# 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. Follow the instructions on the screen to create your account
4. When you are asked for the account usage type, select Personal, unless you plan to reuse this account for Professional reasons

Your account should complete set up within five minutes.

## Verify that you have npm installed on your computer

1. Open your terminal (Windows PowerShell on PC or Terminal on Mac) and run the followingcommand:

8c85903a4bb6:~ agathao$ npm -v

6.2.0

1. If a version of npm is returned, you can proceed to Section 2. If instead, you get an error that npm is not recognized, then you will need to install it.
   1. On a mac:
      1. Run the command: brew -v

8c85903a4bb6:~ agathao$ brew -v

Homebrew 1.7.2

Homebrew/homebrew-core (git revision 41d26; last commit 2018-08-31)

* + 1. If you have a version returned, that means that you have Homebrew installed. If you receive an error instead, then install it by running

8c85903a4bb6:~ agathao$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

* + 1. Once the installation is complete, run the command: brew install node

8c85903a4bb6:~ agathao$ brew install node

* 1. On a pc:
     1. Download the windows installer from <https://nodejs.org/en/>
     2. Run the installer and follow the prompts
     3. Restart your computer

To confirm that everything has been installed properly, run the command: npm -v one more time and verify that a version is returned

# Building your Serverless application

During this workshop you will be building an application that allows you to see all sessions being held at GHC, and add them to your calendar. The static files for the website will be stored on Amazon S3. The APIs necessary will be created via Amazon API Gateway and powered by AWS Lambda. The data for the sessions will be stored in Amazon DynamoDB.

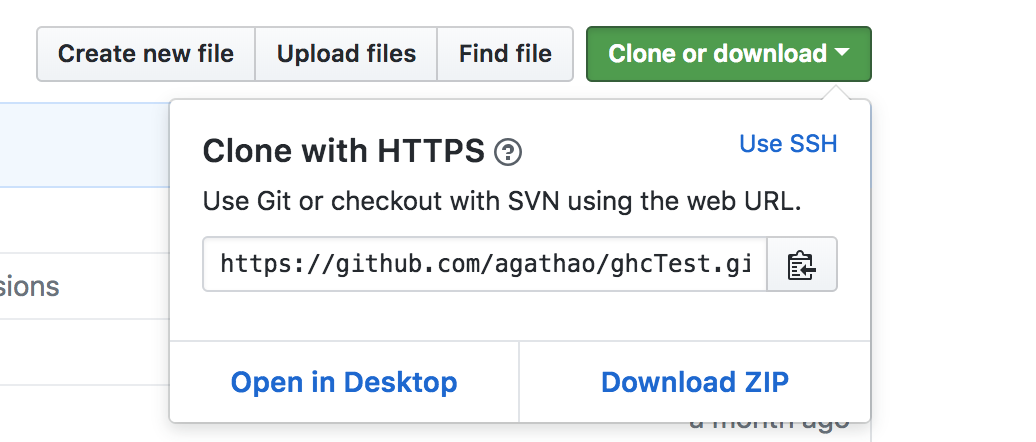
* 1. Downloading website files

1. Open your terminal, and choose the directory under which you will place the files related to this workshop. Then clone the github repository to the folder of your choice. We will use the *Documents* folder.

8c85903a4bb6:~ agathao$ cd Documents/

8c85903a4bb6:Documents agathao$ git clone https://github.com/agathao/ghcTest.git

If you don’t have git on your computer, you can download the repository from github.



1. There should be a *ghcTest* folder created. Go into it and run npm install. This will install all the packages necessary for your site to run.

8c85903a4bb6:Documents agathao$ cd ghcTest/

8c85903a4bb6:ghcTest agathao$ npm install

1. Once the installation is complete, run the command npm start. The application will start under http://localhost:4200/. Navidate to it and see the application running

8c85903a4bb6:ghcTest agathao$ npm start

> agenda@0.0.0 start /Users/agathao/Documents/ghcTest

> ng serve

\*\* Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ \*\*

This initial version executes using mocked data with some GHC sessions. You will be able to see the sessions available and their details, but not to add or remove them from your calendar.

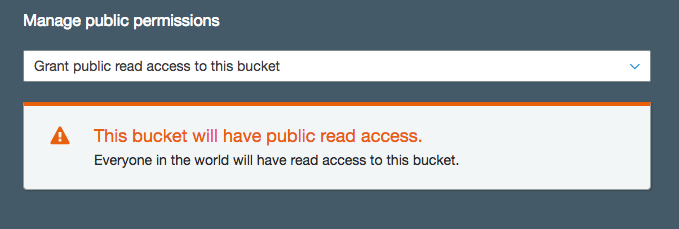
* 1. Storing the files on Amazon S3

1. Let’s prepare the files to be downloaded to S3. Open youe terminal and under the ghcTest folder run the build command. Once the command executes you will see that you have a dist folder under ghcTest and inside of it, there will be a folder called agenda.

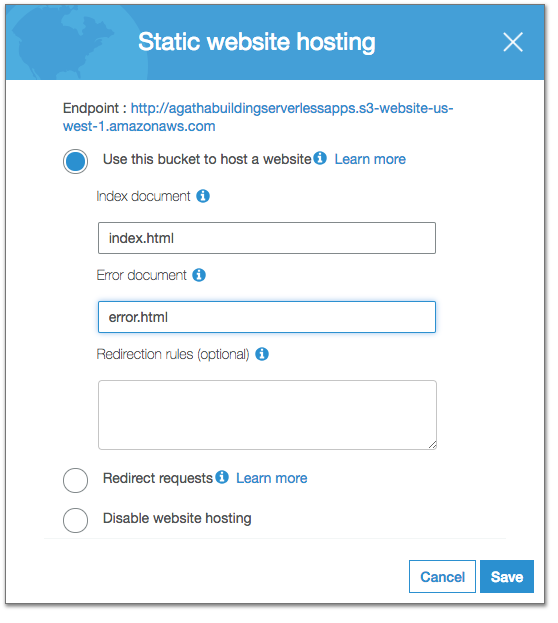
8c85903a4bb6:ghcTest agathao$ npm run-script build

8c85903a4bb6:ghcTest agathao$ cd dist/agenda/

1. Now we will upload the contents of the agenda folder to Amazon S3. Go to the S3 console at <https://s3.console.aws.amazon.com/>. You will need to sign in if you haven’t already.
2. Click on create bucket on the top right
   1. Choose a bucket name. Do not forget that it should be unique across all of AWS. We suggest <name>buildingserverlessapps. E.g.: agathabuildingserverlessapps
   2. For Region, select US West (N. California).
      1. A region is a geographic location where your data is stored
      2. Within a region, there are multiple Availability Zones (AZ). There are isolated locations connected to each other via low latency links.
      3. When your data is stored in a region, it is replicated to at least 3 AZ. This means that even if 2 AZ go down (which is very unlikely), your data is still available from the 3rd.
   3. Click on “Next”. For this workshop we will not be setting any of the properties on this screen. However, you can click on the “Learn more” links next to them to understand what they do.
   4. Click on “Next”. For the permissions, you can see that your account has already been given permissions. However, we want the contents of the application to be accessible to anyone. Under “Manage public permissions” select “Grant public access to this bucket”



* 1. Click on “Next” and then on “Create Bucket”
  2. Now that your bucket is created, click on your bucket name and then go to the “Properties” tab at the top.
  3. Click on “Static website hosting” and select “Use this bucket to host a website”. Then configure the index and the error documents as per the below. Also make note of the Endpoint that you are given. Lastly, click on save:



1. There are many ways of uploading your files. We will do the simplest of them: Go back to the overview tab and click on “Upload”.
   1. Select all files under the agenda folder and drag them to the screen where it says upload.
   2. Click “Next”. Under “Manage public permissions”, select “Grant public read access to this object(s)”
   3. Click “Upload”
2. When your upload is complete, your files will be accessible via the S3 bucket link. If you don’t remember your link, you can go to the “Properties” tab and click on “Website hosting”
3. Open your browser and go to URL to see the site running

## DynamoDB

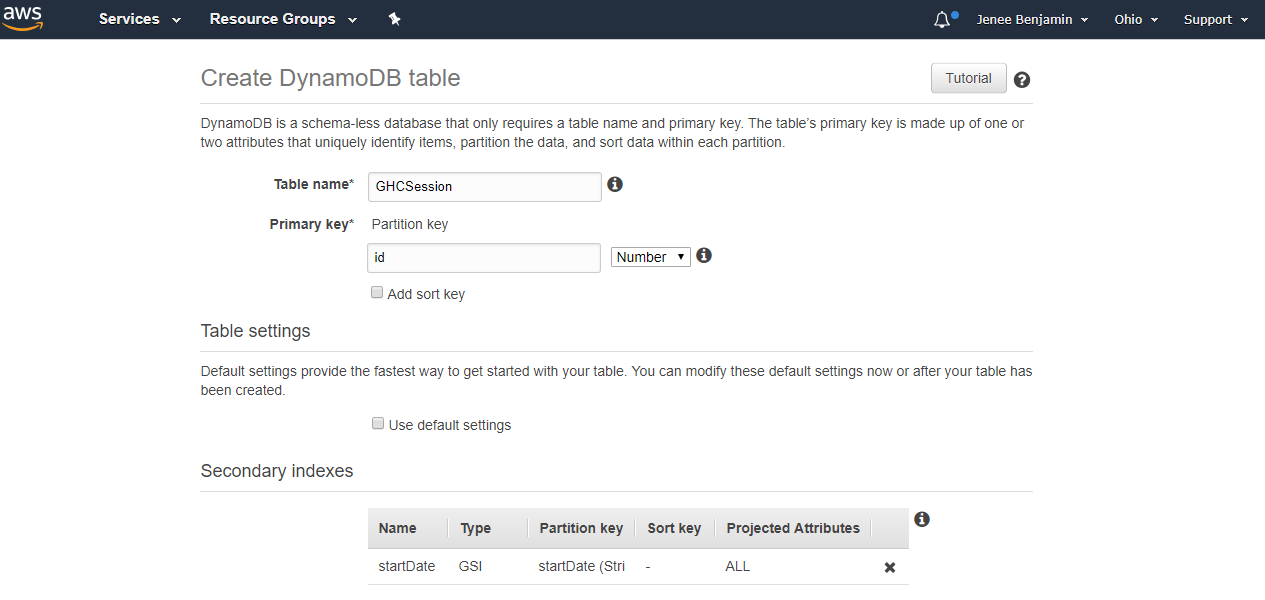
You will now create the DynamoDB table for your application. This table will store all the GHC session information. Below are the steps:

1. Go to the Dynamo DB console at <https://us-east-2.console.aws.amazon.com/dynamodb>. Confirm at the top right that your region is displayed as “Ohio”
2. On the DynamoDB homepage, go ahead and read about DynamoDB and click “Create Table” when you are ready.
3. 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.

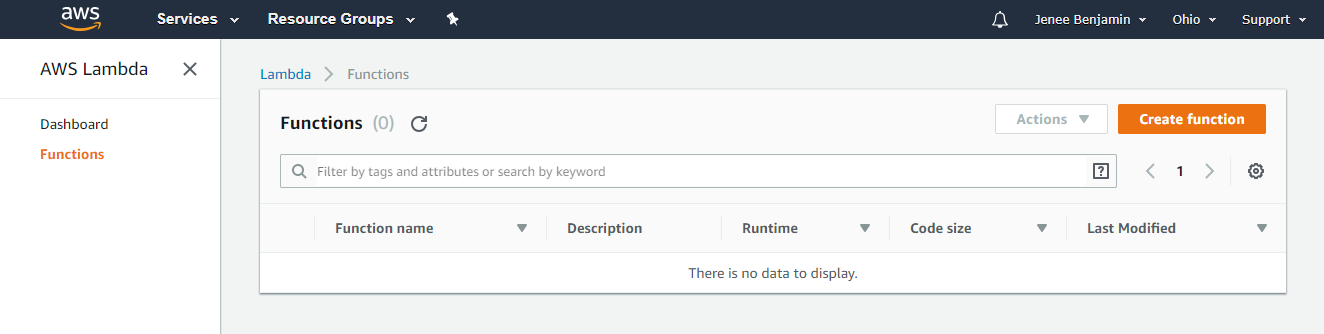
## AWS Lambda

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. Open the AWS Lambda console at <https://us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions> . Confirm at the top right that your region is displayed as “Ohio”

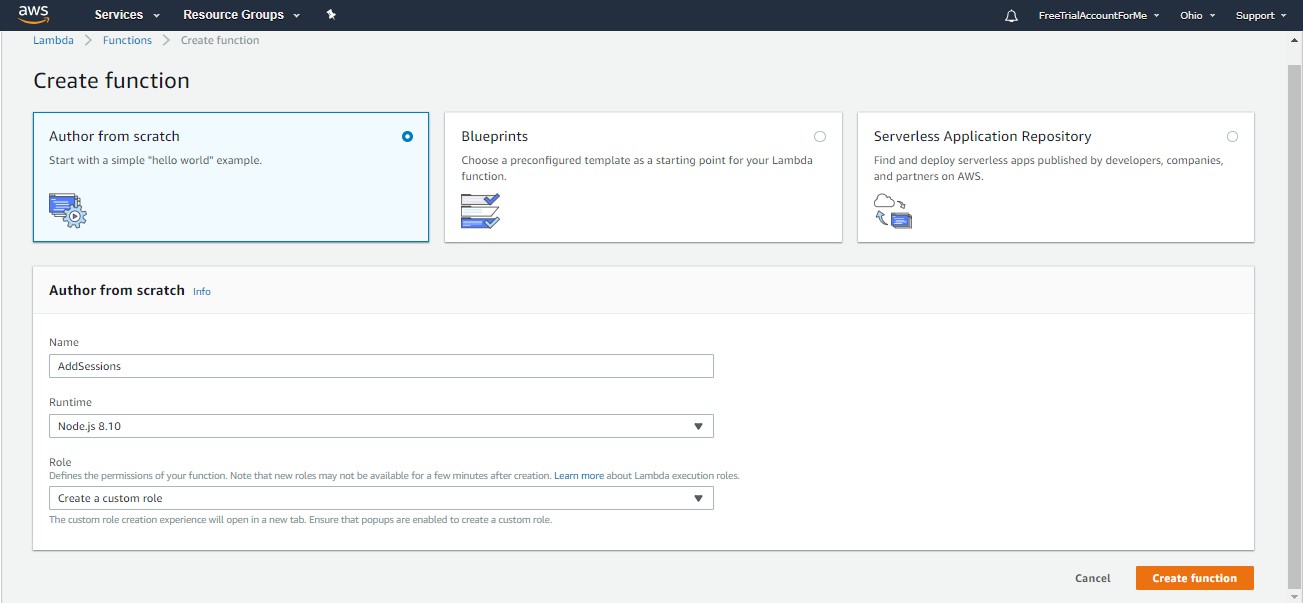


1. Select the “Create function” button
2. Select the first card titled “Author from scratch”
3. Fill out the form for the Lambda with the following details

**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.

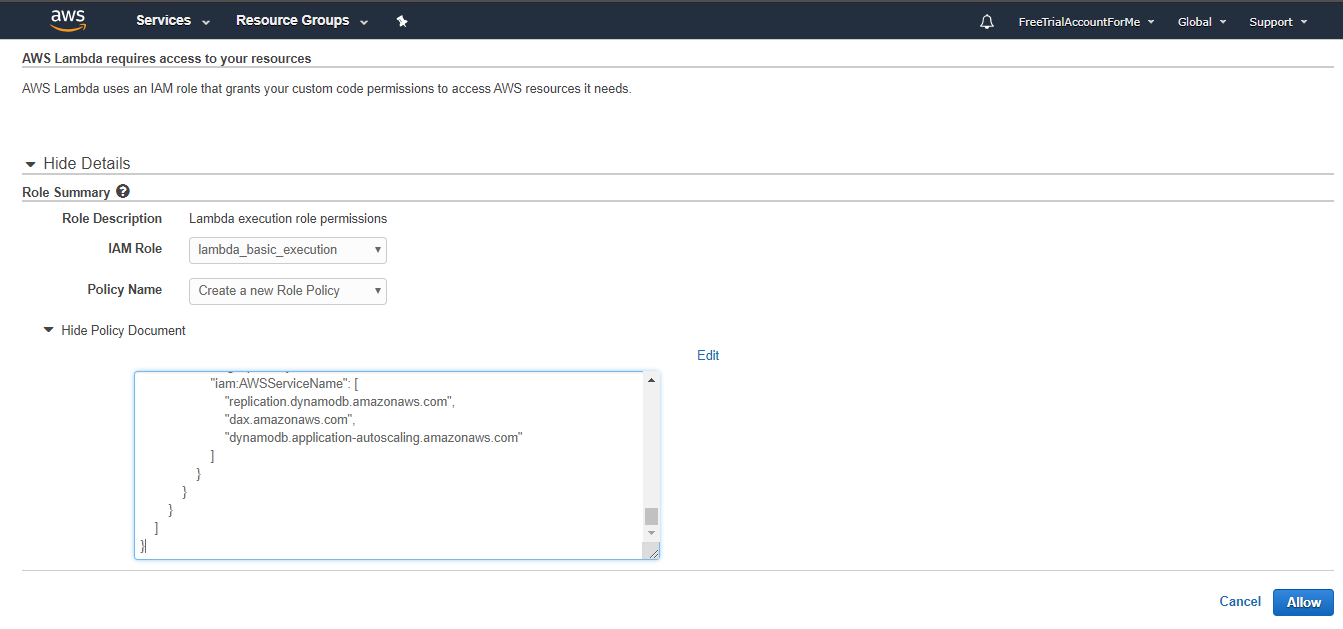
1. In the pop-up window that opens, you will create a new role for this function. You will use this role for this Lambda and the other you create. In the window that opens, enter the following details to create your role.

**Role Description:** Lambda execution role permissions

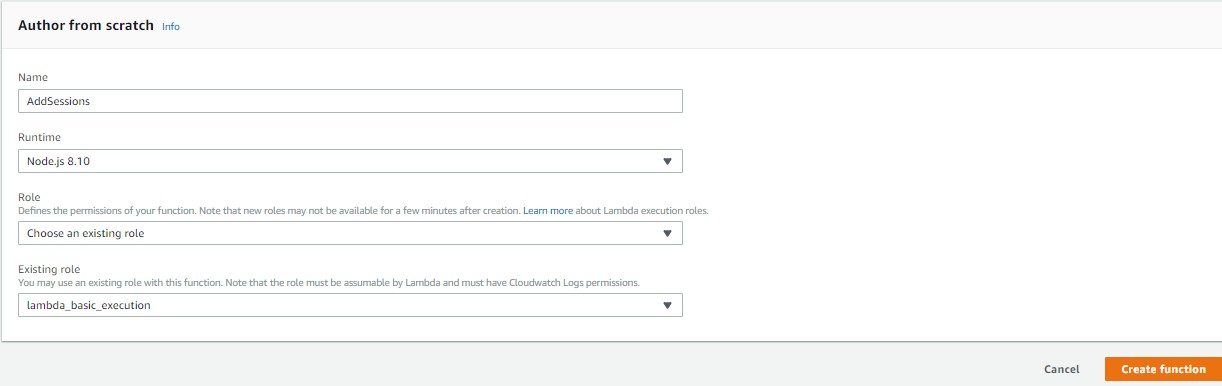
**IAM Role:** Create a new IAM role

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

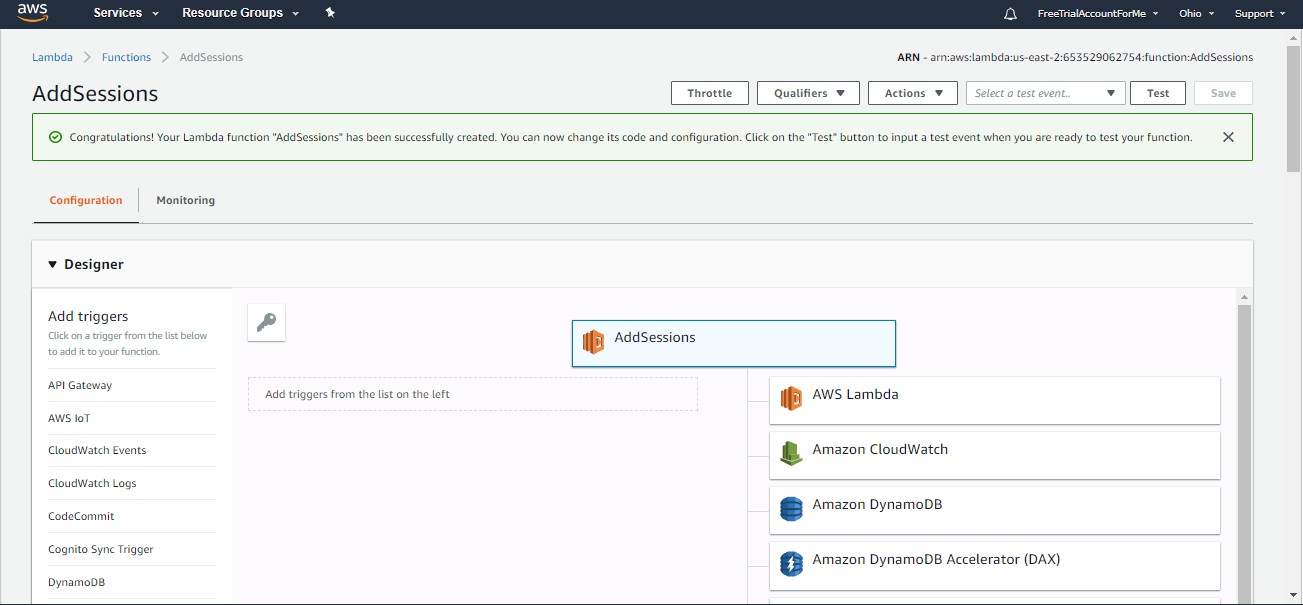
1. View Policy Document and click Edit. The Policy Document textbox should become modifiable.
2. On the GitHub repository, open the iam\_role.txt file located at <https://github.com/JeneeB/GHC2018SchedulerApplication/blob/master/lambda/iam/dynamodb_full_access.txt>
3. Copy the entire file.
4. Paste the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

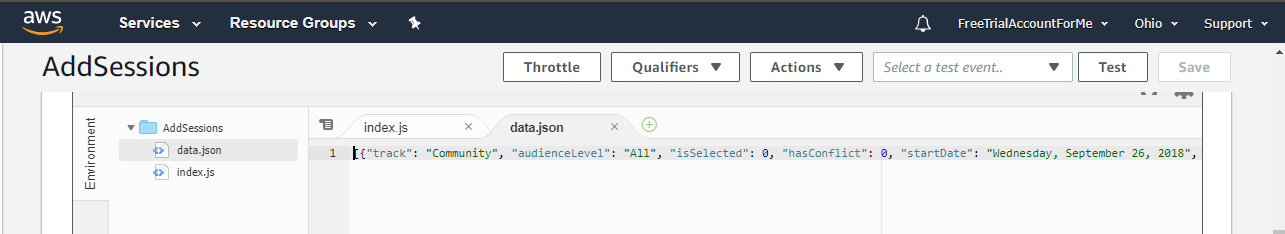
1. Go to File, then New File.
2. Open the GitHub repository and navigate to <https://github.com/JeneeB/GHC2018SchedulerApplication/blob/master/lambda/functions/addSessions/data.json>
3. Copy the data from that file and paste it into the AWS Lambda function code editor for the new file that you have open.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

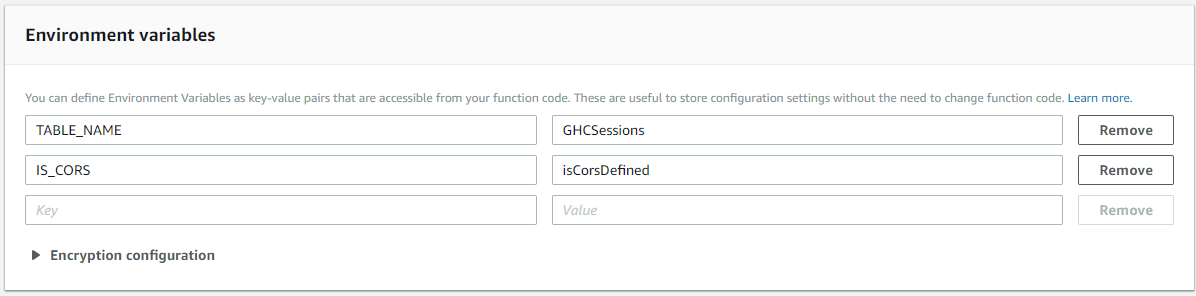
1. Your Function code section should look similar to the below



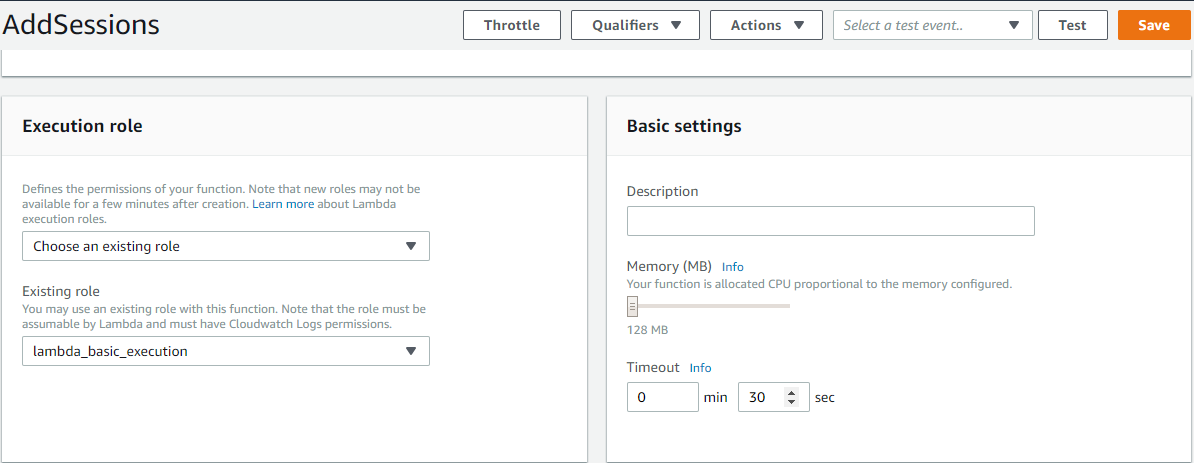
1. Scroll down to the “Environment variables” section on the Lambda detail page. You want to add two Environment variables:

**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.

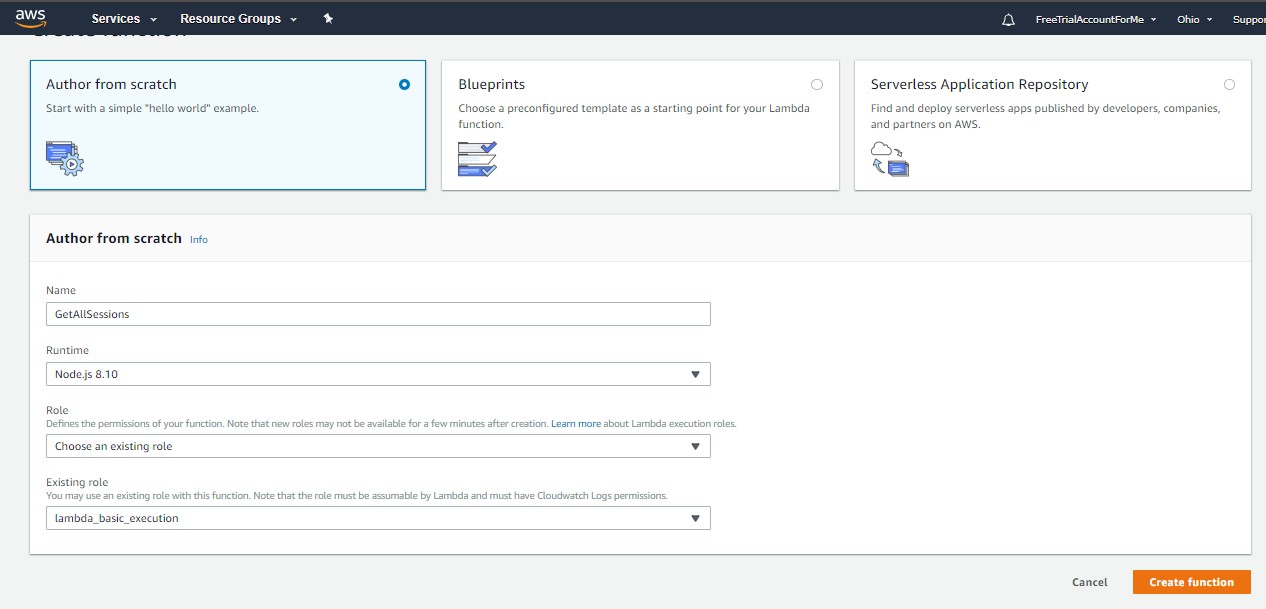
1. Open the AWS Lambda console at https://us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions. Confirm at the top right that your region is displayed as “Ohio”
2. Click the “Create Function” button
3. Select the “Author from scratch” card
4. Fill out the form for the Lambda with the following details

**Name**: GetAllSessions

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

**Role**: Choose an existing role

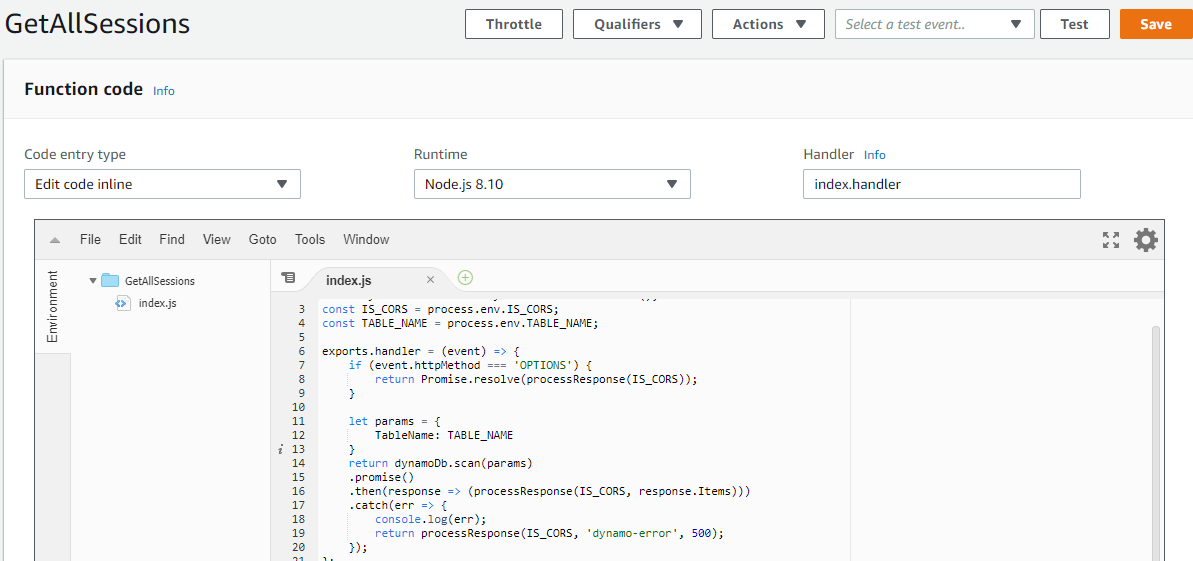
**Existing role**: lambda\_basic\_execution



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.

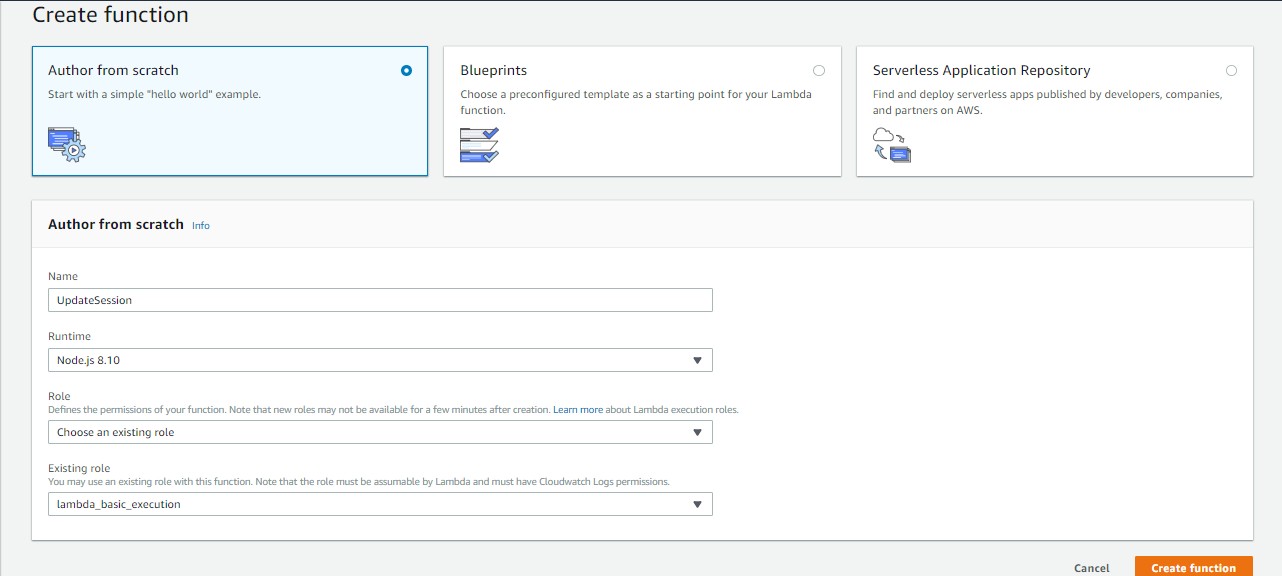
1. Open the AWS Lambda console at <https://us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions> . Confirm at the top right that your region is displayed as “Ohio”
2. Click the “Create Function” button
3. Select the “Author from scratch” card
4. Fill out the form for the Lambda with the following details

**Name**: UpdateSession

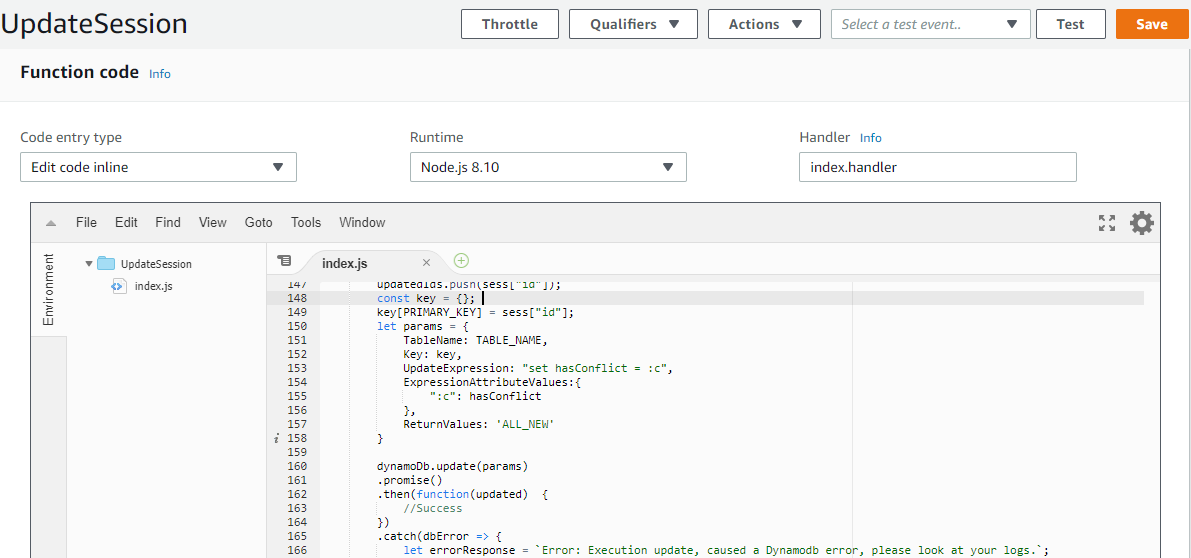
**Runtime**: **Node.js 8.10** *(not 6.10)*

**Role**: Choose an 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 UpdateSession 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.

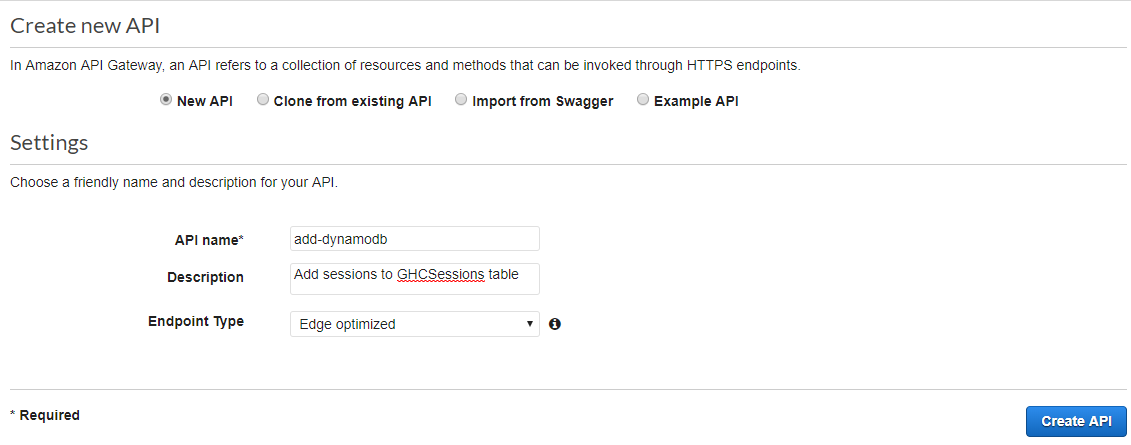
## add-dynamodb

1. Open the Amazon API Gateway console at <https://us-east-2.console.aws.amazon.com/apigateway/home?region=us-east-2#/apis> . Confirm at the top right that your region is displayed as “Ohio”
2. Click “Get Started” or the “Create API” button to start building your first API.
3. Select the “New API” radio button
4. Enter the below details for your API

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

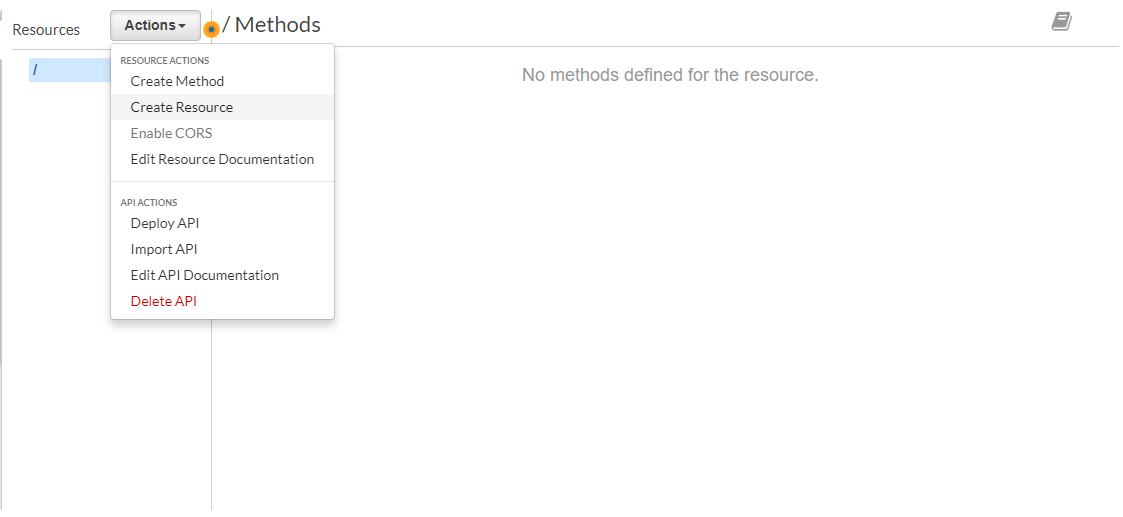
**Description**: Add sessions to GHCSessions table

**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.

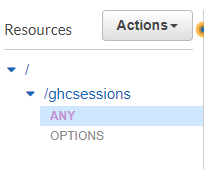
**Resource Name\*:** GHCSessions

**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. With “/ghcsessions” selected in the Resources tree, 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:

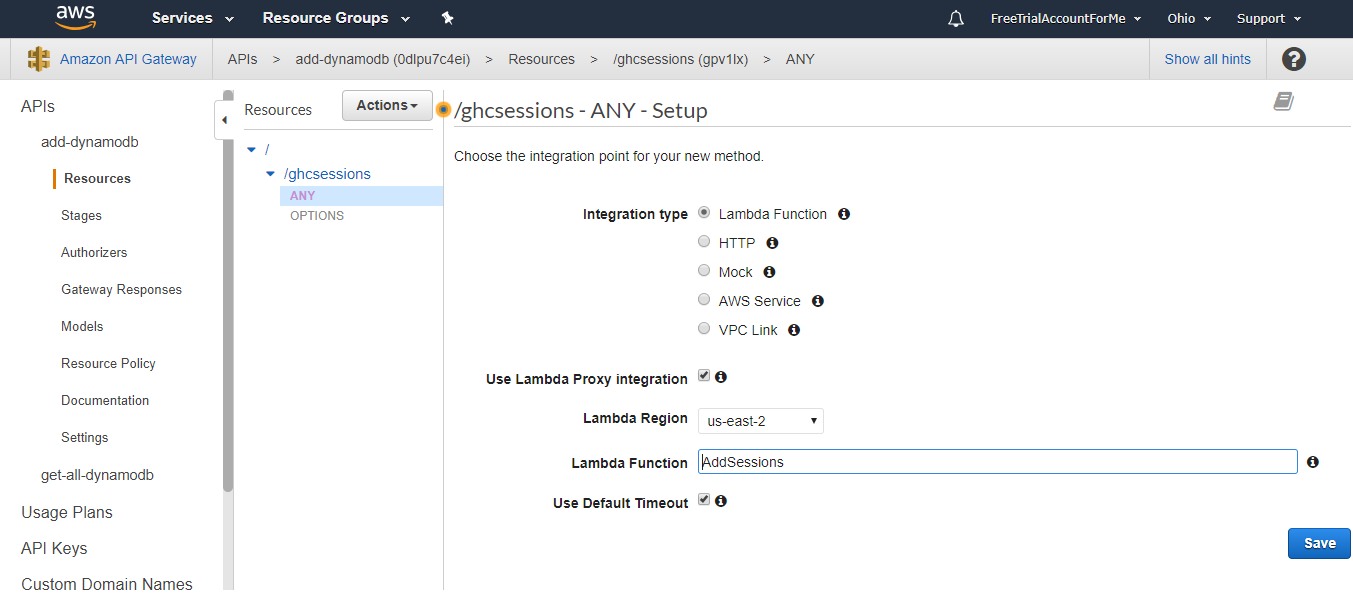
Select the **Lambda Function** radio button.

Select the **Use Lambda Proxy integration** checkbox

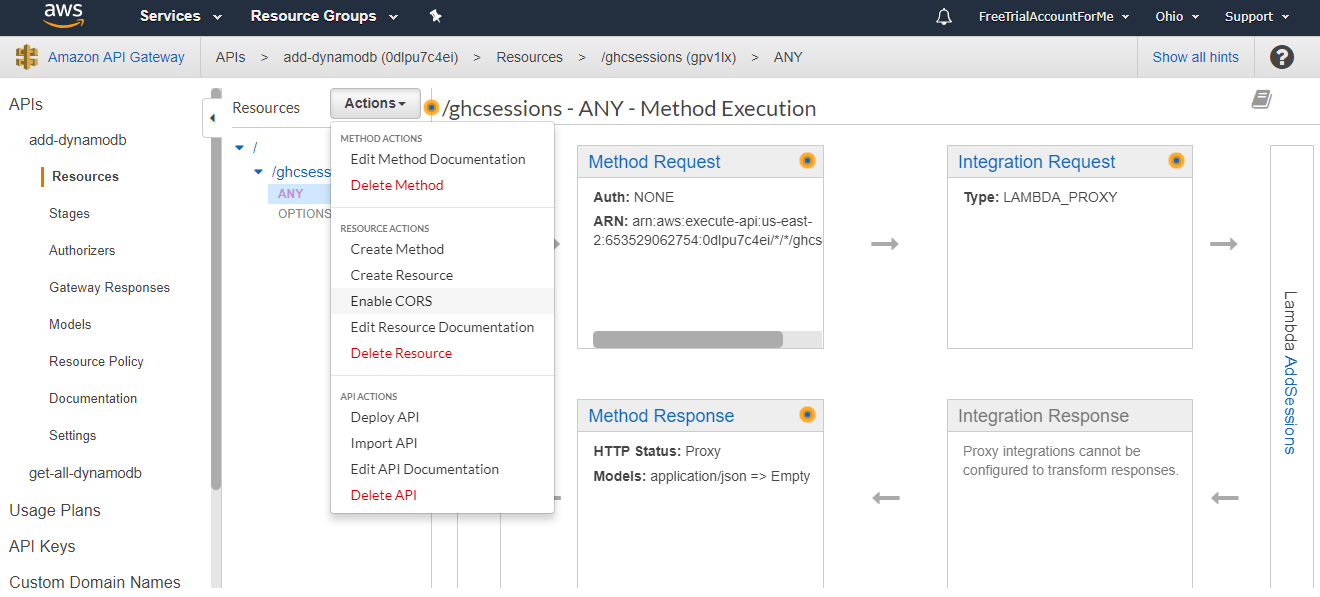
Select the same region that you have been using for this workshop (you should not need to change the option that is pre-selected)

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

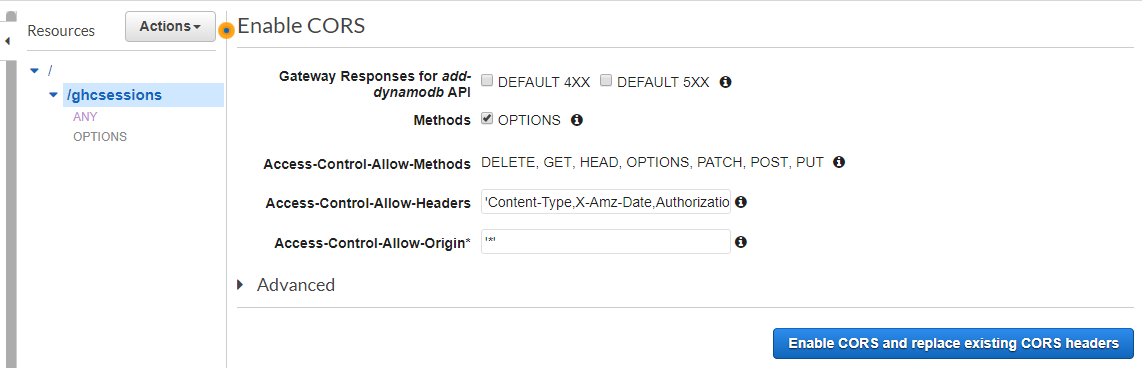
Leave “**Use Default Timeout**” as selected



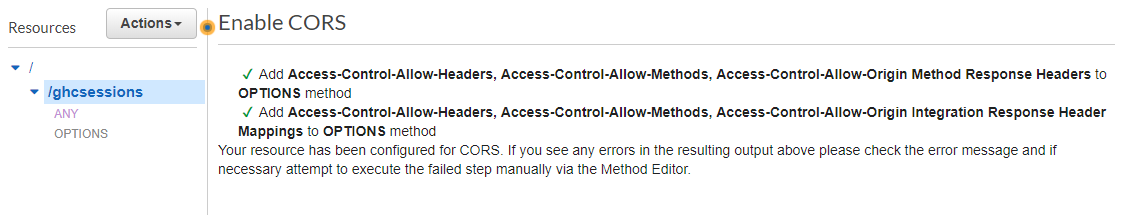
1. Click Save.
2. When you see the “Add Permission to Lambda Function” pop-up, select “Ok”.
3. With “ANY” (or “ghcsessions”) selected in your Resources tree, go ahead and click the “Actions” button again.
4. Click “Enable CORS”



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 and you should see the configuration confirmation



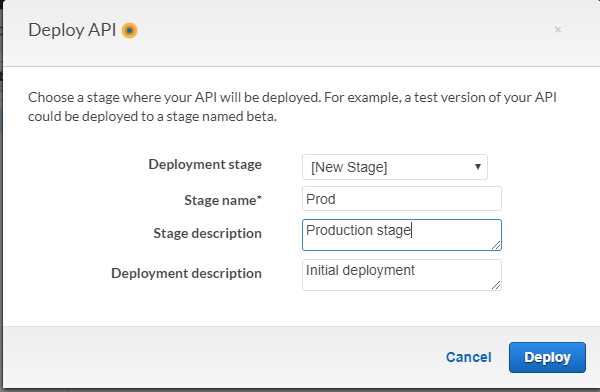
1. Lastly, you will deploy your API. Click on “Actions” then, “Deploy API”. Fill in the details for your deployment stage in the dialog box.

**Deployment stage**: [New Stage]

**Stage name\*:** Prod

**Stage description**: <Enter any description>

**Deployment description**: Initial deployment



1. Click “Deploy”

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. Open the Amazon API Gateway console at <https://us-east-2.console.aws.amazon.com/apigateway/home?region=us-east-2#/apis> . Confirm at the top right that your region is displayed as “Ohio”
2. Click the “Create API” button to start building your next API.
3. Select the “New API” radio button
4. 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”.
3. With “/ghcsessions” selected in the Resources tree, 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 “GET”, and then click the Check mark next to it.
5. Here is where we set up the API integration with Lambda. Fill out the below details:

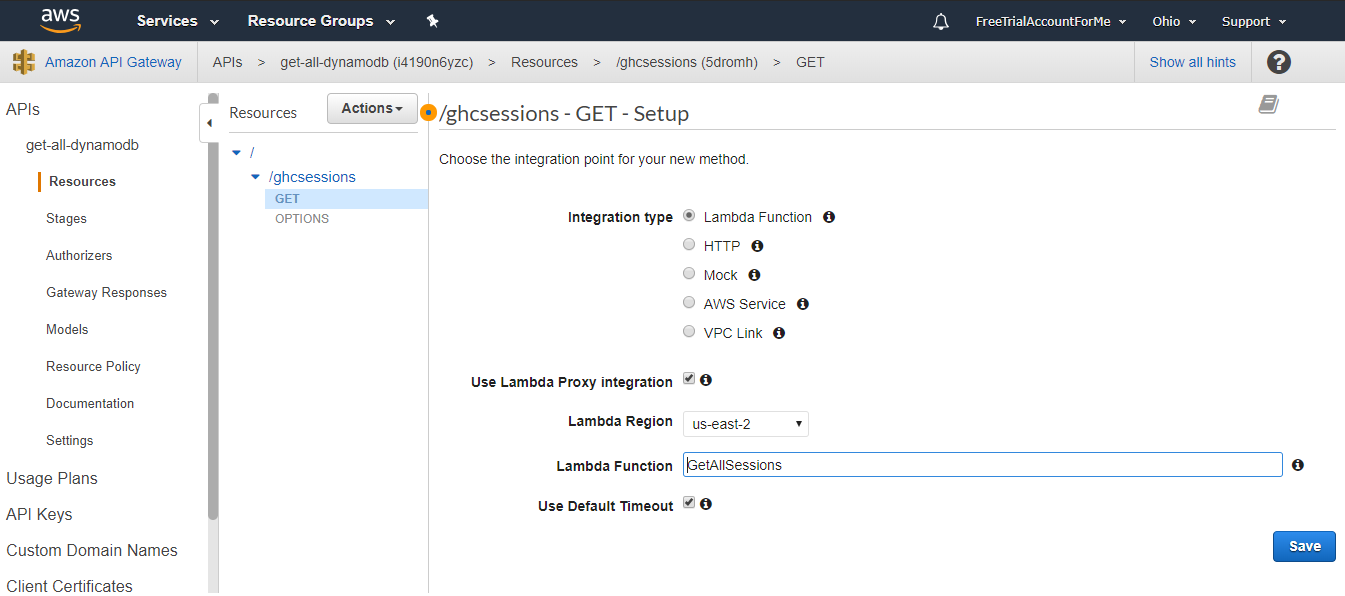
Select the **Lambda Function** radio button.

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”
5. Leave all the options as-is on the “Enable CORS” page.
6. Click “Enable CORS and replace existing CORS headers”
7. Click “Yes, replace existing values” on the next dialog
8. Lastly, you will deploy your API. Click on “Actions” then, “Deploy API”. Fill in the details for your deployment stage in the dialog box.

**Deployment stage**: [New Stage]

**Stage name\*:** Prod

**Stage description**: <Enter any description>

**Deployment description**: Initial deployment

1. Click “Deploy”

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. Open the Amazon API Gateway console at <https://us-east-2.console.aws.amazon.com/apigateway/home?region=us-east-2#/apis> . Confirm at the top right that your region is displayed as “Ohio”
2. Click the “Create API” button to start building your next API.
3. Select the “New API” radio button
4. 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.

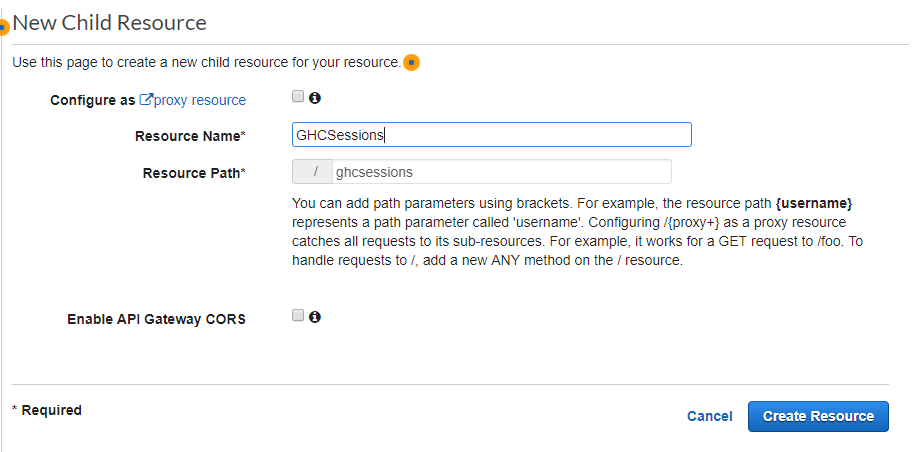
1. Click the “Actions” Button and the dropdown, select “Create Resource”
2. Supply the following details:

Leave **Configure as proxy resource** unchecked.

**Resource Name\*:** GHCSessions

**Resource Path\*:** ghcsessions

Leave the **Enable API Gateway CORS** checkbox unchecked



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. With “/ghcsessions” selected in the Resources tree, click the “Actions” button again, and select “Create Resource”
4. Supply the following details:

Leave **Configure as proxy resource** unchecked.

**Resource Name\*:** id

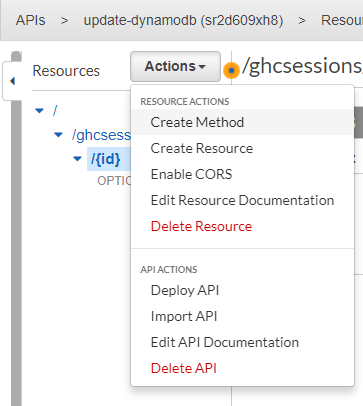
**Resource Path\*:** {id}

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:

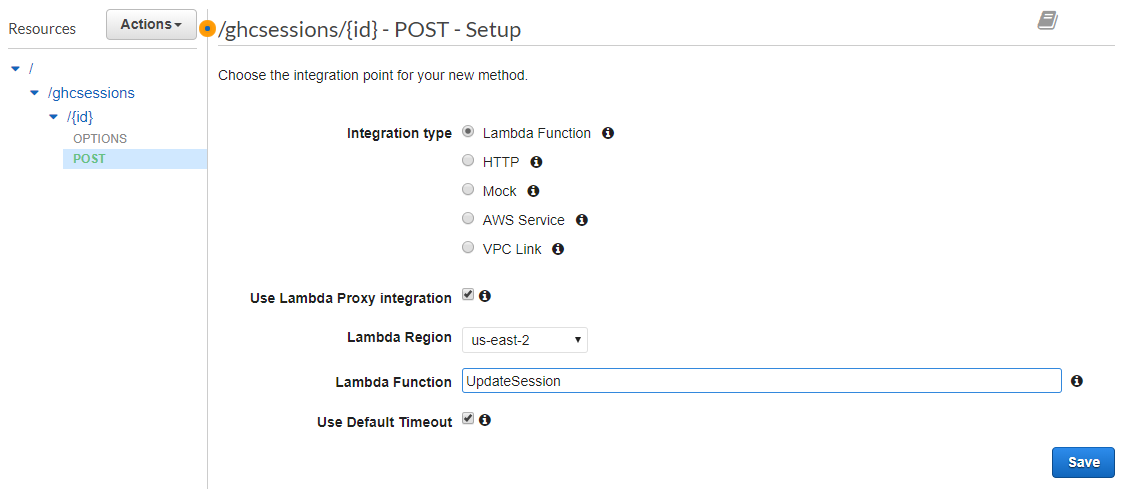
Select the **Lambda Function** radio button.

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”
5. Leave all the options as-is on the “Enable CORS” page.
6. Click “Enable CORS and replace existing CORS headers”
7. Click “Yes, replace existing values” on the next dialog and you should see the configuration confirmation
8. Lastly, you will deploy your API. Click on “Actions” then, “Deploy API”. Fill in the details for your deployment stage in the dialog box.

**Deployment stage**: [New Stage]

**Stage name\*:** Prod

**Stage description**: <Enter any description>

**Deployment description**: Initial deployment

1. Click “Deploy”

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