



aws Lambda

Guía Completa de 0 a 100



PABLO DEL ÁLAMO



¿Qué es AWS Lambda?

AWS Lambda es un servicio de computación sin servidor, que ejecuta tu código en respuesta a eventos y gestiona automáticamente los recursos.



PABLO DEL ÁLAMO



2 ¿Por qué usar Lambda?

Como ya sabrás, en arquitecturas monolíticas o de microservicios, si decides alojarlas en el cloud, estarás pagando a final de mes por la instancia o instancias (servidores) donde ha estado corriendo la app.

Imagina que puedes pagar solo por el tiempo que se ejecuta tu código. Eso es Lambda. ¡No más pagos por tiempo que no has estado usando la app!



PABLO DEL ÁLAMO



Casos de uso comunes

Desde procesamiento de datos en streaming
hasta backend para aplicaciones web o APIs.

¡Lambda lo hace todo!



PABLO DEL ÁLAMO



Lenguajes soportados

Name	Identifier	Operating system	Deprecation date	Block function create	Block function update
Node.js 20	nodejs20.x	Amazon Linux 2023	Not scheduled	Not scheduled	Not scheduled
Node.js 18	nodejs18.x	Amazon Linux 2	Jul 31, 2025	Sep 1, 2025	Oct 1, 2025
Python 3.12	python3.12	Amazon Linux 2023	Not scheduled	Not scheduled	Not scheduled
Python 3.11	python3.11	Amazon Linux 2	Not scheduled	Not scheduled	Not scheduled
Python 3.10	python3.10	Amazon Linux 2	Not scheduled	Not scheduled	Not scheduled
Python 3.9	python3.9	Amazon Linux 2	Not scheduled	Not scheduled	Not scheduled
Java 21	java21	Amazon Linux 2023	Not scheduled	Not scheduled	Not scheduled
Java 17	java17	Amazon Linux 2	Not scheduled	Not scheduled	Not scheduled
Java 11	java11	Amazon Linux 2	Not scheduled	Not scheduled	Not scheduled
Java 8	java8.al2	Amazon Linux 2	Not scheduled	Not scheduled	Not scheduled
.NET 8	dotnet8	Amazon Linux 2023	Not scheduled	Not scheduled	Not scheduled
.NET 6	dotnet6	Amazon Linux 2	Dec 20, 2024	Feb 28, 2025	Mar 31, 2025
Ruby 3.3	ruby3.3	Amazon Linux 2023	Not scheduled	Not scheduled	Not scheduled
Ruby 3.2	ruby3.2	Amazon Linux 2	Not scheduled	Not scheduled	Not scheduled
OS-only Runtime	provided.al2023	Amazon Linux 2023	Not scheduled	Not scheduled	Not scheduled
OS-only Runtime	provided.al2	Amazon Linux 2	Not scheduled	Not scheduled	Not scheduled



PABLO DEL ÁLAMO



5 **Cómo funciona**

Es sencillo: subes tu código, defines el evento (trigger) que lo ejecutará y ¡listo!




PABLO DEL ÁLAMO



Ejemplos Comunes de Triggers para AWS Lambda:

1. Amazon S3

- Evento de Trigger: Cuando se sube, elimina o modifica un objeto en un bucket de S3.
- Uso Común: Procesamiento de imágenes, generar miniaturas, análisis de datos en archivos de texto, etc.
-  Ejemplo: Cada vez que un archivo de imagen se sube a un bucket S3, Lambda lo procesa para generar una versión reducida que se guarda en otro bucket.





2. Amazon DynamoDB Streams

- Evento de Trigger: Cambios en tablas de DynamoDB (modificaciones, inserciones, eliminaciones).
- Uso Común: Sincronización de datos, auditoría de transacciones, creación de notificaciones basadas en cambios en data.
- 📌 Ejemplo: Registrar todos los cambios de una tabla de usuarios en DynamoDB para tener una auditoría detallada de todas las operaciones CRUD.





3. Amazon Kinesis 🤖


- Evento de Trigger: Datos ingresando a un stream de Kinesis.
- Uso Común: Análisis en tiempo real, procesamiento de datos en streaming, manejar grandes volúmenes de logs.
- 📌 Ejemplo: Procesar logs en tiempo real que están siendo enviados a un Kinesis Data Stream desde diferentes fuentes.



PABLO DEL ÁLAMO




4. AWS API Gateway

- Evento de Trigger: Request HTTP a un endpoint API.
- Uso Común: Creación de API RESTful o HTTP para backend sin servidor, validación y procesamiento de peticiones.
-  Ejemplo: Lambda funciona como el backend de una API REST que permite a los usuarios interactuar con una base de datos de productos.





5. Amazon SNS (Simple Notification Service)

- Evento de Trigger: Mensajes publicados en un tema SNS.
- Uso Común: Procesamiento de mensajes, envío de notificaciones.
-  Ejemplo: Al recibir una alerta importante en SNS, Lambda procesa el mensaje para enviar emails personalizados a la lista de usuarios suscritos.






6. AWS CloudWatch Events / EventBridge 🚨

- Evento de Trigger: Eventos basados en cron (schedule) o patrones de evento de sistemas.
- Uso Común: Automatización de tareas de gestión de infraestructura y recursos.
- 👉 Ejemplo: Ejecutar una función cada día a medianoche para realizar respaldo de datos o limpiar logs antiguos.





7. AWS SQS (Simple Queue Service)

- Evento de Trigger: Mensajes llegando a una cola SQS.
- Uso Común: Desacoplar aplicaciones, procesamiento asíncrono de tareas.
-  Ejemplo: Procesar pedidos de una tienda en línea donde cada pedido genera un mensaje en una cola SQS y Lambda se encarga de procesarlo.





8. AWS CloudFormation 🛠️

- Evento de Trigger: Cambios en el ciclo de vida de los stacks de CloudFormation.
- Uso Común: Ejecutar tareas automatizadas como notificaciones o ajustes en configuraciones al cambiar recursos en la infraestructura.
- 👉 Ejemplo: Automatizar la notificación de cambios significativos o rollback en la infraestructura a través de un mensaje a un grupo de DevOps usando Lambda.





6

Creación. Método 1: Crear una Función desde Cero (Archivo ZIP o jar)

- Ve a la consola de AWS y abre el servicio Lambda.



PABLO DEL ÁLAMO

Last fetched 5 minutes ago

↺

Actions ▼

Create function

< 1 >

⚙

Runtime ▼	Last modified ▼
Java 17	<u>1 week ago</u>
Java 11	<u>8 months ago</u>
Python 3.12	<u>5 months ago</u>
Java 17	<u>1 day ago</u>
Python 3.12	<u>5 months ago</u>
Python 3.12	<u>7 months ago</u>
Java 17	<u>3 weeks ago</u>
Python 3.12	<u>8 months ago</u>

© 2024, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences



PABLO DEL ÁLAMO



Crea una nueva función:

- Haz clic en "Create function".
- Elige "Author from scratch".
- Asigna un nombre a tu función
- Elige el lenguaje en el que quieres desarrollar la lambda
- Elige el rol de ejecución (IAM) que tenga los permisos necesarios, o deja el que se crea por defecto. Si necesitas que tu lambda acceda a otros servicios de AWS (S3, Dynamo, etc), siempre puedes añadir los permisos más tarde.



PABLO DEL ÁLAMO

Create function [Info](#)

Choose one of the following options to create your function.

☒ **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.

myFunctionName

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime [Info](#)

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 20.x



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

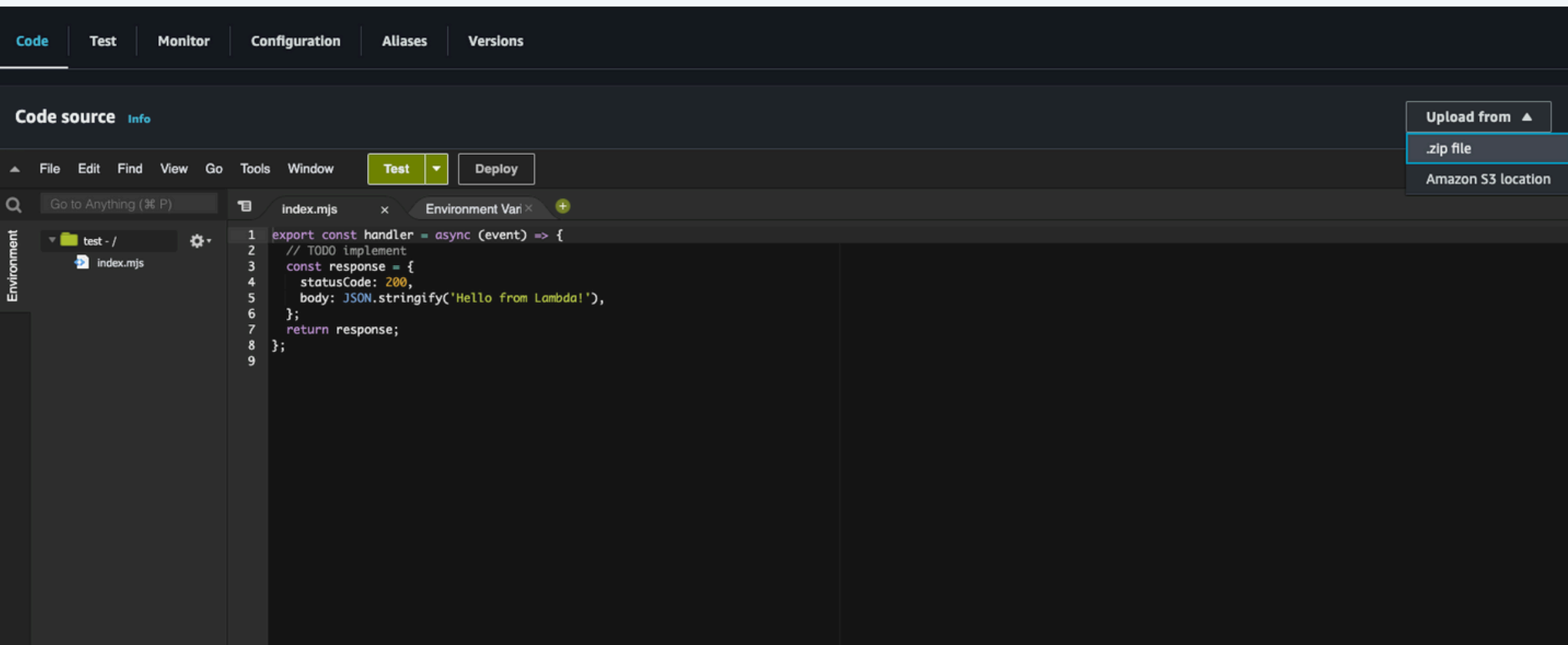
☐ Use an existing role



PABLO DEL ÁLAMO

Sube tu código:

- En "Code source", selecciona "Upload from" y elige "ZIP file" (permite también jar si vas a usar Java) .
- Sube el archivo ZIP de tu código (directamente, o previamente almacenado en un contenedor S3).



PABLO DEL ÁLAMO



Configura ajustes:

- Establece el tiempo de ejecución (Node.js, Python, etc.).
- Ajusta la memoria y el timeout según tus necesidades.

[Configuration](#) [Aliases](#) [Versions](#)

General configuration [Info](#)

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



PABLO DEL ÁLAMO



Establece Triggers: Añade los triggers necesarios (como todos los ya mencionados más arriba en las primeras diapositivas), que serán los que hagan que tu lambda se ejecute, ¡y listo!




PABLO DEL ÁLAMO


Add trigger

Trigger configuration [Info](#)


Select a source ▲

- APIs/Interactive/web
- 


Alexa

alexaiotvoice
- 


API Gateway

awsapisapplication-servicesbackendHTTPRESTserverless
- 


Application Load Balancer

awsHTTPload-balancingserverweb
- 


CodeCommit

awsasynchronousdeveloper-toolsgit
- 


VPC Lattice

awsnetworkingprivateprivatelinkvpc
- Batch/bulk data processing
- 


AWS IoT

awsasynchronousdevicesiot
- 


CloudWatch Logs

awsasynchronouscwloggingmanagement-tools
- 


EventBridge (CloudWatch Events)

awsasynchronousschedulemanagement-tools
- 

S3

awsasynchronousstorage
- 

SNS

awsasynchronousmessagingnotificationspub-subpush
- 

SQS

awsevent-source-mappingpollingqueue

Cancel

Add





7 Creación.

Método 2: Usar una Imagen Docker

- Configura Docker Localmente:
 - Asegúrate de tener Docker instalado en tu máquina.
 - Crea tu Dockerfile con el código que deseas ejecutar.



PABLO DEL ÁLAMO



- Construye tu Imagen:
 - `docker build -t my-lambda-function .`
- Sube la Imagen a ECR (Elastic Container Registry):
 - Crea un repositorio en ECR a través de la consola de AWS.
 - Etiqueta y sube la imagen Docker al repositorio ECR:
 - `docker tag my-lambda-function:latest <account-id>.dkr.ecr.<region>.amazonaws.com/my-lambda-function:latest`
 - `docker push <account-id>.dkr.ecr.<region>.amazonaws.com/my-lambda-function:latest`



PABLO DEL ÁLAMO



8

Ejemplo Práctico

Vamos a crear un ejemplo práctico de una función AWS Lambda escrita en Java.

Supongamos que queremos una función Lambda que se dispare cada vez que se suba un archivo a un bucket de S3.

Esta función simplemente leerá la metadata del archivo y la imprimirá en los logs.



PABLO DEL ÁLAMO

Creamos la lambda como ya hemos explicado antes. Configuramos como trigger, la subida o actualización de archivos a un contenedor de S3 (creado previamente)

Trigger configuration

Info

S3

aws

asynchronous

storage

Bucket

Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

Bucket must be in region eu-west-3

Event types

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events

PUT

POST

Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.

e.g. images/

Suffix - optional

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.

e.g. .jpg



PABLO DEL ÁLAMO

Aquí va una clase de ejemplo para manejar el evento de subida de S3

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3Object;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class S3EventHandler implements RequestHandler<S3Event, String> {
    private static final Logger LOGGER = LoggerFactory.getLogger(S3EventHandler.class);
    private AmazonS3 s3Client = AmazonS3ClientBuilder.standard().build();

    @Override
    public String handleRequest(S3Event s3Event, Context context) {
        s3Event.getRecords().forEach(record -> {
            String bucketName = record.getS3().getBucket().getName();
            String objectKey = record.getS3().getObject().getKey();
            LOGGER.info("New file uploaded: Bucket = {}, Key = {}", bucketName, objectKey);

            try {
                S3Object s3Object = s3Client.getObject(bucketName, objectKey);
                LOGGER.info("Object metadata: ContentType = {}, Size = {}",
                    s3Object.getObjectMetadata().getContentType(),
                    s3Object.getObjectMetadata().getContentLength());
            } catch (Exception e) {
                LOGGER.error("Error fetching object metadata", e);
            }
        });
        return "Success";
    }
}
```



PABLO DEL ÁLAMO



Explicación del Código

- **Logger:** `private static final Logger LOGGER` se utiliza para registrar información de la ejecución y facilitar el debugging.
- **AmazonS3 Client:** `private AmazonS3 s3Client = AmazonS3ClientBuilder.standard().build();` inicializa un cliente de S3 para interactuar con el servicio.



PABLO DEL ÁLAMO



- Método `handleRequest`: Implementa la interfaz `RequestHandler<S3Event, String>`, lo cual hace que esta clase interprete los eventos del bucket de S3 y procese cada archivo subido.
 - `s3Event.getRecords().forEach(record -> {...})`: Itera sobre los registros de eventos en el evento de S3.
 - `String bucketName = record.getS3().getBucket().getName();`: Obtiene el nombre del bucket donde se subió el archivo.



PABLO DEL ÁLAMO

- `String objectKey = record.getS3().getObject().getKey();`: Obtiene la clave del objeto en el bucket (básicamente, la ruta del archivo en el bucket).
- `LOGGER.info(...)`: Imprime la información del archivo nuevo en los logs.
- `s3Client.getObject(bucketName, objectKey)`: Recupera el objeto desde S3 para poder acceder a sus metadatos.
- `try-catch`: Maneja excepciones para que si hay algún error al obtener la metadata, simplemente se loguea como error.



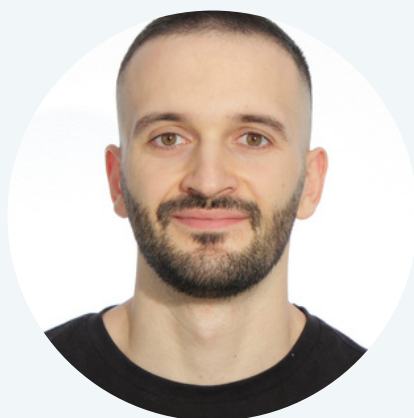
Despliegue

- **Compila y empaqueta el proyecto:** Usando Maven, empaqueta la función en un archivo JAR.
- **Sube el JAR a AWS Lambda:**
 - Ve a la consola de AWS Lambda.
 - Crea una nueva función desde cero.
 - Sube el archivo JAR.
 - Configura el trigger de S3.
 - Prueba la Función: Sube un nuevo archivo a tu bucket S3 y observa cómo se ejecuta la función y registra la metadata del archivo en CloudWatch Logs.





¿Te ha resultado útil?



- Comparte esta guía con tu equipo o amigos desarrolladores.
- Guárdala para tenerla siempre a mano.
- ¡Dale un like o comenta si tienes preguntas!

