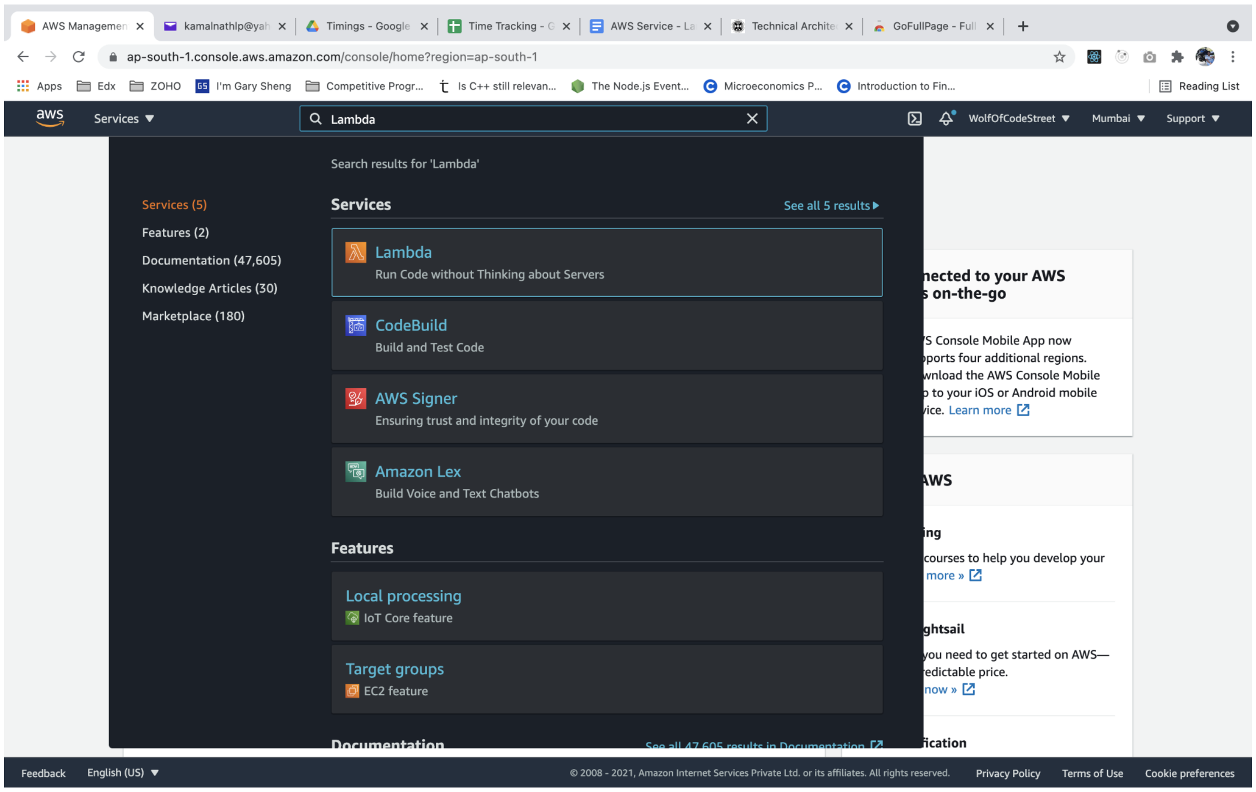
Guide to implement AWS Lambda function with API Gateway:

1. Create an AWS Account
2. On the search bar, enter the value as follows: “Lambda”. Following results will be listed,



ap-south-1.console.aws.amazon.com/console/home?region=ap-south-1

Search results for 'Lambda'

Services

Lambda
Run Code without Thinking about Servers

CodeBuild
Build and Test Code

AWS Signer
Ensuring trust and integrity of your code

Amazon Lex
Build Voice and Text Chatbots

Features

Local processing
IoT Core feature

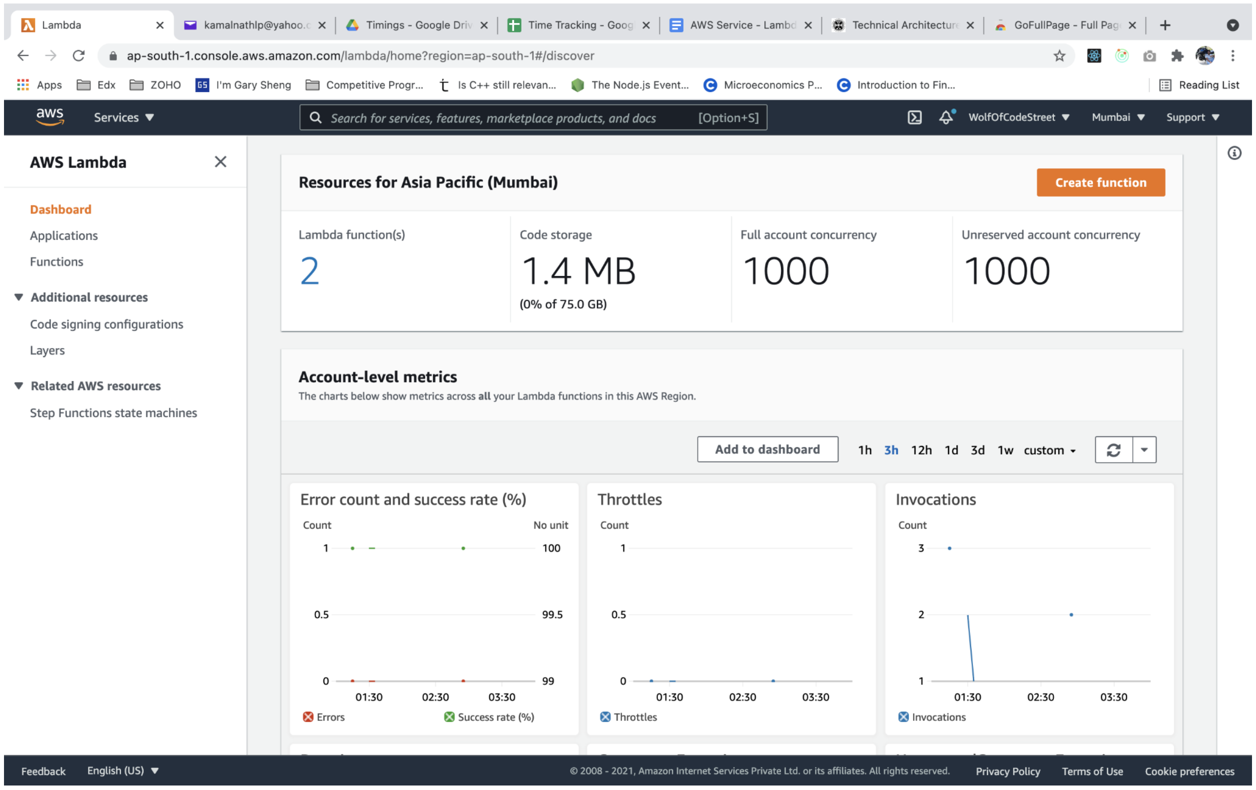
Target groups
EC2 feature

Documentation

See all 47,605 results in Documentation

© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

   choose the Lambda service.
3. After clicking on the lambda service, you will be redirected to the Lambda service dashboard page. Over there choose the “Function” tab in the sidebar.


Lambda | kamalnathlp@yahoo.c | Timings - Google Driv | Time Tracking - Goog | AWS Service - Lambd | Technical Architectur | GoFullPage - Full Pag

ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/discover

Apps | Edx | ZOHO | I'm Gary Sheng | Competitive Progr... | Is C++ still relevan... | The Node.js Event... | Microeconomics P... | Introduction to Fin... | Reading List

aws | Services ▼ | Search for services, features, marketplace products, and docs [Option+S] | WolfOfCodeStreet ▼ | Mumbai ▼ | Support ▼

AWS Lambda

- Dashboard
- Applications
- Functions

▼ Additional resources
- Code signing configurations
- Layers

▼ Related AWS resources
- Step Functions state machines

Resources for Asia Pacific (Mumbai) — Create function

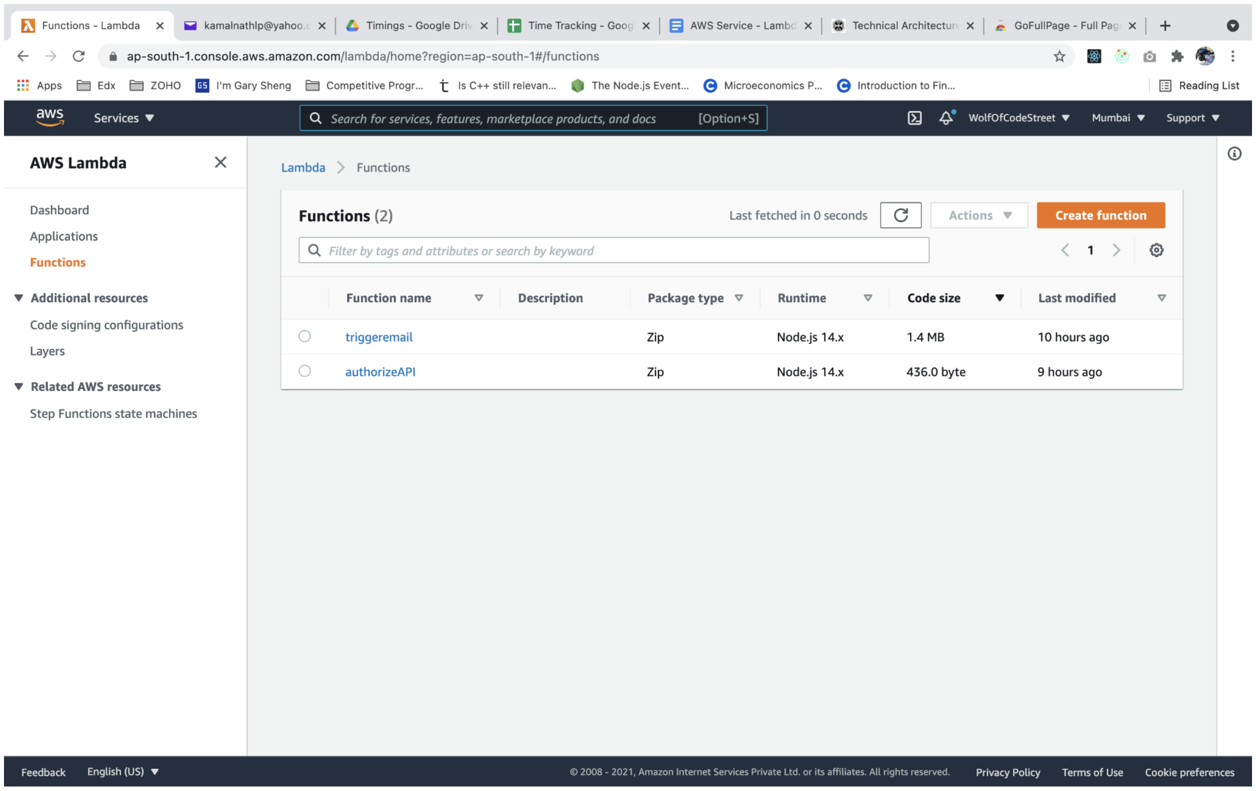
Lambda function(s)	Code storage	Full account concurrency	Unreserved account concurrency
2	1.4 MB (0% of 75.0 GB)	1000	1000

Account-level metrics

The charts below show metrics across **all** your Lambda functions in this AWS Region.

Add to dashboard | 1h | 3h | 12h | 1d | 3d | 1w | custom ▾

Feedback | English (US) ▼ | © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use | Cookie preferences


Functions - Lambda | kamalnathlp@yahoo.c | Timings - Google Driv | Time Tracking - Goog | AWS Service - Lambd | Technical Architectur | GoFullPage - Full Pag

ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions

Apps | Edx | ZOHO | I'm Gary Sheng | Competitive Progr... | Is C++ still relevan... | The Node.js Event... | Microeconomics P... | Introduction to Fin... | Reading List

aws | Services ▼ | Search for services, features, marketplace products, and docs [Option+S] | WolfOfCodeStreet ▼ | Mumbai ▼ | Support ▼

AWS Lambda

- Dashboard
- Applications
- Functions

▼ Additional resources
- Code signing configurations
- Layers

▼ Related AWS resources
- Step Functions state machines

Lambda > Functions

Functions (2) — Last fetched in 0 seconds | Actions ▼ | Create function

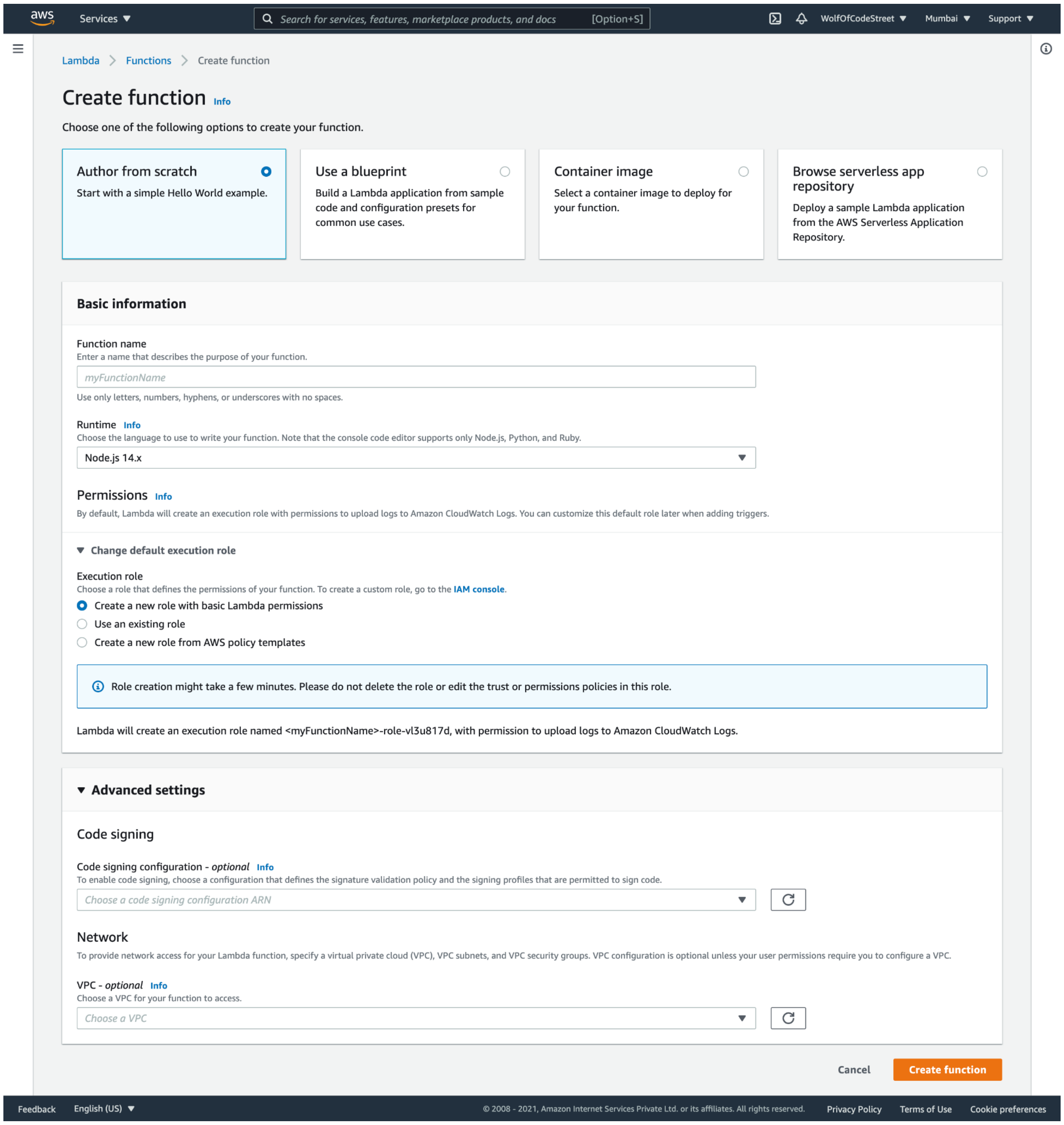
Filter by tags and attributes or search by keyword — < 1 >

	Function name	Description	Package type	Runtime	Code size	Last modified
○	triggeremail		Zip	Node.js 14.x	1.4 MB	10 hours ago
○	authorizeAPI		Zip	Node.js 14.x	436.0 byte	9 hours ago

Feedback | English (US) ▼ | © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use | Cookie preferences

After choosing the Function tab, click on the “Create Function” button.

4. Create Function Page will look like the following ,


aws
Services
Search for services, features, marketplace products, and docs [Option+S]
WolfOfCodeStreet
Mumbai
Support
Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

- [✓ Author from scratch] Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image: Select a container image to deploy for your function.
- Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
myFunctionName
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 14.x

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
- [✓ Create a new role with basic Lambda permissions]
- Use an existing role
- Create a new role from AWS policy templates

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named <myFunctionName>-role-vl3u817d, with permission to upload logs to Amazon CloudWatch Logs.

Advanced settings

Code signing

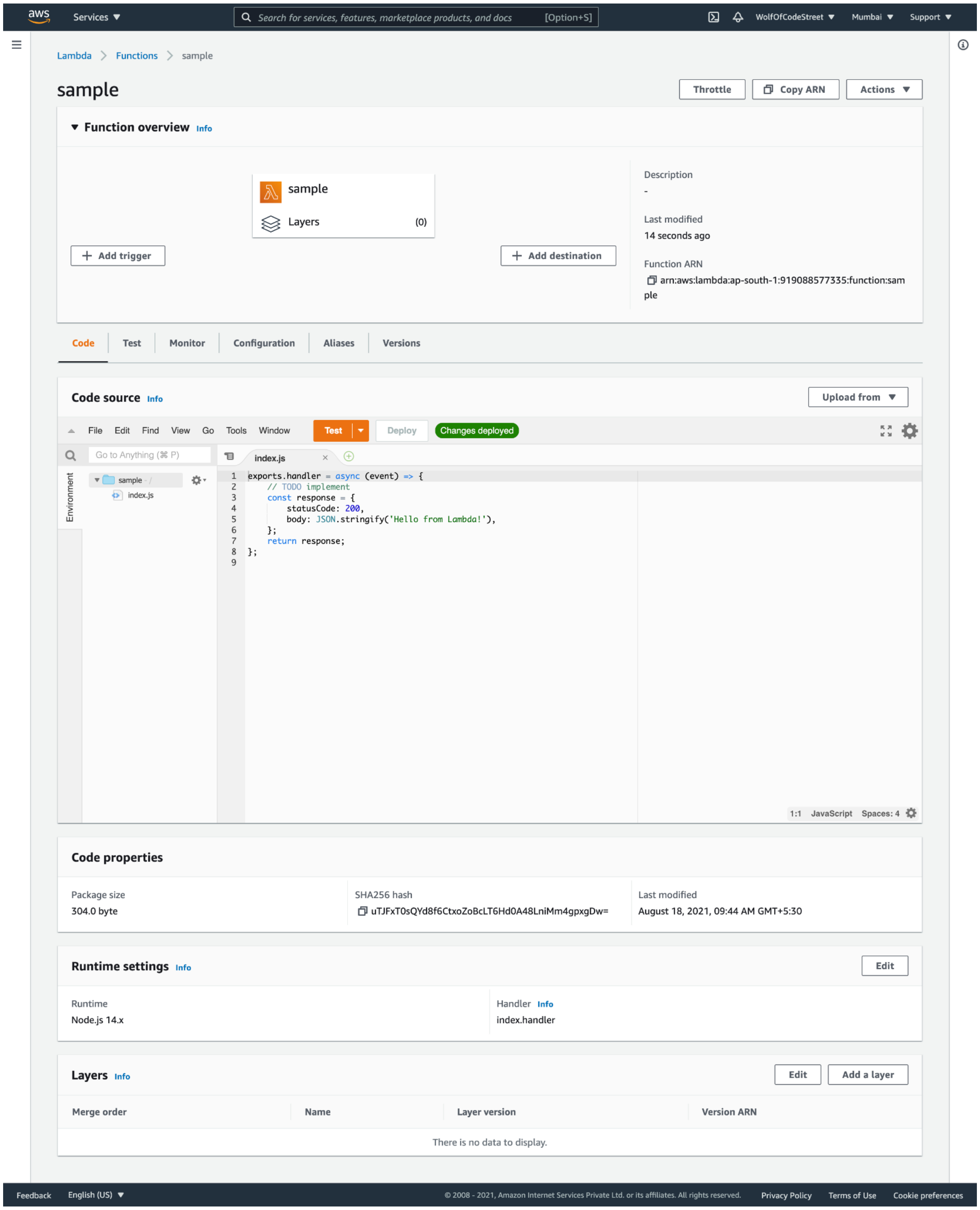
Code signing configuration - *optional* Info
To enable code signing, choose a configuration that defines the signature validation policy and the signing profiles that are permitted to sign code.
Choose a code signing configuration ARN

Network
To provide network access for your Lambda function, specify a virtual private cloud (VPC), VPC subnets, and VPC security groups. VPC configuration is optional unless your user permissions require you to configure a VPC.

VPC - *optional* Info
Choose a VPC for your function to access.
Choose a VPC

Cancel
Create function
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

   a. Choose “Author from scratch” type
   b. In the Basic Information accordion, enter the following values
      i. Function Name - triggeremail
      ii. Runtime - Node.js 14.x
      iii. Execution role - Create a new role with basic lambda permissions
5. After entering the values, click on the create function button.
6. Next step, page will be redirected to the function page and page will look likes following (sample),


aws
Services
Search for services, features, marketplace products, and docs [Option+S]
WolfOfCodeStreet
Mumbai
Support
Lambda > Functions > sample

sample

Function overview Info

Description
-

Last modified
14 seconds ago

Function ARN
arn:aws:lambda:ap-south-1:919088577335:function:sample

Code source Info

```
exports.handler = async (event) => {
    // TODO implement
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

Code properties

Package size	SHA256 hash	Last modified
304.0 byte	uTJFxT0sQYd8f6CtxoZoBcLT6Hd0A48LniMm4gpxgDw=	August 18, 2021, 09:44 AM GMT+5:30

Runtime settings Info

Runtime	Handler Info
Node.js 14.x	index.handler

Layers Info

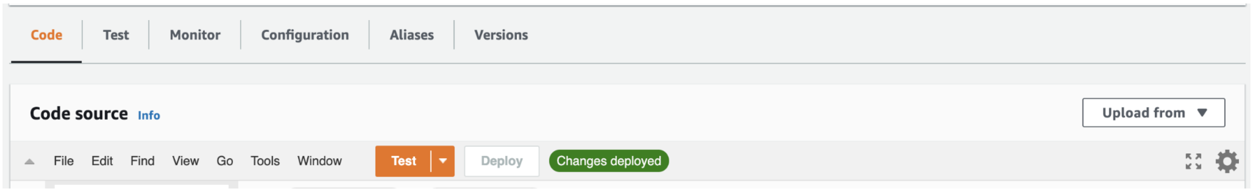
Merge order	Name	Layer version	Version ARN

There is no data to display.

Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Download the zip file "triggeremail-f5d83dad-eab8-4299-9166-cdb7e12a9105.zip" (You will find the file in the directory which is shared with you).

7. Upload the downloaded file by choosing the option “upload from”,

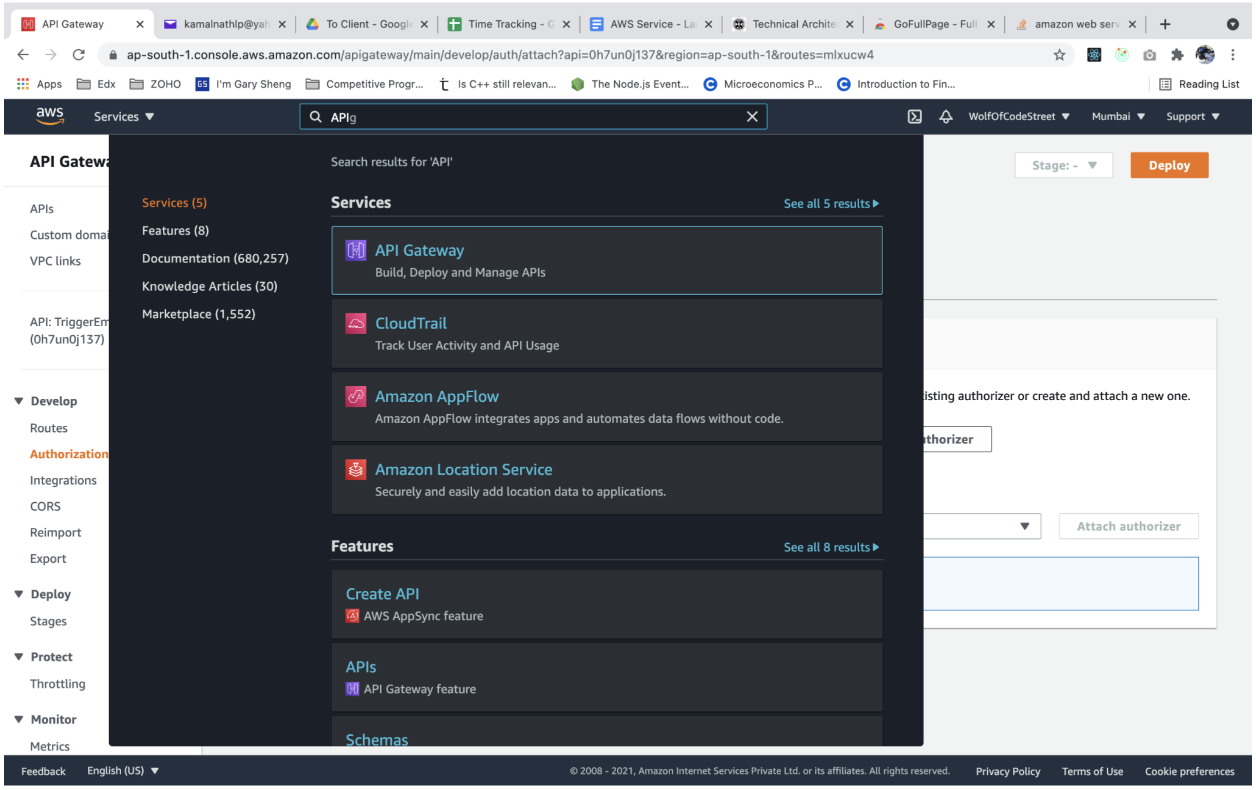

Code | Test | Monitor | Configuration | Aliases | Versions

Code source Info — Upload from ▼

File Edit Find View Go Tools Window — Test ▼ — Deploy — Changes deployed

After uploading, click on the “Deploy” button next to the “Test” button (if deploy button is enabled).

8. After creating the aws lambda function, we need to create an “API gateway” to create the URL routing to access the lambda function.
9. In the search bar, search for the following “API Gateway”,


ap-south-1.console.aws.amazon.com/apigateway/main/develop/auth/attach?api=0h7un0j137&region=ap-south-1&routes=mlxucw4

Search: APIg

Search results for 'API'

Services (5), Features (8), Documentation (680,257), Knowledge Articles (30), Marketplace (1,552)

Services — See all 5 results

API Gateway — Build, Deploy and Manage APIs

CloudTrail — Track User Activity and API Usage

Amazon AppFlow — Amazon AppFlow integrates apps and automates data flows without code.

Amazon Location Service — Securely and easily add location data to applications.

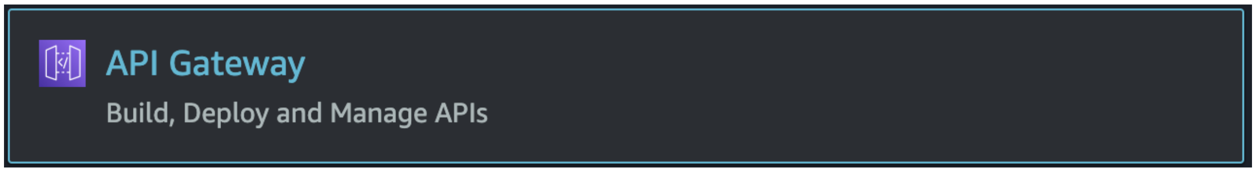
Features — See all 8 results

Create API — AWS AppSync feature

APIs — API Gateway feature

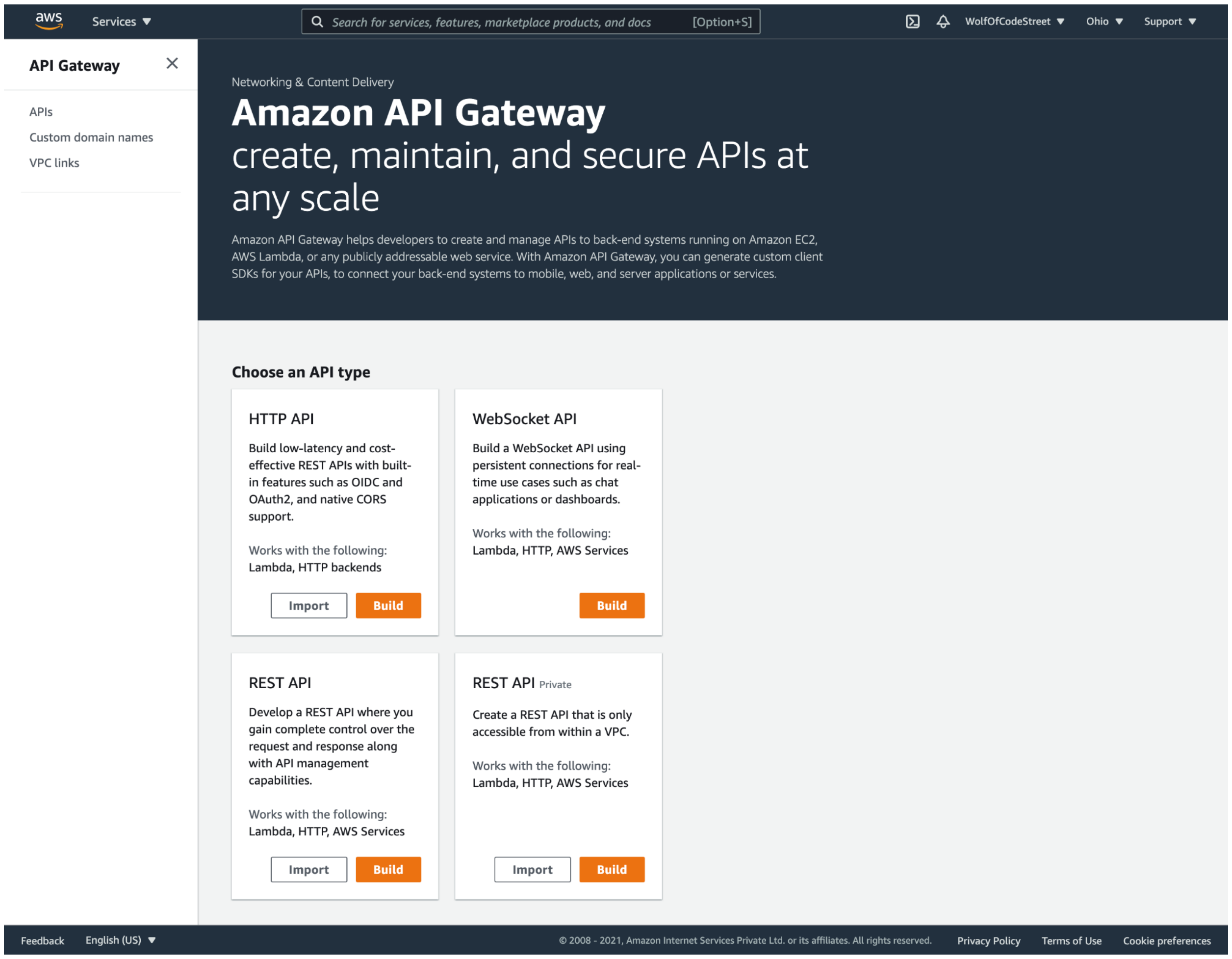
Schemas

Choose the first one named as “API Gateway”.


API Gateway

Build, Deploy and Manage APIs

10. Once you click on the service, it will be redirected to the APi Gateway Page



Networking & Content Delivery

Amazon API Gateway

create, maintain, and secure APIs at any scale

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.

Choose an API type

HTTP API

Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

Works with the following:
Lambda, HTTP backends

WebSocket API

Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.

Works with the following:
Lambda, HTTP, AWS Services

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:
Lambda, HTTP, AWS Services

REST API Private

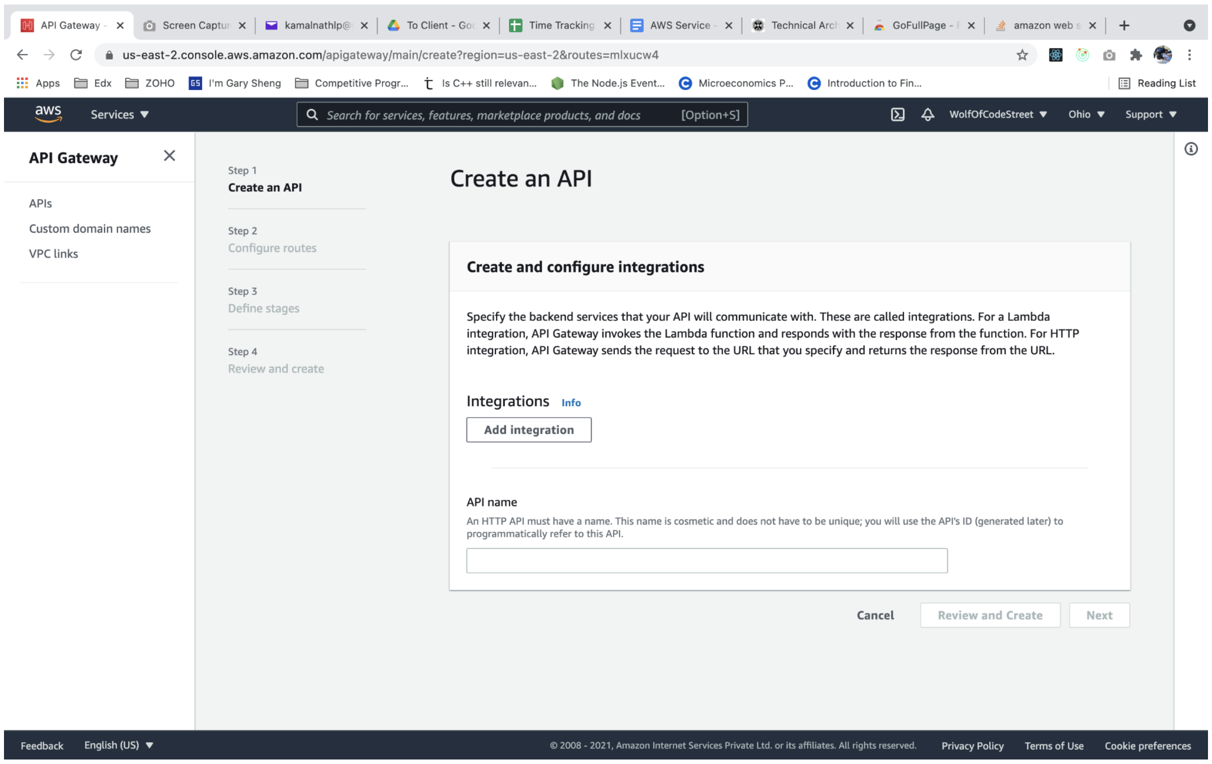
Create a REST API that is only accessible from within a VPC.

Works with the following:
Lambda, HTTP, AWS Services

© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

a. Choose the Build in “HTTP API”

b. There are four steps to create the API Route,



us-east-2.console.aws.amazon.com/apigateway/main/create?region=us-east-2&routes=mlxucw4

Create an API

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

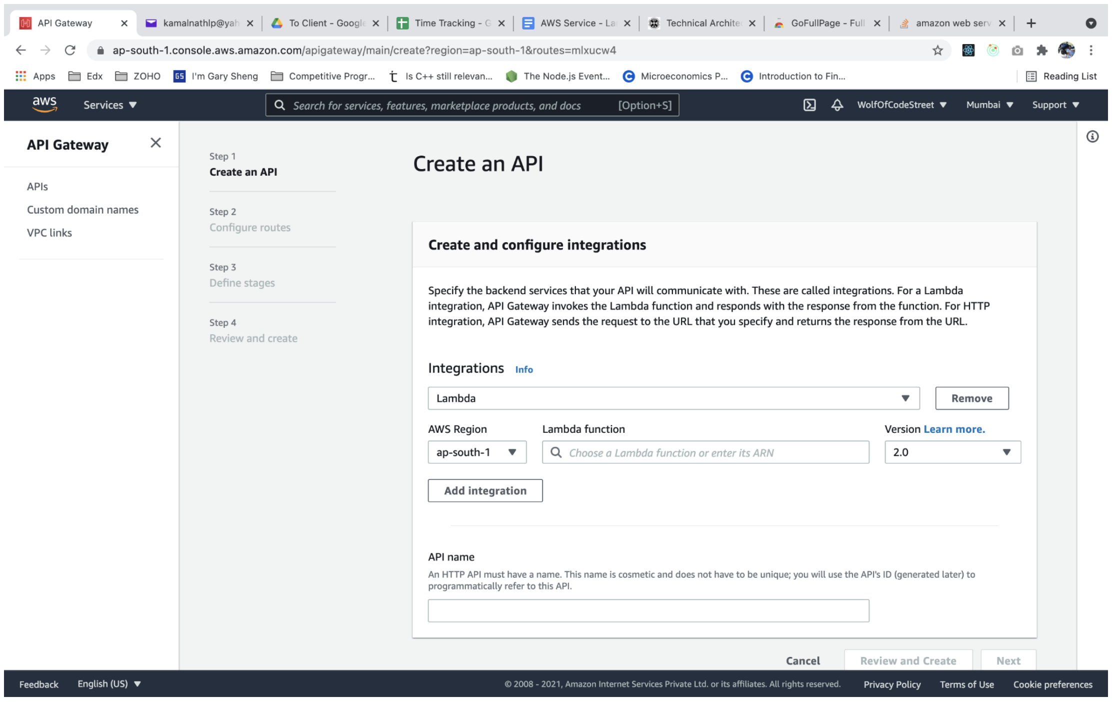
Integrations Info

API name

An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

i. Click on the “Add Integration”, in the dropdown choose Lambda function.



ap-south-1.console.aws.amazon.com/apigateway/main/create?region=ap-south-1&routes=mlxucw4

Create an API

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations Info

Lambda

AWS Region: ap-south-1

Lambda function

Version Learn more.: 2.0

API name

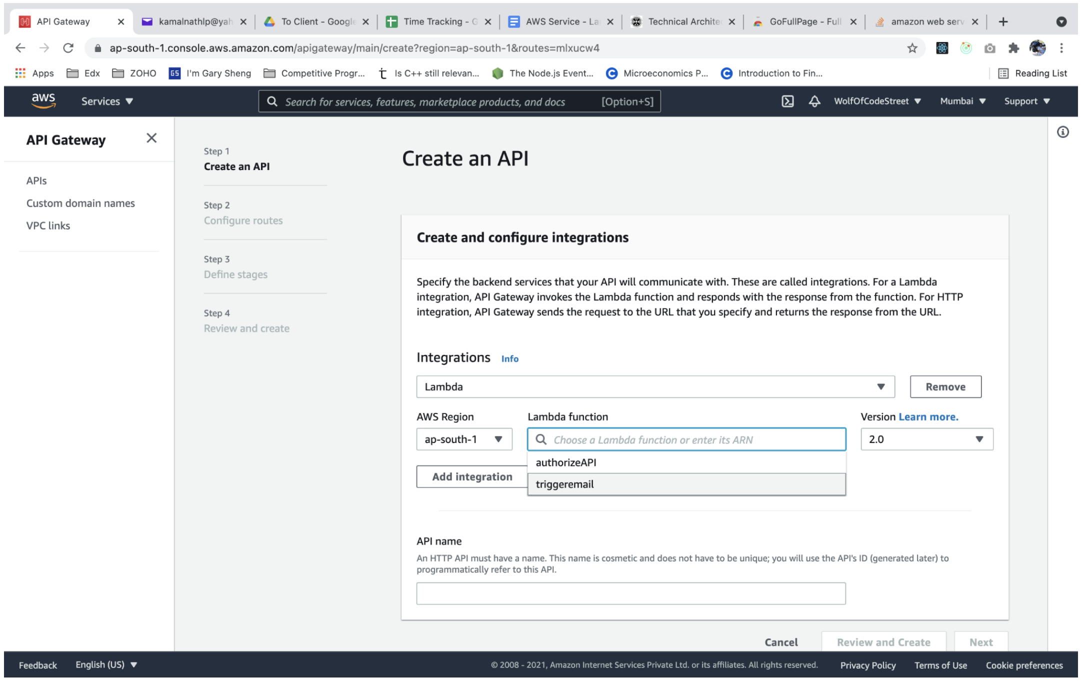
An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Choose,

AWS Region = ap-south-1,

Lambda Function once you click on the input field, our trigger email function will show up.


ap-south-1.console.aws.amazon.com/apigateway/main/create?region=ap-south-1&routes=mlxucw4

API Gateway

APIs
Custom domain names
VPC links

Step 1
Create an API

Step 2
Configure routes

Step 3
Define stages

Step 4
Review and create

Create an API

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations Info

Lambda

AWS Region: ap-south-1
Lambda function: Choose a Lambda function or enter its ARN
authorizeAPI
triggeremail
Version Learn more.: 2.0

Add integration

API name

An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

Cancel | Review and Create | Next

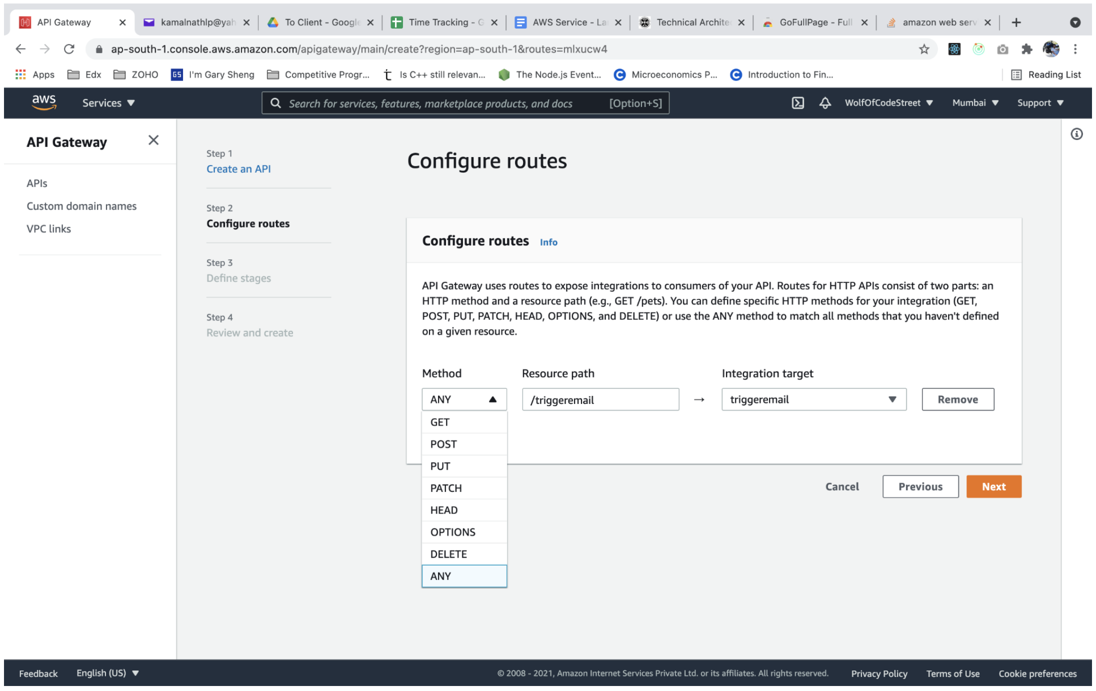
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Version = 2.0

ii. Enter the API Name - TriggerEmail

iii. Click on Next

iv. On the next page, it will ask for the api route method as follows,


ap-south-1.console.aws.amazon.com/apigateway/main/create?region=ap-south-1&routes=mlxucw4

API Gateway

APIs
Custom domain names
VPC links

Step 1
Create an API

Step 2
Configure routes

Step 3
Define stages

Step 4
Review and create

Configure routes

Configure routes Info

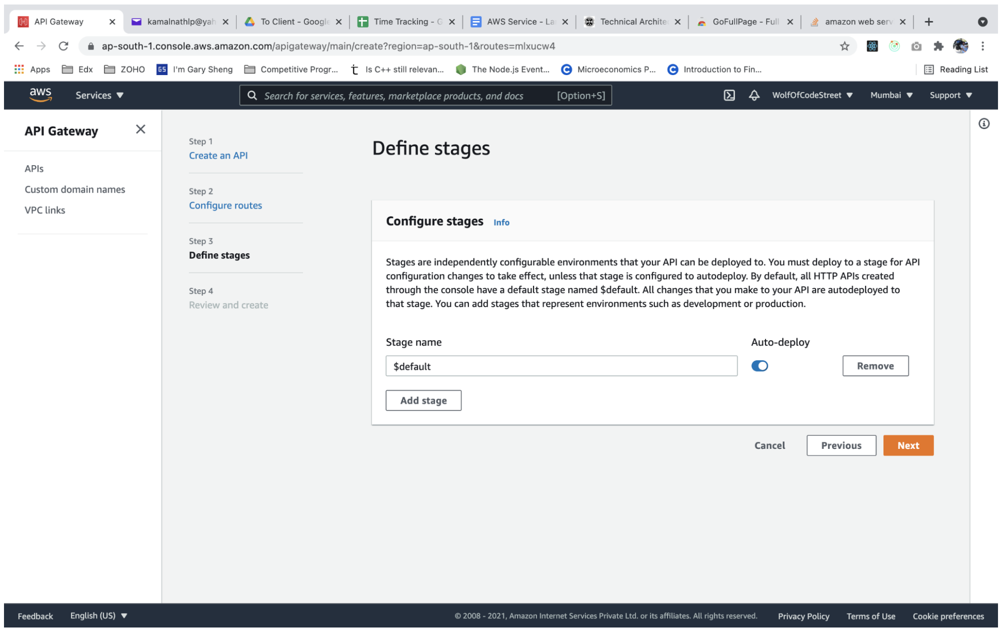
API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

Method	Resource path	Integration target
ANY	/triggeremail	triggeremail

© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Choose the POST Method and click on the Next button

v. Leave the page as it is don't change anything and click on next,



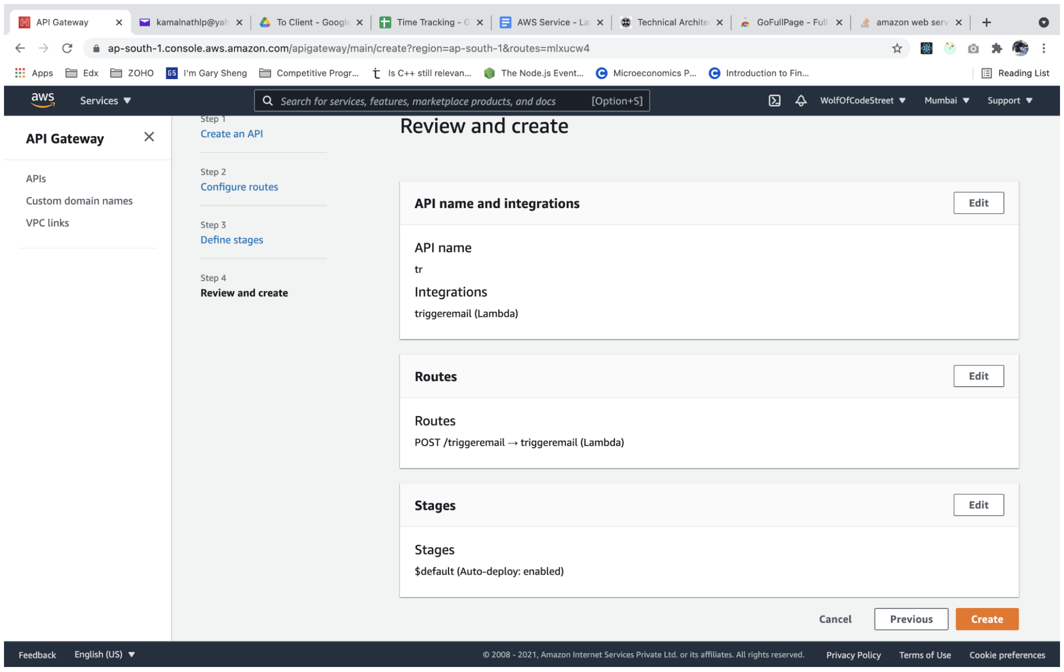
Define stages

Configure stages Info

Stages are independently configurable environments that your API can be deployed to. You must deploy to a stage for API configuration changes to take effect, unless that stage is configured to autodeploy. By default, all HTTP APIs created through the console have a default stage named $default. All changes that you make to your API are autodeployed to that stage. You can add stages that represent environments such as development or production.

Stage name	Auto-deploy
$default	enabled

vi. At last click on the Create Button,



Review and create

API name and integrations

API name
tr

Integrations
triggeremail (Lambda)

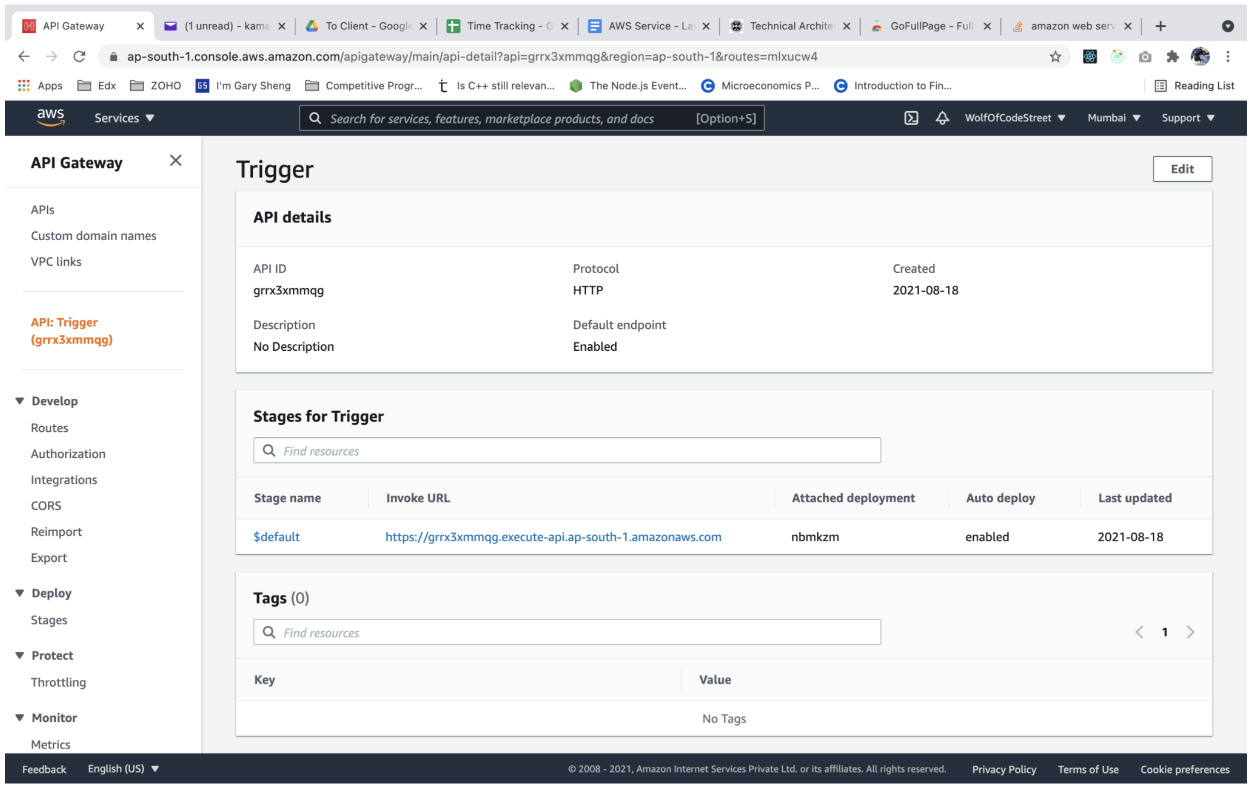
Routes

Routes
POST /triggeremail → triggeremail (Lambda)

Stages

Stages
$default (Auto-deploy: enabled)

c. After Clicking on the create button, you will be redirected to the following page.



ap-south-1.console.aws.amazon.com/apigateway/main/api-detail?api=grrx3xmmqg&region=ap-south-1&routes=mlxucw4

API Gateway

APIs
Custom domain names
VPC links

API: Trigger (grrx3xmmqg)

Develop: Routes, Authorization, Integrations, CORS, Reimport, Export
Deploy: Stages
Protect: Throttling
Monitor: Metrics

Trigger

API details

API ID	Protocol	Created
grrx3xmmqg	HTTP	2021-08-18

Description	Default endpoint
No Description	Enabled

Stages for Trigger

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
$default	https://grrx3xmmqg.execute-api.ap-south-1.amazonaws.com	nbmkzm	enabled	2021-08-18

Tags (0)

Key	Value
No Tags	

© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

d. Copy the Invoke URL. It is the base url.
Our URL to trigger the mail is <BASE URL>/triggeremail

```
const fs = require('fs');
const nodemailer = require('nodemailer');
const handlebars = require('handlebars');
const smtpTransport = require('nodemailer-smtp-transport');

const sessUsername = 'abc@example.com'; // Replace with your username
const sessPassword = 'passss#ss###'; // Replace with your password

exports.handler = async(event) => {
  console.log(event);
  const requestBody = JSON.parse(event.body);
  const body = fs.readFileSync('./templates/order.html').toString();

  const template = handlebars.compile(body);

  let result = null;
```

```javascript
  const replacement = {
    "order_id": requestBody && requestBody.order_id,
    "customer_email": requestBody && requestBody.customer_email,
    "customer_mobile": requestBody && requestBody.customer_mobile,
    "customer_name": requestBody && requestBody.customer_name,
    "products": requestBody && requestBody.products,
    "total_amount": requestBody && requestBody.total_amount,
    "tax_details": requestBody && requestBody.tax_details,
    "bill_amount": requestBody && requestBody.bill_amount,
  };

  const data = template(replacement);

  const transport = nodemailer.createTransport(
    smtpTransport({
      host: "smtp.mail.yahoo.com", // Replace with your SMTP settings, check
your email provider
      port: 465,
      secure: true, // true for 465, false for other ports
      auth: {
        user: sessUsername,
        pass: sessPassword,
      },
    })
  );


  const text = 'Email body goes here';

  const mailOptions = {
    from: sessUsername,
    to: requestBody && requestBody.customer_email,
    subject: `Order Confirmation - ${replacement.order_id}`,
    text: text,
    html: data,
  };

  console.log(`\nMessage Sending: --------\n`);

  try {
    const response = await transport.sendMail(mailOptions);

    console.log(`${requestBody && requestBody.customer_name} -
${JSON.stringify(response)}`);
```

```
    if (response && response.accepted && response.accepted.length) {
      result = {
        statusCode: 200,
        body: {
          errCode: 1,
          errMsg: `Successfully Delivered to ${response.envelope &&
response.envelope.to && response.envelope.to[0] && response.envelope.to[0] }`

        }
      }
    }
    else {
      result = {
        statusCode: 202,
        body: {
          errCode: 0,
          errMsg: `Successfully Not Delivered`
        }
      }
    }

  }
  catch (error) {
    console.log(error);
    result = {
      statusCode: 200,
      body: {
        errCode: 2,
        errMsg: `Successfully Not Delivered`
      }
    }
  }

  console.log(JSON.stringify(result));

  return {
    statusCode: result.statusCode,
    body: JSON.stringify(result.body)
  };
};
```

Sample Request:

< Your URL > /triggeremail

Method: POST
Body:

```
{
   "order_id": "order_1",
   "customer_name": "Demo User",
   "customer_email": "exaple@gmail.com",
   "customer_mobile": "999999999",
   "products": [
       {
           "name": "T-Shirt",
           "quantity": "2",
           "unit_price": "400",
           "sub_total_amount": "800"
       },
       {
           "name": "Trouser",
           "quantity": "2",
           "unit_price": "400",
           "sub_total_amount": "800"
       },
       {
           "name": "Pant",
           "quantity": "2",
           "unit_price": "400",
           "sub_total_amount": "800"
       }
   ],
   "total_amount": "2400",
   "tax_details": "0",
   "bill_amount": "2400"
}
```

Response :

On Success :

{"errCode":1,"errMsg":"Successfully Delivered to codingbysports@gmail.com"}

On Failure:

{"errCode":2,"errMsg":"Successfully Not Delivered"}