

Hello World: AWS Lambda & Gateway Services en Java

En este tutorial presentamos una introducción a microservicios con Java sobre la plataforma de AWS Lambda y AWS Gateway.

Para presentar los conceptos implementamos un servicio REST que calcula el cuadrado de un entero. Los pasos que sigue el tutorial son los siguientes:

1. Crear una clase que implemente el servicio deseado. Para esto usamos Java 8, Maven y Netbeans como IDE.
2. Luego creamos la función lambda y cargamos las clases desarrolladas.
3. Luego configuramos el Gateway para exponer la función.

Crear la clase y método que implementa el servicio

Para crear el servicio creamos un proyecto java con maven. En el proyecto implementamos una clase que implementa un método estático que reciba un entero y que retorne el cuadrado del mismo. Usamos este ejemplo simple para evitar dependencias innecesarias y para que la persona que lea el tutorial pueda ver la mecánica de las funciones lambda.

El código de la función es:

```
package co.edu.escuelaing.services;

public class MathServices {
    public static Integer square(Integer i){
        return i*i;
    }
}
```

Una vez creada la podemos compilar y empaquetar la función para crear un jar.

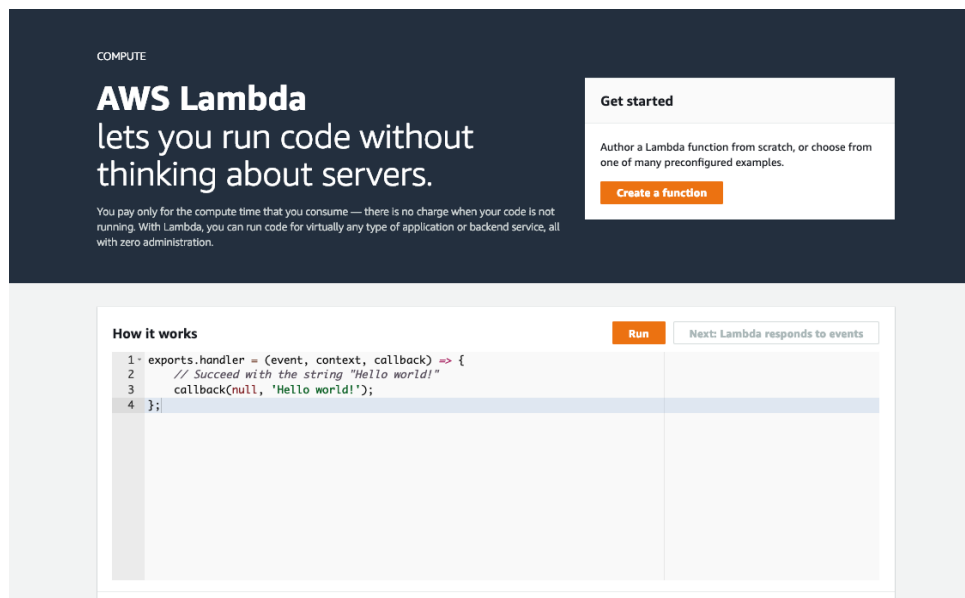
```
$ mvn package
```

El archivo Jar que se cree es el que vamos a cargar en la función lambda.

Cree la función lambda

En la consola de AWS seleccione el servicio Lambda y cree una función siguiendo estos pasos:

1. Cree un ROL en el servicio IAM de AWS (Opcional). Puede solicitar crear el rol automáticamente al crear la función. Este rol debe tener acceso a cloud watch. Este rol sirve para que la función tenga permisos de accesos.
2. Acceda al servicio Lambda



1. Oprima el botón de crear una función. Cree la función desde el inicio (From scratch)
2. Asígnele un nombre, por ejemplo "square"
3. Seleccione el "Runtime" a Java 8
4. Seleccione usar un Rol existente y use el rol que creó en el paso 1.

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.

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

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

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.

Ahora vamos a cargar el código y a probar que funcione:

1. En la sección "Function code" cargue el código
2. En el campo "Handler" escriba: {ruta de la clase incluyendo el paquete}:: {Nombre del método}. Es decir:

```
co.edu.escuelaing.services.MathServices::square
```

Function code

Code entry type

Runtime

Handler [Info](#)

Function package

For files larger than 10 MB, consider uploading using Amazon S3.

1. Oprima el botón de guardar

Ahora vamos a probar que funcione:

1. En la parte superior en el cuadro de selección que dice "Select a test event" seleccione "Configure test events".
2. Asigne un nombre de evento, por ejemplo "testSquare"

3. En el cuadro de texto, borre el JSON que aparece y solo deje el número 5
4. Oprima el botón de crear.

Configure test event ✕

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☒ Create new test event
☐ Edit saved test events

Event template

Hello World ▼

Event name

testSquare

1

5

Cancel

Create

1. Una vez creado ya puede oprimir el boton de "Test" y debería obtener un resultado similar a este:

to test your function.

Lambda > Functions > square

ARN - arn:aws:lambda:us-east-1:334102246988:function:square

square

Throttle Qualifiers Actions testSquare Test Save

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

25

Summary

Code SHA-256	Request ID
BuXqf6Dnjq80AExDtr/9eDJB6RUUPKo9LpZPXA1z8zs=	670512ca-292e-4dc7-beb0-219f48d34226
Duration	Billed duration
134.25 ms	200 ms
Resources configured	Max memory used
512 MB	87 MB

Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: 670512ca-292e-4dc7-beb0-219f48d34226 Version: $LATEST
END RequestId: 670512ca-292e-4dc7-beb0-219f48d34226
REPORT RequestId: 670512ca-292e-4dc7-beb0-219f48d34226 Duration: 134.25 ms Billed Duration: 200 ms Memory
Size: 512 MB Max Memory Used: 87 MB
```

Es importante ver que esta función no la probamos con un JSON sino con un solo valor de 5. Esto es importante para mapear la función en el siguiente paso.

Configurar el API Gateway para exponer el servicio

1. Abra el servicio de API Gateway



Amazon API Gateway

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.

[Get Started](#)

[Getting Started Guide](#)



Streamline API development

Amazon API Gateway lets you simultaneously run multiple versions and release stages of the same API, allowing you to quickly iterate, test, and release new versions.

[Learn More](#)



Performance at scale

Amazon API Gateway helps you improve performance by managing traffic to your existing back-end systems, throttling API call spikes, and enabling result caching.

[Learn More](#)



SDK generation

Amazon API Gateway can generate client SDKs for JavaScript, iOS, and Android, which you can use to quickly test new APIs from your applications and distribute SDKs to third-party developers.

[Learn More](#)

1. Oprima el botón de "Getting Started"
2. Seleccione las opciones de API REST, New AP

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

[Import](#)

[Build](#)

1. Asígnale un nombre al API, ejemplo, mathServices
2. Y seleccione el end point de tipo regional.
- 3.

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

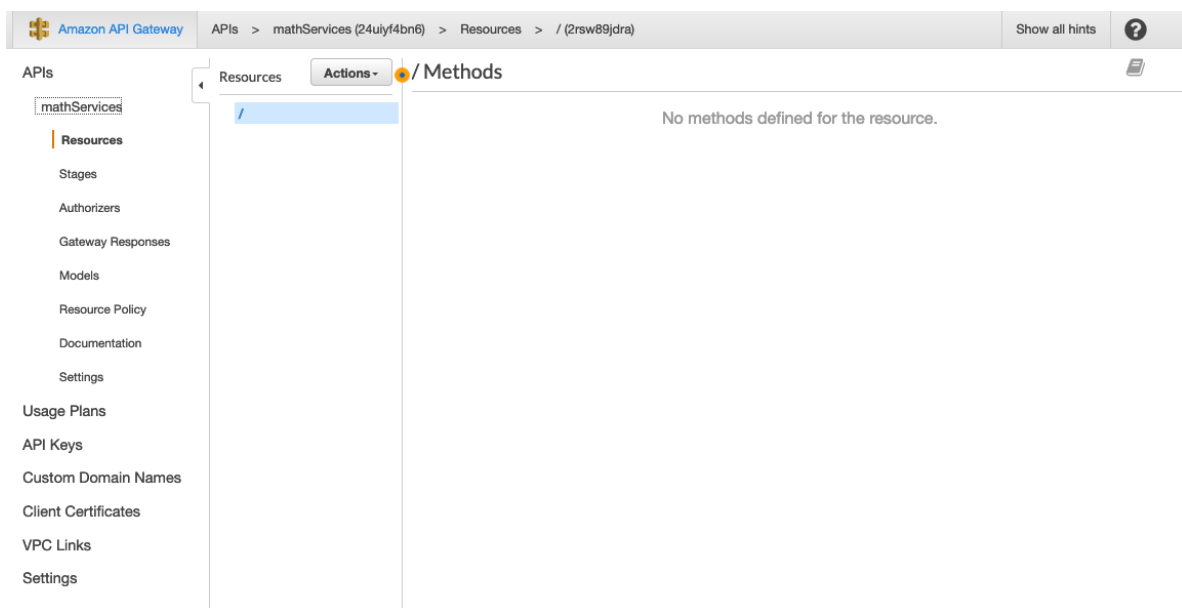
Choose a friendly name and description for your API.

API name*
Description
Endpoint Type ⓘ

* Required

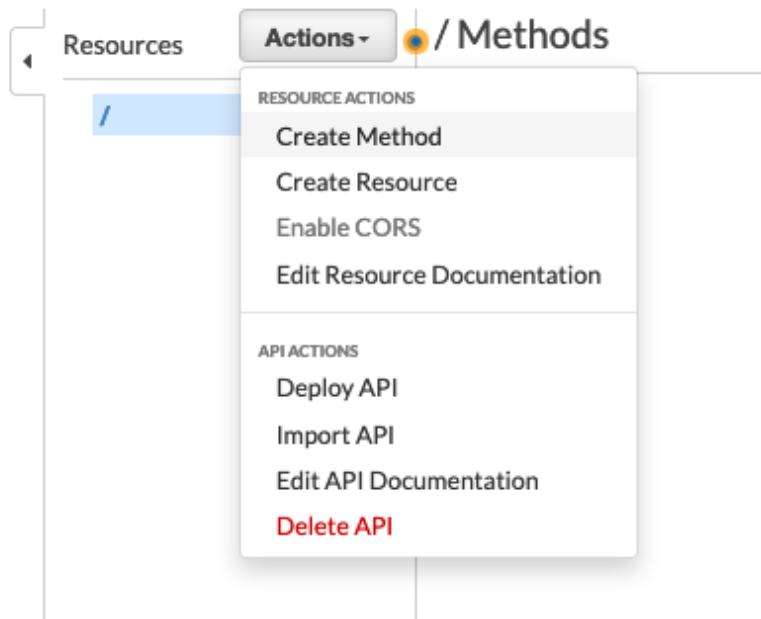
Create API

Le debe aparecer algo así

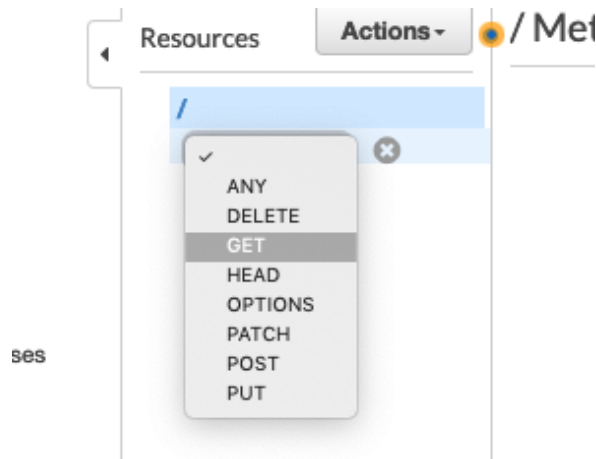


Ahora vamos a crear una forma de acceder a este API. Para esto vamos a crear un método http GET que reciba la solicitud e invoque nuestra función:

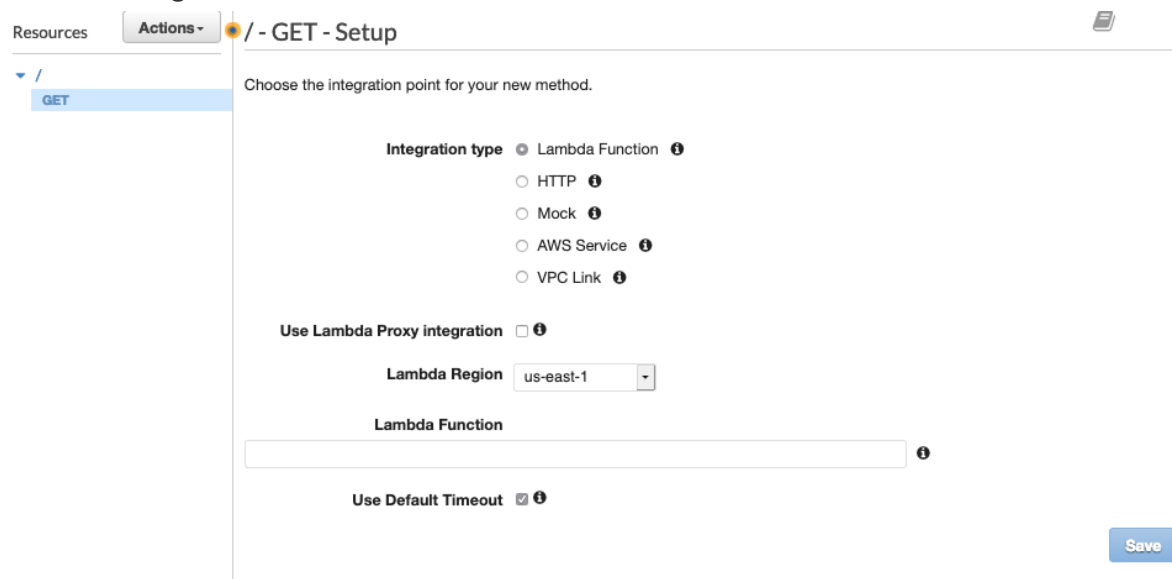
1. En el API de mathServices en el botón de acciones al lado de recurso seleccione "Create Method"



1. Seleccione el método GET y confírmelo oprimiendo el chulo.



Debe ver algo así:



1. En el campo de "Lambda Function" escriba el nombre de la función que creó en el paso anterior y salve.

Resources Actions ▾ / - GET - Setup

Choose the integration point for your new method.

Integration type ☒ Lambda Function ⓘ
☐ HTTP ⓘ
☐ Mock ⓘ
☐ AWS Service ⓘ
☐ VPC Link ⓘ

Use Lambda Proxy integration ☐ ⓘ

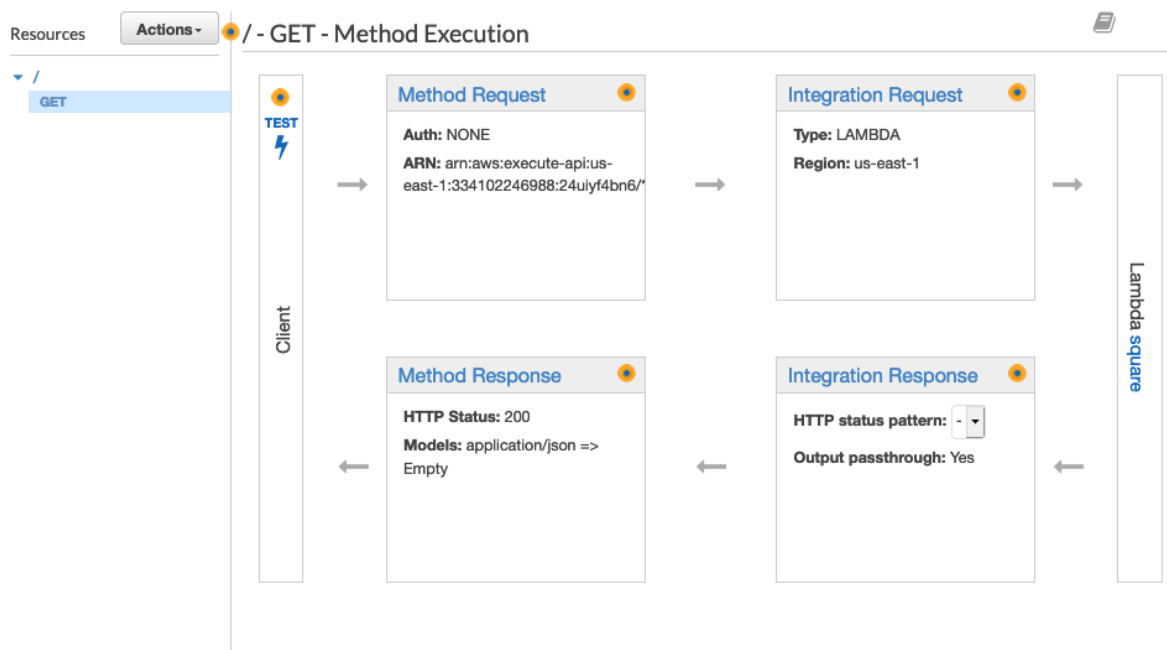
Lambda Region us-east-1 ▾

Lambda Function
square ⓘ

Use Default Timeout ☒ ⓘ

Save

Al crealo debes ver algo así:



1. Haga Click sobre Method Request



1. Agregue el parametro "value" en "URL Query String Parameters":

Resources Actions ▾

← Method Execution / - GET - Method Request

GET

Provide information about this method's authorization settings and the parameters it can receive.

Settings

Authorization NONE ⓘ

Request Validator NONE ⓘ

API Key Required false ⓘ

URL Query String Parameters ⓘ

Name	Required	Caching	
value	<input type="checkbox"/>	<input type="checkbox"/>	ⓘ ⊕

+ Add query string

HTTP Request Headers

Request Body ⓘ

SDK Settings

1. Regrese a la definición del método oprimiendo ""Method Execution"

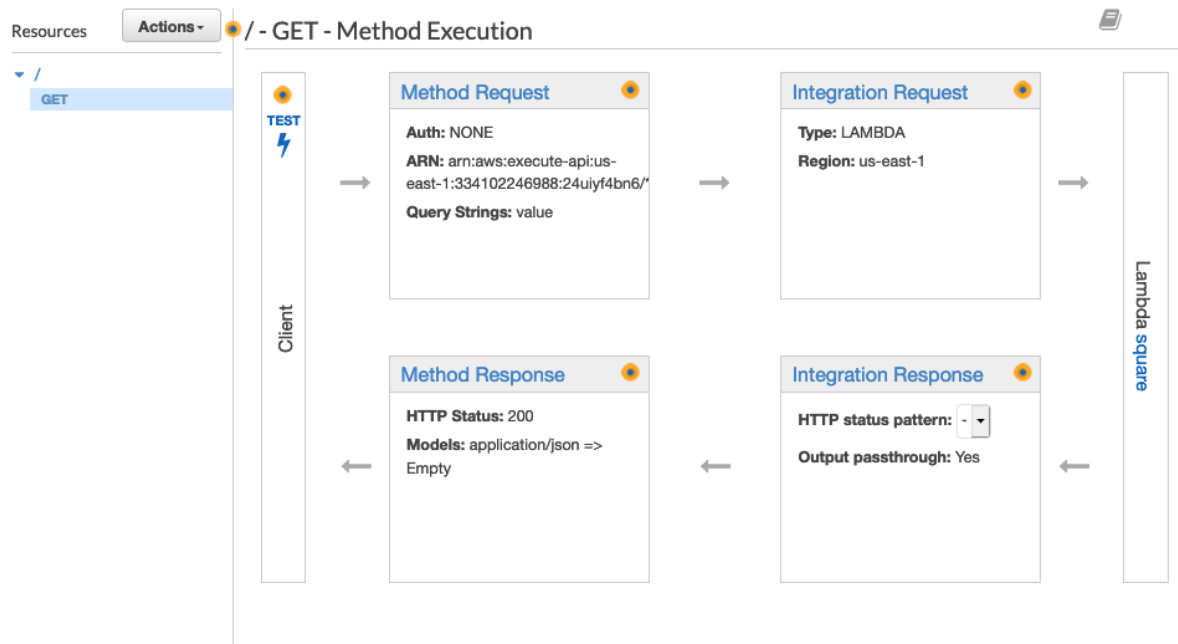
Resources Actions ▾

← Method Execution / - GET - Method Request

GET

Provide information about this method's authorization settings and the par

Debe ver nuevamente:



1. Ingrese a "Integration Request"

Resources Actions ▾ ← Method Execution / - GET - Integration Request

GET

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type ☒ Lambda Function ⓘ

- ☐ HTTP ⓘ
- ☐ Mock ⓘ
- ☐ AWS Service ⓘ
- ☐ VPC Link ⓘ

Use Lambda Proxy integration ☐ ⓘ

Lambda Region us-east-1 ✎

Lambda Function square ✎

Execution role ✎

Invoke with caller credentials ☐ ⓘ

Credentials cache Do not add caller credentials to cache key ✎

Use Default Timeout ☒ ⓘ

▸ URL Path Parameters



▸ URL Query String Parameters

▸ HTTP Headers

▸ Mapping Templates ⚡

1. Mapee el parámetro que identificó en el paso anterior para poder usarlo. Para esto cree en "URL Query String Parameters" un parámetro denominado "value" que obtenga el valor del anterior que se había extraído de la cadena de query. PARA esto debe mapearlo a "method.request.querystring.value" así:

▼ URL Query String Parameters

Name	Mapped from ⓘ	Caching	
value	method.request.querystring.value	<input type="checkbox"/>	 


+ Add query string

▼ HTTP Headers

1. Ahora configure un mapping template para indicar cómo se pasarán los parámetros a la función lambda. OJO: Por defecto muchas de las opciones y ejemplo en internet intentan pasar un objeto JSON con los parámetros, esto es muy útil. Pero para efectos del tutorial vamos a pasar solo un número.
2. Adicione una mapping template así:

▼ Mapping Templates 🟡

Request body passthrough ☐ When no template matches the request Content-Type header ⓘ
☒ When there are no templates defined (recommended) ⓘ
☐ Never ⓘ



Content-Type	
application/json	

+ Add mapping template

application/json

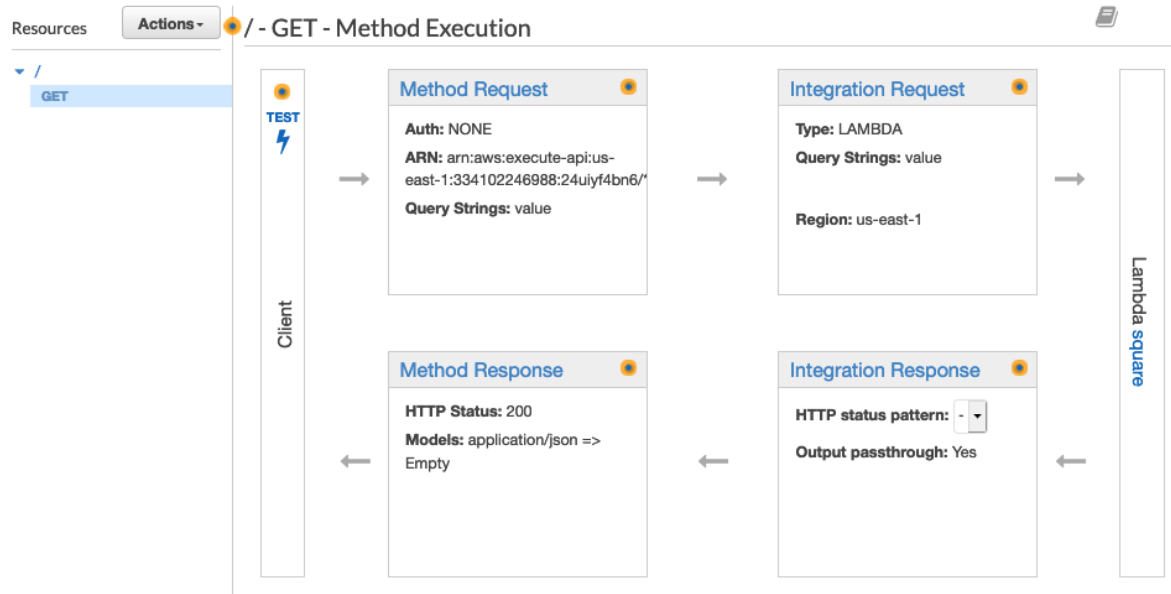
Generate template:

1 5

Cancel Save

1. Mire que asignamos un solo valor por ahora, esto para que entienda que esto es los que queremos construir, esto es los que le vamos a pasar la función lambda. Usted ya puede probar la función:
2. Saliendo a la definición del método,



1. Luego entra al test y ejecútelo

Resources

Actions ▾

▼ /
GET

← Method Execution / - GET - Method Test

Make a test call to your method with the provided input

Path

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

Query Strings

param1=value1¶m2=value2

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 Body

Request Body is not supported for GET methods.

⚡ Test

1. El resultado siempre será el mismo porque estamos pasando una constante

Resources
Actions
Method Execution / - GET - Method Test

GET

Make a test call to your method with the provided input

Path

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

Query Strings

param1=value1¶m2=value2

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 Body

Request Body is not supported for GET methods.

Test

Request: /

Status: 200

Latency: 45 ms

Response Body

25

Response Headers

```
{
  "X-Amzn-Trace-Id": "Root=1-5c7f05cc-e72739de2b59e16bdbcb6a1e;Sampled=0",
  "Content-Type": "application/json"
}
```

Logs

```
Execution log for request 31e2a3ba-3f9e-11e9-b18e-277934cdab6b
Tue Mar 05 23:27:08 UTC 2019 : Starting execution for request: 31e2a3ba-3f9e-11e9-b18e-277934cdab6b
Tue Mar 05 23:27:08 UTC 2019 : HTTP Method: GET, Resource Path: /
Tue Mar 05 23:27:08 UTC 2019 : Method request path: {}
Tue Mar 05 23:27:08 UTC 2019 : Method request query string: {}
Tue Mar 05 23:27:08 UTC 2019 : Method request headers: {}
Tue Mar 05 23:27:08 UTC 2019 : Method request body before transformations:
Tue Mar 05 23:27:08 UTC 2019 : Endpoint request URI: https://lambda.us-east-1.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-east-1:334102246...
```

- Ahora vamos a modificar el "Template de mapeo" para que podamos pasar parámetros. En el template la variable \$ input representa la carga útil de entrada y los parámetros que debe procesar su template. Puede encontrar la documentación de los templates de Mapeo en : <https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-mapping-template-reference.html>

Por ahora lo que necesita es extraer el valor del parámetro que configuró arriba. Escriba lo siguiente en la caja de texto para configurar el mapeo.

```
$input.params("value")
```

Debería obter una salida así:

▼ Mapping Templates

Request body passthrough ☐ When no template matches the request Content-Type header ⓘ
☒ When there are no templates defined (recommended) ⓘ
☐ Never ⓘ

Content-Type	
application/json	⊖

⊕ [Add mapping template](#)

application/json

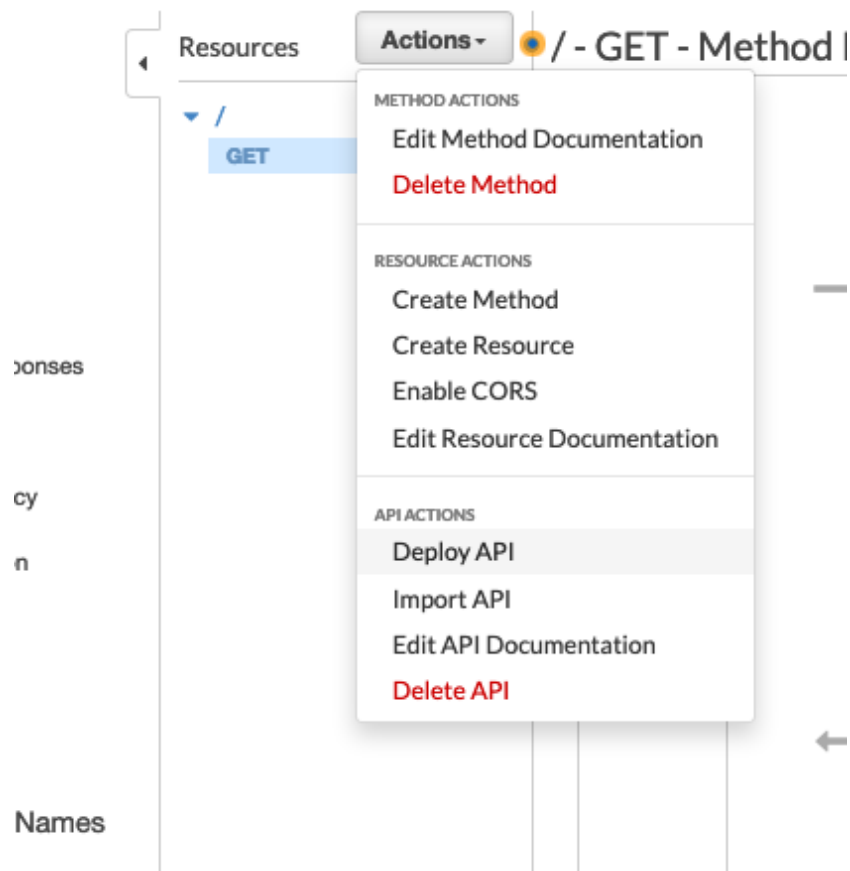
Generate template:

```
1 $input.params("value")
```

Cancel

Save

1. Ahora pruebe nuevamente la funcionalidad. no olvide incluir una parámetro "value" con un valor entero en la prueba. si no lo incluye se debe reportar un error. Antes de hacer la prueba usted verá algo así:



1. En la ventana que aparece seleccione la opción de "[New Stage]" y dele un nombre cualquiera, por ejemplo Beta. Aquí podrá manejar otras etapas como pruebas y producción.

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*	Beta
Stage description	Beta stage
Deployment description	Beta deployment of Math services

Cancel
Deploy

1. El resultado debe ser como se muestra a continuación:

Stages **Create** Beta Stage Editor Delete Stage

Invoke URL: <https://24uiyf4bn6.execute-api.us-east-1.amazonaws.com/Beta>

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](#)

Enable throttling ☒

Rate requests per second

Burst requests

Web Application Firewall (WAF) [Learn more.](#)

Select the Web ACL to be applied to this stage.

Web ACL [Create Web ACL](#)

Client Certificate

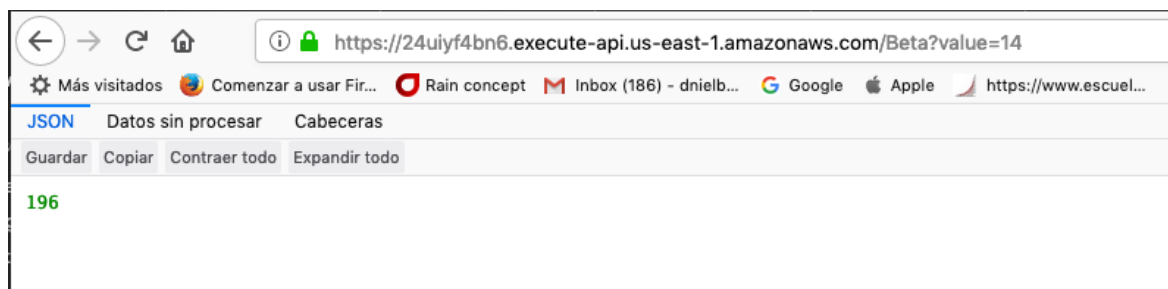
Select the client certificate that API Gateway will use to call your integration endpoints in this stage.

Certificate

Taas

1. La URL que aparece en la parte superior le sirve para invocar el servicio: Intente invocarlo con una URL como esta (no olvide el parámetro):
<https://24uiyf4bn6.execute-api.us-east-1.amazonaws.com/Beta?value=14>

Y deberá obtener una resultado com este:



1. No olvide borrar la función Lambda y el API para que no le genere gastos innecesarios.

Esta introducción le debe mostrar la forma básica de exponer servicios en AWS usando AWS Lambda y el API Gateway.

Sin embargo usted usará formas de mapeo y estructuras de datos más complejas basadas en objetos JSON. En general las plantillas de Mapeo permiten mapear los mensajes que van a la función o los mensajes que salen de la función. En la va la entrada de una función se considera el primer parámetro de la función. En java en particular se soporta lo siguiente:

El inputType y el outputType de una función pueden ser uno de los siguientes:

1. Tipos de Java primitivos (como String o int).
2. Tipos de eventos de AWS predefinidos definidos en la biblioteca aws-lambda-java-events. Por ejemplo, S3Event es uno de los POJO predefinidos en la biblioteca que proporciona métodos para que pueda leer fácilmente la información del evento Amazon S3 entrante.
3. **También puedes escribir su propia clase de POJO. AWS Lambda serializará y deserializará automáticamente la entrada y salida JSON según el tipo de POJO.**