# Pre-requisite steps for creating your serverless application

## Create your AWS free trial account

1. Visit <https://aws.amazon.com/free/start-your-free-trial/>
2. Click “Create your account” button
3. Enter your email address, a password and an account name
4. For account usage type, select Personal, unless you plan to reuse this account for Professional reasons
5. Enter your phone number, an address, and agree to the terms and conditions.
6. Enter CC information
7. You will need to confirm your account using a mobile number. Follow the instructions on the screen to get a confirmation number. When Amazon calls your mobile phone, enter the confirmation number in your keypad.

Your account should complete set up within five minutes.

## Open GitHub repository containing code for this session

1. Verify that you have access to the following GitHub repository: <https://github.com/JeneeB/GHC2018SchedulerApplication>

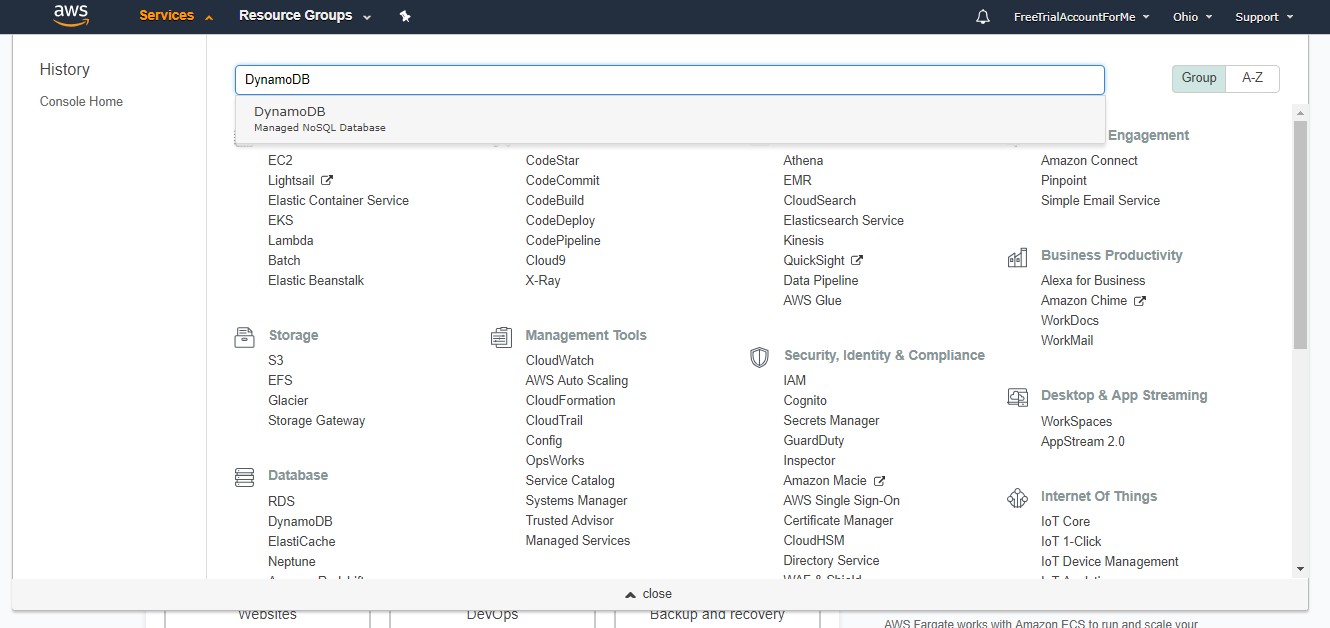
# Building your Serverless application

For the purposes of this workshop, you will continue using the region that you were defaulted to. In many cases, that region will be Ohio. However, you can use any region, as long as you use the same region for all steps.

## DynamoDB

You will now create the DynamoDB table for your application. This table will store all the GHC session information. Below are the steps:

1. On the Navigation bar at the top of the page, click on “Services”. A dropdown of all the services that AWS provides in the region will appear.
2. Find the “DynamoDB” link in the Database section in the left column and select it. You can also type into the Search bar “DynamoDB” and select it from the search suggestions

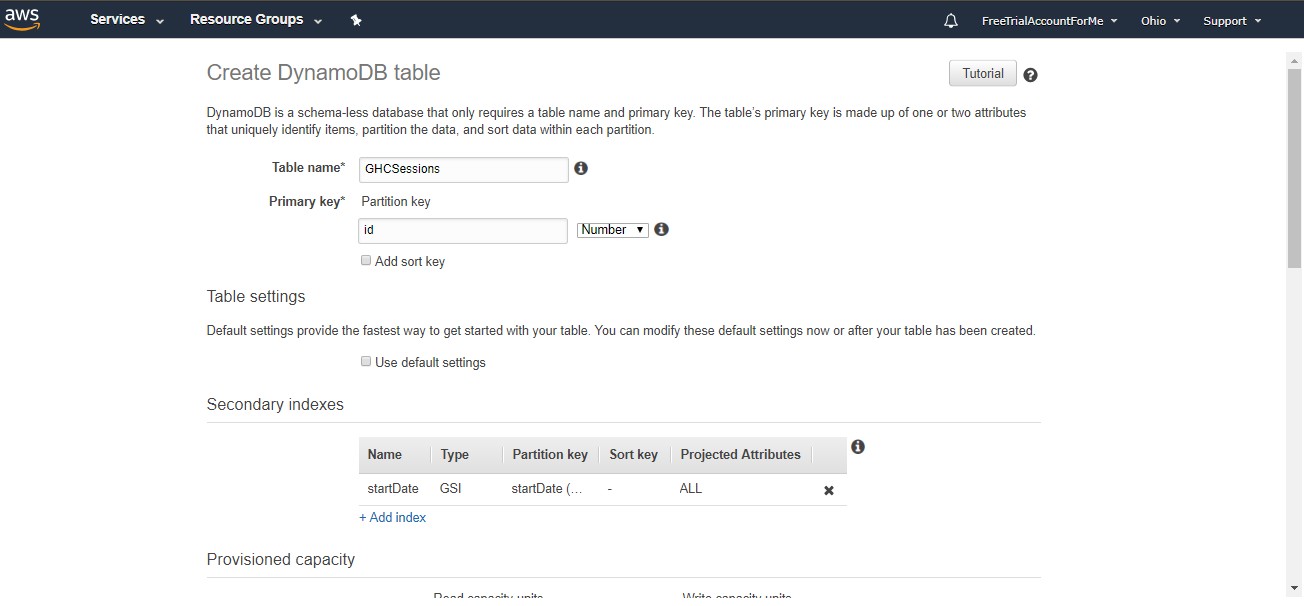


1. On the DynamoDB homepage, go ahead and read about DynamoDB and click “Create Table” when you are ready.
2. Create a table with the below details:

**Table name**: GHCSessions

**Primary key:** id

Make sure your Primary Key is **Number**, not String or Binary.



\*Note: If you created the table with a primary key that is not Number, the function to populate the table will not work. You will need to delete and re-create the table. Make sure your Primary Key is **Number.**

1. Leave “Add sort key” as unchecked.
2. In the Secondary Indexes section, unselect the “Use default settings” box. You will be creating a Secondary Index on this table.

Secondary Indexes allows efficient access to data with attributes other than the primary key.

1. Enter the below details for the secondary index:

Primary key\*: startDate

Make sure this Primary Key is **String**

Your index name should be auto-filled: startDate-index

1. Leave “Add sort key” as unchecked and leave “Projected attributes” as All
2. Click “Add Index” to create your Secondary Index
3. Leave everything else on the page as-is and scroll to the bottom. Click “Create” and wait for DynamoDB to create your table. This may take a few minutes.

In the meantime, you will create our AWS Lambda functions.

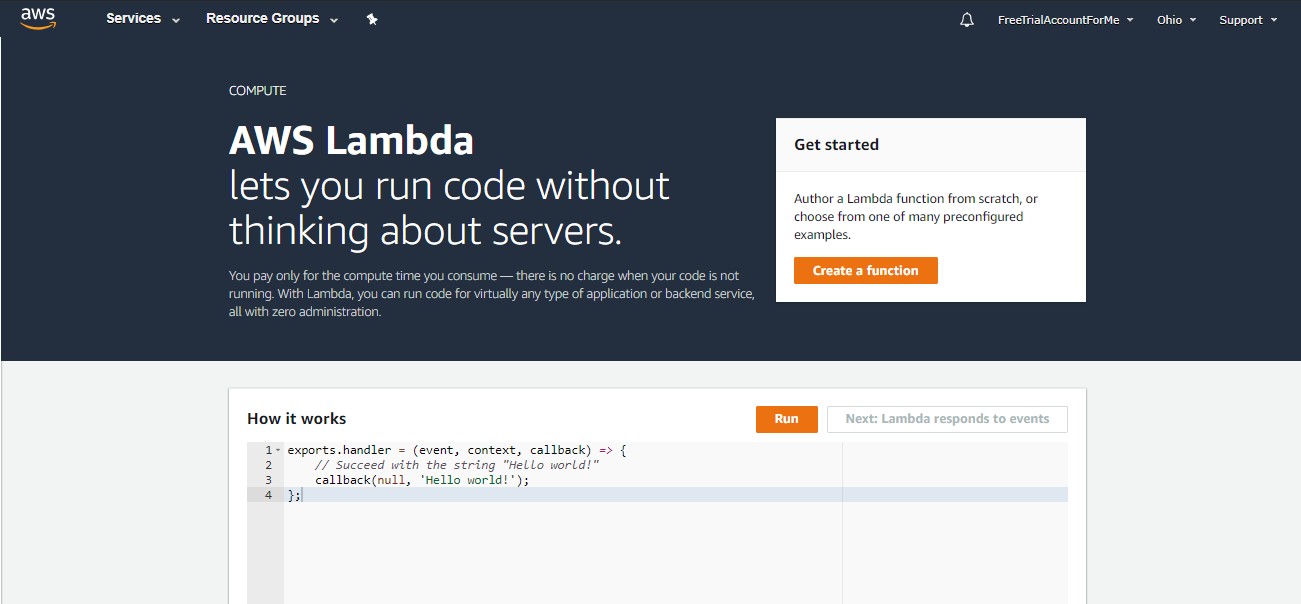
## AWS Lambda

For this setup, make sure you have pop-up blocker turned off. You will create three AWS Lambda functions through the console.

### AddSessions

The first Lambda you create will contain the code to populate the DynamoDB table that you just created. This function and the all of the functions in this exercise will use a **Node.js 8.10** runtime environment.

1. On the Navigation bar at the top of the page, click on “Services”. A dropdown of all the services that AWS provides in the region will appear.
2. Find the “Lambda” link under the “Compute” section in the left column and select it. You can also type into the Search bar “Lambda” and select it from the search suggestions

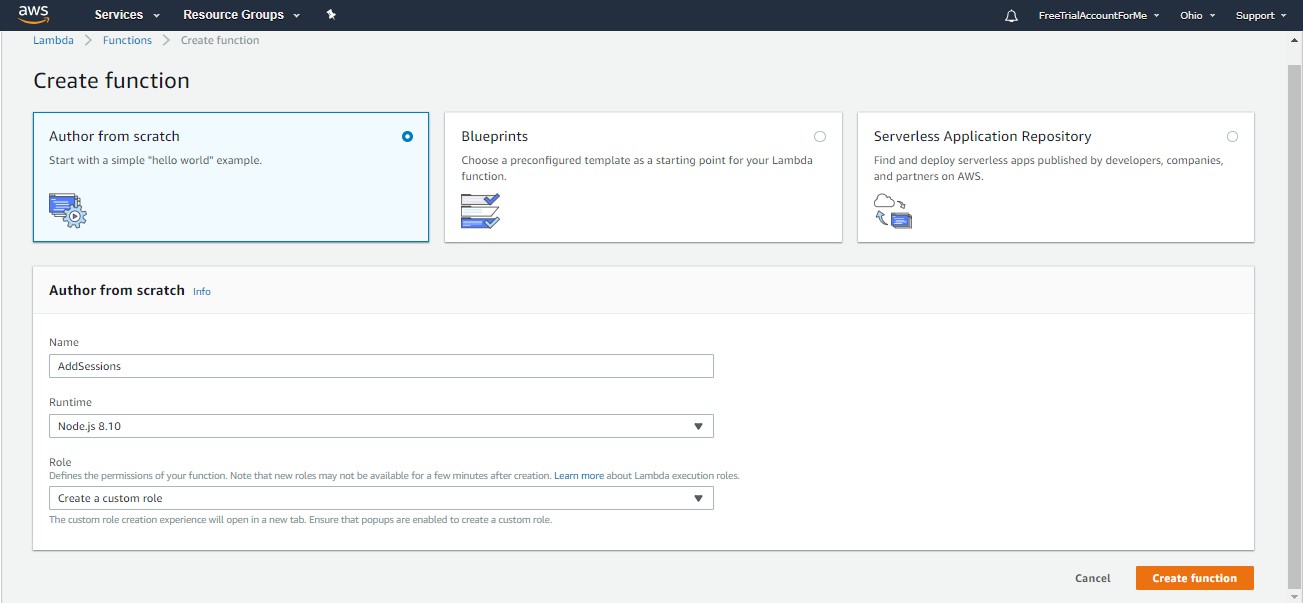


1. Select the “Create a function” button
2. Select the first card titled “Author from scratch”
3. Fill out the form for the Lambda with the following details

**Name:** AddSessions

**Runtime:** Node.js8.10*(not 6.10)*

**Role:** Create a custom role



Your IAM (Identity Access Management) role is what defines the permissions for your lambda, such as which AWS services your lambda will be able to access.

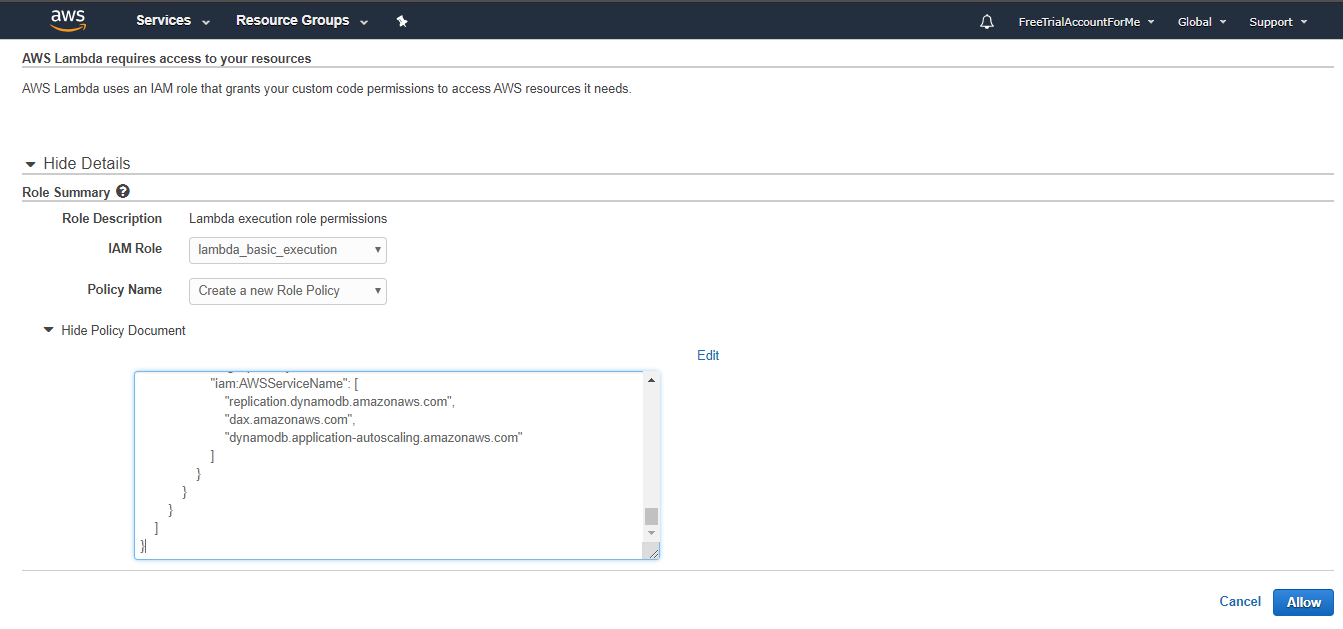
1. In the pop-up window that opens, you will create a new role for this function. You will use this role for this Lambda and the other you create. In the window that opens, enter the following details to create your role.

**Role Description:** Lambda execution role permissions

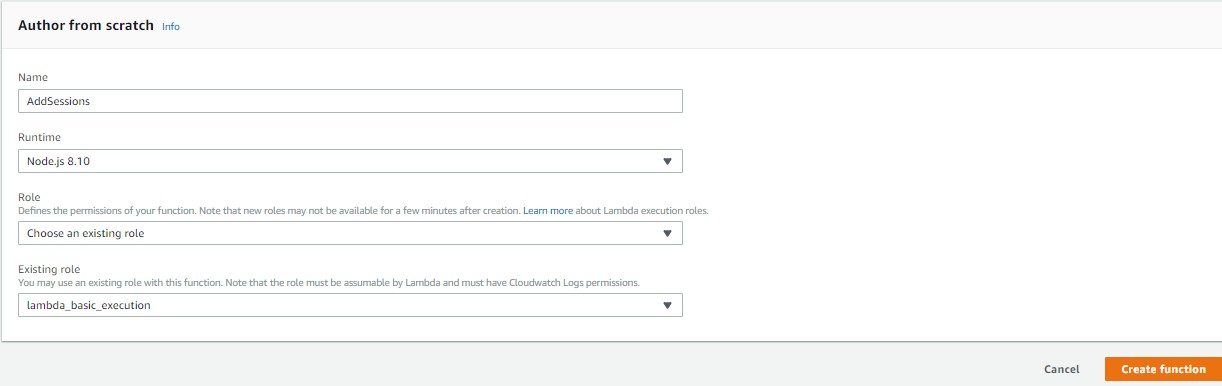
**IAM Role:** Create a new IAM role

**Role Name:** lambda\_basic\_execution

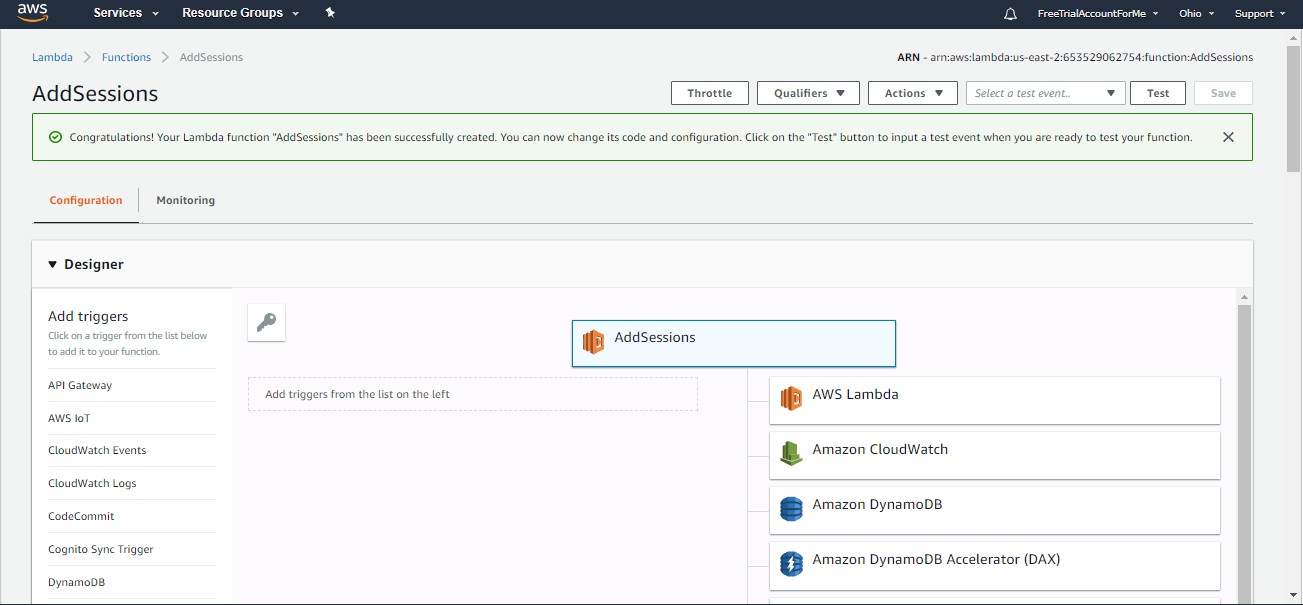
1. View Policy Document and click Edit. The Policy Document textbox should become modifiable.
2. On the GitHub repository, open the iam\_role.txt file located at <https://github.com/JeneeB/GHC2018SchedulerApplication/blob/master/lambda/iam/dynamodb_full_access.txt>
3. Copy the entire file.
4. Paste the policy into the Policy Document textbox on the IAM page



1. Click the “Allow” Button at the bottom and the window will close.
2. Go ahead and continue the Lambda setup by clicking “Create Function”.



You should now see the Lambda’s detail page.



1. Scroll down to the Function code section. You will be editing the index.js file that is already open in the editor.
2. Open the GitHub repository and navigate to <https://github.com/JeneeB/GHC2018SchedulerApplication/blob/master/lambda/functions/addSessions/index.js>
3. Copy the code from that file and paste it into the AWS Lambda function code editor.
4. Go to File, then click Save to save your changes.

You will also create a new file in the AddSessions directory.

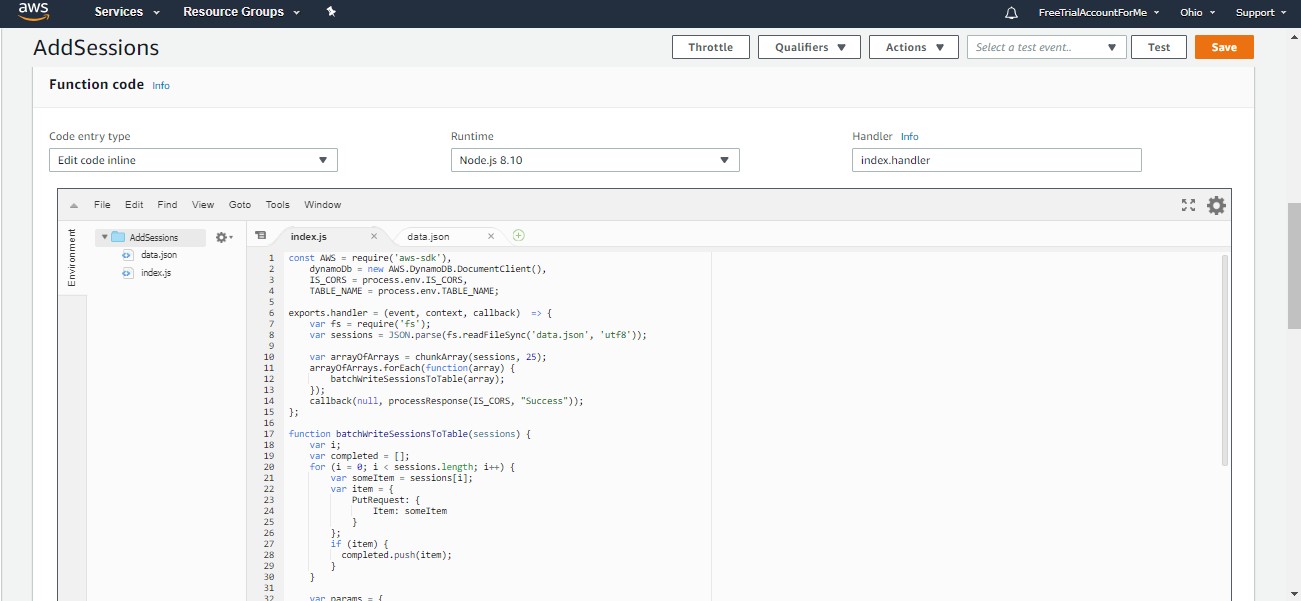
1. Go to File, then New File.
2. Open the GitHub repository and navigate to <https://github.com/JeneeB/GHC2018SchedulerApplication/blob/master/lambda/functions/addSessions/data.json>
3. Copy the data from that file and paste it into the AWS Lambda function code editor for the new file that you have open.
4. Go to File, then click Save to save your changes. Use the below details to save the file.

**Filename**: data.json

Make sure AddSessions is selected

**Folder**: /

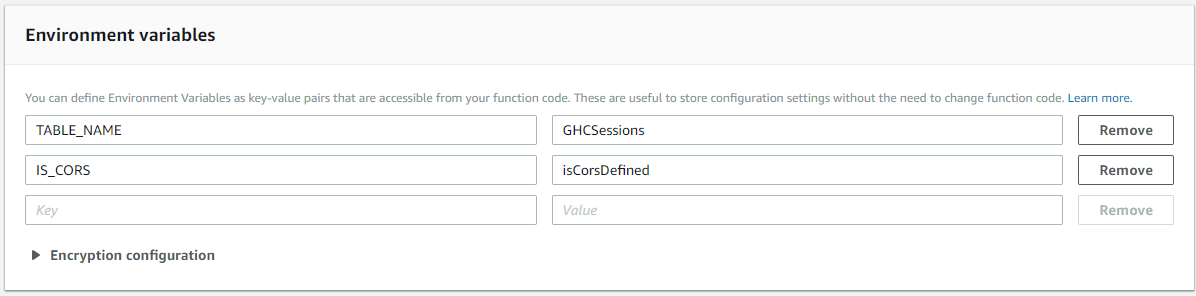
1. Click Save. Your Function code section should look similar to the below



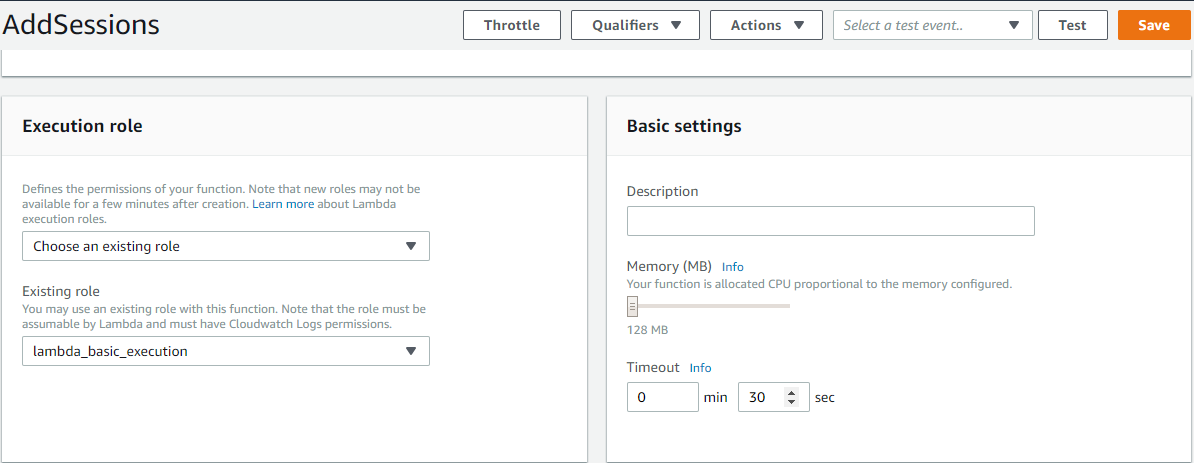
1. Scroll down to the “Environment variables” section on the Lambda detail page. You want to add two Environment variables:

**Key:** TABLE\_NAME **Value:** GHCSessions

**Key:** IS\_CORS **Value:** isCorsDefined



1. In “Basic settings” increase the timeout from 3 seconds to 30 seconds. This will give your function more time to run and populate the DynamoDB table.



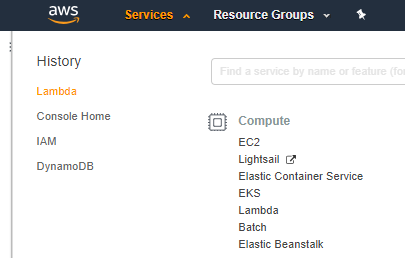
1. Keep everything else as-is, and then click the Orange “Save” button at the top of the screen.

And that’s all! You have created your first Lambda function. And don’t worry, the setup for the next two lambdas are simpler. You will set up the trigger for this function when you set up the Amazon APIs later.

Let’s create the next lambda function for our application.

### GetAllSessions

The next Lambda you create will contain the code to fetch all the session from the GHCSessions DynamoDB table.

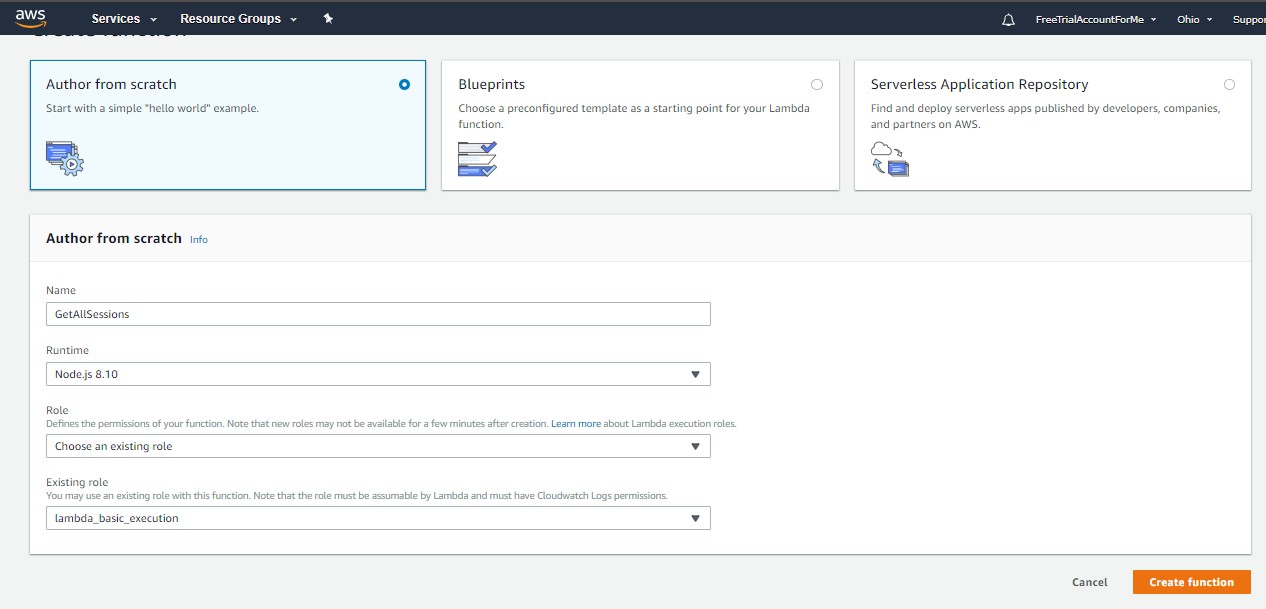
1. At the top of the page, go to “Services” and select Lambda. Lambda should already be near the top of your history in the left column. 
2. Click the Orange Create Function button
3. Select the “Author from scratch” card
4. Fill out the form for the Lambda with the following details

**Name**: GetAllSessions

**Runtime**: Node.js 8.10*(not 6.10)*

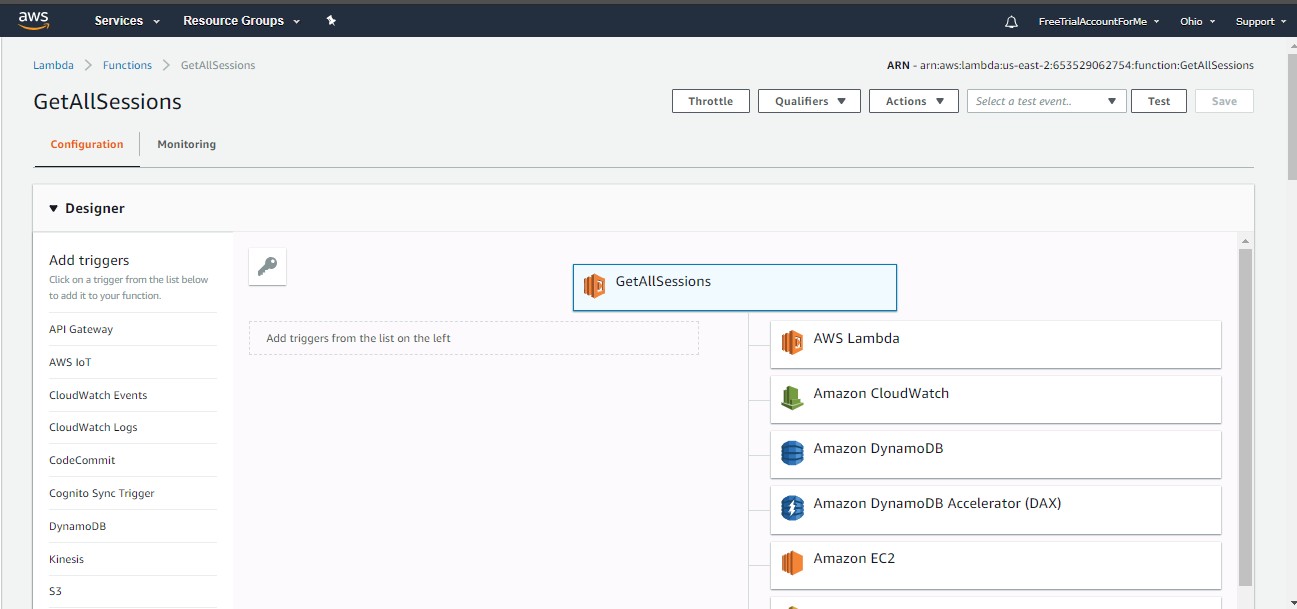
**Role**: Choose and existing role

**Existing role**: lambda\_basic\_execution

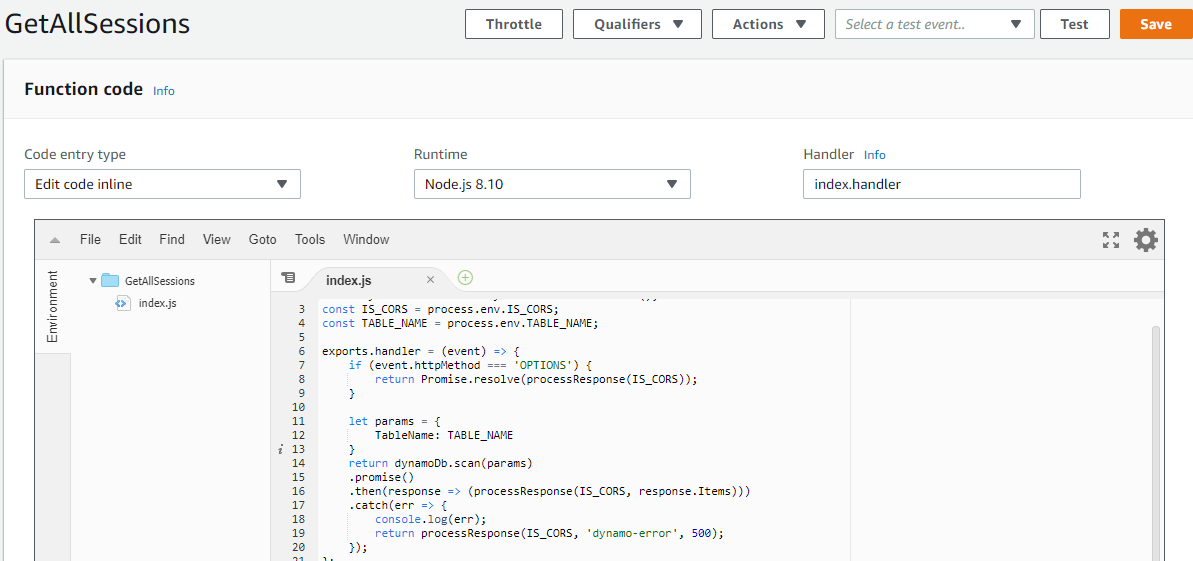


1. Continue the Lambda setup by clicking “Create Function”.

You should now see the Lambda’s detail page.



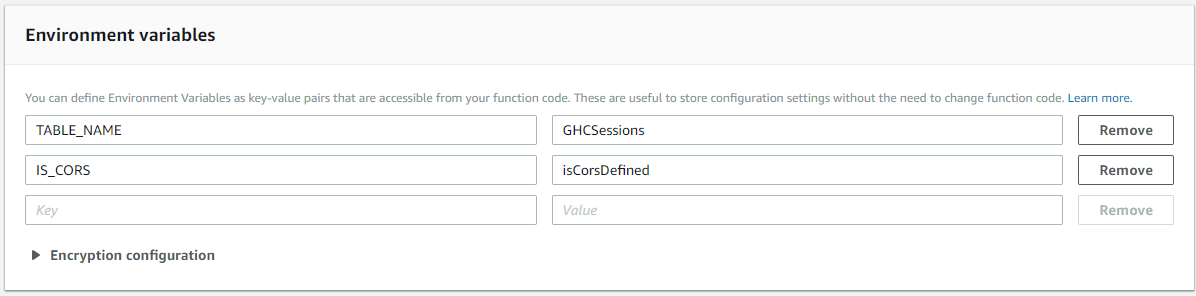
1. Scroll down to the Function code section. You will be editing the index.js file that is already open.
2. Open the GitHub repository and navigate to <https://github.com/JeneeB/GHC2018SchedulerApplication/blob/master/lambda/functions/getAllSessions/index.js>
3. Copy the code from that file and paste it into the AWS Lambda function code editor.
4. Go to File, then click Save to save your changes.



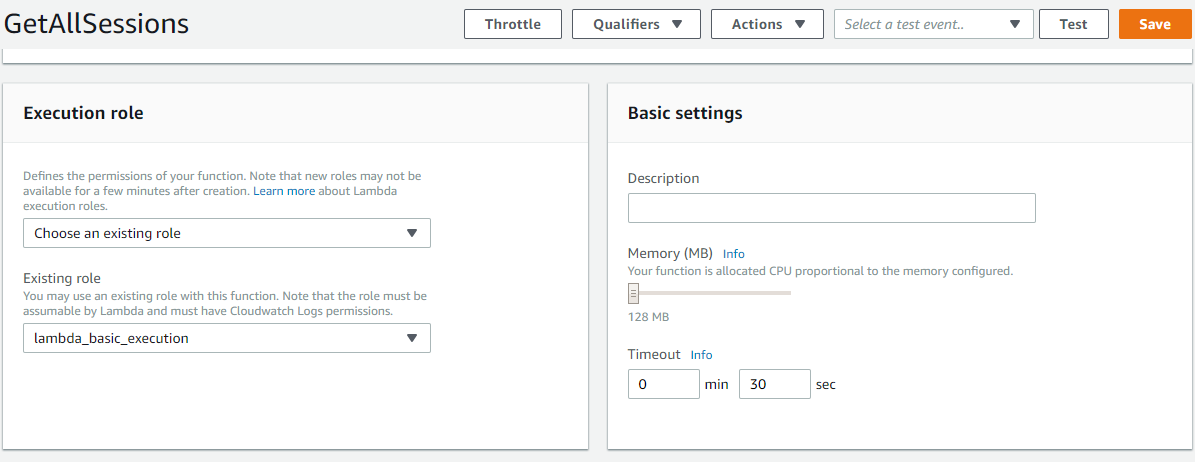
1. Scroll down to the “Environment variables” section on the Lambda detail page. You want to add two Environment variables.

**Key**: TABLE\_NAME **Value**: GHCSessions

**Key**: IS\_CORS **Value**: isCorsDefined



1. In “Basic settings” increase the timeout from 3 seconds to 30 seconds. This will give your function more time to run and get all the sessions from the DynamoDB table.
2. Keep everything else as-is, and then click the Orange “Save” button at the top of the screen.



You have now created your second Lambda function. You will set up the trigger for this function when you set up our Amazon APIs later on.

The third lambda function needed for our application is very similar to this second one.

### UpdateSession

This last Lambda will be used to update existing sessions in the DynamoDB GHCSessions table.

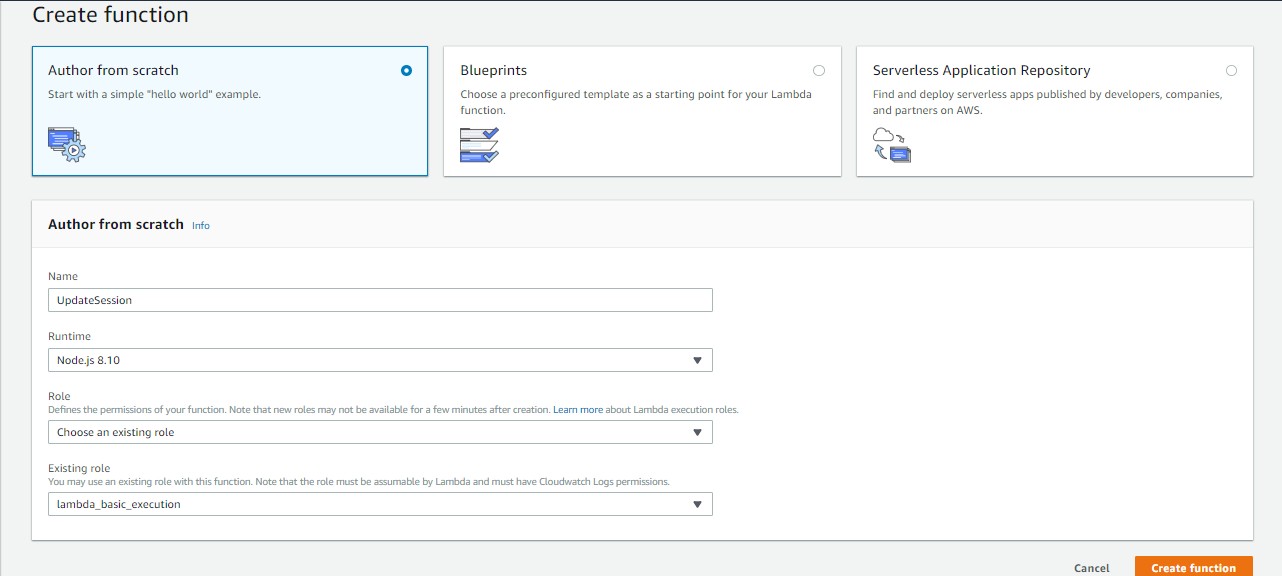
1. At the top of the page, go to “Services” and select Lambda. Lambda should already be near the top of your history in the left column.
2. Click the Orange Create Function button.
3. Select the “Author from scratch” card
4. Fill out the form for the Lambda with the following details

**Name**: UpdateSession

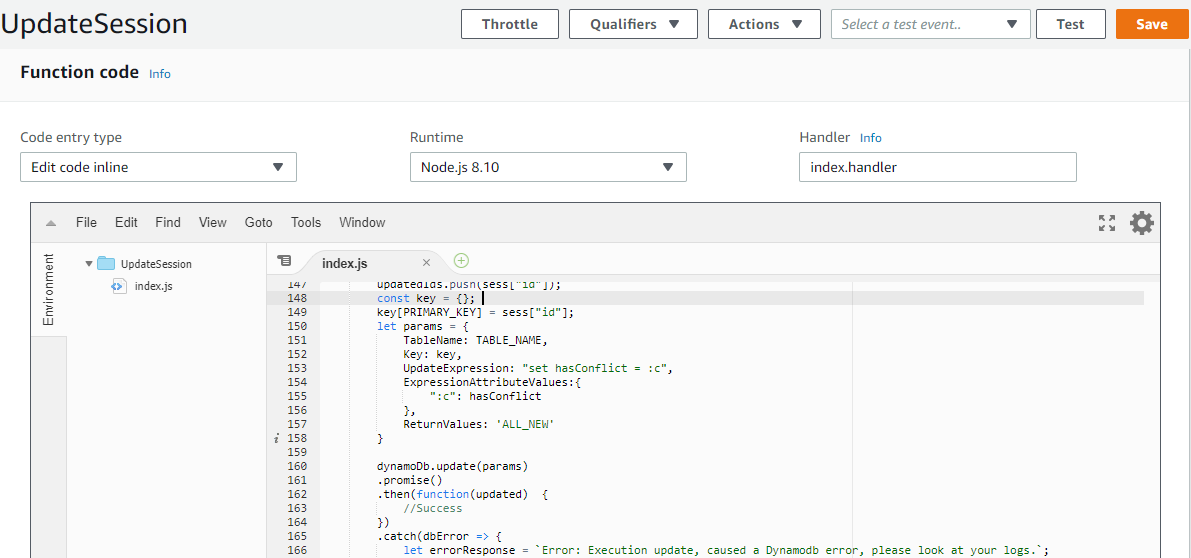
**Runtime**: **Node.js 8.10** *(not 6.10)*

**Role**: Choose and existing role

**Existing role**: lambda\_basic\_execution



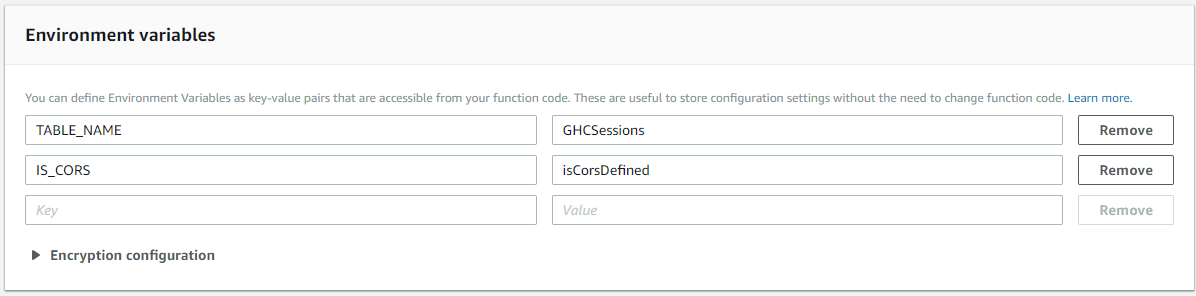
1. Continue the Lambda setup by clicking “Create Function”.
2. Scroll down to the Function code section. You will be editing the index.js file that is already open.
3. Open the GitHub repository and navigate to <https://github.com/JeneeB/GHC2018SchedulerApplication/blob/master/lambda/functions/updateSession/index.js>
4. Copy the code from that file and paste it into the AWS Lambda function code editor.
5. Go to File, then click Save to save your changes.



1. Setup the GetOneSession function with the same “Environment Variables” and the same “Basic Settings” as GetAllSessions:

**Key**: TABLE\_NAME **Value**: GHCSessions

**Key**: IS\_CORS **Value**: isCorsDefined



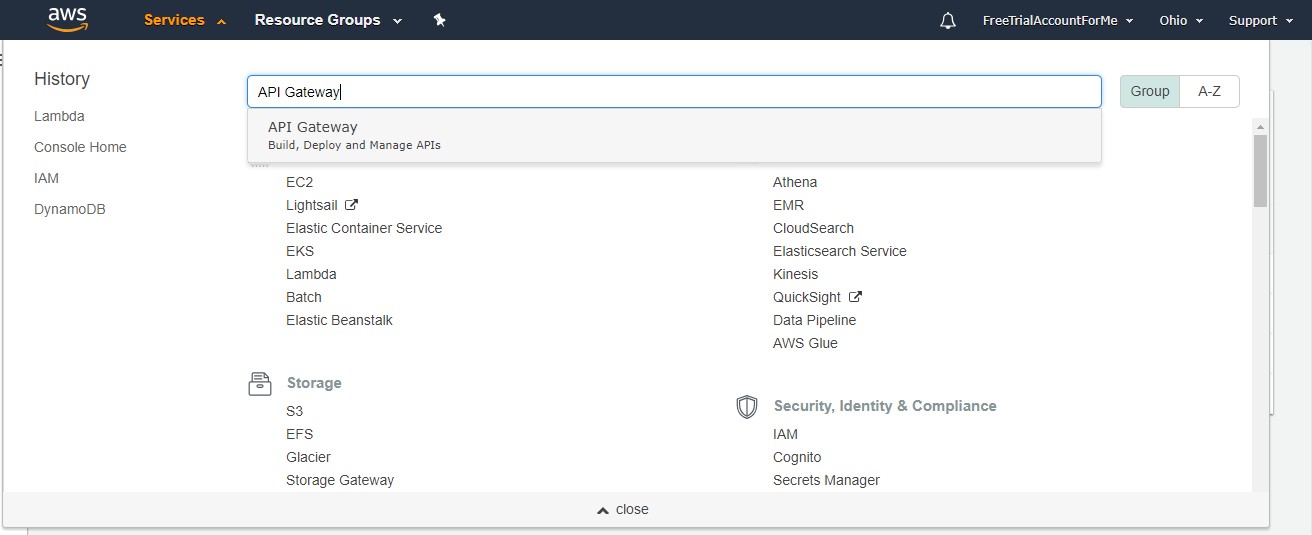
1. In “Basic settings” increase the timeout from 3 seconds to 30 seconds.
2. Then, click Save.

## Amazon API Gateway

Now that you have your Lambdas, you will now create the APIs that will trigger the functions. You will create three APIs in Amazon API Gateway for this exercise.

## add-dynamodb

1. On the Navigation bar at the top of the page, click on “Services”. A dropdown of all the services that AWS provides in the region will appear.
2. Find the “API Gateway” link under the “Networking and Content Delivery” section in the left column and select it. You can also type into the Search bar “API Gateway” and select it from the search suggestions

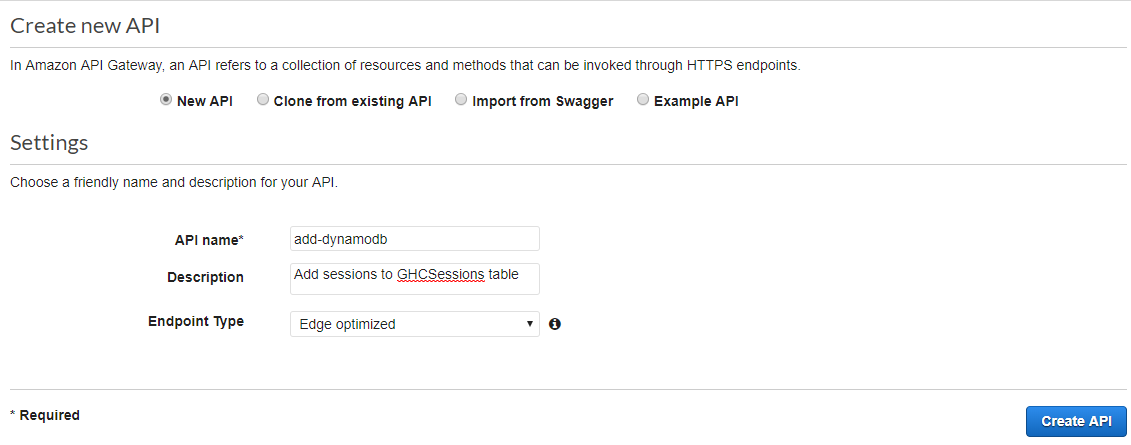


1. Click “Get Started” to start building your first API.
2. Select the “New API” radio button
3. Enter the below details for your API

**API name\*:** add-dynamodb

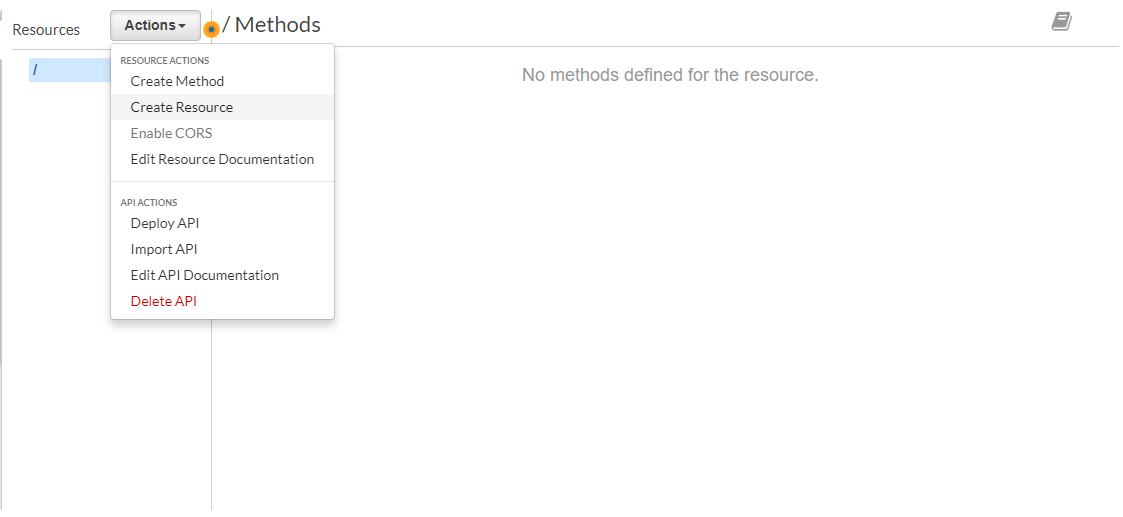
**Description**: Add sessions to GHCSessions table

**Endpoint Type**: Edge optimized



You will be taken to the API dashboard where you can define your methods and resources for your API.

1. Click the “Actions” Button and the dropdown, select “Create Resource”



1. Supply the following details:

Leave **Configure as proxy resource** unchecked.

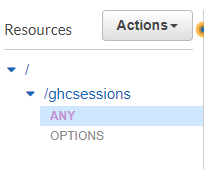
**Resource Name\*:** GHCSessions

**Resource Path\*:** ghcsessions

Select the **Enable API Gateway CORS** checkbox.



1. Click “Create Resource” button.
2. You should now see the resource that you just created appear in the Resources tree. Click on its name “/ghcsessions”.
3. Click the “Actions” button again, and select “Create Method”
4. You will see that a dropdown menu element gets added to the tree below “OPTIONS”. In that dropdown, select “ANY”, and then click the Check mark next to it.



1. Here is where we set up the API integration with Lambda. Fill out the below details:

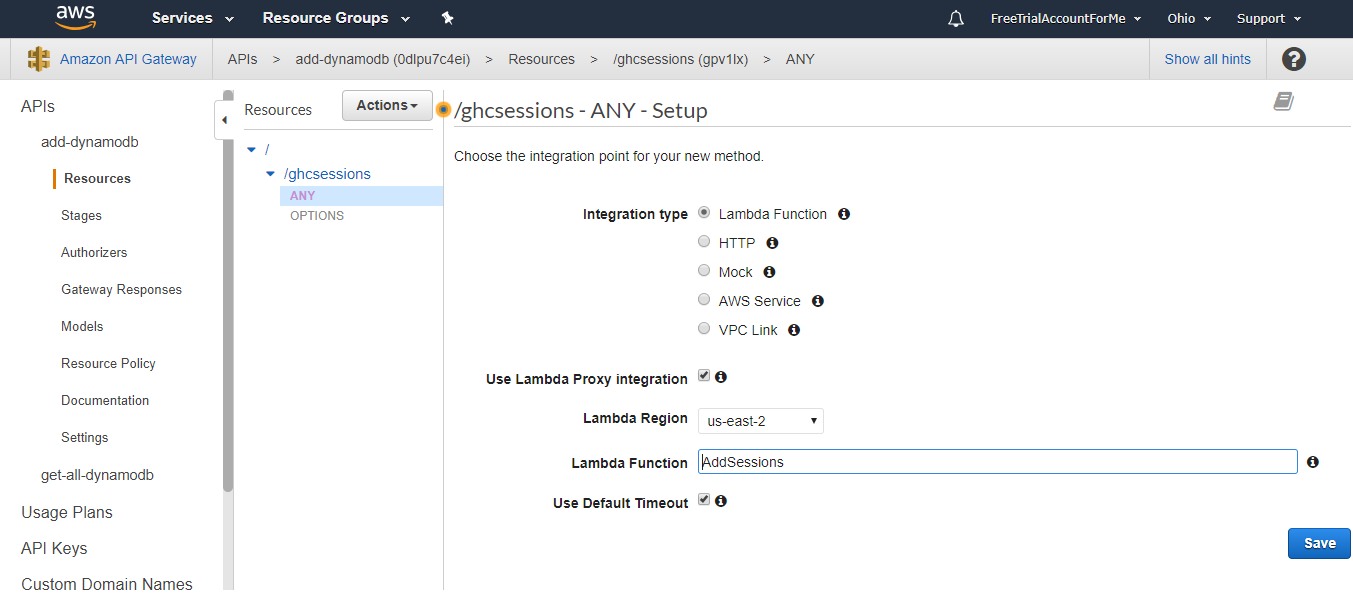
Select the **Lambda Function** radio button.

Select the **Use Lambda Proxy integration** checkbox

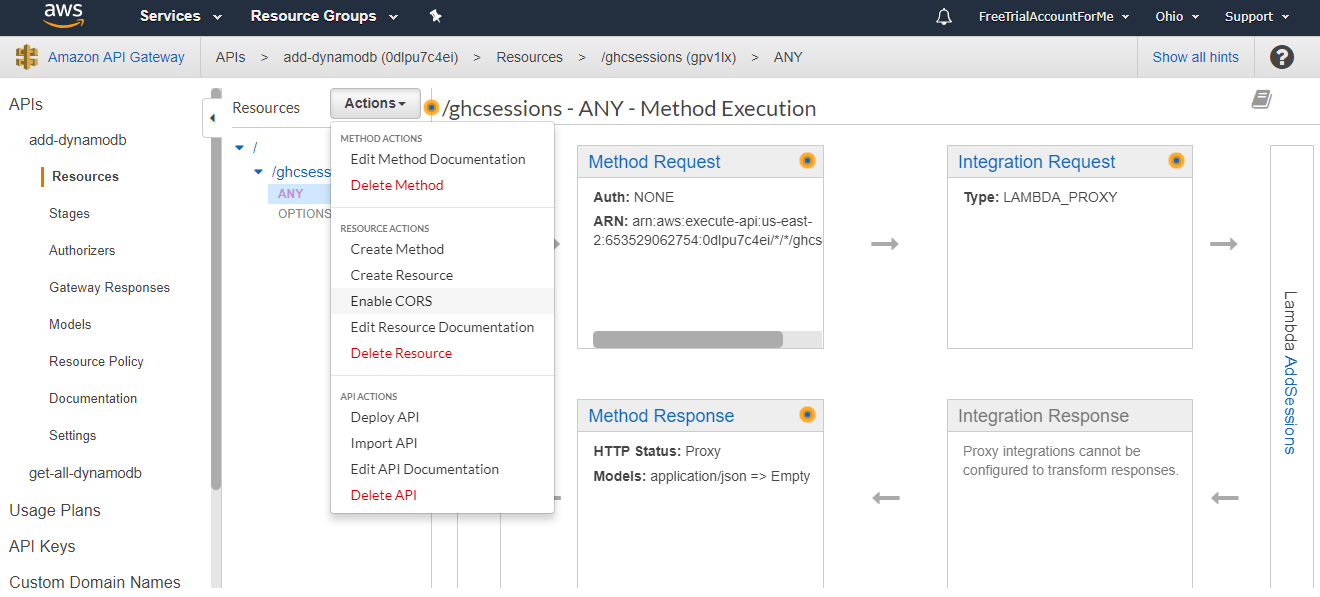
Select the same region that you have been using for this workshop (you should not need to change the option that is pre-selected)

In **Lambda Function,** enter the name of the first function that we created earlier that gets all the sessions in the table – **AddSessions**

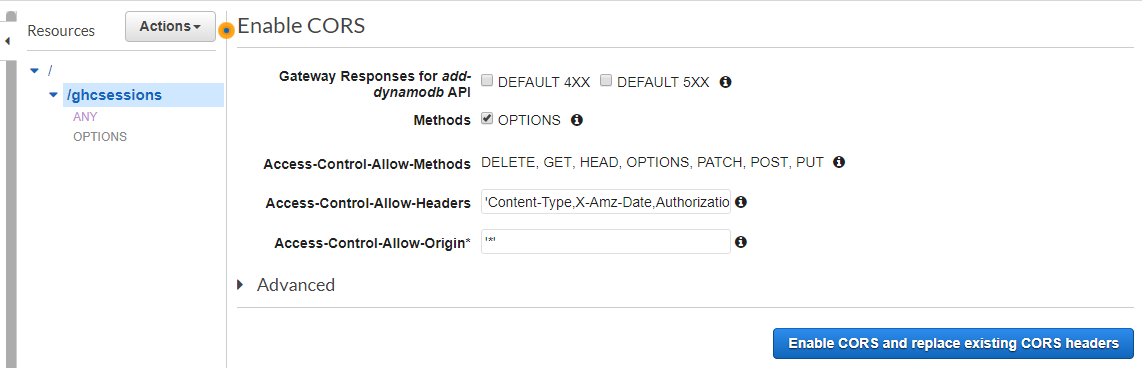
Leave “**Use Default Timeout**” as selected



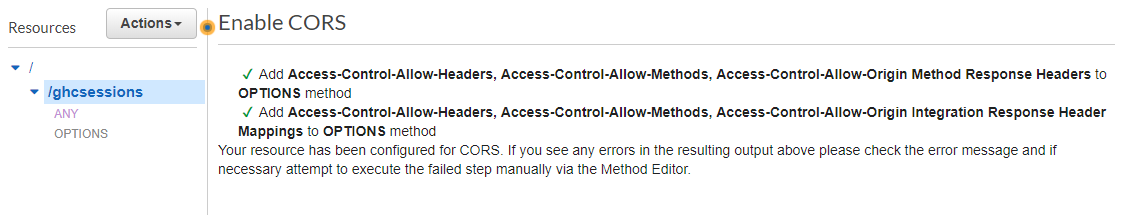
1. Click Save.
2. When you see the “Add Permission to Lambda Function” pop-up, select “Ok”.
3. With “ANY” (or “ghcsessions”) selected in your Resources tree, go ahead and click the “Actions” button again.
4. Click “Enable CORS”



1. Leave all the options as-is.



1. Click “Enable CORS and replace existing CORS headers”
2. Click “Yes, replace existing values” on the next dialog



That’s all! You have created your first API using Amazon API Gateway. Follow the directions below to create your next two APIs.

## get-all-dynamodb

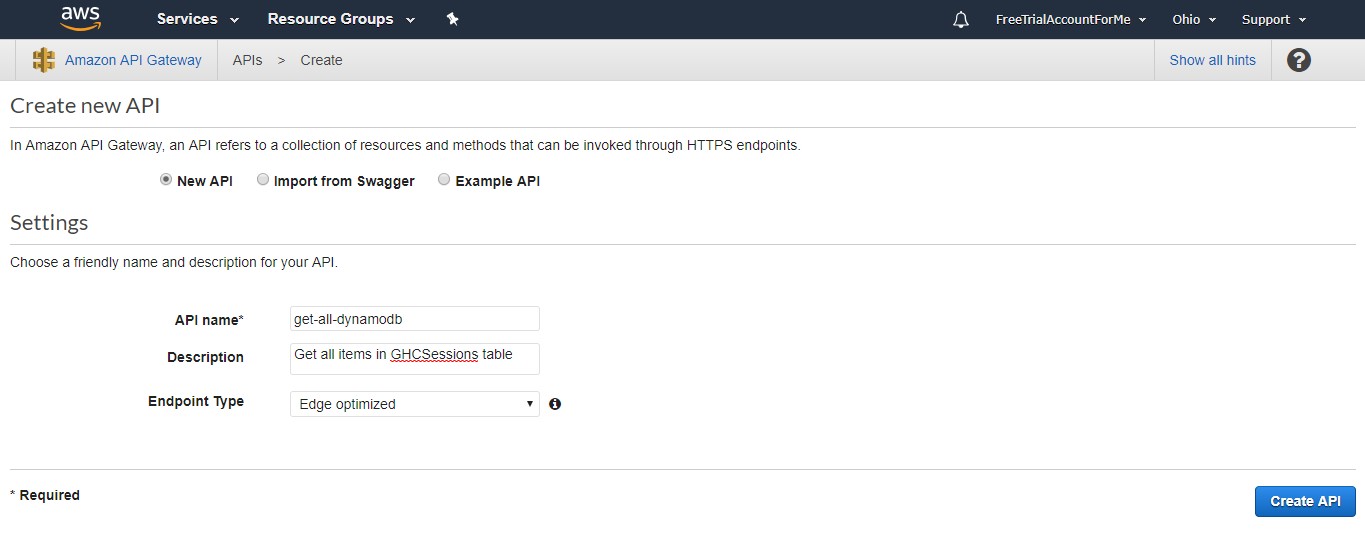
You will now create your next API which will get all items in the GHCSessions table.

1. On the Navigation bar at the top of the page, click on “Services”. A dropdown of all the services that AWS provides in the region will appear.
2. Find the “API Gateway” link under the “Networking and Content Delivery” section in the left column and select it. You can also type into the Search bar “API Gateway” and select it from the search suggestions
3. Click the “Create API” button to start building your next API.
4. Select the “New API” radio button
5. Enter the below details for your API

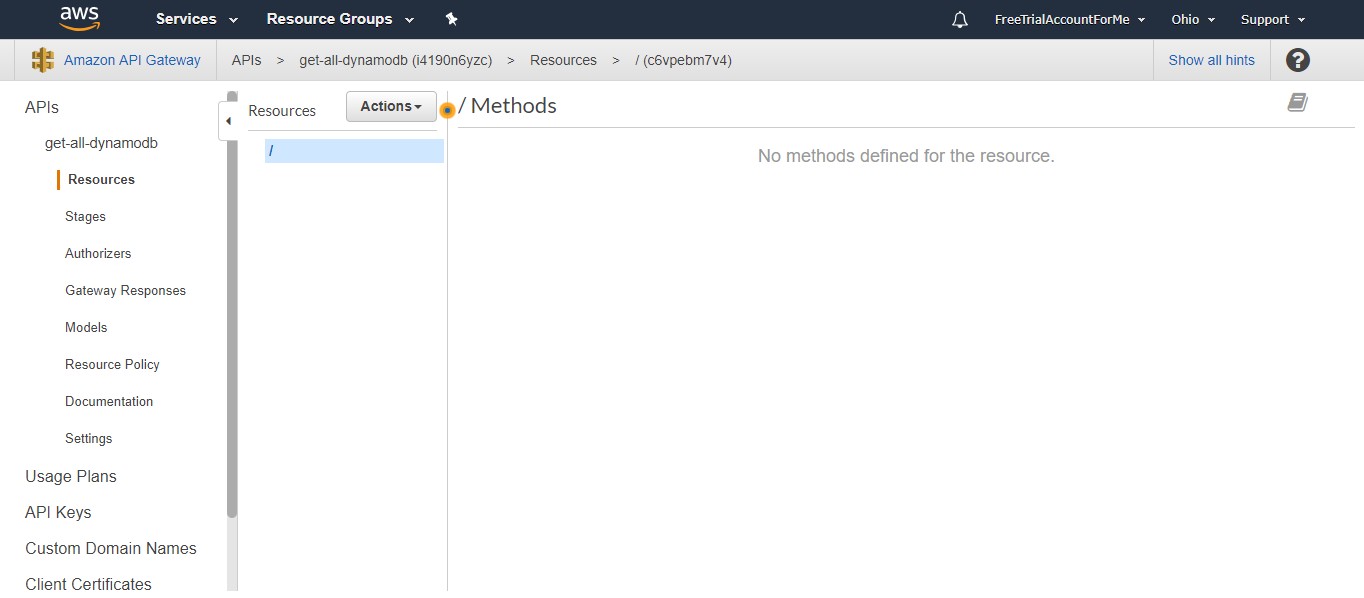
**API name\*:** get-all-dynamodb

**Description**: Get all items in GHCSessions table

**Endpoint Type**: Edge optimized



You will be taken to the API dashboard where you can define your methods and resources for your API.



1. Click the “Actions” Button and the dropdown, select “Create Resource”
2. Supply the following details:

Leave **Configure as proxy resource** unchecked.

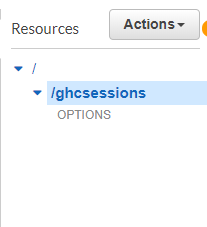
**Resource Name\*:** GHCSessions

**Resource Path\*:** ghcsessions

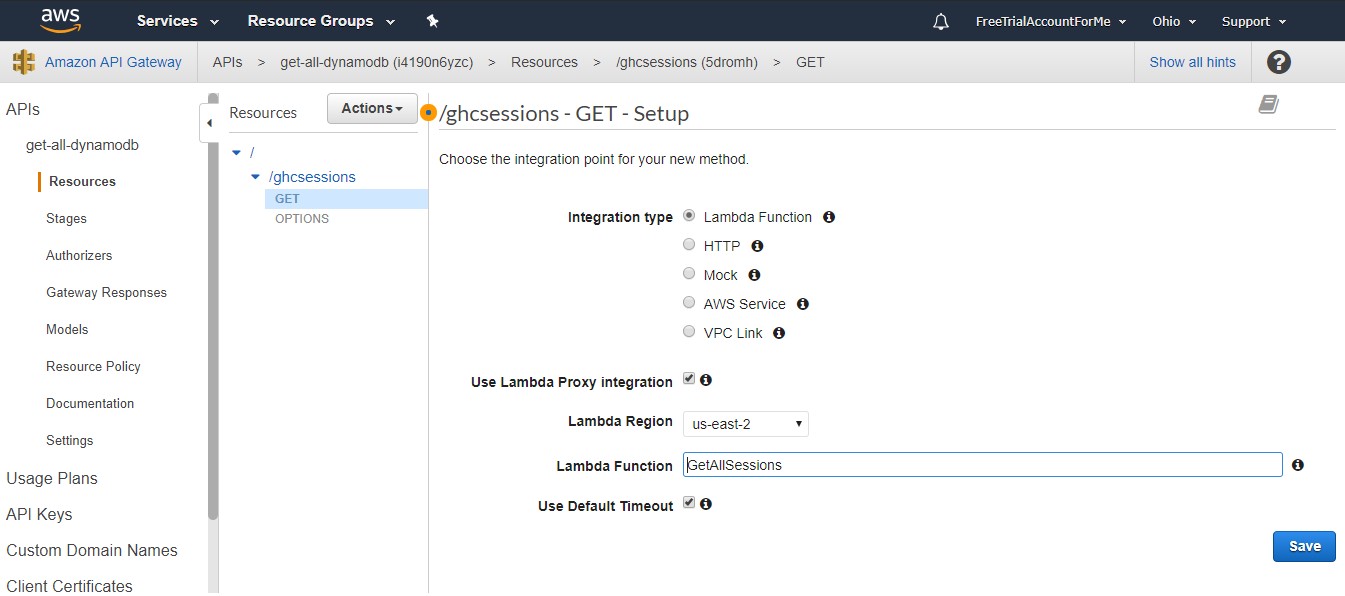
Select the **Enable API Gateway CORS** checkbox.



1. Click “Create Resource” button.
2. You should now see the resource that you just created appear in the Resources tree. Click on its name “/ghcsessions”.



1. With “/ghcsessions” selected in the Resources tree, click the “Actions” button again, and select “Create Method”
2. You will see that a dropdown menu element gets added to the tree below “OPTIONS”. In that dropdown, select “GET”, and then click the Check mark next to it.



1. Here is where we set up the API integration with Lambda. Fill out the below details:

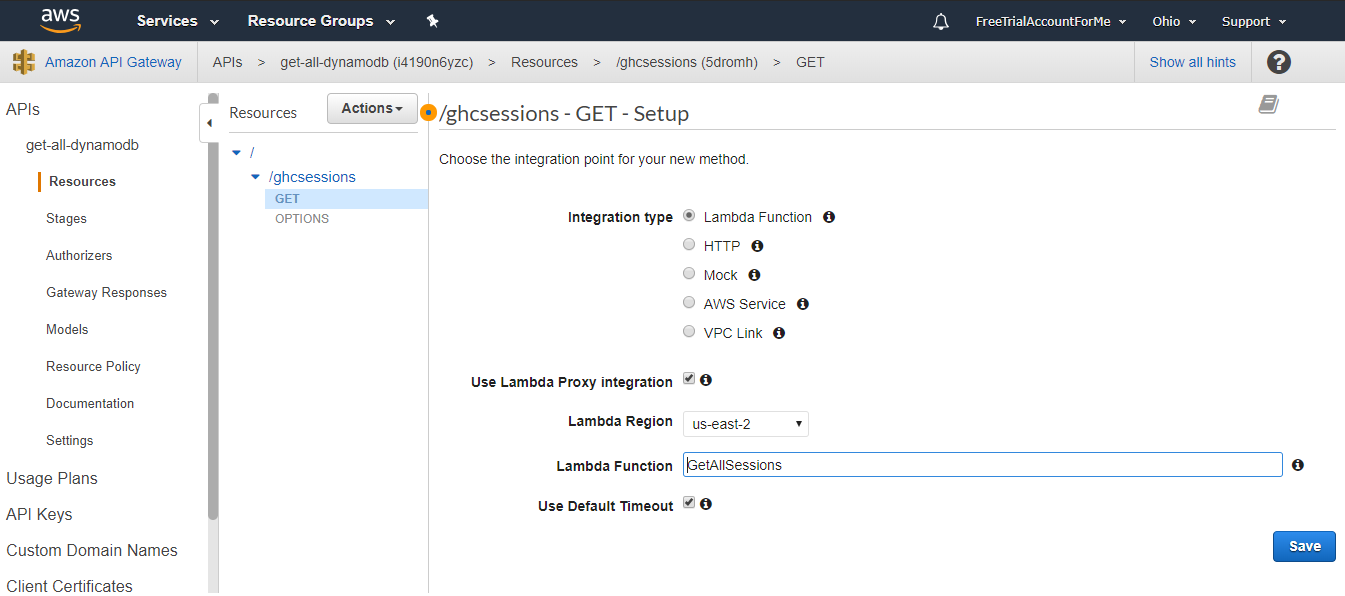
Select the **Lambda Function** radio button.

Select the **Use Lambda Proxy integration** checkbox

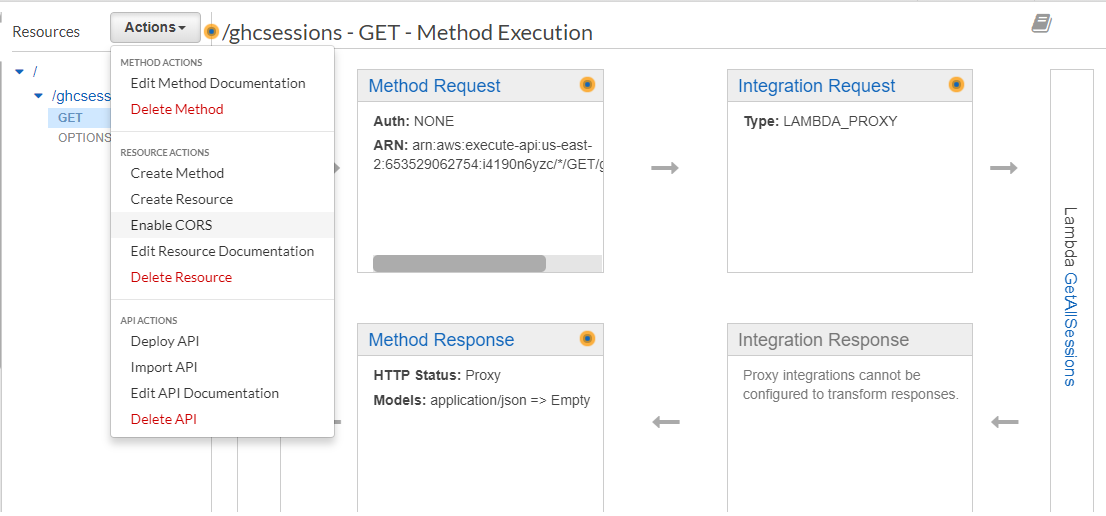
Select the same region that you have been using for this workshop (you should not need to change the option that is pre-selected)

In **Lambda Function,** enter the name of the first function that we created earlier that gets all the sessions in the table – **GetAllSessions**

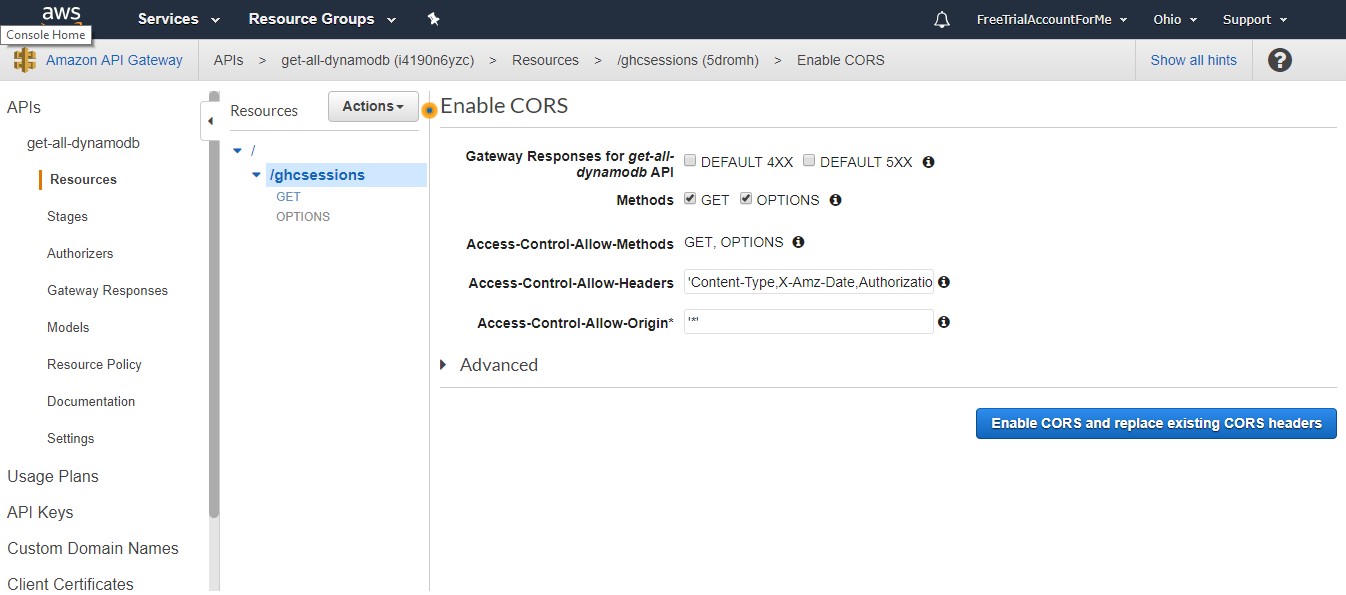
Leave “**Use Default Timeout**” as selected



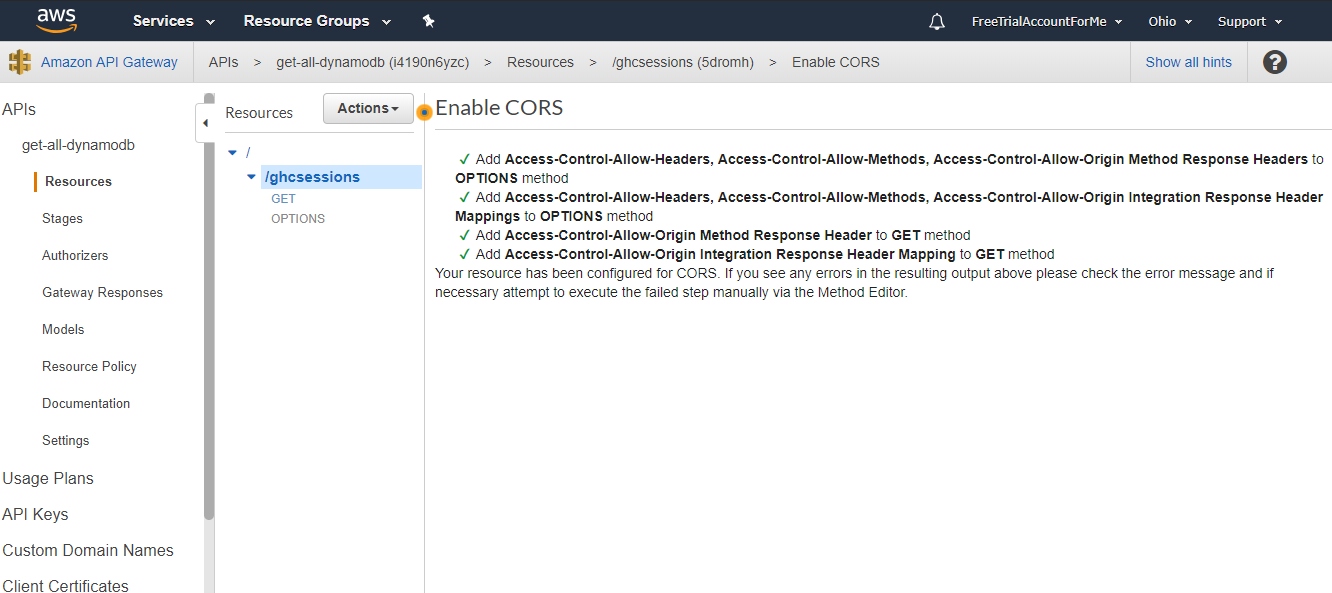
1. Click Save.
2. When you see the “Add Permission to Lambda Function” pop-up, select “Ok”.
3. With “GET” (or “ghcsessions”) selected in your Resources tree, go ahead and click the “Actions” button again.
4. Click “Enable CORS”



1. Leave all the options as-is.



1. Click “Enable CORS and replace existing CORS headers”
2. Click “Yes, replace existing values” on the next dialog



That is all. You have now created your second API. Next, you will create the last API for your application.

## update-dynamodb

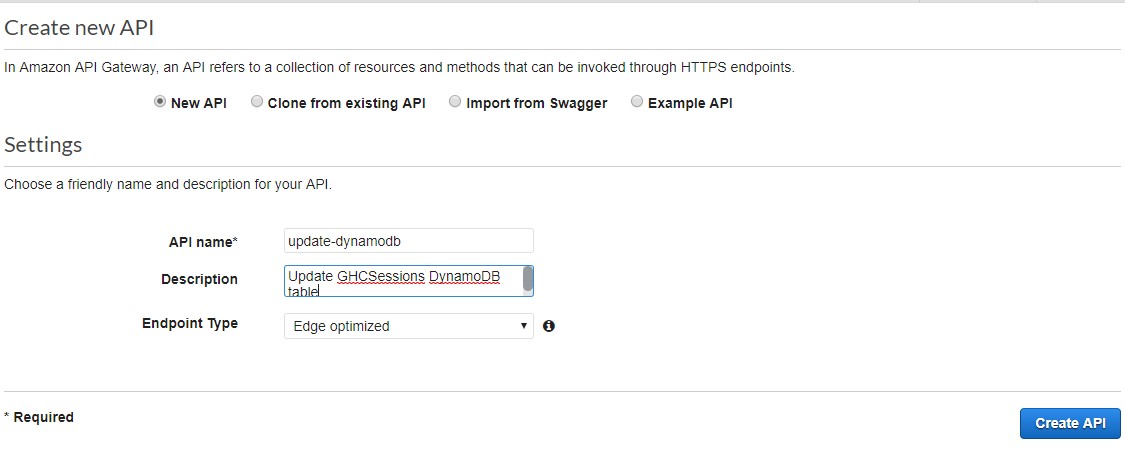
You will now create your last API which will get update an item in the GHCSessions table.

1. On the Navigation bar at the top of the page, click on “Services”. A dropdown of all the services that AWS provides in the region will appear.
2. Find the “API Gateway” link under the “Networking and Content Delivery” section in the left column and select it. You can also type into the Search bar “API Gateway” and select it from the search suggestions
3. Click the “Create API” button to start building your next API.
4. Select the “New API” radio button
5. Enter the below details for your API

**API name\*:** update-dynamodb

**Description**: Update GHCSessions DynamoDB table

**Endpoint Type**: Edge optimized



You will be taken to the API dashboard where you can define your methods and resources for your API.

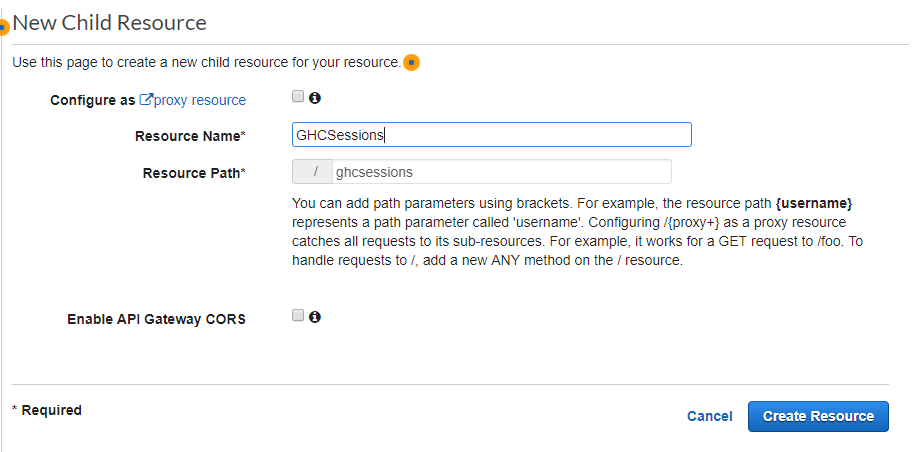
1. Click the “Actions” Button and the dropdown, select “Create Resource”
2. Supply the following details:

Leave **Configure as proxy resource** unchecked.

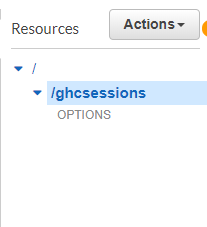
**Resource Name\*:** GHCSessions

**Resource Path\*:** ghcsessions

Leave the **Enable API Gateway CORS** checkbox unchecked (optional).



1. Click “Create Resource” button.
2. You should now see the resource that you just created appear in the Resources tree. Click on its name “/ghcsessions”.



*OPTIONS may or may not be present depending on whether you selected* ***Enable API Gateway CORS*** *for this resource, which is fine*

1. Click the “Actions” button again, and select “Create Resource”
2. Supply the following details:

Leave **Configure as proxy resource** unchecked.

**Resource Name\*:** id

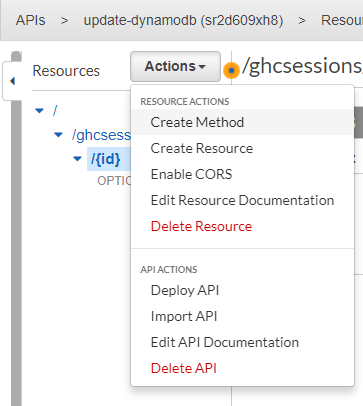
**Resource Path\*:** {id}

Select the **Enable API Gateway CORS** checkbox.

*\*Include the curly brackets around “id” in the Resource Path*



1. Click the “Create Resource” button
2. With “id” selected in the Resources tree, click the “Actions” button again, and select “Create Method”



1. You will see that a dropdown menu element gets added to the tree below “OPTIONS”. In that dropdown, select “POST”, and then click the Check mark next to it.
2. Here is where we set up the API integration with Lambda. Fill out the below details:

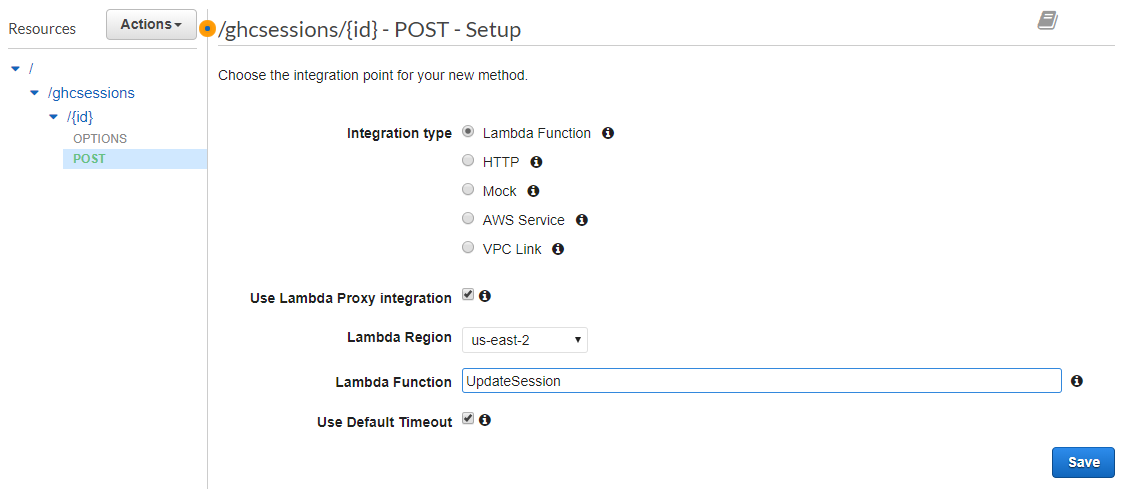
Select the **Lambda Function** radio button.

Select the **Use Lambda Proxy integration** checkbox

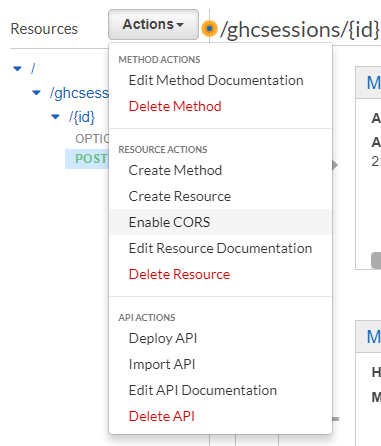
Select the same region that you have been using for this workshop (you should not need to change the option that is pre-selected)

In **Lambda Function,** enter the name of the first function that we created earlier that gets all the sessions in the table – **UpdateSession**

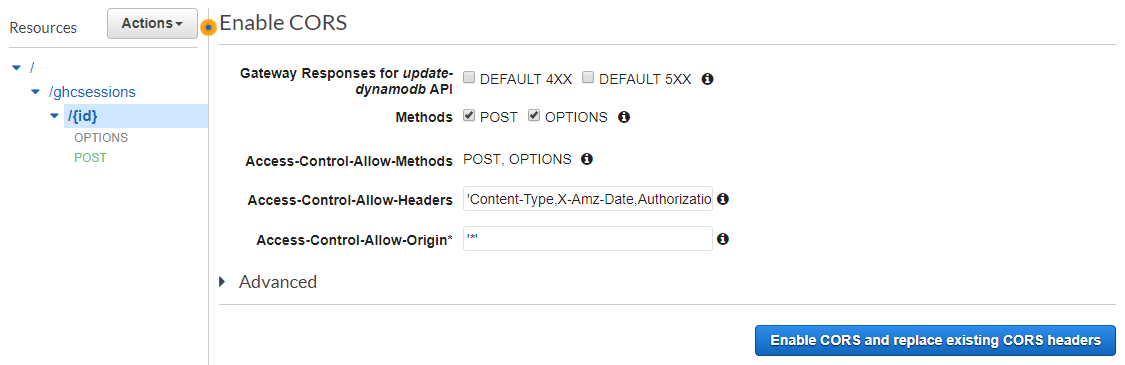
Leave “**Use Default Timeout**” as selected



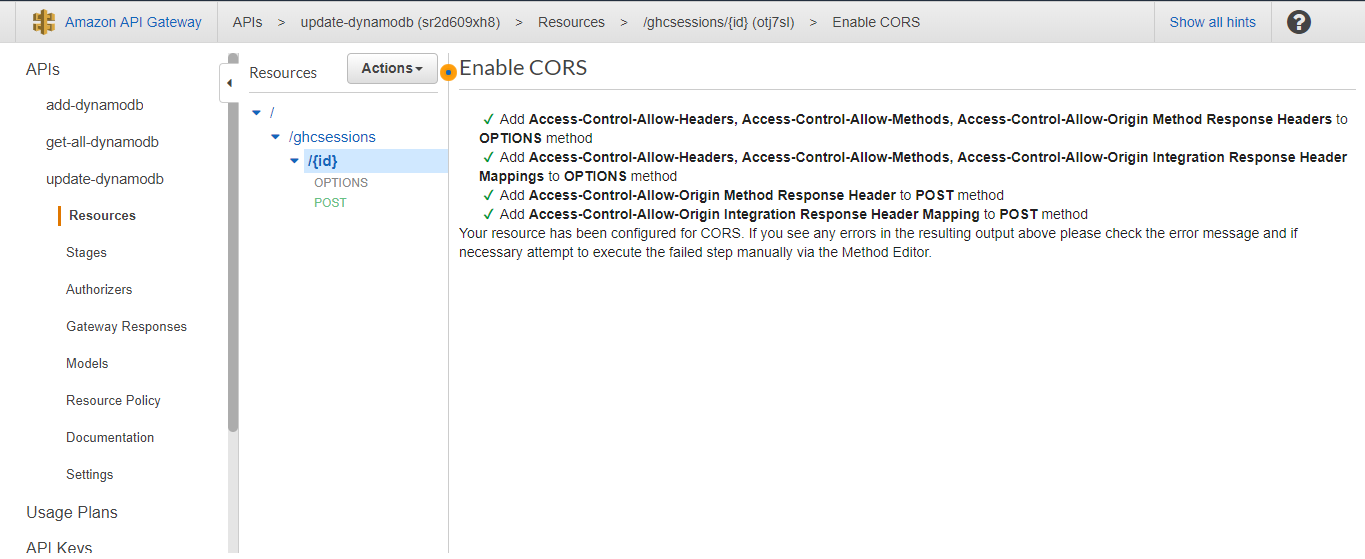
1. Click Save.
2. When you see the “Add Permission to Lambda Function” pop-up, select “Ok”.
3. With “POST” (or “{id}”) selected in your Resources tree, go ahead and click the “Actions” button again.
4. Click “Enable CORS”



1. Leave all the options as-is.



1. Click “Enable CORS and replace existing CORS headers”
2. Click “Yes, replace existing values” on the next dialog



You have created all the APIs for your GHC Scheduler application. Congratulations!

# Trial account limits

## AWS Free Tier (12 Month Introductory Period)

These free tier offers are only available to new AWS customers, and are available for 12 months following your AWS sign-up date. When your 12 month free usage term expires or if your application use exceeds the tiers, you simply pay standard, pay-as-you-go service rates (see each service page for full pricing details). Restrictions apply; see [offer terms](https://aws.amazon.com/free/terms/?ft=nt) for more details.

### Amazon API Gateway

* 1 Million API Calls per month\*

### Amazon Simple Storage Service (S3)

* 5 GB of [Amazon S3](https://aws.amazon.com/s3/) standard storage, 20,000 Get Requests, and 2,000 Put Requests\*

## AWS Free Tier (Non-expiring Offers)

These free tier offers do not automatically expire at the end of your 12 month AWS Free Tier term and are available to all AWS customers.

### Amazon DynamoDB

* 25 GB of Storage, 25 Units of Read Capacity and 25 Units of Write Capacity – enough to handle up to 200M requests per month with [Amazon DynamoDB](https://aws.amazon.com/dynamodb/).\*\*

### AWS Lambda

* 1,000,000 free requests per month\*\*
* Up to 3.2 million seconds of compute time per month\*\*