

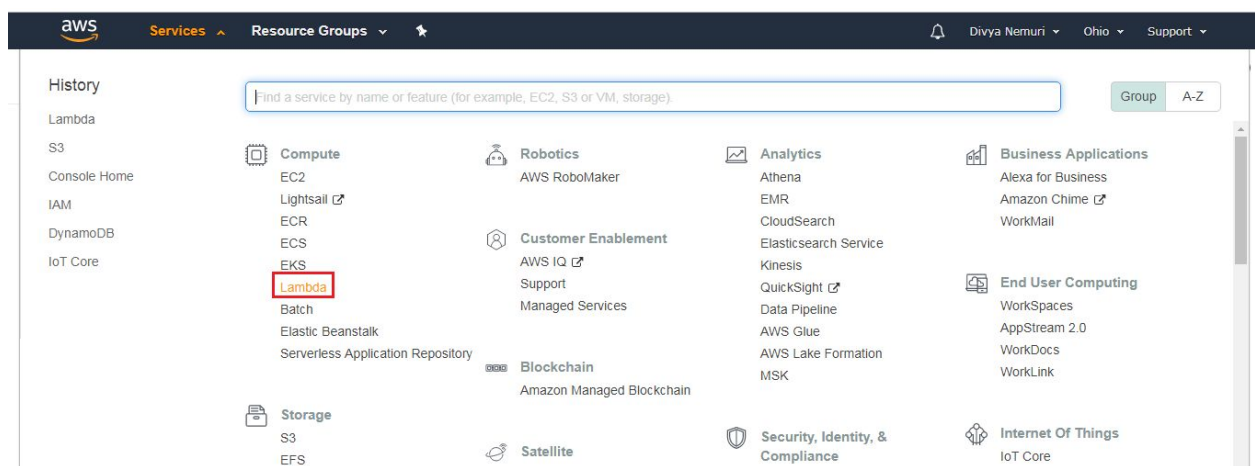
**Aim:** To create an API to publish the data to AWS IOT core from the mobile application using the HTTP request.

**Services Used:**

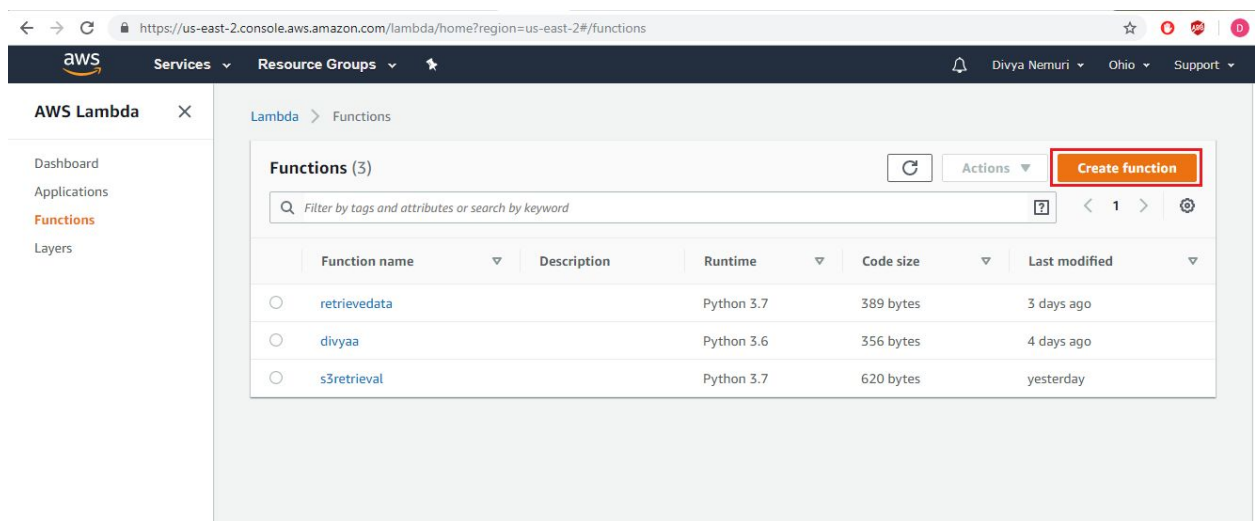
- AWS IoT Core
- Lambda
- API gateway
- IAM Service

Create a lambda function

- Login to your AWS console and search for lambda service



- Double click on the lambda service and click on create function



- There you select an author from scratch, give some function name and select the language as python 3.7

**Create function** [Info](#)

Choose one of the following options to create your function.

**Author from scratch** [Info](#)  
 Start with a simple Hello World example.

**Use a blueprint**  
 Build a Lambda application from sample code and configuration presets for common use cases.

**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.  
  
 Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
 Choose the language to use to write your function.

**Permissions** [Info](#)  
 Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.  
[▶ Choose or create an execution role](#)

[Cancel](#) [Create function](#)

- After selecting the required parameters click on create function. then you will be getting a code editor. there you write your python code for publishing data to the AWS IOT topic.

**sendcommand** [Throttle](#) [Qualifiers](#) [Actions](#) [Select a test event](#) [Test](#) [Save](#)

**Function code** [Info](#)

Code entry type:  Runtime:  Handler:

**Function code**

```

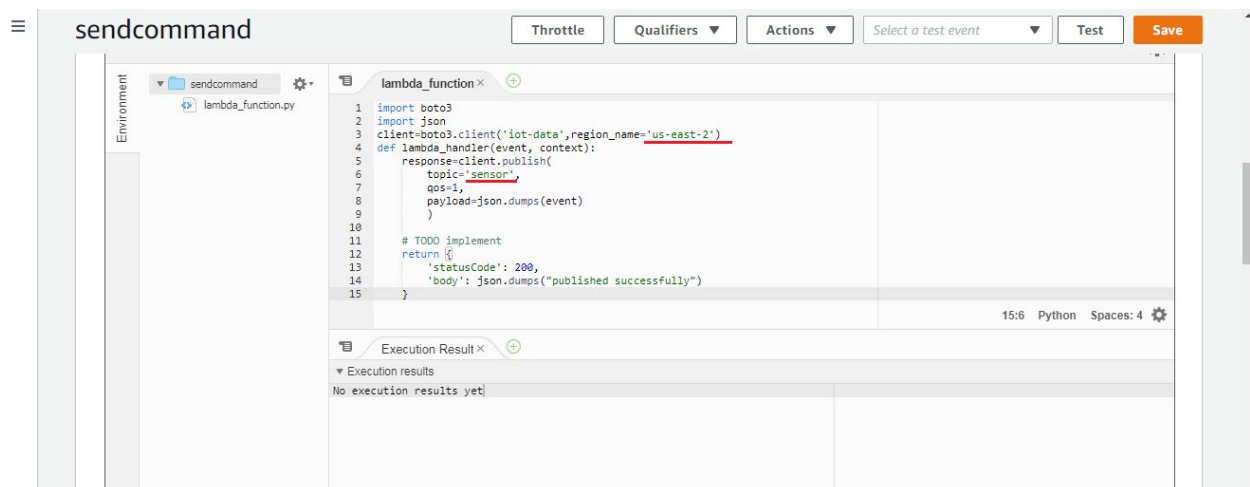
1 import json
2
3 def lambda_handler(event, context):
4     # TODO: Implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
  
```

- In the code change the topic name and the aws IoT region in which you have created the AWS IOT service.

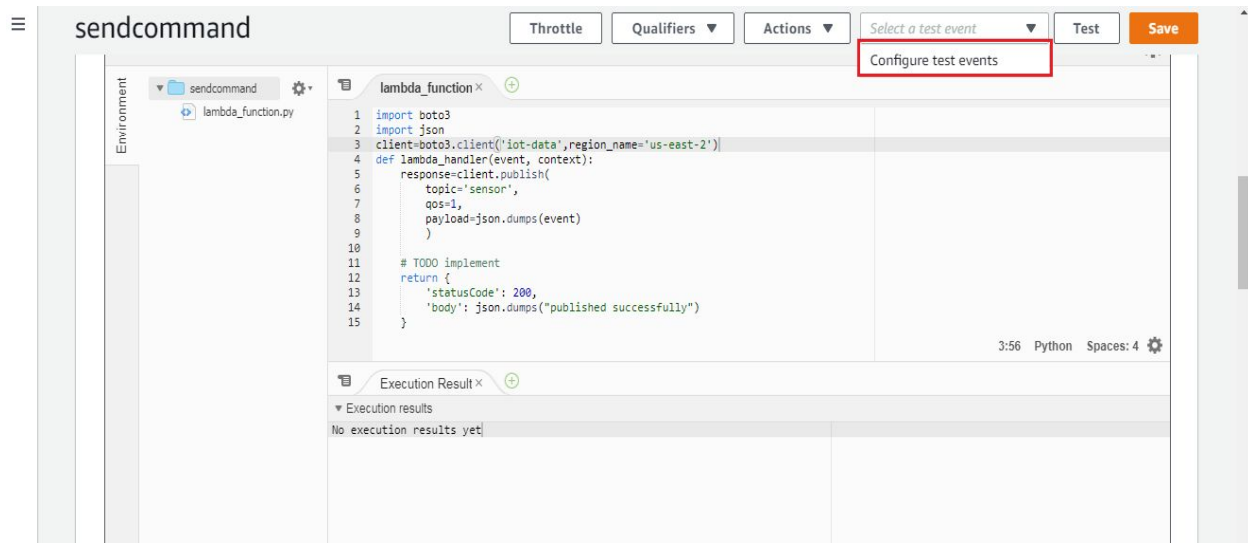
## Lambda function-python code:

```
import boto3
import json
client=boto3.client('iot-data',region_name='us-east-2')
def lambda_handler(event, context):
    response=client.publish(
        topic='sensor',
        qos=1,
        payload=json.dumps(event)
    )

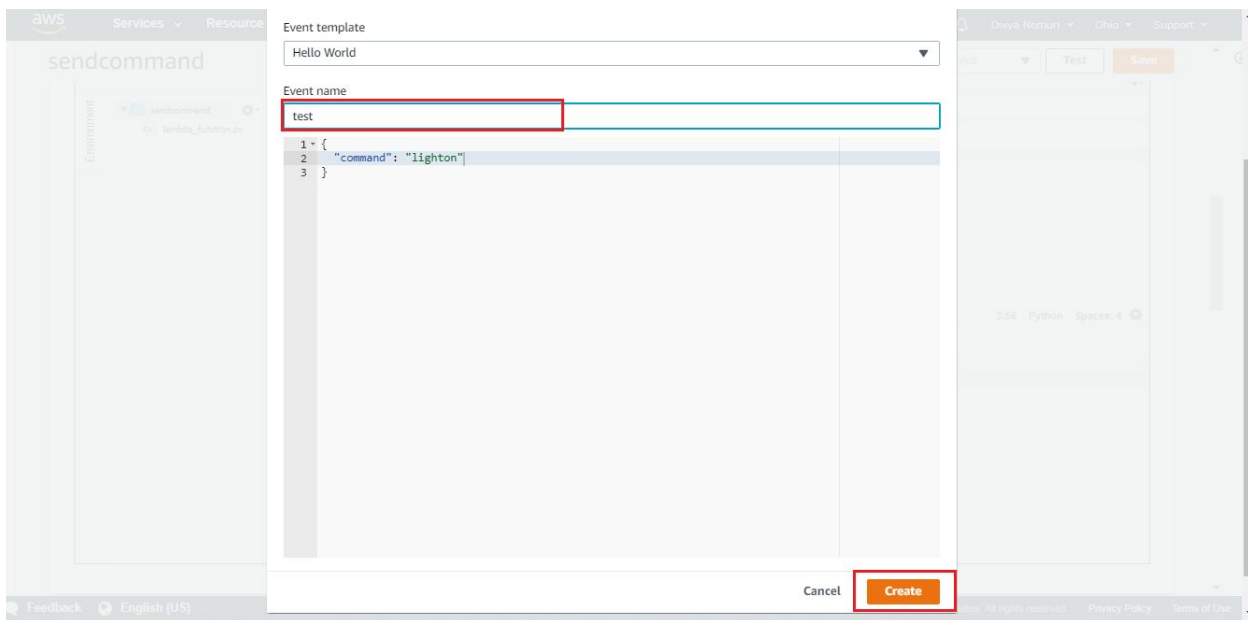
# TODO implement
return {
    'statusCode': 200,
    'body': json.dumps("published successfully")
}
```



- Now in order to test your Lambda function whether it is correct or not, you can test it by using the test option.
- As we don't have any API now we can pass the data using the configure test events.



- Click on the configure test event and there enter some name.
- Enter the JSON data {"command": "lighton"} or {"message": "lighton"} and click on create.



- Now to test your Lambda function we need to attach the IAM role for accessing the AWS IOT core using the lambda function.
- To attach the IAM role just scroll down and under the execution role, you can select your IAM role.

sendcommand

Throttle Qualifiers Actions test Test Save

### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/sendcommand-role-nws73vbl

[View the sendcommand-role-nws73vbl role on the IAM console.](#)

### Basic settings

Description

Memory (MB) [Info](#)

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout [Info](#)

0 min 3 sec

### Network

Virtual Private Cloud (VPC) [Info](#)

Choose a VPC for your function to access.

No VPC

### AWS X-Ray [Info](#)

Enable active tracing to record timing and error information for a subset of invocations.

☐ Active tracing

[View traces in X-Ray](#)

- Under the execution role click on the IAM console, you will be redirected to the IAM console where you can create the IAM role.
- In that console select the AWS Service and choose the service as lambda and click on next.

**AWS service**  
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 the service that will use this role

**EC2**  
Allows EC2 instances to call AWS services on your behalf.

**Lambda**  
Allows Lambda functions to call AWS services on your behalf.

API Gateway	CodeDeploy	ElastiCache	Lambda	S3
AWS Backup	Comprehend	Elastic Beanstalk	Lex	SMS
AWS Chatbot	Config	Elastic Container Service	License Manager	SNS
AWS Support	Connect	Elastic Transcoder	Machine Learning	SWF
Amplify	DMS	ElasticLoadBalancing	Macie	SageMaker
AppStream 2.0	Data Lifecycle Manager	Forecast	MediaConvert	Security Hub
AppSync	Data Pipeline	Global Accelerator	Migration Hub	Service Catalog
Application Auto Scaling	DataSync	Glue	OpsWorks	Step Functions

\* Required

Cancel [Next: Permissions](#)

- In the next page search for AWSIoTDATAACCESS, select that policy and click on next.

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

Create policy ↺

Filter policies  Showing 1 result

	Policy name	Used as
<input checked="" type="checkbox"/>	AWSIoTDataAccess	Permissions policy (2)

› Set permissions boundary

\* Required Cancel Previous Next: Tags

- On the next page leave the key value as empty and click on next.

Create role 1 2 3 4

Add tags (optional)

IAM tags are key-value pairs you can add to your role. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this role. [Learn more](#)

Key	Value (optional)	Remove
<input type="text" value="Add new key"/>	<input type="text"/>	

You can add 50 more tags.

Cancel Previous Next: Review

- Here it will ask to enter the role name , give some name and click on create role.

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 AWSIoTDataAccess [↗](#)

Permissions boundary Permissions boundary is not set

No tags were added.

\* Required Cancel Previous Create role

- Now go back to your lambda page and under the execution role select your IAM role which you have created. If you don't get your role in the existing list just refresh that and select the desired role.

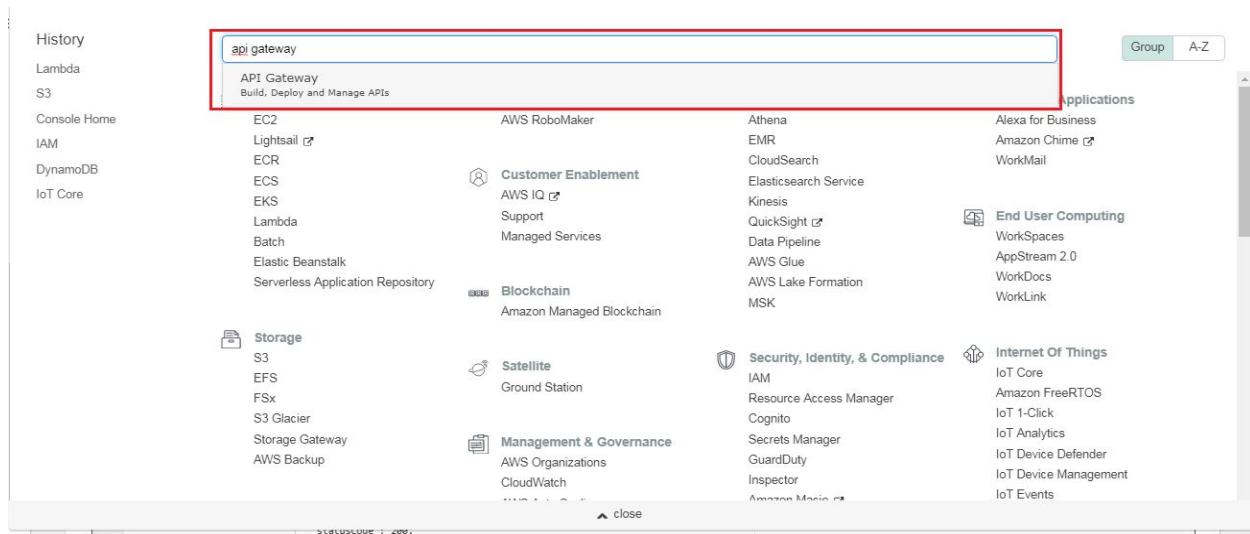
The screenshot shows the AWS Lambda console for a function named 'sendcommand'. The 'Execution role' dropdown is highlighted with a red box and shows 'sendcommand' as the selected role. Other sections include 'Basic settings' (Description, Memory, Timeout), 'Network' (VPC), and 'AWS X-Ray'.

- After selecting the IAM role you can save your lambda function and now you can test it by clicking on the test option.

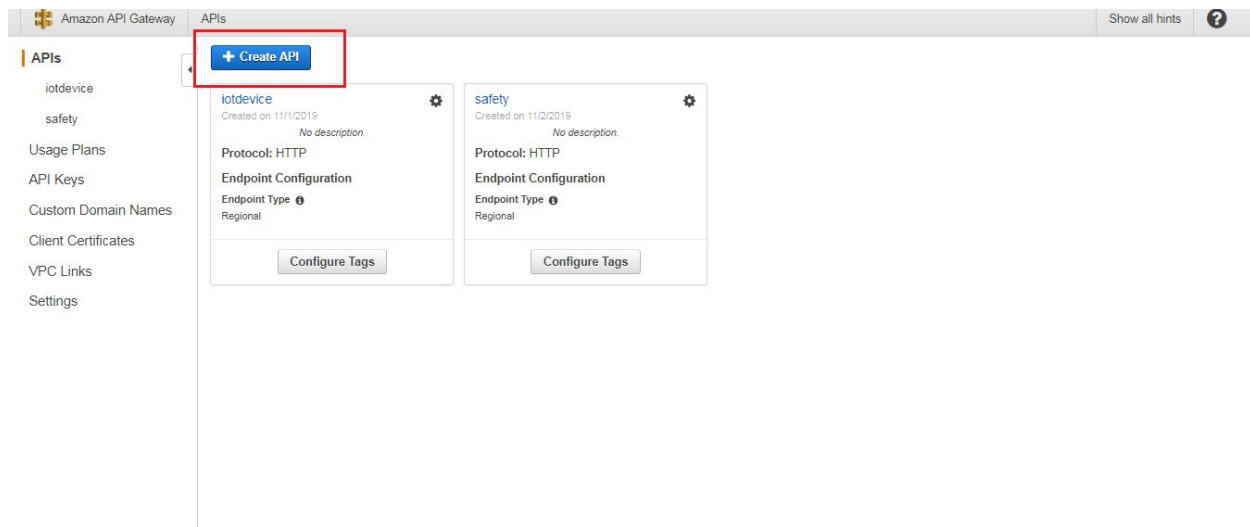
The screenshot shows the AWS Lambda console for a function named 'sendcommand'. The 'Test' button is highlighted with a red box. Below the code editor, the 'Execution Result' tab is open, showing a successful execution with a status of 'Succeeded' and a response body of '{"statusCode": 200, "body": \"published successfully\\n\"}'.

- After testing you can check the execution results. You need to get published successfully message.
- If you got that message then your lambda function is created successfully and then you need to create an API by using the API gateway.

- Now you search for API gateway under the services.



- Double click on that service and you will be redirected to the page where you can create the API. click on create API.



- On the next page select the protocol as rest API, create new API and then give some name to your API and click on create API.



Amazon API Gateway APIs > Create

Show all hints ?

**APIs**

- iotdevice
- safety
- Usage Plans
- API Keys
- Custom Domain Names
- Client Certificates
- VPC Links
- Settings

### 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

- After creating the API you will get an option of selecting the action. select the action as to create resource.

Amazon API Gateway APIs > sendcommand (s9wckjvxd) > Resources > / (5hoz8loxf7)

Show all hints ?

**APIs**

- iotdevice
- safety
- sendcommand
- Resources**
- Stages
- Authorizers
- Gateway Responses
- Models
- Resource Policy
- Documentation
- Settings
- Usage Plans
- API Keys
- Custom Domain Names

### Resources

Actions / Methods

RESOURCE ACTIONS

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

API ACTIONS

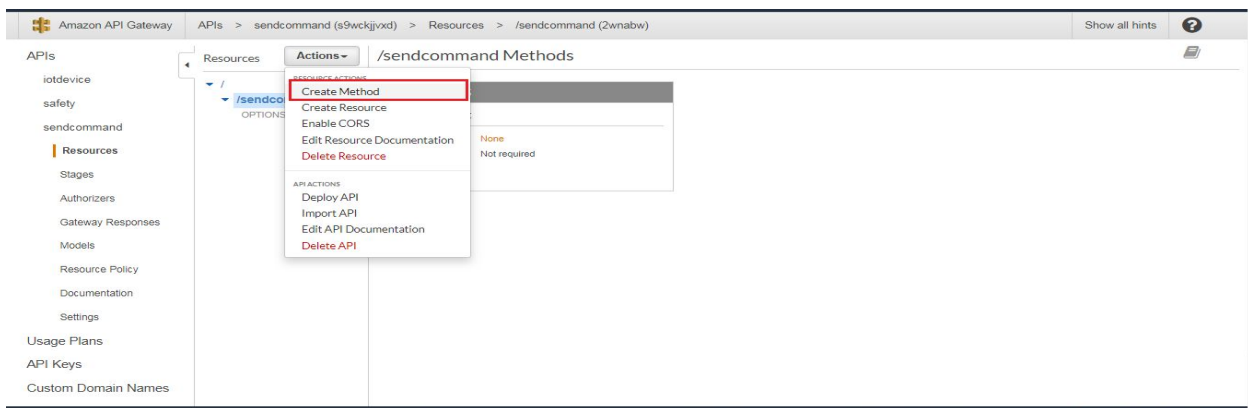
- Deploy API
- Import API
- Edit API Documentation
- Delete API

No methods defined for the resource.

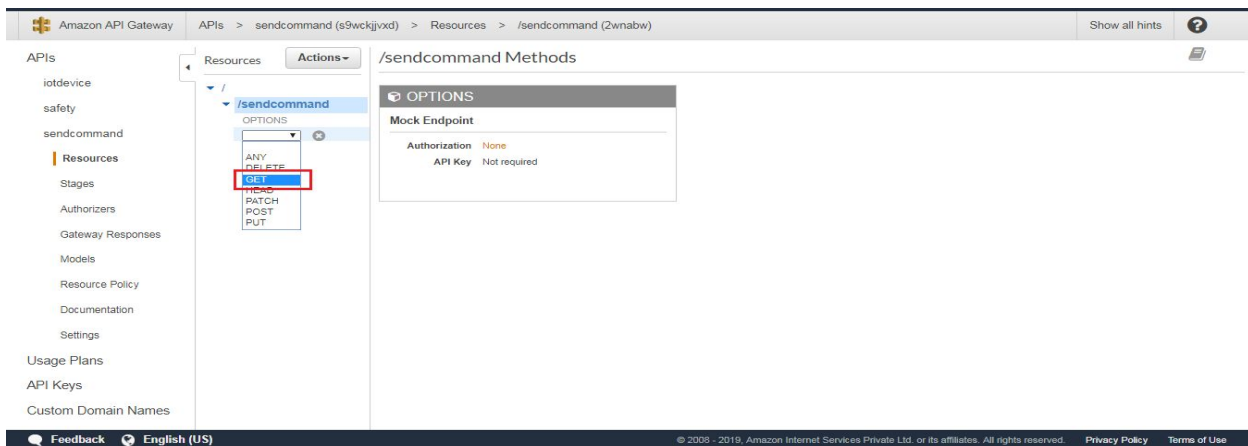
- Give some resource name, enable API Gateway CORS and click on create resource.



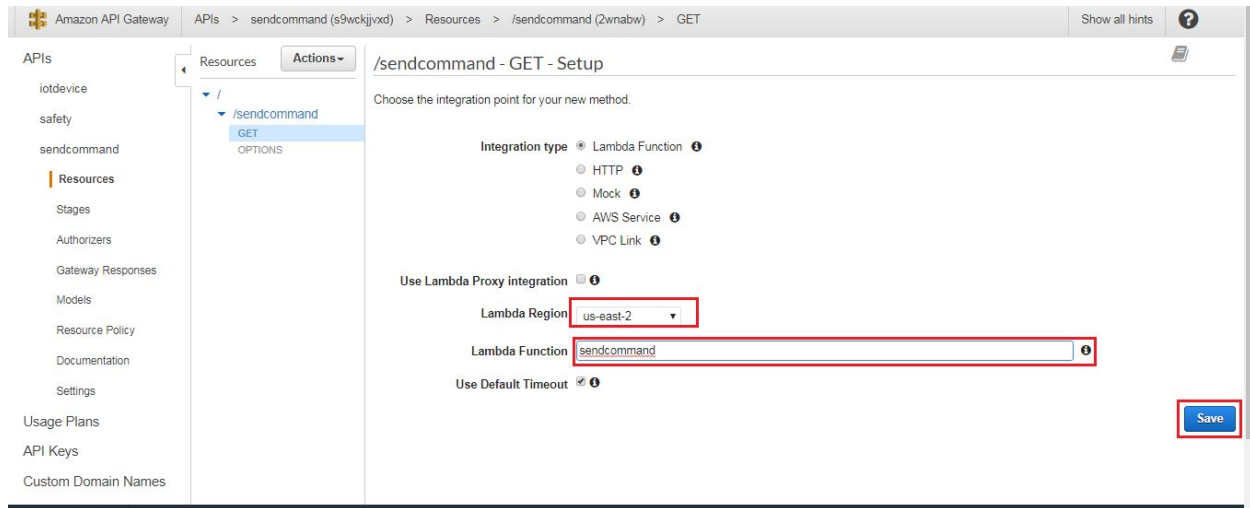
- After creating the resource you need to add the method, go to the actions and click on create method.



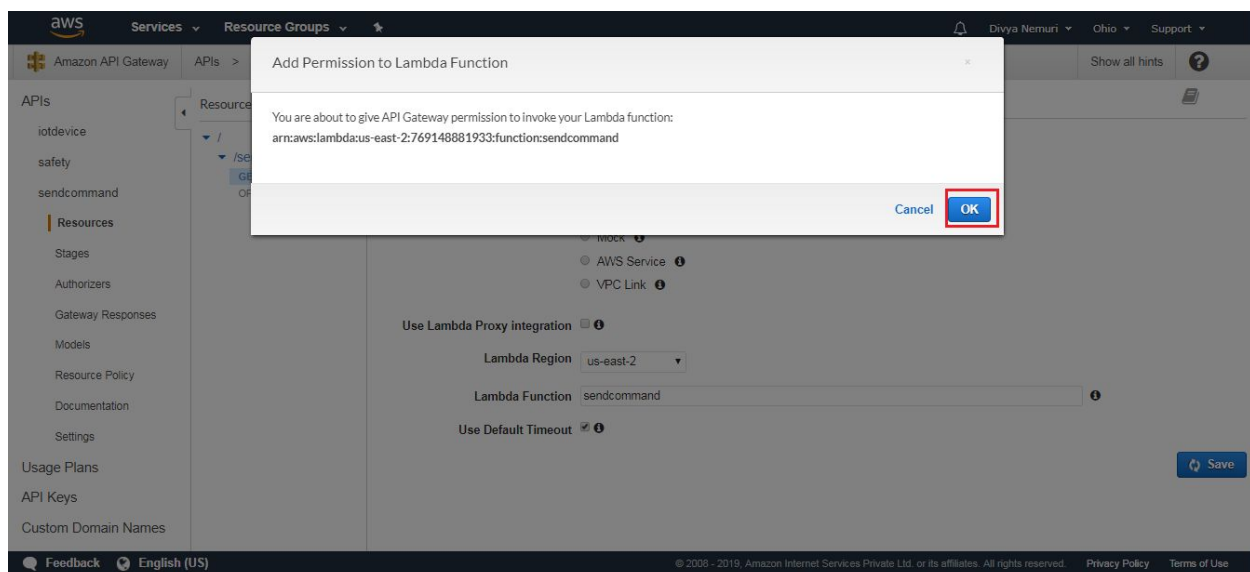
- After creating the method you will get an option of selecting the method.



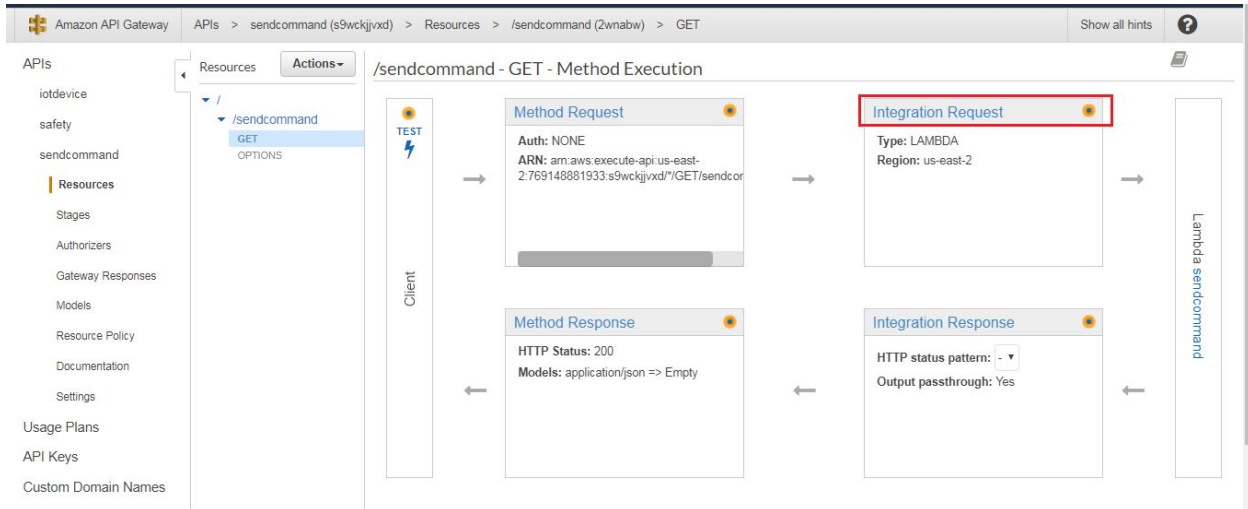
- select GET and click on the tick mark which you got beside.
- Now you need to select the Lambda function region and select the lambda function which you have created and save it.



- You will get a popup to add the permission for lambda function, click on ok.



- On this page click on the integration request.



- On that page scroll down to mapping templates and click on that.

Amazon API Gateway console showing the Integration Request configuration page for the `/sendcommand - GET` method. The page displays various settings for the integration, including Integration type (Lambda Function), Lambda Region (us-east-2), Lambda Function (sendcommand), and Mapping Templates.

**Integration type:**

- ☒ Lambda Function
- ☐ HTTP
- ☐ Mock
- ☐ AWS Service
- ☐ VPC Link

**Use Lambda Proxy integration:**

- ☐ Use Lambda Proxy integration

**Lambda Region:** us-east-2

**Lambda Function:** sendcommand

**Execution role:**

**Invoke with caller credentials:**

**Credentials cache:** Do not add caller credentials to cache key

**Use Default Timeout:** ☒

**Mapping Templates:**

- URL Path Parameters
- URL Query String Parameters
- HTTP Headers
- Mapping Templates

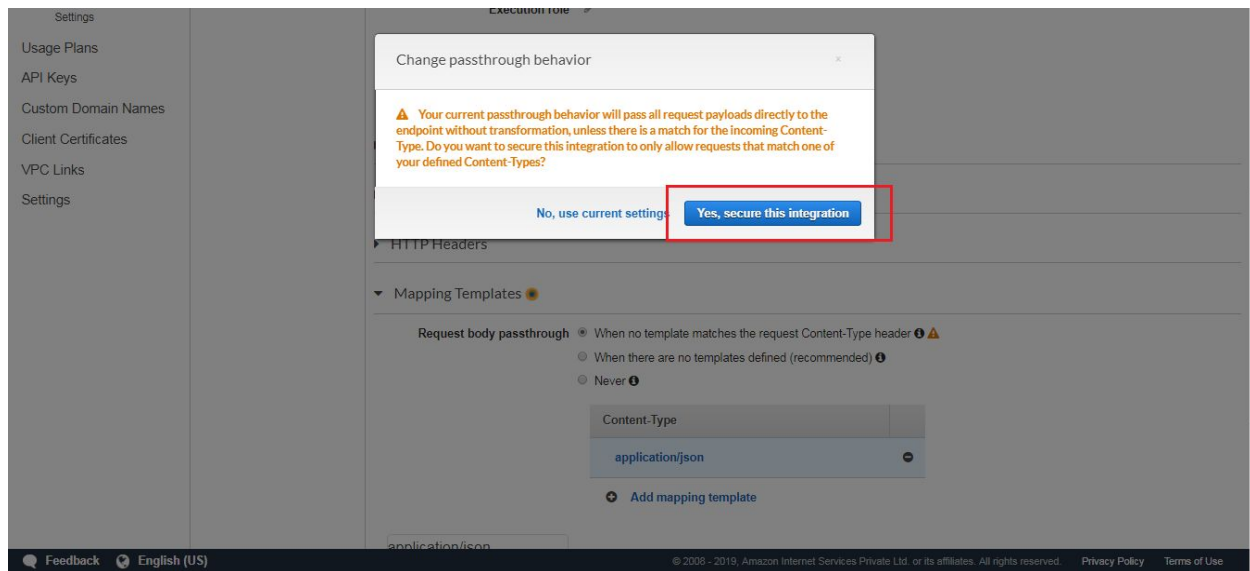
- Click on Add mapping templates

The screenshot shows the 'Mapping Templates' section in the AWS IAM console. On the left is a sidebar with navigation links: Settings, Usage Plans, API Keys, Custom Domain Names, Client Certificates, VPC Links, and Settings. The main panel is titled 'Execution role' and contains several settings: 'Invoke with caller credentials' (checked), 'Credentials cache' (Do not add caller credentials to cache key), and 'Use Default Timeout' (checked). Below these are expandable sections for 'URL Path Parameters', 'URL Query String Parameters', and 'HTTP Headers'. The 'Mapping Templates' section is expanded, showing 'Request body passthrough' options: 'When no template matches the request Content-Type header' (selected), 'When there are no templates defined (recommended)', and 'Never'. Below these options is a 'Content-Type' input field with the text 'No mapping templates defined. The request body will be passed through to the integration endpoint'. The 'Add mapping template' button is highlighted with a red box.

- Add the content type as application/json and click on the tick mark which you got beside.

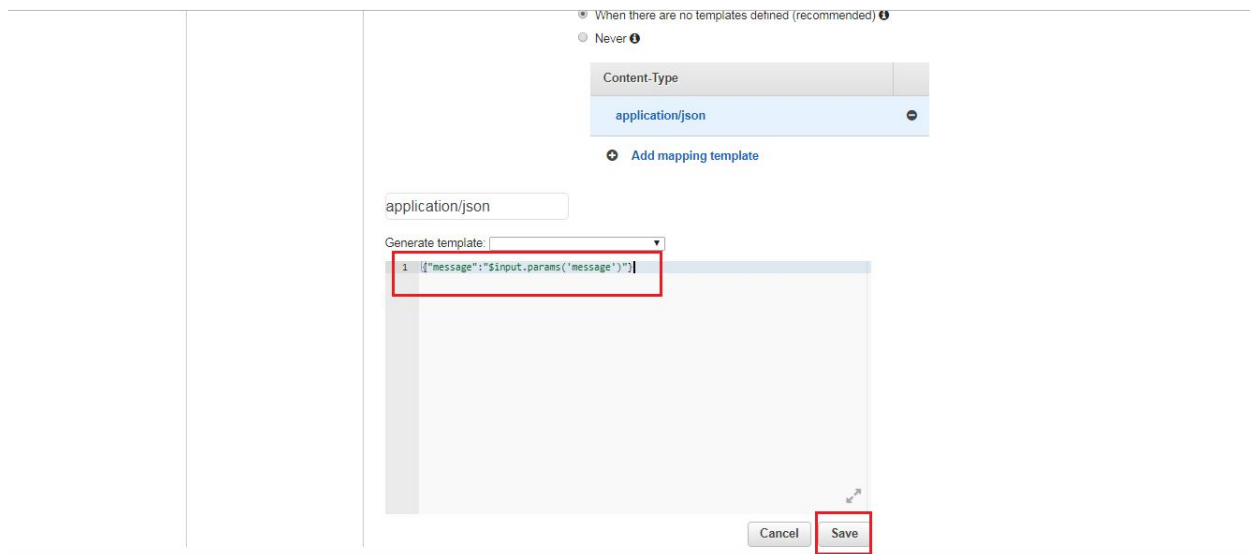
The screenshot shows the 'Mapping Templates' section in the AWS IAM console. The 'Content-Type' input field is now filled with 'application/json'. The tick mark in the adjacent column is checked. The 'Add mapping template' button is highlighted with a red box.

- You will get a popup. Click on yes secure this integration.

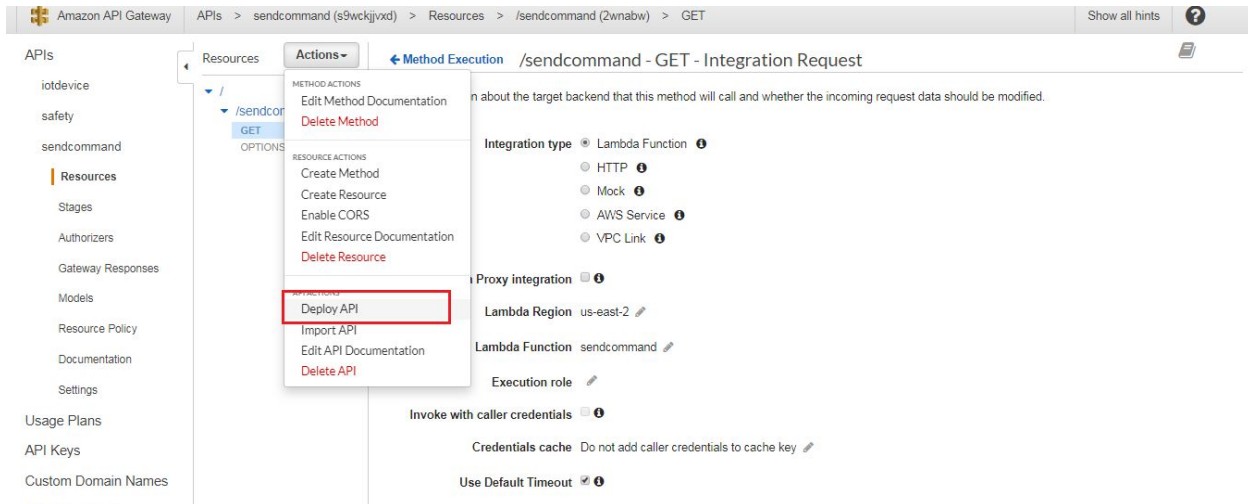


- Now you will get an option of writing the template. There you enter the below code and save it.

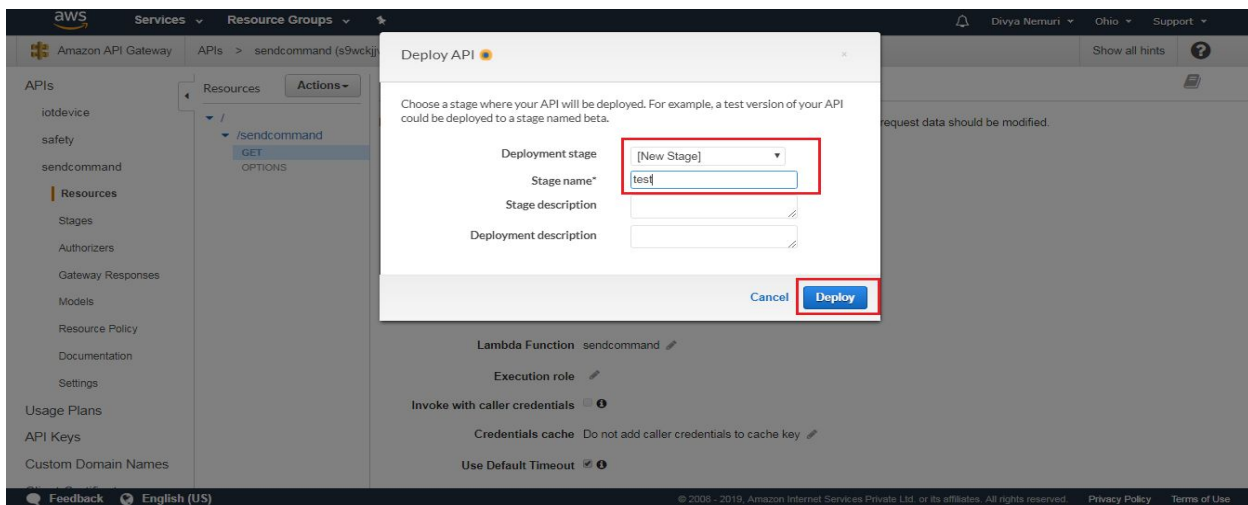
```
{“message”:$input.params(‘message’)}
```



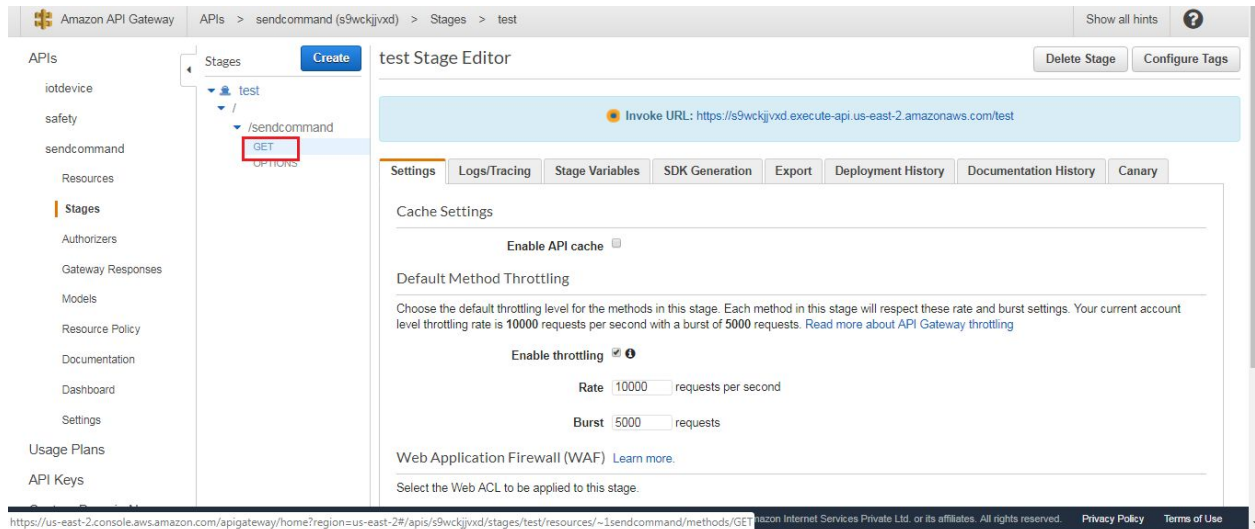
- Now you deploy your API, goto actions and click on deploy API



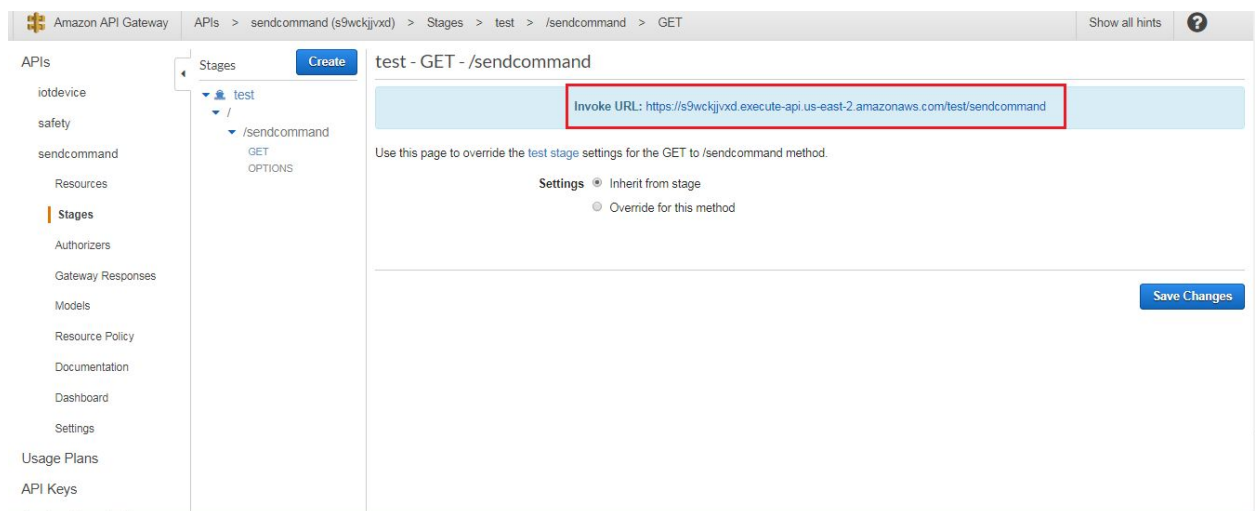
- Select the deployment stage and give some stage name and click on deploy.



- After deploying you will get the stages option in the left side.click on that you will get the options there you click on GET



- Now you will get an Invoke URL.
- This is the URL that you can use in your web application or mobile application to send the command to AWS IoT.



- You can test the URL using the browser, open the new tab and paste the URL and give your command

<https://s9wckjvxd.execute-api.us-east-2.amazonaws.com/test/sendcommand?message=lighton>

- After appending **?message=lighton** to your URL click enter then you need to get a published successfully message.



