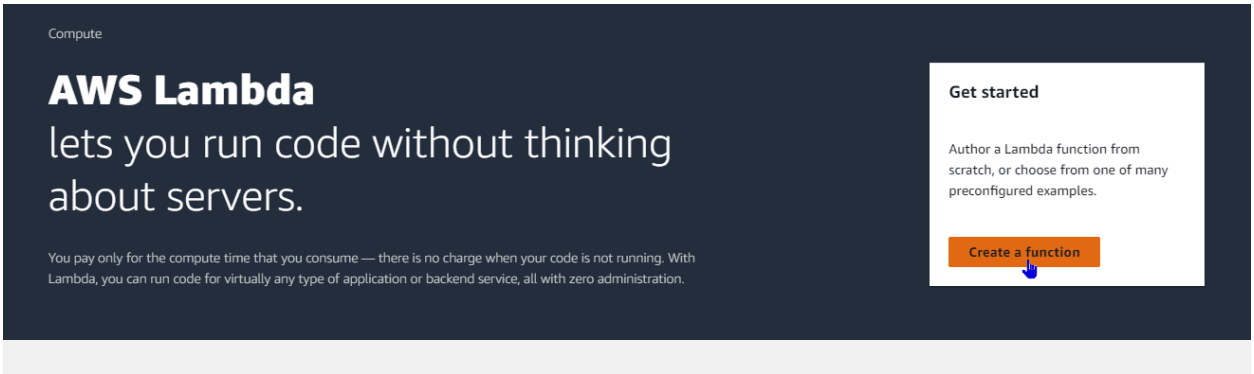


In the AWS console search for Lambda and click it.

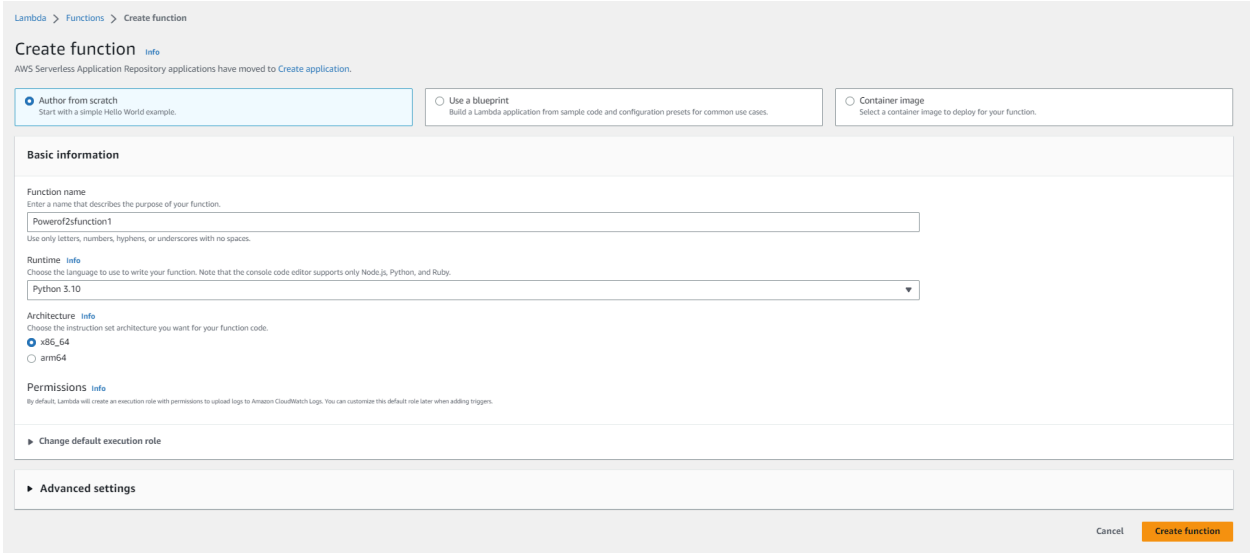


Submitted by: Emilie Dionisio

In AWS Lambda page, click on Create a function



Enter a Function name (name it what you like) but you can also use the same name, “Powerof2sfunction1” and select the latest verion of Python for Runtime, in this case I chose Python 3.10 then click “Create function” button.

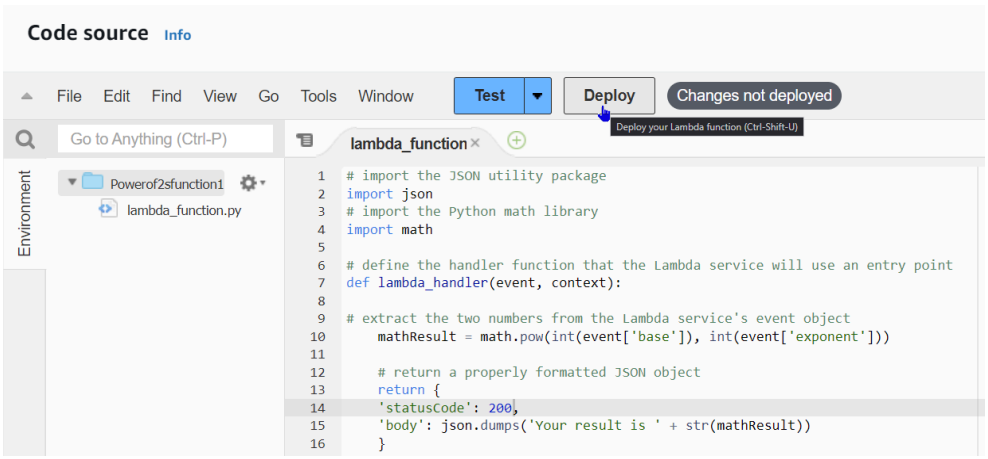


Once it’s created, scroll down to the code source, copy and paste the Lambda function (see link for: Lambda-function.txt) which contains a simple math in python that will calculate the math from the Python math library and it will return a json object that will give the result. After pasting the code on the code source, click Ctrl+S or click on File then save.

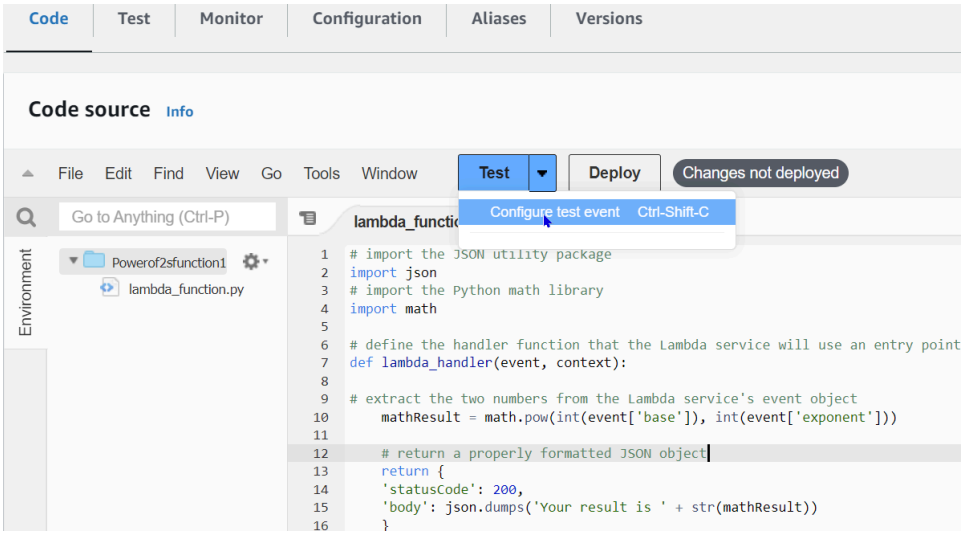
Submitted by: Emilie Dionisio

```
# import the JSON utility package
import json
# import the Python math library
import math

# define the handler function that the Lambda service will use an entry point
def lambda_handler(event, context):

# extract the two numbers from the Lambda service's event object
    mathResult = math.pow(int(event['base']), int(event['exponent']))

    # return a properly formatted JSON object
    return {
    'statusCode': 200,
    'body': json.dumps('Your result is ' + str(mathResult))
    }
```

Click on Deploy next to the Test button.



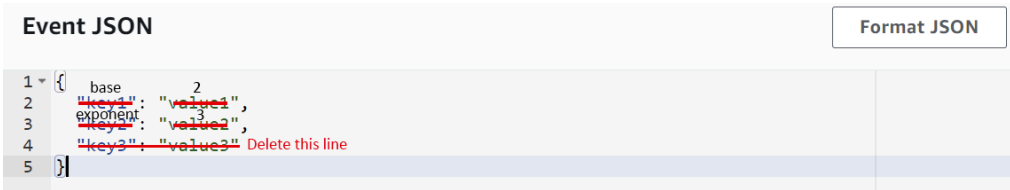
Click the arrow on the “Test” button then click on “Configure test event” or Ctrl+Shift+C to test the event.



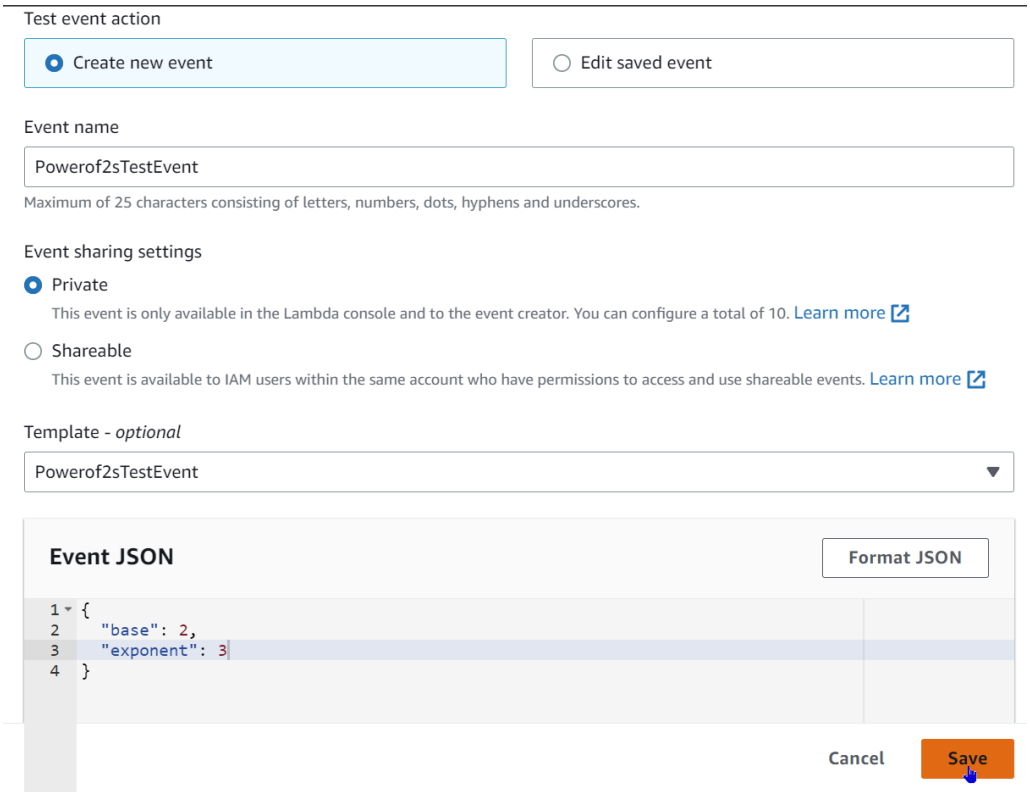
Create a new event, under the Event name enter “Powerof2stestevent”. Scroll down to EvenJSON field and edit the following:

Submitted by: Emilie Dionisio

- delete the fourth line "key3": "value3
- change the second line “key1” to “base”, change the value1 to “2”
- change the third line “key2” to “exponent”, change the value2 to “3”

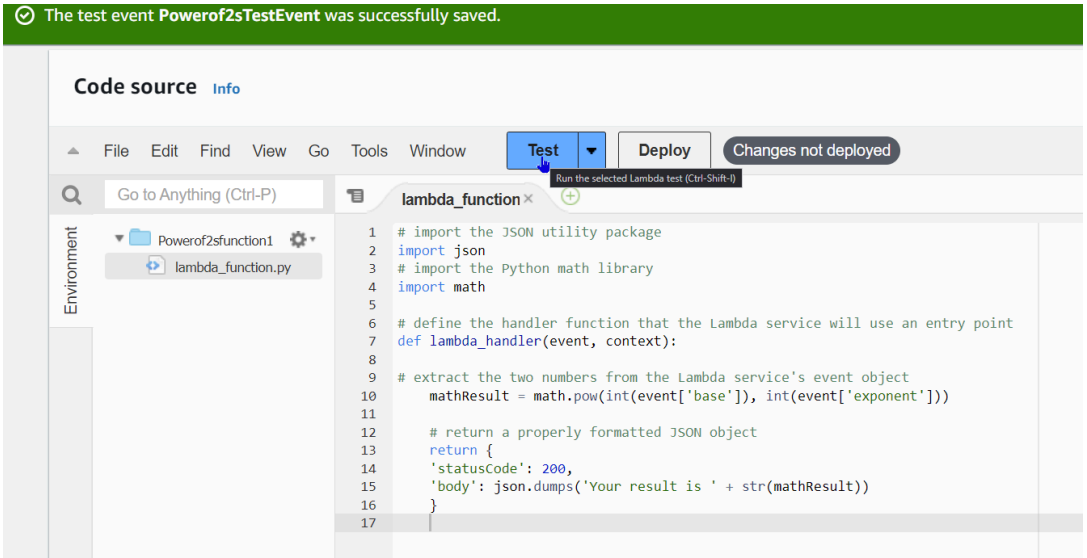


It should look like this:

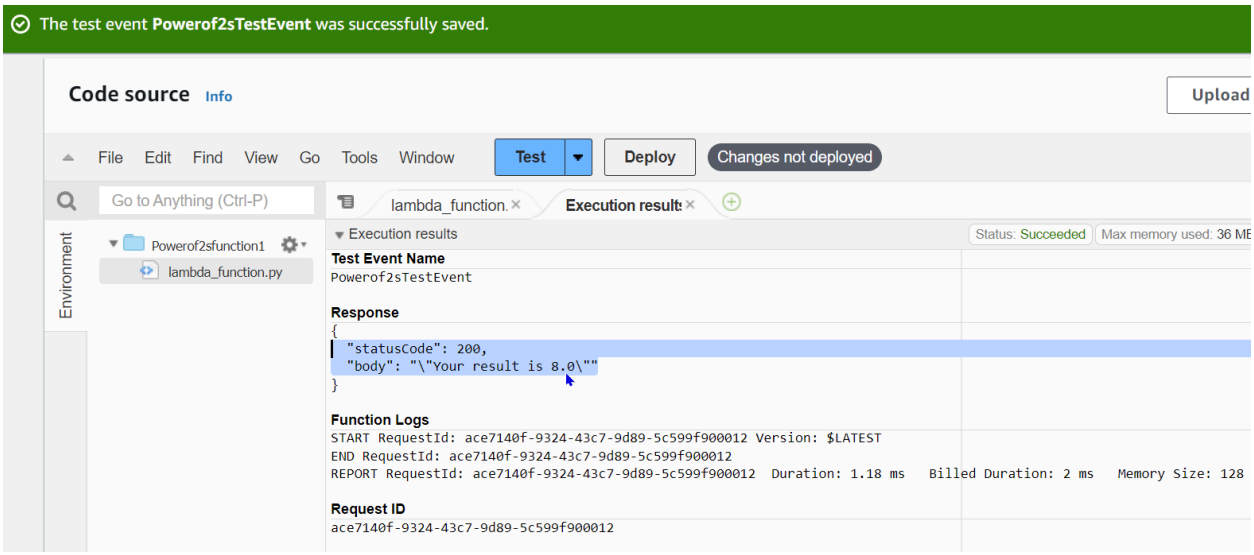


To run the test, click the “Test” button.

Submitted by: Emilie Dionisio

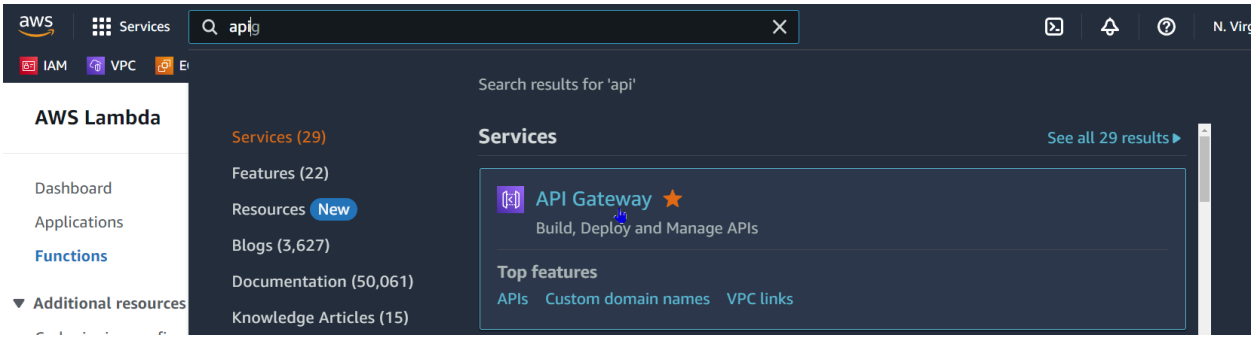


You know that the Lambda function is working when you see “200” for status code and “8” for the result which is (2^3).



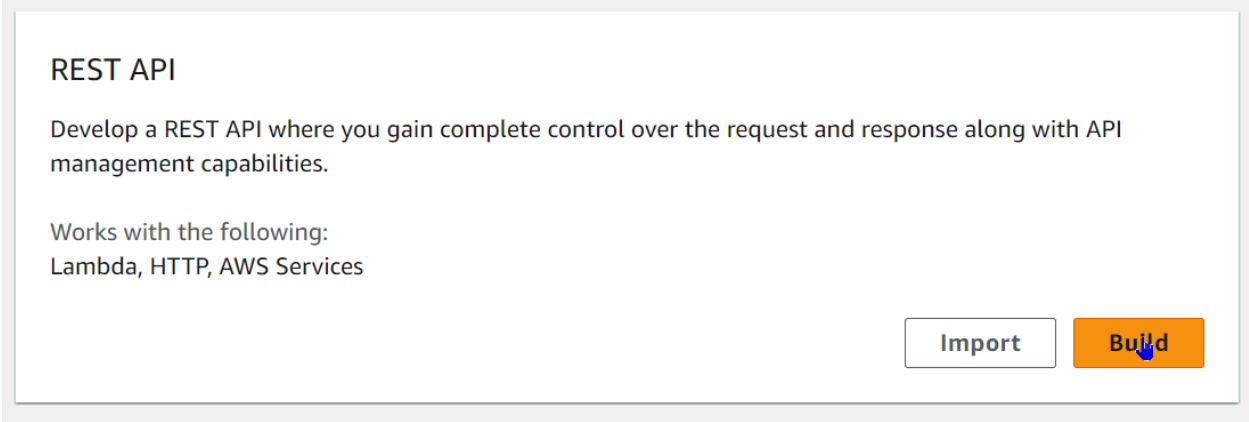
Invoking the math functionality

In AWS Console, search for API Gateway

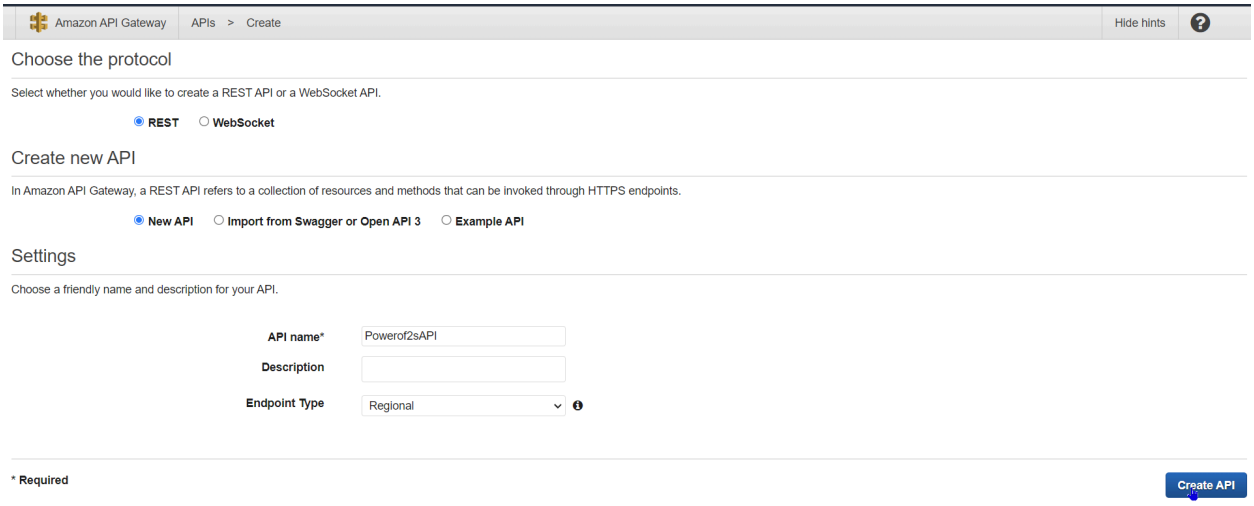


Submitted by: Emilie Dionisio

In choose an API type, select Rest API then click “Build” button.

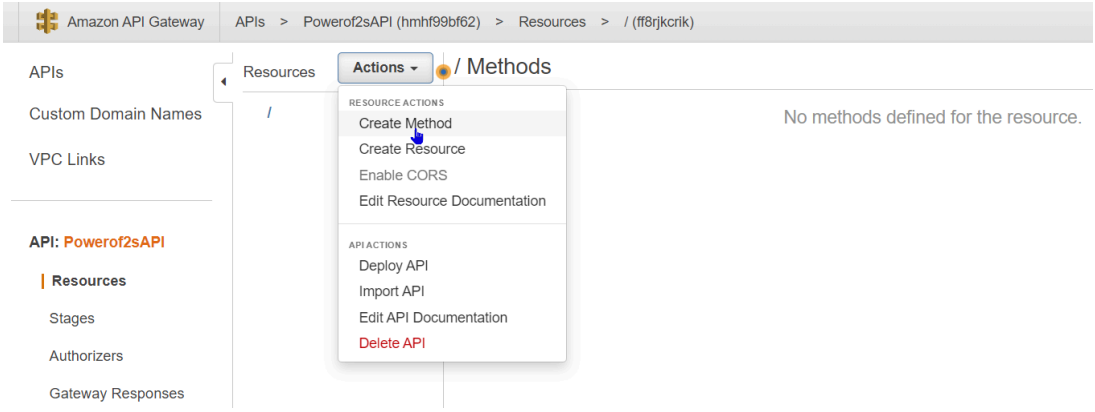


In Choose the protocol, leave the Rest radio button selected and select “New API” under Create new API, give it a name, and press “Create API” button.

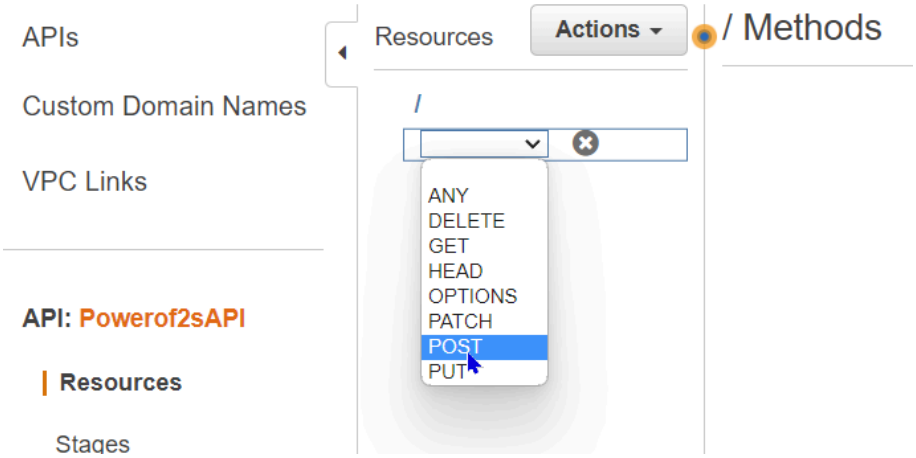


On the API Methods page, on the left navigation, make sure that “Resources” is selected and in the middle make sure that the backslash (/) is highlighted. Click on the Actions menu and select “Create Method”.

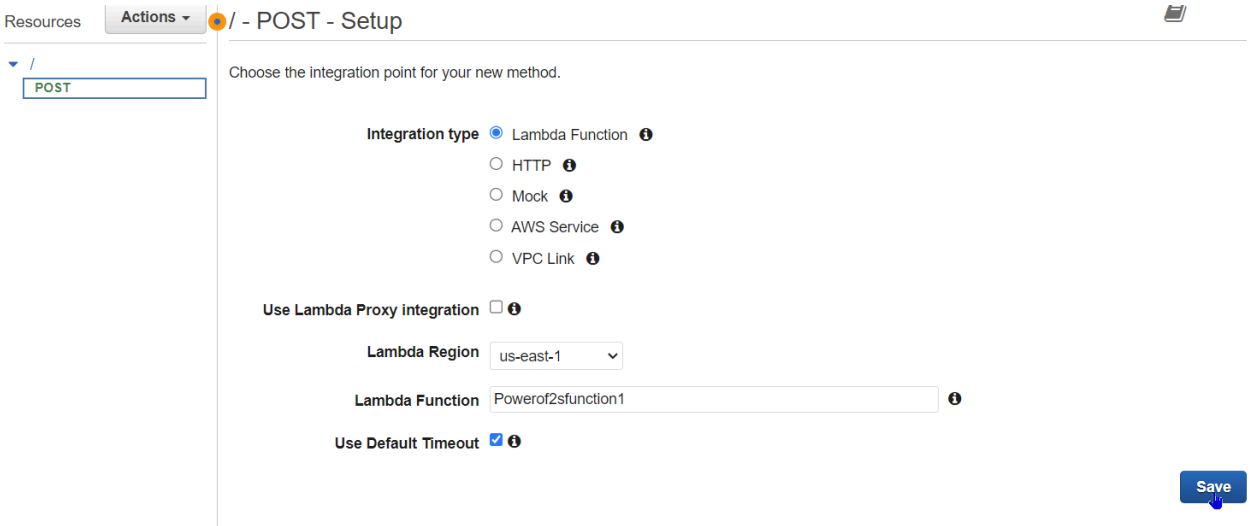
Submitted by: Emilie Dionisio



On the backslash menu, choose the type of method which is “Post” then click the check mark.

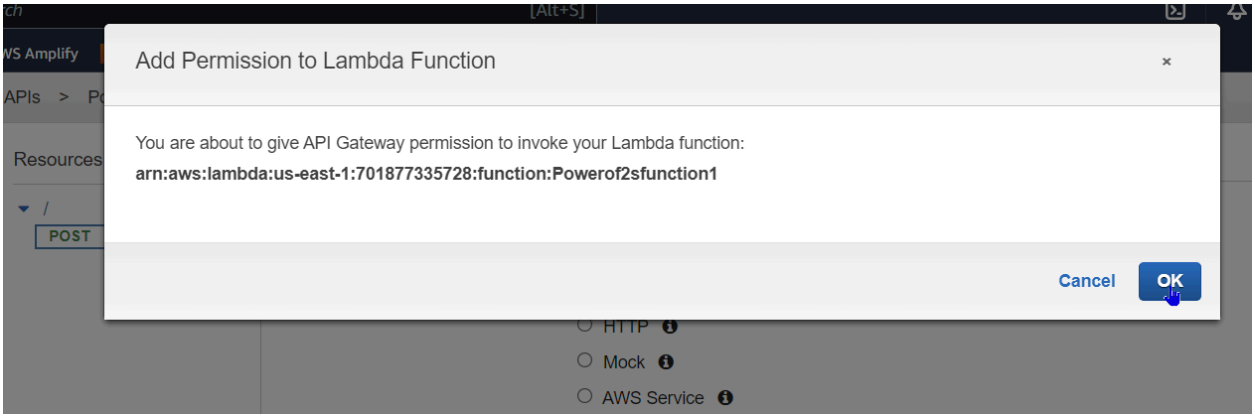


For integration type, choose “Lambda Function” then under Lambda Function choose the function that was created before which was the “Powerof2sfunction1” function then click “Save”.

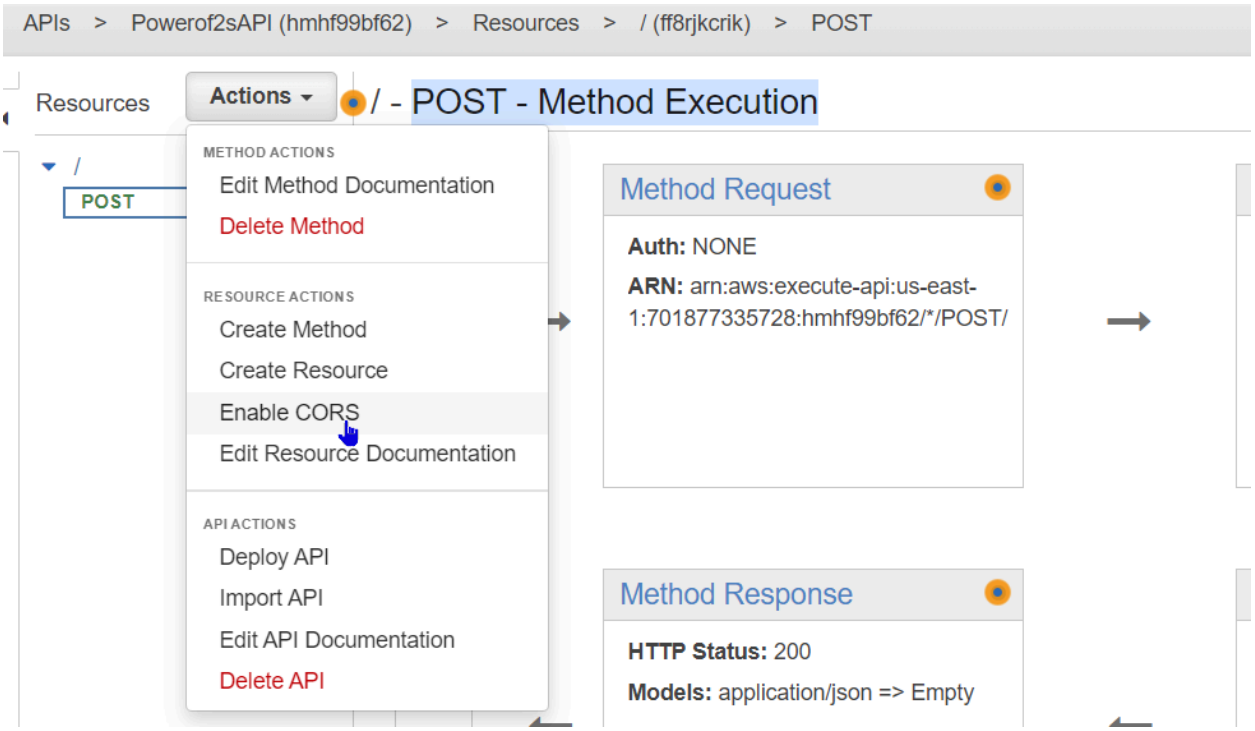


Click Ok when you see “Add Permission to Lambda Function”. It states that we are giving API Gateway a permission to invoke this Lambda function.

Submitted by: Emilie Dionisio



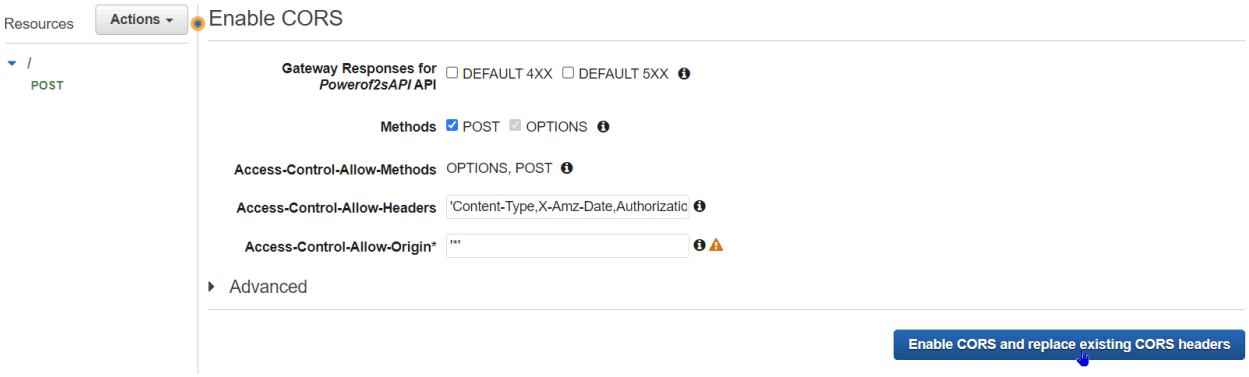
Next, we need to enable CORS (Cross origin resource sharing). On the POST - Method Execution page, make sure POST is selected, click Actions menu and select “Enable CORS”.



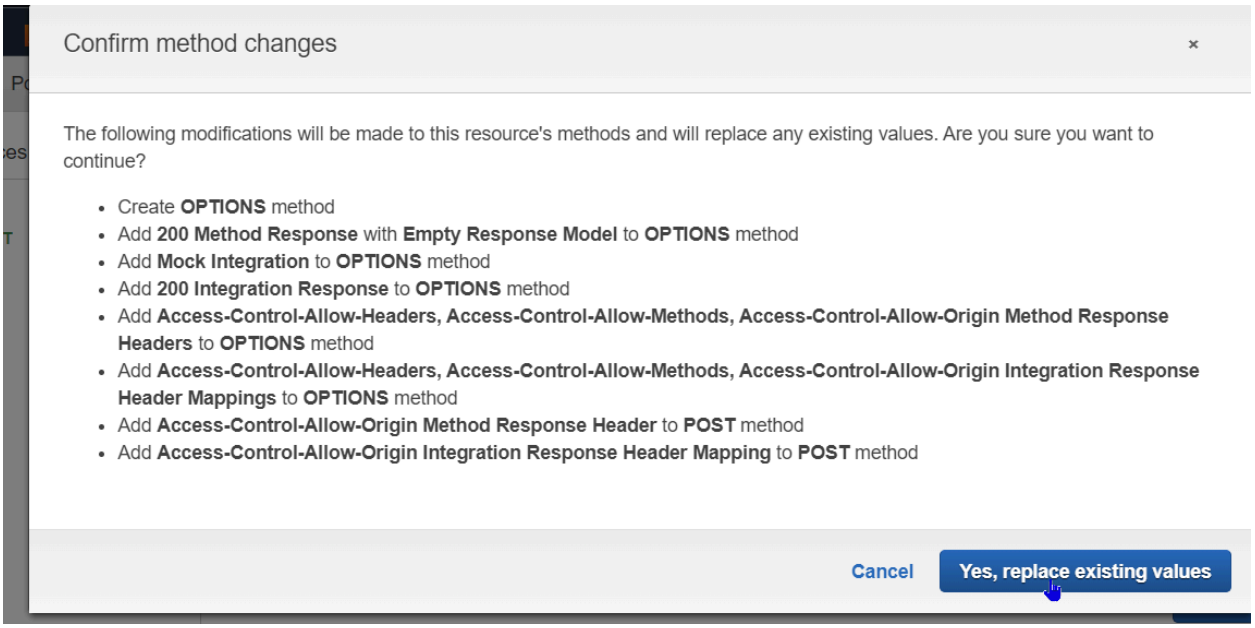
Note: What this does is, it allows a web application running in one origin or domain to be able to access resources on a different origin or domain because our web application is running on one domain and amplify, our Lambda function is going to be running in another and we need to be able to work across those domains or origins.

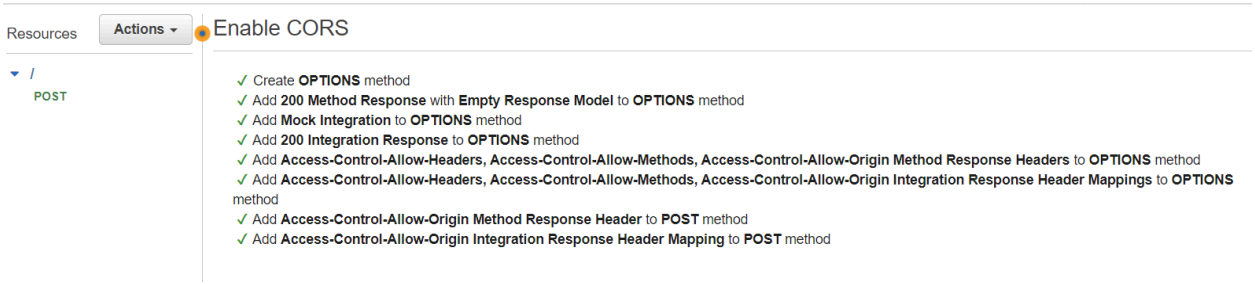
Click on the “Enable CORS and replace existing CORS headers” button.

Submitted by: Emilie Dionisio



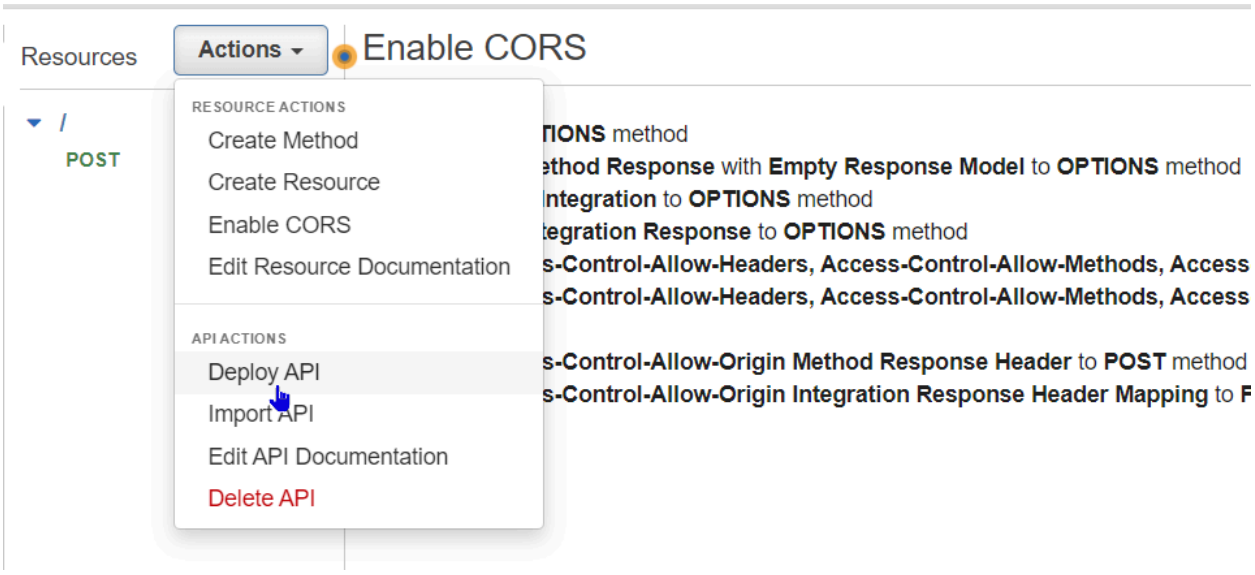
Confirm by clicking “Yes, replace existing values” button.



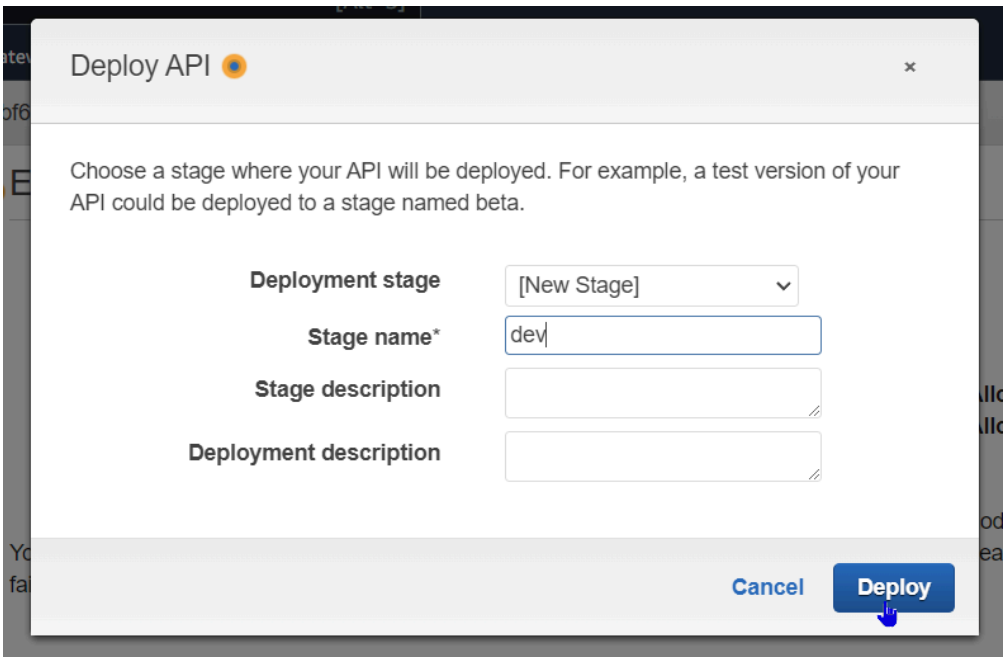


Now, we can deploy the API to test it out. Click on Actions and select “Deploy API”

Submitted by: Emilie Dionisio

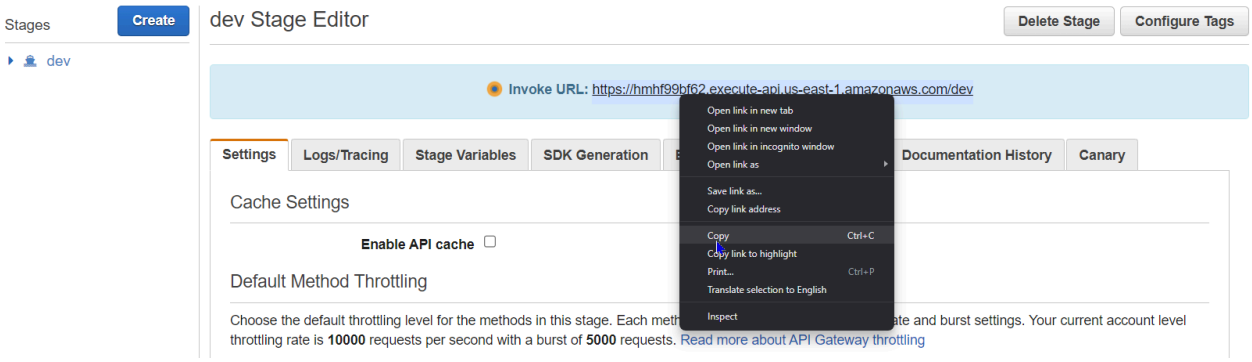


On the Deploy API pop up menu, select [New Stage] under Deployment stage, and under Stage name, type "dev" then click "Deploy" button.

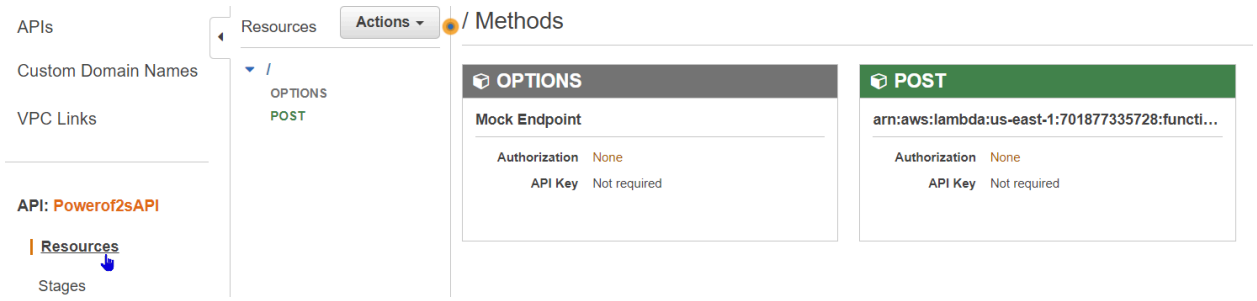


On the **dev Stage Editor** page, copy the Invoke URL to a notepad (You will need the URL for later).

Submitted by: Emilie Dionisio



Next, we need to validate this. On the same page on the left navigation, click Resources



Incorporating database on this little project. We are going to use DynamoDB for our database.

We are going to persist or store that math result somewhere like a database and return it to the user to see the result.

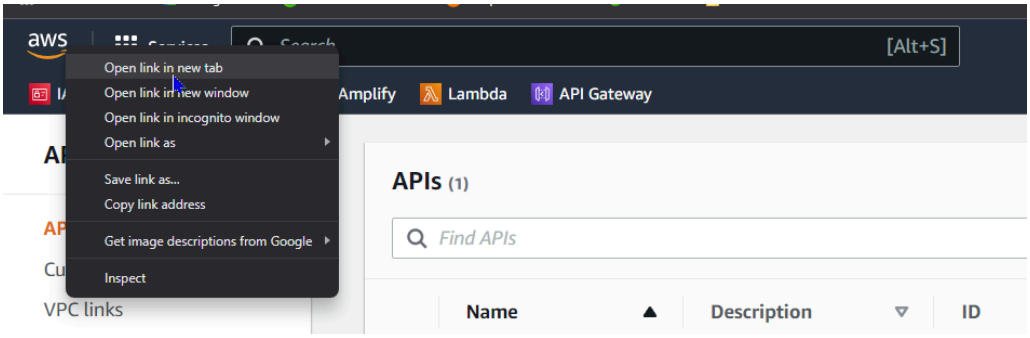
Note: We don't actually need a database to keep the results; in fact, we can just provide the results to the user because we aren't retaining a large quantity of data. Databases are required in the real world, thus we want to demonstrate in this scenario that we are incorporating databases to store information for learning purposes.

We are going to need to handle permissions between the different parts of the application. For this section we are going to use the following:

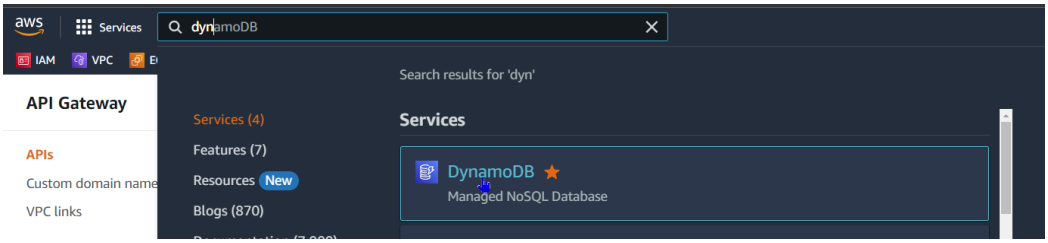
- DynamoDB - this is a key value or NOSQL database (see Index for more information). It's a lighter weight compared to a relational database.
- Permissions to have access to Lambda function

On the AWS Console, right click the AWS logo on the top left and click "Open link in new tab" so we can use it when creating DynamoDB.

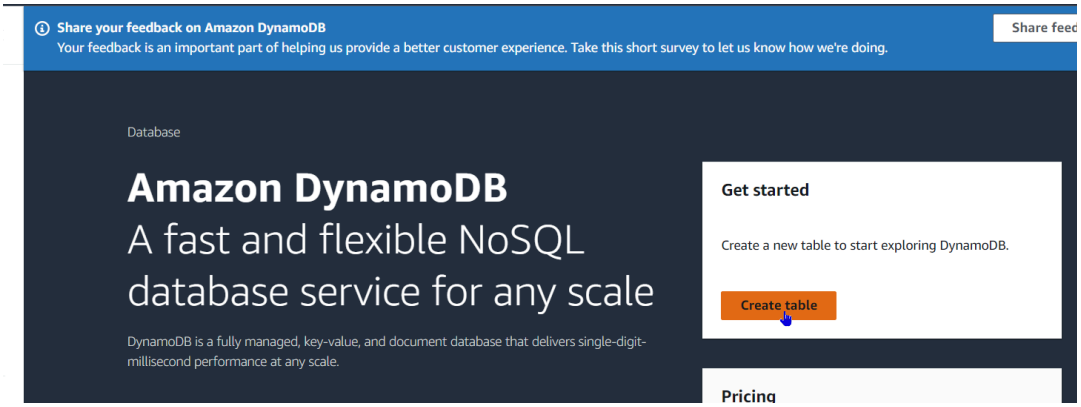
Submitted by: Emilie Dionisio



On the AWS Console, search DynamoDB and click it.



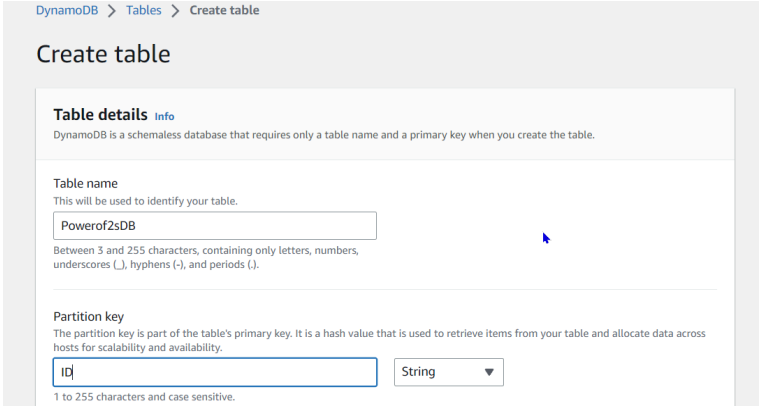
On the DynamDB page, click on “Create table”.

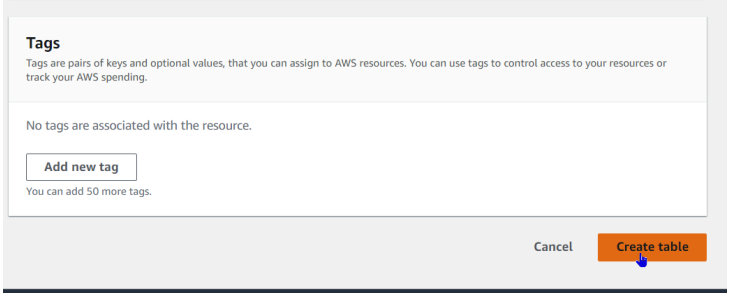


Enter the following information:

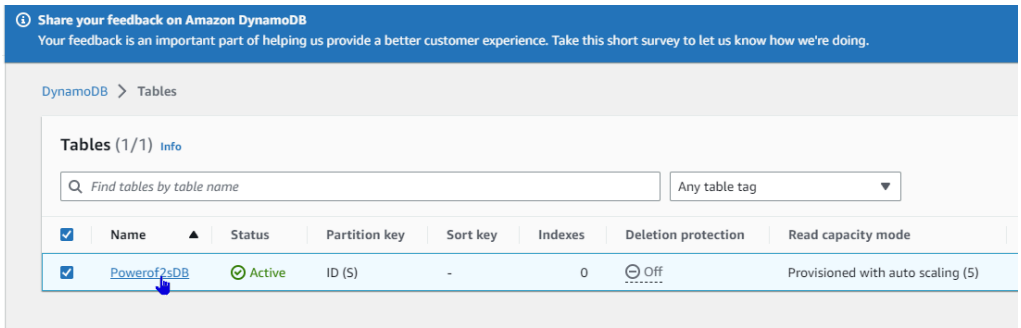
- Table name: Enter your table name (ie. Powerof2sDB)
- Partition key: Enter “ID”
- Leave the rest the same
- Click Table.

Submitted by: Emilie Dionisio



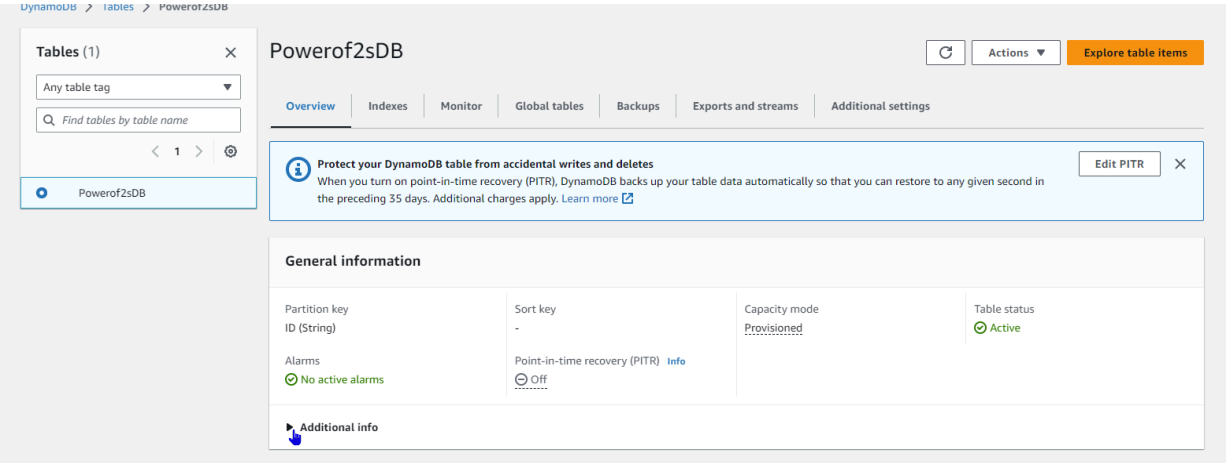


On the DynamoDB table's page, make sure the table is Active, refresh the page to make sure it's active. Click on the table: Powerof2sDB.

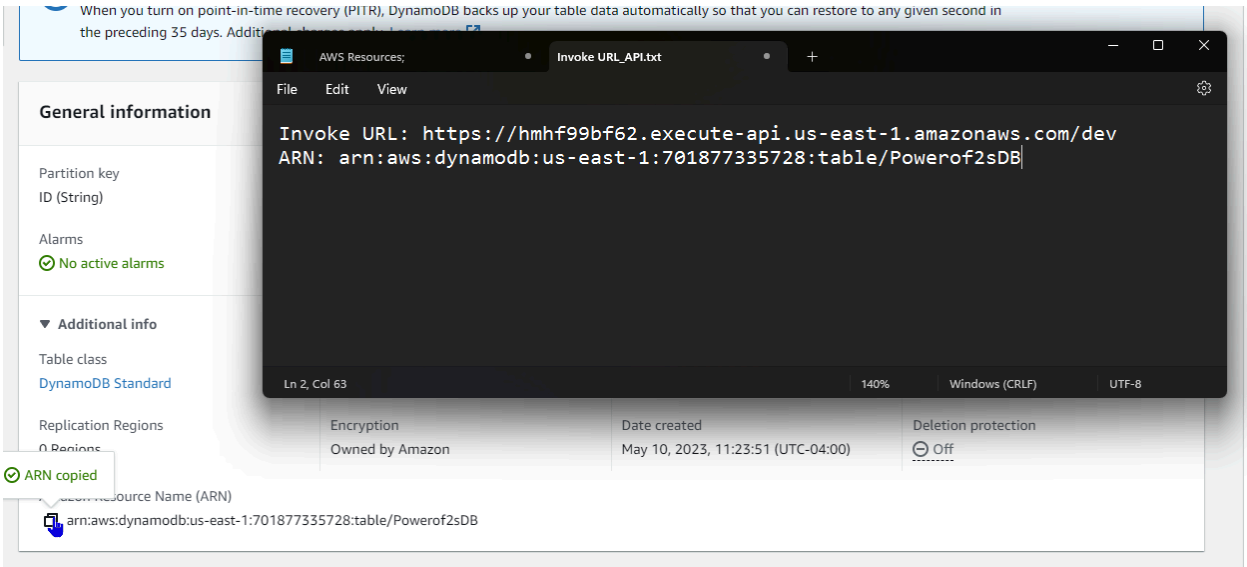


On the details page, click on the "Additional Info"

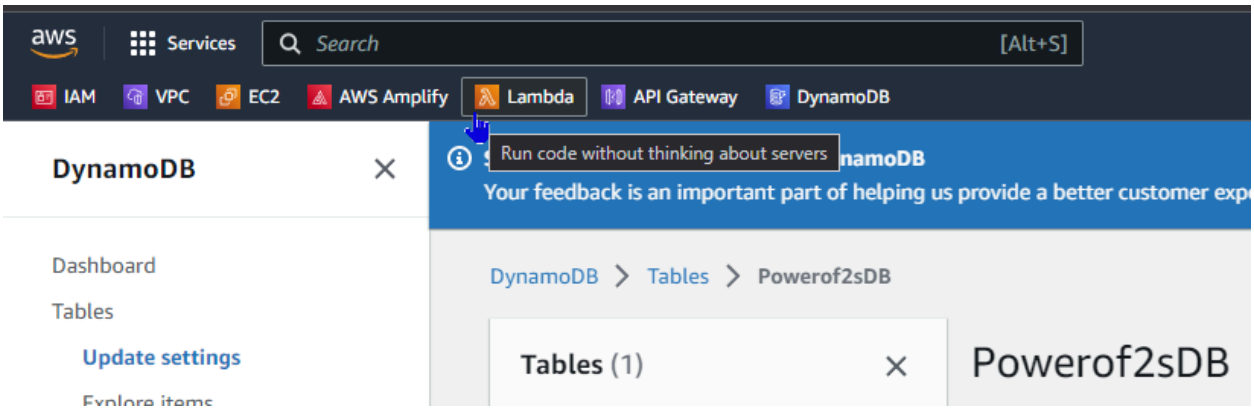
Submitted by: Emilie Dionisio



Copy the ARN (Amazon Resource Name) in a notepad together with the Invoke URL that you copied before.

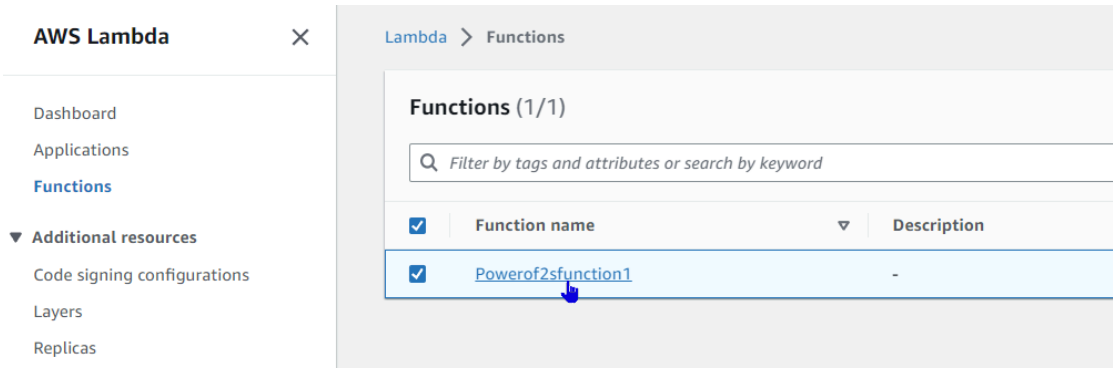


We need our Lambda function to have permission to write to our table. Go back to Lambda by clicking the shortcut or search for Lambda.

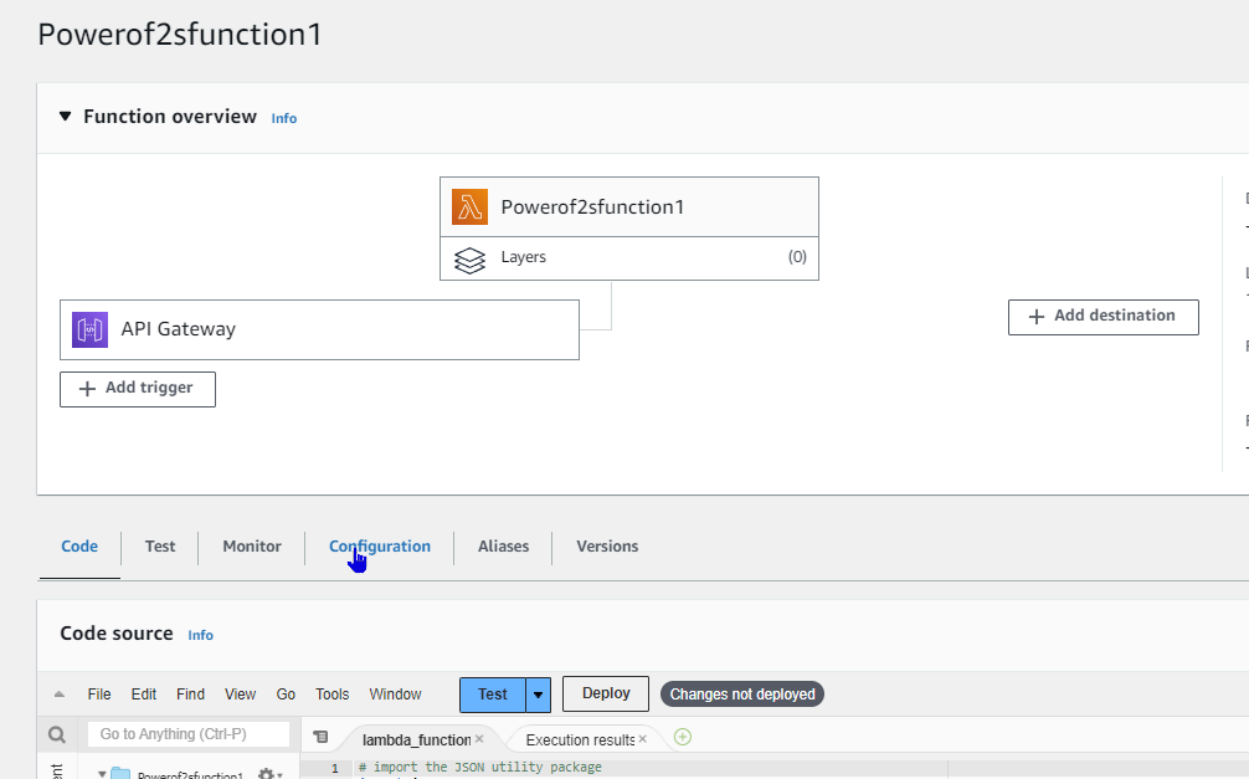


Submitted by: Emilie Dionisio

On the Lambda Dashboard, on the left navigation, click on Functions, click on the function name (ie. Powerof2sfunction1)

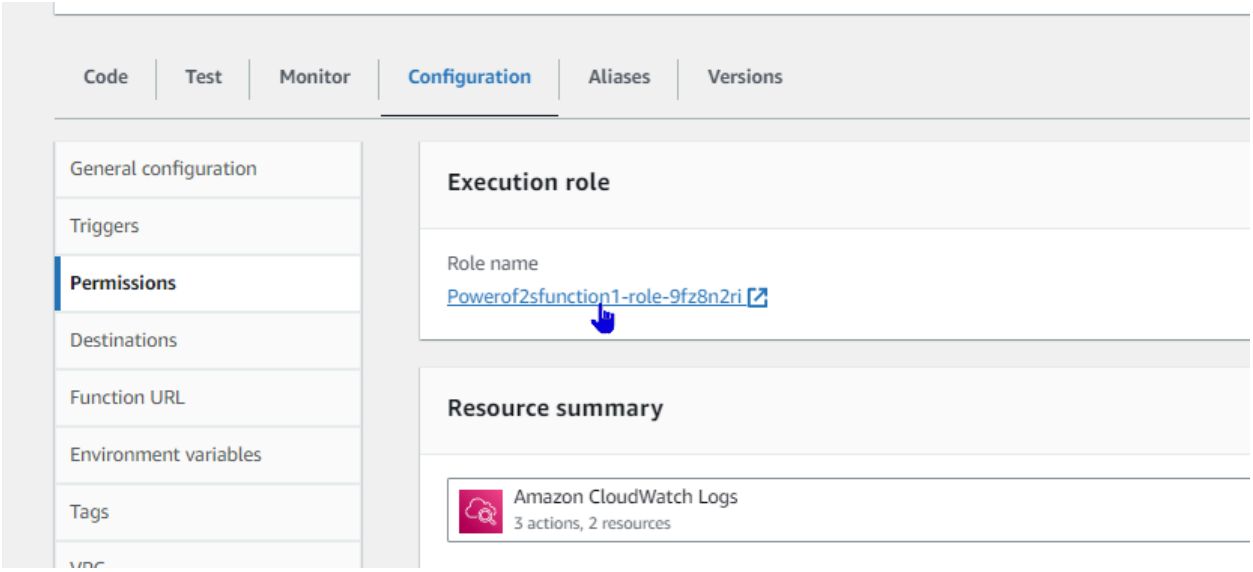


On the Powerof2sfunction page, click on Configuration tab.

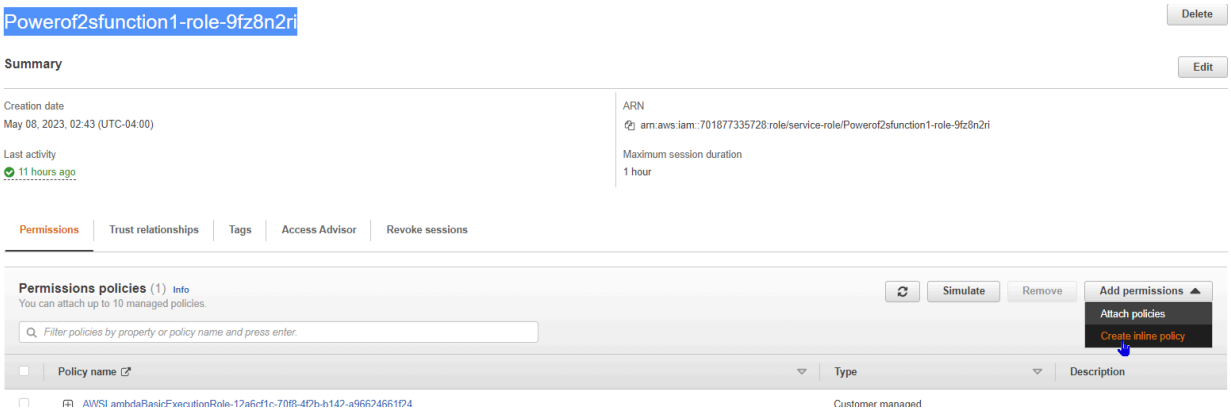


On the left navigation, make sure Permissions is selected and click the Role name: Powerof2sfunction1-role-xxxxx then it will open a new tab for it.

Submitted by: Emilie Dionisio

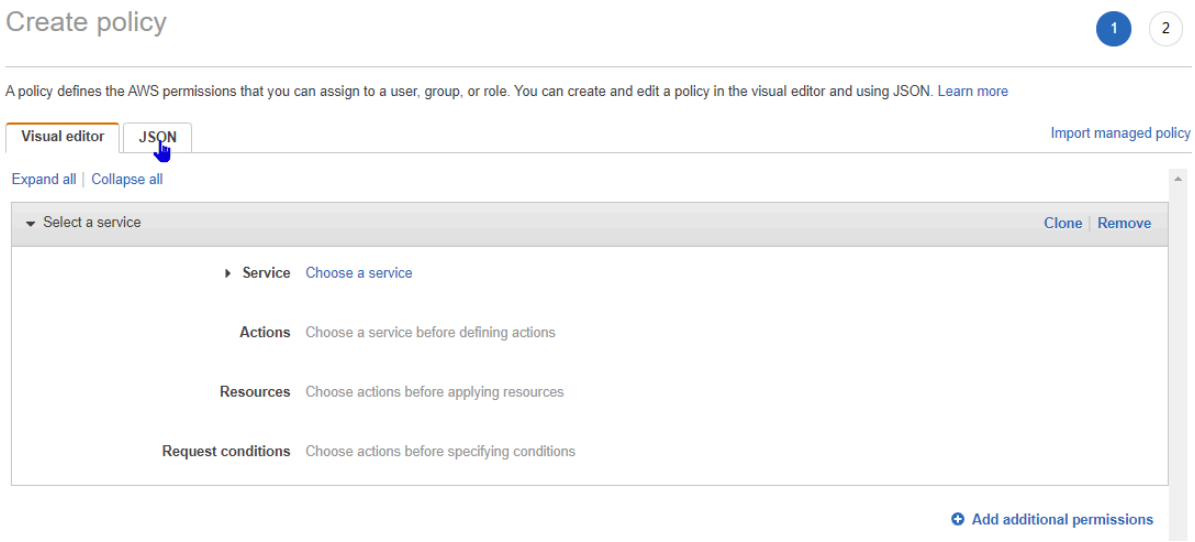


On this step, we need to add some permissions pertaining to DynamoDB. To do that, on the "Powerof2sfunction1-rolexxxxxxxx" page on the right side, click on Add permissions and click on "Create inline permissions".

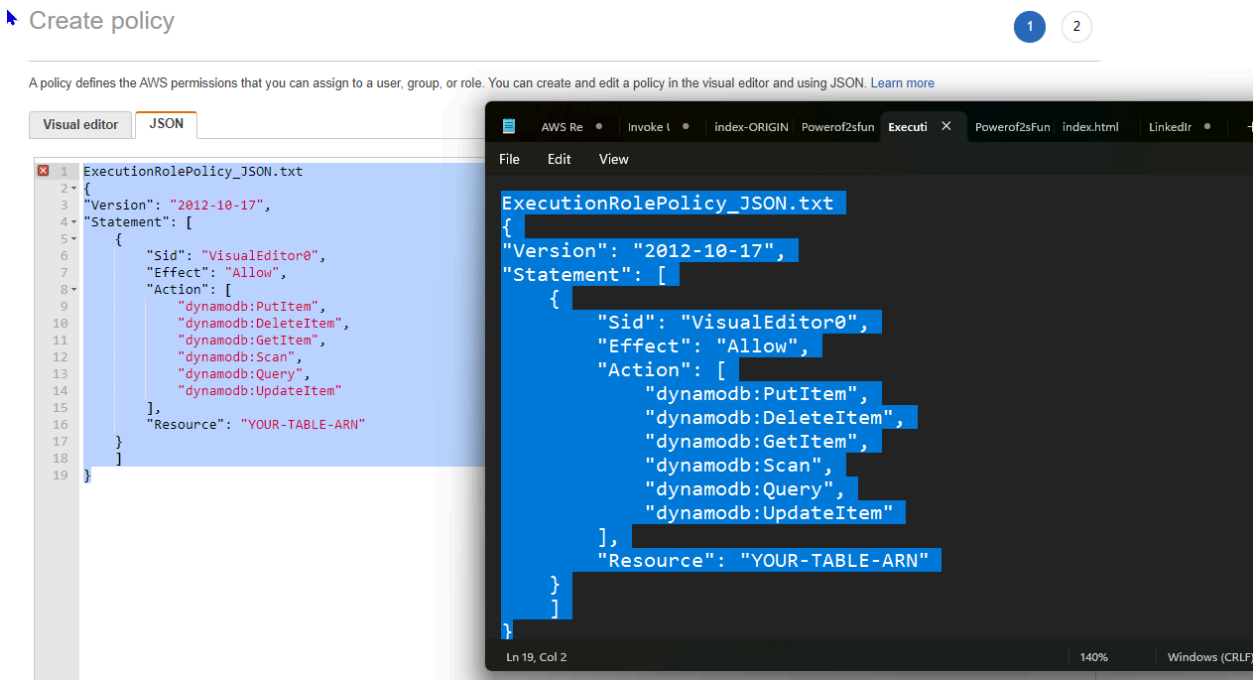


Submitted by: Emilie Dionisio

In “Create a Policy” page, select the JSON tab and grab the “ExecutionRolePolicyJSON.txt” that is included on this instruction.Co

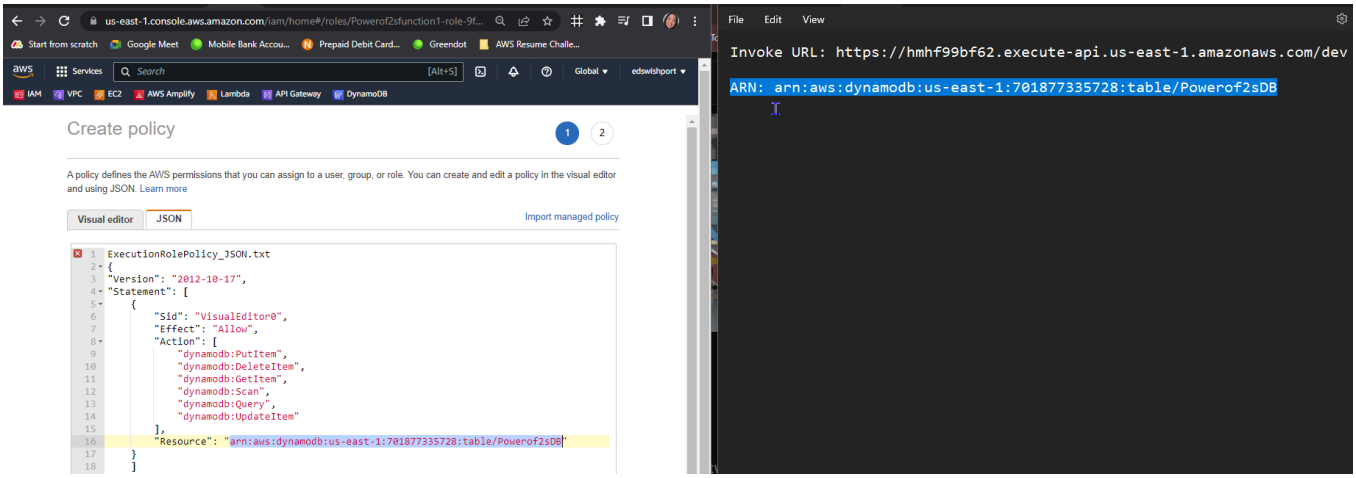


Copy the entire code and replace everything on the JSON editor.

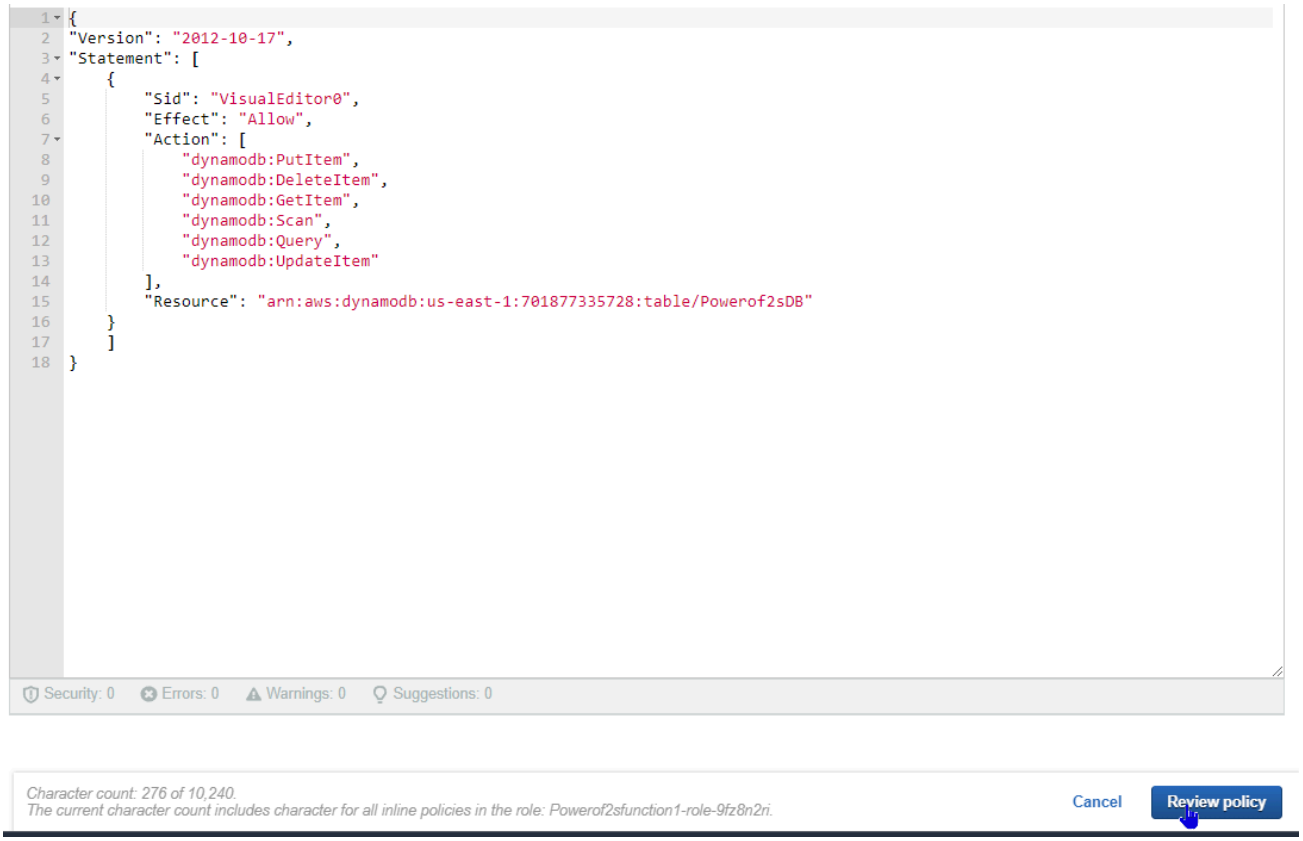


This code is basically stated to allow different actions in DynamoDB so the Lambda function will have permissions to do all of these tasks on the DynamoDB table. Also, it’s important to update the “Resource”: “YOUR-TABLE-ARN” field. Copy the ARN that you saved earlier on the notepad and paste it on where it says: “YOUR-TABLE-ARN”.

Submitted by: Emilie Dionisio

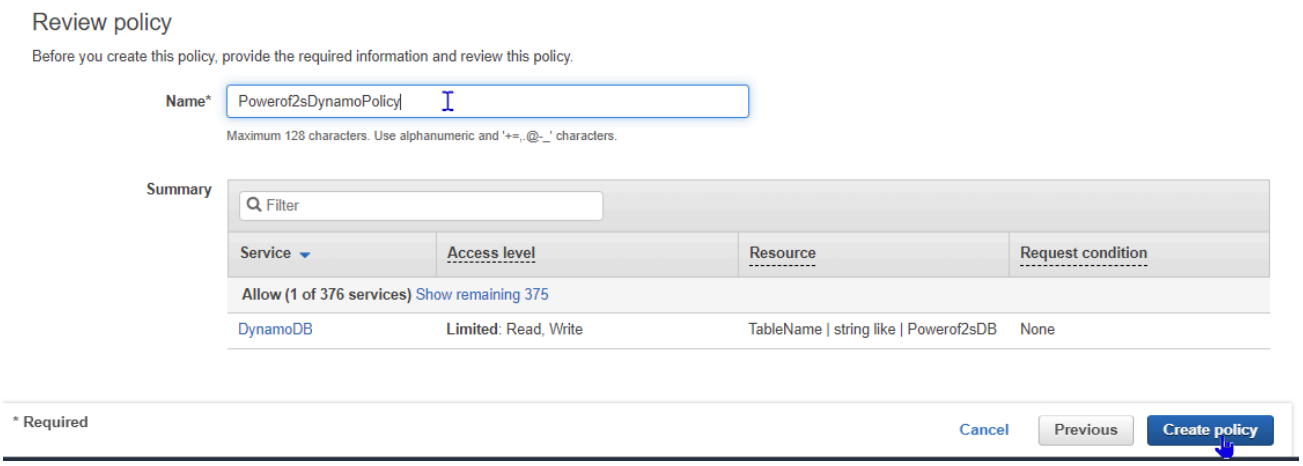


Click “Review Policy” button

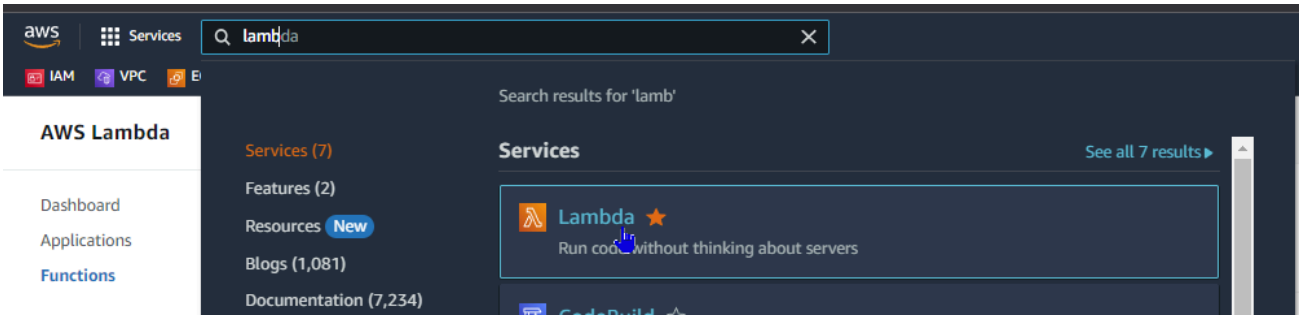


Give it a name: Powerof2sDynamoPolicy then click “Create a Policy” button.

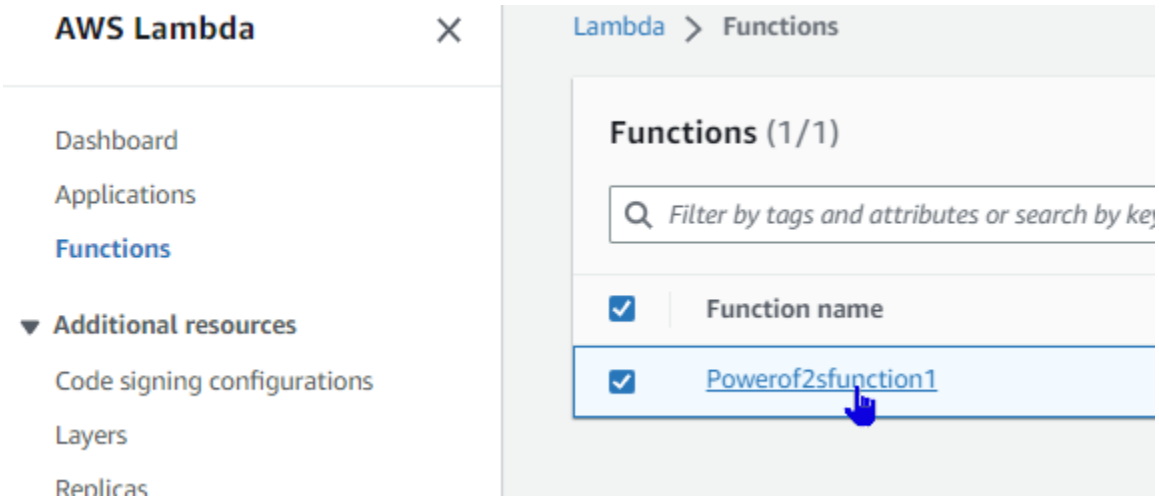
Submitted by: Emilie Dionisio



Go back to Lambda by searching or clicking your shortcut button.

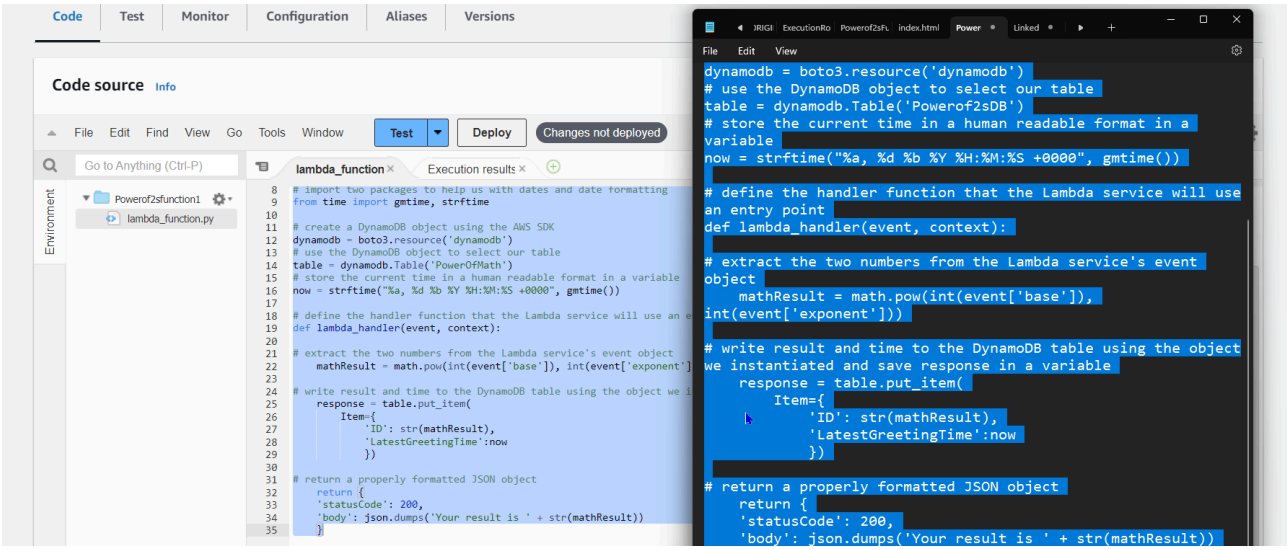


On the left navigation, click on Functions, click on the “Powerof2sfunction1”



On the Function Overview page, grab the “Powerof2sFunction - Lambda-Final.txt”, copy the entire code and paste on lambda_function code source page.

Submitted by: Emilie Dionisio



High level explanation of the code:

- Line 6: we imported AWS sdk (software development kit), we are using boto3 for Python. What is Boto3? Boto3 allows you to directly create, update, and delete AWS resources from your Python scripts.

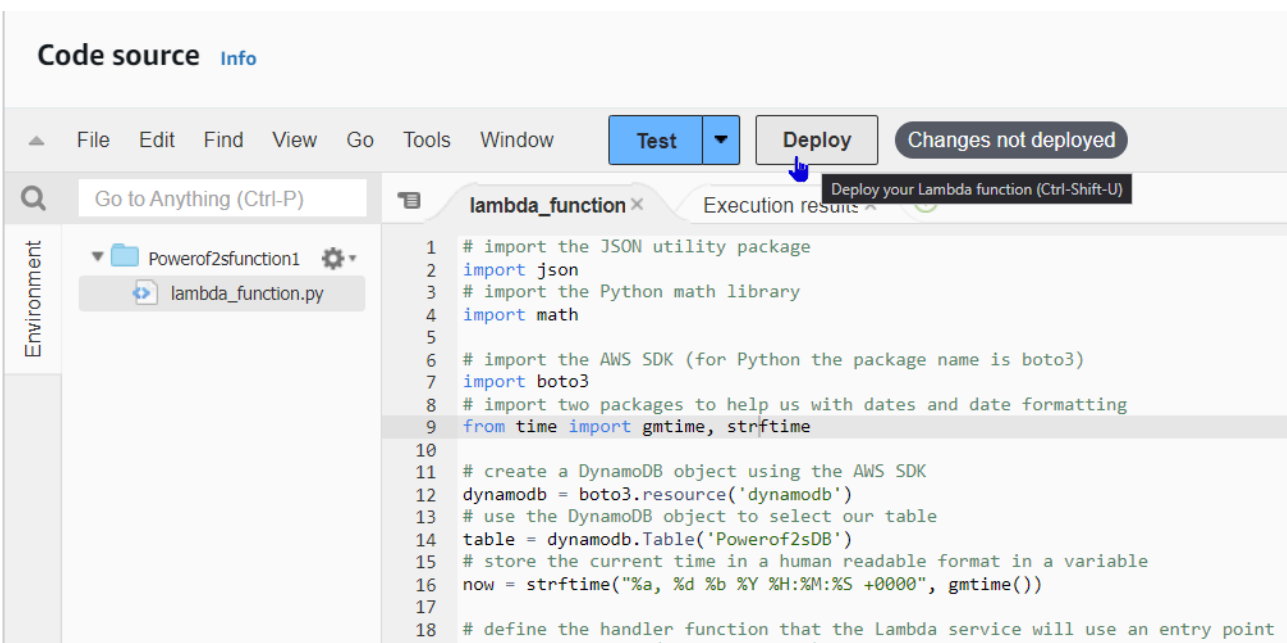
```
6  # import the AWS SDK (for Python the package name is boto3)
7  import boto3
```

- Line 8-9: Another AWS SDK package that helps with date and time formatting. When we insert the math resulto into the DynamoDB table, it will also insert the current time.

```
8  # import two packages to help us with dates and date formatting
9  from time import gmtime, strftime
```

- The rest are the same as before when we tested the calculation result.

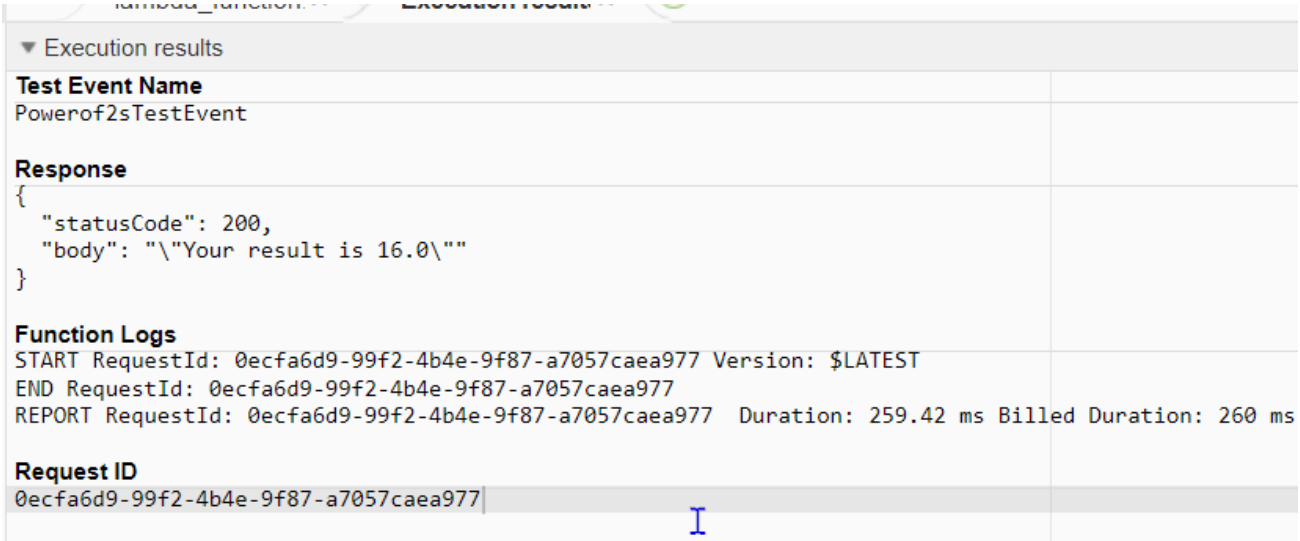
Press CTRL+S on the keyboard to save your code or click on File and save then click Deploy.



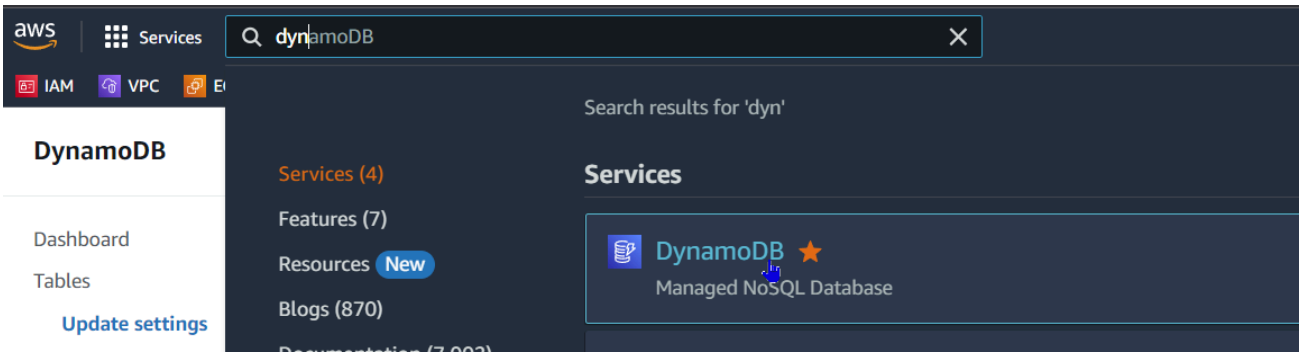
Submitted by: Emilie Dionisio

Click the “Test” button and you should still have the old math test result that we created earlier. Try to change the “Exponent” to get a different result then click “Save” button.

Click the “Test” button and you should get the right result.

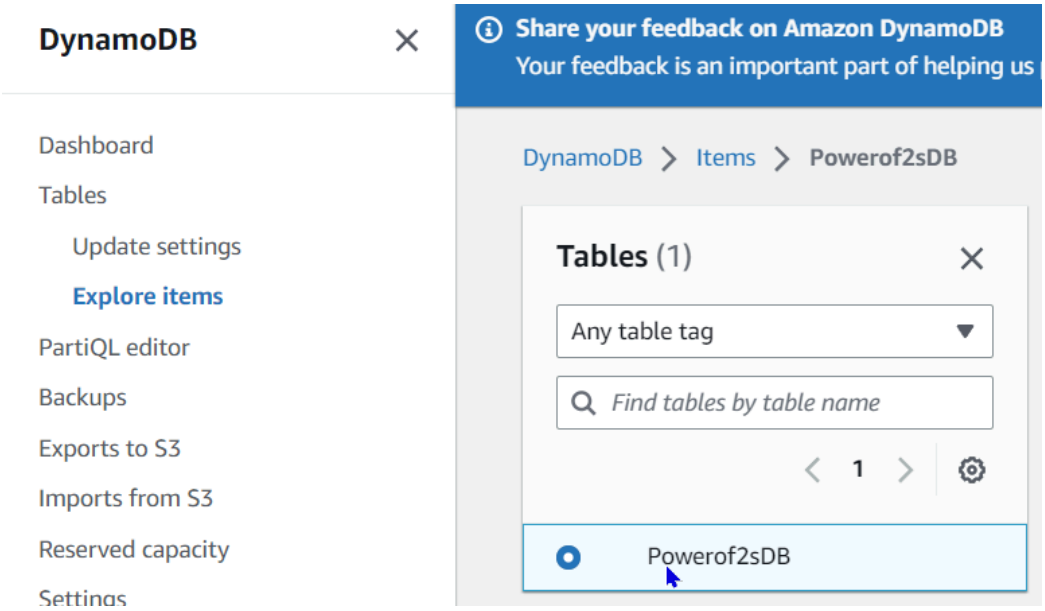


Go back to DynamoDB by clicking the shortcut button or search for it on the AWS Console Search then click on it.

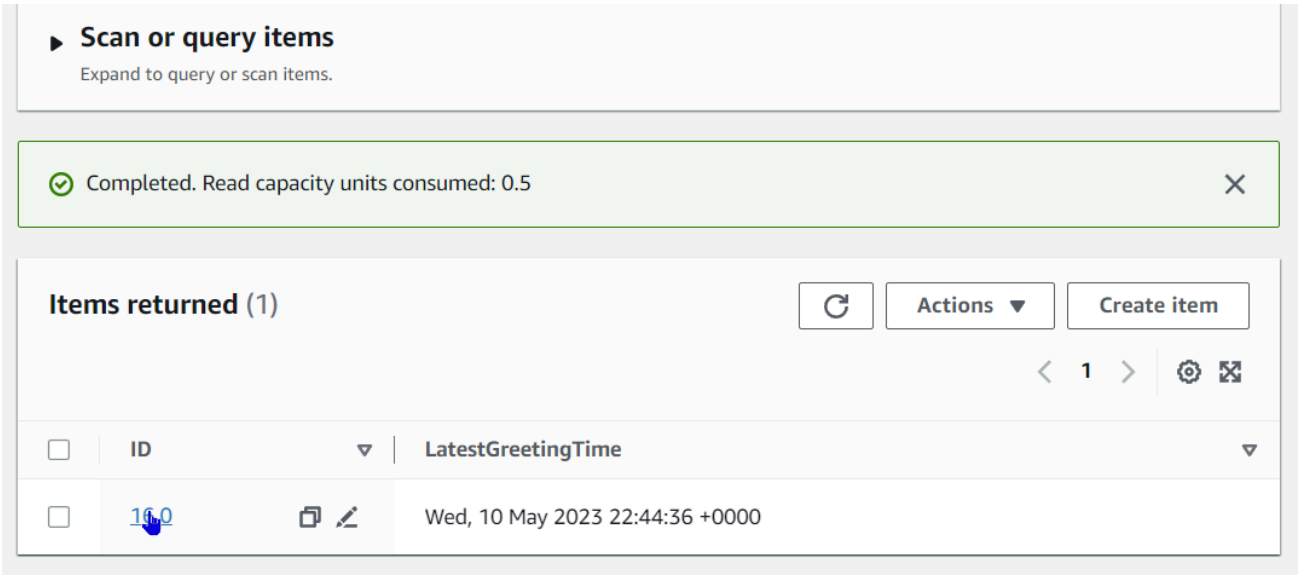


On the left navigation, click on “Explore items” then click on “Powerof2sDB”

Submitted by: Emilie Dionisio



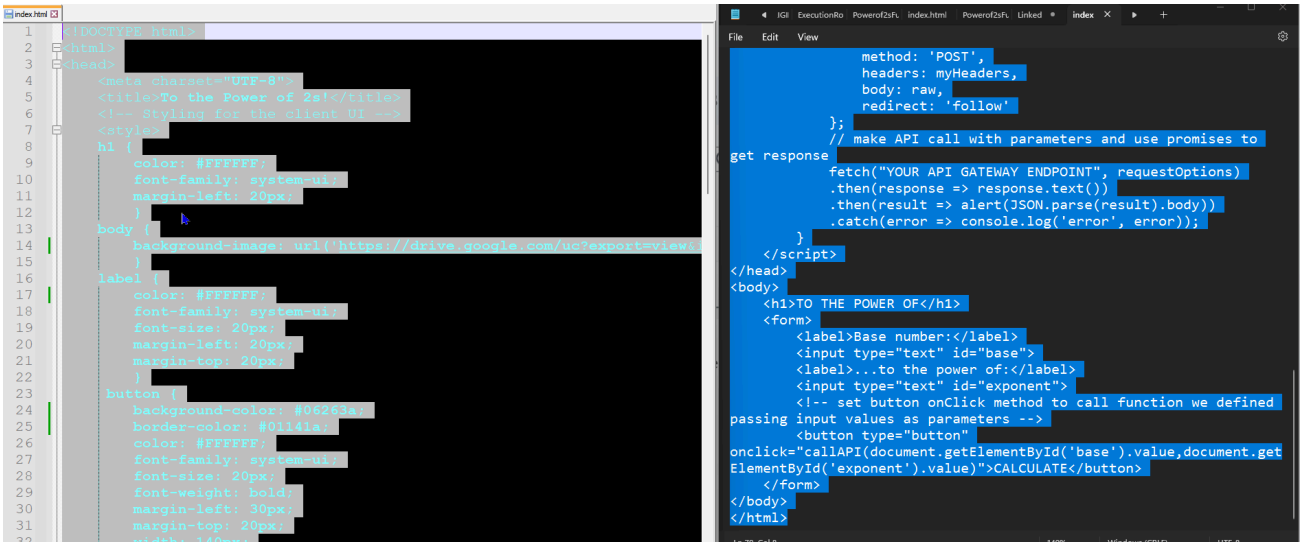
In the middle under Items returned, you will see the result of the query.



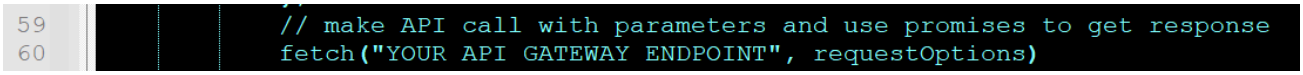
We need to update our index.html file that we created earlier. Open both index.html and the index-ORIGINAL.hmtl copies in a notepad.

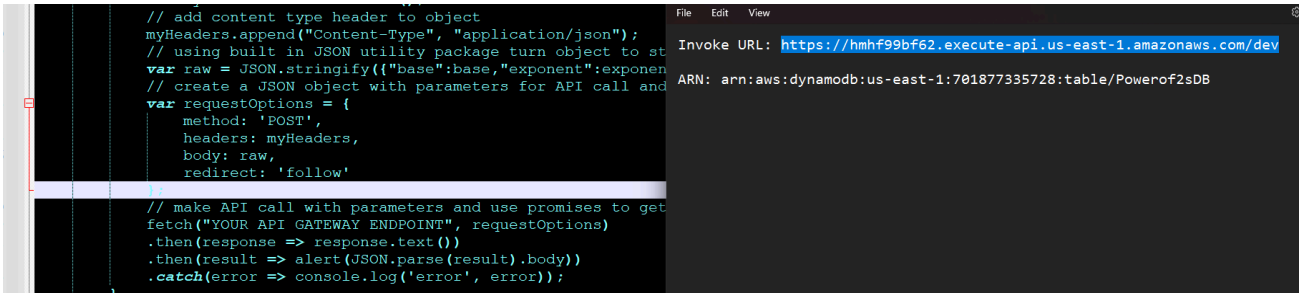
Copy the entire source from index.html to the index-ORIGINAL.html.

Submitted by: Emilie Dionisio



On the index.html file, look for “YOUR API GATEWAY” and grab the URL for the Invoke on the notepad and copy the URL on the “Your API Gateway Endpoint”.



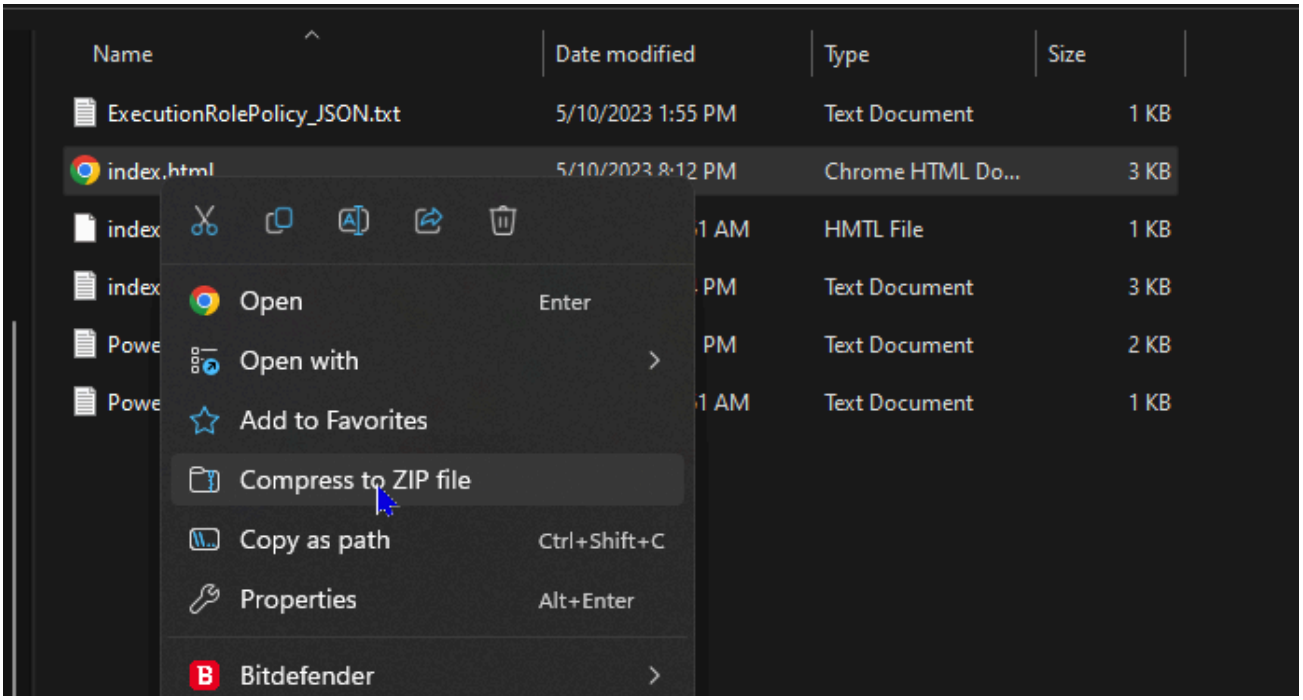


The index.html should look like this after you copied it.

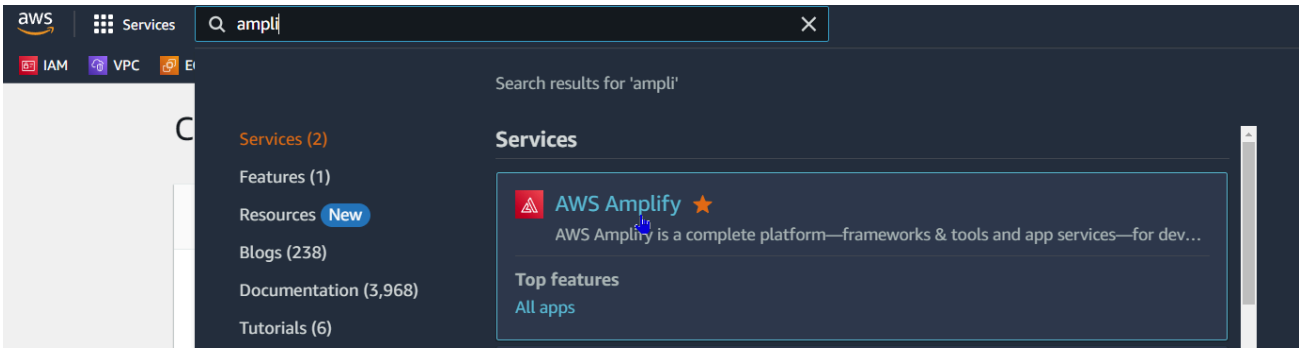
After you copy the URL, save your file.

We are now going to compress the index.html to make a zip file so we can upload it back to AWS Amplify. Right click the index.html and click on compress to Zip file.

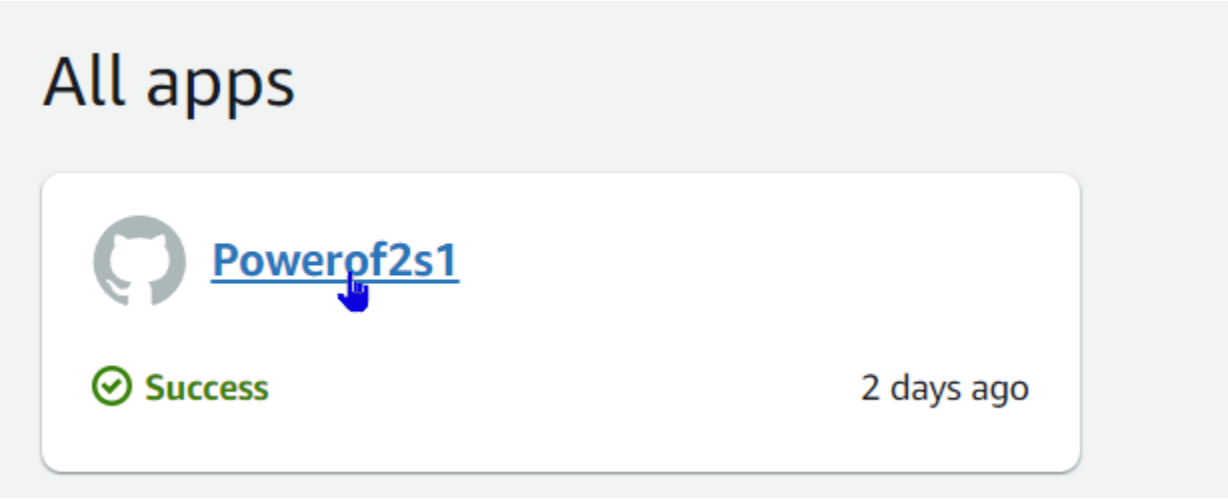
Submitted by: Emilie Dionisio



Go back to AWS Console and search for AWS Amplify.

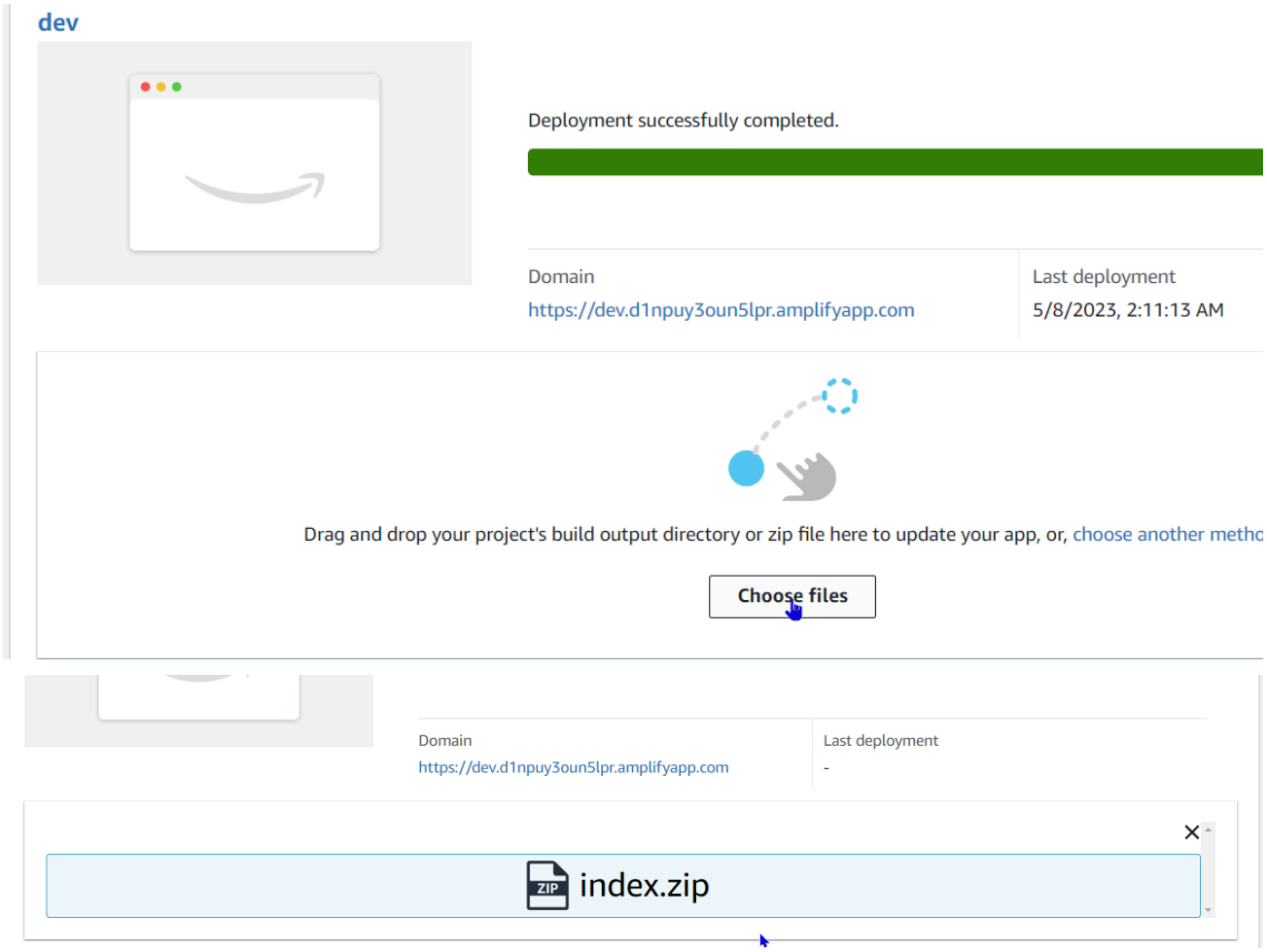


Click on “Powerof2s1”

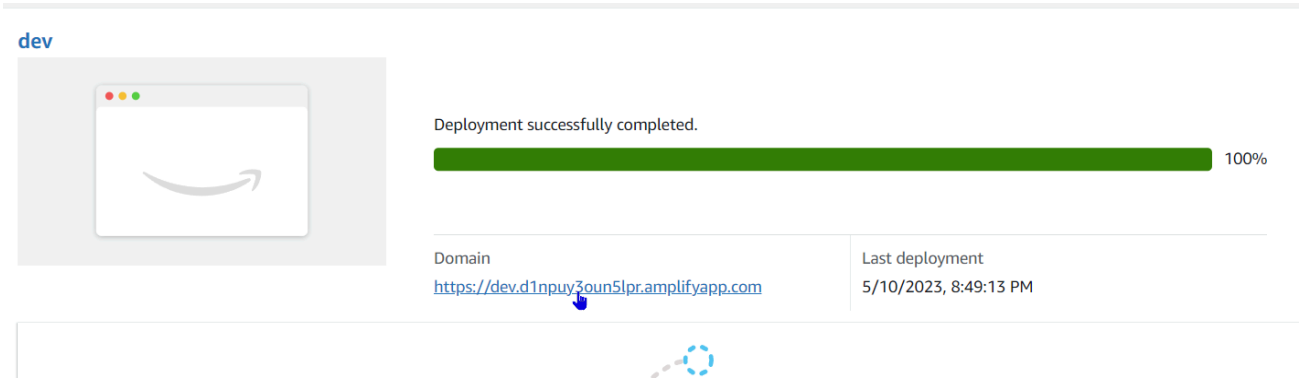


Click on “Choose files” and look for the index.zip in your local drive and upload it to AWS Amplify.

Submitted by: Emilie Dionisio

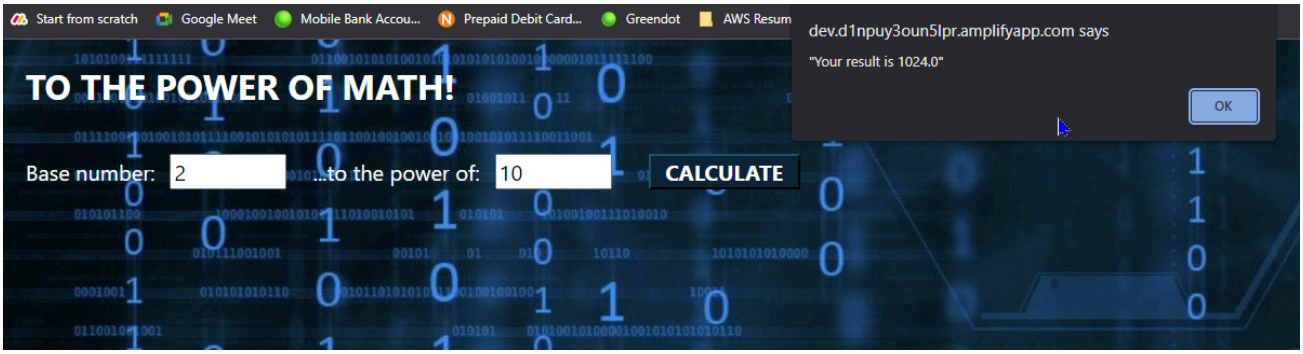


Click on the URL link to open another tab.



You can now test the calculator if it's working. You should see something like this and that means you successfully created a calculator that calculates exponents.

Submitted by: Emilie Dionisio



Demo site: https://dev.d1npuy3oun5lpr.amplifyapp.com/

Submitted by: Emilie Dionisio

Submitted by: Emilie Dionisio