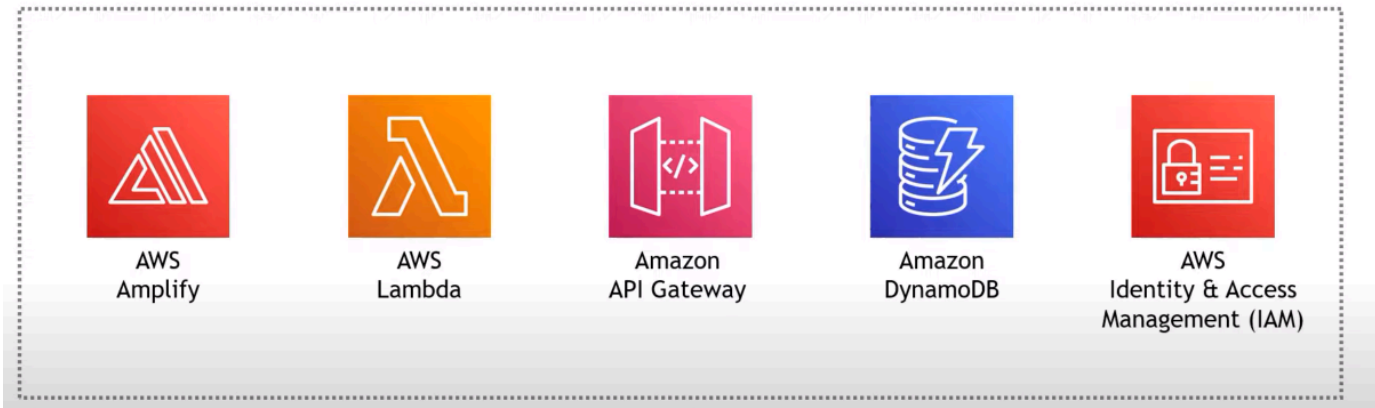


AWS Web Application using 5 services in AWS

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.

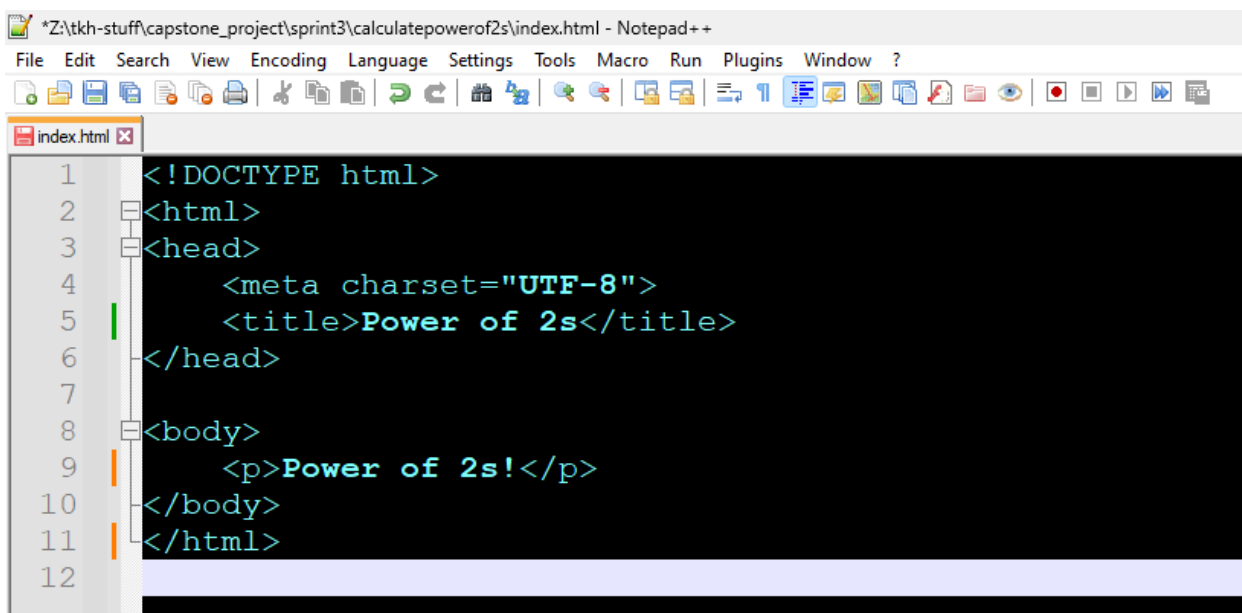
Submitted by: Emilie Dionisio

- 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:

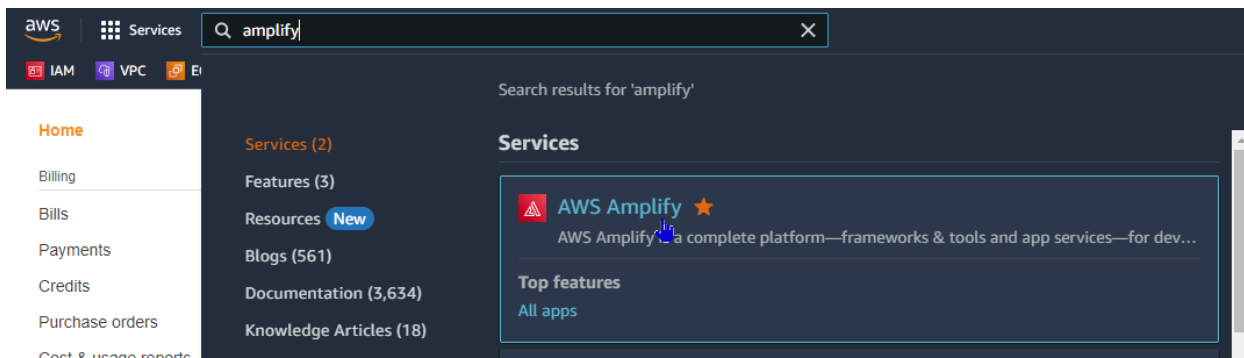
- Create a webpage host
 - Amplify - is used to build and host a websites
- Invoke a math function
- Calculate the math
- Store and return the math result to the user
- Handle permissions
- Text Editor (notepad++, visual studio)
- AWS account
- Basic knowledge of AWS

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.

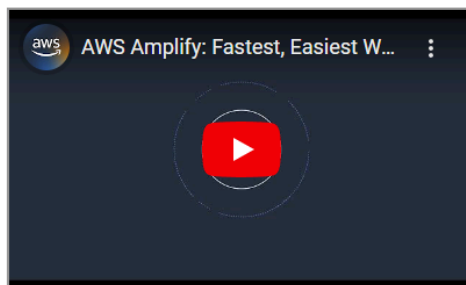
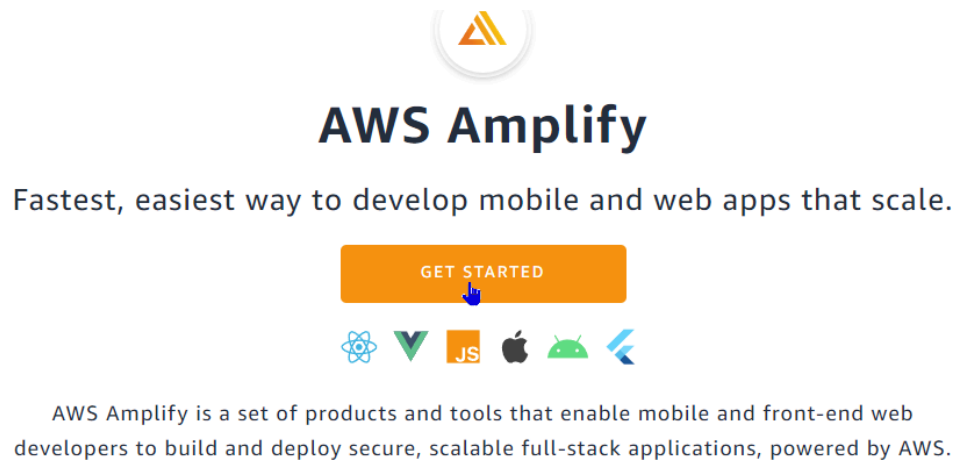


```
*Z:\tkh-stuff\capstone_project\sprint3\calculatepowerof2s\index.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>Power of 2s</title>
6 </head>
7
8 <body>
9     <p>Power of 2s!</p>
10 </body>
11 </html>
12
```

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



In AWS Amplify page, select “Get Started” button



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

Get started

Amplify Studio



Build an app

Build an app backend with auth, data, and storage, and create custom UI components. Then integrate them in your app with just a few steps.



Get started

Amplify Hosting



Host your web app

Connect your Git repository to continuously deploy your frontend and backend. Host it on a globally available CDN.



Get started

Already have existing Cognito, S3, or other AWS resources? Connect to them from your app with the Amplify Libraries. [Go to docs](#)

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

Get started with Amplify Hosting

Amplify Hosting is a fully managed hosting service for web apps. Connect your repository to build, deploy, and host your web app.

From your existing code

Connect your source code from a Git repository or upload files to host a web app in minutes.

☐ GitHub



☐ Bitbucket



☐ GitLab



☐ AWS CodeCommit



☒ Deploy without Git provider



Continue

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.

Manual deploy

Manually upload objects to deploy your app. You can choose to drag and drop the artifacts directly, pull a zip from an existing S3 bucket or any other URL.

Start a manual deployment

App name

Give this app a name or we will generate a default for you

Powerof2s1

Environment name

Give this resource a meaningful environment name, like dev, test, or prod, or we will generate a default for you

dev

Method

☒ Drag and drop



☐ Amazon S3



☐ Any URL



index.zip



Cancel

Previous

Save and deploy

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

All apps > Powerof2s1

Powerof2s1

Actions

The app homepage lists all deployed frontend and backend environments.

Hosting environments

Backend environments

This tab lists all connected branches, select a branch to view build details.

Add environment

dev



Deployment successfully completed.



Domain

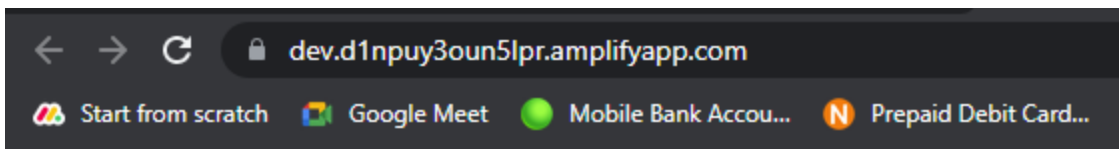
<https://dev.d1npuy3oun5lpr.amplifyapp.com>

Last deployment

5/8/2023, 2:11:13 AM

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.

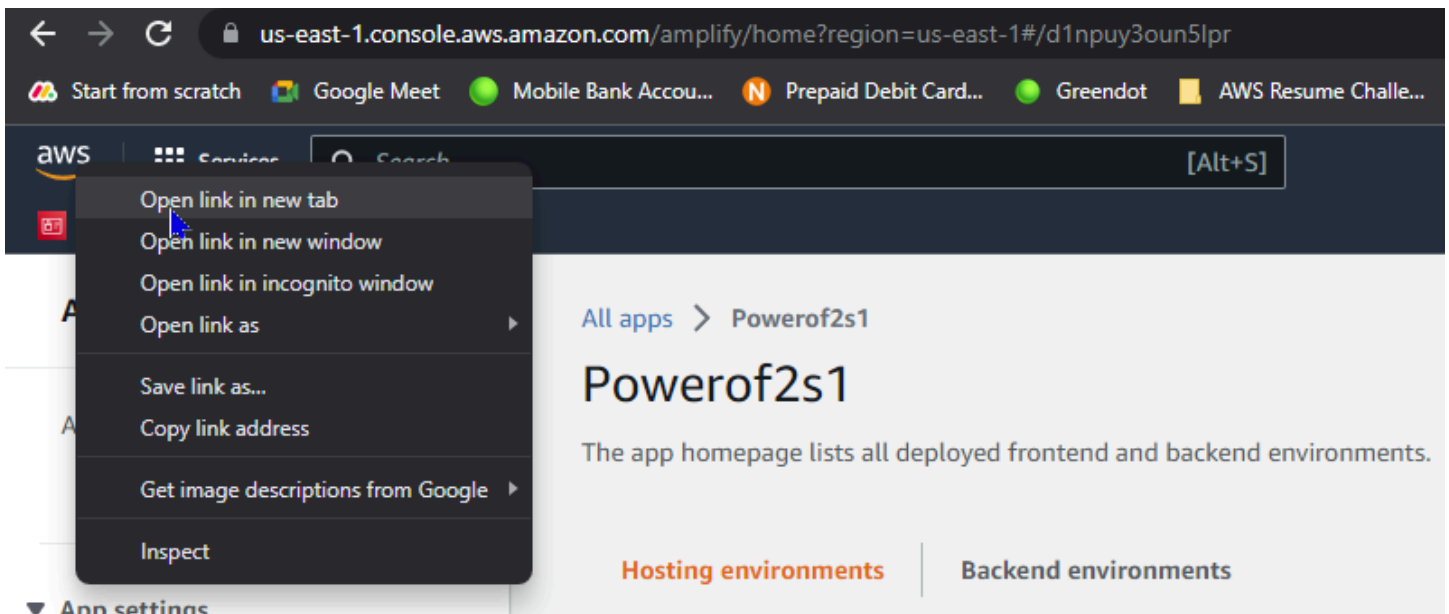
Submitted by: Emilie Dionisio



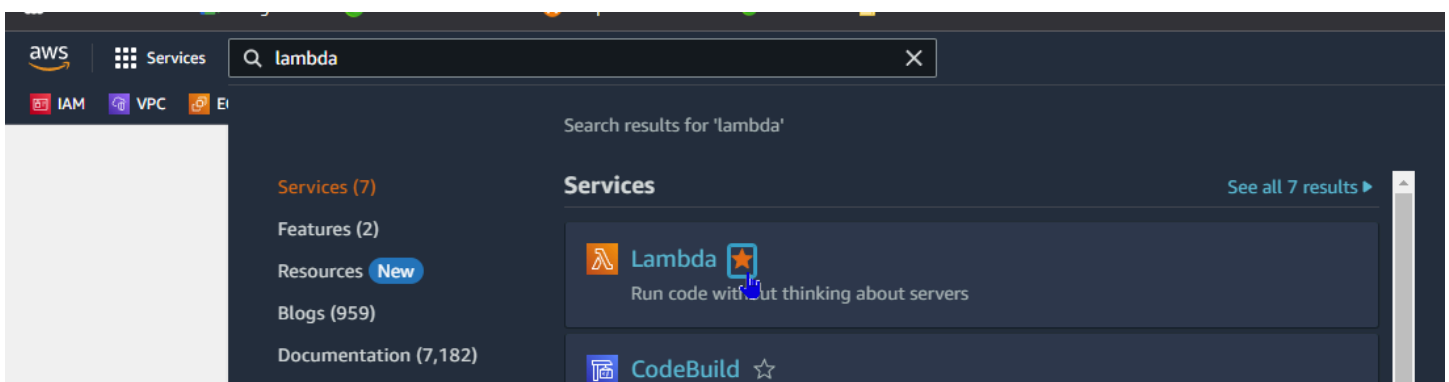
Power of 2s!

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.

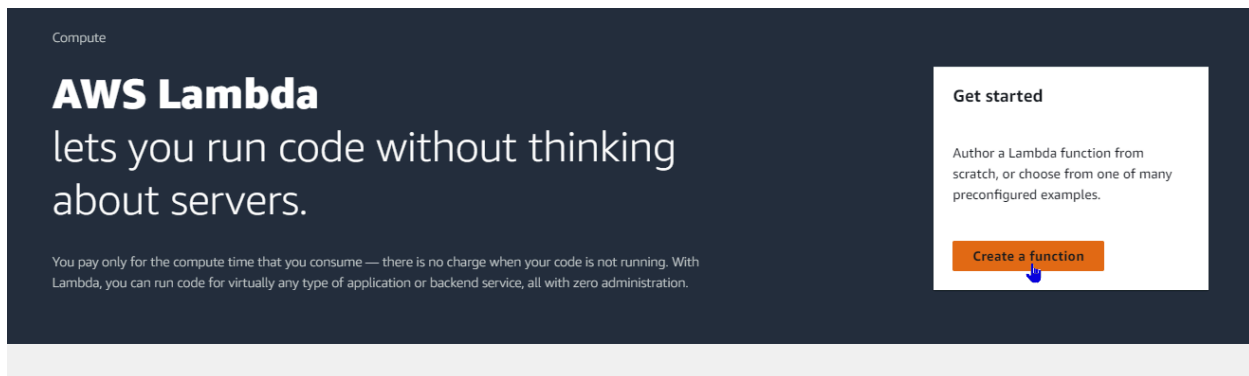
On AWS Console, right click the AWS logo and click on "Open link in new tab" to keep Amplify handy.



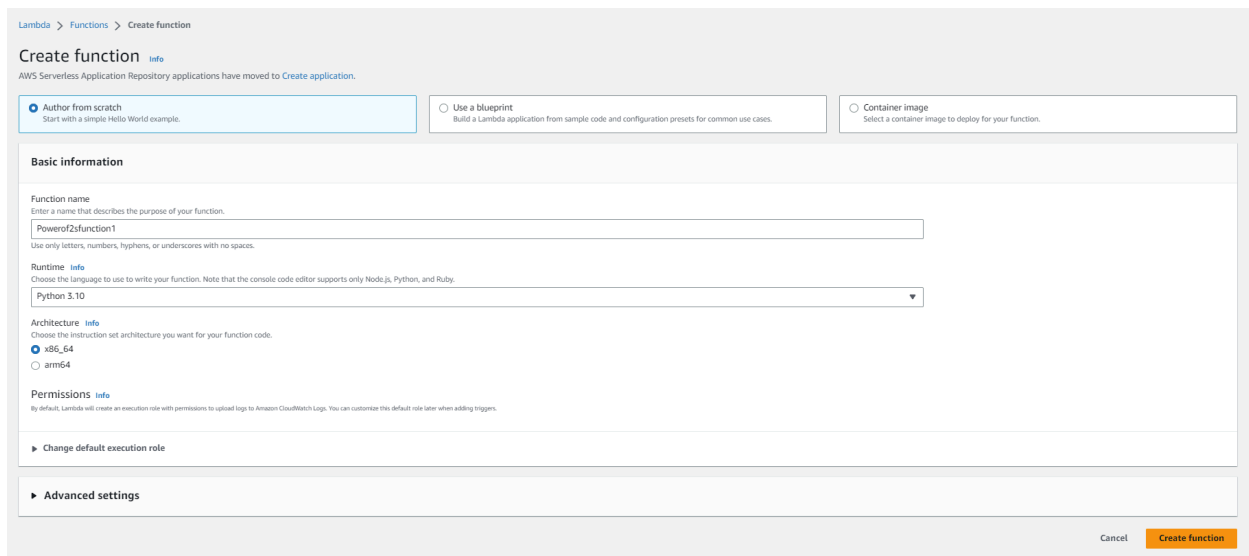
In the AWS console search for Lambda and click it.



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 version of Python for Runtime, in this case I chose Python 3.10 then click “Create function” button.

The image is a screenshot of the AWS Lambda "Create function" console. At the top, it shows the breadcrumb "Lambda > Functions > Create function". The main heading is "Create function" with an "info" link. Below this, a note states: "AWS Serverless Application Repository applications have moved to Create application." There are three tabs: "Author from scratch" (selected, with subtext "Start with a simple Hello World example."), "Use a blueprint" (with subtext "Build a Lambda application from sample code and configuration presets for common use cases."), and "Container image" (with subtext "Select a container image to deploy for your function."). The "Basic information" section contains: "Function name" with a text input field containing "Powerof2sfunction1" and a note "Enter a name that describes the purpose of your function. Use only letters, numbers, hyphens, or underscores with no spaces."; "Runtime" with a dropdown menu set to "Python 3.10" and a note "Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby."; "Architecture" with two radio buttons, "x86_64" (selected) and "arm64", with a note "Choose the instruction set architecture you want for your function code."; and "Permissions" with a note "By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers." Below these are expandable sections for "Change default execution role" and "Advanced settings". At the bottom right, there are "Cancel" and "Create function" buttons.

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.

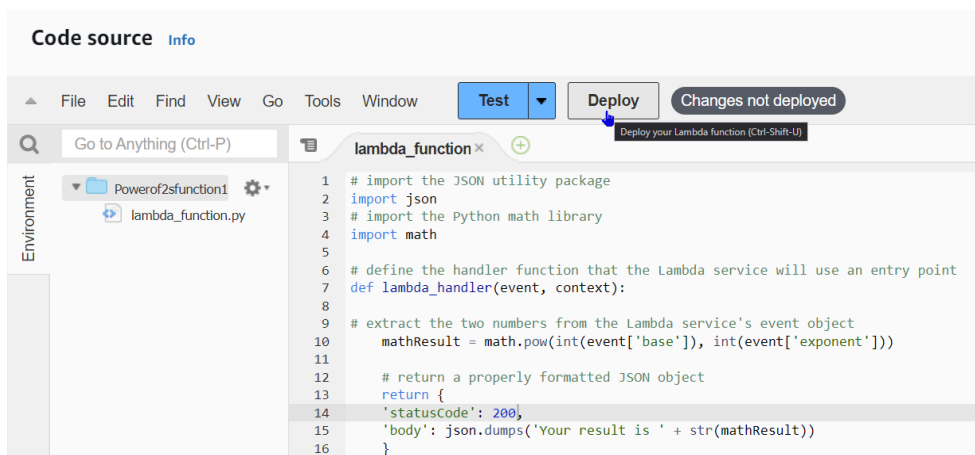
```
# import the JSON utility package
import json
# import the Python math library
import math

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

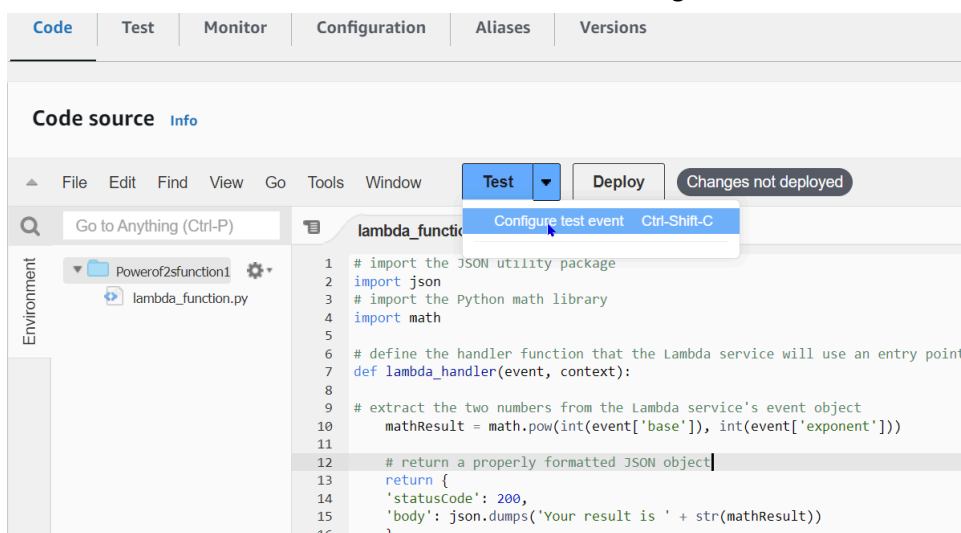
# extract the two numbers from the Lambda service's event object
    mathResult = math.pow(int(event['base']), int(event['exponent']))

    # 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"

Event JSON

Format JSON

1 {
2 "key1": "value1",
3 "key2": "value2",
4 "key3": "value3"
5 }

Delete this line

It should look like this:

Test event action

☒ Create new event

☐ Edit saved event

Event name

Powerof2sTestEvent

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

Powerof2sTestEvent

Event JSON

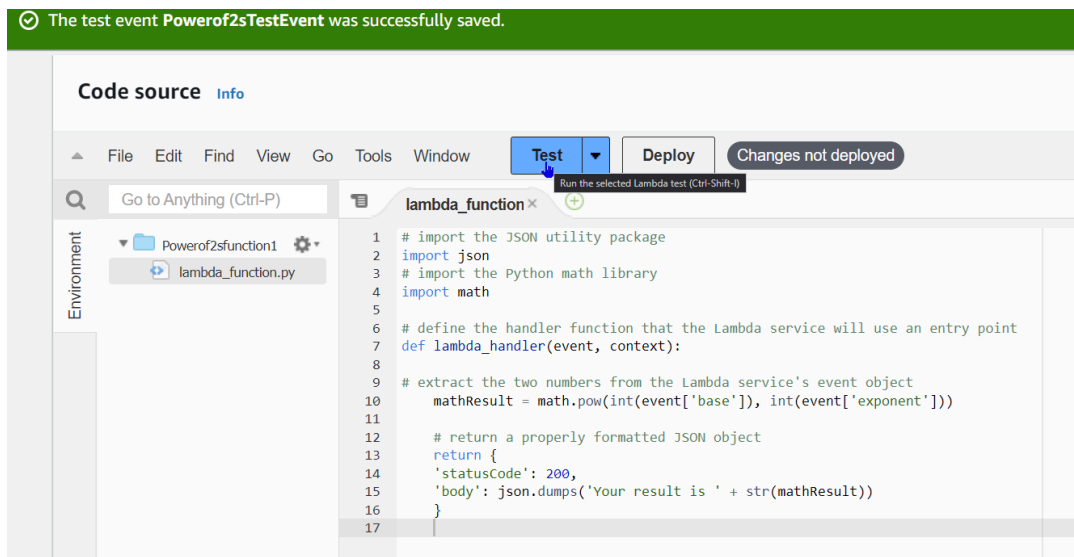
Format JSON

1 {
2 "base": 2,
3 "exponent": 3
4 }
5 }

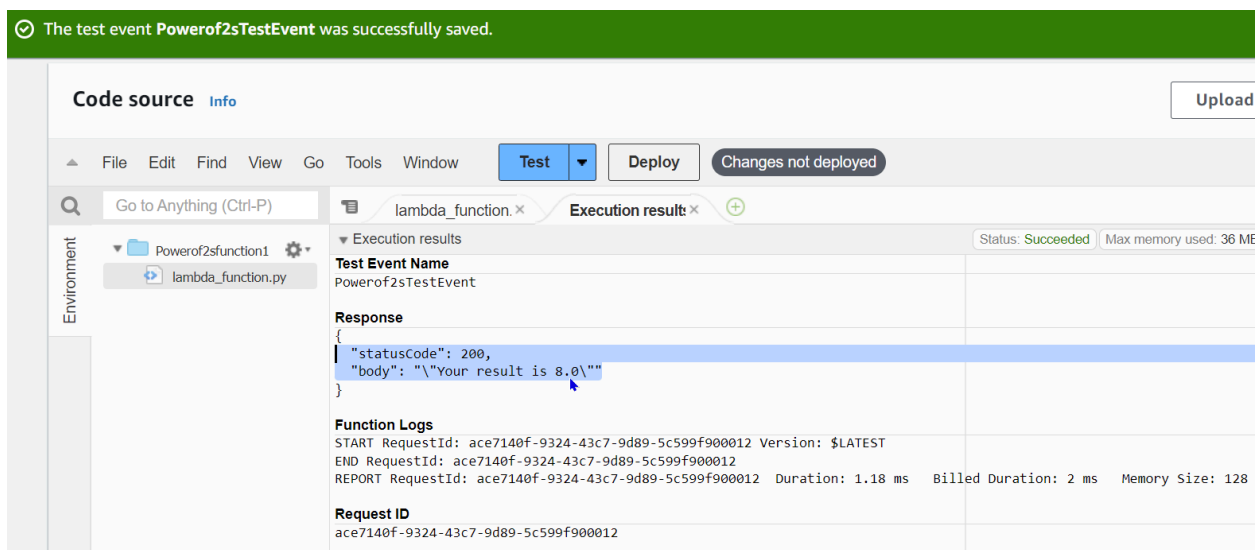
Cancel

Save

To run the test, click the "Test" button.

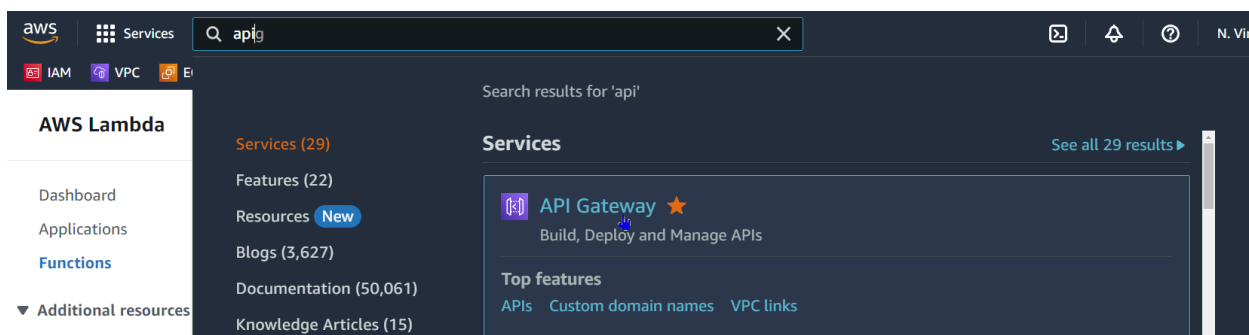


You know that the Lambda function is working when you see “200” for status code and “8” for the result which is (2^3).



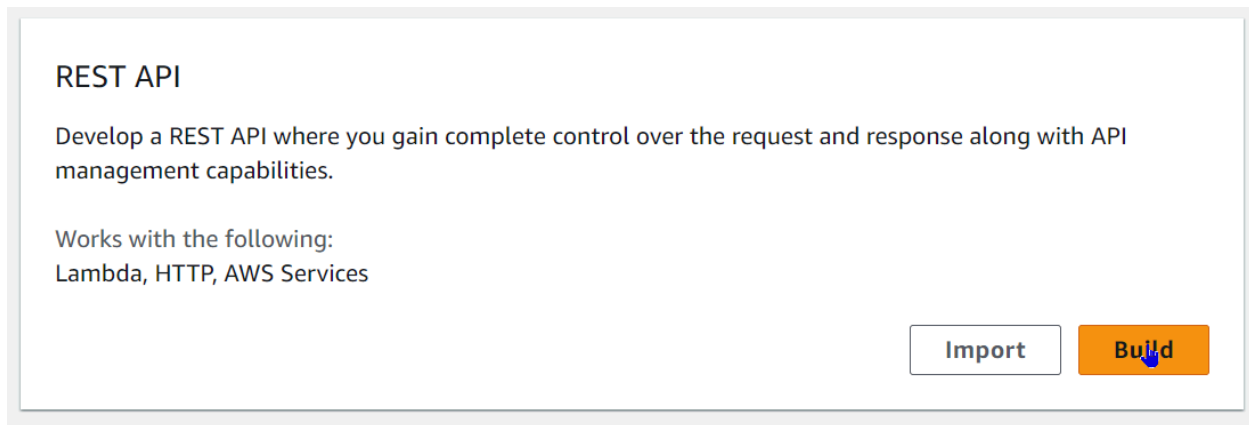
Invoking the math functionality

In AWS Console, search for API Gateway



Submitted by: Emilie Dionisio

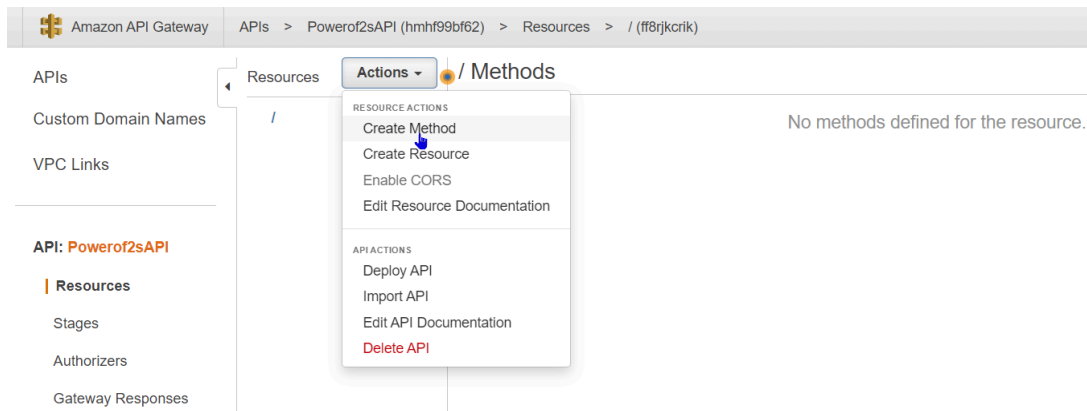
In choose an API type, select Rest API then click “Build” button.



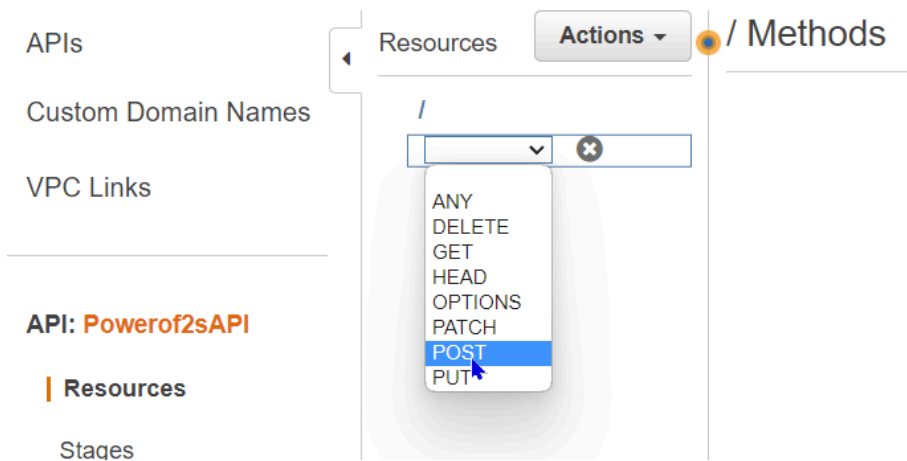
In Choose the protocol, leave the Rest radio button selected and select “New API” under Create new API, give it a name, and press “Create API” button.

A screenshot of the 'Create new API' form in the AWS console. The breadcrumb trail at the top is 'Amazon API Gateway > APIs > Create'. The page has three main sections: 'Choose the protocol' with radio buttons for 'REST' (selected) and 'WebSocket'; 'Create new API' with radio buttons for 'New API' (selected), 'Import from Swagger or Open API 3', and 'Example API'; and 'Settings' with input fields for 'API name*' (containing 'Powerof2sAPI'), 'Description', and a dropdown for 'Endpoint Type' (set to 'Regional'). A 'Create API' button is at the bottom right. A legend at the bottom left indicates '* Required'.

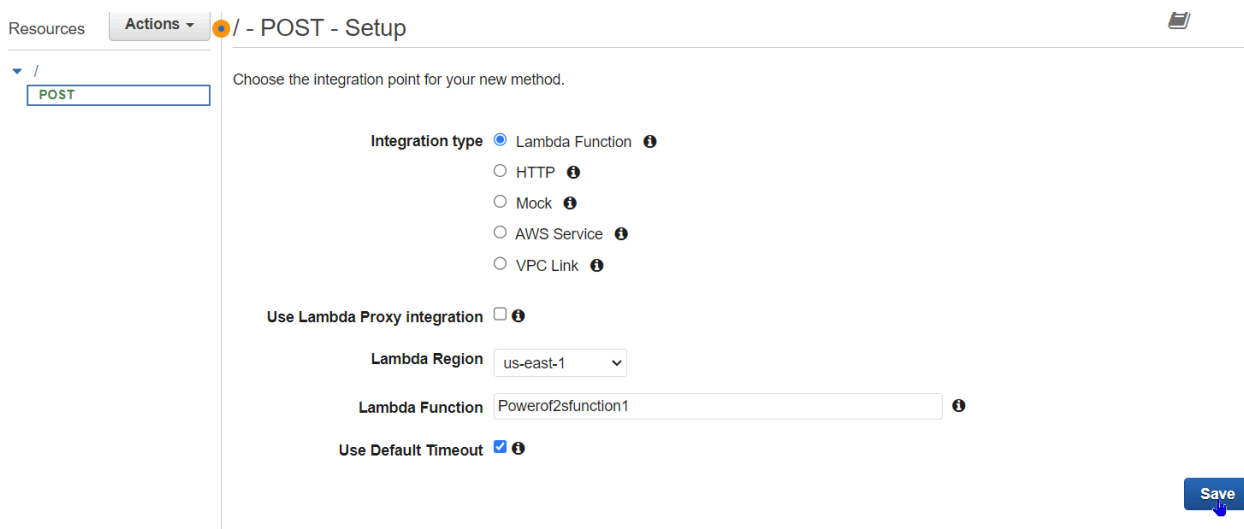
On the API Methods page, on the left navigation, make sure that “Resources” is selected and in the middle make sure that the backslash (/) is highlighted. Click on the Actions menu and select “Create Method”.



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

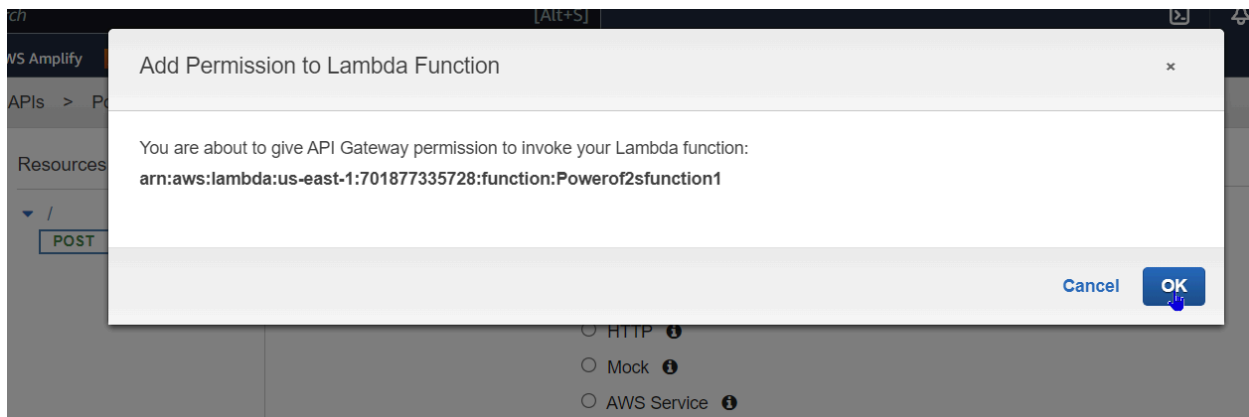


For integration type, choose “Lambda Function” then under Lambda Function choose the function that was created before which was the “Powerof2sfunction1” function then click “Save”.

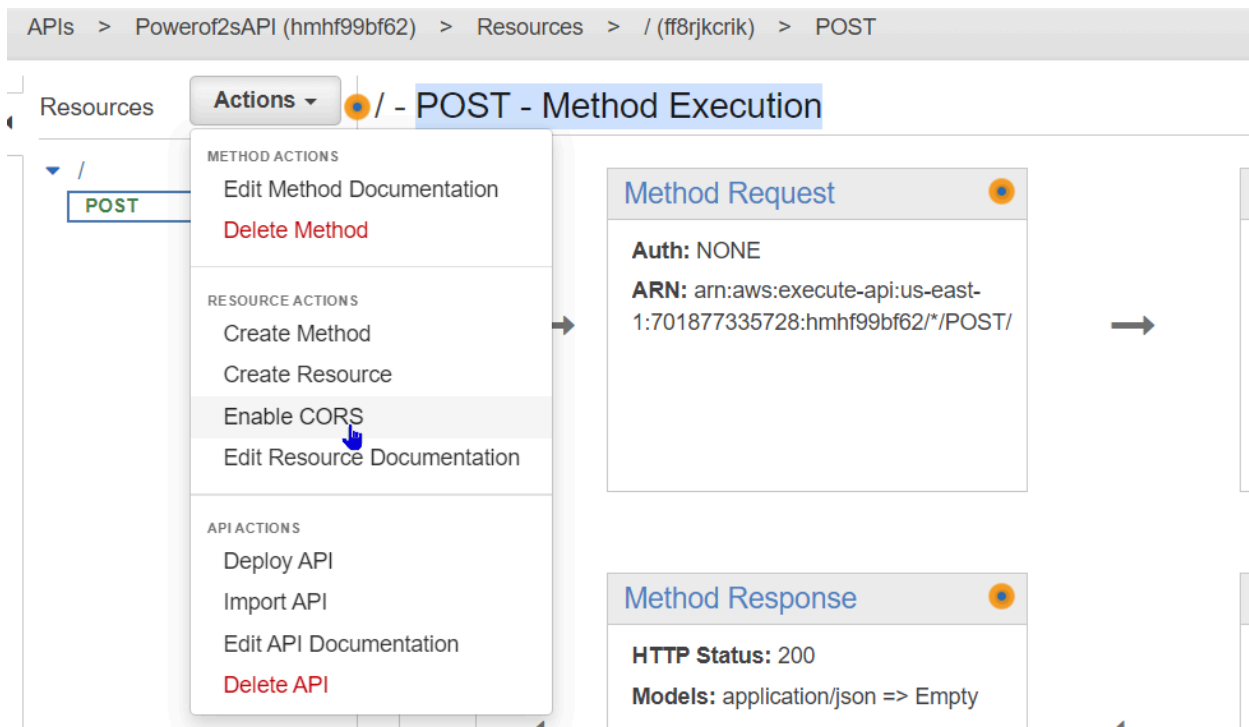


Click Ok when you see “Add Permission to Lambda Function”. It states that we are giving API Gateway a permission to invoke this Lambda function.

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.

Resources **Actions** **Enable CORS**

POST

Gateway Responses for *Powerof2sAPI API* ☐ DEFAULT 4XX ☐ DEFAULT 5XX ⓘ

Methods ☒ POST ☐ OPTIONS ⓘ

Access-Control-Allow-Methods OPTIONS, POST ⓘ

Access-Control-Allow-Headers 'Content-Type,X-Amz-Date,Authorization' ⓘ

Access-Control-Allow-Origin* "" ⓘ ⚠

▶ Advanced

Enable CORS and replace existing CORS headers

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

Confirm method changes ✕

The following modifications will be made to this resource's methods and will replace any existing values. Are you sure you want to continue?

- Create **OPTIONS** method
- Add **200 Method Response** with **Empty Response Model** to **OPTIONS** method
- Add **Mock Integration** to **OPTIONS** method
- Add **200 Integration Response** to **OPTIONS** method
- Add **Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers** to **OPTIONS** method
- Add **Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings** to **OPTIONS** method
- Add **Access-Control-Allow-Origin Method Response Header** to **POST** method
- Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **POST** method

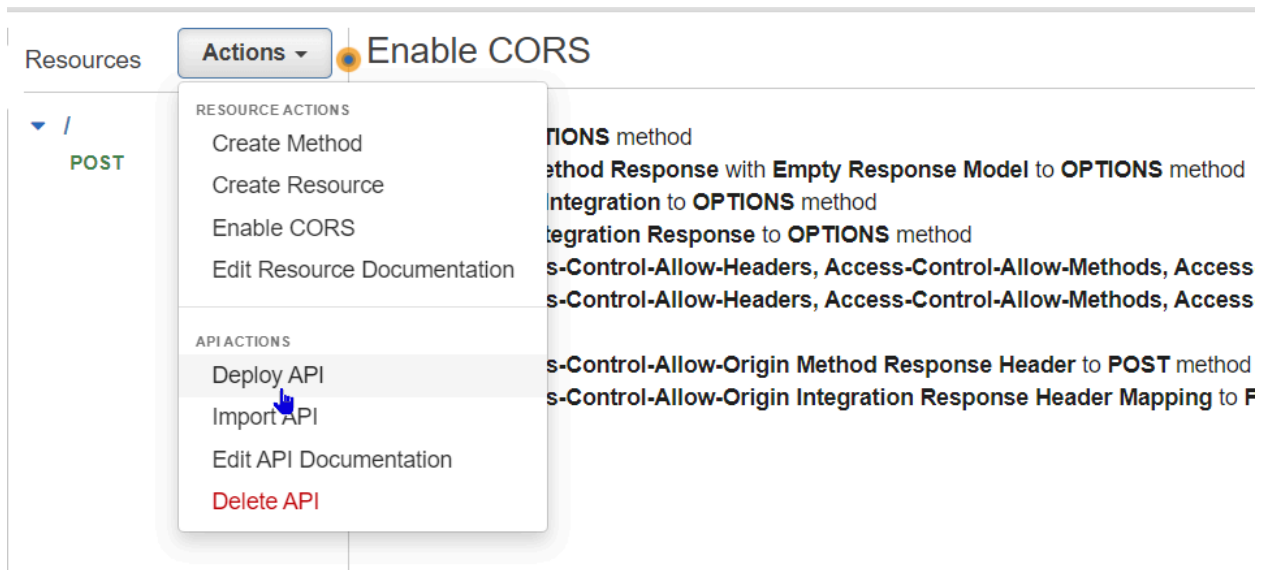
Cancel **Yes, replace existing values**

Resources **Actions** **Enable CORS**

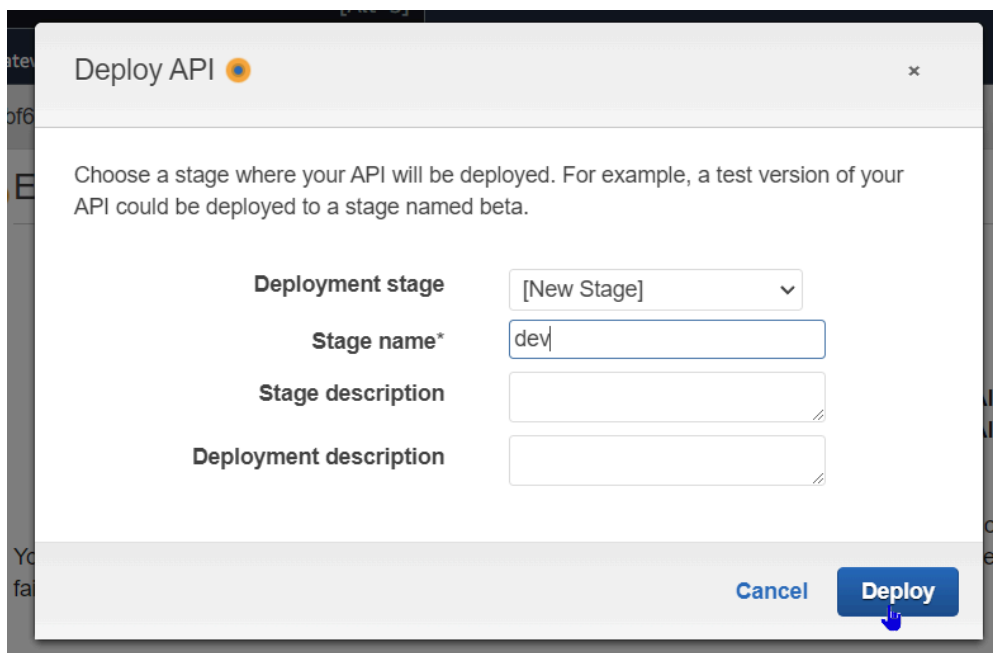
POST

- ✓ Create **OPTIONS** method
- ✓ Add **200 Method Response** with **Empty Response Model** to **OPTIONS** method
- ✓ Add **Mock Integration** to **OPTIONS** method
- ✓ Add **200 Integration Response** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Origin Method Response Header** to **POST** method
- ✓ Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **POST** method

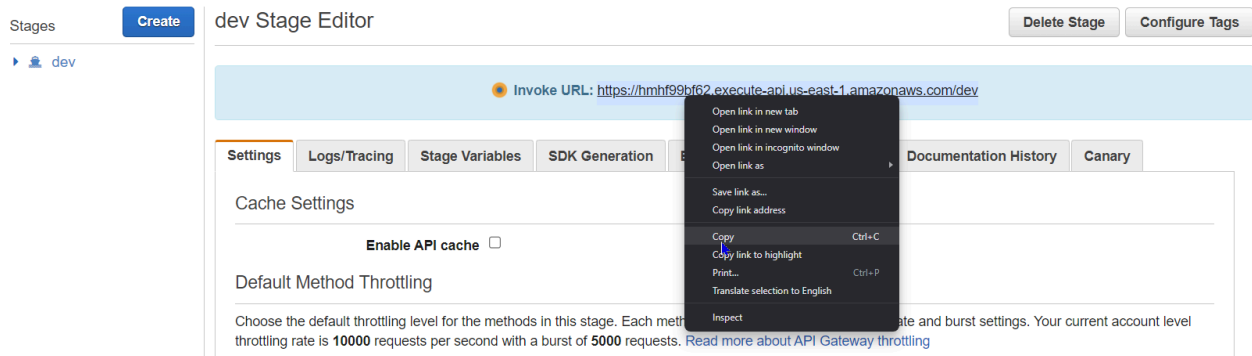
Now, we can deploy the API to test it out. Click on Actions and select “Deploy API”



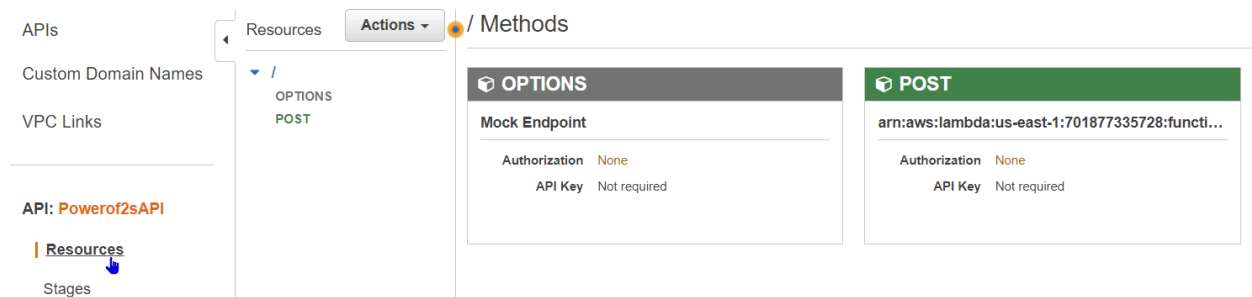
On the Deploy API pop up menu, select [New Stage] under Deployment stage, and under Stage name, type “dev” then click “Deploy” button.



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



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.

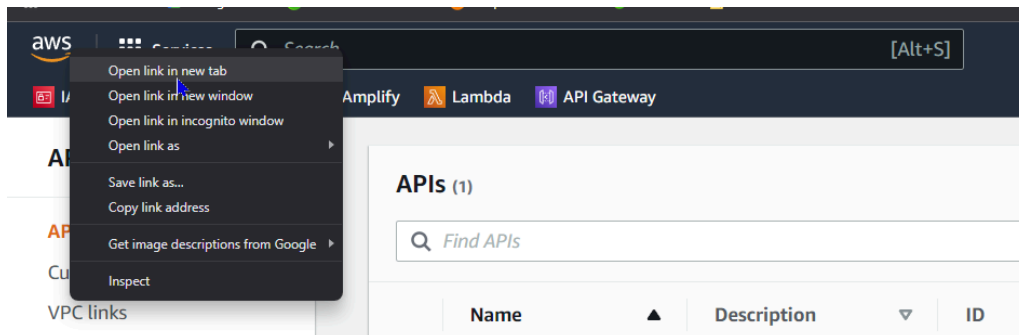
We are going to persist or store that math result somewhere like a database and return it to the user to see the result.

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.

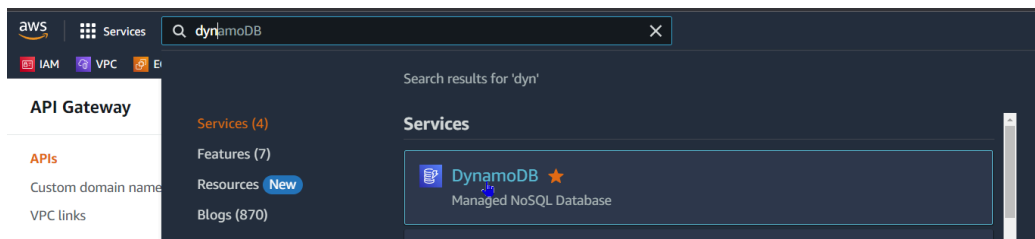
We are going to need to handle permissions between the different parts of the application. For this section we are going to use the following:

- 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

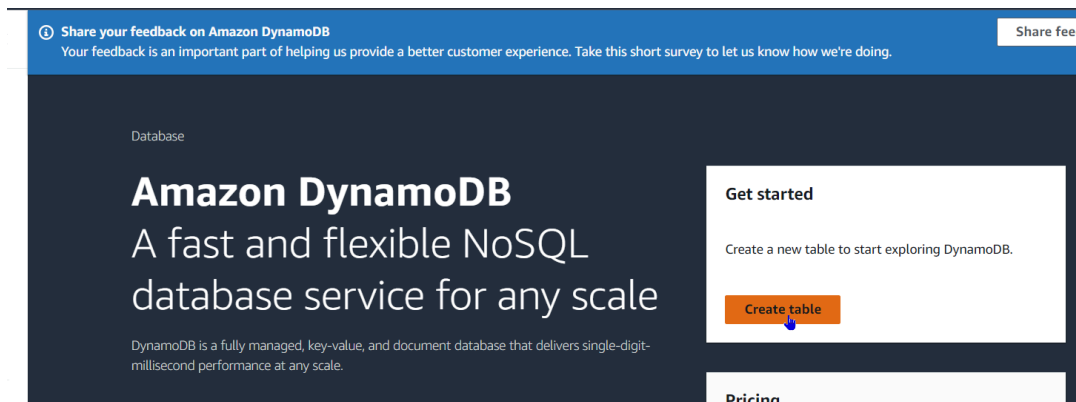
On the AWS Console, right click the AWS logo on the top left and click "Open link in new tab" so we can use it when creating DynamoDB.



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

DynamoDB > Tables > Create table

Create table

Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

Powerof2sDB

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

ID

String

1 to 255 characters and case sensitive.

Tags
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#) [Create table](#)

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.

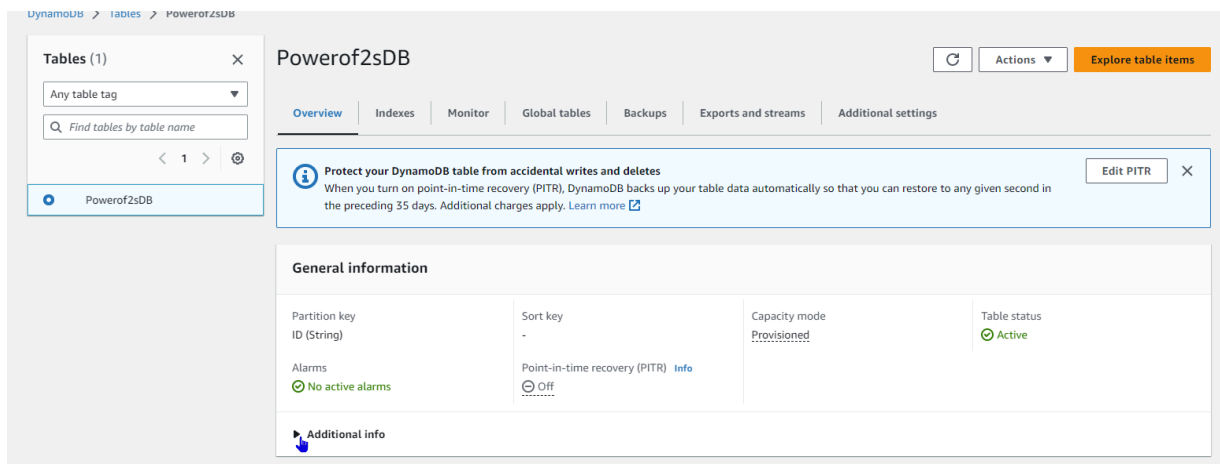
[Share your feedback on Amazon DynamoDB](#)
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

DynamoDB > Tables

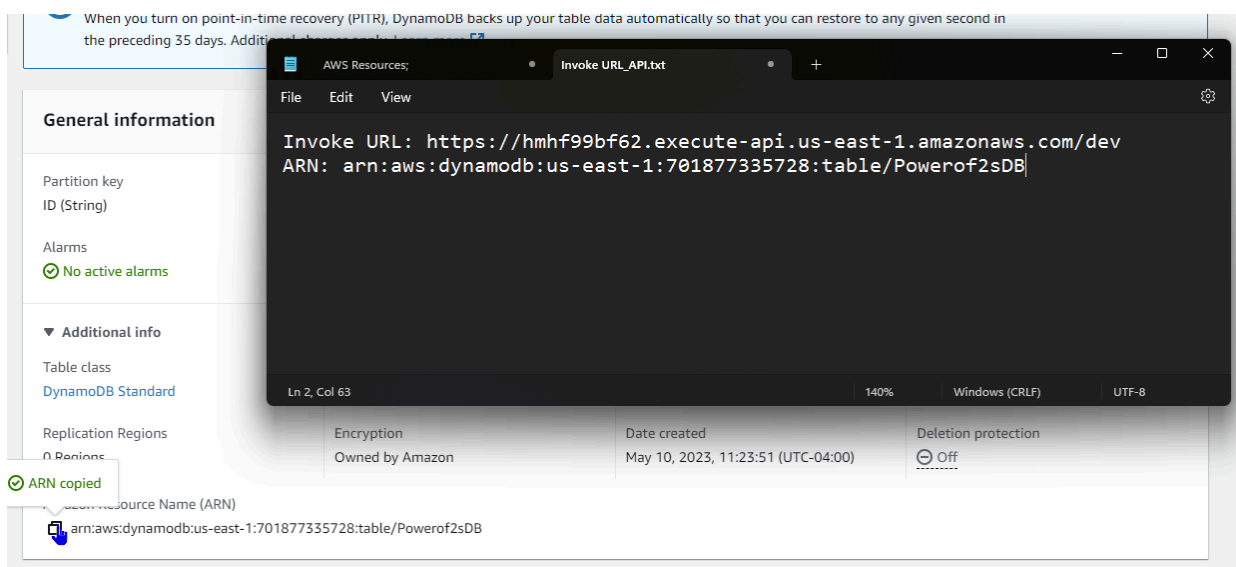
Tables (1/1) [Info](#)

| <input checked="" type="checkbox"/> | Name | Status | Partition key | Sort key | Indexes | Deletion protection | Read capacity mode |
|-------------------------------------|-----------------------------|--------|---------------|----------|---------|---------------------|-----------------------------------|
| <input checked="" type="checkbox"/> | Powerof2sDB | Active | ID (S) | - | 0 | Off | Provisioned with auto scaling (5) |

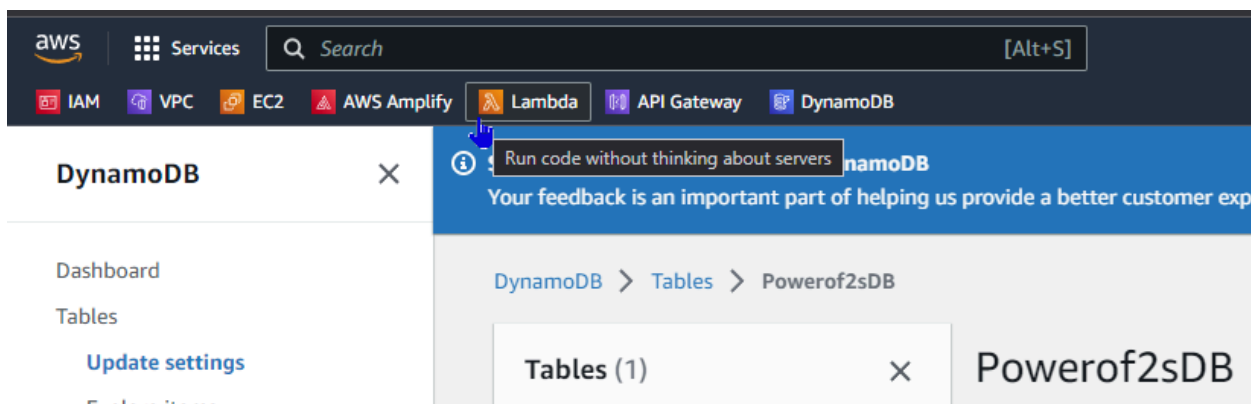
On the details page, click on the “Additional Info”



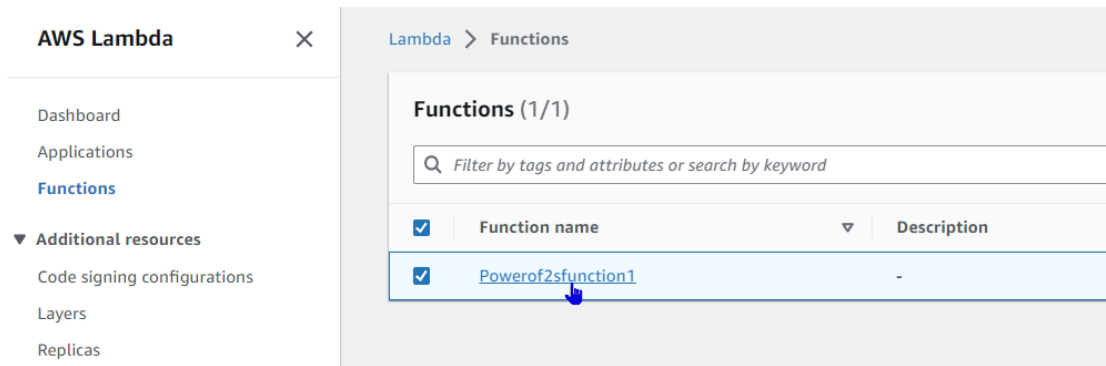
Copy the ARN (Amazon Resource Name) in a notepad together with the Invoke URL that you copied before.



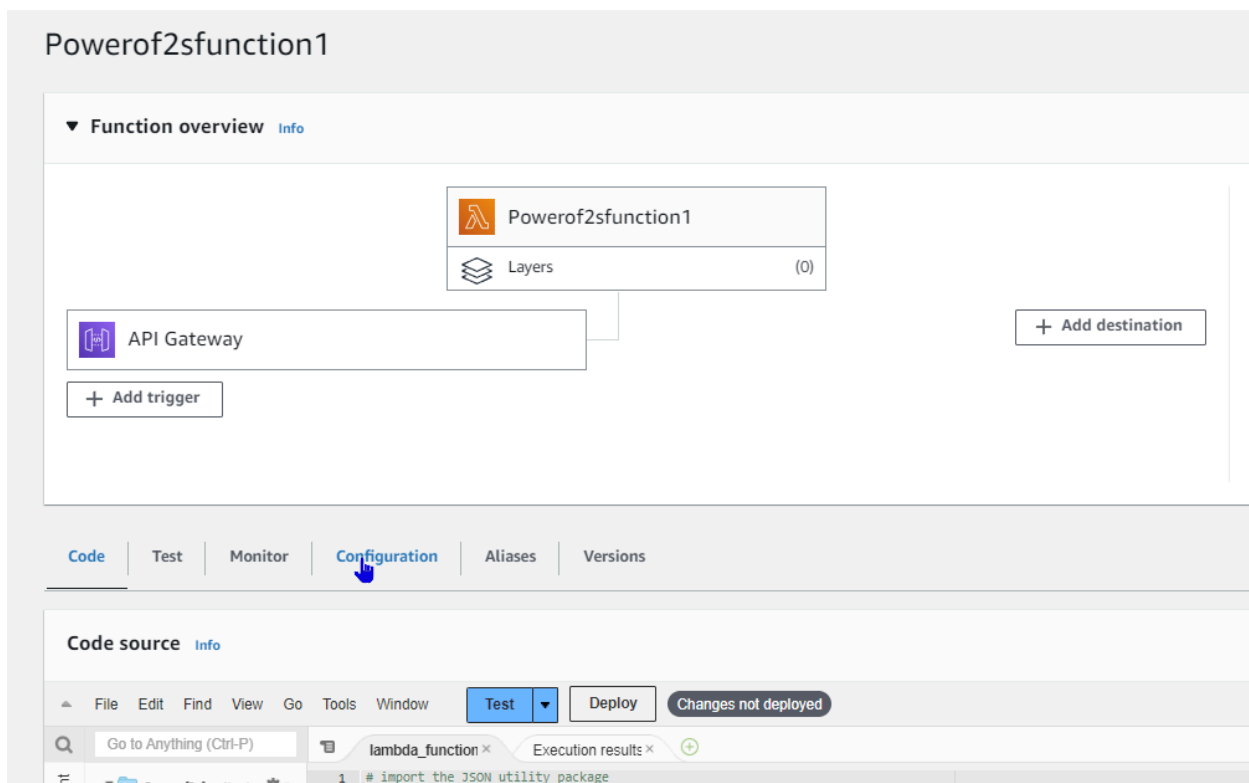
We need our Lambda function to have permission to write to our table. Go back to Lambda by clicking the shortcut or search for Lambda.



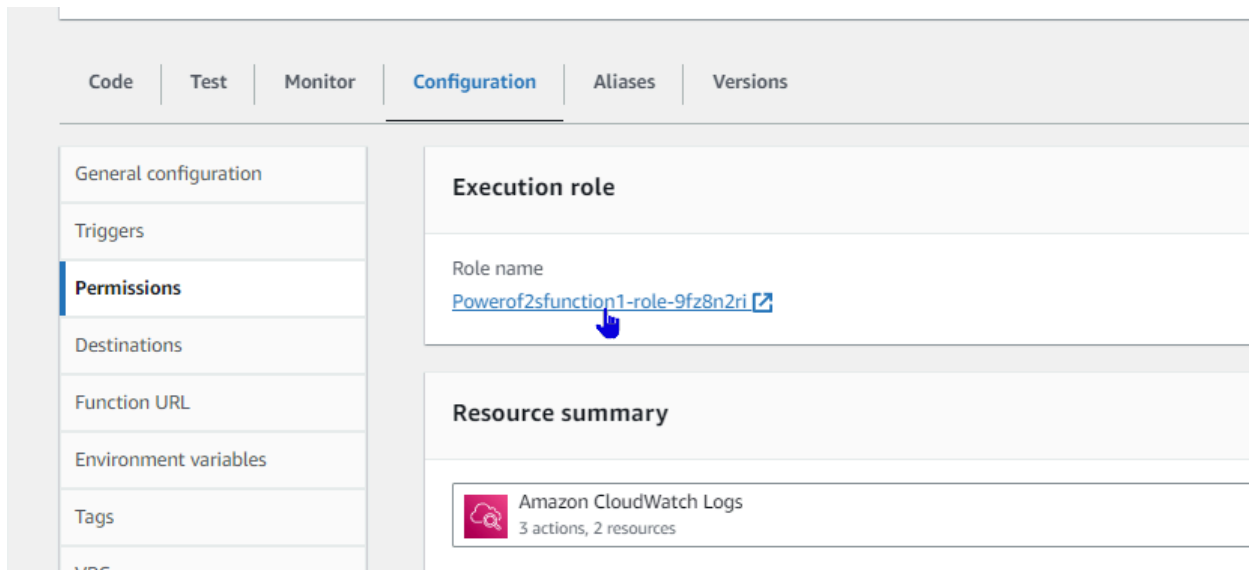
On the Lambda Dashboard, on the left navigation, click on Functions, click on the function name (ie. Powerof2sfunction1)



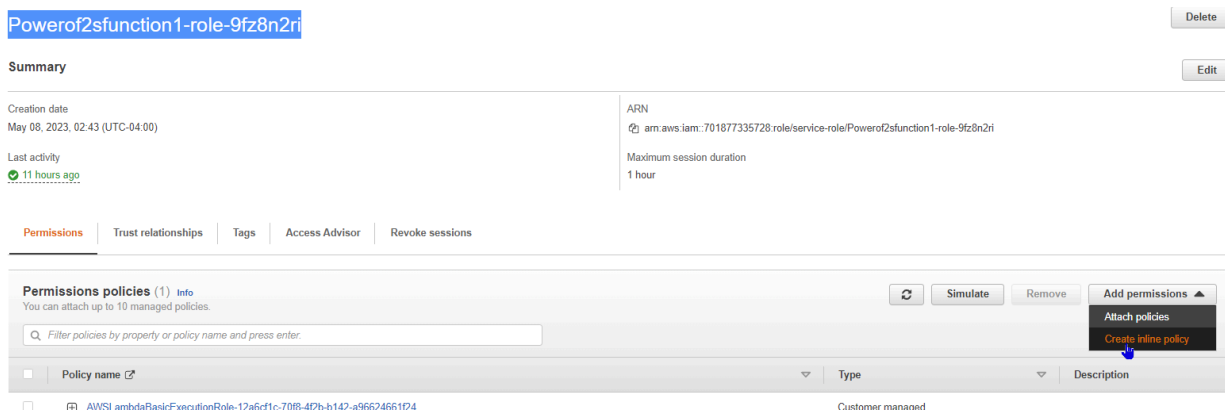
On the Powerof2sfunction page, click on Configuration tab.



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.



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



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

Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor

JSON

Import managed policy

Expand all | Collapse all

▼ Select a service

Clone | Remove

► Service

Choose a service

Actions

Choose a service before defining actions

Resources

Choose actions before applying resources

Request conditions

Choose actions before specifying conditions

[+ Add additional permissions](#)

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

Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor

JSON

ExecutionRolePolicy_JSON.txt

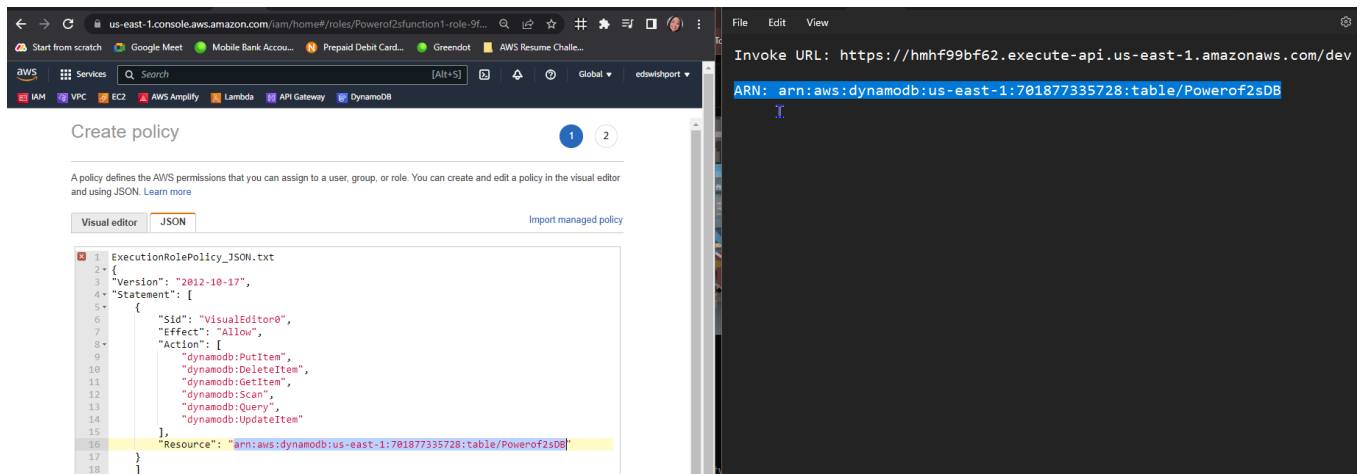
```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": [
8         "dynamodb:PutItem",
9         "dynamodb>DeleteItem",
10        "dynamodb:GetItem",
11        "dynamodb:Scan",
12        "dynamodb:Query",
13        "dynamodb:UpdateItem"
14      ],
15      "Resource": "YOUR-TABLE-ARN"
16    }
17  ]
18 }
19 }
```

ExecutionRolePolicy_JSON.txt

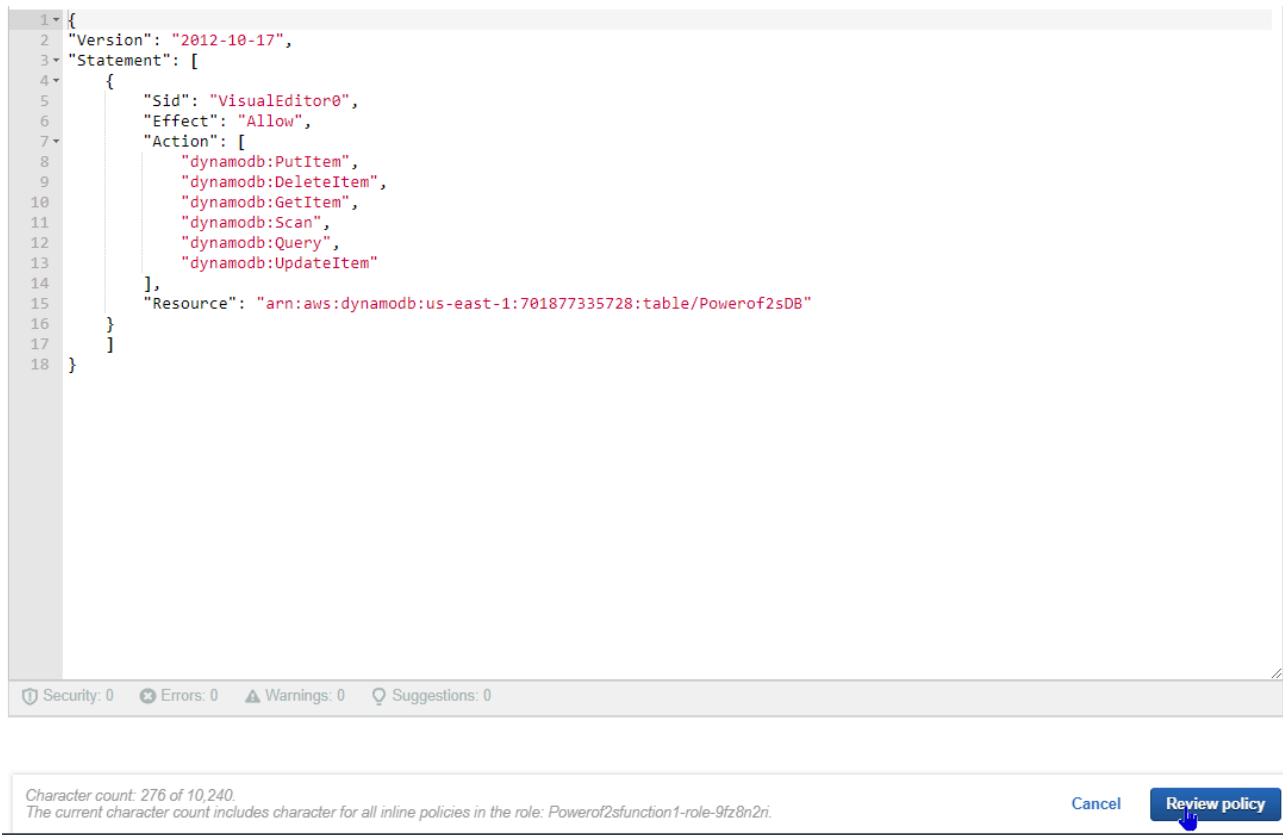
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb>DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:Scan",
        "dynamodb:Query",
        "dynamodb:UpdateItem"
      ],
      "Resource": "YOUR-TABLE-ARN"
    }
  ]
}
```

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

Review policy

Before you create this policy, provide the required information and review this policy.

Name* Powerof2sDynamoPolicy|

Maximum 128 characters. Use alphanumeric and '+-=, @_.' characters.

Summary

| Filter | | | |
|--|----------------------|---------------------------------------|-------------------|
| Service | Access level | Resource | Request condition |
| Allow (1 of 376 services) Show remaining 375 | | | |
| DynamoDB | Limited: Read, Write | TableName string like Powerof2sDB | None |

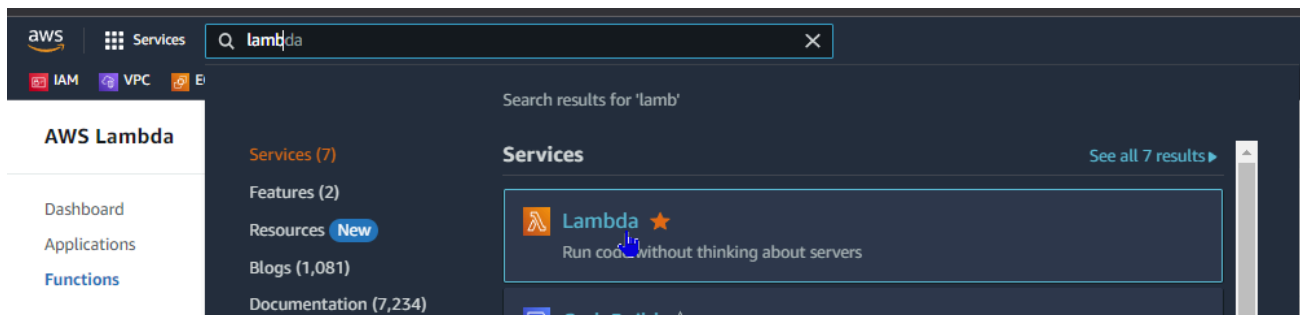
* Required

[Cancel](#)

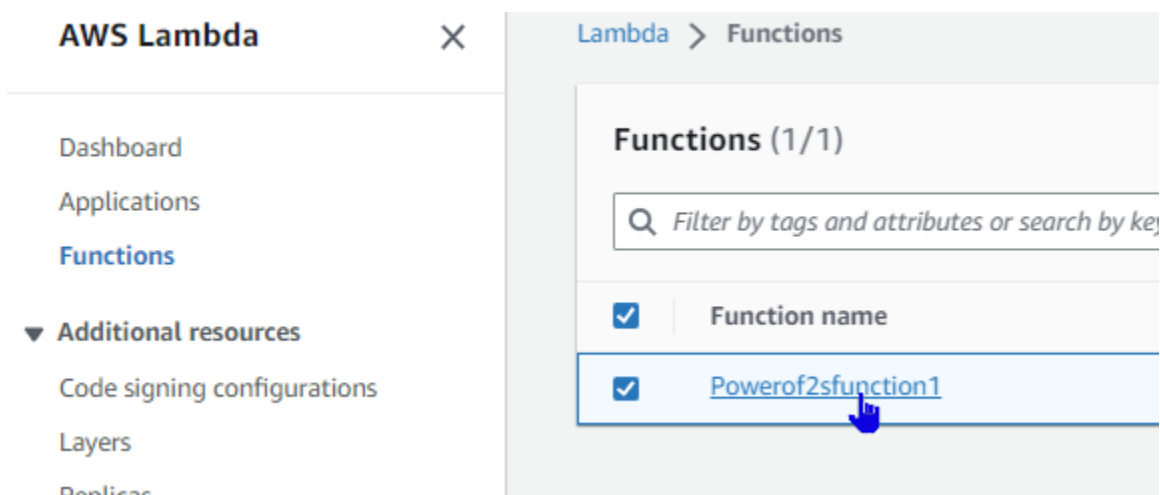
[Previous](#)

[Create policy](#)

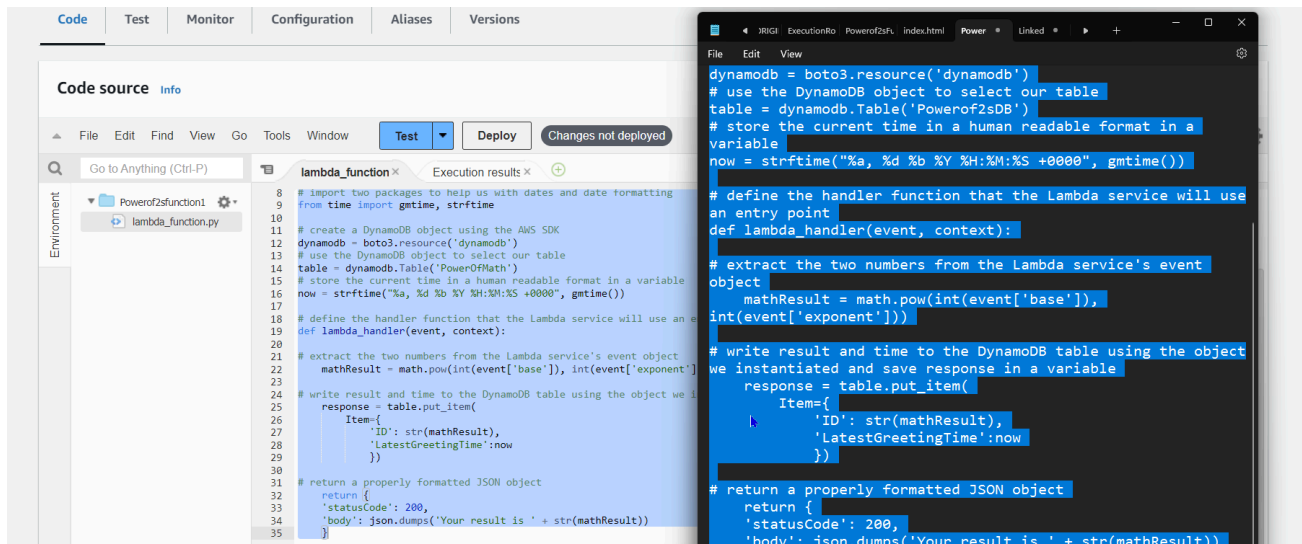
Go back to Lambda by searching or clicking your shortcut button.



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.



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 result into the DynamoDB table, it will also insert the current time.

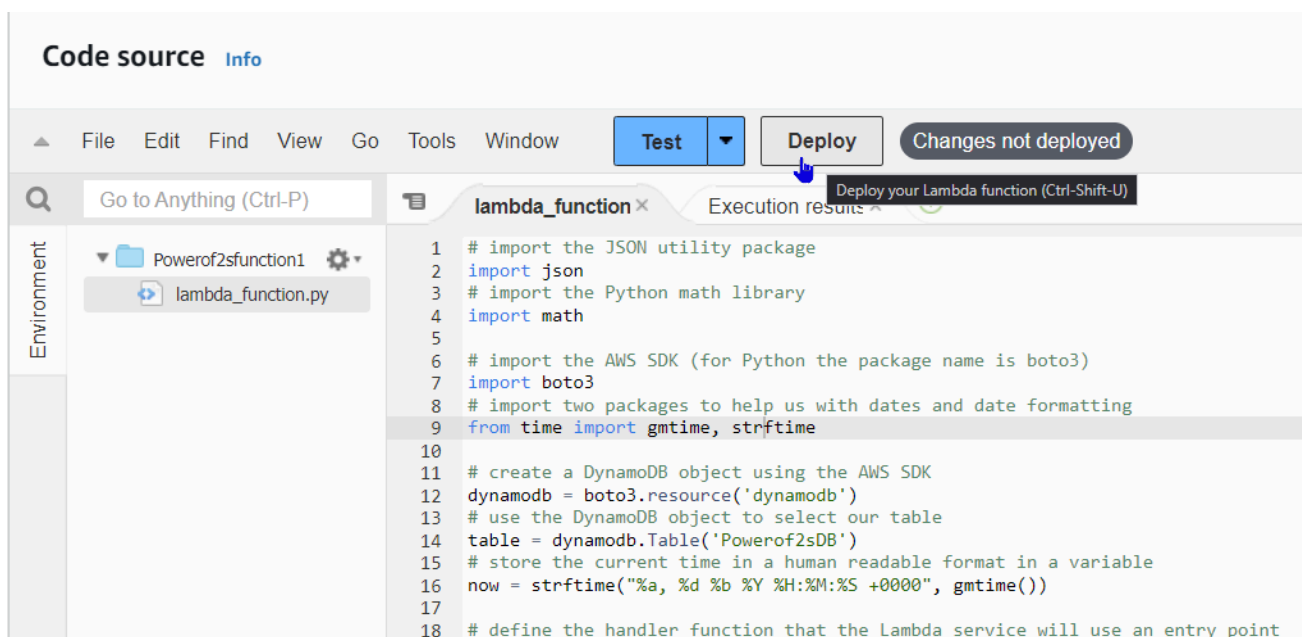
```

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.

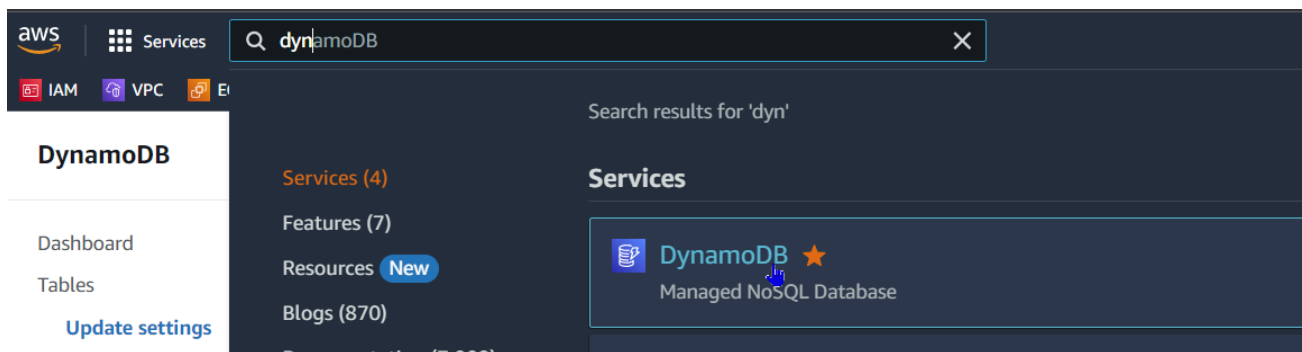
```
1 {
2   "base": 2,
3   "exponent": 4
4 }
```

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

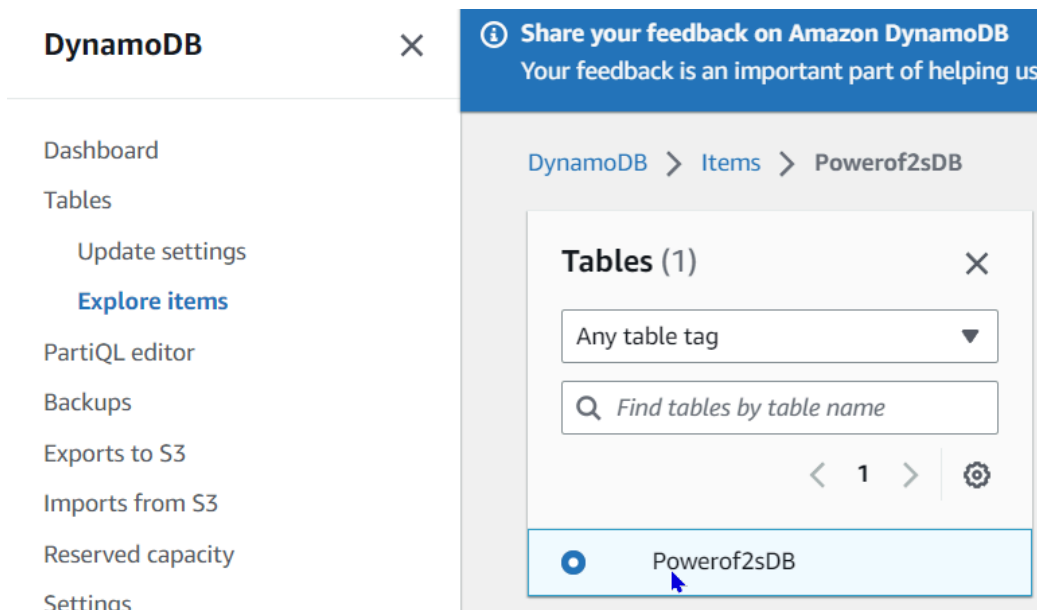
▼ Execution results

| | |
|------------------------|--|
| Test Event Name | Powerof2sTestEvent |
| Response | <pre>{ "statusCode": 200, "body": "\"Your result is 16.0\"" }</pre> |
| Function Logs | START RequestId: 0ecfa6d9-99f2-4b4e-9f87-a7057caea977 Version: \$LATEST END RequestId: 0ecfa6d9-99f2-4b4e-9f87-a7057caea977 REPORT RequestId: 0ecfa6d9-99f2-4b4e-9f87-a7057caea977 Duration: 259.42 ms Billed Duration: 260 ms |
| Request ID | 0ecfa6d9-99f2-4b4e-9f87-a7057caea977 |

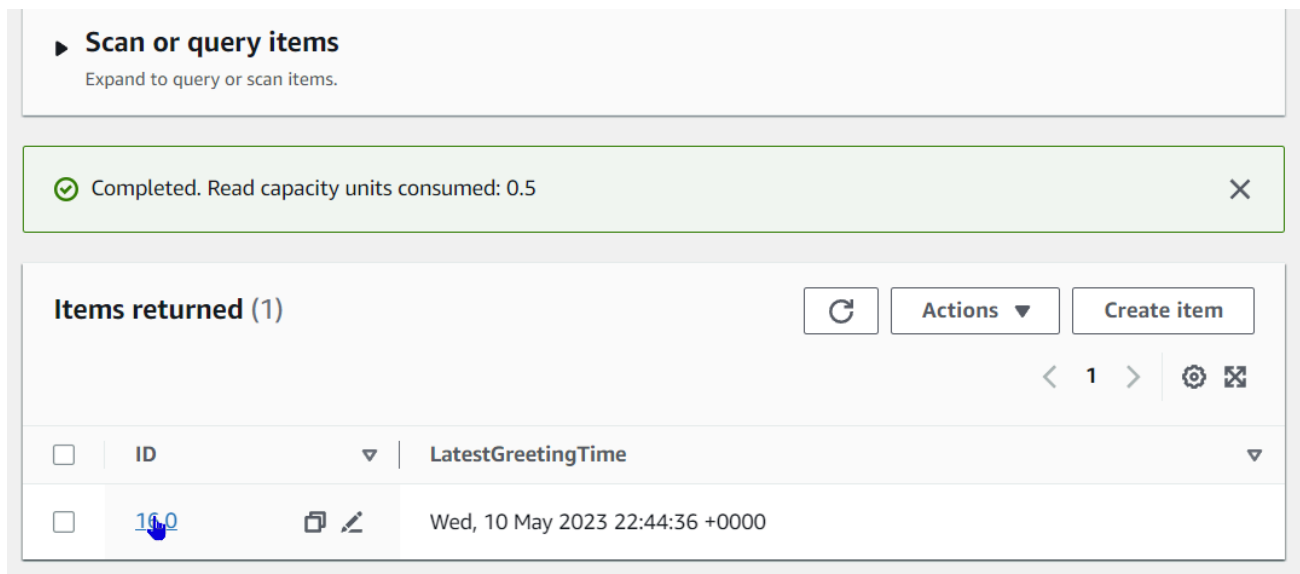
Go back to DynamoDB by clicking the shortcut button or search for it on the AWS Console Search then click on it.



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

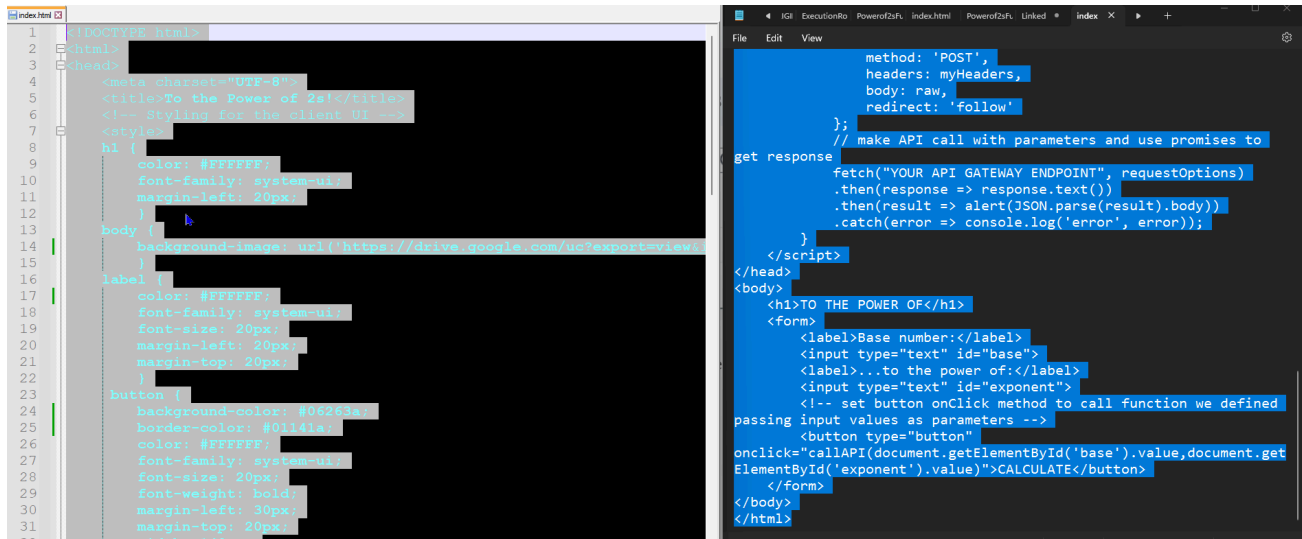


In the middle under Items returned, you will see the result of the query.



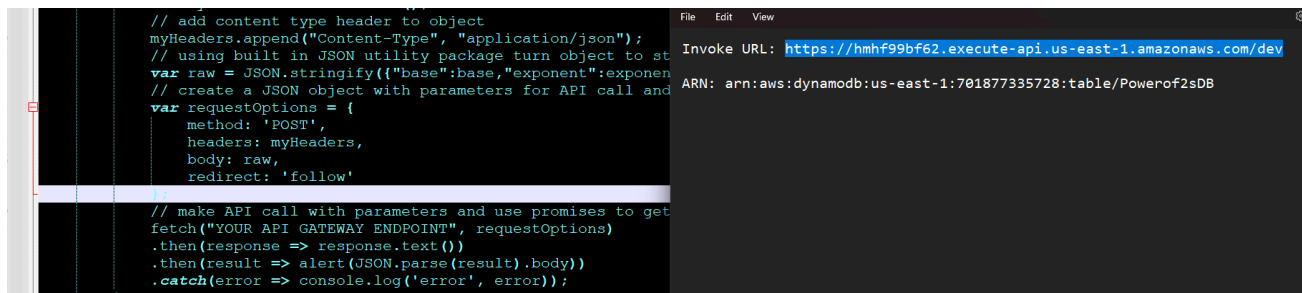
We need to update our index.html file that we created earlier. Open both index.html and the index-ORIGINAL.html copies in a notepad.

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



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”.

```
59 // make API call with parameters and use promises to get response
60 fetch("YOUR API GATEWAY ENDPOINT", requestOptions)
```

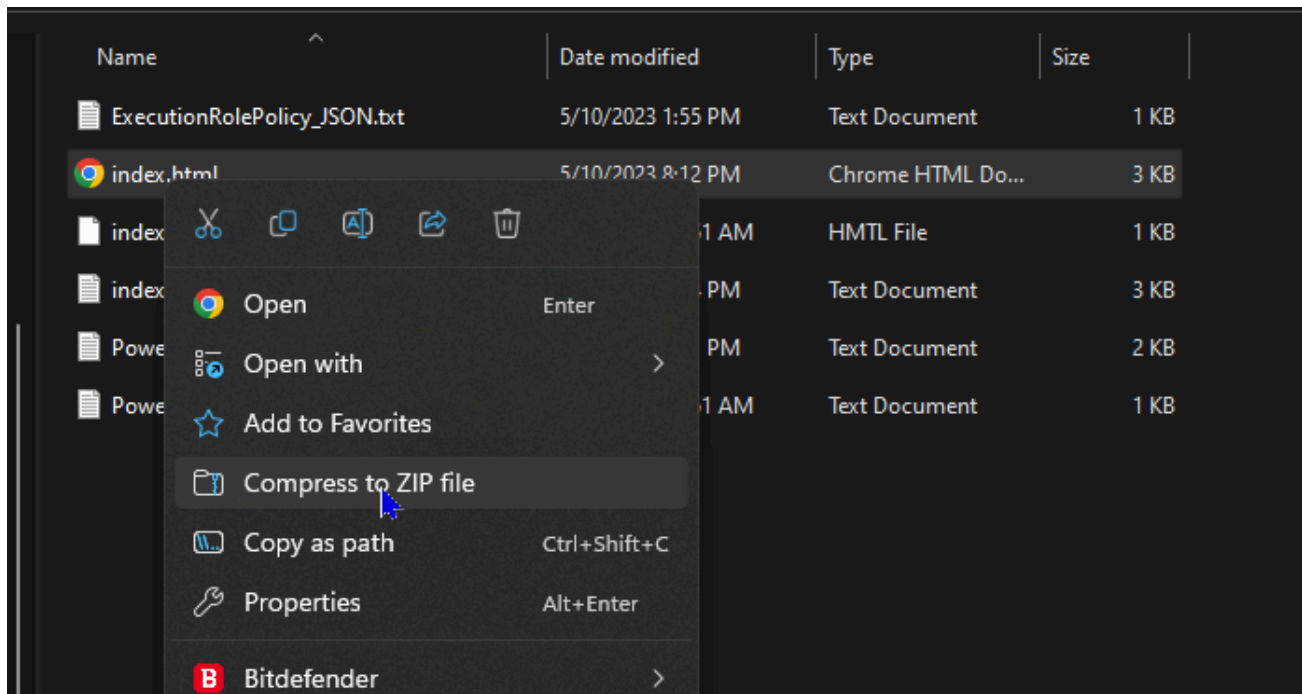


The index.html should look like this after you copied it.

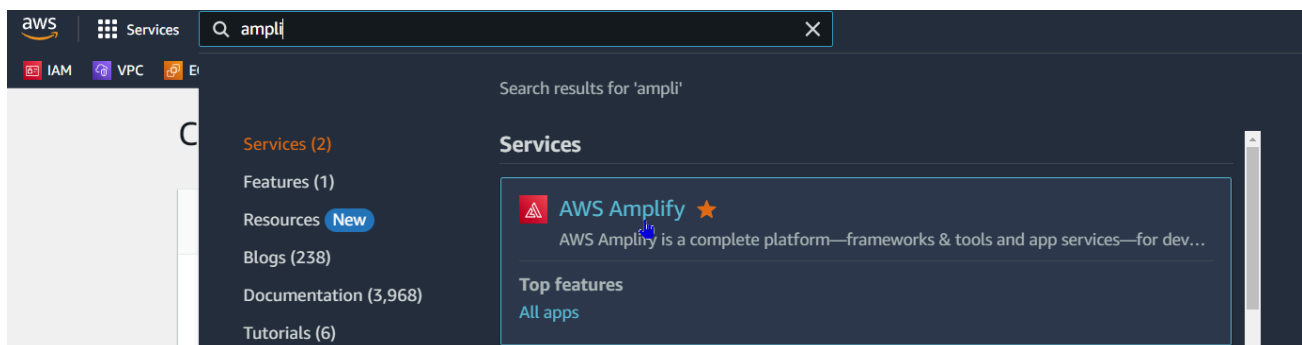
```
60 fetch("https://hmf99bf62.execute-api.us-east-1.amazonaws.com/dev", requestOptions)
```

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.



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

dev



Deployment successfully completed.



Domain
<https://dev.d1npuy3oun5lpr.amplifyapp.com>

Last deployment
5/8/2023, 2:11:13 AM




Drag and drop your project's build output directory or zip file here to update your app, or, [choose another method](#)

Choose files



Domain
<https://dev.d1npuy3oun5lpr.amplifyapp.com>

Last deployment
-

 index.zip



Click on the URL link to open another tab.

dev



Deployment successfully completed.



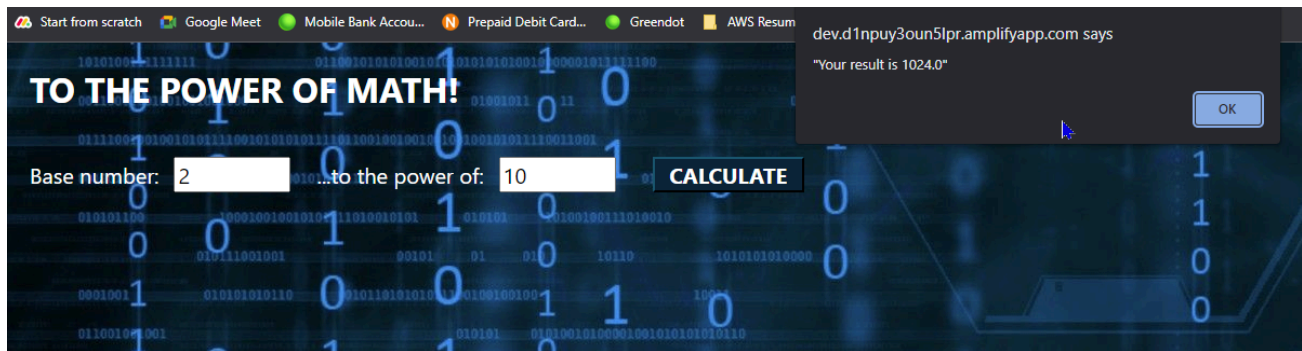
Domain
<https://dev.d1npuy3oun5lpr.amplifyapp.com>

Last deployment
5/10/2023, 8:49:13 PM



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/>

