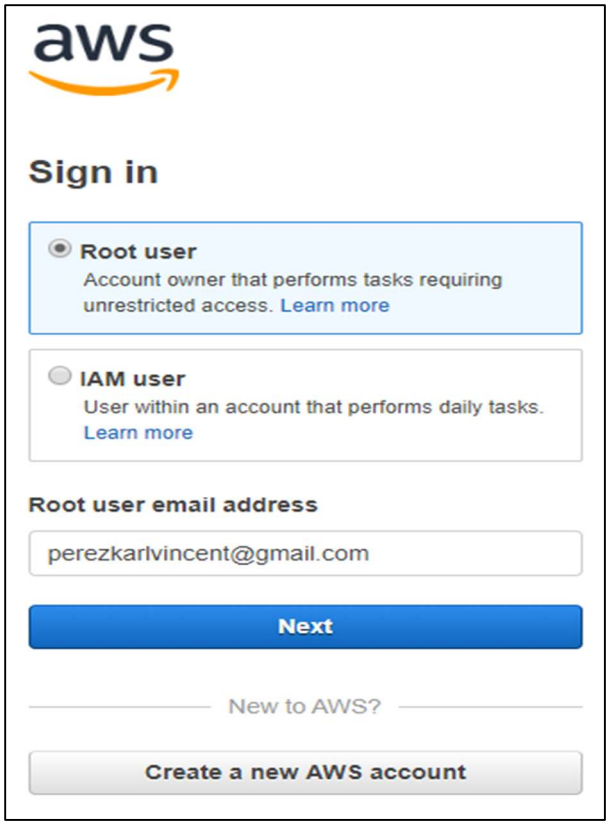
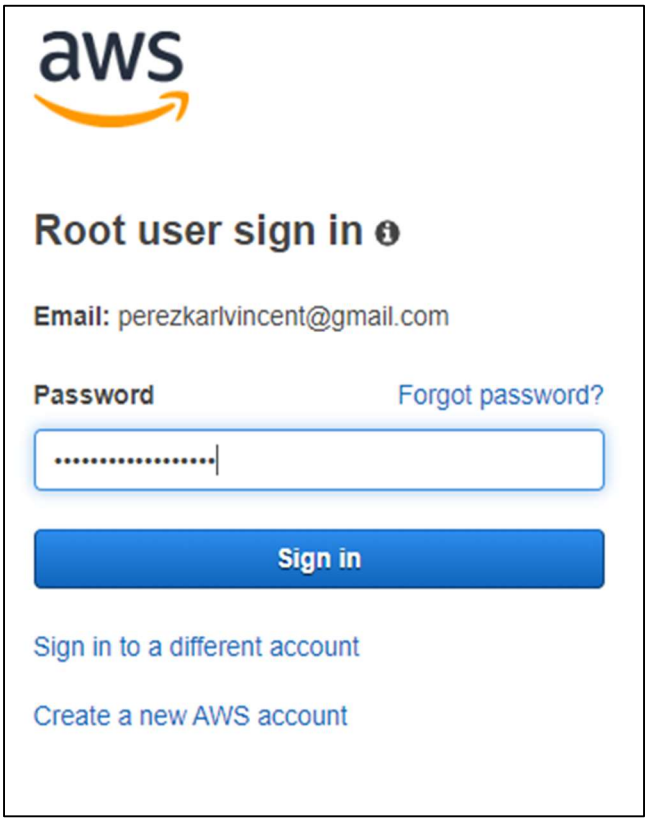
Setting Up AWS Lambda Function, DynamoDB Tables and API Gateway

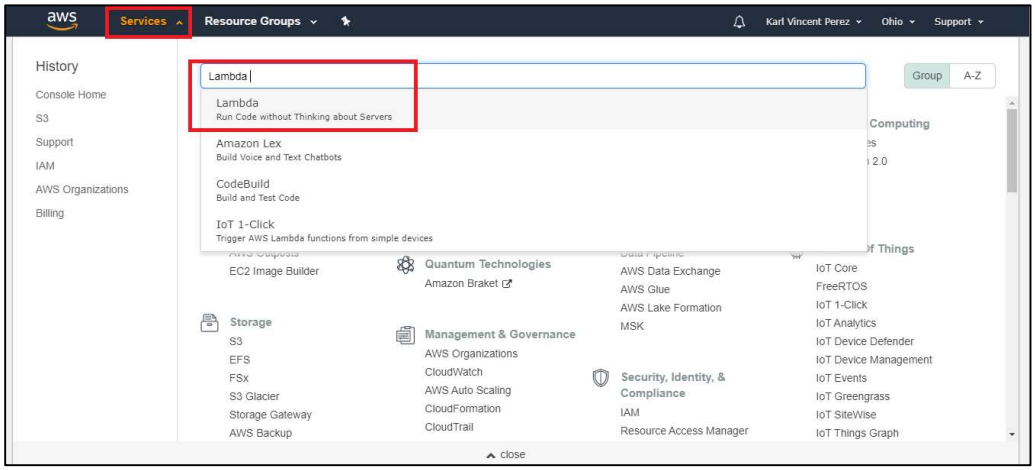
Note: Make sure you already created your AWS Account. If you haven't account yet, Go to https://aws.amazon.com/free/

Step 1: Sign In on Amazon Web Services (https://aws.amazon.com/) as **Root** account.

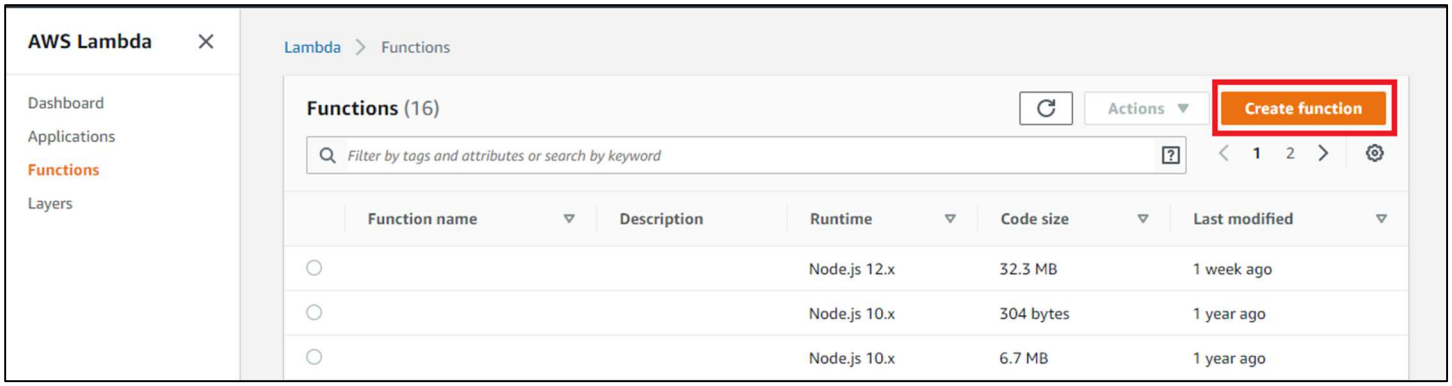




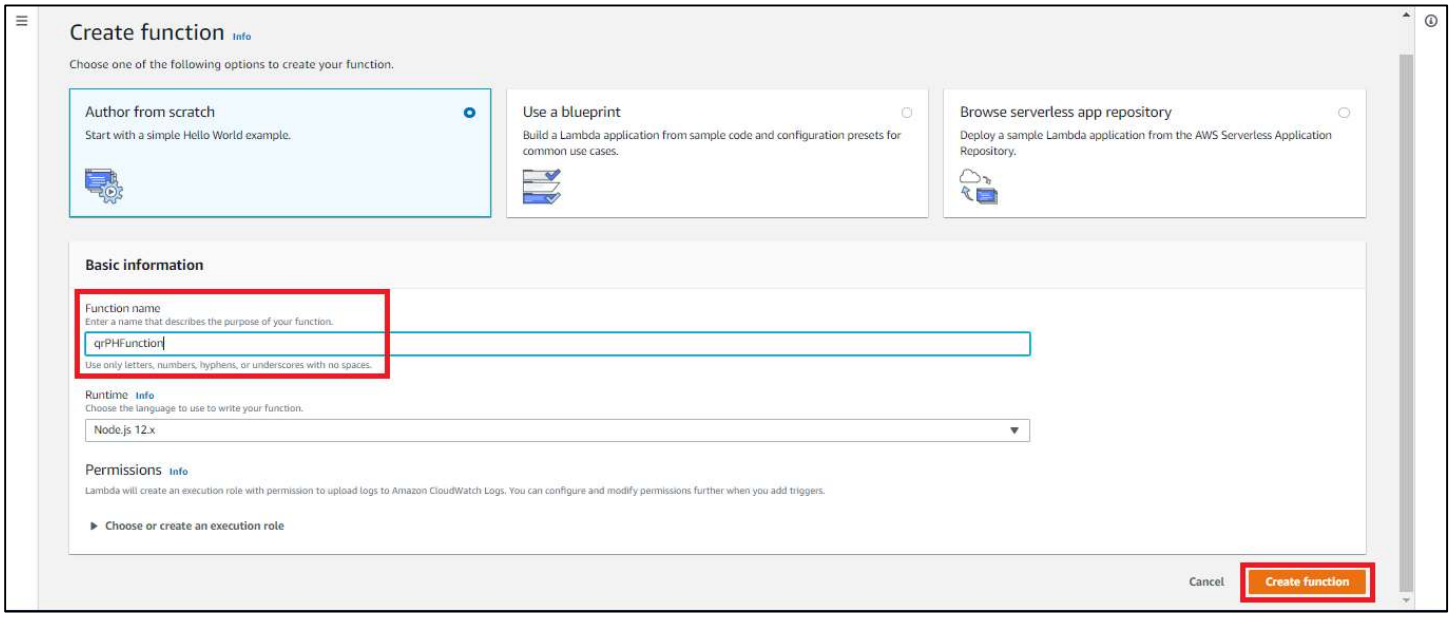
Step 2: Click Services and Search for Lambda.



Step 3: On AWS Lambda Page, Click **Create Function** button.

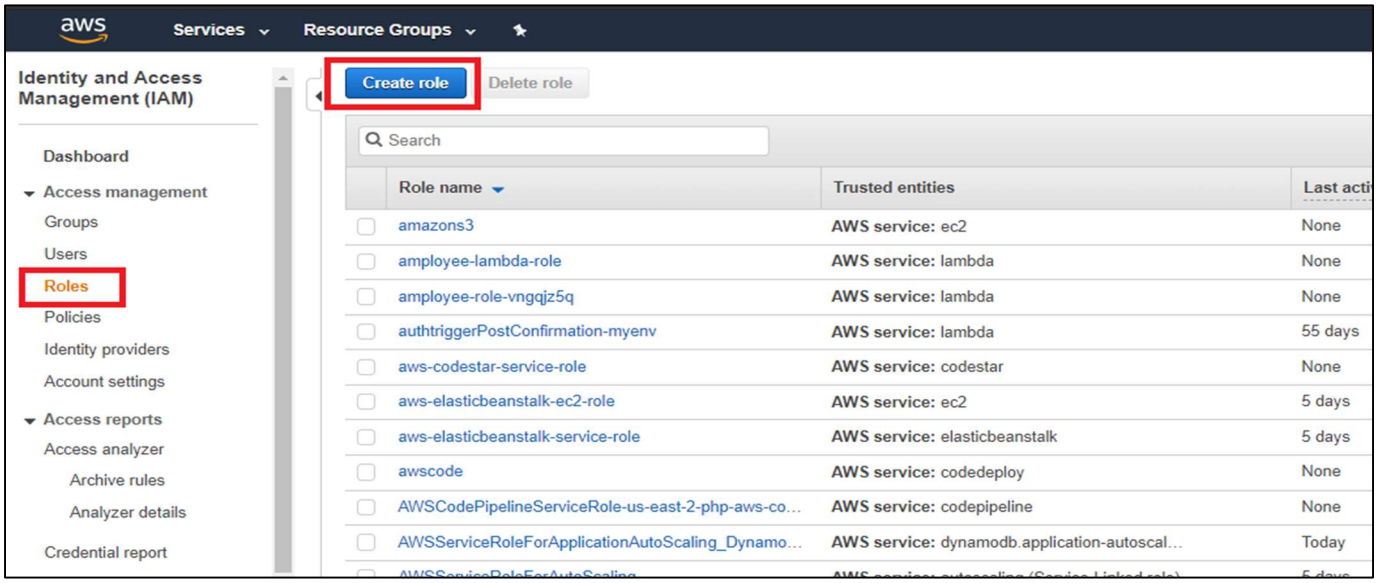


Step 4: On Create Function Page, under **Basic Information** Section, enter your function name (ex.qrPHFunction). Leave the **Permissions** field as it is, we will get back later to it to add permission our Lambda to access DynamoDB. then Click **Create Function** button.

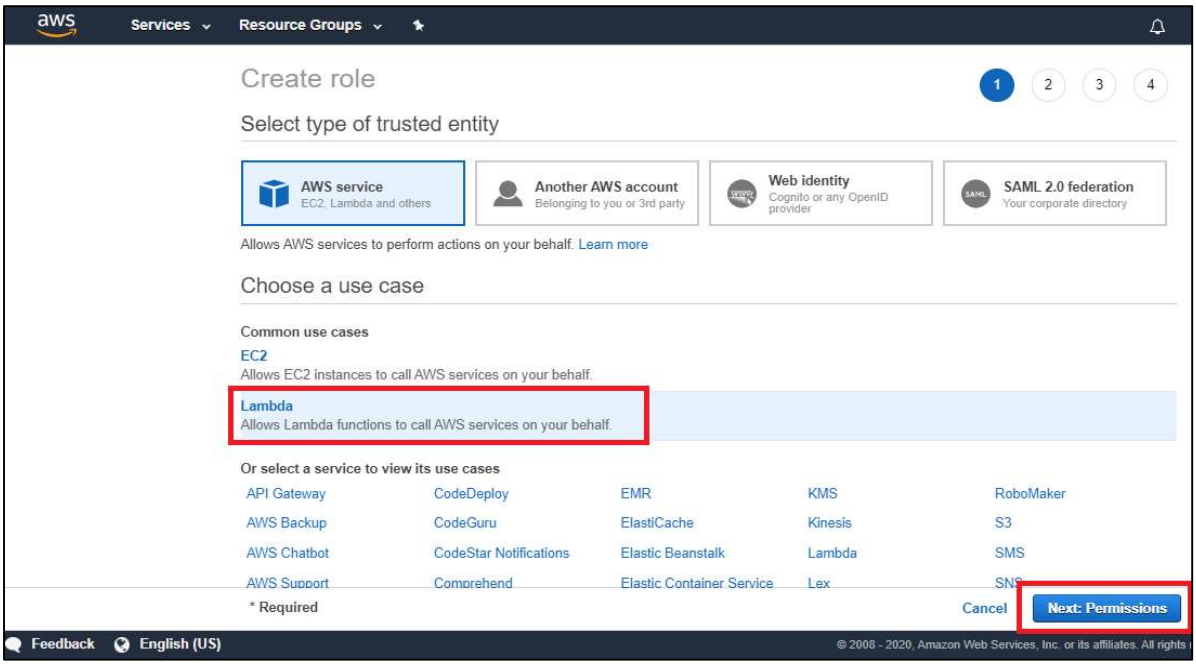


Now, when our function is created, let's switch for a moment to **IAM**(Identity and Access Management) and create a role for it. We will need it, as I already mentioned, to grant access to **DynamoDB** – the database we will use.

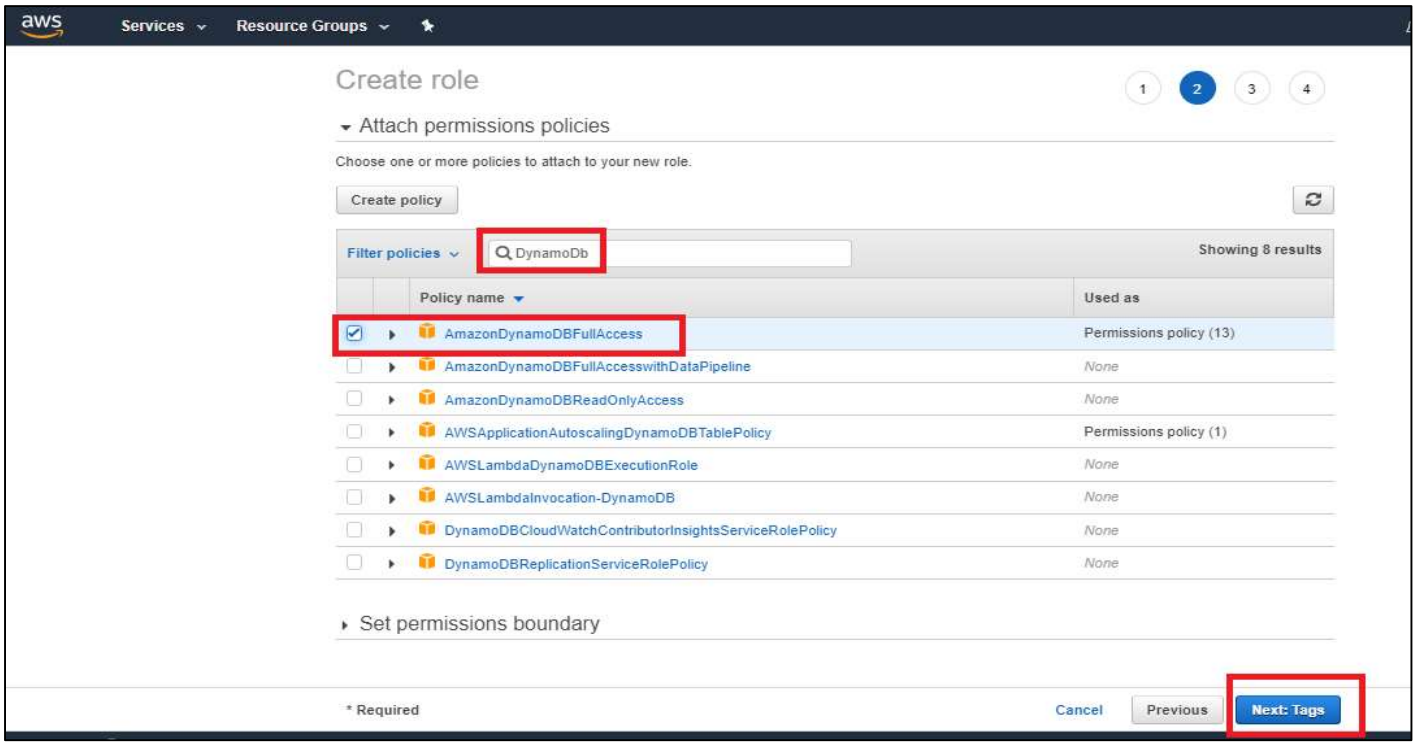
Step 5: Go to https://console.aws.amazon.com/iam. On the IAM Dashboard, Select **Roles** then click **Create Role** button



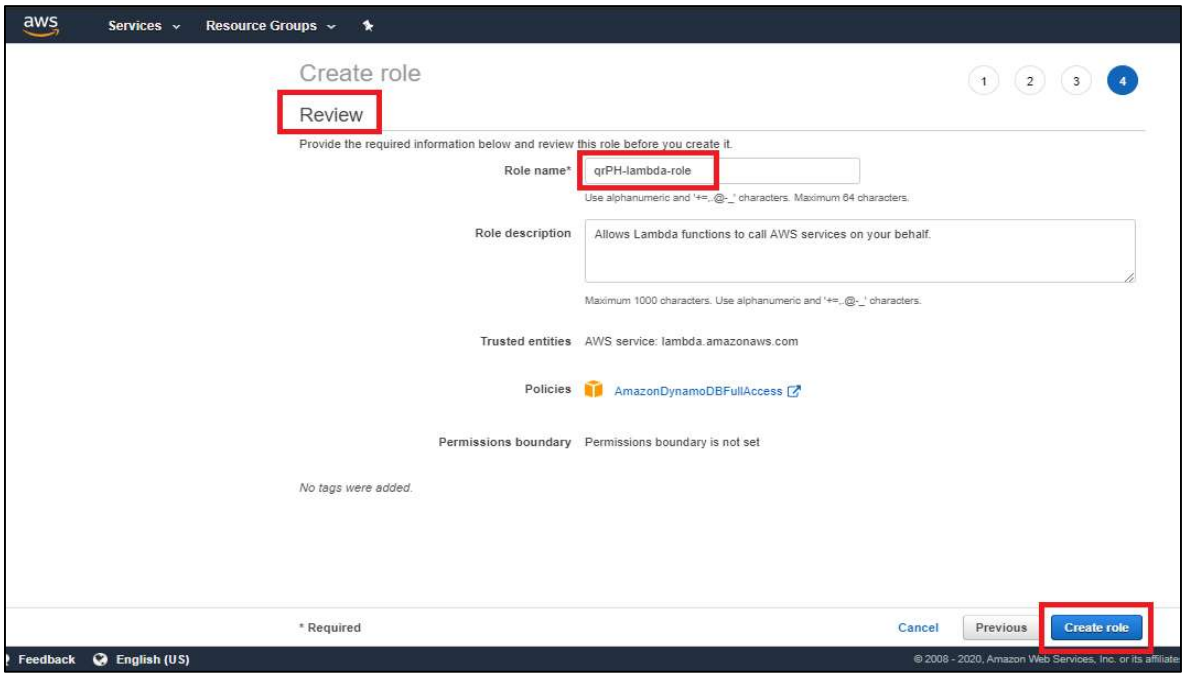
Step 6: Choose the service that will use this role – in our case **Lambda** and click **Next: Permissions.**



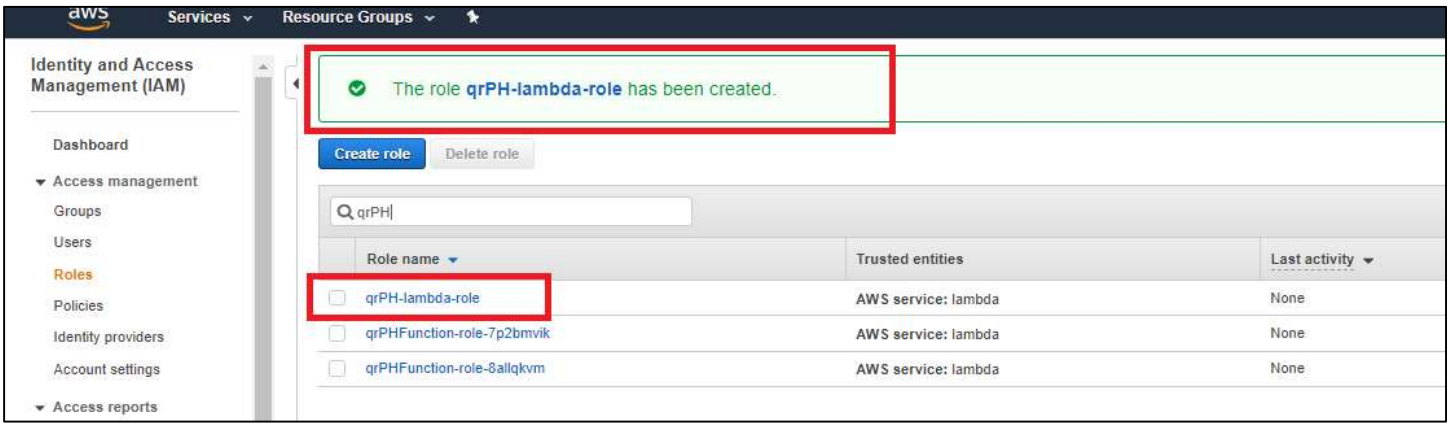
Step 7: Here, we can create our own custom policy or use the already available ones. The policies are basically rules in JSON format that tells the role what permissions should be given to the service attached to it. For our example here, we will use the already available **AmazonDynamoDBFullAccess** policy.



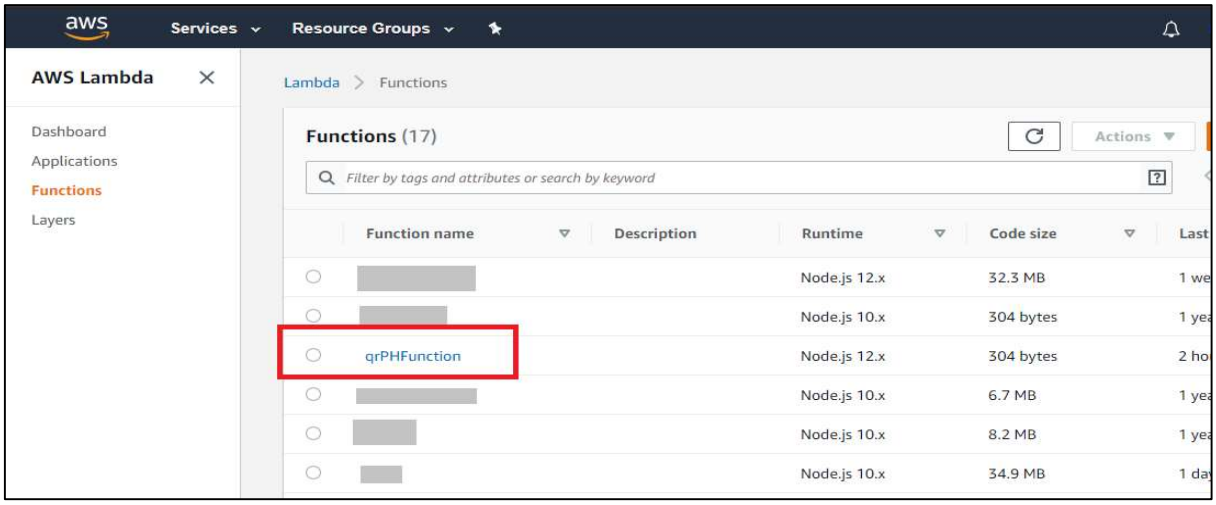
Step 8: Click **Next** and again **Next** and you should view the **Review** part. On the **Review** Section, fill the desired role name, something like '*qrPH-lambda-role*' and click **Create Role** button.



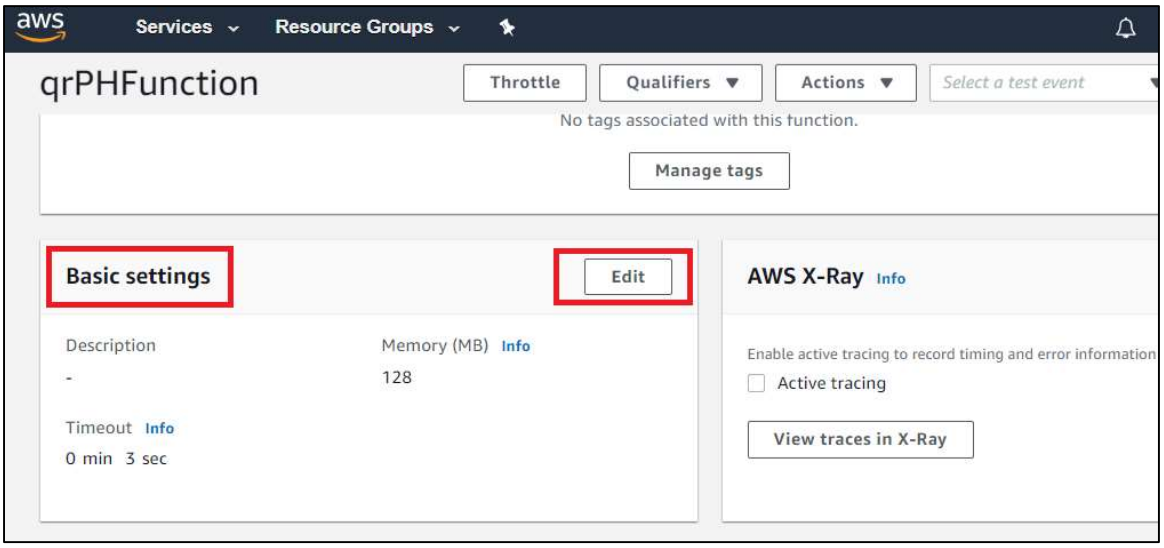
The role should be created and available in the list of roles available in **IAM**.



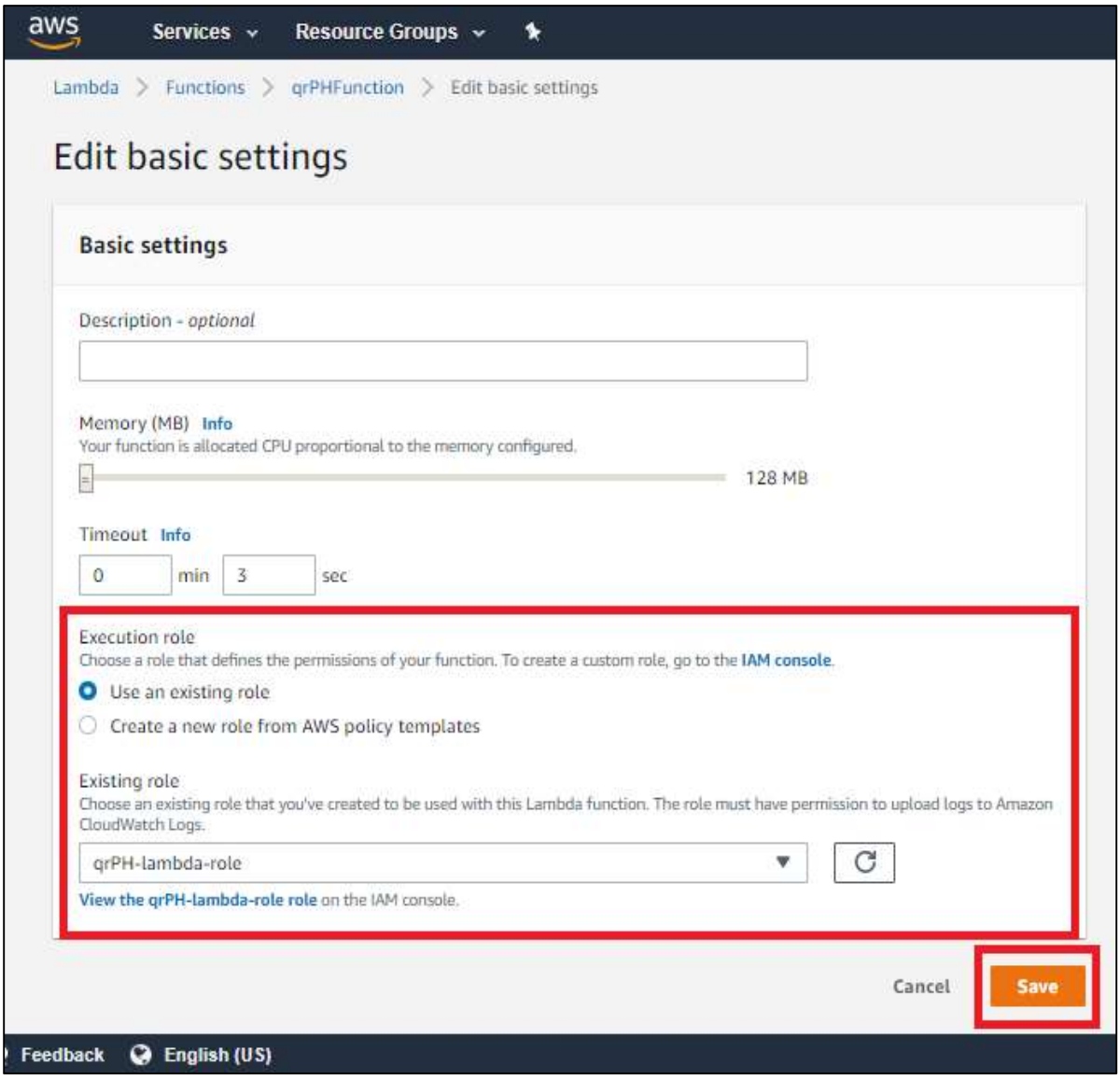
Step 9: Now, we can get back to our lambda and assign this role to it.
Goto https://console.aws.amazon.com/lambda and choose your function.



Step 10: You will be redirected to the specific Lambda page. Scroll down to **Basic Settings** and Click **Edit** button.



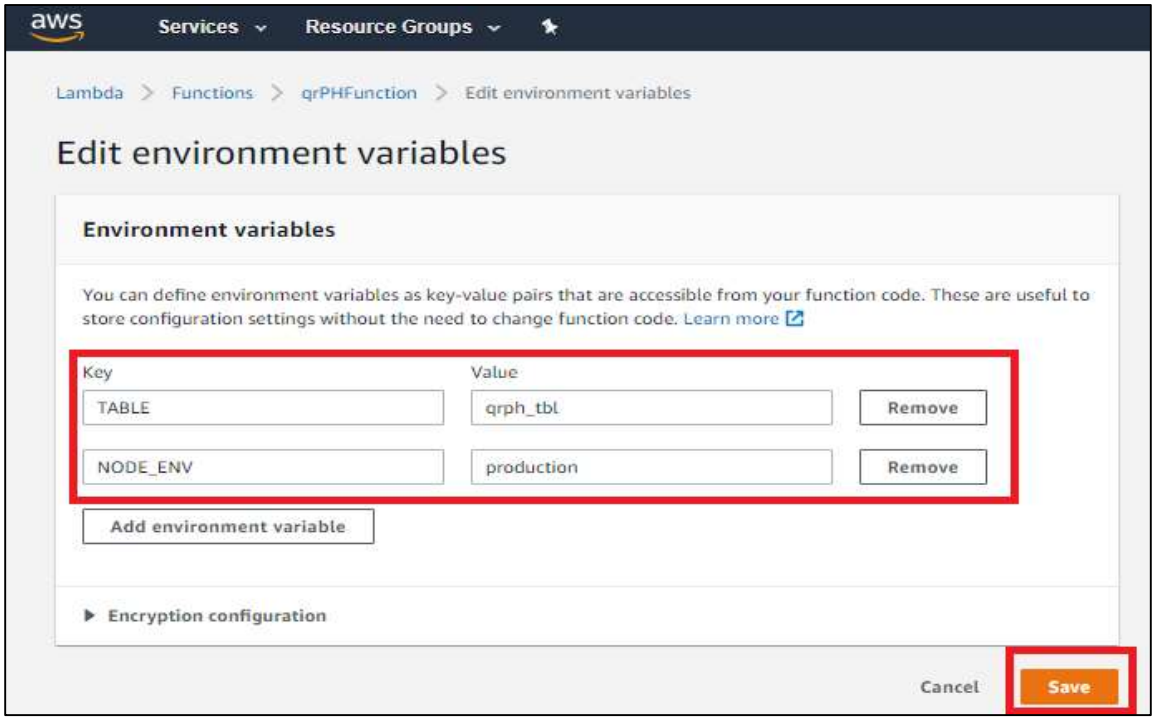
Step 11: On **Edit Basic Settings** Page, Scroll down to **Execution Role Section** then Select **Use an Existing Role** and then Choose the existing role that we've created earlier (ex. qrPH-lambda-role). Then click **Save** button



Step 12: After saving the **Basic Settings,** Go to the **Environment Variables** section then click **Edit** button. On **Edit Environment Variable** Page, Click **Add Environment Variable** button and include two variables which we will use later when writing our Node.JS logic then click **Save** button.

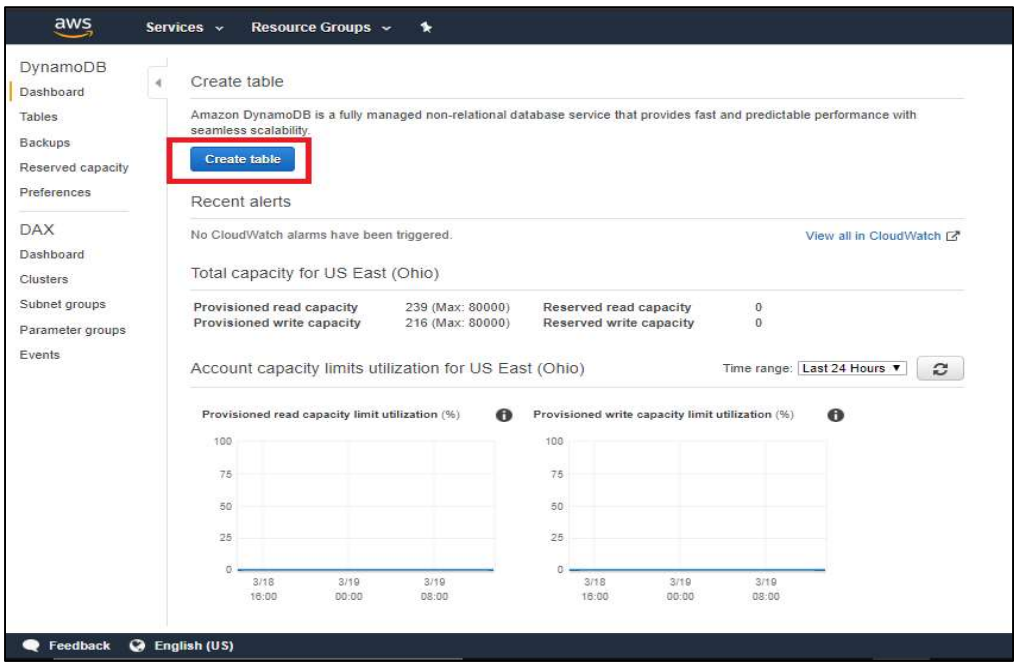
TABLE: qrph_tbl // the name of our future database table (will create it soon)

NODE_ENV: production // the environment, let's call it 'production'. This will help us to identify if it is local or serverless instance of the server

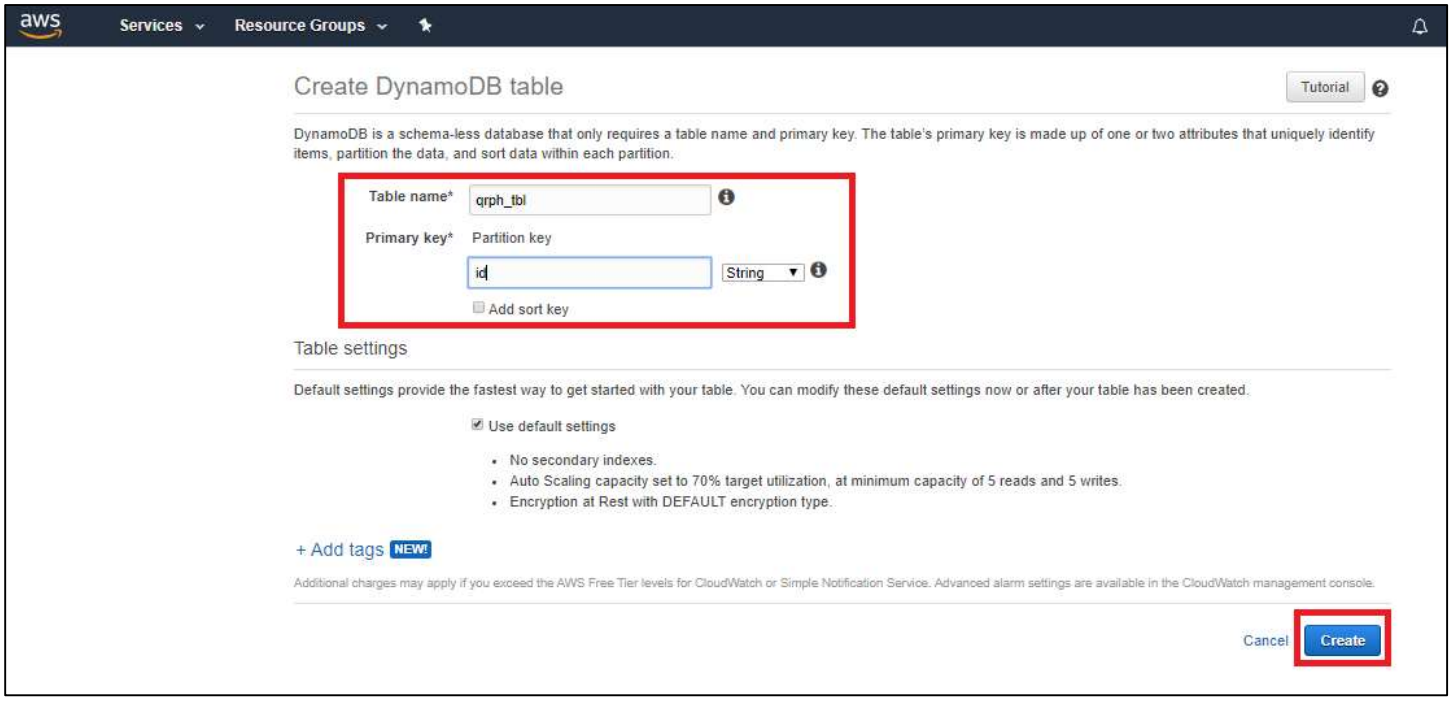


Configure DynamoDb

Step 13: Go to https://console.aws.amazon.com/dynamodb. On **DynamoDB** Dashboard, click **Create Table** button.



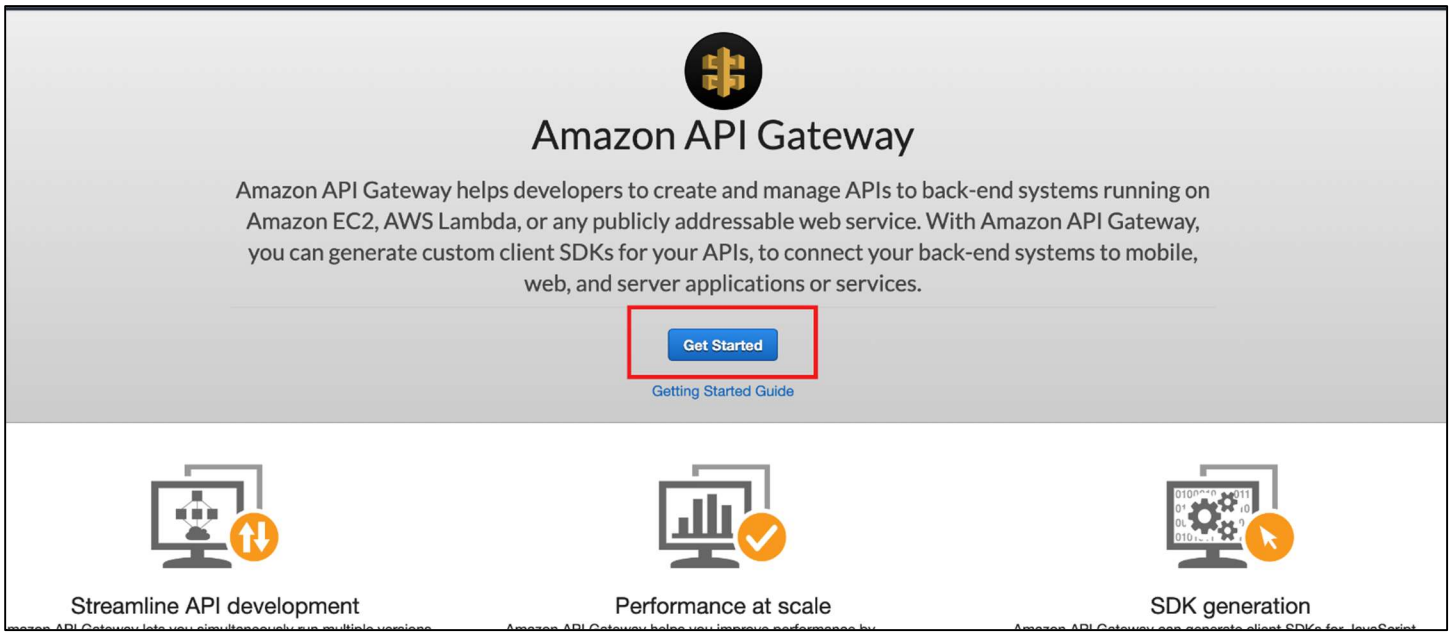
Step 14: On **Create DynamoDB Table** Page, fill the table name to **qrph_tbl** and Primary key **id** with type string. Then click **Create** button



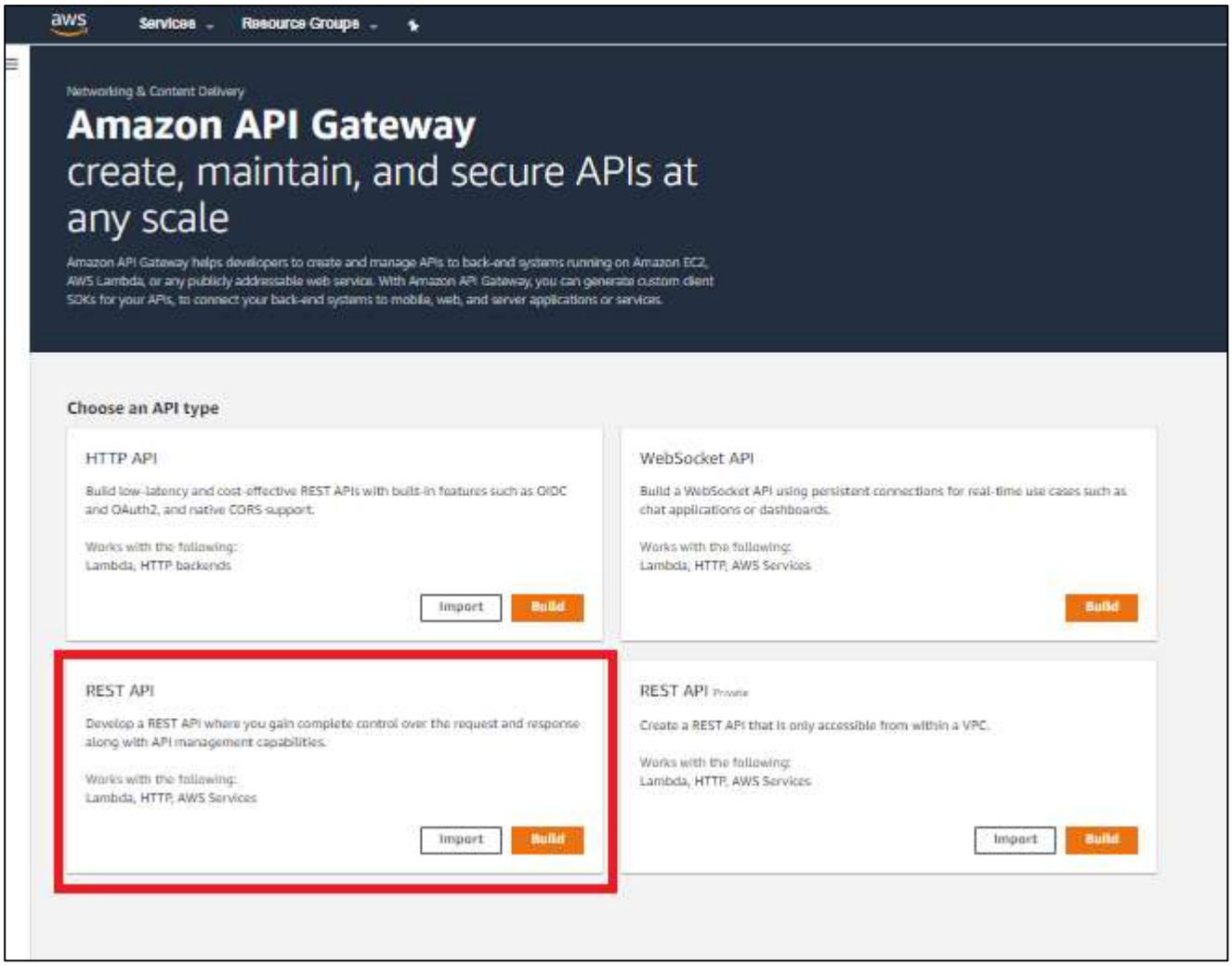
When the table is created you should be redirected to the table management route. This means that your table creation was successfully. We are done with the **DynamoDB** configuration.

Now it's time to create **API Gateway** and connect it to the **Lambda** we have created earlier.

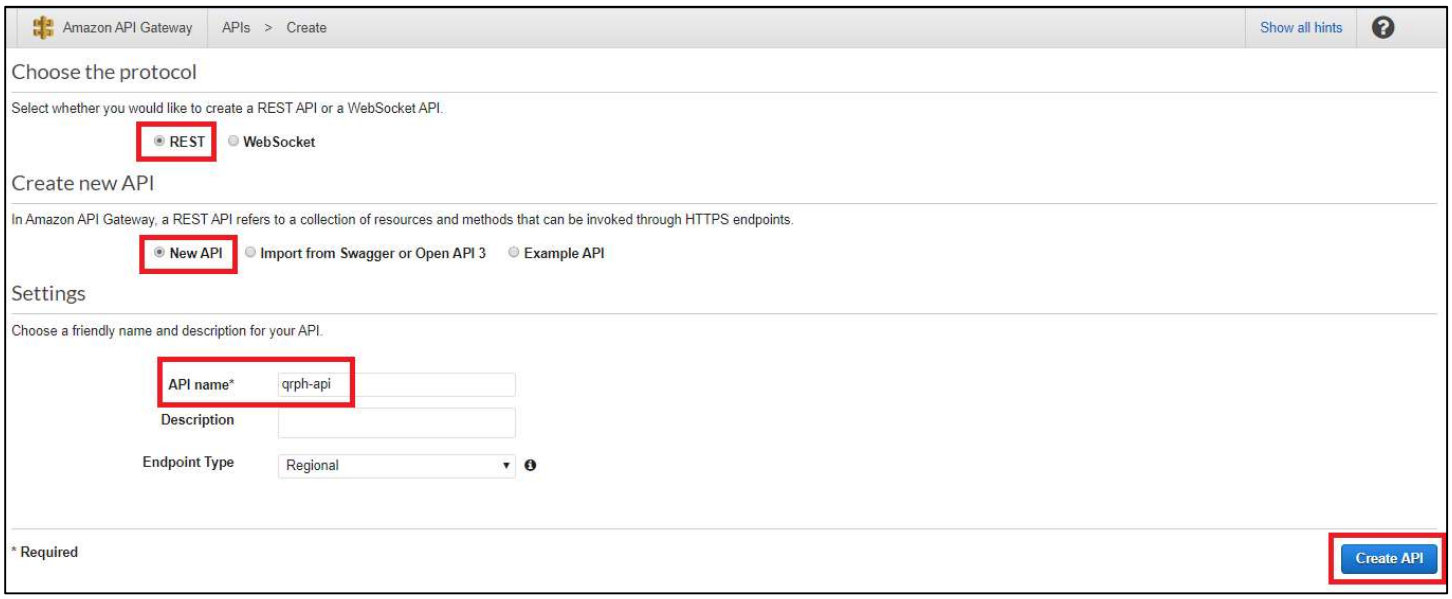
Step 15: Go to https://console.aws.amazon.com/apigateway and click **Get Started**.



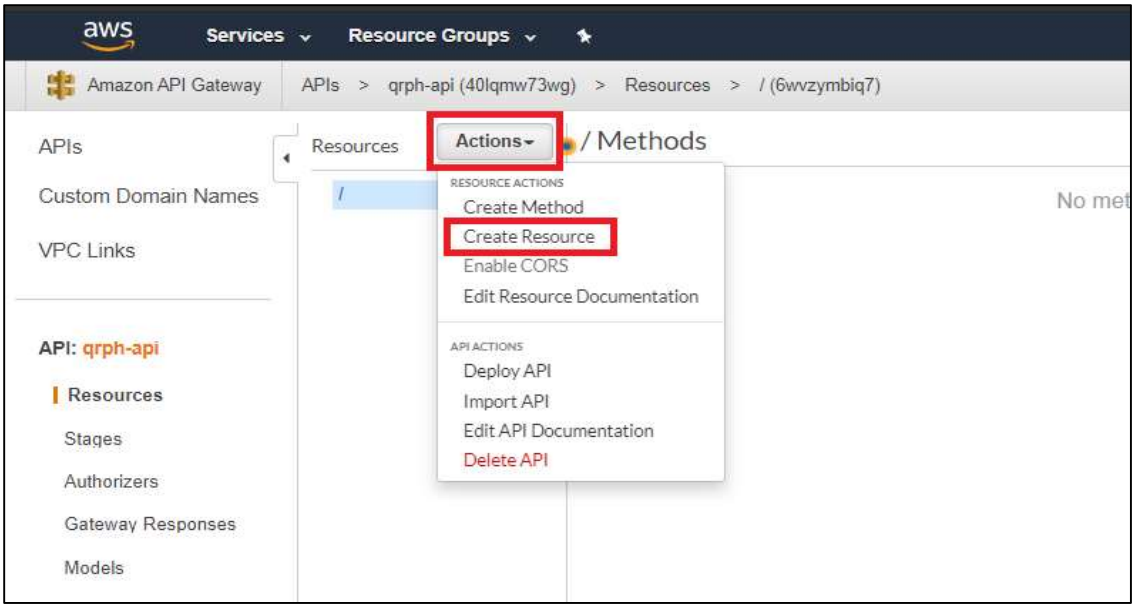
Step 16: Choose **Rest API** then click **Build** button.



Step 17: On **Choose the protocol** Section, select **REST API, New API,** then Enter the **API Name** (ex. qrph-api) and Click **Create API** button.



Step 18: Click **Create API** and soon you will be redirected to the newly created API. Go to the **Actions Tab** and choose **Create Resource**.



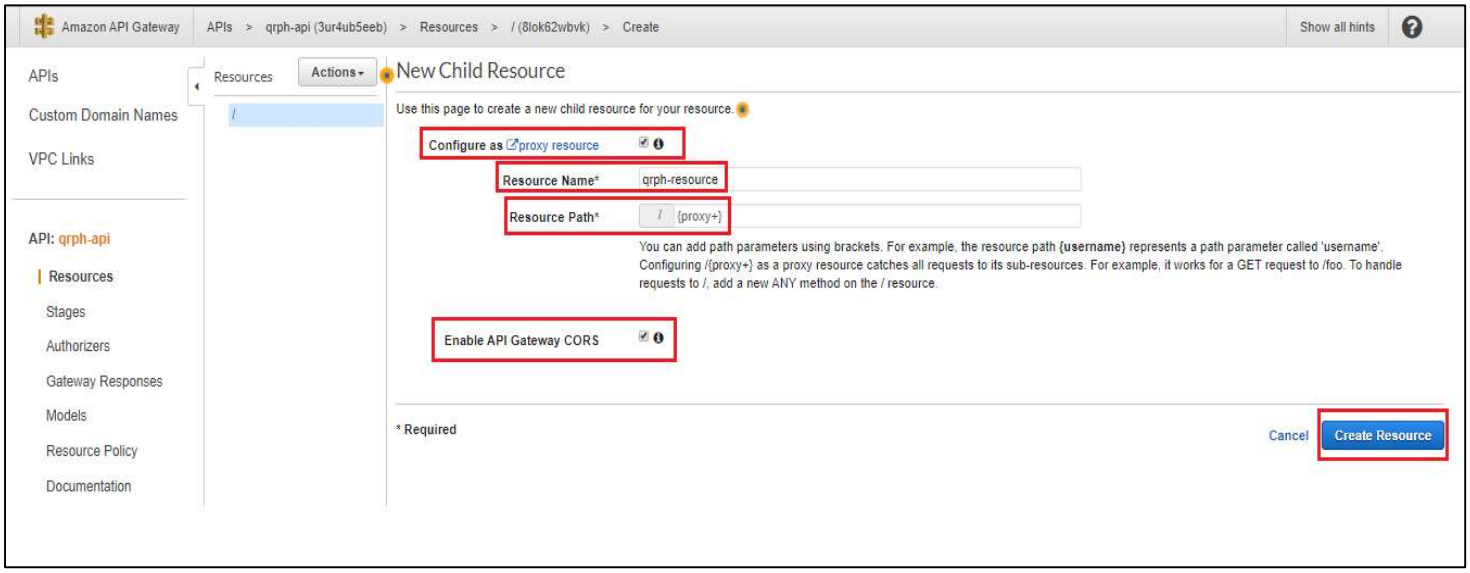
Step 19: On **New Child Resource** Page,

Configure as proxy resource have to be checked – this way we will handle the routes in our Lambda function and there will be no need to manually add every endpoint in the gateway every time we create one.

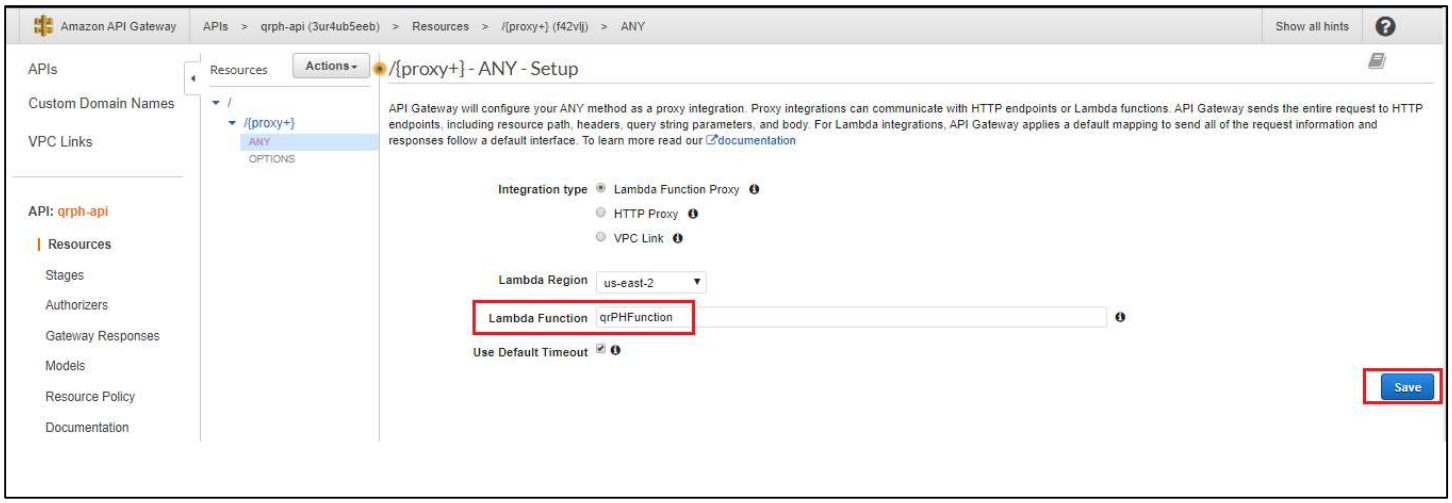
Resource name – 'empoyee-api' and **Resource path** – {proxy+} (you can find more information below the field what {proxy+} means).

Enable API Gateway CORS is not required, but I suggest you to also check it. This way you can configure later the origins you want to have access to your resource, methods and etc.

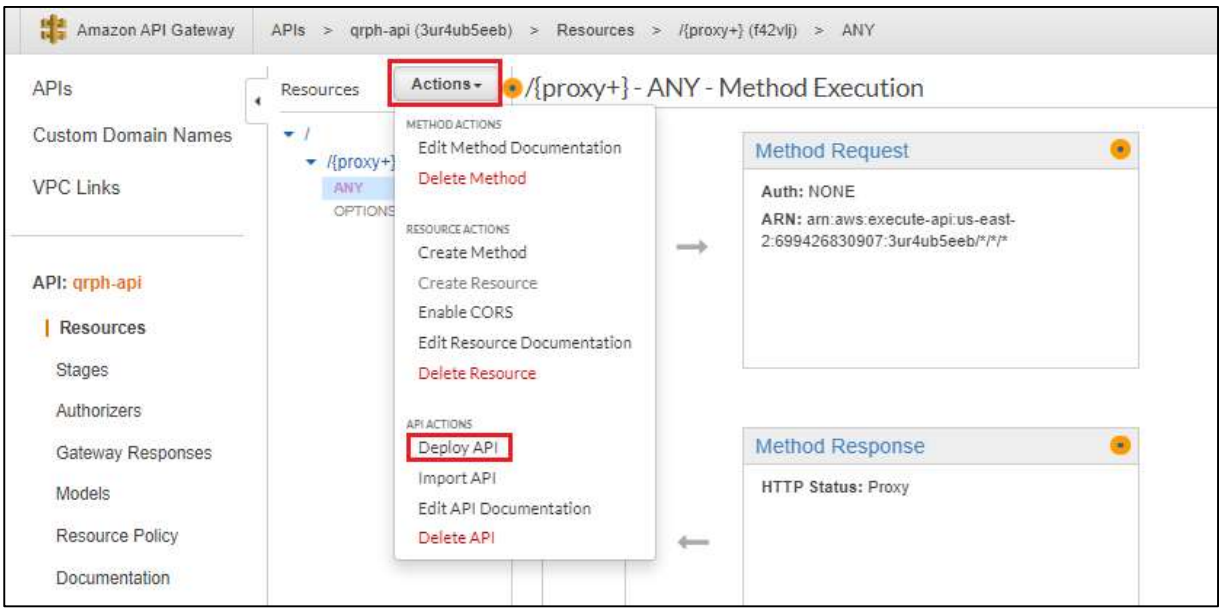
Then click **Create Resource** button



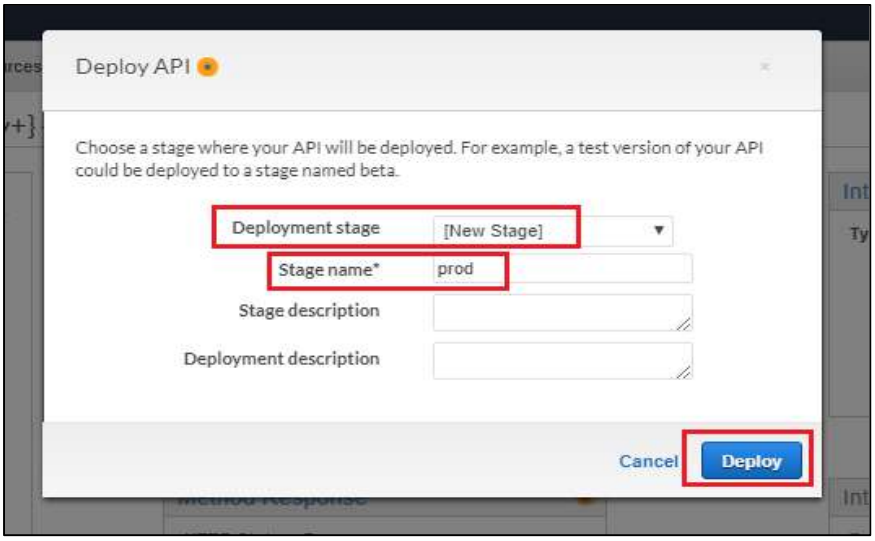
Then specify the **Lambda function** you want to connect to your newly created API Gateway resource.



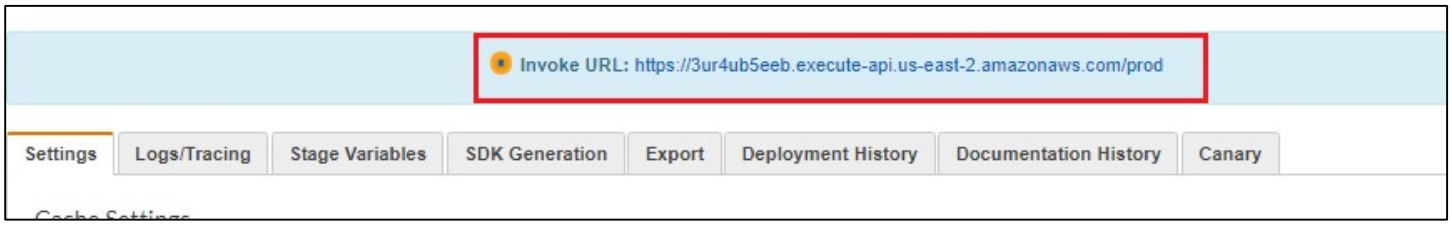
Step 20: Deploy the **Resource**. Click the **Action** button then Select **Deploy API.**



Then, Choose the **Deployment stage -> [New Stage]** and **Stage Name -> 'prod'.** The others are fields are optional.



Click Deploy and soon you will see your **API endpoint**, like the one below:



This is your **base API u**rl, which we will use from now on to access it.

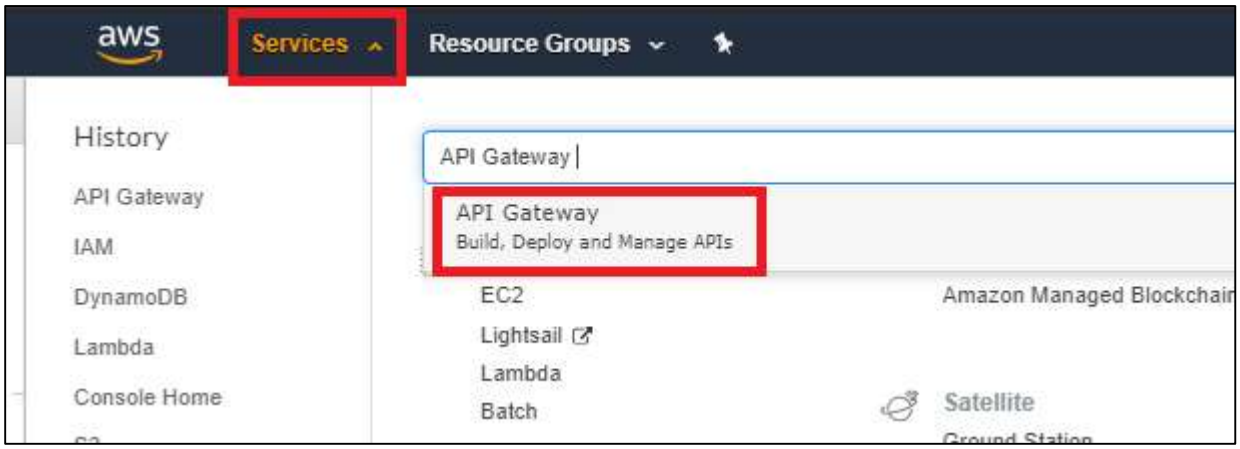
How To Test IT:

Go to your Browser then paste the generated **API Endpoint**, then add the API Name on the end of the URL. (ex. https://3ur4ub5eeb.execute-api.us-east-2.amazonaws.com/prod/test)

Setting the API Key Security

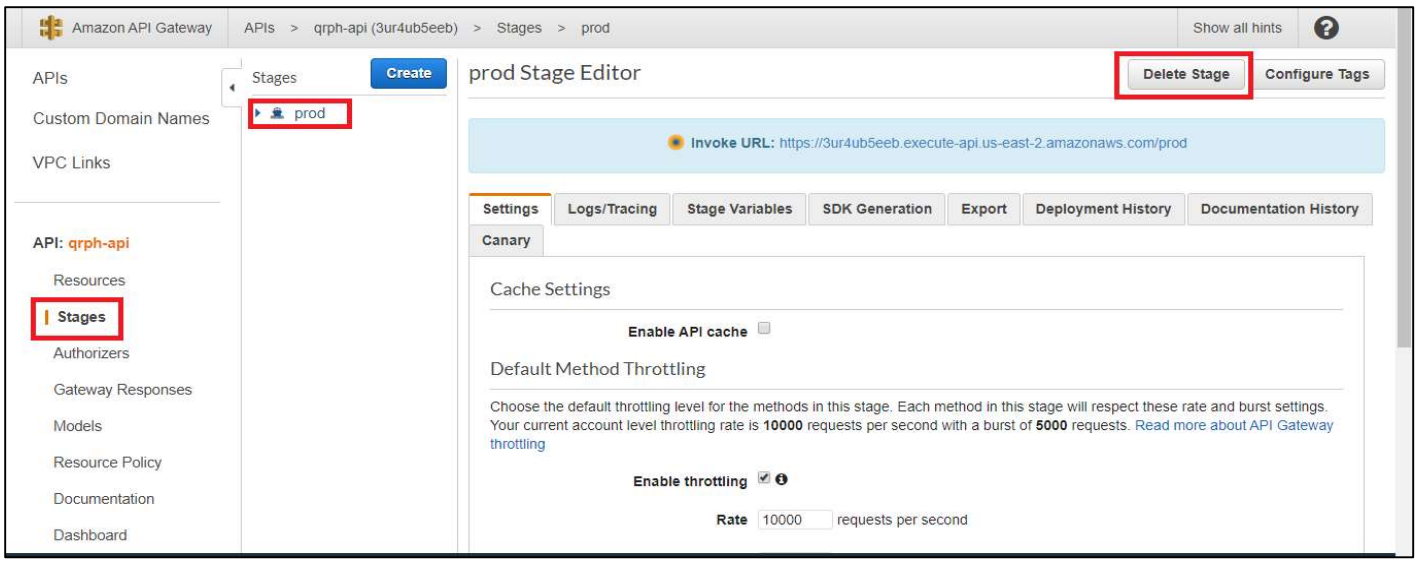
Step 1: Go to https://aws.amazon.com/ and login on your account as **Root** account.

Step 2: Click **Services** tab then enter **API Gateway** to search field.

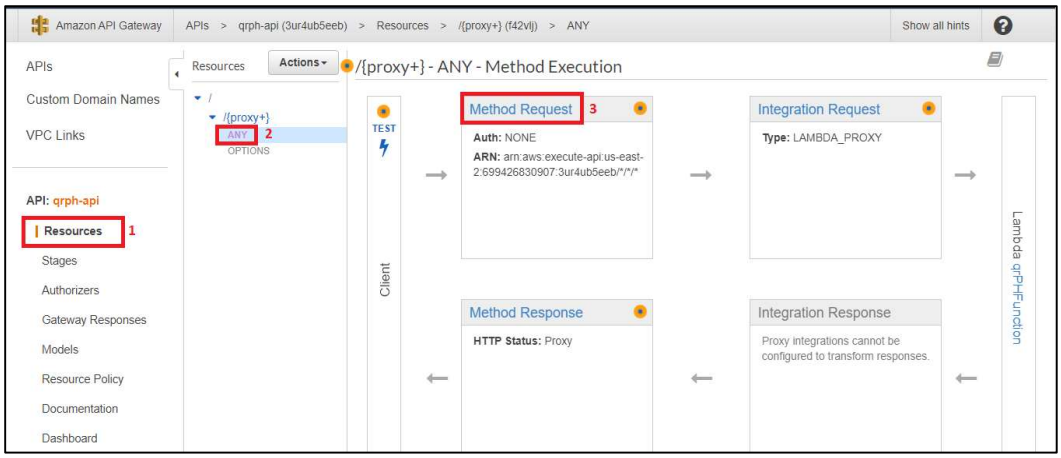


Step 3: Select your api (ex.qrph-api).Then select Stages

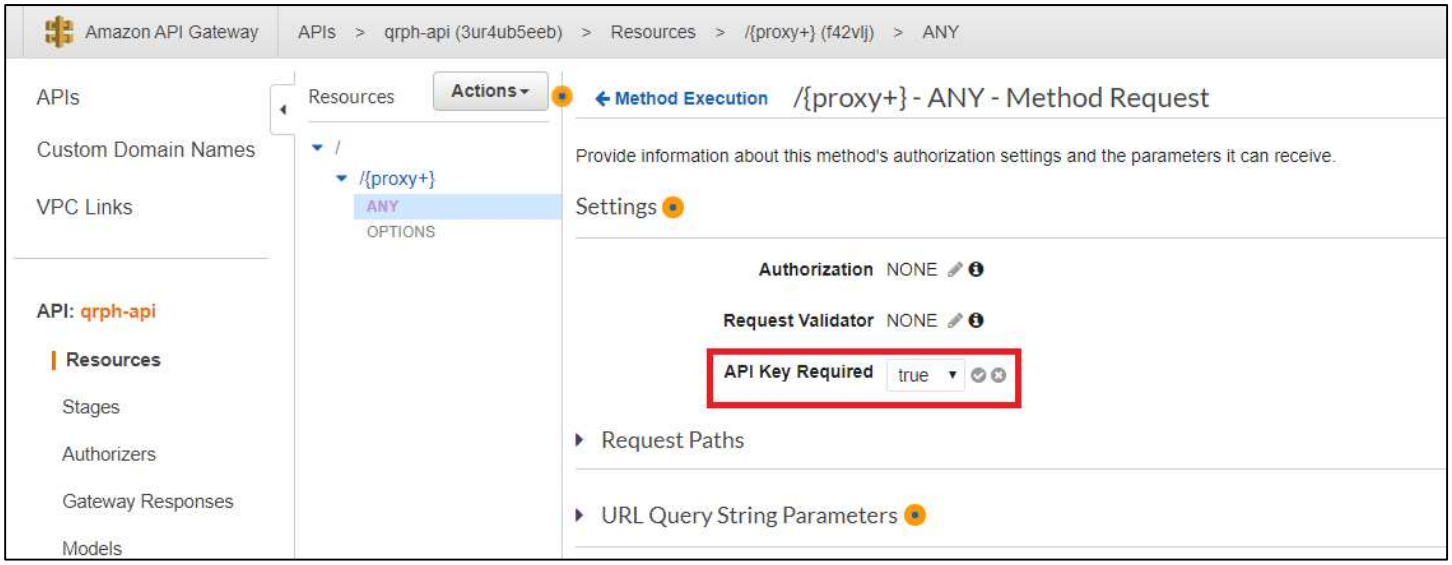
Note: If there is an existing Stage on Stages, Select the existing stage and Delete the existing Stage.



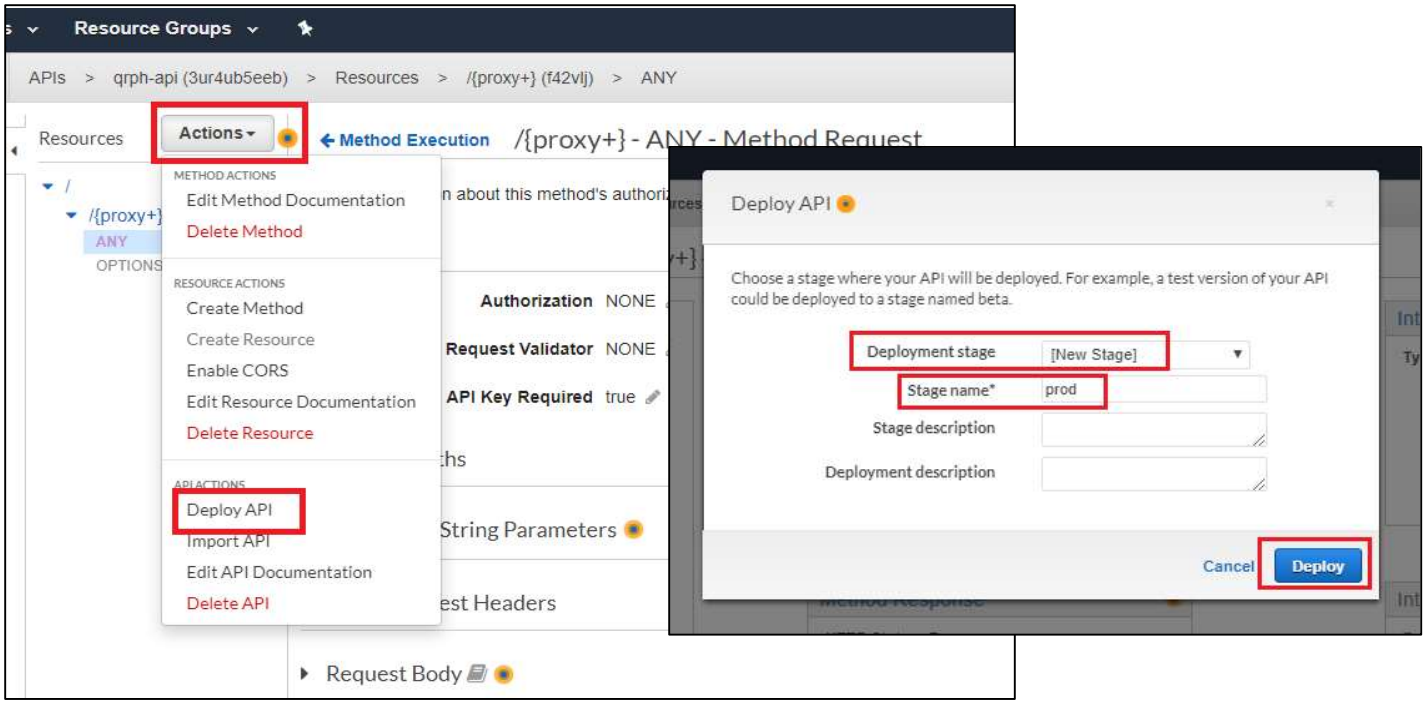
Step 4: Select Resources > Any > Method Request



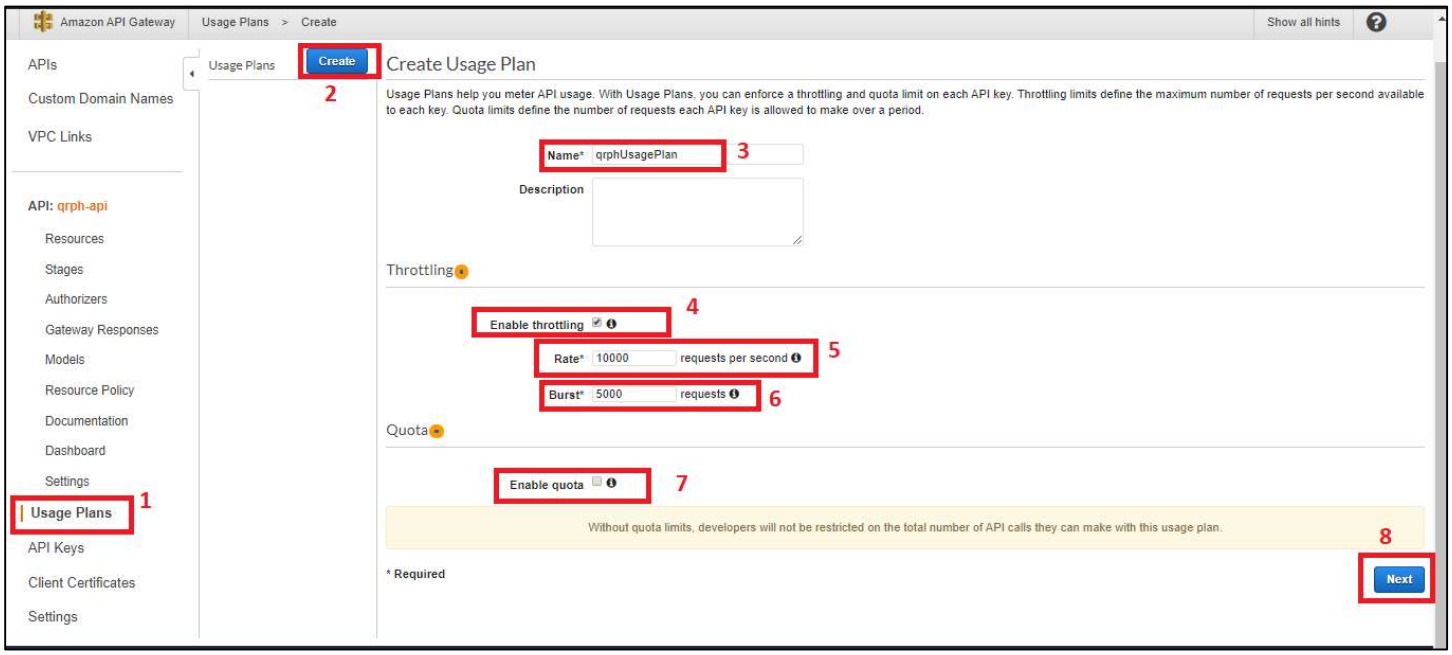
Step 5: Change the **API Key Required** to true. Then click the **Check** icon.



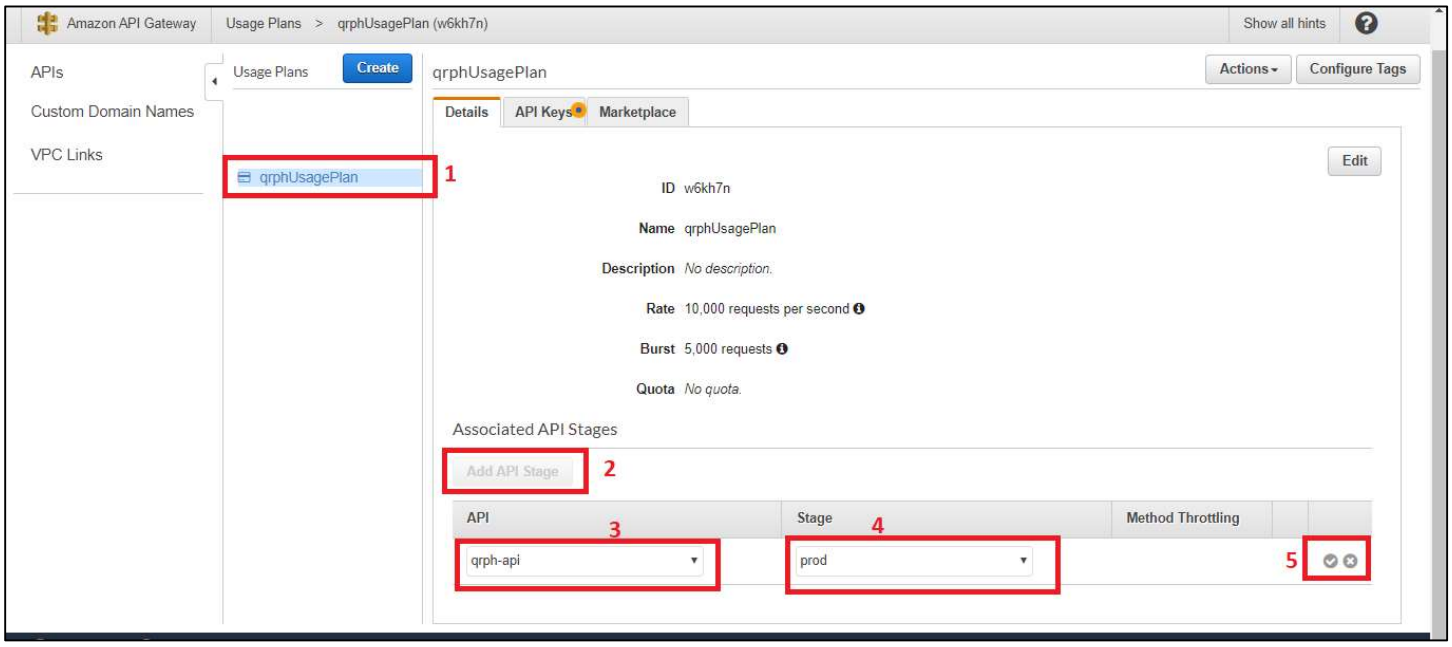
Step 6: Click **Action** button then **Select Deploy API**. Select '*[New Stage]*' for Deployment Stage and Stage Name as '*prod*' then Click **Deploy** button.



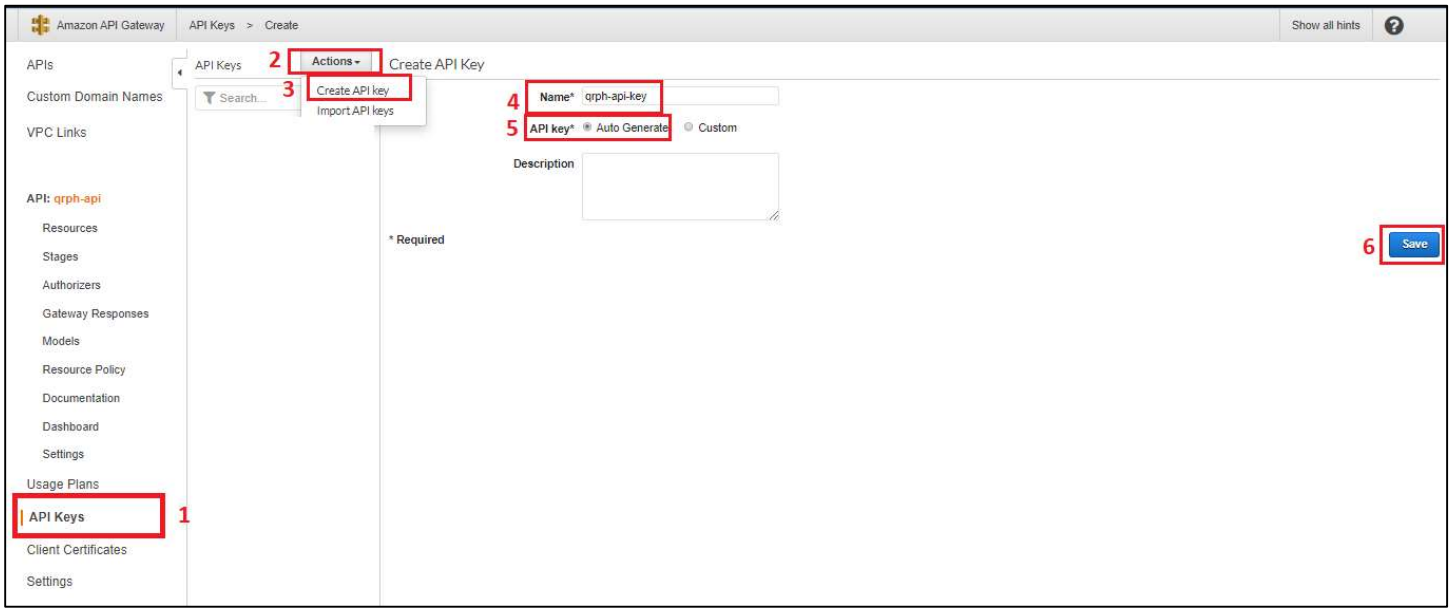
Step 7: Go to **Usage Plans** then click **Create** button. Enter a **Usage Plan Name** then under **Throttling**, set the **Rate** to *10000* and **Burst** to *5000* (default rate and burst) then uncheck Quota then Click **Next** > **Skip** > **Done.**



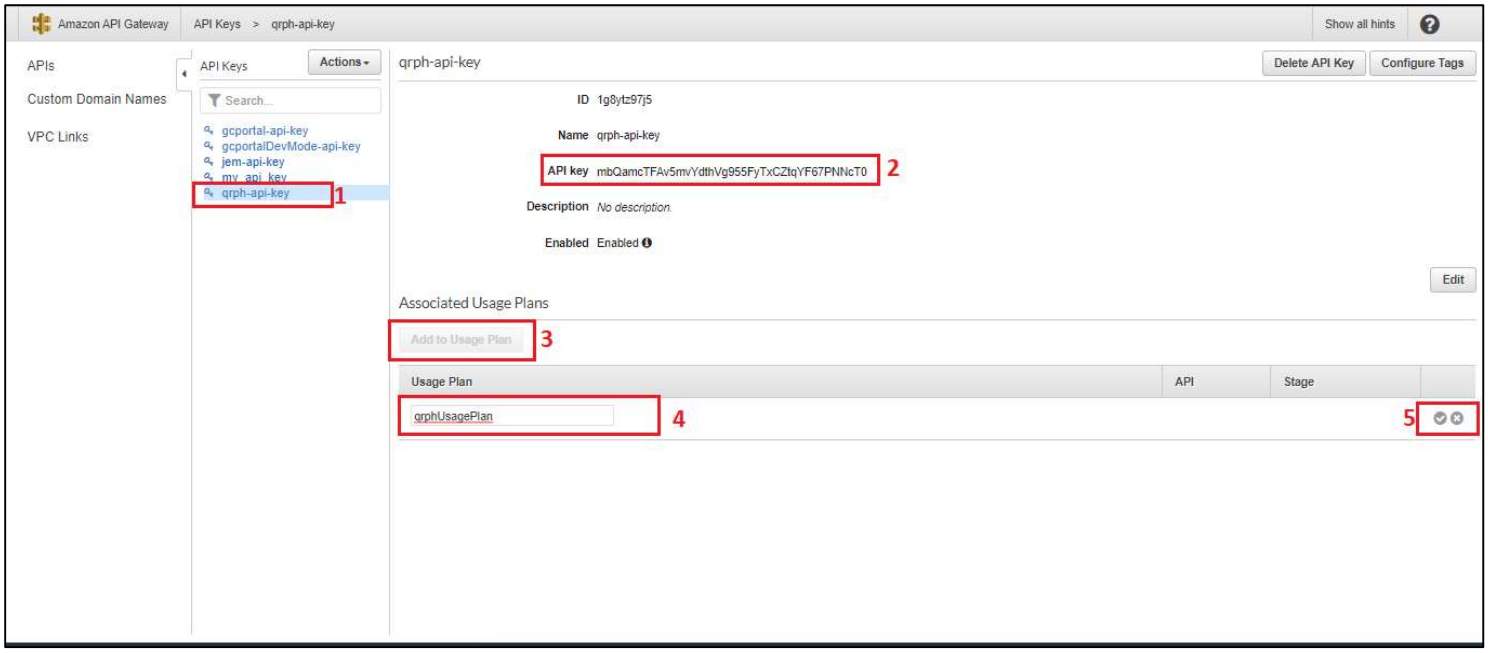
Step 8: Click your newly created **API Plan** then under **Associated API Stages**, Click **Add API Stage** then Select your **API** (ex.qrph-api) and **Stage** (ex.prod) then Click **Add/Check** button



Step 9: Select **API Keys** > **Actions** > **Create API Key.** Then Enter your own **API key** then **Select Auto Generate** then Click **Save** button.



Step 10: Select your **created api key**. On API key, click **Show** to display the auto generated API key (ex.aAPI Key: mbQamcTFAv5mvYdthVg955FyTxCZtqYF67PNNcT0). Then click **Add to Usage Plan** button then Enter your **Usage Plan Name** (ex. qrphUsagePlan) then Click **Add/Check** button.

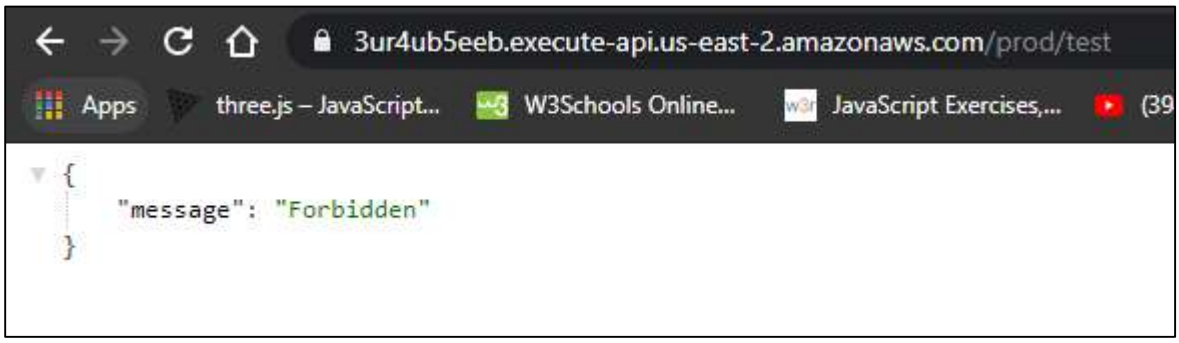


How To Test It:

Note: Install MODHeader browser extension to access the secured api.

Step 1: Go to your Browser then paste the **API Endpoint**, then add the **API Name** on the end of the URL. (ex. https://3ur4ub5eeb.execute-api.us-east-2.amazonaws.com/prod/test).

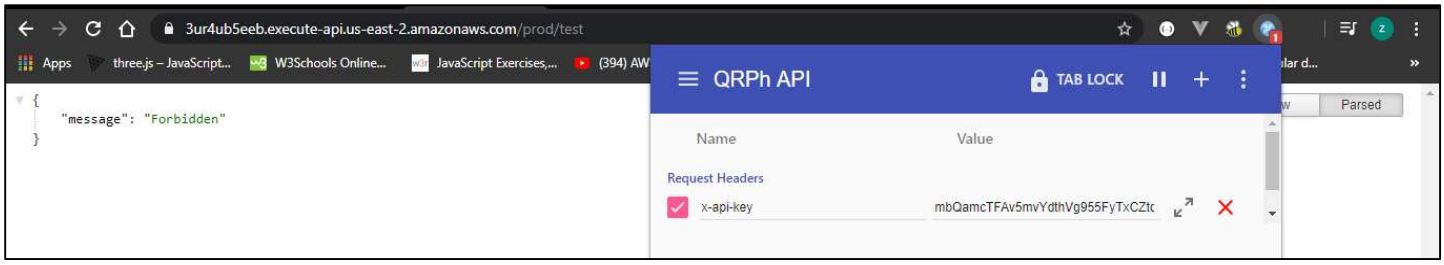
The API response must be “Forbidden”.



Step 2: Open the MODHeader extension then enter the **API KEY CREDENTIALS.**

NAME: x-api-key

VALUE: mbQamcTFAv5mvYdthVg955FyTxCZtqYF67PNNcT0



Step 3: Refresh your browser then the API response must be “Hello from Lambda!”