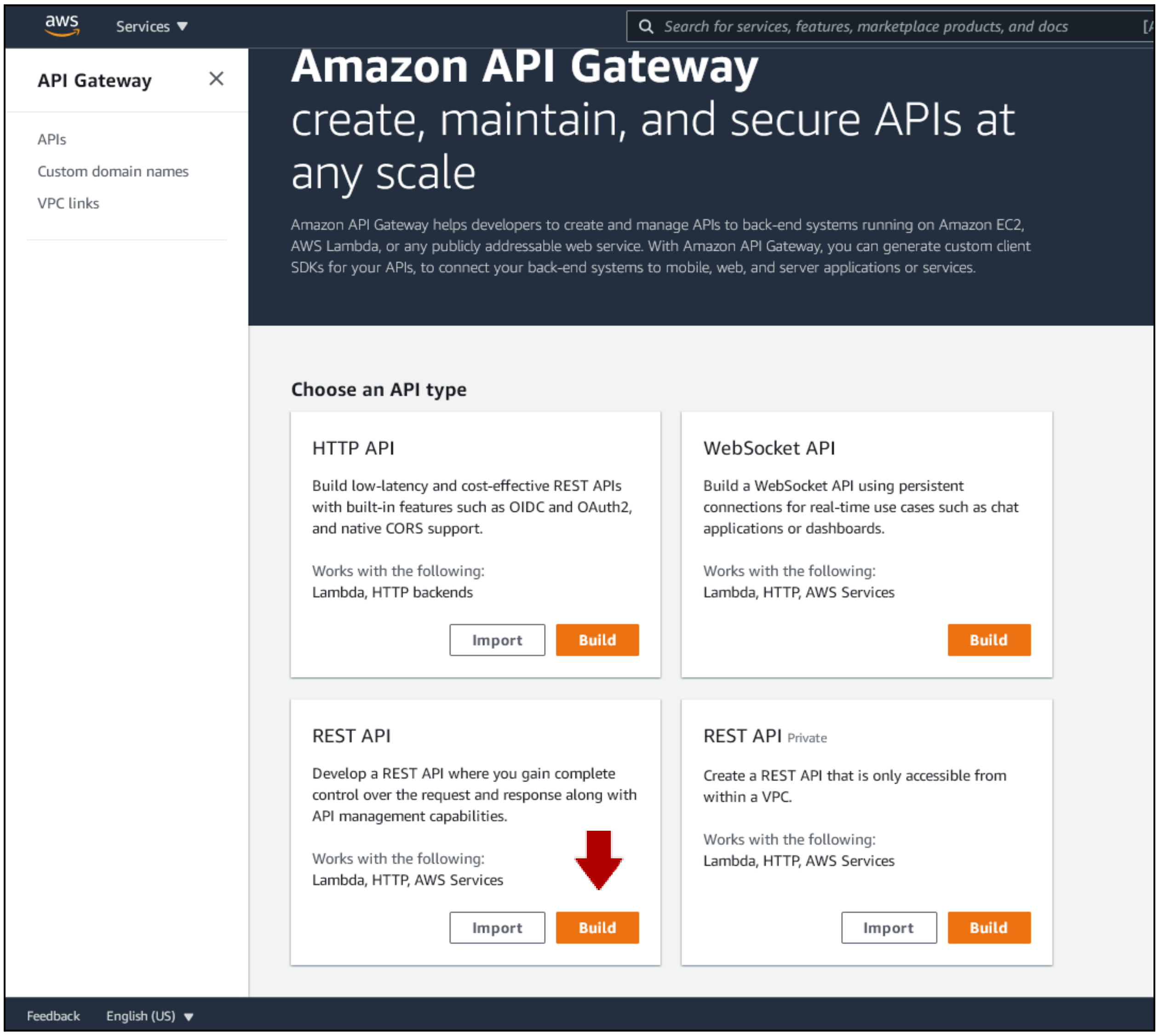
CS 470 Module Four Assignment Two Guide

Part One – Creating an API Using API Gateway

1. Navigate to the **API Gateway** in the AWS Console. It is located under the **Networking & Content Delivery** section. You can also search for it.
2. Select **REST API** by clicking the **Build** button in the **REST API** box.



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

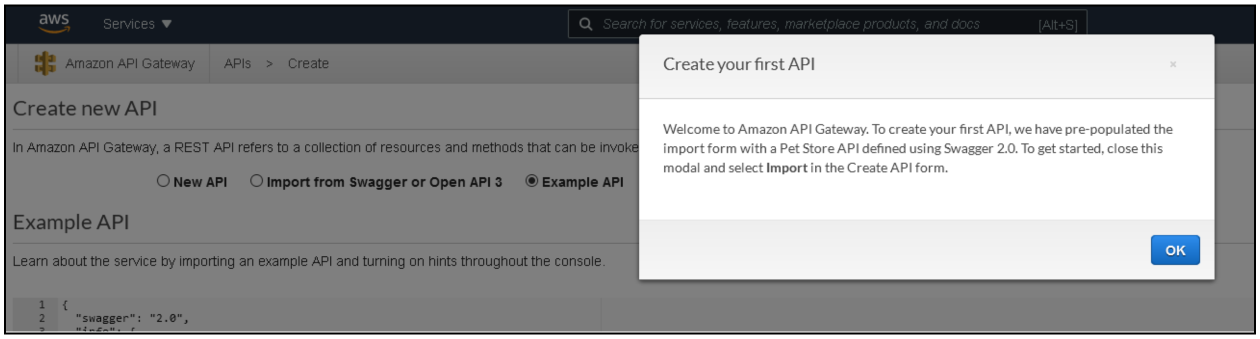
HTTP API
Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.
Works with the following:
Lambda, HTTP backends

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

3. Dismiss the dialog box shown below, and set Create New API to **New API**.


aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
Amazon API Gateway
APIs > Create

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoke

New API [✓ Example API] Import from Swagger or Open API 3

Example API

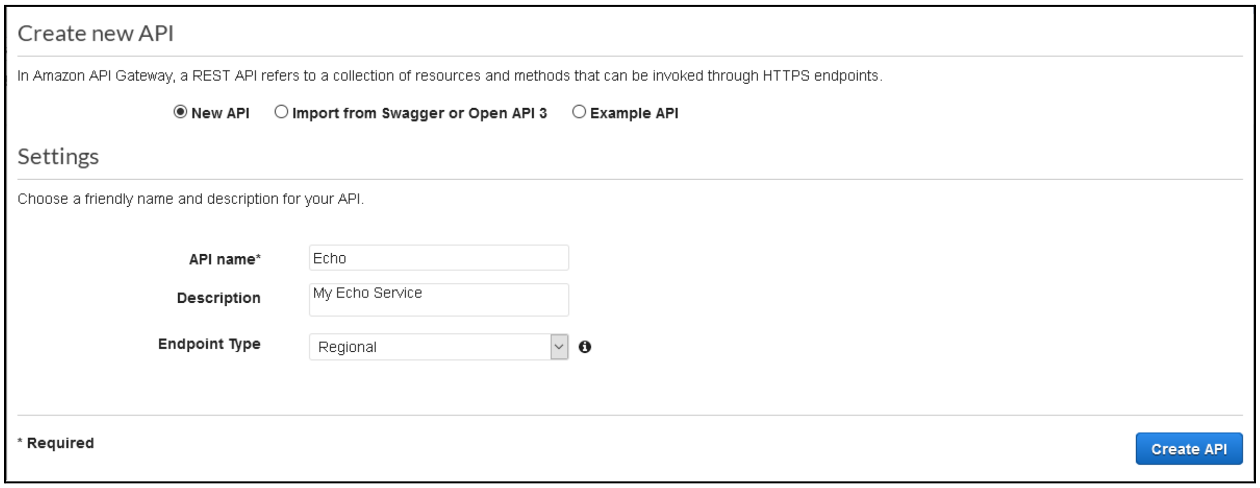
Learn about the service by importing an example API and turning on hints throughout the console.

```
{
  "swagger": "2.0",
```

Create your first API

Welcome to Amazon API Gateway. To create your first API, we have pre-populated the import form with a Pet Store API defined using Swagger 2.0. To get started, close this modal and select **Import** in the Create API form.

4. Set the API Name to “Echo” and give it a description of “My Echo Service”.
5. Leave the Endpoint Type as **Regional**.
6. Now click the blue **Create API** button in the lower-right corner.



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] Import from Swagger or Open API 3 Example API

Settings

Choose a friendly name and description for your API.

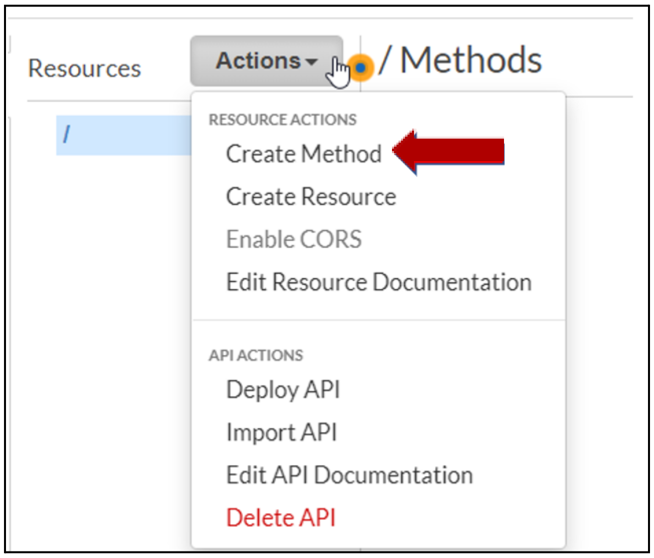
API name*	Echo
Description	My Echo Service
Endpoint Type	Regional

* Required

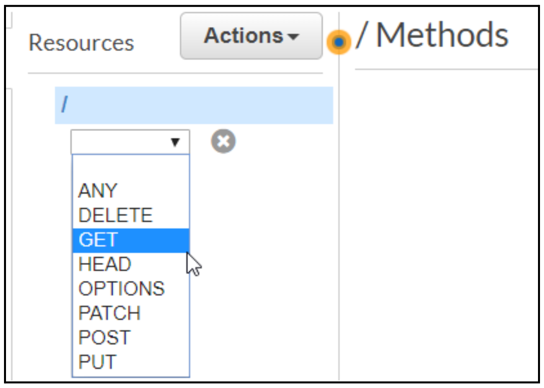
7. You will now see a mostly blank screen with the “/Methods” title.
8. Believe it or not, you have now created an API. It will not do anything yet, but that is for the next part.

Part Two – Adding a Method to Your API

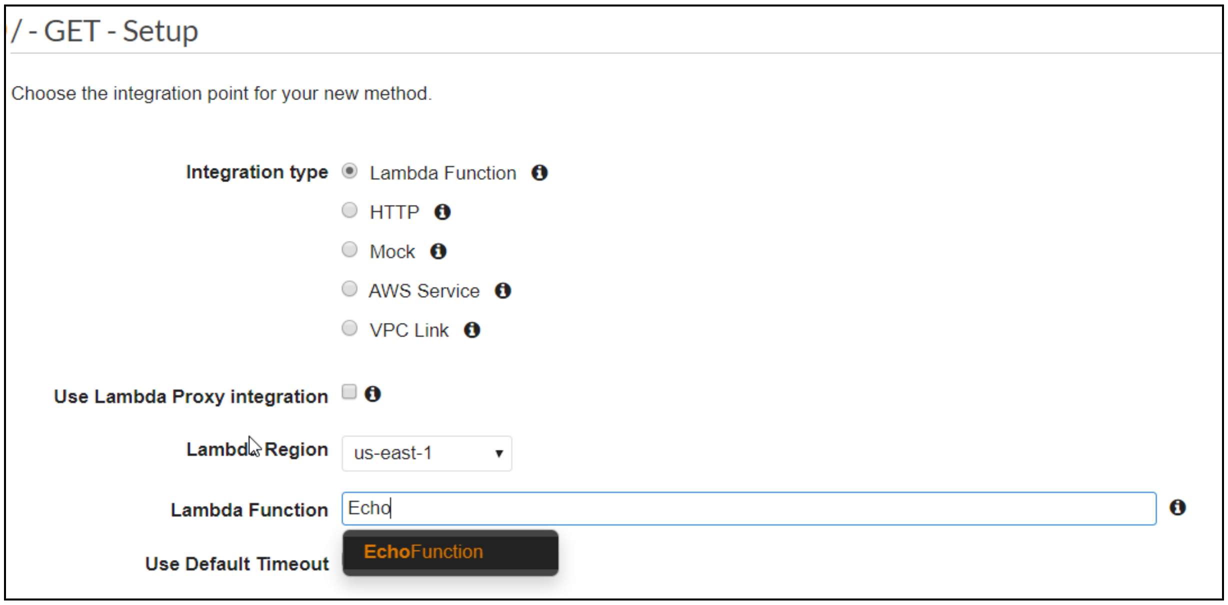
1. Select the **Actions** drop-down menu and click **Create Method**.



2. A new blank drop-down box will be created under the forward slash (/). This is where you select the REST methods to process for your API. In this case, you will select GET to handle HTTP GET requests. The other options are direct mappings to HTTP methods. However, ANY is used to match all of the requests. It will match any request not explicitly handled by another method definition.



3. Click the check box next to the drop-down menu that just appeared.
4. Now we need to set up our method. Make sure the integration type is **Lambda Function**.
5. Make sure the **Use Lambda Proxy Integration** check box is checked.
6. Leave the region as "us-east-1".
7. Type "EchoFunction" into the **Lambda Function** box. AWS will suggest it for you as you type.


/ - GET - Setup

Choose the integration point for your new method.

Integration type: [✓ Lambda Function] HTTP Mock AWS Service VPC Link

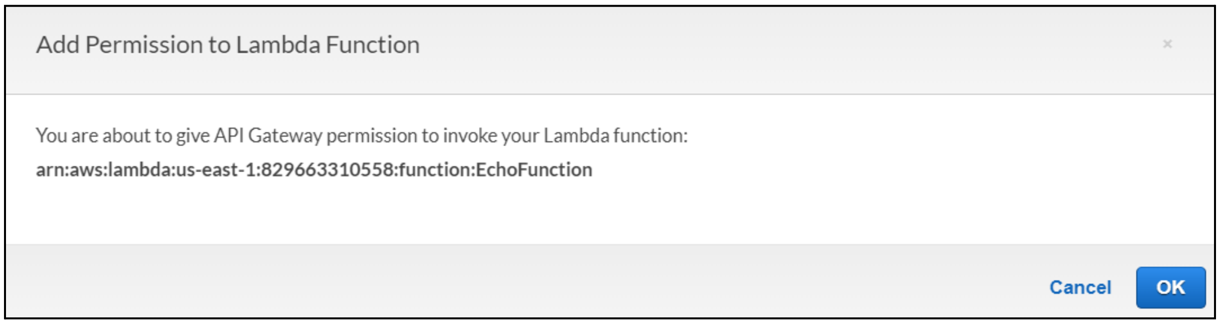
Use Lambda Proxy integration: ☐

Lambda Region: us-east-1

Lambda Function: Echo

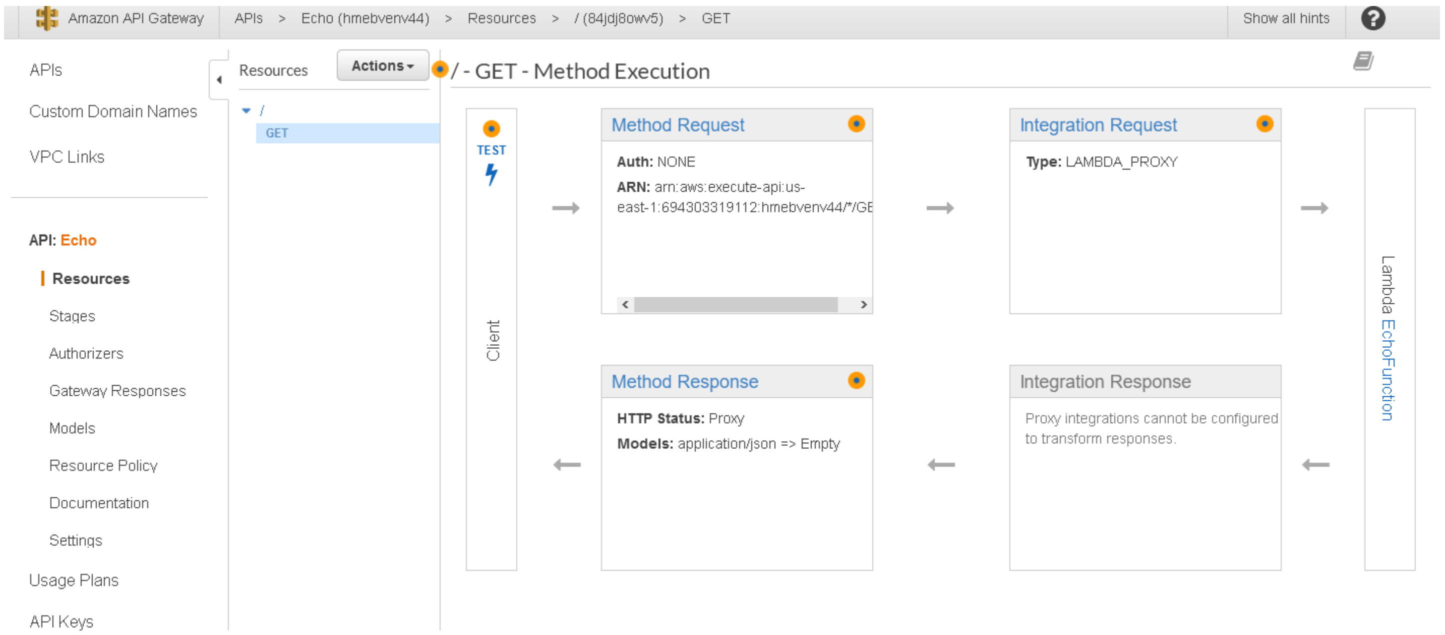
Use Default Timeout

8. Make sure **Use Default Timeout** is checked.
9. Click the blue **Save** button in the lower-right corner.
10. You will be asked if you want to **Add Permission to Lambda Function**.


Add Permission to Lambda Function

You are about to give API Gateway permission to invoke your Lambda function:
arn:aws:lambda:us-east-1:829663310558:function:EchoFunction

11. Click **OK**. This is how you tell the API Gateway that the GET method has permission to execute your Lambda.
12. You will now see the fully populated **Method Execution** page.



Auth: NONE
ARN: arn:aws:execute-api:us-east-1:694303319112:hmebvenv44/*/GE

Type: LAMBDA_PROXY

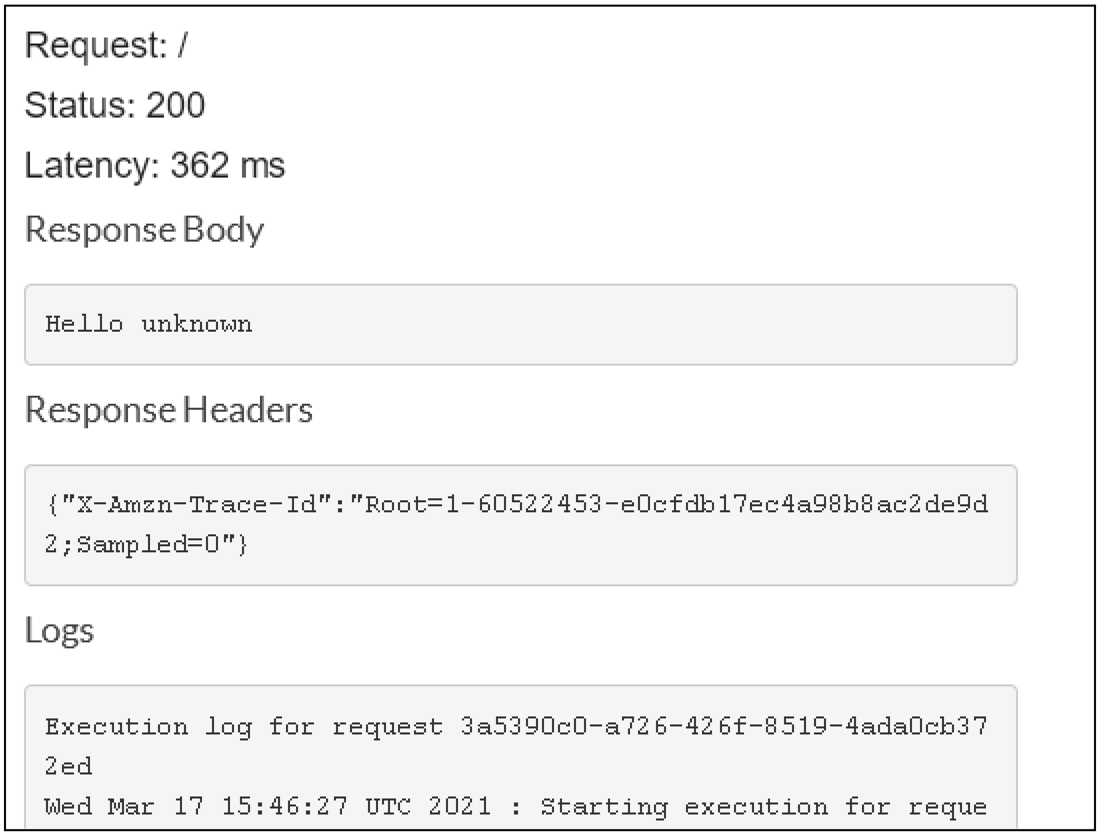
HTTP Status: Proxy
Models: application/json => Empty

Proxy integrations cannot be configured to transform responses.

13. Congratulations! You have now wired up your API to your Lambda.

Part Three – Testing Your API to Lambda Method

1. Click the **Test** link above the lightning bolt.
2. Click the blue **Test** button.
3. The right side of the page will show you the **Response Body, Response Headers**, and **API Gateway Logs**. It should look something like this:



Request: /

Status: 200

Latency: 362 ms

Response Body

```
Hello unknown
```

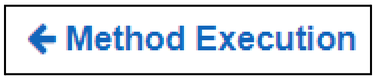
Response Headers

```
{"X-Amzn-Trace-Id":"Root=1-60522453-e0cfdb17ec4a98b8ac2de9d
2;Sampled=0"}
```

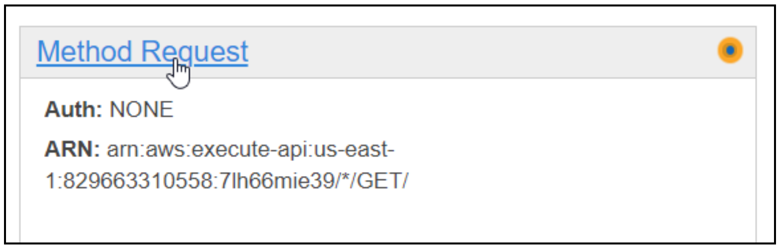
Logs

```
Execution log for request 3a5390c0-a726-426f-8519-4ada0cb37
2ed
Wed Mar 17 15:46:27 UTC 2021 : Starting execution for reque
```

4. It worked! But, not really. We didn't pass it a name to reply with, so it says "Hello unknown". Let's fix that.
5. Select the **Method Execution** link with the arrow pointing left, located in the upper right.



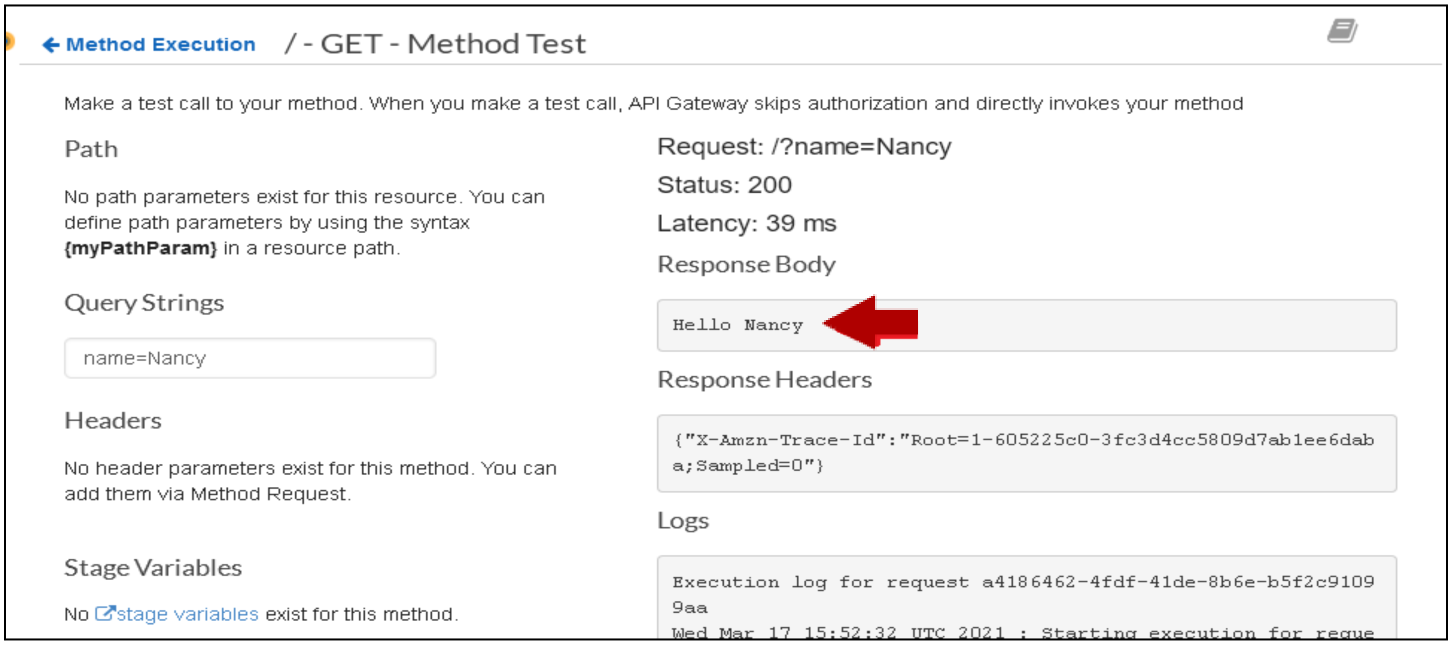
6. Select the **Method Request** box by clicking on its name.



Auth: NONE

ARN: arn:aws:execute-api:us-east-1:829663310558:7lh66mie39/*/GET/

7. Open the **URL Query String Parameters** section.
8. Click the **Plus (+) Add Query** string.
9. Enter “name” (without quotes) in the input box and click the grey check mark on the right side.
10. Select the **Method Execution** link in the upper right.
11. Click **Test**.
12. Now you can enter a query string for your test. Type “name=Nancy” (without quotes).
13. Click the **Test** button.



Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

name=Nancy

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No stage variables exist for this method.

Request: /?name=Nancy

Status: 200

Latency: 39 ms

Response Body

```
Hello Nancy
```

Response Headers

```
{"X-Amzn-Trace-Id":"Root=1-605225c0-3fc3d4cc5809d7ab1ee6dab
a;Sampled=0"}
```

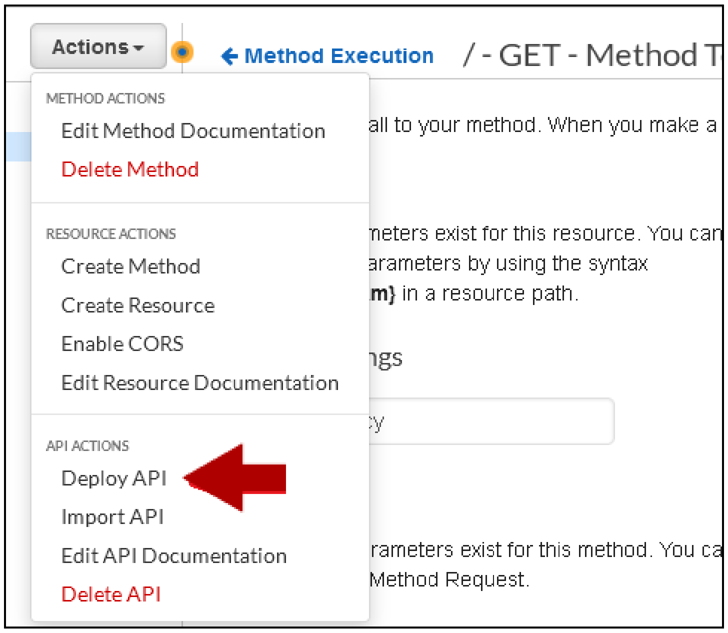
Logs

```
Execution log for request a4186462-4fdf-41de-8b6e-b5f2c9109
9aa
Wed Mar 17 15:52:32 UTC 2021 : Starting execution for reque
```

14. Success. You have now tested your API with a parameter calling your Lambda.

Part Four – Deploying Your API

1. You have a shiny new API, but it is not yet callable from outside of AWS. We need to deploy it to allow that. To deploy your API, select **Deploy API** from the **Actions** drop-down menu.



Actions ▾ ← Method Execution / - GET - Method T

METHOD ACTIONS
Edit Method Documentation
Delete Method

RESOURCE ACTIONS
Create Method
Create Resource
Enable CORS
Edit Resource Documentation

API ACTIONS
Deploy API
Import API
Edit API Documentation
Delete API

all to your method. When you make a

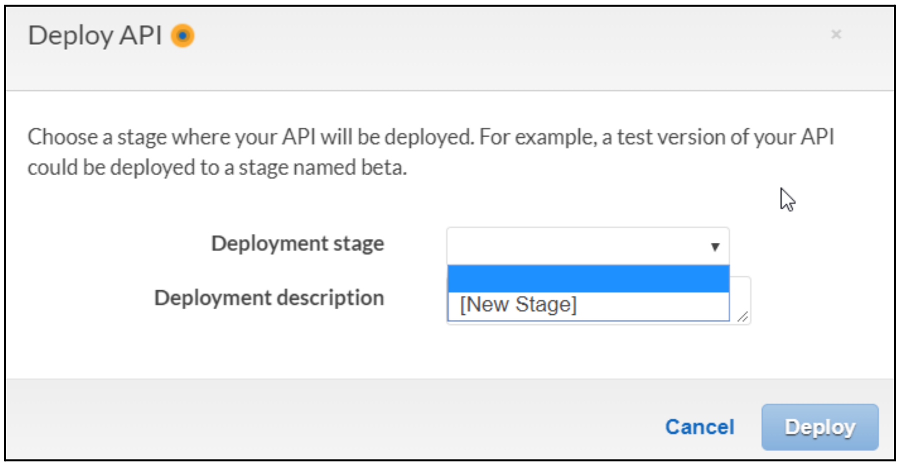
meters exist for this resource. You can
arameters by using the syntax
m} in a resource path.

ngs

cy

rameters exist for this method. You ca
Method Request.

2. Select **New Stage** in the **Deployment stage** drop-down menu.



Deploy API

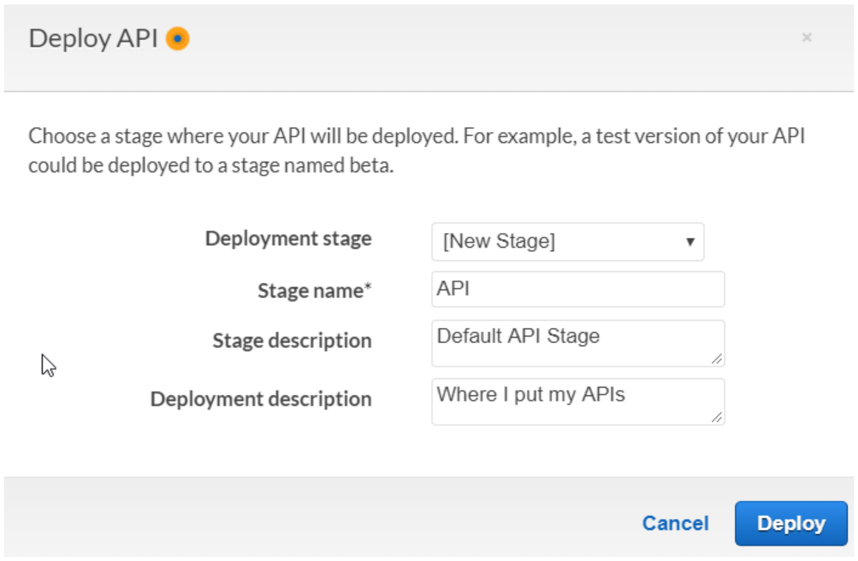
Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage ▾
[New Stage]

Deployment description

Cancel Deploy

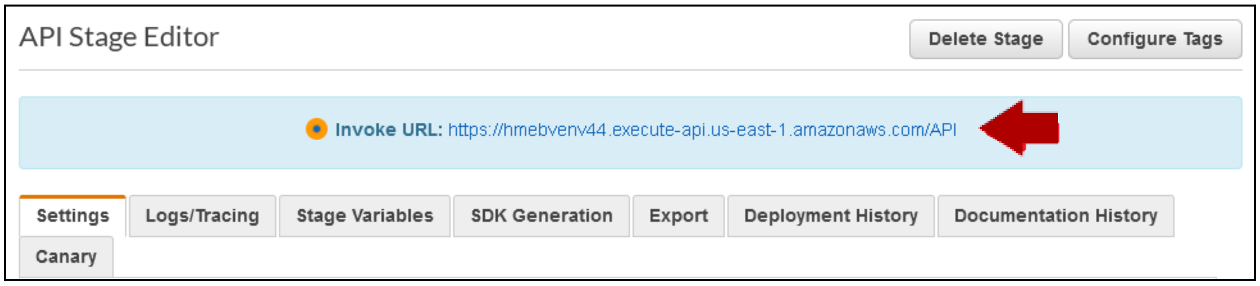
3. Type “API” as your stage name. The two description fields are optional.


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.

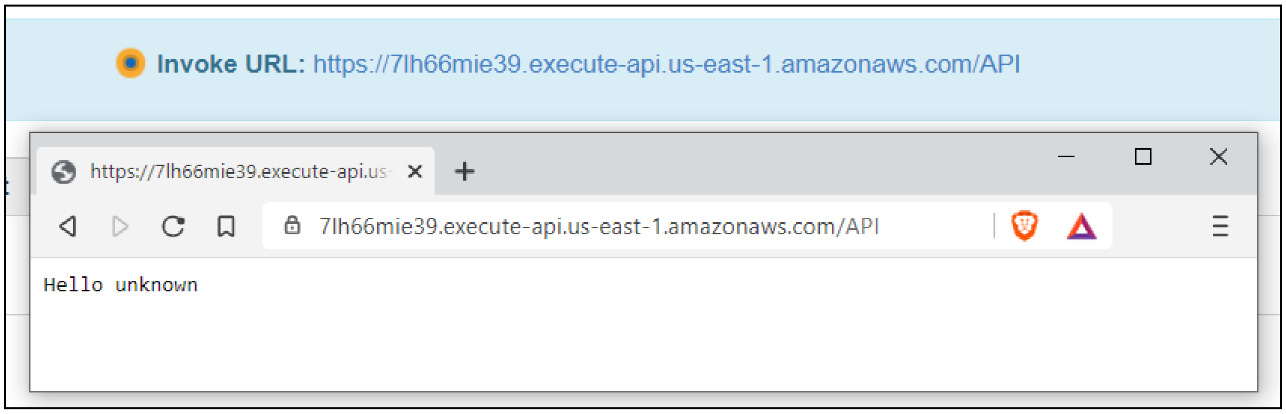
Deployment stage	[New Stage]
Stage name*	API
Stage description	Default API Stage
Deployment description	Where I put my APIs

4. Click **Deploy.**
5. A few seconds later, your API will be deployed, and you will have your shiny new URL.


API Stage Editor

Invoke URL: https://hmebvenv44.execute-api.us-east-1.amazonaws.com/API

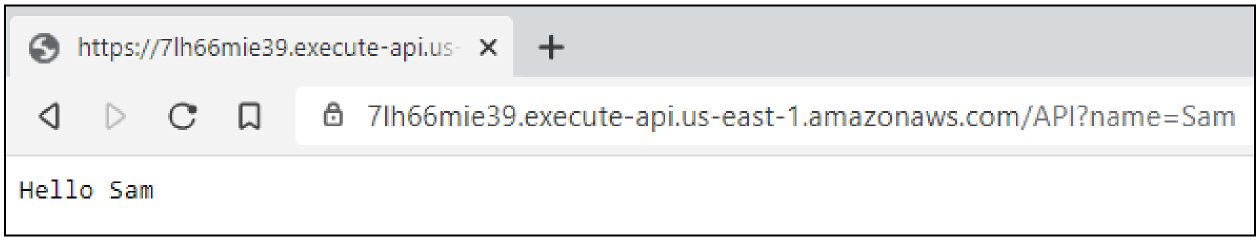
6. If you navigate to the URL, it will work in your browser. (Make sure you are using a browser other than Firefox, when you try this).


Invoke URL: https://7lh66mie39.execute-api.us-east-1.amazonaws.com/API

7lh66mie39.execute-api.us-east-1.amazonaws.com/API

```
Hello unknown
```

7. You can put in a query parameter to make the message more personalized.


https://7lh66mie39.execute-api.us-
7lh66mie39.execute-api.us-east-1.amazonaws.com/API?name=Sam

```
Hello Sam
```

That's it. You have created a Lambda, tested it, created an API, connected it to your Lambda, and tested it again. Finally, you deployed your API. Your project will build upon these functions to connect your Angular application to new Lambdas you will write, which will connect to the database we will learn about in the next module.

Great work!