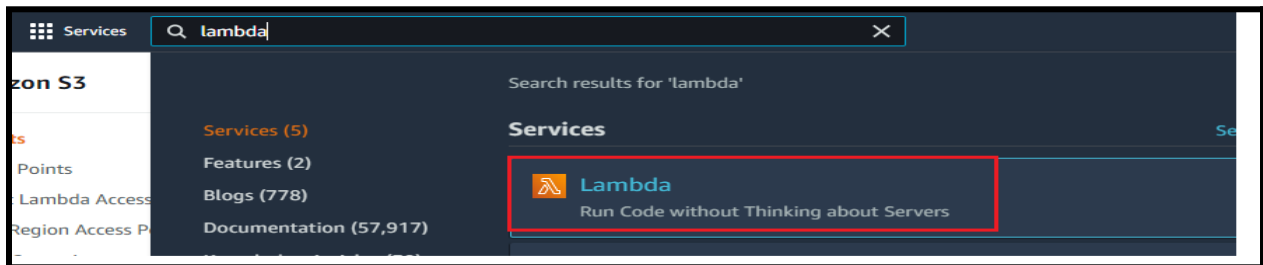
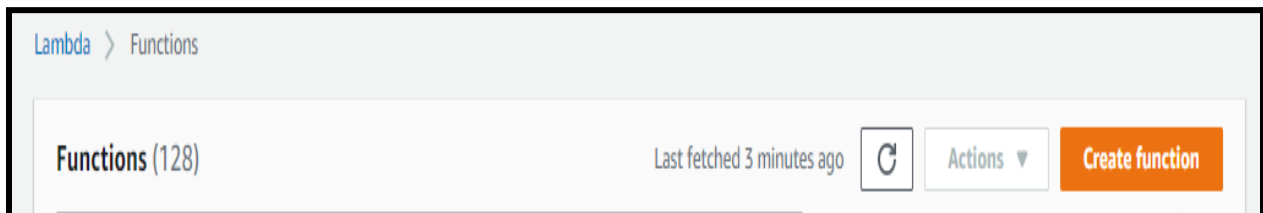


Create Lambda Function:

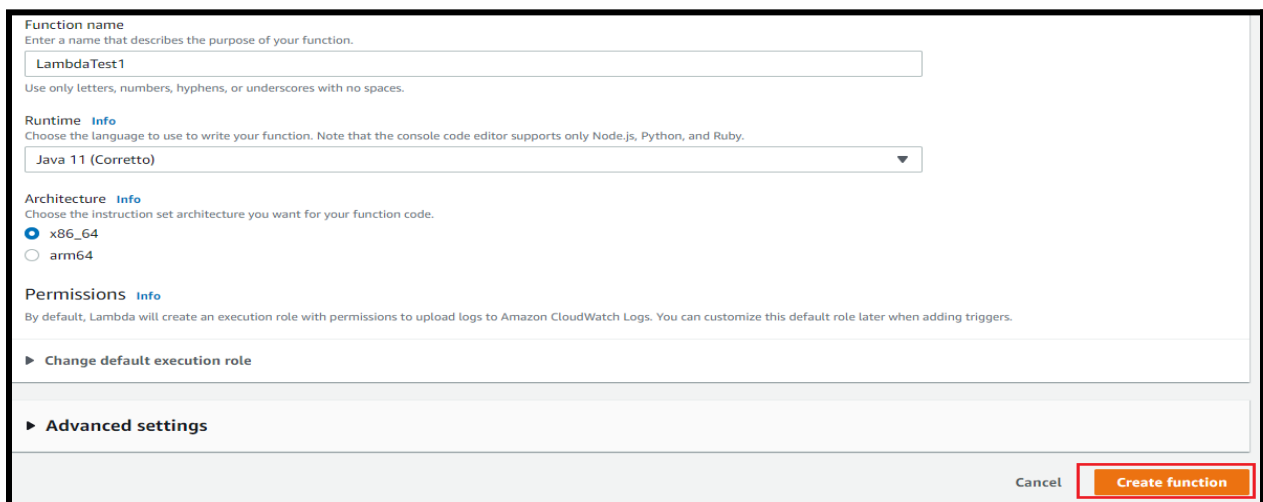
- Login to AWS Console. Click on Services, Search for Lambda, and open Lambda service.



- Click on the “Create Function” button

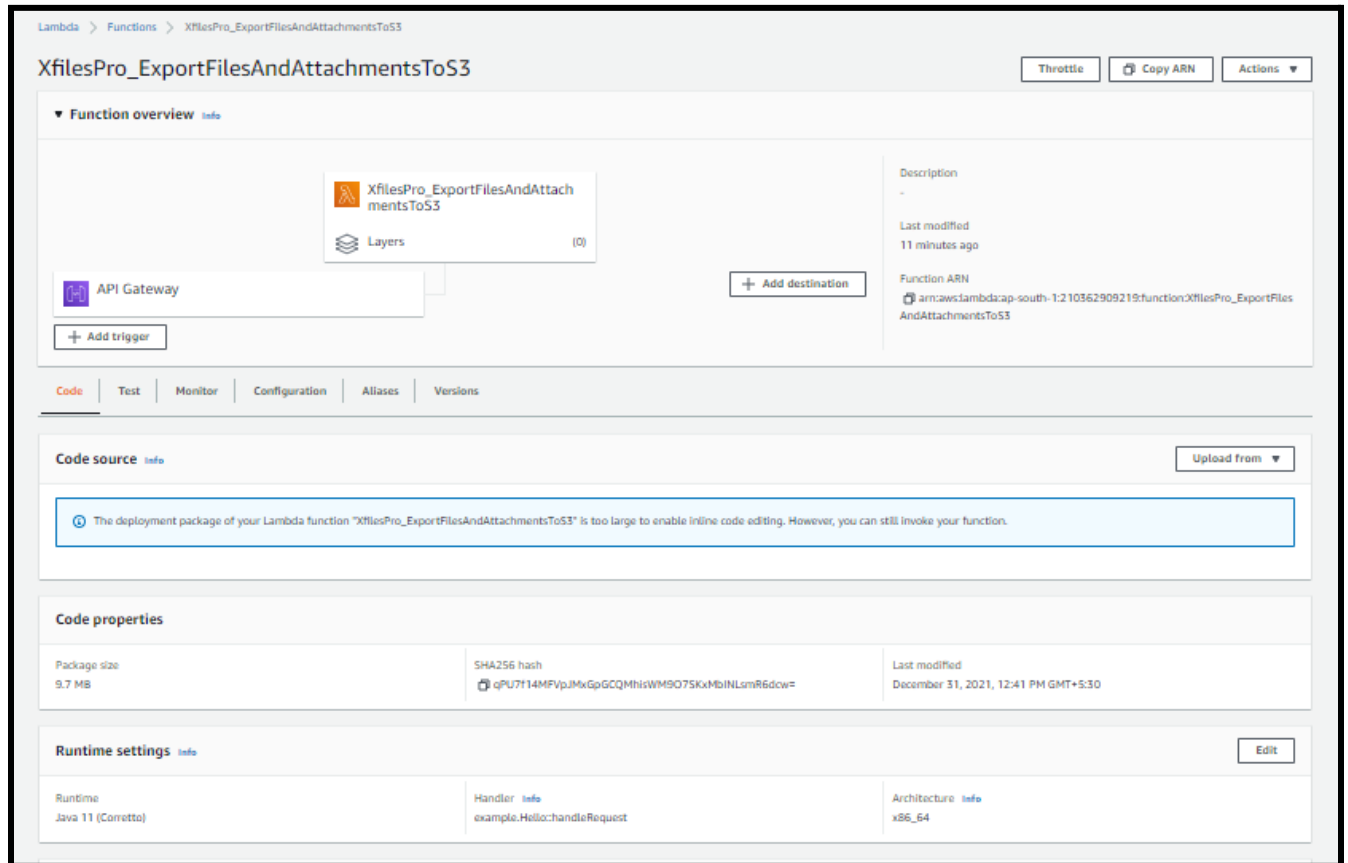


- Provide the Basic Information:
 - **Function Name** - Provide any name
 - **Runtime** - Select Java 11(Corretto) from the dropdown.
 - Click on the ‘Create Function’ button.



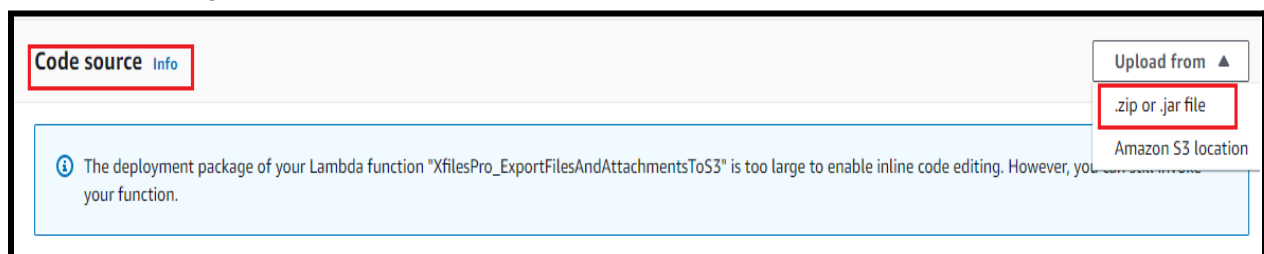
The screenshot shows the 'Create Function' form in the AWS console. The 'Function name' field is filled with 'LambdaTest1'. The 'Runtime' dropdown is set to 'Java 11 (Corretto)'. The 'Architecture' dropdown is set to 'x86_64'. The 'Permissions' section is expanded, showing 'Change default execution role' and 'Advanced settings' options. The 'Create function' button is highlighted with a red box.

- Function gets created and the below screen appears on the same screen.



The screenshot shows the AWS Lambda console for the function 'XfilesPro_ExportFilesAndAttachmentsToS3'. The 'Function overview' section displays the function name, layers (0), and a description. The 'Code source' section shows a message: 'The deployment package of your Lambda function "XfilesPro_ExportFilesAndAttachmentsToS3" is too large to enable inline code editing. However, you can still invoke your function.' The 'Code properties' section shows the package size (9.7 MB), SHA256 hash, and last modified date (December 31, 2021, 12:41 PM GMT+5:30). The 'Runtime settings' section shows the runtime (Java 11 (Corretto)), handler (example.Hello.handleRequest), and architecture (x86_64).

- From the **Code Source** section, click on **Upload from** (the dropdown button) and select **.zip or .jar file**



The screenshot shows the 'Code source' section of the AWS Lambda console. The 'Upload from' dropdown menu is open, showing options: '.zip or .jar file' and 'Amazon S3 location'. The '.zip or .jar file' option is highlighted with a red box.

- From the **Runtime Settings** section, click on the **Edit** button.
 - Select the Runtime as **Java 11 (Corretto)**
 - The handler should be given as “**com.ceptes.LambdaFunctionHandler**”
 - Architecture as **x86_64**
 - Click on the **Save** button.

Runtime settings Info			Edit
Runtime Java 11 (Corretto)	Handler Info com.ceptes.LambdaFunctionHandler::handleRequest	Architecture Info x86_64	

Lambda > Functions > XfilesPro_ExportFilesAndAttachmentsToS3 > Edit runtime settings

Edit runtime settings

Runtime settings [Info](#)

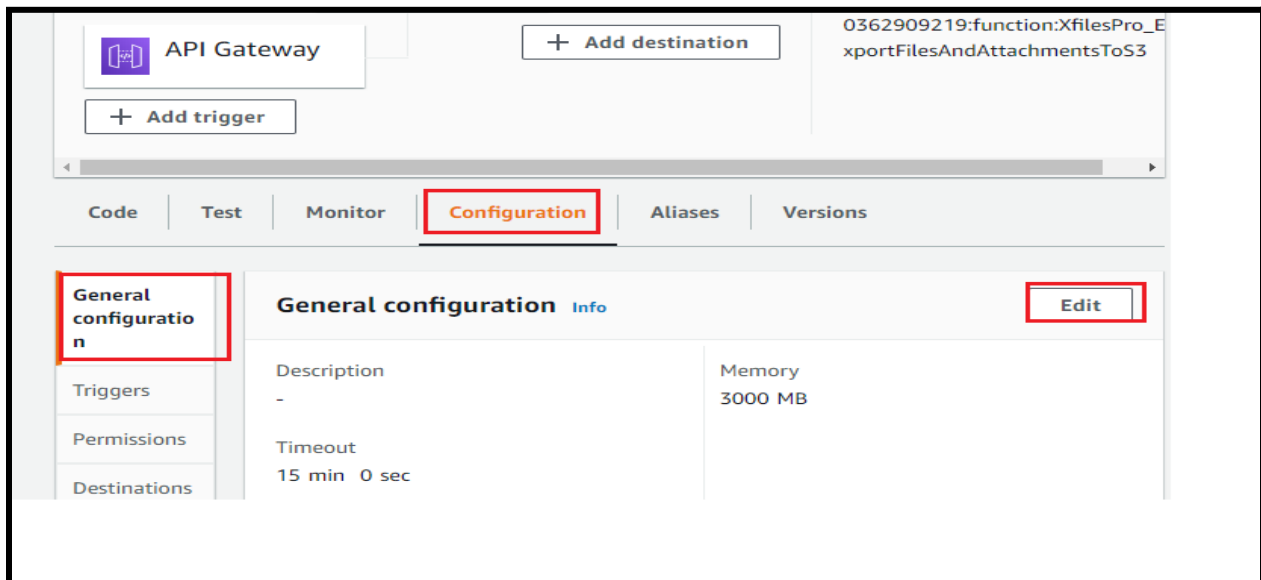
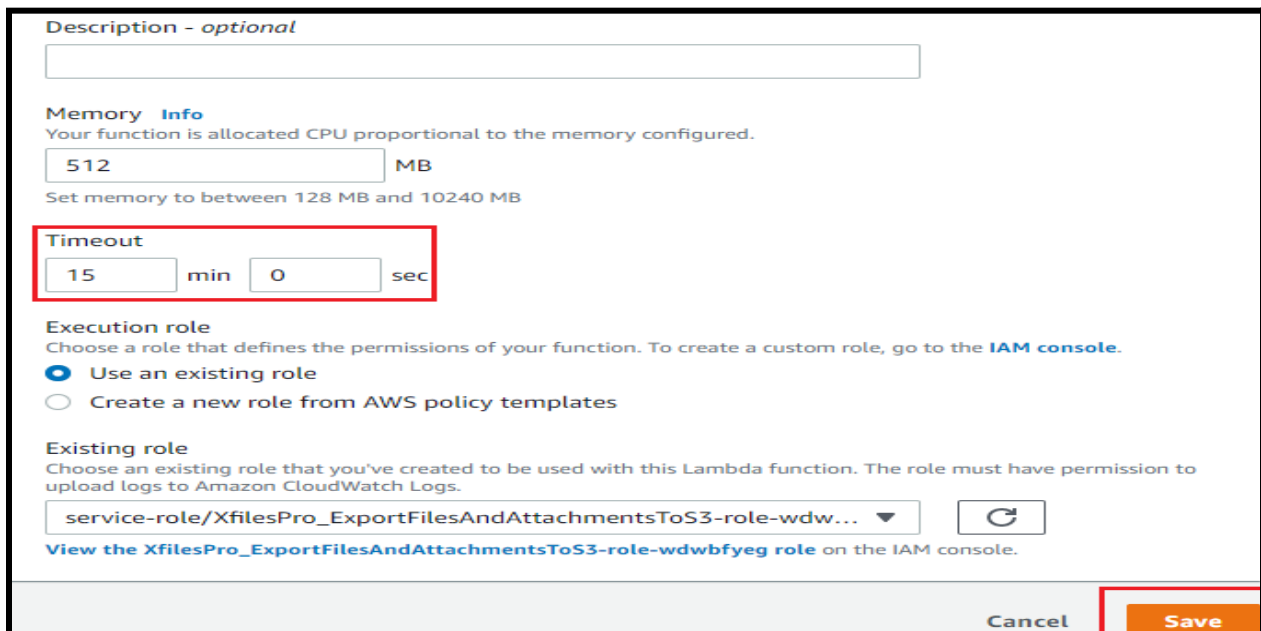
Runtime
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Java 11 (Corretto) ▼

Handler [Info](#)
com.ceptes.LambdaFunctionHandler::handleRequest

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

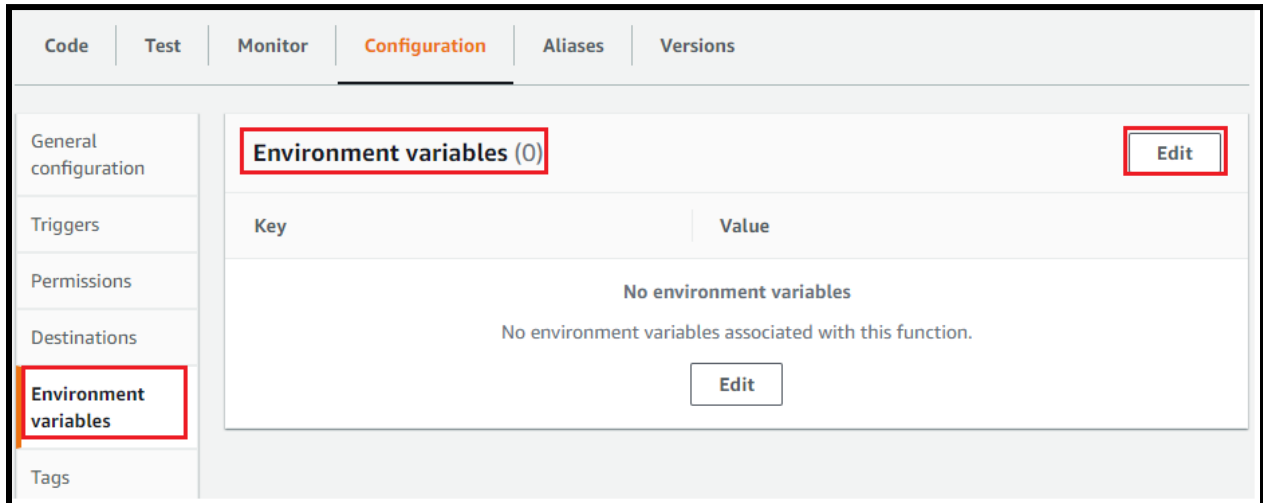
Cancel **Save**

- Click on **Configuration tab** and from the navigation section select **General Configuration** and Click on **Edit** button.
 - Let the memory be 512MB
 - Provide the Timeout as **15 min**
 - Click on **Save** button

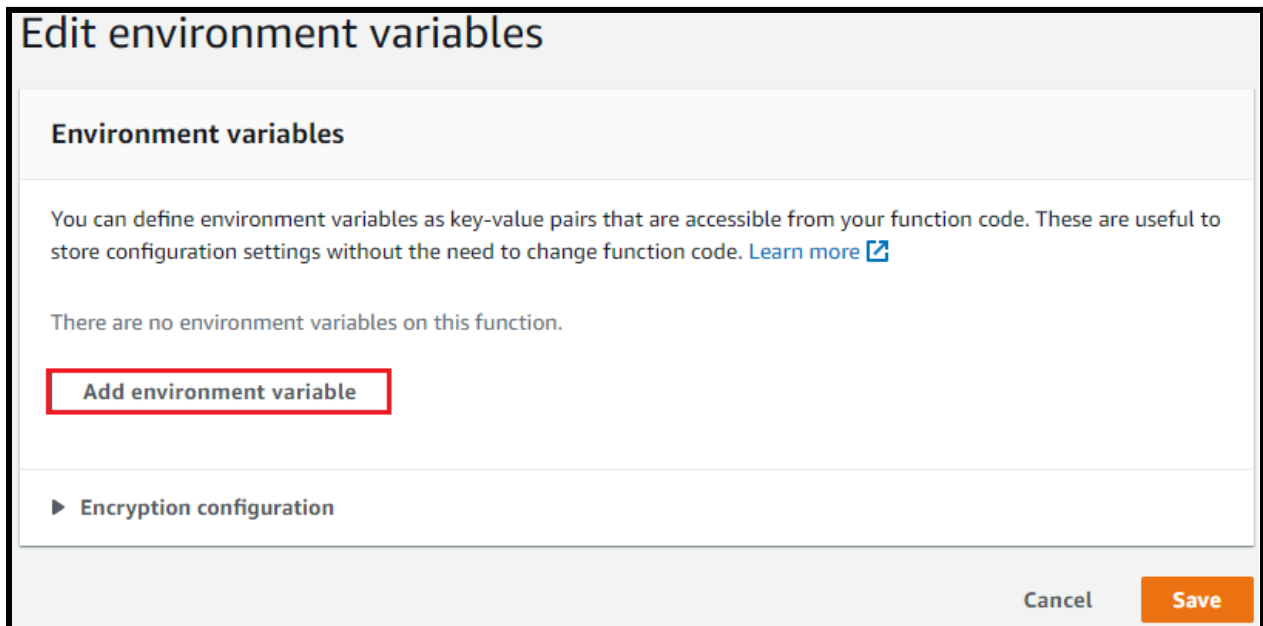
Add Environment Variables

- From the same **Configuration** tab, in the navigation bar select **Environment Variables** and click on **Edit** button.



The screenshot shows the 'Configuration' tab selected in the top navigation bar. On the left sidebar, 'Environment variables' is highlighted. The main content area displays 'Environment variables (0)' with an 'Edit' button. Below this, a message states 'No environment variables associated with this function.' with an 'Edit' button.

- Click On **Add environment variable** and add the following variables (listed in the table).



The screenshot shows the 'Edit environment variables' dialog box. It has a title 'Environment variables' and 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, it says 'There are no environment variables on this function.' A red box highlights the 'Add environment variable' button. At the bottom, there is an 'Add environment variable' button and an 'Encryption configuration' section.

Environment variables

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](#)

Key

Value

Remove

⚠ This field is required.

⚠ This field is required.

Add environment variable

Please provide the **Keys** and **Values** for the following environment variables:

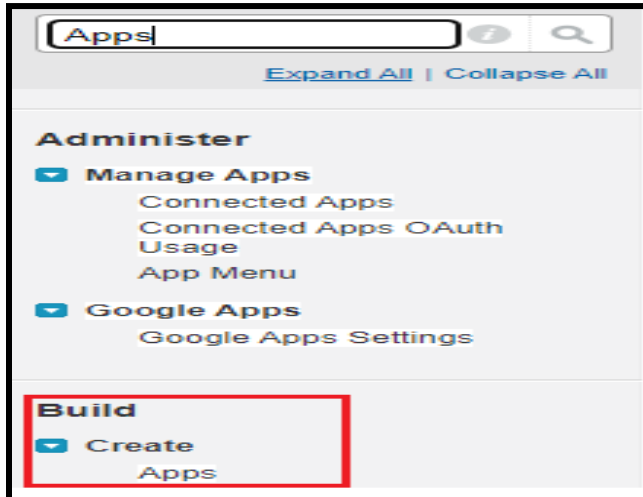
Key	Value
XFILES_ConnectedApp_ConsumerKey	Consumer Key of the connected App (see steps below to get the value)
XFILES_ConnectedApp_ConsumerSecret	ConsumerSecret of the Connected App (see steps below to get the value)
XFILES_SFDC_UserName	Salesforce User Name
XFILES_SFDC_Password	Salesforce Password and Security Token (see steps below to get the value)
XFILES_SFDC_Environment	Salesforce login URL
XFILES_S3_AccessKey	S3 Access Key
XFILES_S3_SecretAccessKey	S3 Secret Access Key
XFILES_SFDC_Namespace	XFILES

Key	Value	
XFILES_ConnectedApp_ConsumerKey	5MVG5nCAh8HhK1U98glaX0Wn8FV7t	Remove
XFILES_ConnectedApp_ConsumerSecret	50376E90455FD457E32300641D3844H	Remove
XFILES_S3_AccessKey	AKIATBGU3HRSU577R7E	Remove
XFILES_S3_SecretAccessKey	LPDynyJom7hpVL702ptkEOLdVvORscUj	Remove
XFILES_SFDC_Environment	https://login.salesforce.com	Remove
XFILES_SFDC_Namespace	XFILES	Remove
XFILES_SFDC_Password	Ceptes@123qweEW10XUJDTmP7LnUs	Remove
XFILES_SFDC_UserName	xfpreprodqa@xfiles.com	Remove

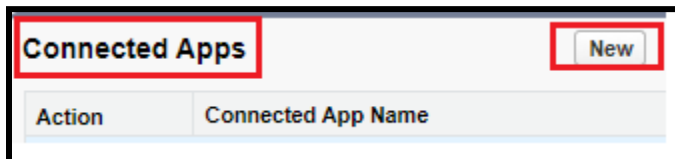
- Click On **Save** Button.

How to get the **ConsumerKey** and **ConsumerSecret**

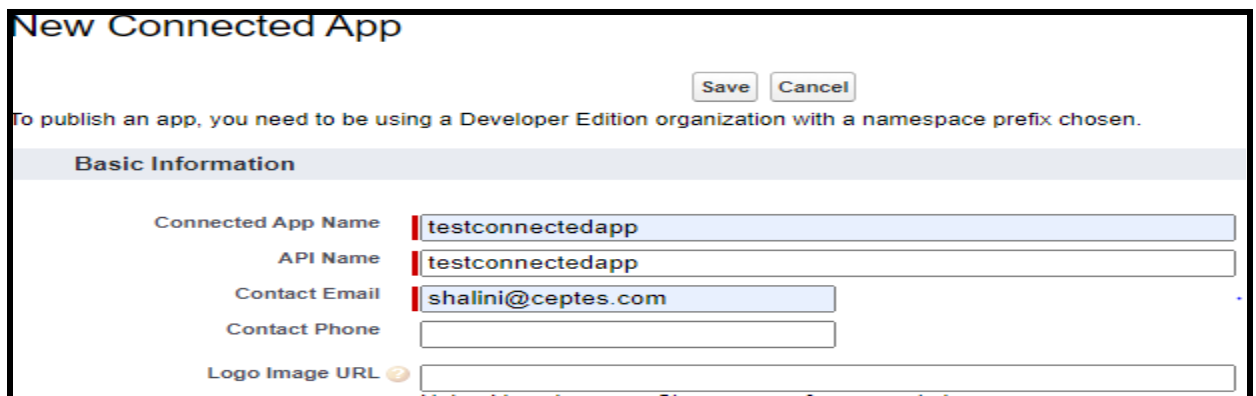
1. In Quick find search for **Create** and click on **Apps**



2. From the **Connected Apps** section, click on the **New** button.



3. **Basic Information** section, Provide the mandatory fields - Connected App Name, API Name and Contact Email.



New Connected App

Save Cancel

To publish an app, you need to be using a Developer Edition organization with a namespace prefix chosen.

Basic Information

Connected App Name: testconnectedapp

API Name: testconnectedapp

Contact Email: shalini@ceptes.com

Contact Phone:

Logo Image URL:

4. API (Enable OAuth Settings) section.
 - a. **Enable OAuth Settings**: should be checked
 - b. **Callback URL**: should be the org domain url.
 - c. **Selected OAuth Scopes**: provide **Access Connect REST API resources (chatter_api)**.
 - d. Click on **Save** button and again click on **Continue**.

API (Enable OAuth Settings)

Enable OAuth Settings ☒

Enable for Device Flow ☐

Callback URL

https://xfpreprod.my.salesforce.com/

Use digital signatures ☐

Selected OAuth Scopes

Available OAuth Scopes

Access Analytics REST API Charts Geodata resources (eclair_api)

Access Analytics REST API resources (wave_api)

Access Connect REST API resources (chatter_api)

Access Lightning applications (lightning)

Access Visualforce applications (visualforce)

Access content resources (content)

Access custom permissions (custom_permissions)

Access the identity URL service (id, profile, email, address, phone)

Access unique user identifiers (openid)

Manage Pardot services (pardot_api)

Add

Remove

Selected OAuth Scopes

Full access (full)

Connected App gets created with **Consumer Key** and **Consumer Secret** (Click on Click to reveal button)

API (Enable OAuth Settings)

Consumer Key

3MVG9Nk1FpUrSQHccc5X0.7Z6LcsQZNF0ZP104RPaTVMaB0T0iW7f0JdS3fP3s9GZFk.EZ3LgTkyfEOvtqB

Copy

Consumer Secret

341A98A7

Copy

Selected OAuth Scopes

Full access (full)

Enable for Device Flow ☐

Callback URL

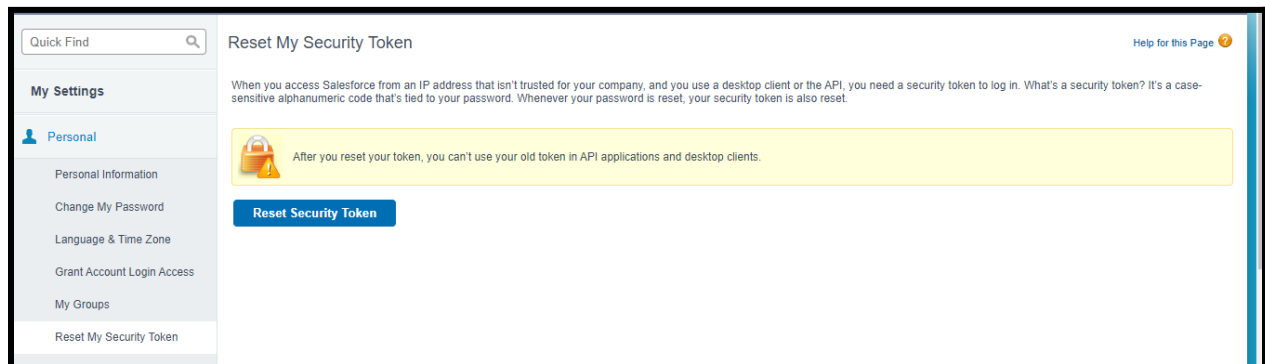
https://xfpreprod.my.salesforce.com/

Require Secret for

☒

How to get the **Security Token** (if you don't have the token already)

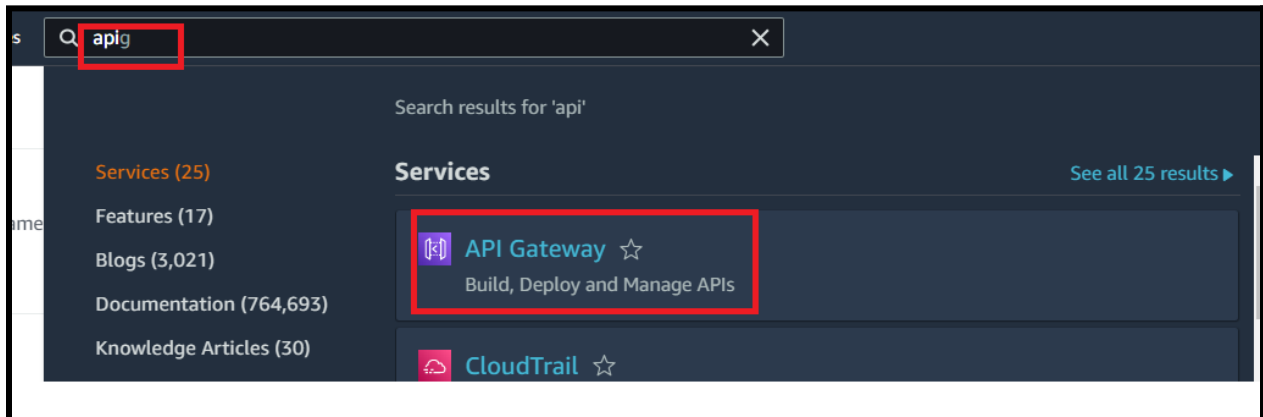
1. In the salesforce setup go to **My Setting**.
2. Click on **Personal**.
3. Click on **Reset My Security Token** and Click On **Reset Security Token** (see screenshot below).
- 4.



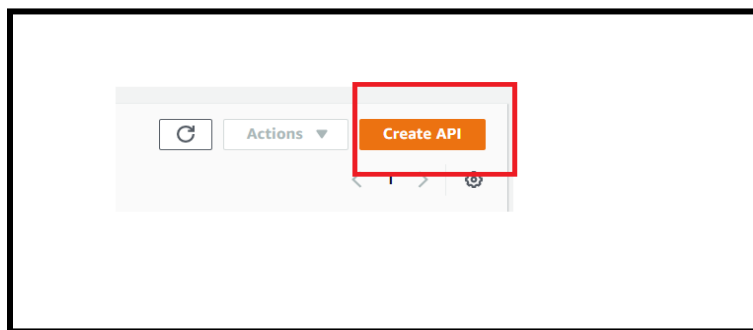
5. Please check your email and copy the token.
6. Add the environment variable value in the below format.
For example: if your password is ceptes123 and the security token is etrtyqtewryey then add the value as **ceptes123etrtyqtewryey**.
Note: Here we are not supporting passwords with special characters.

Create API Gateway

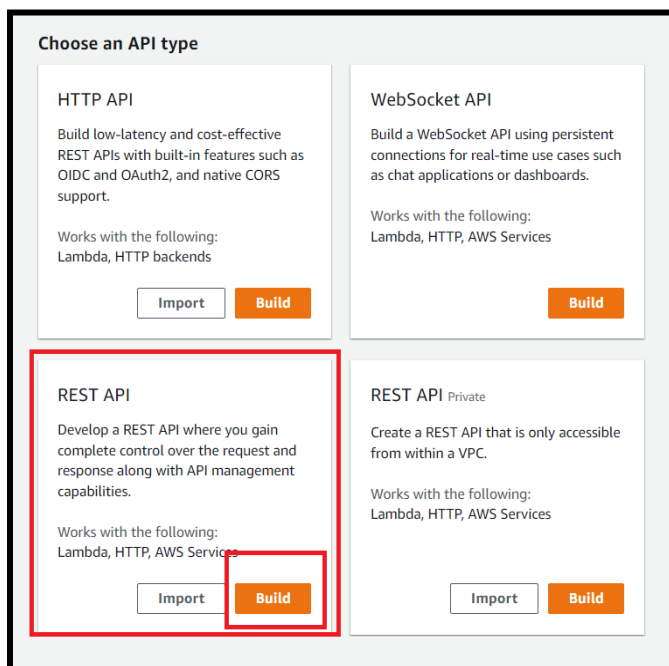
- Click on Services, open **API Gateway** service.



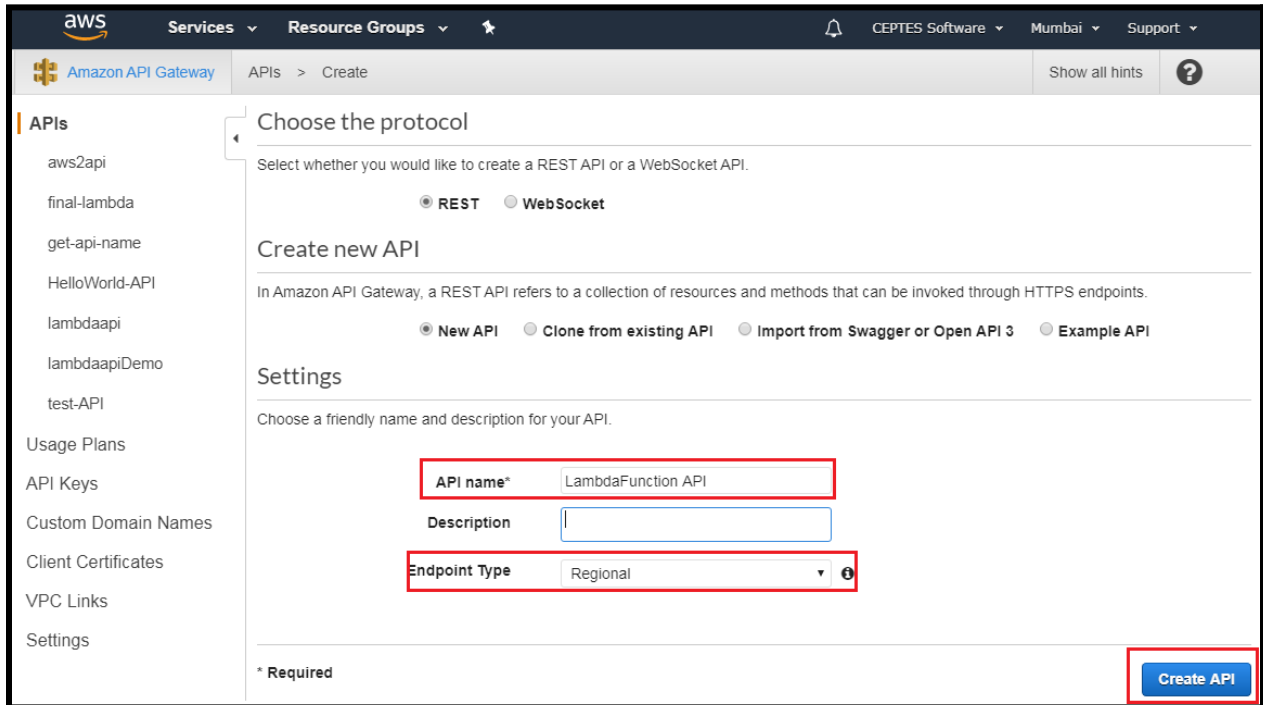
- Click on Create API



- Click on **Build**



- From the Settings section, please enter the following:
 - Provide API name (any)
 - Description (if required)
 - Endpoint Type as Regional
- Click on the **'Create API'** button.



Choose the protocol
Select whether you would like to create a REST API or a WebSocket API.

☒ REST ☐ WebSocket

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.

☒ New API ☐ Clone from existing API ☐ Import from Swagger or Open API 3 ☐ Example API

Settings
Choose a friendly name and description for your API.

API name*

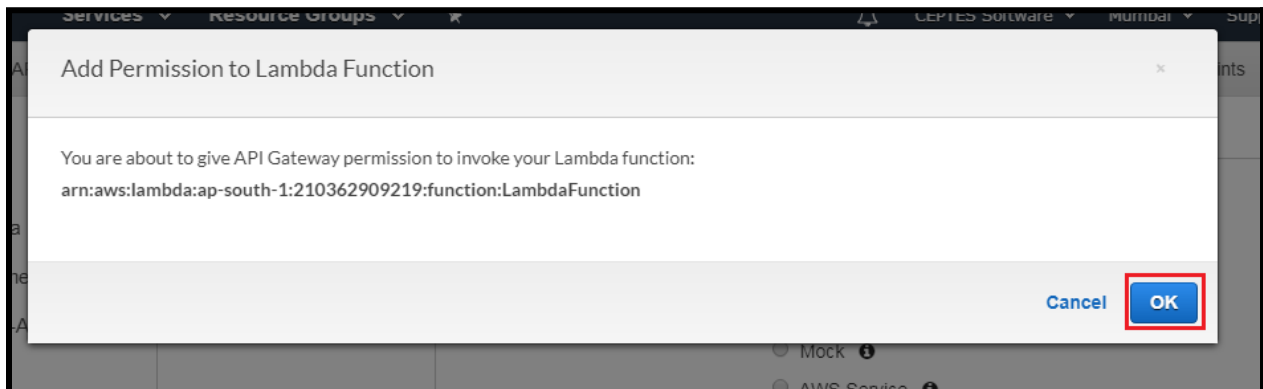
Description

Endpoint Type

* Required

Create API

- Click on the **'OK'** button.

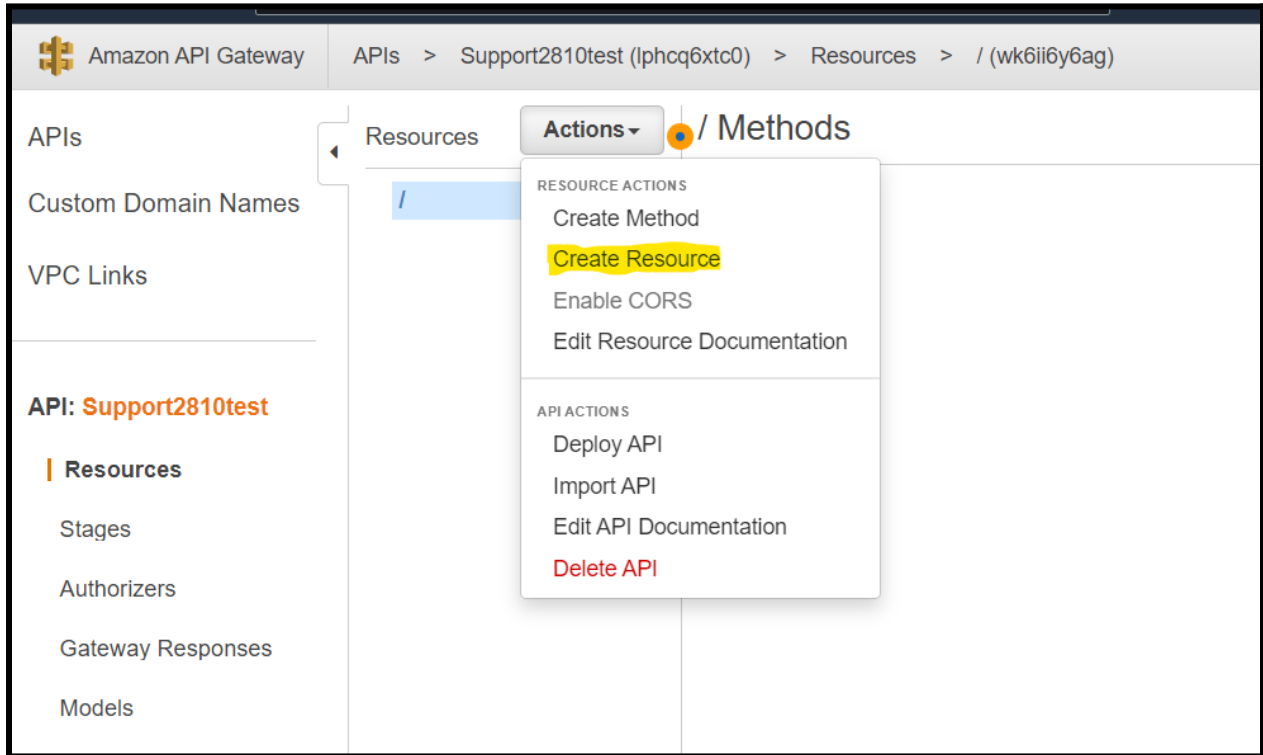


Add Permission to Lambda Function

You are about to give API Gateway permission to invoke your Lambda function:
arn:aws:lambda:ap-south-1:210362909219:function:LambdaFunction

Cancel **OK**


- The below screen appears. Now Click on the '**Action**' dropdown and select '**Create Resource**'



Now, Give the **Resource Name**, **Resource path** will be filled automatically.
Enable API Gateway CORS and Click on **Create Resources**.

New Child Resource


Use this page to create a new child resource for your resource.

Configure as ☒ proxy resource ☐ 

Resource Name*

Resource Path*

You can add path parameters using brackets. For example, the resource path `{username}` represents a path parameter called 'username'. Configuring `/[proxy+]` as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to `/foo`. To handle requests to `/`, add a new ANY method on the `/` resource.

Enable API Gateway CORS ☒ 

* Required

[Cancel](#) [Create Resource](#)

APIs

Custom Domain Names

VPC Links

API: **Support2810test**

Resources

Stages

Resources

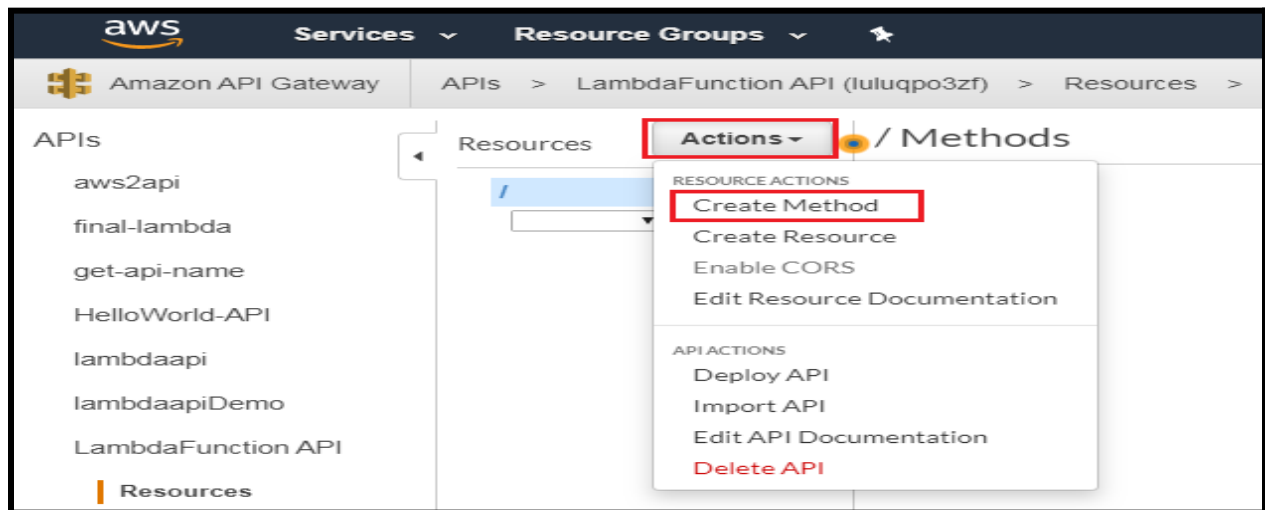
▼ /

/newtestacc

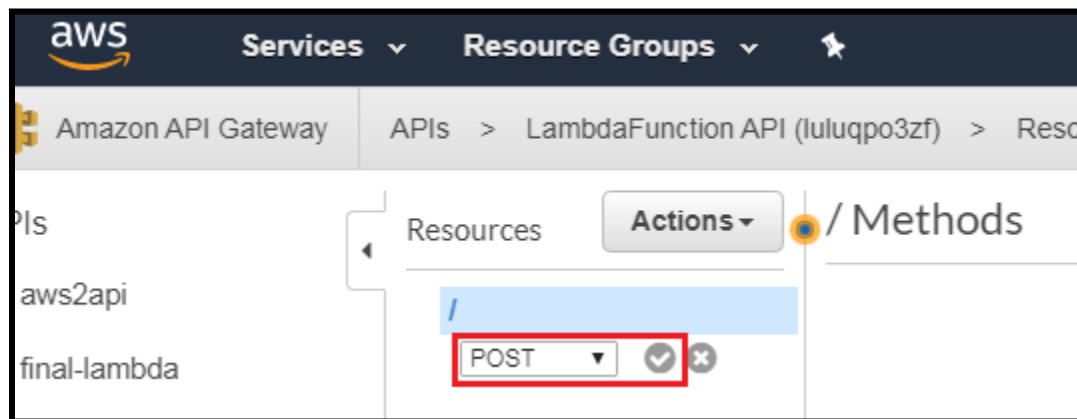
Actions ▼

/newtestacc Methods

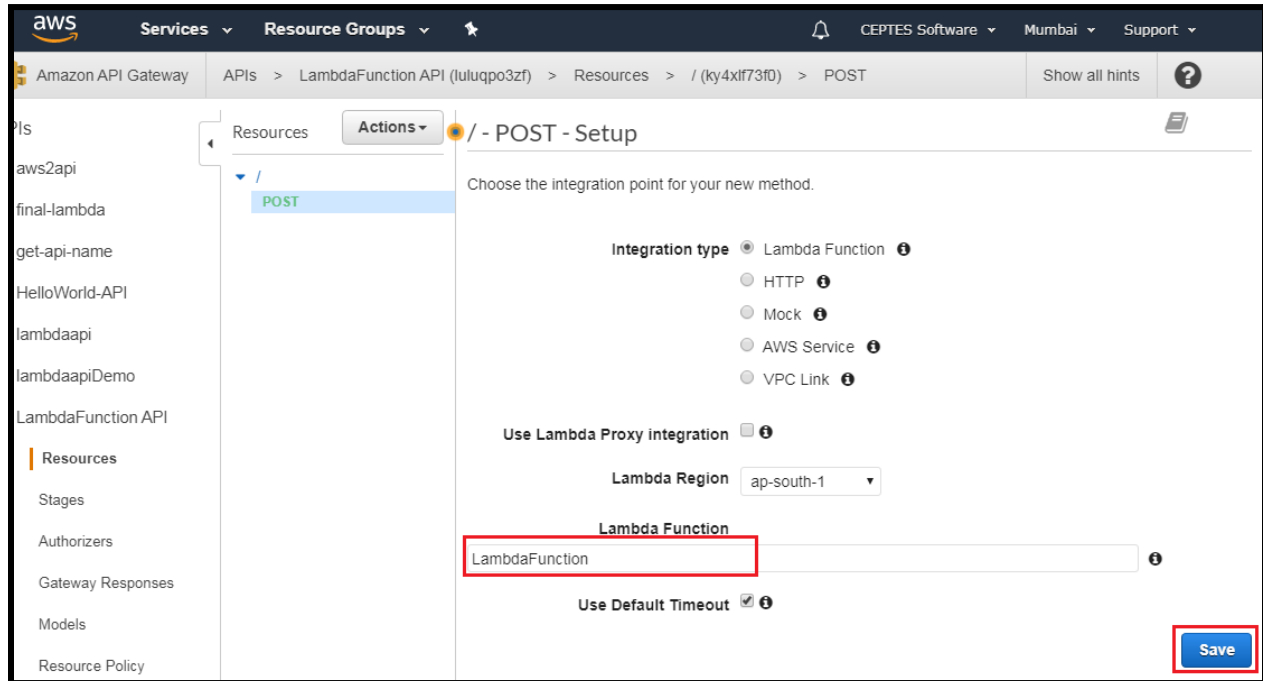
- The below screen appears. Now Click on the '**Action**' dropdown and select '**Create Method**'



- Now select '**Post**' from the dropdown and click on '**Right Mark**'



- The below screen appears. Now give the '**Lambda Function**' name (previously created Lambda Function)
And Click on '**Save**' button.



aws Services Resource Groups

Amazon API Gateway APIs > LambdaFunction API (luluqpo3zf) > Resources > / (ky4xlf73f0) > POST Show all hints ?

Resources Actions

POST

Choose the integration point for your new method.

Integration type ☒ Lambda Function ⓘ

☐ HTTP ⓘ

☐ Mock ⓘ

☐ AWS Service ⓘ

☐ VPC Link ⓘ

Use Lambda Proxy integration ☐ ⓘ

Lambda Region ap-south-1

Lambda Function

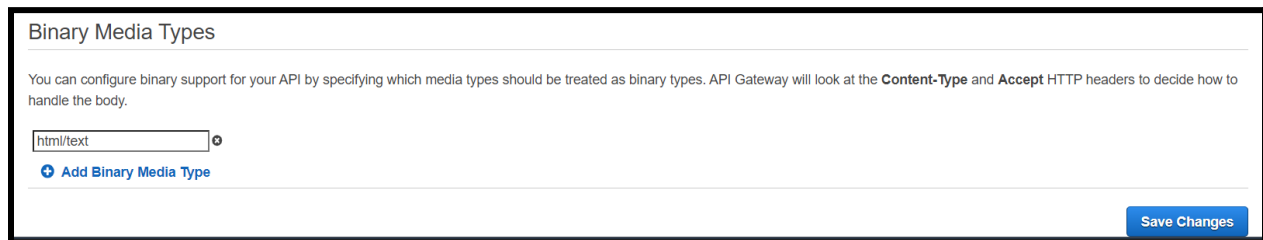
LambdaFunction ⓘ

Use Default Timeout ☒ ⓘ

Save

Encode the parameters from Gateway (optional)

- Click on the **Settings** tab from the left panel.
- Scroll down to **Binary Media Types** section and add **html/text**
- Click on **Save Changes**



Binary Media Types

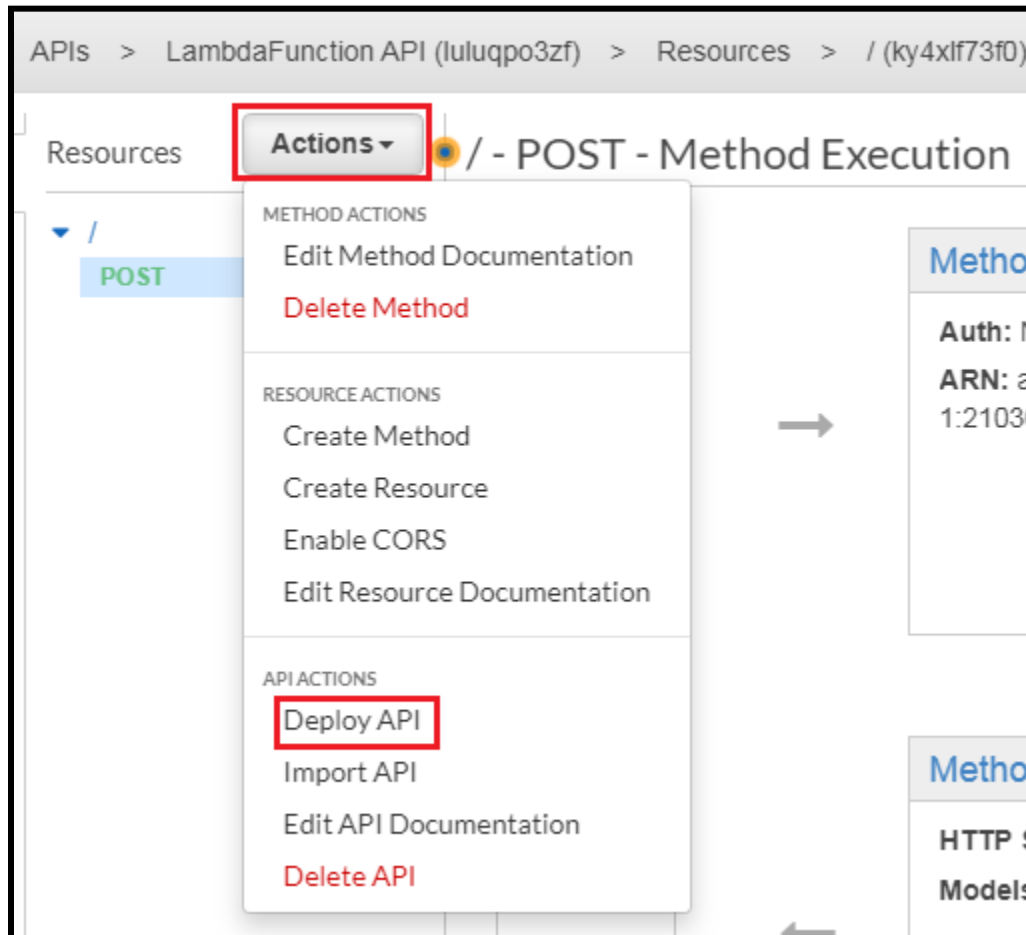
You can configure binary support for your API by specifying which media types should be treated as binary types. API Gateway will look at the **Content-Type** and **Accept** HTTP headers to decide how to handle the body.

html/text ⓘ

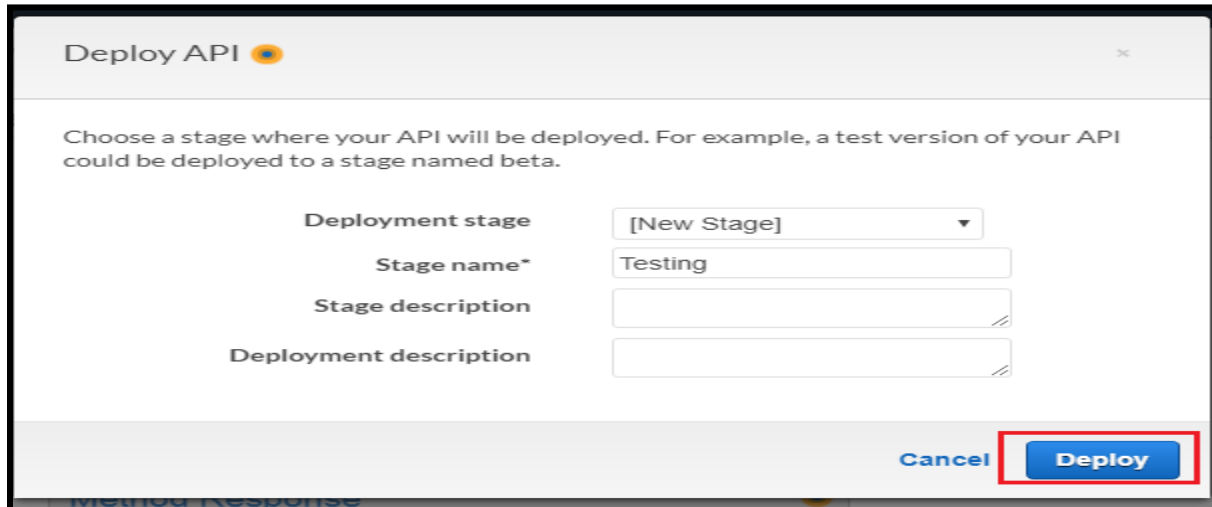
+ Add Binary Media Type

Save Changes

- The below screen appears. Click on the **'Actions'** dropdown and Click on **'Deploy API'**



- Provide the Deployment Stage as '**New Stage**' and Provide the '**Stage name**' and Click on the '**Deploy**' button



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] ▼

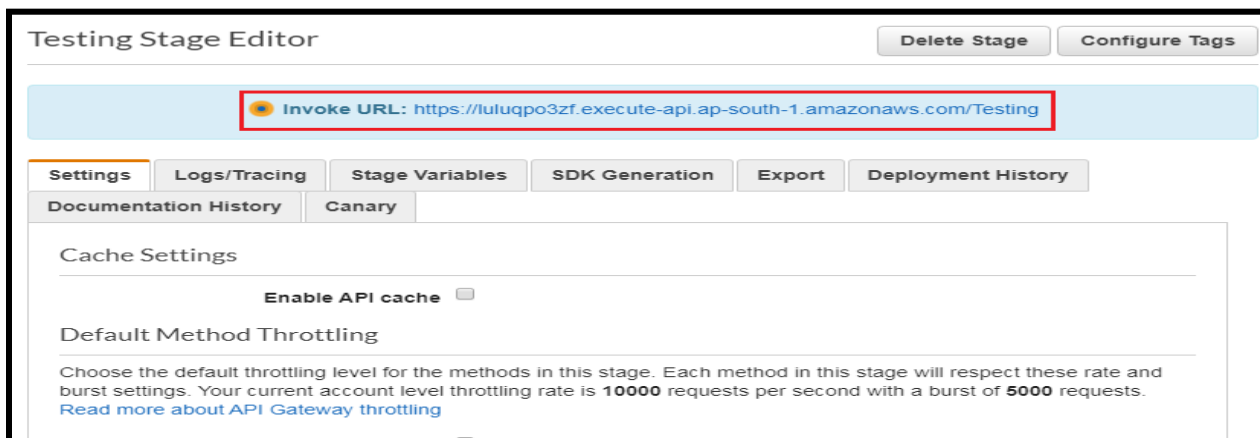
Stage name*: Testing

Stage description:

Deployment description:

Cancel Deploy

- The below page appears with the Invoke URL and keep this url (this is required to be added in the Salesforce's custom settings)



Testing Stage Editor

Delete Stage Configure Tags

Invoke URL: <https://luluqpo3zf.execute-api.ap-south-1.amazonaws.com/Testing>

Settings Logs/Tracing Stage Variables SDK Generation Export Deployment History

Documentation History Canary

Cache Settings

Enable API cache ☐

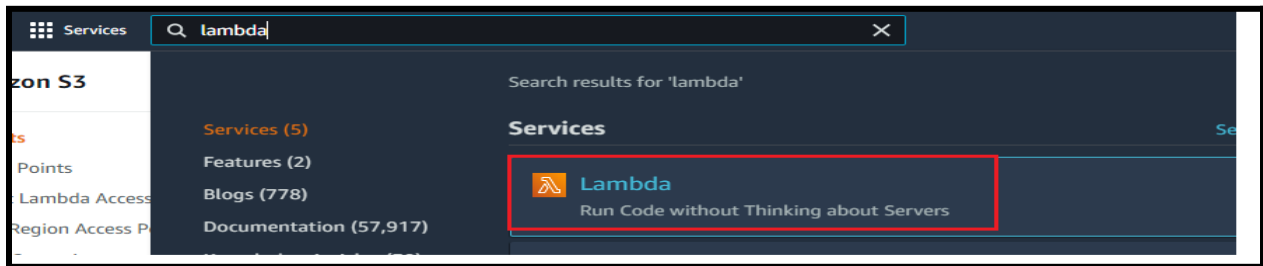
Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. [Read more about API Gateway throttling](#)

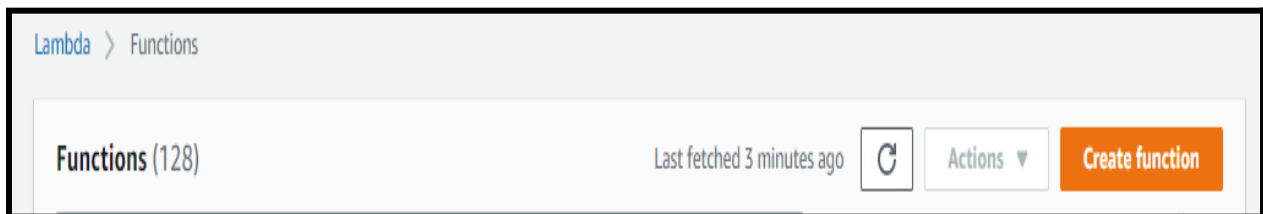
Add Authorizer

Create Authorizer Lambda Function

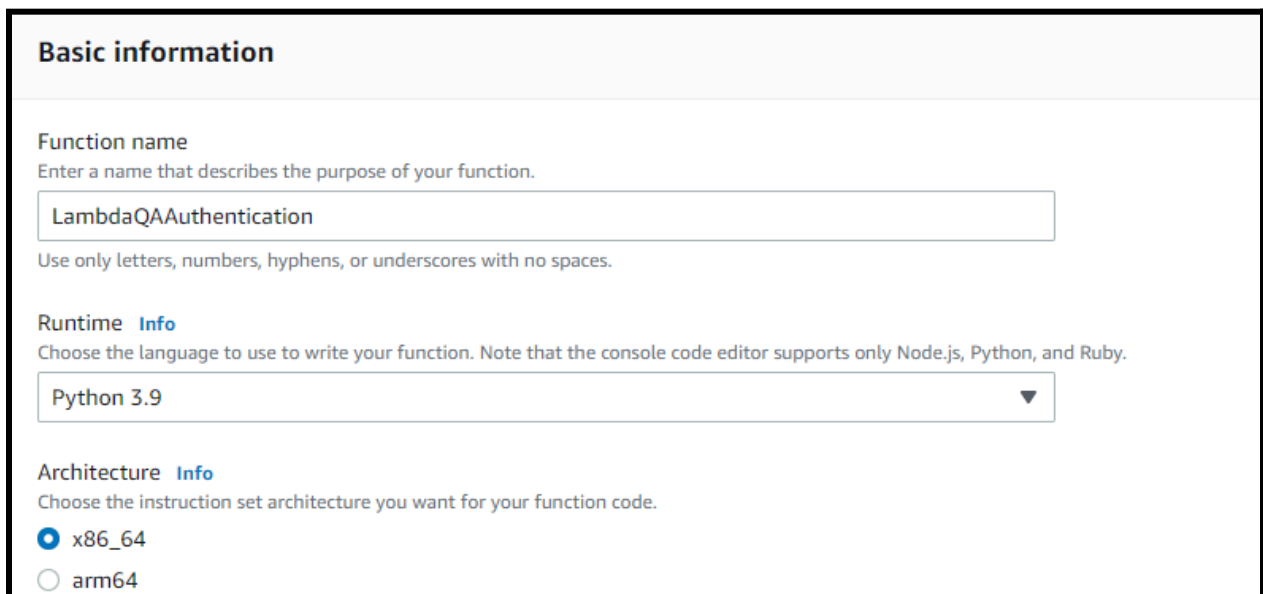
- Go to AWS Console. Click on Services, Search for Lambda, and open Lambda service.



- Click on the “Create Function” button



- Provide the Basic Informations**
 - Function Name** - Provide any name
 - Runtime** - Select **Python 3.9** from the dropdown And click on the **Create Function** button.



The screenshot shows the 'Basic information' form for creating a new Lambda function. The form has the following fields:

- Function name:** A text input field containing 'LambdaQAAuthentication'. Below the field, a note states: 'Enter a name that describes the purpose of your function. Use only letters, numbers, hyphens, or underscores with no spaces.'
- Runtime:** A dropdown menu showing 'Python 3.9'. Above the dropdown, there is an 'Info' link and a note: 'Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.'
- Architecture:** Radio buttons for 'x86_64' (selected) and 'arm64'. Above the buttons, there is an 'Info' link and a note: 'Choose the instruction set architecture you want for your function code.'

- Add the below python code snippet and make a note of the `authorizationToken` (in the below code it is 'abc123'), this **authorization token** should be added in the Salesforce.

```
import json

def lambda_handler(event, context):
    auth = 'Deny'

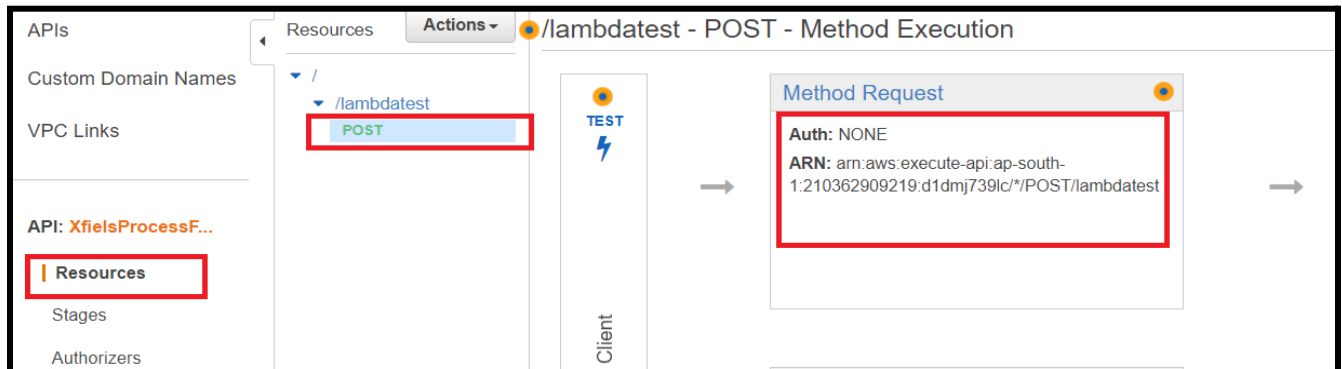
    if event['authorizationToken'] == abc123:
        auth = 'Allow'
    else:
        auth = 'Deny'

    authResponse = { "policyDocument": { "Version": "2012-10-17", "Statement":
[{"Action": "execute-api:Invoke", "Resource":
["arn:aws:execute-api:ap-south-1:210362909219:y0p9k245f8/*/*"], "Effect":
auth}] } }
    return authResponse
```

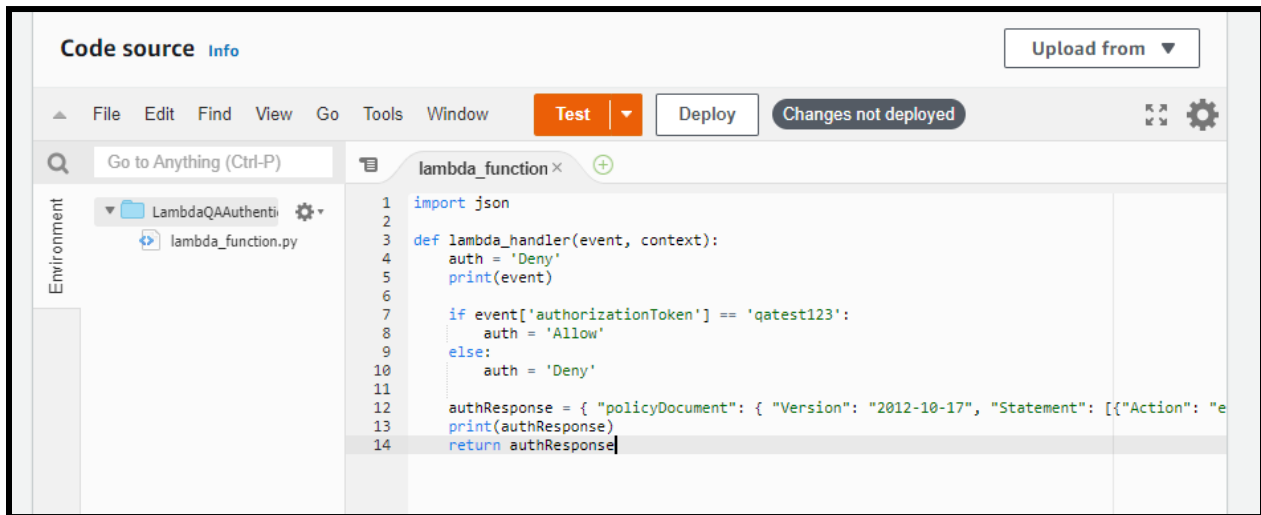
Update above python code with your lambda function's ARN

Get the ARN from the earlier created API Gateway as shown in the below screenshot.

Example: "arn:aws:execute-api:ap-south-1:210362909219:gz8ek9a24c"

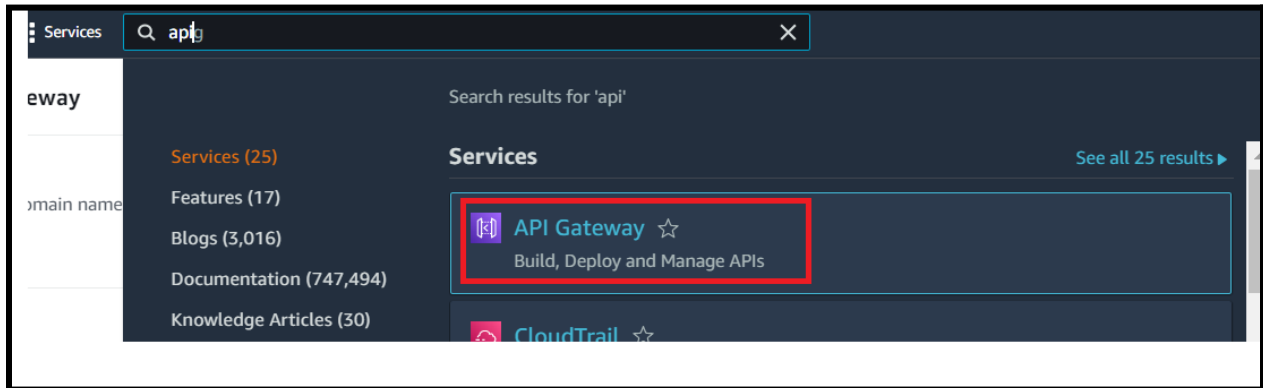


- Click On the **Deploy** button

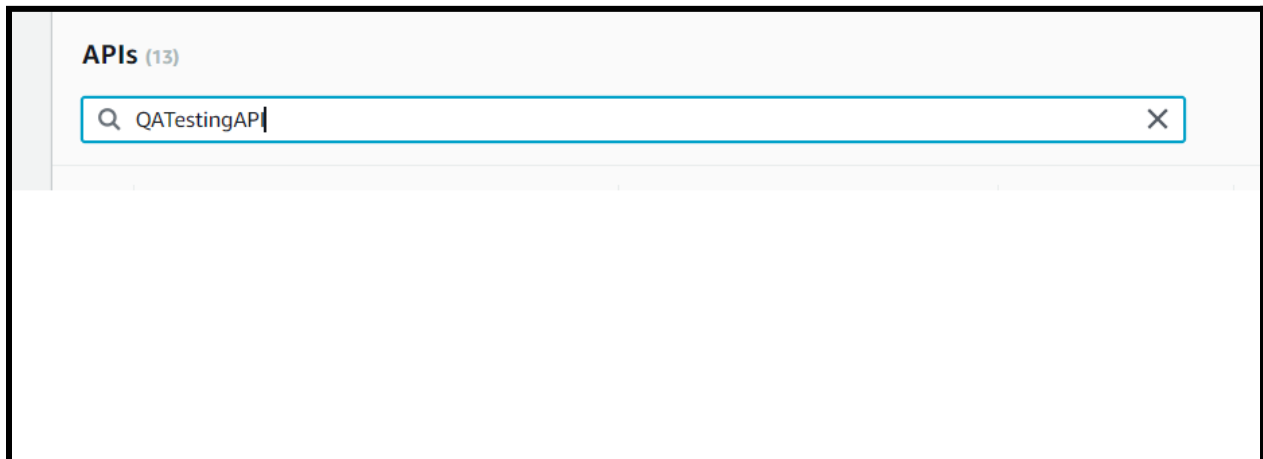


Create Authorizer for Gateway API

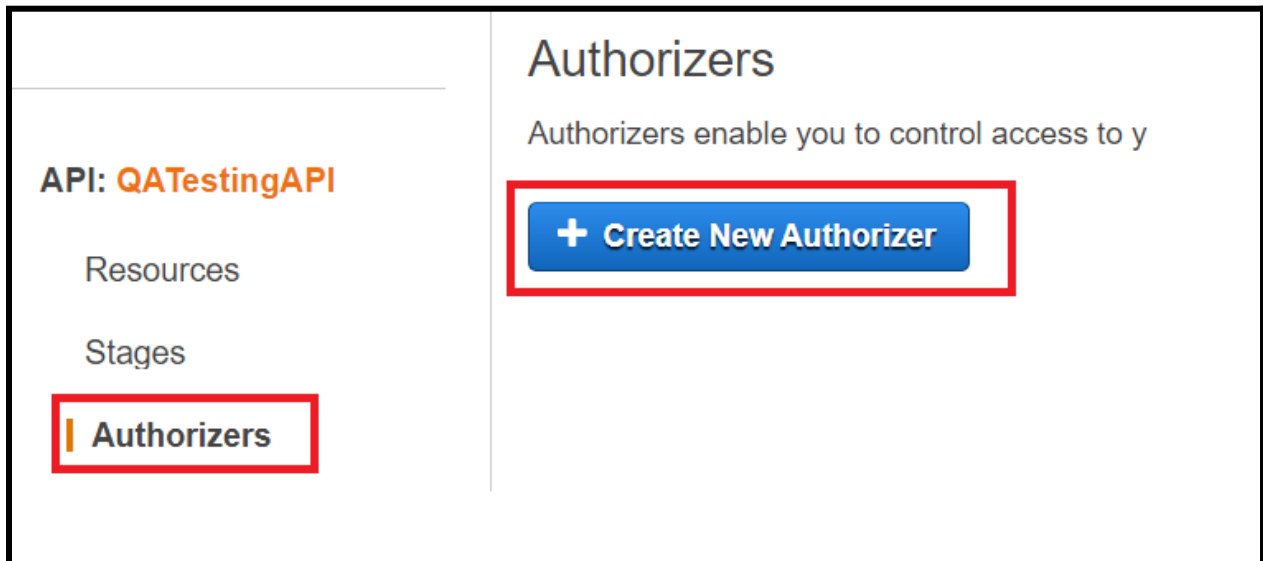
- Search for **API Gateway** and open it.



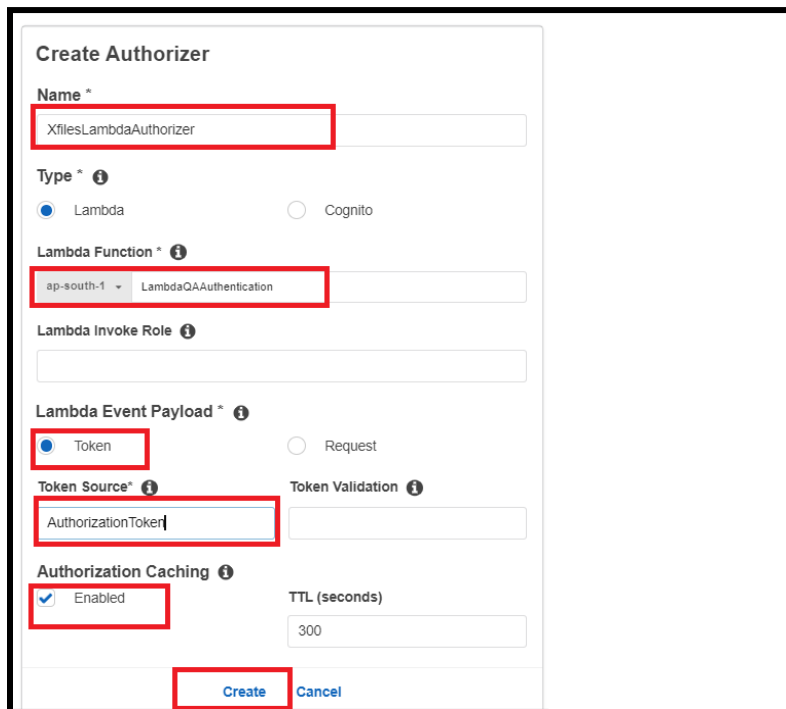
- Open the Gateway API which was created earlier.



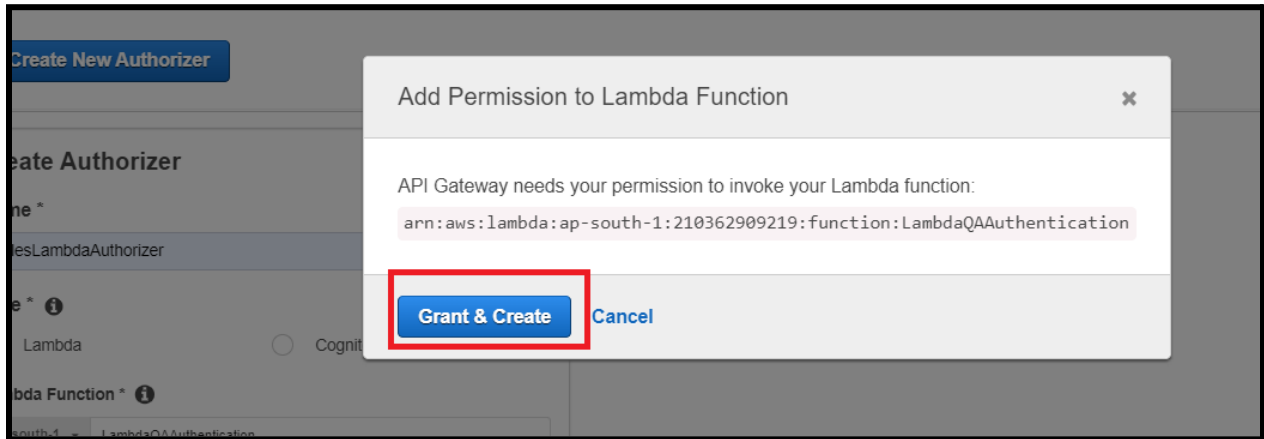
- Click on **Authorizers** and select **Create New Authorization**



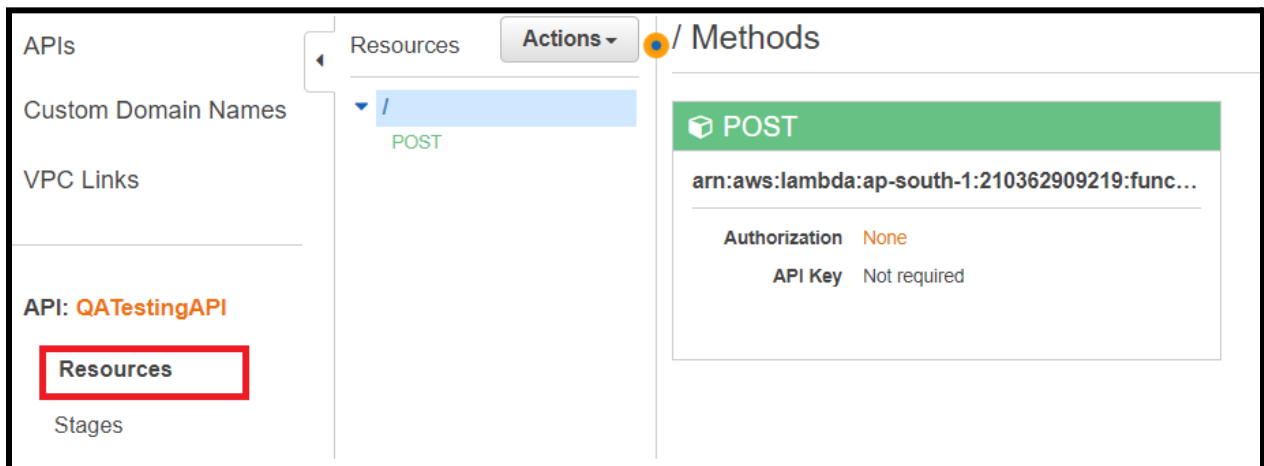
- Provide the Information:
 - **Function Name** - provide any name
 - **Type**: Lambda
 - **Lambda Function Name**: Select Authorizer Lambda Function (second lambda function having authorization python code).
 - **Token Source**: AuthorizationToken
- Click on the **Create** link.



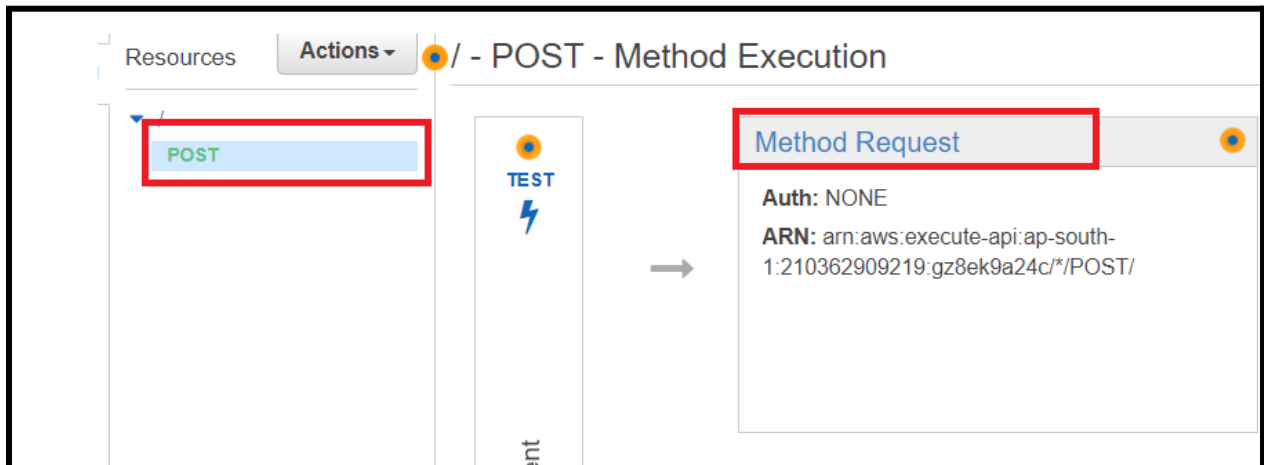
- Click on **Grant & Create**.



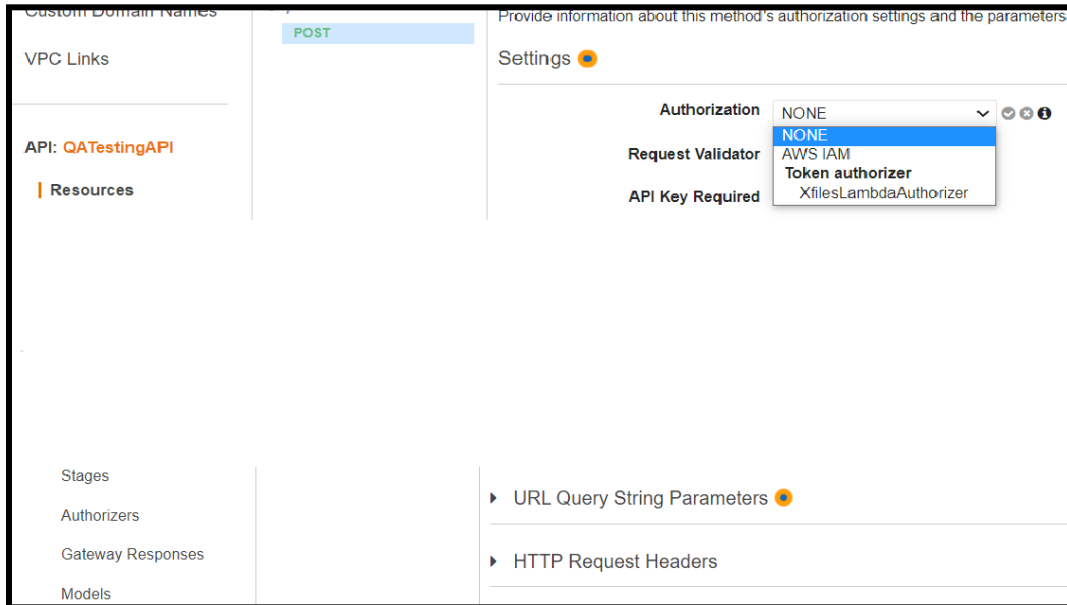
- Then click on **Resources**.



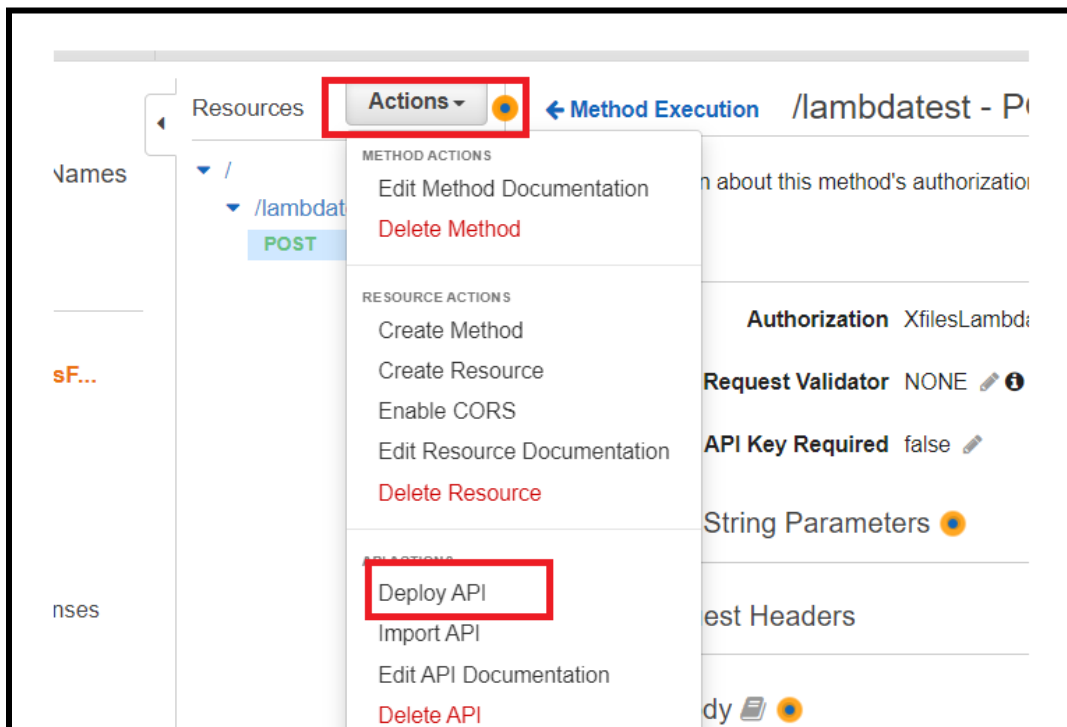
- Click on Post and then click on **Method Request**.



- In the **Authorization** field, select the authorizer Lambda function and click on the right mark.

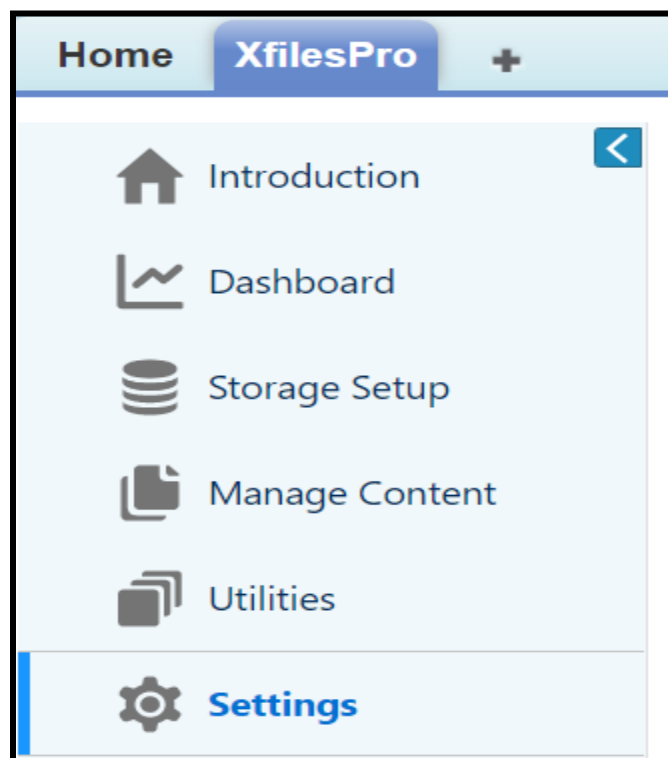


- Then click on the Action button select **Deploy API**

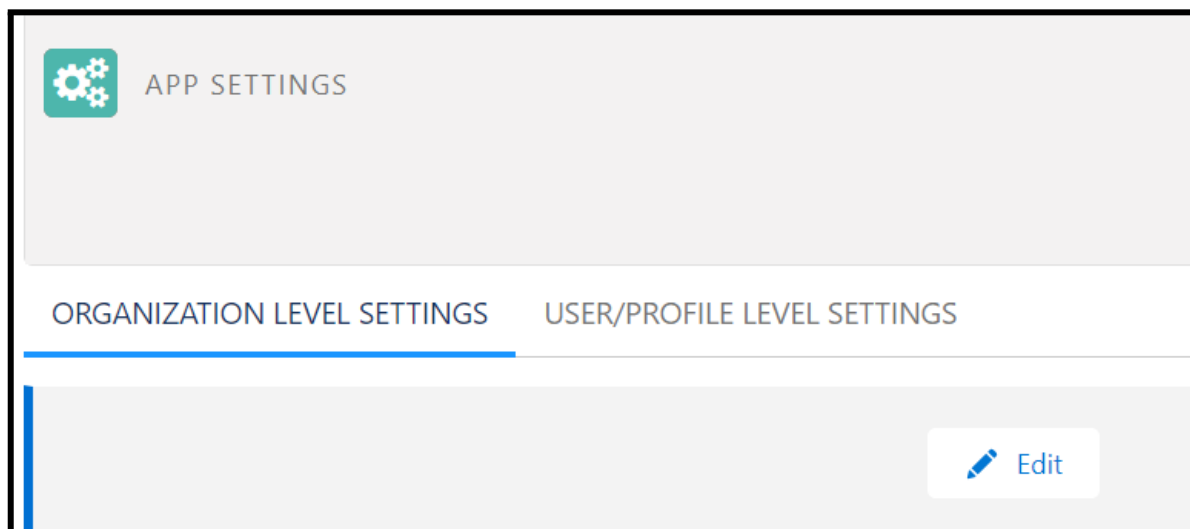


Add the Gateway API invoke URL and Authorization Token

1. Go to **XfilesPro Setting**



2. Click On **Edit**



3. In the **Lambda URL** field, paste the Gateway API invoke URL (copied from AWS gateway)

Lambda URL	<input type="text" value="https://f6jgt6xm9a.execute-api.ap"/>
------------	--

4. In the **AWS Authorization Token** field, enter the Authorization Token (Which was given in python code)

AWS Authorization Token	<input type="text"/>
-------------------------	----------------------

Note: For security reasons, the Authorization Token will be saved in the encrypted form.

5. Click On **Save**

Remote Site Settings

1. Go to Salesforce setup.
2. Search for Remote Site and open **Remote Site Settings**.
3. Click on **New Remote Site**.
4. Provide the Information:
 - **Remote Site Name:** provide any name.
 - **Remote Site URL:** Add Gateway API invoke URL (from AWS API Gateway).

Remote Site Edit

[Help for this P](#)

Enter the URL for the remote site. All s-controls, JavaScript OnClick commands in custom buttons, Apex, and AJAX proxy calls can access this Web address from salesforce.com.

Remote Site Edit
Save Save & New Cancel

Remote Site Name

Remote Site URL

Disable Protocol Security ☐ i

Description

CSP Trusted Site

5. Go to Salesforce setup.
6. Search for CSP and open **CSP Trusted Site**.
7. Click on **New Trusted Site**.
8. Provide the Information:
 - **Trusted Site Name:** provide any name.'
 - **Trusted Site URL:** Add Gateway API invoke URL (from aws API Gateway).

Content Security Policy Trusted Site Edit

Save Save & New Cancel

General Information

Trusted Site Name

Trusted Site URL

Description

Active ☒

Context All