

## Paso 1

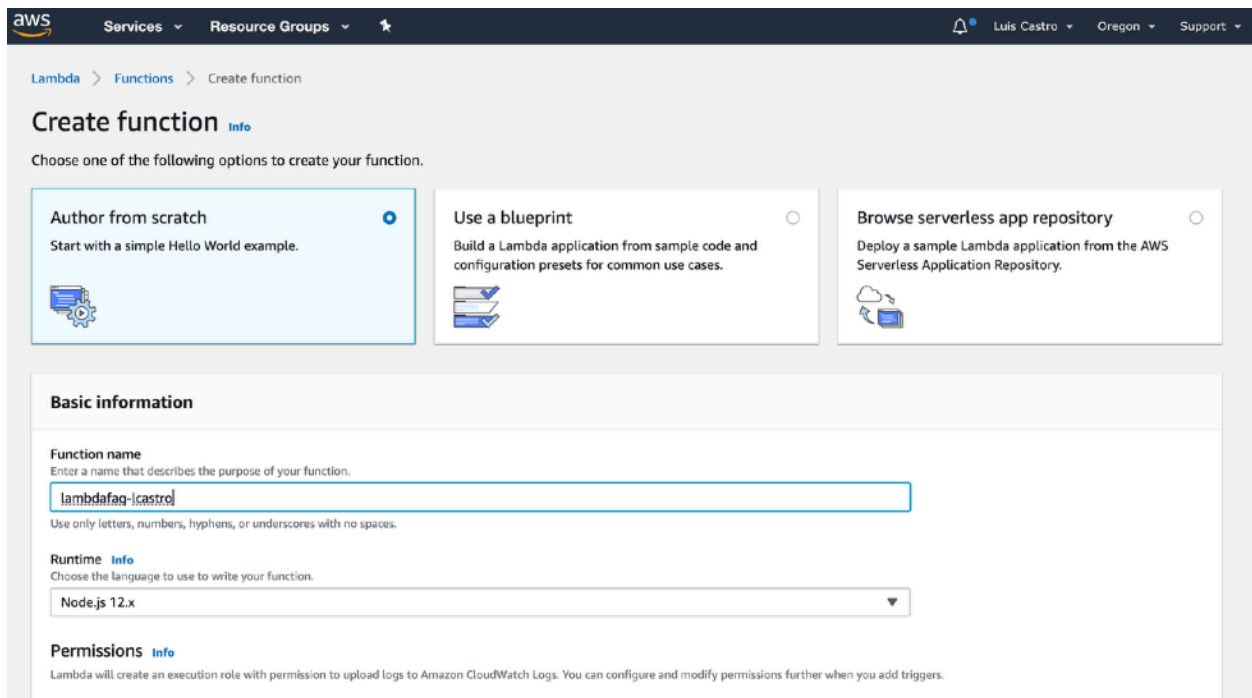
Acceder a la consola de AWS mediante el siguiente link:

<https://lcastrose.signin.aws.amazon.com/console>

## Paso 2

Ingresar al servicio de **Lambda** y crear una función con la siguiente nomenclatura:

- **lambdafaq-<nombre de usuario>**
- Ejemplo:
  - lambdafaq-lcastro
- Escoger el Runtime
  - **Node-.js 10.x**
- Choose or create an execution role
  - Use an existing role
    - **lambda\_basic\_execution**
- **Create Function**



The screenshot shows the AWS Lambda 'Create function' page. At the top, there's a navigation bar with 'aws', 'Services', 'Resource Groups', and a user profile 'Luis Castro' in 'Oregon'. The breadcrumb trail is 'Lambda > Functions > Create function'. The main heading is 'Create function' with an 'Info' link. Below it, a prompt says 'Choose one of the following options to create your function.' There are three options: 'Author from scratch' (selected with a blue circle), 'Use a blueprint', and 'Browse serverless app repository'. The 'Author from scratch' option has a description 'Start with a simple Hello World example.' and a gear icon. Below the options is the 'Basic information' section. It has a 'Function name' field with the value 'lambdafaq-lcastro' and a description 'Enter a name that describes the purpose of your function. Use only letters, numbers, hyphens, or underscores with no spaces.' Below that is the 'Runtime' section with a dropdown menu showing 'Node.js 12.x' and a description 'Choose the language to use to write your function.' At the bottom is the 'Permissions' section with a description: '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.'

### Paso 3

Abrir el archivo llamado **faqlambda.txt**, copiarlo y seguidamente pegarlo como **Code Entry Type**. Borrar el código actual dentro del **Index.js**



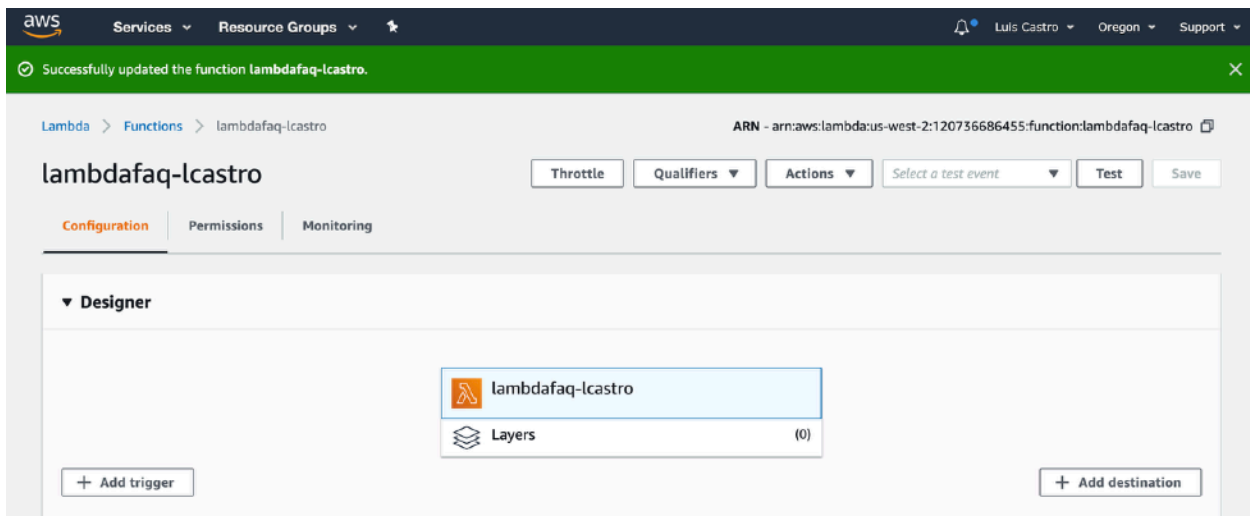
### Paso 4

Escoger las siguientes opciones

Handler

- Index.handler

Escoger los demás valores default y hacer click en **Save**



Input test event

It looks like you have not configured a test event for this function yet. Use the editor below to enter an event to test your function with (please remember that this will actually execute the code!). You can always edit the event later by choosing **Configure test event** in the Actions list. Note that changes to the event will only be saved locally.

Sample event template

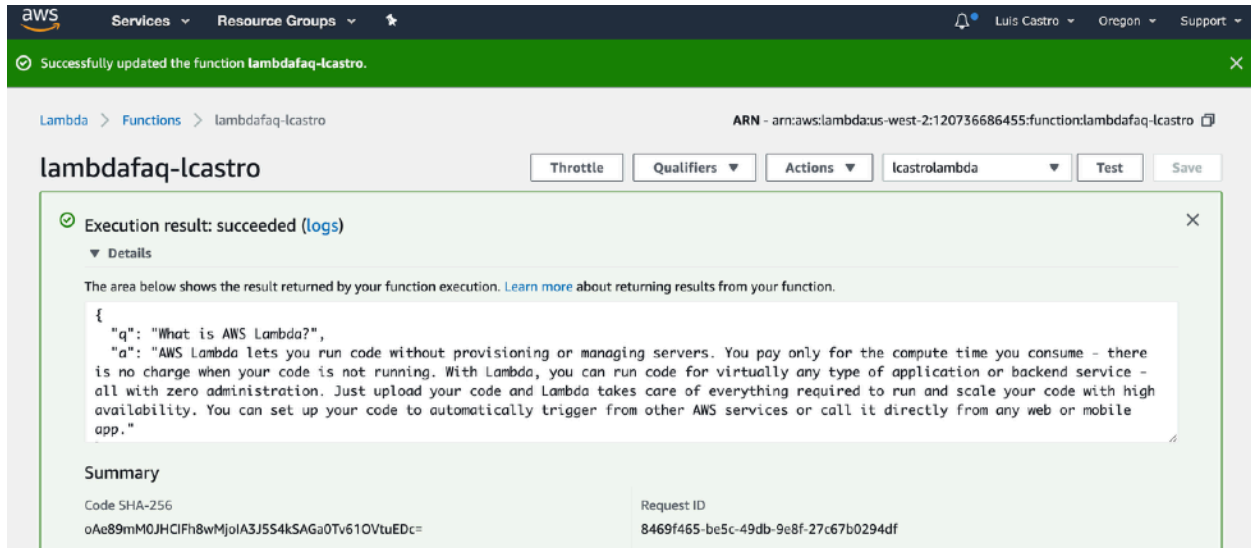
Hello World

1

## Paso 6

Escoger el test creado anteriormente (Ej: lcastrolambda) y hacer click en **Test**

Execution result: Succeeded



Execution result: succeeded (logs)

Details

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

```
{
  "q": "What is AWS Lambda?",
  "a": "AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app."
}
```

Summary

Code SHA-256: oAe89mM0JHCiFh8wMjolA3J5S4kSAGa0Tv61OVtuEDc=

Request ID: 8469f465-be5c-49db-9e8f-27c67b0294df

## Paso 10

Verificar en el **Summary** de ejecución los siguientes parámetros

- **Duration**
  - 9,40 ms
- **Billed Duration**
  - 100 ms
- **Max Mem Used**
  - 36 MB



Summary

Code SHA-256: Hv9wEvH8BNLNUcdDUGK83Uvmbf0CgAQCSHSKv988=

Request ID: 99a8a347-1c61-11e6-ac78-8d0f1e410f73

Duration: 9.40 ms

Billed duration: 100 ms

Resources configured: 128 MB

Max memory used: 36 MB

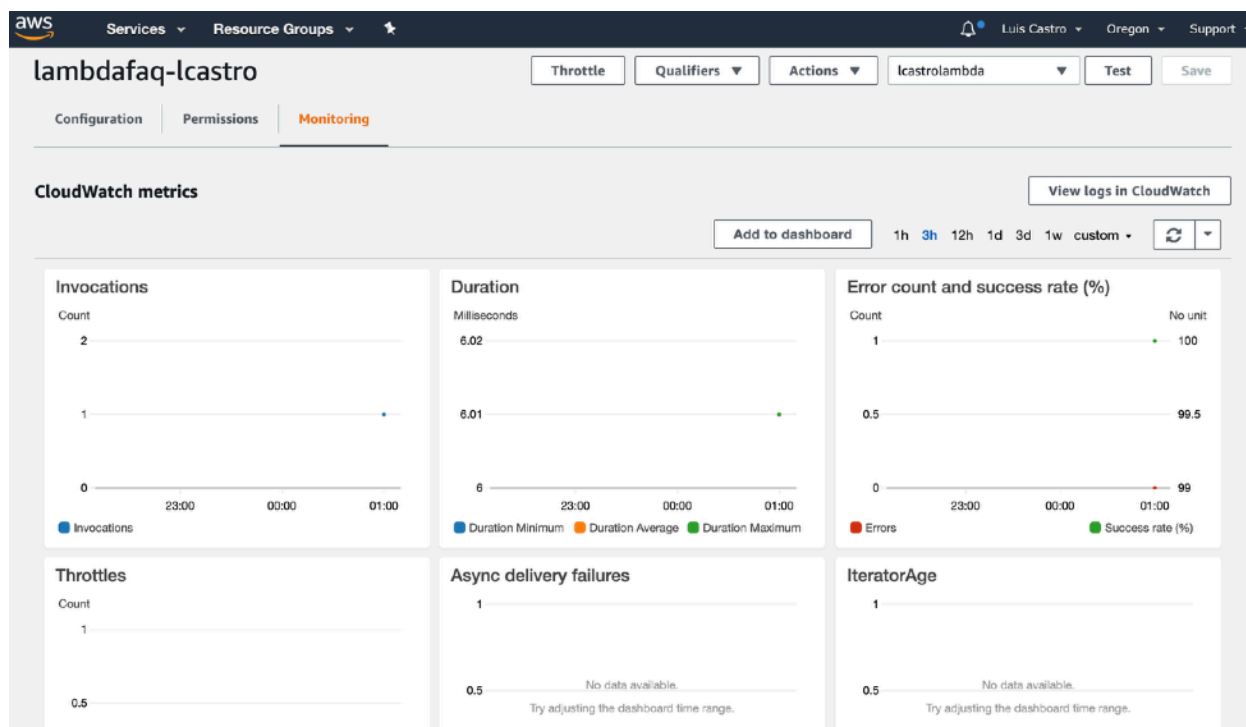
Log output

The area 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: 99a8a347-1c61-11e6-ac78-8d0f1e410f73 Version: $LATEST
2016-05-17T19:00:23.344Z 99a8a347-1c61-11e6-ac78-8d0f1e410f73 Quote selected: 10
END RequestId: 99a8a347-1c61-11e6-ac78-8d0f1e410f73
REPORT RequestId: 99a8a347-1c61-11e6-ac78-8d0f1e410f73 Duration: 9.40 ms Billed Duration: 100 ms Memory S
```

## Paso 11

Hacer click en “Monitoring” y verificar las alarmas de CloudWatch



## Paso 12

Entrar al Designer de la función creada y hacer click en **Add Trigger**

Escoger los siguientes valores:

1. API Gateway
2. Create an API
3. API Type: REST API
4. Security: Open
5. Additional Settings
  - API Name: dejar el valor default
  - Deployment Stage: default
6. Add

Hacer click en el API Endpoint Generado

**API Gateway**

**lambdafaq-lcastro-API**  
arn:aws:execute-api:us-west-2:120736686455:7glqhh8hu6/\*/\*/\*lambdafaq-lcastro

**Details**  
API: api-gateway/7glqhh8hu6/\*/\*/\*lambdafaq-lcastro  
API endpoint: <https://7glqhh8hu6.execute-api.us-west-2.amazonaws.com/dev/lambdafaq-lcastro>  
API name: lambdafaq-lcastro-API  
API type: rest  
Authorization: **NONE**  
Enable metrics and error logging: **No**  
Method: **ANY**  
Resource path: /lambdafaq-lcastro  
Security: **NONE**  
Stage: **dev**

### Paso 13

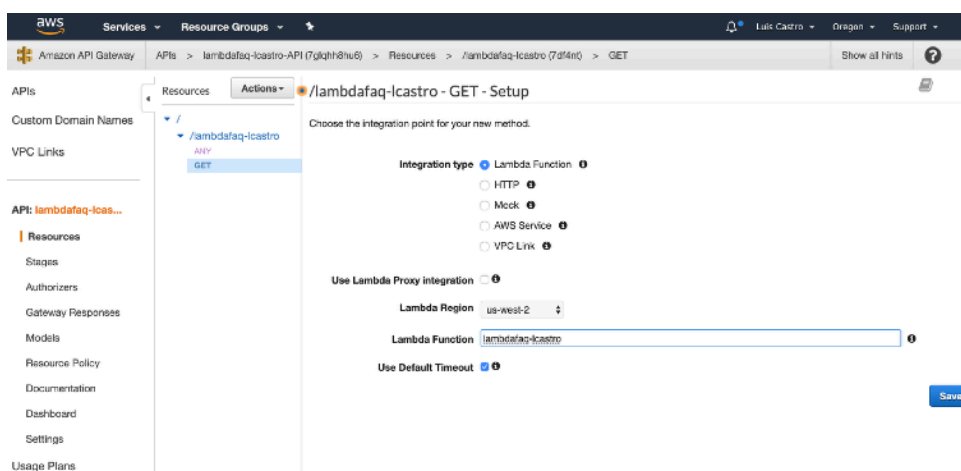
Ir al servicio de **API Gateway** y hacer click en el API Creado

En **Resources** escoger la función lambda creada y hacer click en **Actions>Create Method**

Escoger **GET** y hacer Click en el **Check Mark**

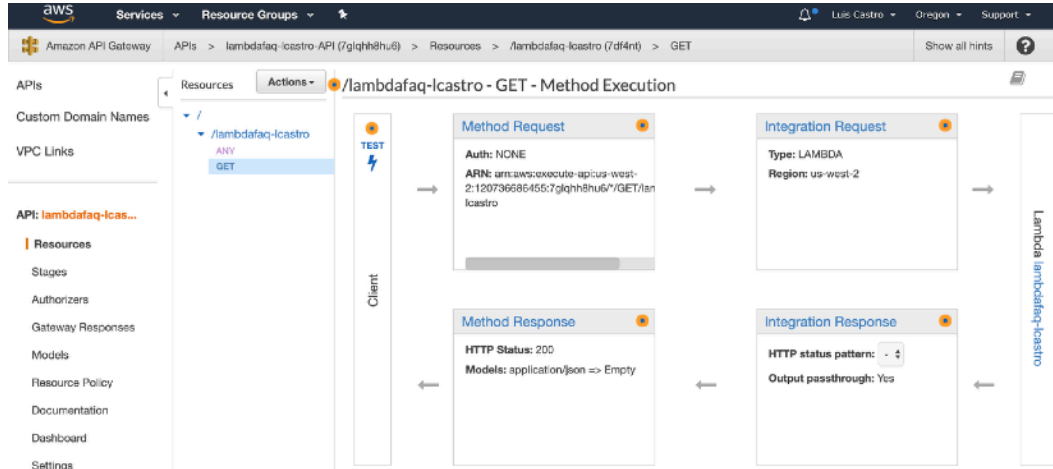
Escoger los siguientes valores:

1. Integration Type: **Lambda Function**
2. Use Lambda Proxy Integration: **Dejar sin marcar**
3. Lambda Region: **La region donde fue creada la Lambda**
4. Lambda Function: **Escribir el nombre de la función Lambda**
5. **Save**



## Paso 14

Probar que el API Gateway responda sobre la función Lambda, haciendo Click en Test



## Paso 15

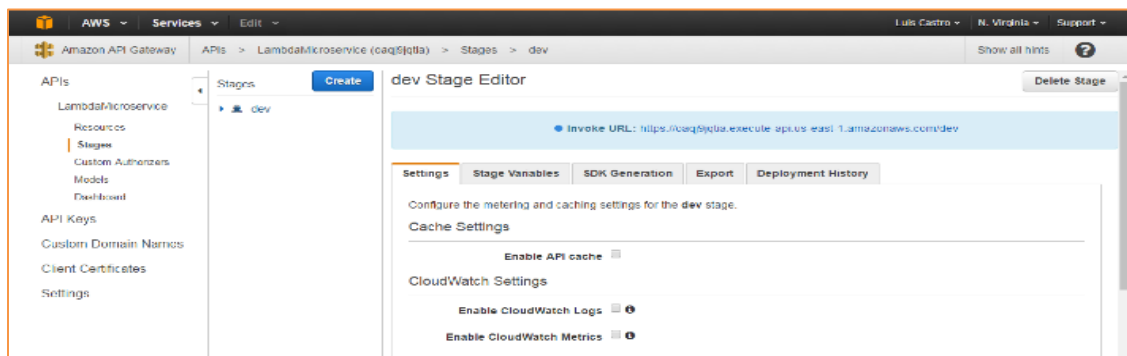
Validar que el Test regrese los valores esperados de la función Lambda

The screenshot shows the 'Method Test' results in the AWS API Gateway console. The breadcrumb navigation is: Services > Amazon API Gateway > APIs > lambdafaq-icastro-API (7g1qht8hu6) > Resources > /lambdafaq-icastro (7d4nt) > GET > Method Test. The left sidebar shows the API's resources, with the GET method selected. The main area displays the test results. On the left, there are sections for 'Path' (no parameters), 'Query Strings' (param1=value1&param2=value2), 'Headers' (Accept:application/json), 'Stage Variables' (none), and 'Request Body'. On the right, the 'Request' is '/lambdafaq-icastro', the 'Status' is '200', the 'Latency' is '36 ms', and the 'Response Body' is a JSON object: { "body": "{\\\"q\\\":\\\"Can I use threads and processes in my AWS Lambda function code?\\\",\\\"a\\\":\\\"Yes. AWS Lambda allows you to use normal language and operating system features, such as creating additional threads and processes. Resources allocated to the Lambda function, including memory, execution time, disk, and network use, must be shared among all the threads/processes it uses. You can launch processes using any language supported by Amazon Linux.\\\"}\"" }. Below this, the 'Response Headers' are shown: { "X-Amzn-Trace-Id": "Root=1-5e7a13f6-dc7a557844c89109e24a535c;Sampled=0", "Content-Type": "application/json" }. The 'Logs' section is also visible at the bottom.

## Paso 16

Seguidamente en el menú izquierdo marcar **Stages** y verificar el **Invoke URL** en un nuevo browser

- Debe de ser similar al siguiente
  - <https://oaj9jqtia.execute-api.us-east-1.amazonaws.com/dev>
  - Nota:
    - Si tiene errores a la hora de abrir el URL, agregue el nombre de la función creada (Ej: **lambdafaq-lcastro**) al final de la siguiente manera:
    - <https://oaj9jqtia.execute-api.us-east-1.amazonaws.com/dev/lambdafaq-lcastro>



## Paso 17

Verificar que el URL funcione apropiadamente mostrando información de FAQ, hacerlo varias veces y validar que cada vez muestre información diferente

