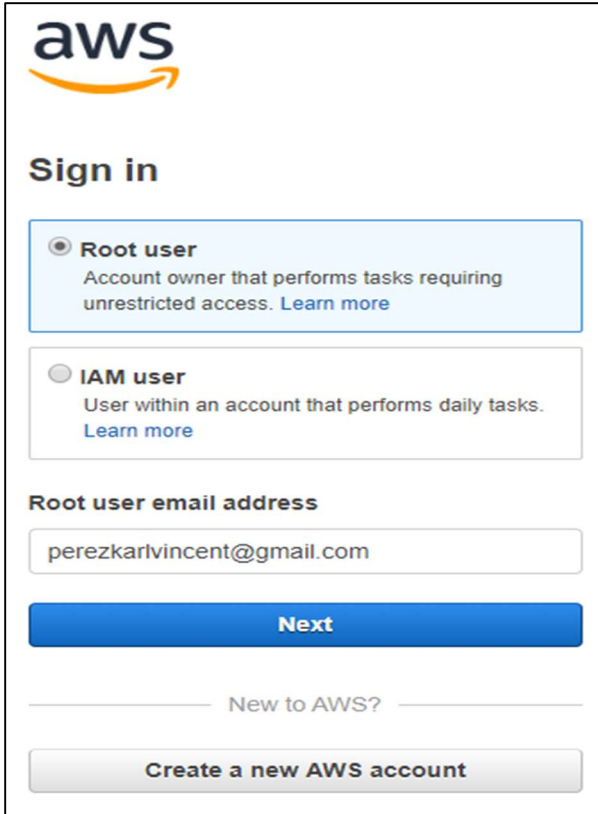


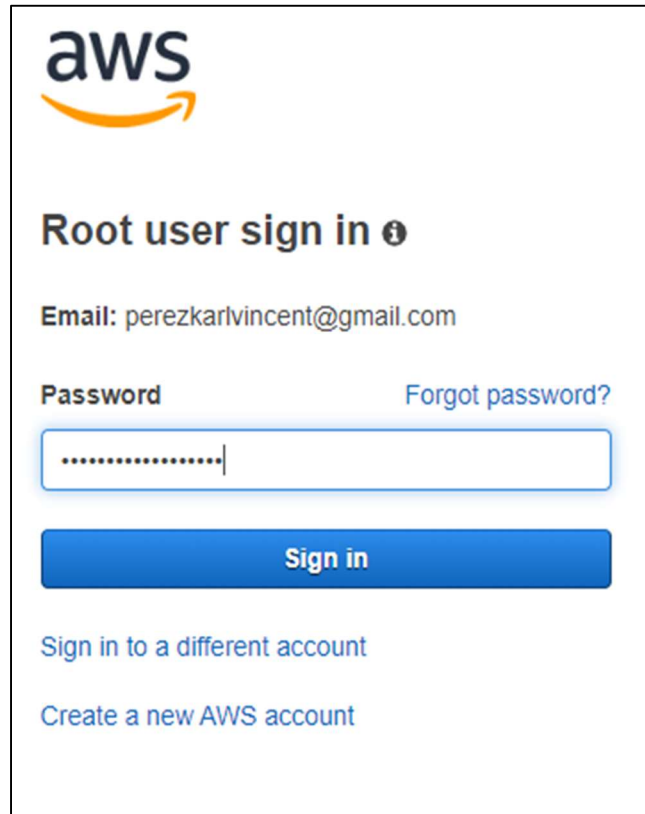
Setting Up AWS Lambda Function, DynamoDB Tables and API Gateway

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.

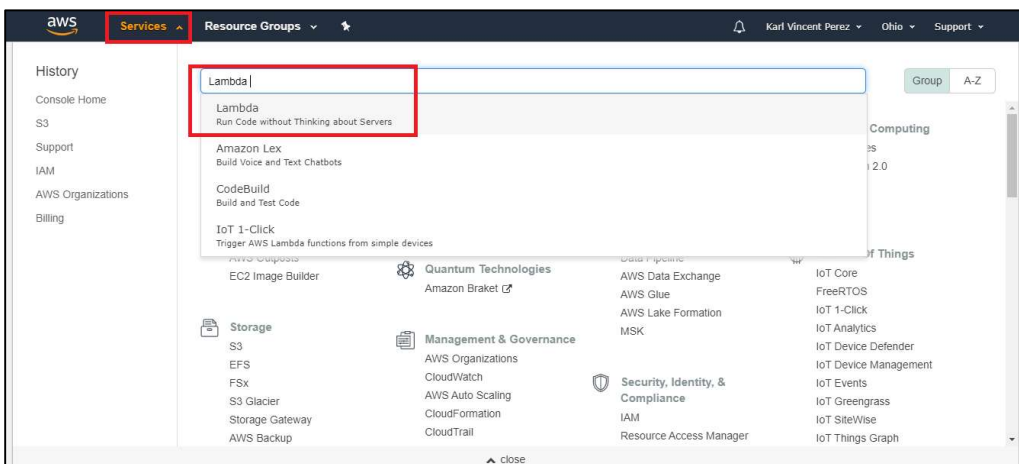


The AWS Sign in page features the AWS logo at the top. Below it is the 'Sign in' heading. There are two main options: 'Root user' (selected) and 'IAM user'. The 'Root user' option is described as 'Account owner that performs tasks requiring unrestricted access' with a 'Learn more' link. The 'IAM user' option is described as 'User within an account that performs daily tasks' with a 'Learn more' link. Below these is a field for 'Root user email address' containing 'perezkarlvincent@gmail.com'. A blue 'Next' button is positioned below the email field. At the bottom, there is a 'New to AWS?' link and a 'Create a new AWS account' button.

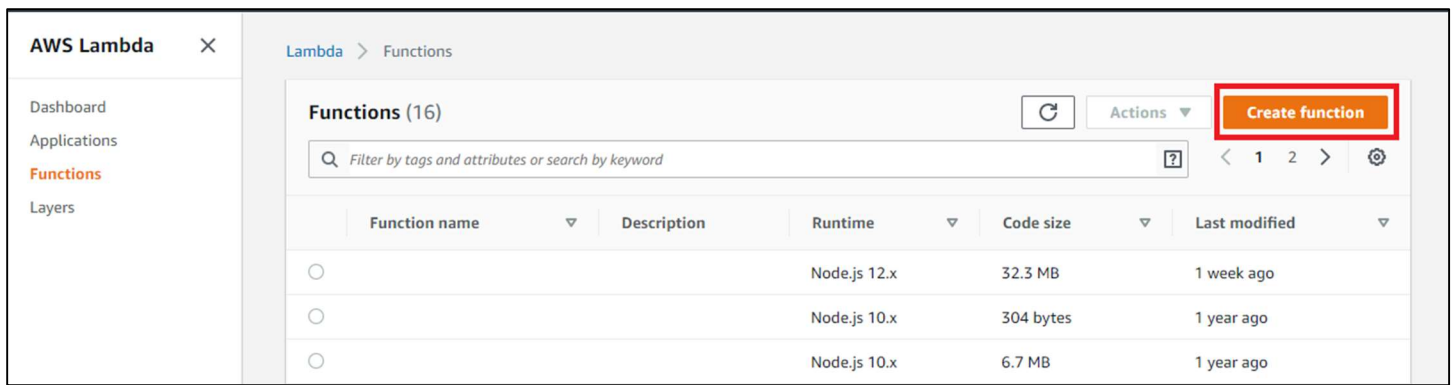


The AWS Root user sign in page features the AWS logo at the top. Below it is the 'Root user sign in' heading. The 'Email' field contains 'perezkarlvincent@gmail.com'. The 'Password' field is masked with dots. A 'Forgot password?' link is located to the right of the password field. A blue 'Sign in' button is positioned below the password field. At the bottom, there are two links: 'Sign in to a different account' and 'Create a new AWS account'.

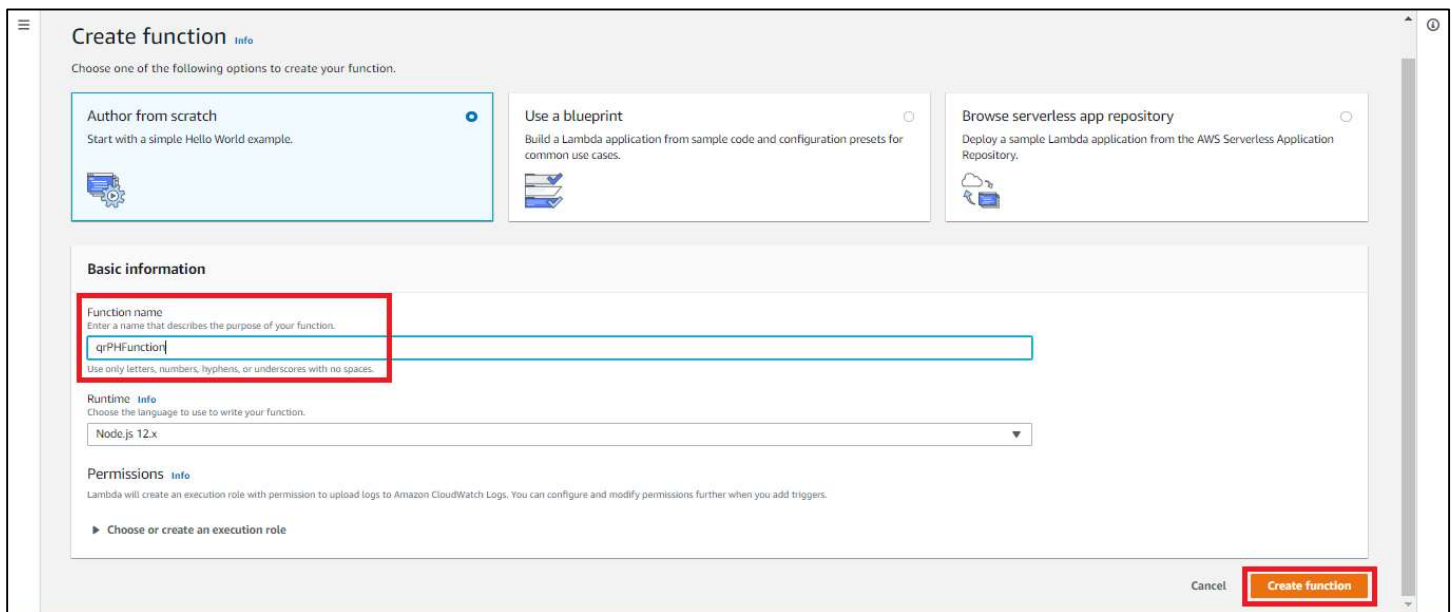
Step 2: Click Services and Search for Lambda.



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

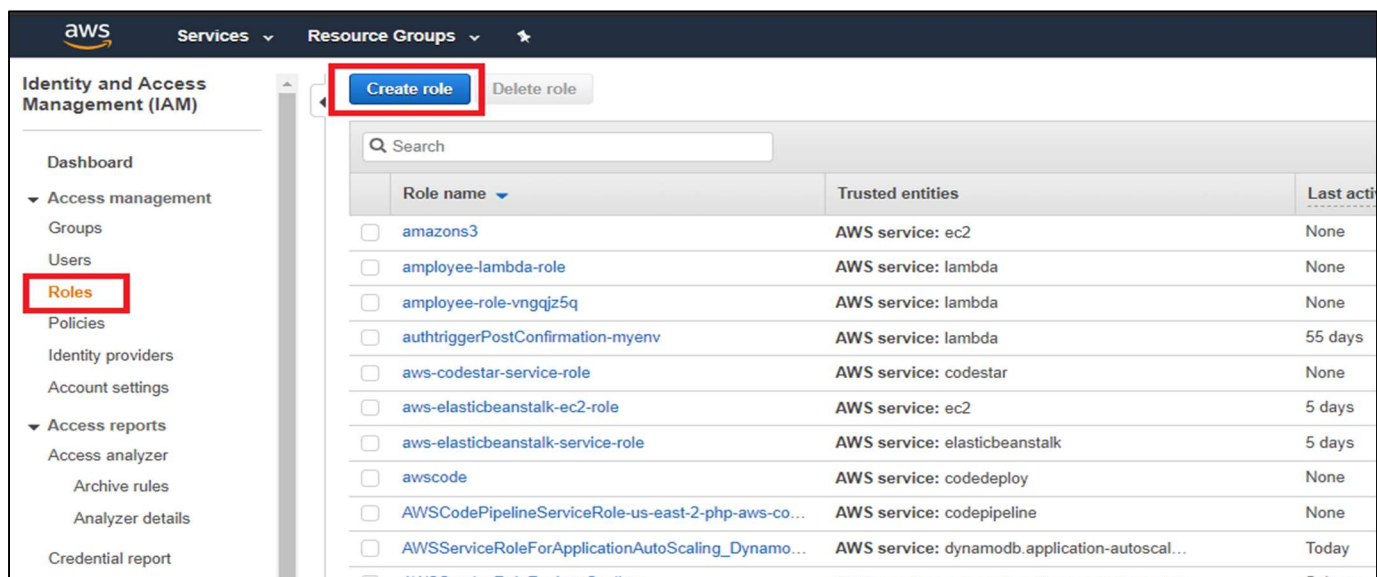


Step 4: On Create Function Page, under **Basic Information** Section, enter your function name (ex.qrPHFunction). Leave the **Permissions** field as it is, we will get back later to it to add permission our Lambda to access DynamoDB. then Click **Create Function** button.



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**.

The screenshot shows the 'Create role' page in the AWS IAM console. The 'Select type of trusted entity' section has four options: 'AWS service' (selected), 'Another AWS account', 'Web identity', and 'SAML 2.0 federation'. Below this, the 'Choose a use case' section lists common use cases. The 'Lambda' use case is highlighted with a red box. At the bottom right, the 'Next: Permissions' button is also highlighted with a red box.

Create role

Select type of trusted entity

- AWS service** (selected): EC2, Lambda and others
- Another AWS account: Belonging to you or 3rd party
- Web identity: Cognito or any OpenID provider
- SAML 2.0 federation: Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose a use case

Common use cases

- Lambda** (highlighted): Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

API Gateway	CodeDeploy	EMR	KMS	RoboMaker
AWS Backup	CodeGuru	ElastiCache	Kinesis	S3
AWS Chatbot	CodeStar Notifications	Elastic Beanstalk	Lambda	SMS
AWS Support	Comprehend	Elastic Container Service	Lex	SNS

* Required

[Cancel](#) [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.

The screenshot shows the 'Create role' page in the AWS IAM console, Step 2: Attach permissions policies. The 'Filter policies' search bar contains 'DynamoDb'. The 'AmazonDynamoDBFullAccess' policy is selected with a checkbox. At the bottom right, the 'Next: Tags' button is highlighted with a red box.

Create role

Attach permissions policies

Choose one or more policies to attach to your new role.

[Create policy](#)

Filter policies: Showing 8 results

	Policy name	Used as
<input checked="" type="checkbox"/>	AmazonDynamoDBFullAccess (highlighted)	Permissions policy (13)
<input type="checkbox"/>	AmazonDynamoDBFullAccessWithDataPipeline	None
<input type="checkbox"/>	AmazonDynamoDBReadOnlyAccess	None
<input type="checkbox"/>	AWSApplicationAutoscalingDynamoDBTablePolicy	Permissions policy (1)
<input type="checkbox"/>	AWSLambdaDynamoDBExecutionRole	None
<input type="checkbox"/>	AWSLambdaInvocation-DynamoDB	None
<input type="checkbox"/>	DynamoDBCloudWatchContributorInsightsServiceRolePolicy	None
<input type="checkbox"/>	DynamoDBReplicationServiceRolePolicy	None

Set permissions boundary

* Required

[Cancel](#) [Previous](#) [Next: Tags](#)

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.

aws Services Resource Groups

Create role

1 2 3 4

Review

Provide the required information below and review this role before you create it.

Role name*

Use alphanumeric and '+@_-' characters. Maximum 64 characters.

Role description

Maximum 1000 characters. Use alphanumeric and '+@_-' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies AmazonDynamoDBFullAccess

Permissions boundary Permissions boundary is not set

No tags were added.

* Required

Cancel Previous **Create role**

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliate

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

aws Services Resource Groups

Identity and Access Management (IAM)

Dashboard

Access management

- Groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

Access reports

The role qrPH-lambda-role has been created.

Create role Delete role

Q qrPH|

Role name	Trusted entities	Last activity
<input checked="" type="checkbox"/> qrPH-lambda-role	AWS service: lambda	None
<input type="checkbox"/> qrPHFunction-role-7p2bmvik	AWS service: lambda	None
<input type="checkbox"/> qrPHFunction-role-8allqkvm	AWS service: lambda	None

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.

aws Services Resource Groups

AWS Lambda

Dashboard

Applications

Functions

Layers

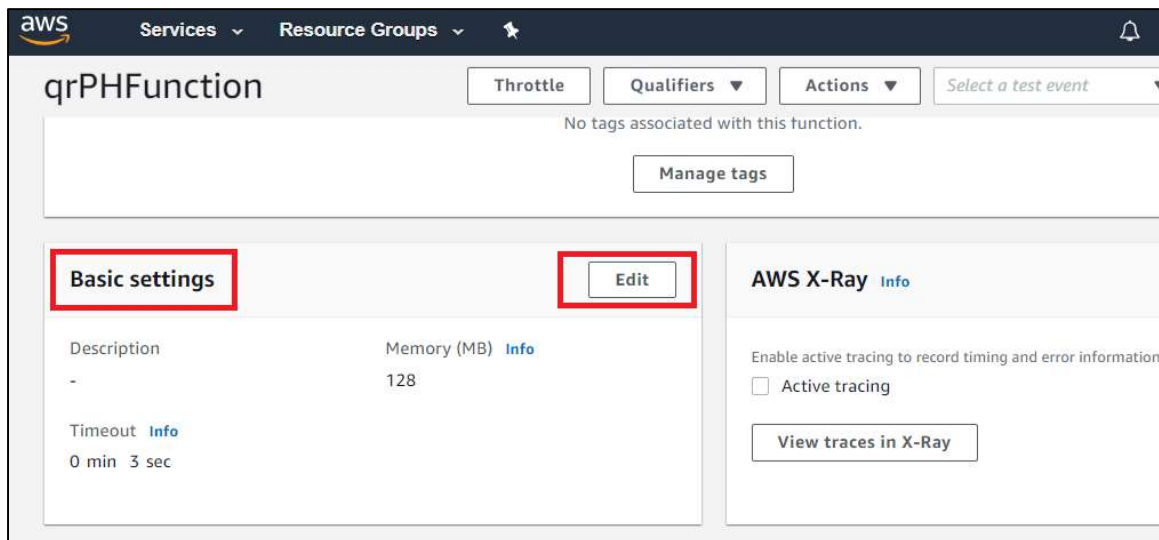
Lambda > Functions

Functions (17)

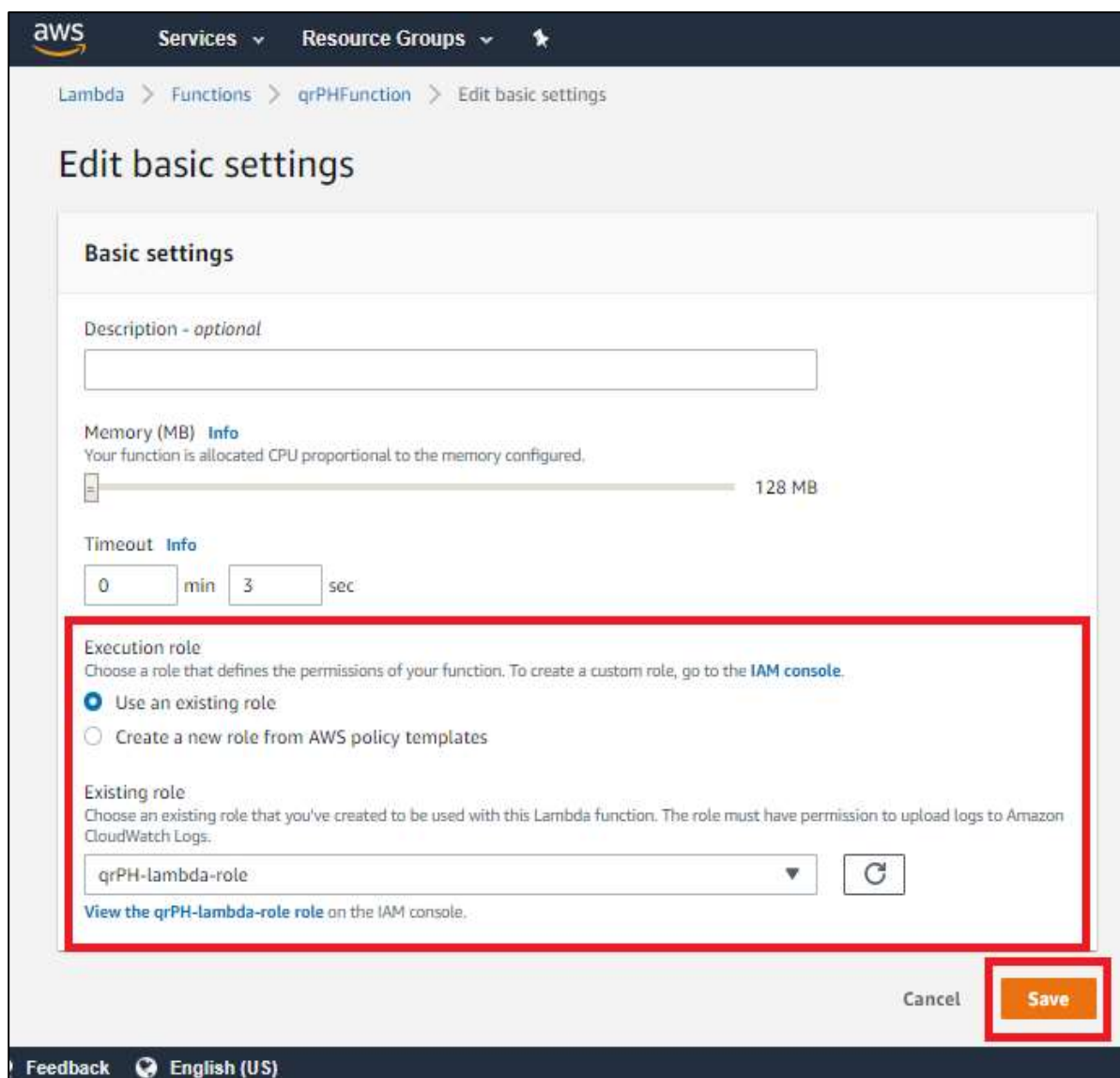
Filter by tags and attributes or search by keyword

	Function name	Description	Runtime	Code size	Last
<input type="radio"/>			Node.js 12.x	32.3 MB	1 we
<input type="radio"/>			Node.js 10.x	304 bytes	1 ye
<input checked="" type="radio"/>	qrPHFunction		Node.js 12.x	304 bytes	2 ho
<input type="radio"/>			Node.js 10.x	6.7 MB	1 ye
<input type="radio"/>			Node.js 10.x	8.2 MB	1 ye
<input type="radio"/>			Node.js 10.x	34.9 MB	1 da

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

The screenshot shows the 'Edit environment variables' page for a Lambda function named 'qrPHFunction'. The page title is 'Edit environment variables'. Below the title, there's a section 'Environment variables' with a description: 'You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)'. Below this, there's a table with two rows of environment variables, both highlighted with a red box:

Key	Value	
TABLE	qrph_tbl	<button>Remove</button>
NODE_ENV	production	<button>Remove</button>

Below the table is an Add environment variable. At the bottom right of the page, there are Cancel and Save buttons, with the Save button also highlighted by a red box.

Configure DynamoDb

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

The screenshot shows the 'Create table' page in the AWS DynamoDB console. The page title is 'Create table'. Below the title, there's a description of Amazon DynamoDB: 'Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability.' Below this, there's a Create table button, which is highlighted with a red box. Below the button, there's a 'Recent alerts' section, a 'Total capacity for US East (Ohio)' section, and an 'Account capacity limits utilization for US East (Ohio)' section with two charts.

Total capacity for US East (Ohio)			
Provisioned read capacity	239 (Max: 80000)	Reserved read capacity	0
Provisioned write capacity	216 (Max: 80000)	Reserved write capacity	0

Below the table, there's an 'Account capacity limits utilization for US East (Ohio)' section with a 'Time range' dropdown set to 'Last 24 Hours' and a refresh button. Below this, there are two charts: 'Provisioned read capacity limit utilization (%)' and 'Provisioned write capacity limit utilization (%)'. Both charts show utilization over time, with the x-axis labeled with dates and times: 3/18 16:00, 3/19 00:00, and 3/19 08:00.

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

The screenshot shows the 'Create DynamoDB table' page in the AWS Management Console. The 'Table name' field is filled with 'qrph_tbl'. The 'Primary key' section shows 'id' as the partition key with a type of 'String'. Below this, there are 'Table settings' including 'Use default settings' (checked) and a list of default settings: 'No secondary indexes', 'Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes', and 'Encryption at Rest with DEFAULT encryption type'. At the bottom right, the 'Create' button is highlighted with a red box.

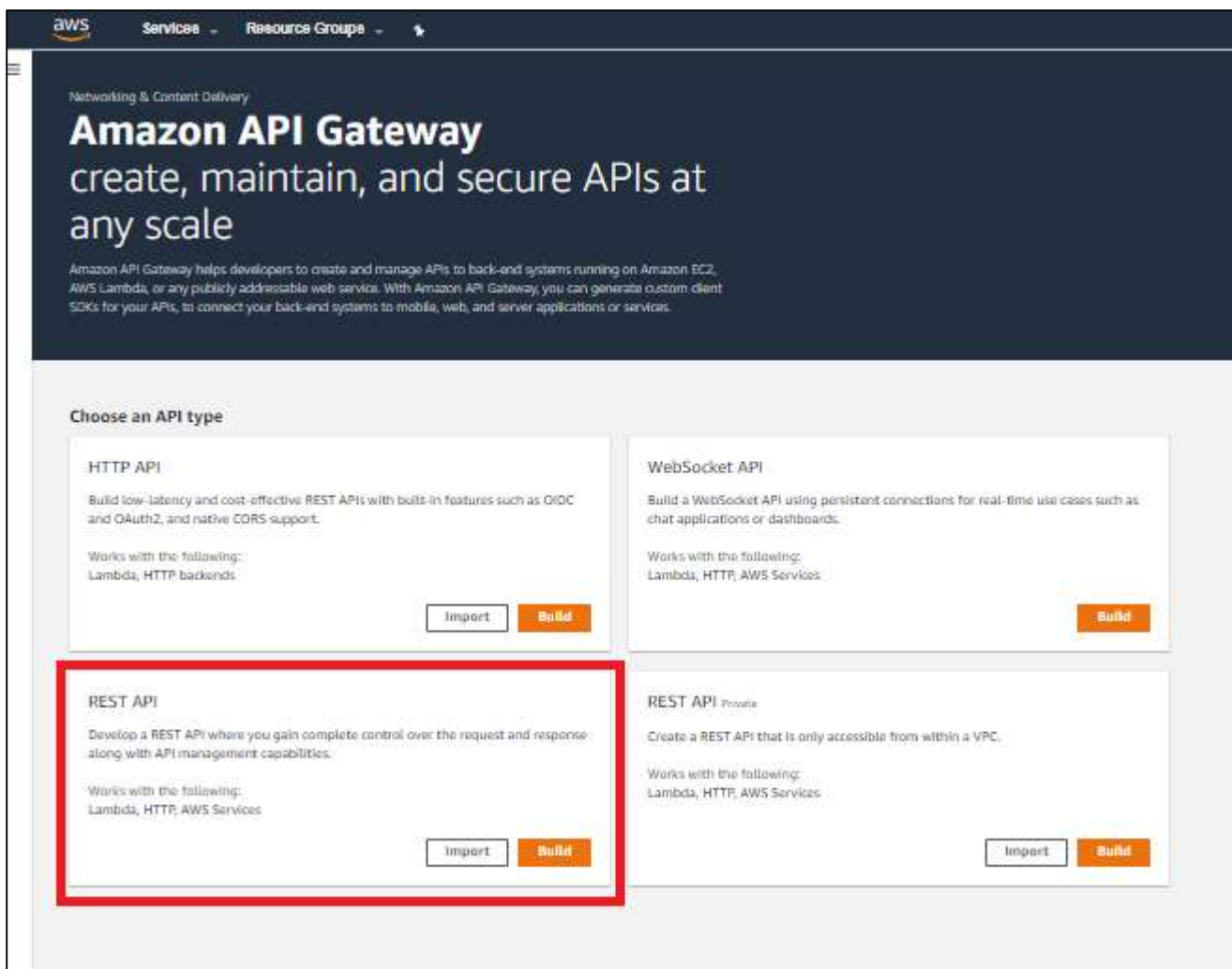
When the table is created you should be redirected to the table management route. This means that your table creation was successfully. We are done with the **DynamoDB** configuration.

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**.

The screenshot shows the 'Amazon API Gateway' landing page. It features the Amazon API Gateway logo at the top. Below the logo, there is a description of the service: 'Amazon API Gateway helps developers to create and manage APIs to back-end systems running on Amazon EC2, AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can generate custom client SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or services.' A 'Get Started' button is highlighted with a red box. Below the main content, there are three sections: 'Streamline API development', 'Performance at scale', and 'SDK generation', each with an icon and a brief description.

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.

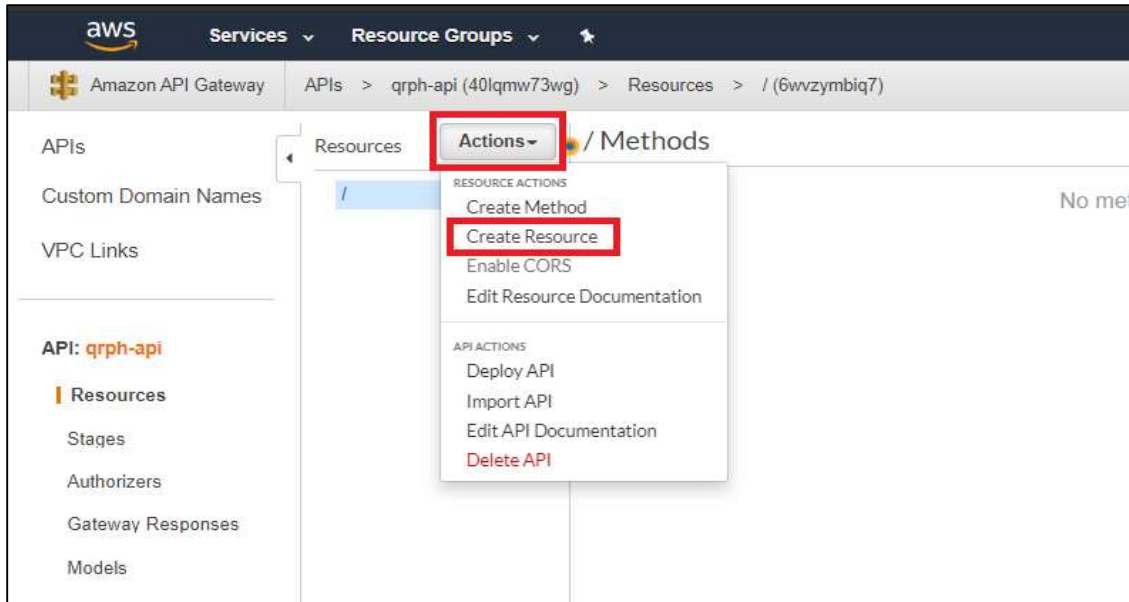
The screenshot shows the 'Create new API' form in the Amazon API Gateway console. The breadcrumb navigation at the top reads 'Amazon API Gateway > APIs > Create'. There are links for 'Show all hints' and a help icon.

The form is divided into sections:

- Choose the protocol**: Select whether you would like to create a REST API or a WebSocket API. The **REST** radio button is selected and highlighted with a red box.
- Create new API**: In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints. The **New API** radio button is selected and highlighted with a red box. Other options are 'Import from Swagger or Open API 3' and 'Example API'.
- Settings**: Choose a friendly name and description for your API. The **API name*** field contains 'qrph-api' and is highlighted with a red box. The **Description** field is empty. The **Endpoint Type** is set to 'Regional' with a dropdown arrow and an information icon.

At the bottom left, there is a note: '* Required'. At the bottom right, the **Create API** button is highlighted with a red box.

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**.



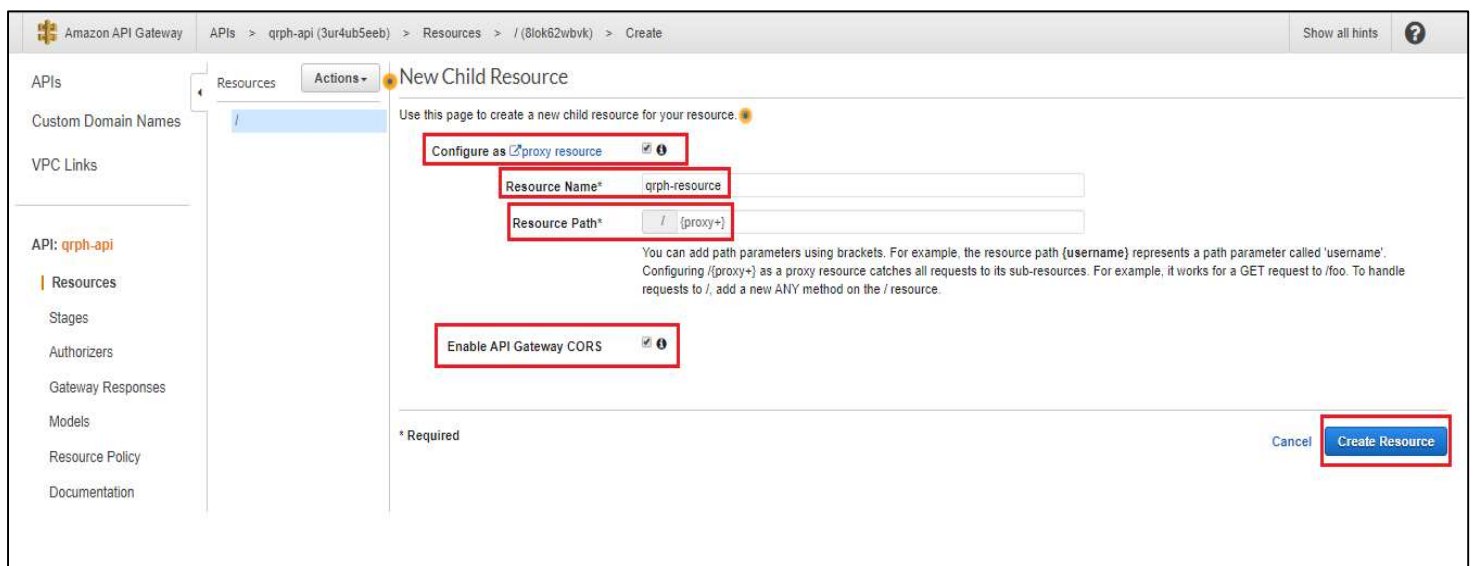
Step 19: On **New Child Resource Page**,

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 – 'employee-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.

Amazon API Gateway

APIs > qrph-api (3ur4ub5eeb) > Resources > /{proxy+} (f42vlf) > ANY

Resources

Actions - /{proxy+} - ANY - Setup

API Gateway will configure your ANY method as a proxy integration. Proxy integrations can communicate with HTTP endpoints or Lambda functions. API Gateway sends the entire request to HTTP endpoints, including resource path, headers, query string parameters, and body. For Lambda integrations, API Gateway applies a default mapping to send all of the request information and responses follow a default interface. To learn more read our [documentation](#)

Integration type

- ☒ Lambda Function Proxy
- ☐ HTTP Proxy
- ☐ VPC Link

Lambda Region: us-east-2

Lambda Function: qrPHFunction

Use Default Timeout: ☒

Save

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

Amazon API Gateway

APIs > qrph-api (3ur4ub5eeb) > Resources > /{proxy+} (f42vlf) > ANY

Resources

Actions - /{proxy+} - ANY - Method Execution

METHOD ACTIONS

- Edit Method Documentation
- Delete Method

RESOURCE ACTIONS

- Create Method
- Create Resource
- Enable CORS
- Edit Resource Documentation
- Delete Resource

API ACTIONS

- Deploy API**
- Import API
- Edit API Documentation
- Delete API

Method Request

Auth: NONE

ARN: arn:aws:execute-api:us-east-2:699426830907:3ur4ub5eeb/*/*/*

Method Response

HTTP Status: Proxy

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

Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage: [New Stage]

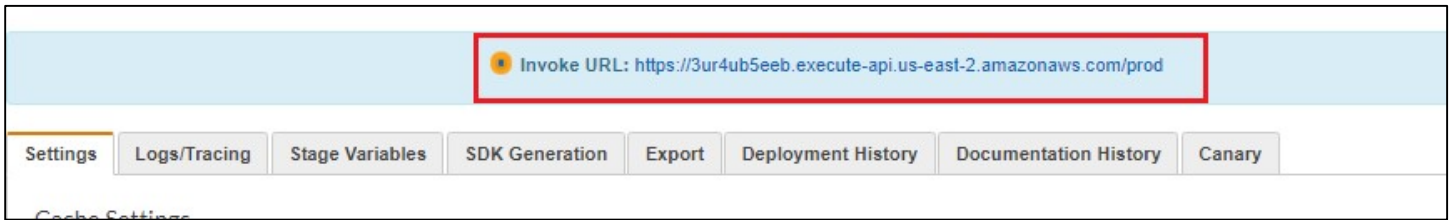
Stage name*: prod

Stage description:

Deployment description:

Cancel Deploy

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



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

How To Test IT:

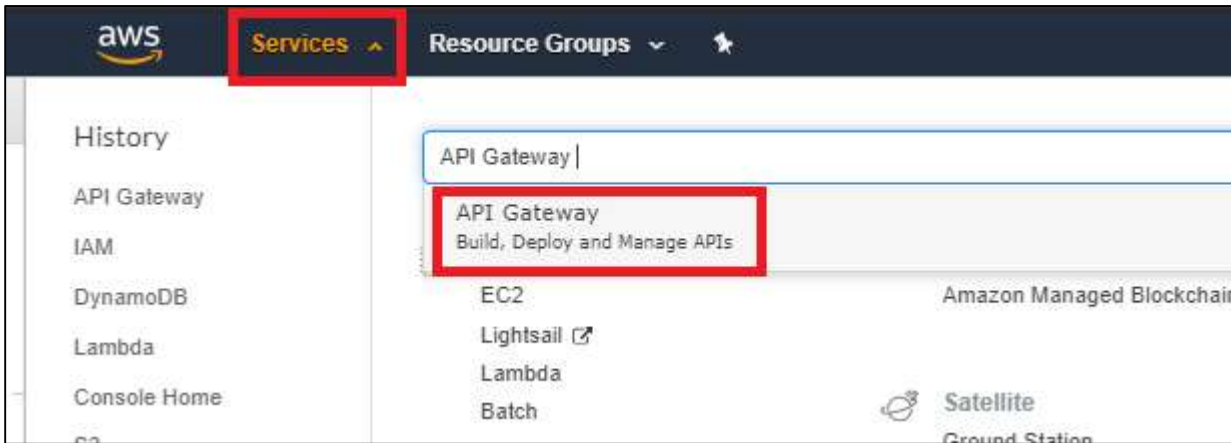
Go to your Browser then paste the generated **API Endpoint**, then add the API Name on the end of the URL. (ex. <https://3ur4ub5eeb.execute-api.us-east-2.amazonaws.com/prod/test>)



Setting the API Key Security

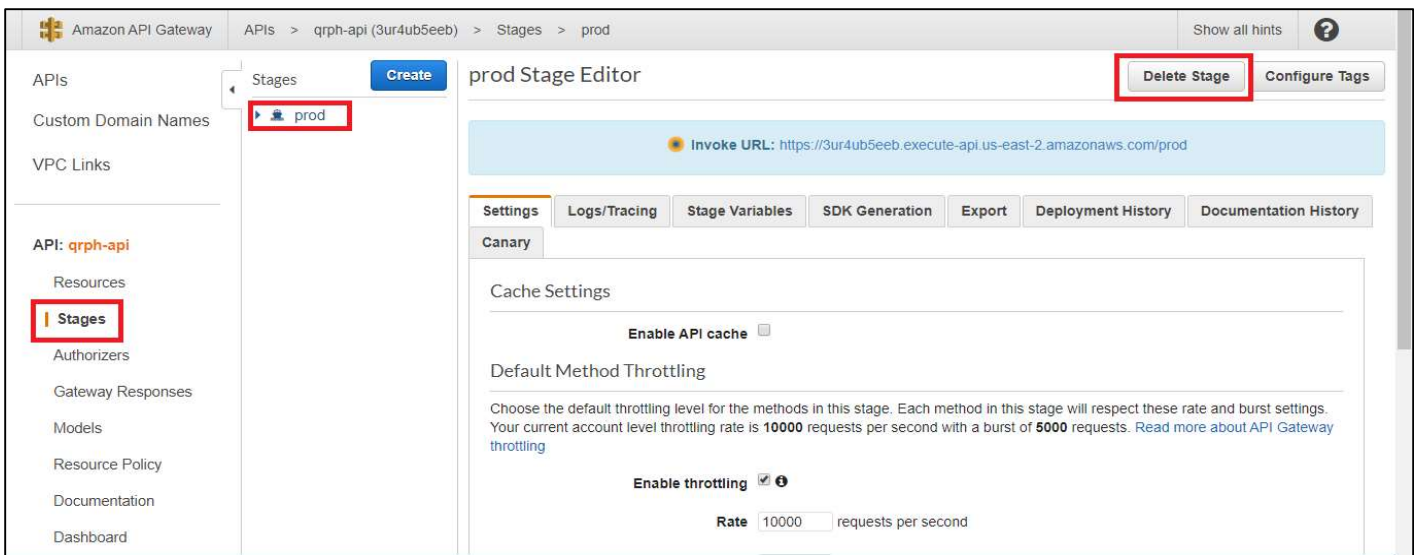
Step 1: Go to <https://aws.amazon.com/> and login on your account as **Root** account.

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

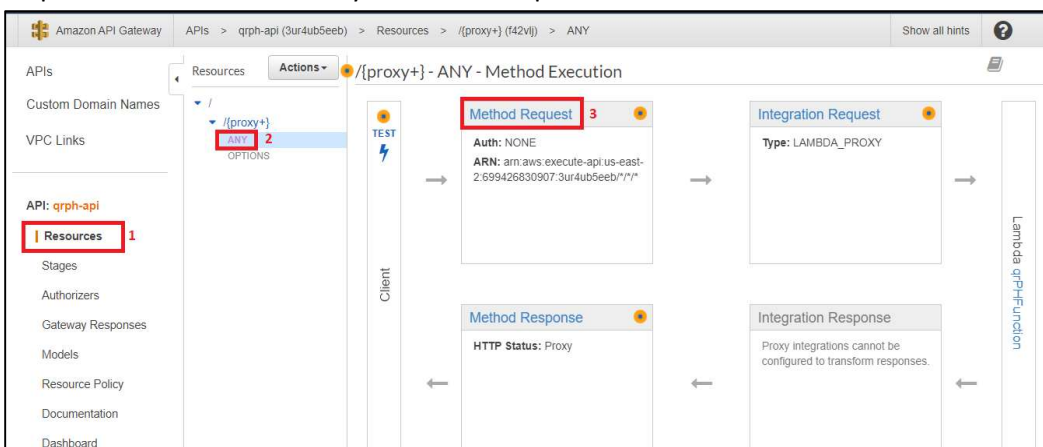


Step 3: Select your api (ex.qrph-api).Then select Stages

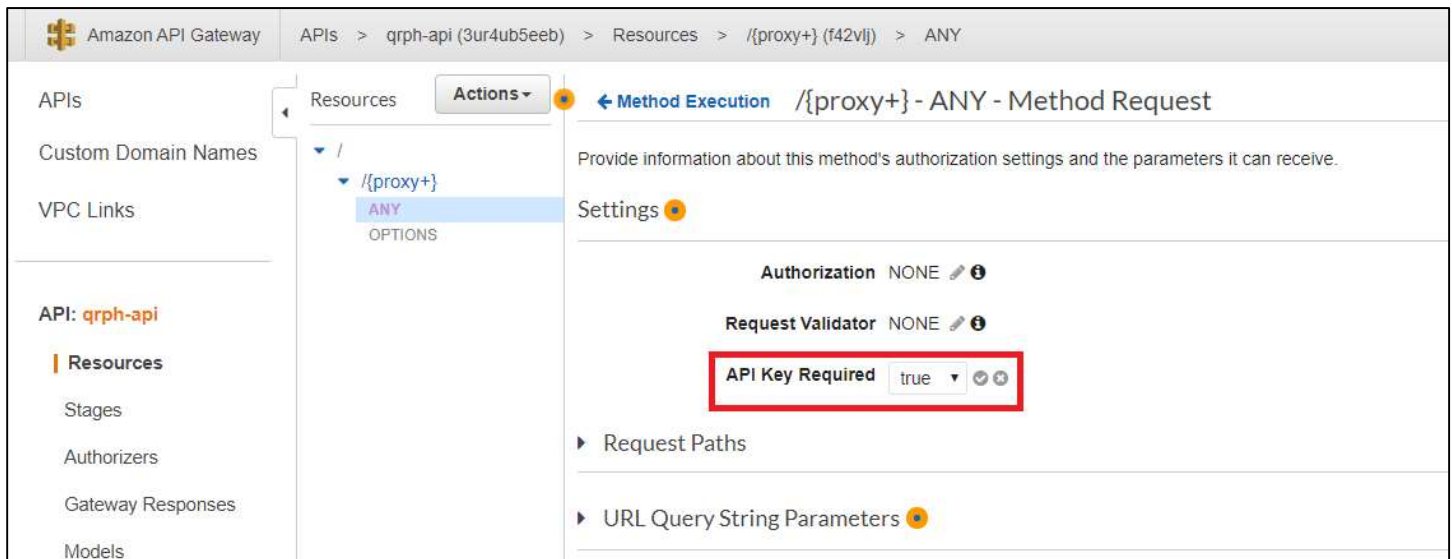
Note: If there is an existing Stage on Stages, Select the existing stage and Delete the existing Stage.



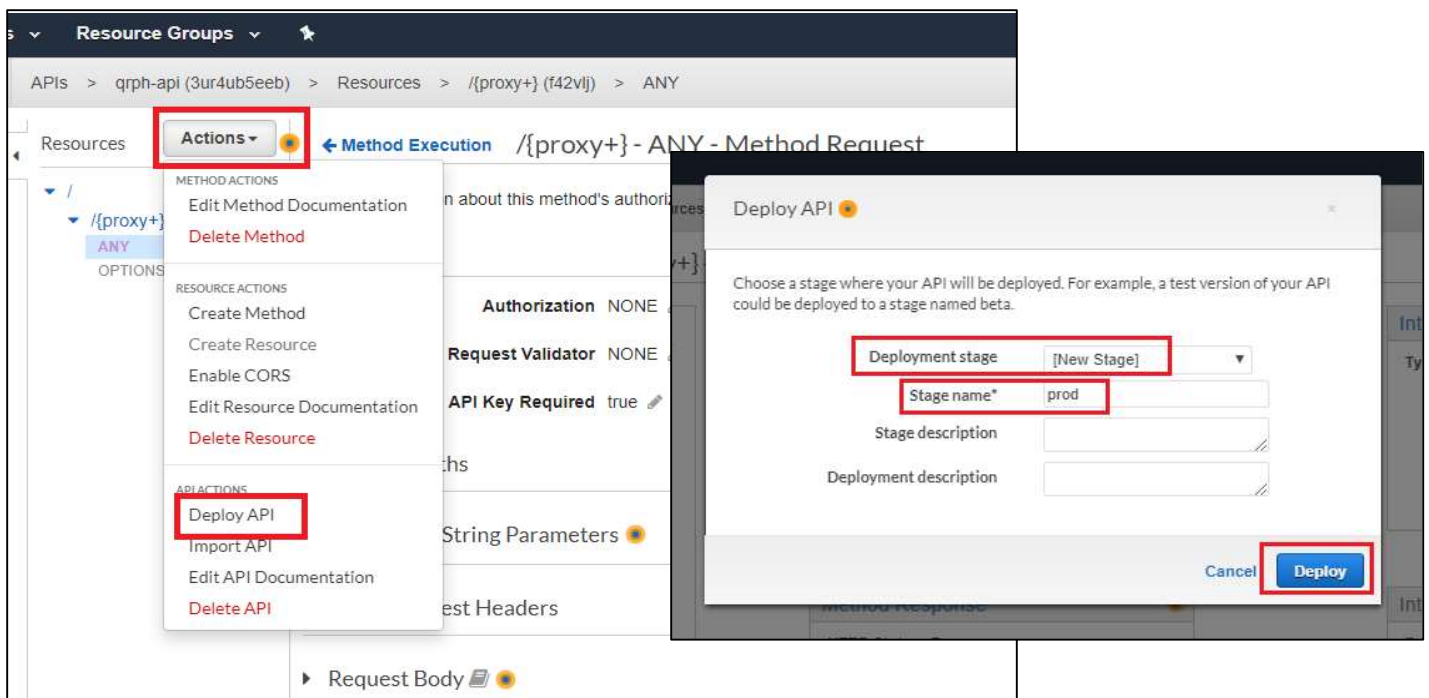
Step 4: Select Resources > Any > Method Request



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**.

Amazon API Gateway Usage Plans > Create

Usage Plans **Create** Create Usage Plan

Usage Plans help you meter API usage. With Usage Plans, you can enforce a throttling and quota limit on each API key. Throttling limits define the maximum number of requests per second available to each key. Quota limits define the number of requests each API key is allowed to make over a period.

Name* qrphUsagePlan

Description

Throttling

Enable throttling ☒

Rate* 10000 requests per second

Burst* 5000 requests

Quota

Enable quota ☐

Without quota limits, developers will not be restricted on the total number of API calls they can make with this usage plan.

* Required

Next

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

Amazon API Gateway Usage Plans > qrphUsagePlan (w6kh7n)

qrphUsagePlan

ID w6kh7n

Name qrphUsagePlan

Description No description.

Rate 10,000 requests per second

Burst 5,000 requests

Quota No quota.

Associated API Stages

Add API Stage

API	Stage	Method Throttling
qrph-api	prod	<input checked="" type="checkbox"/>

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

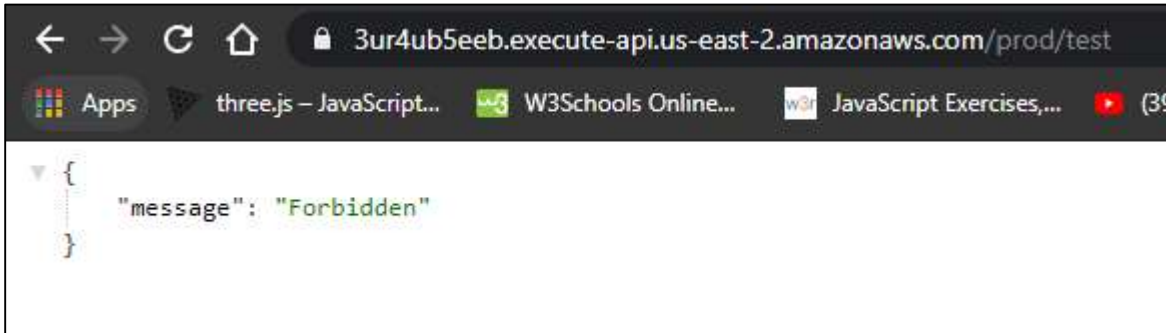
Step 10: Select your **created api key**. On API key, click **Show** to display the auto generated API key (ex.aAPI Key: mbQamcTFAv5mvYdthVg955FyTxCZtqYF67PNNcT0). Then click **Add to Usage Plan** button then Enter your **Usage Plan Name** (ex. qrphUsagePlan) then Click **Add/Check** button.

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>).

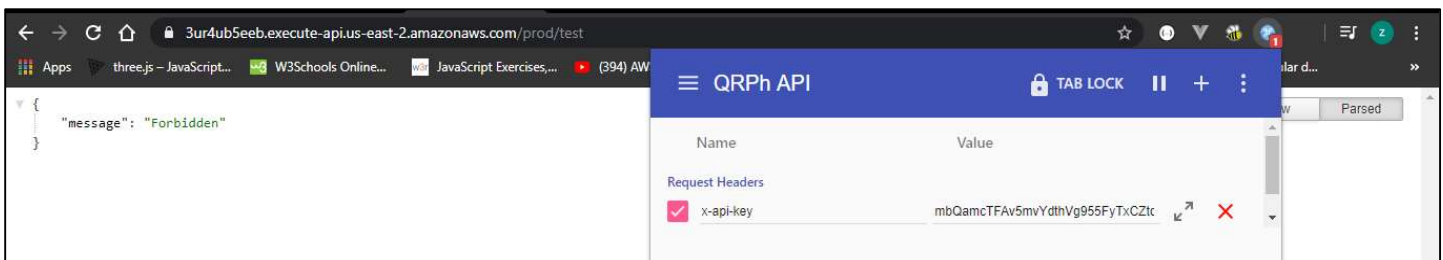
The API response must be “Forbidden”.



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!”

