

## Create your AWS account

You can just search for "aws create account" in your browser.

Here's a quicker link: <https://aws.amazon.com/resources/create-account/>

After creating your account, redirect to Amazon Sagemaker(you can type in the search bar to locate faster), and click on "Studio" on the left side bar.

## Create the endpoint

Follow the instruction in the link: <https://aws.amazon.com/blogs/machine-learning/llama-2-foundation-models-from-meta-are-now-available-in-amazon-sagemaker-jumpstart/>, and deploy the chosen model.

Notice the endpoint may require you to run on particular instances, for Llama-2-7b-chat, you will likely to receive the following message:

" We encountered an error while preparing to deploy your endpoint. You can get more details below.

An error occurred (ResourceLimitExceeded) when calling the CreateEndpoint operation: The account-level service limit 'ml.g5.2xlarge for endpoint usage' is 0 Instances, with current utilization of 0 Instances and a request delta of 1 Instances. Please use AWS Service Quotas to request an increase for this quota. If AWS Service Quotas is not available, contact AWS support to request an increase for this quota."

## Request instance

In order to use a compute enhanced notebook instance, you must submit a request for a service limit increase to the AWS Support Center.

1. Open the AWS Support Center console.
2. On the AWS Support Center page, choose Create Case and then choose Looking for service limit increases?

AWS Support > Your support cases > Create case

Hello! We're here to help.  
Account: 049362270068 • Support plan: Basic • [Change](#)

How can we help?

Additional information

Contact us

How can we help?

Choose the related issue for your case.

☒ **Account and billing**  
Assistance for your account, such as billing, pricing, and reserved instances.

☐ **Technical**  
Support for service-related technical issues, such as Amazon EC2, Amazon S3 and more.

[Looking for service limit increases?](#)

Cancel **Next step: Additional information**

3. In the Limit type panel, search for SageMaker Notebook Instances.
4. In the Request panel, choose the Region that you are working in(US East (Northern Virginia)). For Resource Type, choose SageMaker Notebook.

5. For Limit choose ml.g5.2xlarge instances.
6. For New Limit Value, verify that the value is 1.

Limit type  
SageMaker Notebook Instances ▼

Severity [Info](#)  
The severity levels available are determined by your support subscription.  
General question ▼

Requests

*ⓘ* To request additional limit increases for the same limit type, choose **Add another request**. To request an increase for a different limit type, create a separate limit increase request.

Request 1

Remove

Region  
US East (Northern Virginia) ▼

Resource Type  
SageMaker Notebook Instances ▼

Limit  
ml.g5.2xlarge ▼

New limit value  
1 ▼

Add another request

7. In Case description, provide a brief explanation of why you need the Service limit increase. For example, I need to use this to deploy Llama-2-7b-chat.
8. In Contact options, provide some details about how you would like to be contacted by the AWS service support team on the status of your Service limit increase request.
9. Choose submit.

PS: I followed the instruction in th link:

<https://docs.aws.amazon.com/deepcomposer/latest/devguide/deepcomposer-service-limit.html> amd submitted case. I was later contacted by Amazon saying I "submitted for the wrong resource type. The error you provided mentions the CreateEndpoint operation, but the request was submitted for Notebook Instance resources (CreateNotebookInstance operation)." Of course through communication, they assigned me the quota. I did not attempt with submitting "CreateEndpoint" operation as the resource type, but anyone is encouraged to try this.

As long as you deploy the model in Sagemaker Studio, which will take a while, you should be able to open a notebook like the one below, beginning from section "Run inference on the Llama 2 endpoint you have created." to "Query endpoint that you have created", including all major contents, along with some of our edits.

## Something to note

The script expects that you have created a SageMaker endpoint named "jumpstart-dft-meta-textgeneration-llama-2-7b-f"(or any name you have on your end) and that you have appropriate access rights to access the

S3 bucket and perform actions on the SageMaker endpoint. Keep in mind that this script uses a specific Llama 2 model with predefined parameters. If you want to use a different model or set different parameters, you can modify the payload dictionary within the loop.

## Sagemaker auto-generated endpoint notebook

You should be able to open a notebook after deploying the endpoint. Here's ours for reference: [meta-textgeneration-llama-2-7b-f](#)

Run inference on the Llama 2 endpoint you have created.

If you wish to read files or data, you will need to create a bucket in S3:

The screenshot shows the 'Create bucket' page in the AWS S3 console. At the top, there's a breadcrumb trail: 'Amazon S3 > Buckets > Create bucket'. The main heading is 'Create bucket' with an 'Info' link. Below this, a sub-header says 'Buckets are containers for data stored in S3. Learn more' with a link. The page is divided into two main sections: 'General configuration' and 'Object Ownership'. In the 'General configuration' section, there's a 'Bucket name' field with the value 'myawsbucket', a note about naming rules, an 'AWS Region' dropdown set to 'US East (N. Virginia) us-east-1', and a 'Choose bucket' button. The 'Object Ownership' section has two radio button options: 'ACLs disabled (recommended)' (selected) and 'ACLs enabled'. The 'ACLs disabled' option explains that objects are owned by the account and access is via policies. The 'ACLs enabled' option explains that objects can be owned by other accounts and access is via ACLs. At the bottom, it shows 'Object Ownership' is set to 'Bucket owner enforced'.

Amazon S3 > Buckets > Create bucket

### Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

#### General configuration

Bucket name

myawsbucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region

US East (N. Virginia) us-east-1

Copy settings from existing bucket - *optional*  
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

#### Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ **ACLs disabled (recommended)**  
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ **ACLs enabled**  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner enforced

After creating a bucket, you can upload desired files and datasets.

Here's the link for more info on buckets:

[https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html?icmpid=docs\\_amazons3\\_console](https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html?icmpid=docs_amazons3_console)

## API Call Using Amazon API Gateway and AWS Lambda

### Resources for API call (tested on postman)

To allow an API call, one needs to build an API gateway and a lambda function to invoke the endpoint then return to the user.

We referenced from this blog: <https://aws.amazon.com/blogs/machine-learning/call-an-amazon-sagemaker-model-endpoint-using-amazon-api-gateway-and-aws-lambda/>

1. Create a role that allows lambda function to invoke the sage maker endpoint Direct to IAM policies and create a new policy. Select **Create policy**.

IAM > Policies

**Policies** (1124) [Info](#)  
A policy is an object in AWS that defines permissions.

	Policy name	Type	Used as	Description
<input type="radio"/>	<a href="#">AmazonSageMaker-ExecutionPolicy-20230716T124701</a>	Customer managed	Permissions policy (1)	
<input type="radio"/>	<a href="#">AmazonSageMakerServiceCatalogProductsUseRole-20230721T171313</a>	Customer managed	Permissions policy (1)	

Switch to **JSON** and copy the following code as your policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "sagemaker:InvokeEndpoint",
      "Resource": "*"
    }
  ]
}
```

You shall see the page like this:

IAM > Policies > Create policy

Step 1  
**Specify permissions**

Step 2  
Review and create

### Specify permissions [Info](#)

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

**Policy editor**

Visual JSON Actions

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": "sagemaker:InvokeEndpoint",
8       "Resource": "*"
9     }
10  ]
11 }
```

**Edit statement**

Select a statement

Select an existing statement in the policy or add a new statement.

[+ Add new statement](#)

Click 'Next' and name your policy then click **Create policy** to finish.

[IAM](#) > [Policies](#) > Create policy

Step 1  
[Specify permissions](#)

Step 2  
**Review and create**

### Review and create

Review the permissions, specify details, and tags.

#### Policy details

Policy name  
Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+=, @-.' characters.

Description - *optional*  
Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and '+=, @-.' characters.

**Permissions defined in this policy** [Info](#)

Edit

Permissions in the policy document specify which actions are allowed or denied.

**Allow (1 of 384 services)** Show remaining 383 services

Service	Access level	Resource	Request condition
<a href="#">SageMaker</a>	Limited: Read	All resources	None

Now we can create a role using the new policy. Direct to IAM Roles and click **Create role**.

[IAM](#) > [Roles](#)

**Roles (27)** [Info](#)

Refresh

Delete

Create role

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

< 1 2 >

Settings

<input type="checkbox"/>	Role name	Trusted entities
<input type="checkbox"/>	<a href="#">AmazonSageMaker-ExecutionRole-20230716T124701</a>	AWS Service: sagemaker
<input type="checkbox"/>	<a href="#">AmazonSagemakerCanvasForecastRole-1689973988385</a>	AWS Service: forecast

We selected Lambda as our service. Click **Next** when you are done.

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

### Select trusted entity [Info](#)

**Trusted entity type**

☒ **AWS service**  
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**  
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**  
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**  
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**  
Create a custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

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

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

Use cases for other AWS services:

Choose a service to view use case

CancelNext

You shall see a list of all policies. Select the one you just created. You can search for a quicker result.

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

### Add permissions [Info](#)

**Permissions policies** (Selected 1/880) [Info](#)  
Choose one or more policies to attach to your new role.

Filter policies by property or policy name and press enter.

1 match

< 1 > ⚙

"invoke" X "invoke\_endpoint" X

Clear filters

<input checked="" type="checkbox"/>	Policy name <a href="#">↗</a>	Type	Description
<input checked="" type="checkbox"/>	<div>⊕ invoke_endpoint</div>		Customer managed

▶ Set permissions boundary - optional [Info](#)

Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission management to others.

CancelPreviousNext

Click **Next** when you are finished.

Name your role and you shall see the policy you added.

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

role\_invoke

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

Description

Add a short explanation for this role.

Allows Lambda functions to call AWS services on your behalf.

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

Step 1: Select trusted entities

Edit

1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": [  
7         "sts:AssumeRole"  
8       ],  
9       "Principal": {  
10         "Service": [  
11          "lambda.amazonaws.com"  
12         ]  
13       }  
14     ]  
15   }]  
16 }

2. Create a Lambda function

Create a new function and select function language. We used Python 3.11 in our case.  
In **Change default execution role**, select **Use an existing role** and add your role.

Lambda > Functions > Create function

Create function info

AWS Serverless Application Repository applications have moved to [Create application](#).

☒ 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.

Basic information

Function name

Enter a name that describes the purpose of your function.

myfunctionapi

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.

Python 3.11

Architecture info

Choose the instruction set architecture you want for your function code.

☒ x86\_64

☐ arm64

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

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.

role\_invoke

View the role\_invoke role on the IAM console.

► Advanced settings

Cancel

Create function

7 / 12

After creating the function, you shall see a page like this:

The screenshot shows the AWS Lambda console for a function named 'myfunctionapi'. The 'Function overview' tab is active, displaying a card for the function with a Lambda icon and a 'Layers' section showing '(0)' layers. To the right, a 'Description' section shows the function's details: 'Last modified 5 seconds ago', 'Function ARN: arn:aws:lambda:us-east-1:049362270068:function:myfunctionapi', and 'Function URL: info'. Below the overview, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code source' tab is selected, showing a code editor with a Python lambda handler function. The code is as follows:

```
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
```

You can edit your function in **Code** section and configure more details using **Configuration**. For example, the default timeout is 3 seconds, which might not be enough in your case and will likely return an error. You can increase the length in **Configuration** and edit the preferable timeout.

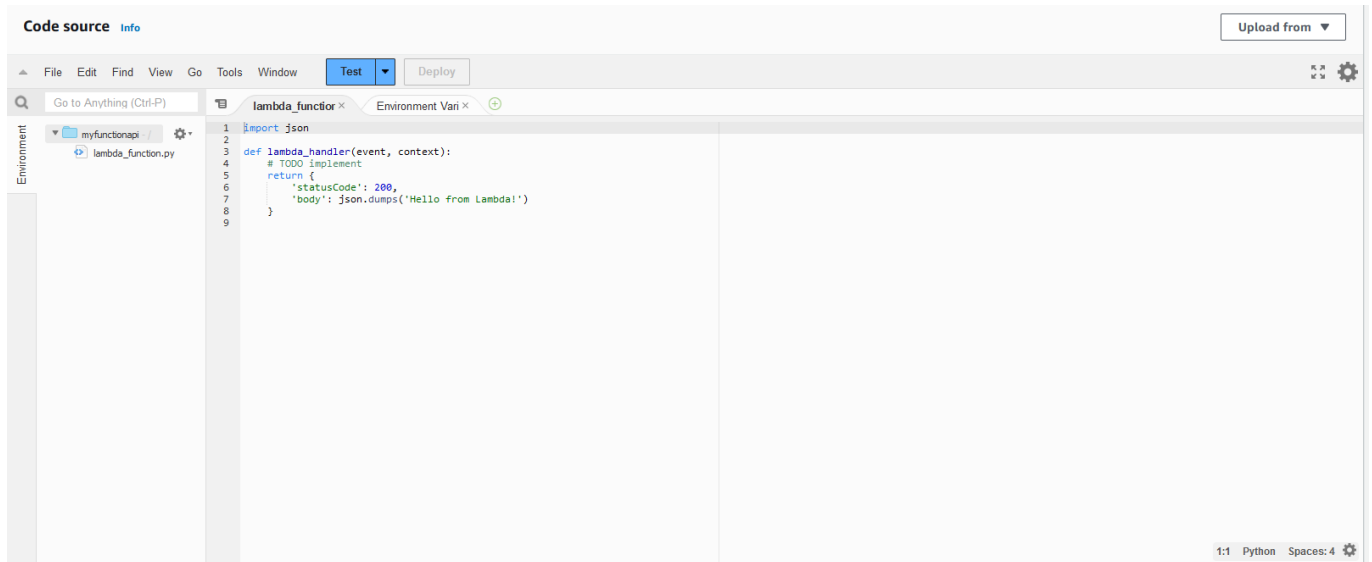
The screenshot shows the 'Configuration' tab for the 'myfunctionapi' function. The 'General configuration' section is active, displaying a table with the following details:

General configuration		
Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart	
0 min 3 sec	None	

Here are more info about Lambda: [https://docs.aws.amazon.com/lambda/latest/dg/welcome.html?icmpid=docs\\_lambda\\_help](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html?icmpid=docs_lambda_help)

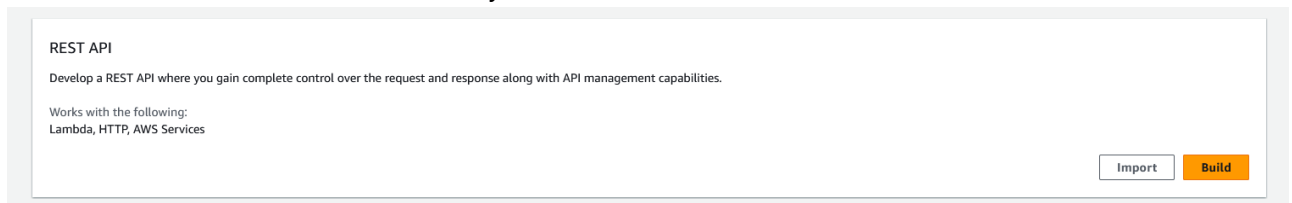
When you make any changes, make sure to click **Deploy** to save the changes. You can also create test case(s) to examine your code.



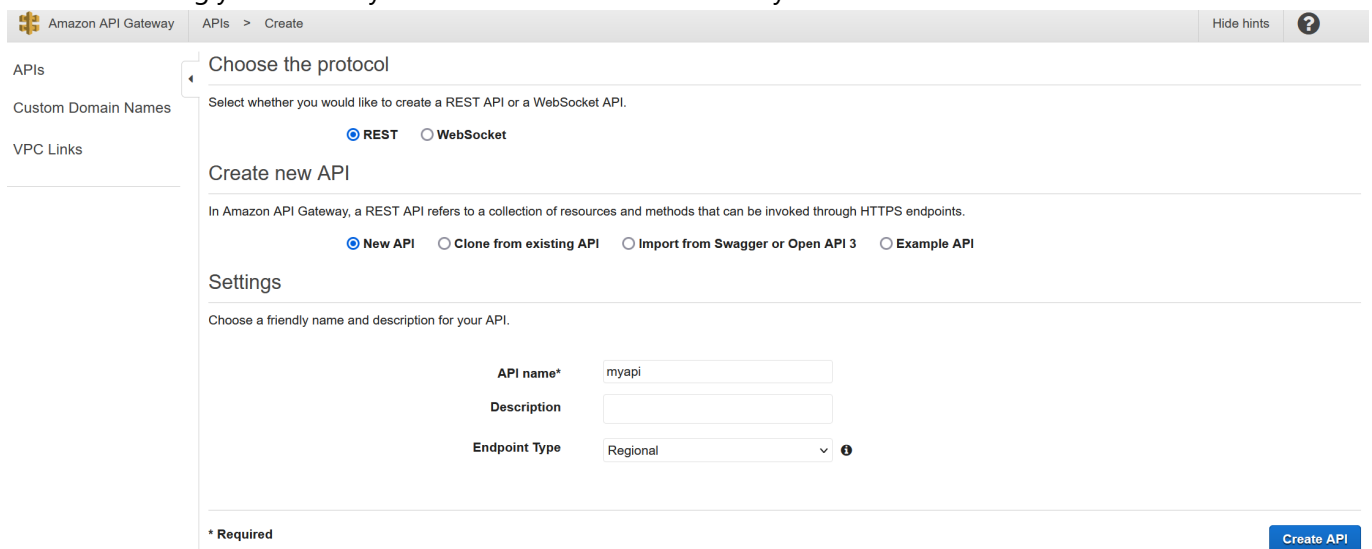


Here's a quick access to our function: [Lambda\\_Function](#)

### 3. Create a REST API Direct to API Gateway and select **Build** of REST API.

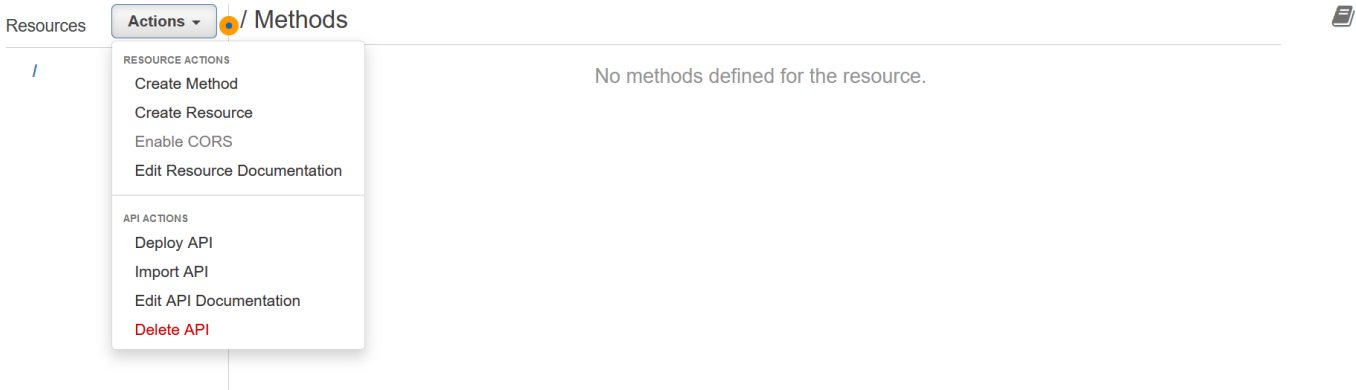


Select accordingly and name your API. Click **Create API** when you are done.

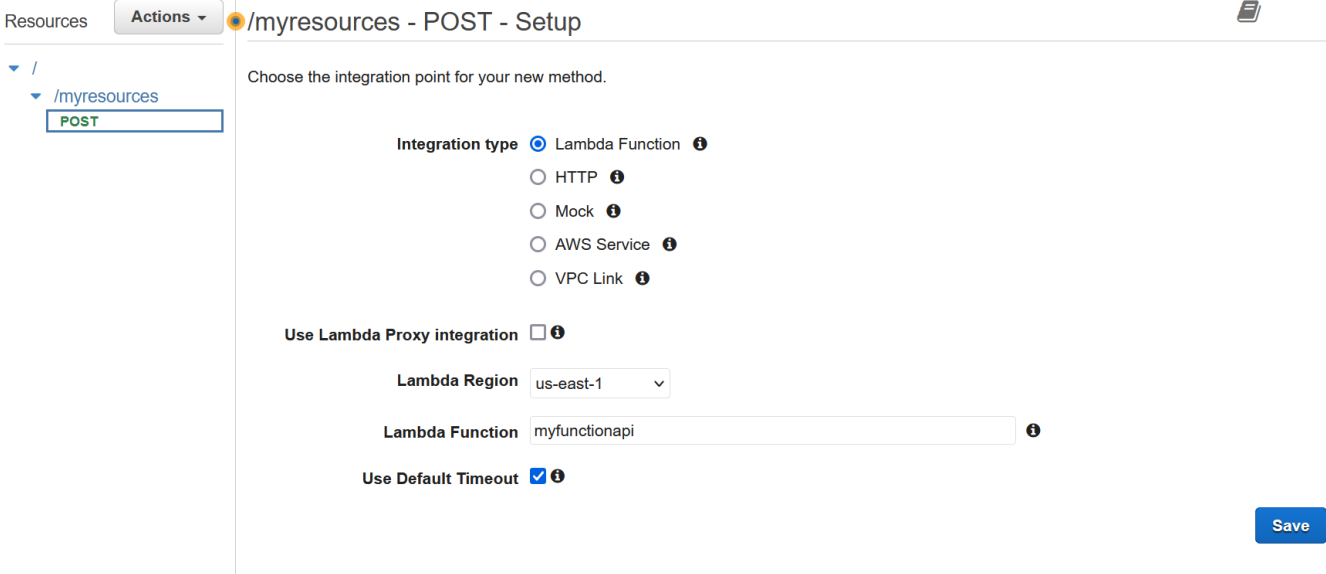


On the **Actions** menu, choose **Create resource**. Enter a name for the resource. After the resource is created, on the **Actions** menu, choose **Create Method** to create a method (It could be any method depends on your

need. We created POST).

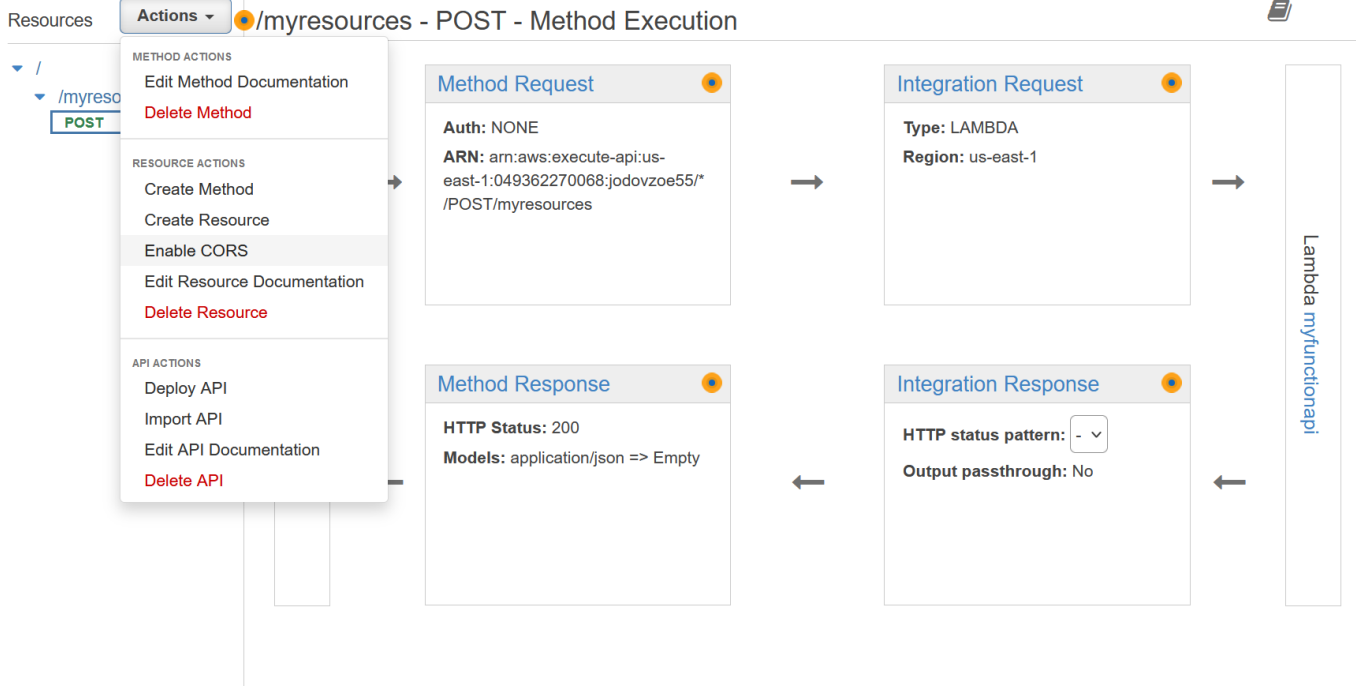


When you created the resource and method, you shall see the following content:




Enter your Lambda function and click **Save**.

Now we can deploy the API through **Actions - Deploy API**:




One last step, name your stage and hit **Deploy** when you are ready.

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\*

newstage

Stage description

Deployment description

Cancel


Deploy

You will receive your invoke url:

newstage Stage Editor

Delete Stage

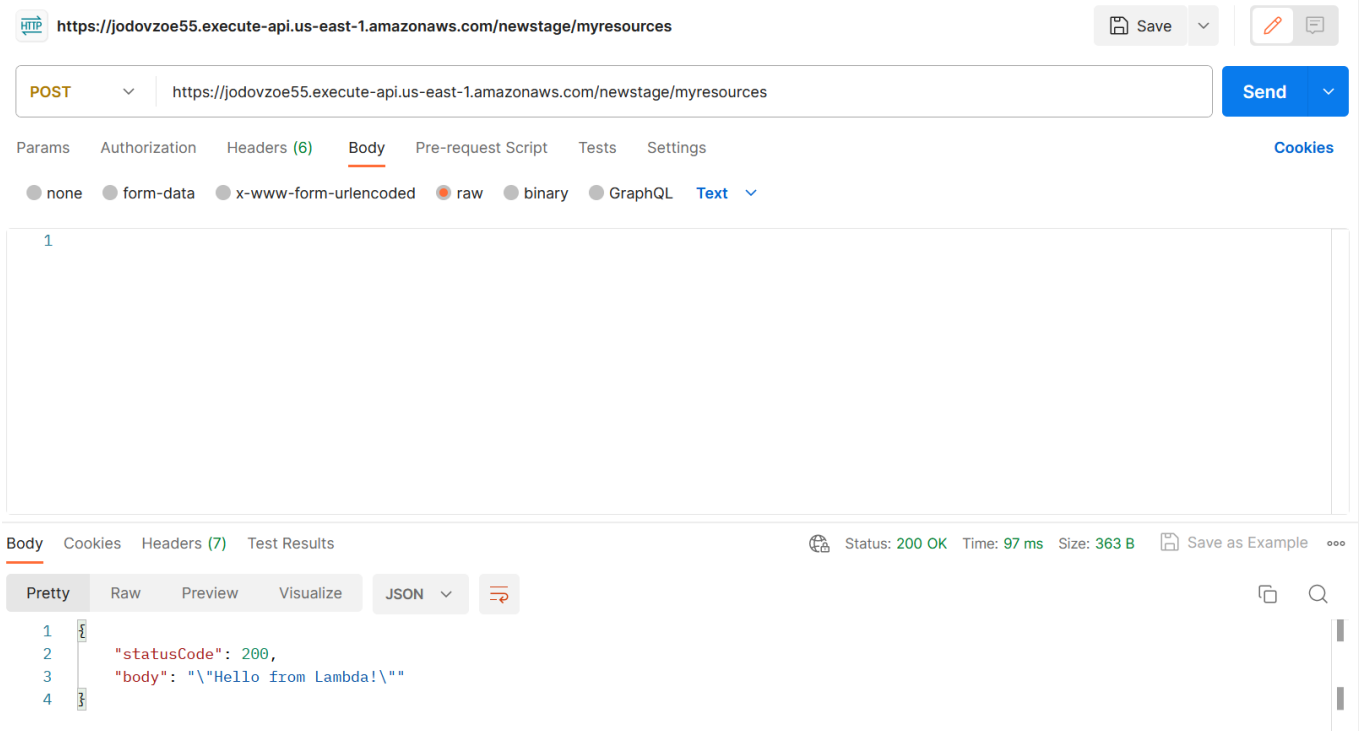
Configure Tags

 Invoke URL: <https://jodovzoe55.execute-api.us-east-1.amazonaws.com/newstage>

#### 4. Test in Postman

You input url should follow the format: `https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/{resource_name}` In our case, we select **POST** as the method and enter our complete url, with the result returned below:

11 / 12



If your lambda function aims to take inputs and return outputs, you can test by input data in **Body - raw**.

## More references

<https://docs.aws.amazon.com/sagemaker/latest/dg/realtime-endpoints-test-endpoints.html>

[https://docs.aws.amazon.com/sagemaker/latest/APIReference/API\\_runtime\\_InvokeEndpoint.html](https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_runtime_InvokeEndpoint.html)

<https://aws.amazon.com/blogs/machine-learning/llama-2-foundation-models-from-meta-are-now-available-in-amazon-sagemaker-jumpstart/>

<https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html>

## Clean up

When using AWS, a platform both powerful and expensive, you want to clean up all your running resources to avoid accidental charges. Following the guidelines through this link:

<https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-cleanup.html>, in additional with Step 6 in another link:

<https://aws.amazon.com/tutorials/machine-learning/tutorial-deploy-model-to-real-time-inference-endpoint/>.

Then you should be able to shut down all running resources. If you want to revisit some resources later, you are welcome to keep them, but be aware this could incur accumulating charges.

Usually the free tier does have free usage for Lambda Function and API, but you may want to delete them to prevent charges and clean up some space. Here's a guideline:

<https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html>

To monitor your bills, you can setup alerts as preventive measures:

-- Budgets: <http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/budgets-managing-costs.html>

-- CloudWatch billing alerts and alarms: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-cloudwatch.html>