

## INTEGRANTES:

- William Andres Yaruro Cuan
- Jose Camilo Ricardo Vilorio
- Brian Andres Acuña Donado

## PRIMERO CREAR UN USUARIO Y GRUPO PARA ESTE TALLER.

Añadir usuario(s)

1 2 3 4 5



### Correcto

Ha creado correctamente los usuarios que se muestran a continuación. Puede ver y descargar las credenciales de seguridad de los usuarios. También puede enviar a los usuarios un correo electrónico con instrucciones para iniciar sesión en la consola de administración de AWS. Esta es la última vez que las credenciales estarán disponibles para descargarlas. Sin embargo, puede crear otras en cualquier momento.

Los usuarios con acceso a la consola de administración de AWS pueden iniciar sesión en:  
<https://175452962618.signin.aws.amazon.com/console>

Descargar .csv

	Usuario	ID de clave de acceso	Clave de acceso secreta
▶	✓ serverless-william	AKIASRWOPX45PIPH4XH7	m6oV1anODR3lIXLBywDI0z DwlJm+DcYpFpU+j2oZ <a href="#">Ocultar</a>

## DESCARGAR E INSTALAR AWS CLI

RECURSOS

[Interfaz de línea de comandos de AWS](#)

VÍNCULOS RELACIONADOS

[Documentación](#)

[Herramientas](#)

[Notas de la versión](#)

Comience con AWS de forma gratuita

[Cree una cuenta gratuita](#)

### Interfaz de línea de comandos de AWS

La interfaz de línea de comandos (CLI) es una herramienta unificada para administrar los productos de AWS. Solo tendrá que descargar y configurar una única herramienta para poder controlar varios servicios de AWS desde la línea de comando y automatizarlos mediante secuencias de comandos.

La interfaz de línea de comandos (CLI) de AWS presenta un nuevo conjunto de [comandos de archivo](#) simples para que las transferencias de archivos entrantes y salientes de Amazon S3 sean eficientes.

1

Introducción »

Referencia de la CLI »

Proyecto GitHub »

Comunidad de la comunidad »

**Windows**

Descargue y ejecute el instalador de Windows de [64](#) o [32](#) bits.

**Mac y Linux**

Se requiere [Python](#) 2.6.5 o superior. Instalación con [pip](#).

```
pip install awscli
```

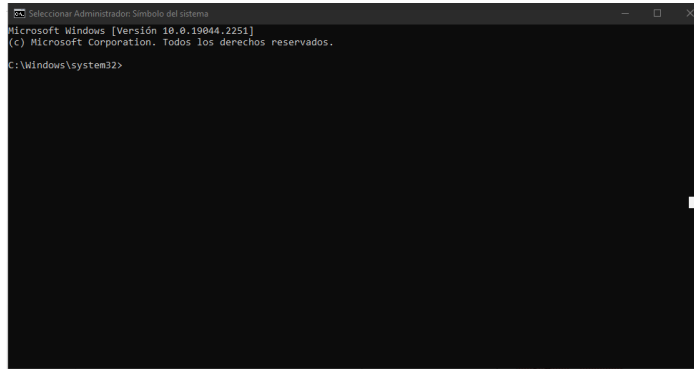
**Amazon Linux**

AWS CLI viene preinstalado en [Amazon Linux AMI](#).

**Notas de la versión**

Consulte las [notas de la versión](#) para obtener más información sobre la versión más reciente.

Una vez finalizada la instalación vamos a proceder a abrir la consola de comandos (Windows + R) (Type: cmd)



Verificamos que esté instalado correctamente aws CLI

```
C:\Windows\system32>aws --version
aws-cli/1.27.21 Python/3.8.10 Windows/10 botocore/1.29.21
```

Utilizamos el cmd “aws configure” para configurar aws....

Configuramos nuestras credenciales de nuestro usuario...

```
C:\>aws configure
AWS Access Key ID [None]: AKIASRWOPX45PIPH4XH7
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
```

Ahora vamos a configurar SERVERLESS

## Installation

Install the `serverless` CLI via NPM:

```
npm install -g serverless
```

Para instalar SERVERLESS lo haremos con ayuda de NODE.JS

```
C:\>npm install -g serverless
npm WARN deprecated querystring@0.2.1: The querystring API is considered Legacy. new code should use the URLSearchParams
API instead.
[.....] - fetchMetadata: sill resolveWithNewModule sprintf-js@1.0.3 checking installable status
```

```
Serverless Framework successfully installed!
```

```
To start your first project run "serverless".
```

```
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.2 (node_modules\serverless\node_modules\chokidar\node_modules\fsevents):
```

```
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
```

```
+ serverless@3.25.1
```

```
added 394 packages from 324 contributors in 48.585s
```

Seleccionamos una ruta para crear SERVERLESS..

```
C:\Users\WillYer\Desktop\AWS>serverless

Creating a new serverless project

? What do you want to make? (Use arrow keys)
> AWS - Node.js - Starter
  AWS - Node.js - HTTP API
  AWS - Node.js - Scheduled Task
  AWS - Node.js - SQS Worker
  AWS - Node.js - Express API
  AWS - Node.js - Express API with DynamoDB
  AWS - Python - Starter
  AWS - Python - HTTP API
  AWS - Python - Scheduled Task
  AWS - Python - SQS Worker
  AWS - Python - Flask API
  AWS - Python - Flask API with DynamoDB
  Other
```

```
C:\Users\WillYer\Desktop\AWS>serverless

Creating a new serverless project

? What do you want to make? AWS - Node.js - HTTP API
? What do you want to call this project? aws-lambda-crud
[+] Project successfully created in aws-lambda-crud folder

? Do you want to login/register to Serverless Dashboard? No

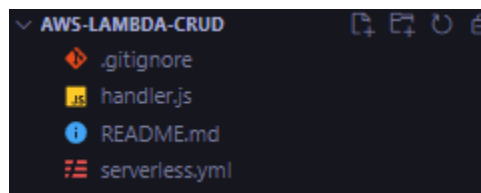
? Do you want to deploy now? (Y/n) n
? Do you want to deploy now? No

What next?
Run these commands in the project directory:

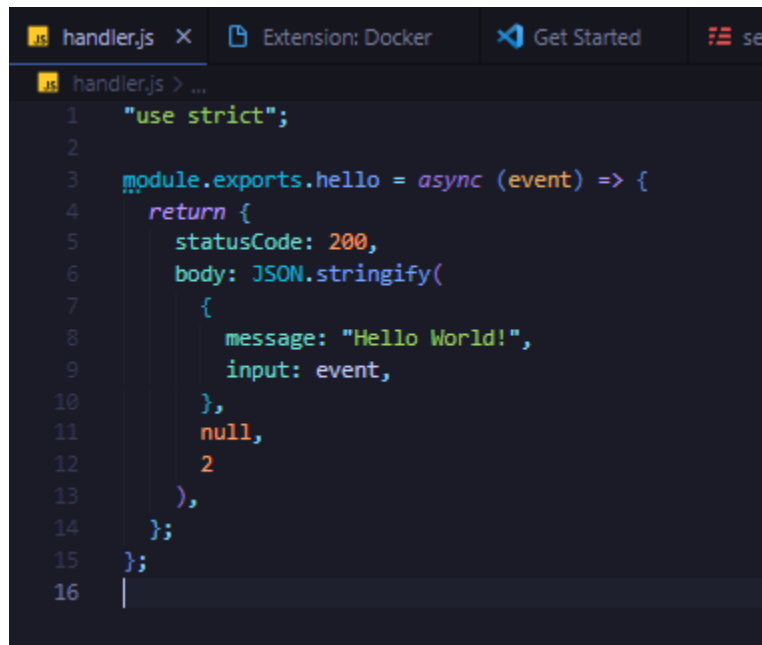
serverless deploy    Deploy changes
serverless info      View deployed endpoints and resources
serverless invoke    Invoke deployed functions
serverless --help    Discover more commands

C:\Users\WillYer\Desktop\AWS>
```

Y ya con esto quedaría creado el proyecto básico creado..



Ahora vamos a abrir nuestro editor de código, en este caso estaremos utilizando VISUAL STUDIO CODE.



```
handler.js x  Extension: Docker  Get Started  se
handler.js > ...
1  "use strict";
2
3  module.exports.hello = async (event) => {
4    return {
5      statusCode: 200,
6      body: JSON.stringify(
7        {
8          message: "Hello World!",
9          input: event,
10         },
11         null,
12         2
13       ),
14     };
15   };
16  |
```

Configuramos la región en la que vamos a estar utilizando los servicios de aws.



```
service: aws-lambda-crud
frameworkVersion: '3'

provider:
  name: aws
  runtime: nodejs14.x
  region: us-east-1

functions:
  hello:
    handler: handler.hello
    events:
      - httpApi:
          path: /
          method: get
```

Vamos a desplegar la aplicación con una función de serverless.

```
PS C:\Users\WillYer\Desktop\AWS\aws-lambda-crud> serverless deploy --verbose

Deploying aws-lambda-crud to stage dev (us-east-1)

Packaging
Excluding development dependencies for service package
Retrieving CloudFormation stack
Creating CloudFormation stack
Creating new change set
Waiting for new change set to be created
Change Set did not reach desired state, retrying
Executing created change set
  CREATE_IN_PROGRESS - AWS::CloudFormation::Stack - aws-lambda-crud-dev
  CREATE_IN_PROGRESS - AWS::S3::Bucket - ServerlessDeploymentBucket
  CREATE_IN_PROGRESS - AWS::S3::Bucket - ServerlessDeploymentBucket
  CREATE_COMPLETE - AWS::S3::Bucket - ServerlessDeploymentBucket
  CREATE_IN_PROGRESS - AWS::S3::BucketPolicy - ServerlessDeploymentBucketPolicy
  CREATE_IN_PROGRESS - AWS::S3::BucketPolicy - ServerlessDeploymentBucketPolicy
  CREATE_COMPLETE - AWS::S3::BucketPolicy - ServerlessDeploymentBucketPolicy
  CREATE_COMPLETE - AWS::CloudFormation::Stack - aws-lambda-crud-dev
Uploading
Uploading CloudFormation file to S3
Uploading State file to S3
Uploading service aws-lambda-crud.zip file to S3 (1.69 kB)
Updating CloudFormation stack
Creating new change set
Waiting for new change set to be created
Change Set did not reach desired state, retrying
Change Set did not reach desired state, retrying
Executing created change set
  UPDATE_IN_PROGRESS - AWS::CloudFormation::Stack - aws-lambda-crud-dev
  CREATE_IN_PROGRESS - AWS::IAM::Role - IamRoleLambdaExecution
  CREATE_IN_PROGRESS - AWS::Logs::LogGroup - HelloLogGroup
  CREATE_IN_PROGRESS - AWS::ApiGatewayV2::Api - HttpApi
  CREATE_IN_PROGRESS - AWS::IAM::Role - IamRoleLambdaExecution
  CREATE_IN_PROGRESS - AWS::Logs::LogGroup - HelloLogGroup
  CREATE_COMPLETE - AWS::Logs::LogGroup - HelloLogGroup
  CREATE_IN_PROGRESS - AWS::ApiGatewayV2::Api - HttpApi
  CREATE_COMPLETE - AWS::ApiGatewayV2::Api - HttpApi
  CREATE_IN_PROGRESS - AWS::ApiGatewayV2::Stage - HttpApiStage
  CREATE_IN_PROGRESS - AWS::ApiGatewayV2::Stage - HttpApiStage
  CREATE_COMPLETE - AWS::ApiGatewayV2::Stage - HttpApiStage
  CREATE_COMPLETE - AWS::IAM::Role - IamRoleLambdaExecution
  CREATE_IN_PROGRESS - AWS::Lambda::Function - HelloLambdaFunction
  CREATE_IN_PROGRESS - AWS::Lambda::Function - HelloLambdaFunction
  CREATE_COMPLETE - AWS::Lambda::Function - HelloLambdaFunction
  CREATE_IN_PROGRESS - AWS::Lambda::Version - HelloLambdaVersion3PSRgS7v61Ab5sxxwU4j0R5FQ0Un7iza3cf6w1Ghx0
  CREATE_IN_PROGRESS - AWS::ApiGatewayV2::Integration - HttpApiIntegrationHello
  CREATE_IN_PROGRESS - AWS::Lambda::Permission - HelloLambdaPermissionHttpApi
  CREATE_IN_PROGRESS - AWS::Lambda::Version - HelloLambdaVersion3PSRgS7v61Ab5sxxwU4j0R5FQ0Un7iza3cf6w1Ghx0
  CREATE_IN_PROGRESS - AWS::Lambda::Permission - HelloLambdaPermissionHttpApi
  CREATE_COMPLETE - AWS::Lambda::Version - HelloLambdaVersion3PSRgS7v61Ab5sxxwU4j0R5FQ0Un7iza3cf6w1Ghx0
  CREATE_IN_PROGRESS - AWS::ApiGatewayV2::Integration - HttpApiIntegrationHello
  CREATE_COMPLETE - AWS::ApiGatewayV2::Integration - HttpApiIntegrationHello
  CREATE_IN_PROGRESS - AWS::ApiGatewayV2::Route - HttpApiRouteGet
  CREATE_IN_PROGRESS - AWS::ApiGatewayV2::Route - HttpApiRouteGet
  CREATE_COMPLETE - AWS::ApiGatewayV2::Route - HttpApiRouteGet
  CREATE_COMPLETE - AWS::Lambda::Permission - HelloLambdaPermissionHttpApi
  UPDATE_COMPLETE_CLEANUP_IN_PROGRESS - AWS::CloudFormation::Stack - aws-lambda-crud-dev
```

```

UPDATE_COMPLETE - AWS::CloudFormation::Stack - aws-lambda-crud-dev
Retrieving CloudFormation stack
Removing old service artifacts from S3

✓ Service deployed to stack aws-lambda-crud-dev (114s)

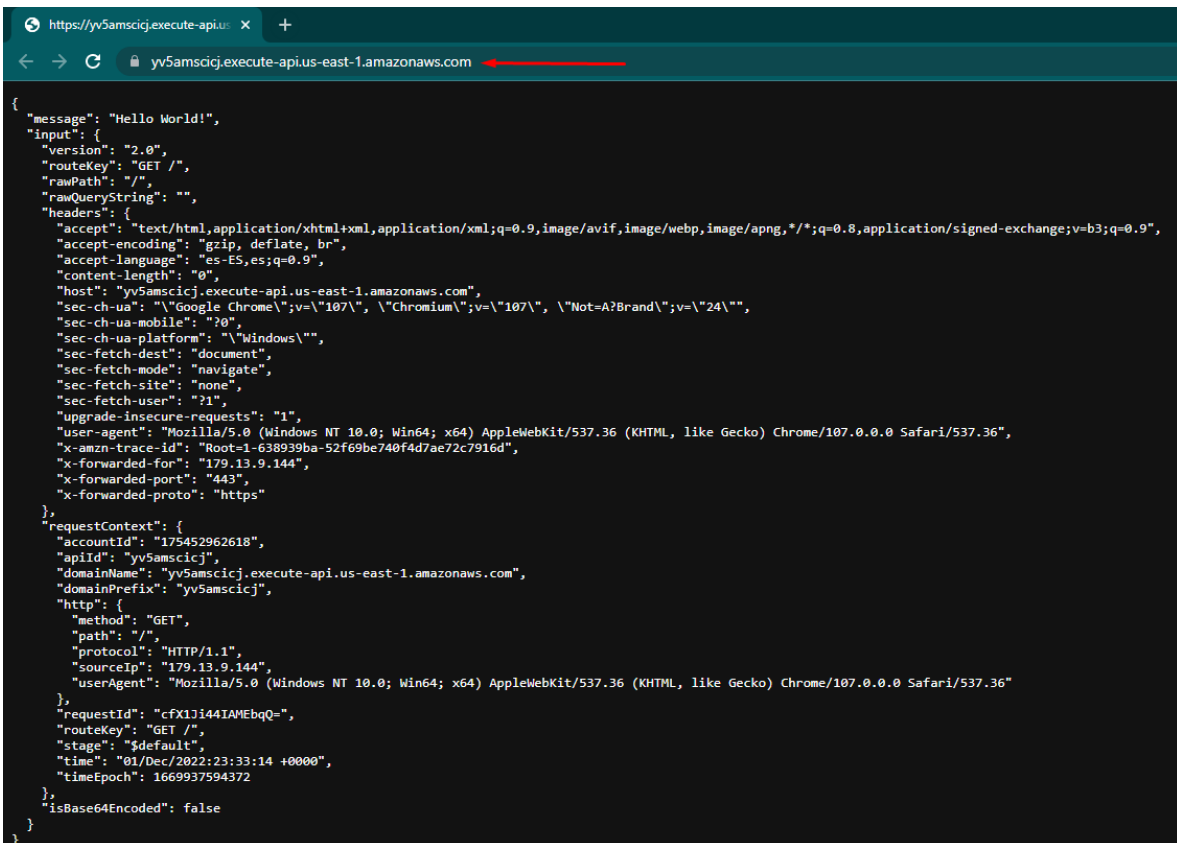
endpoint: GET - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/
functions:
  hello: aws-lambda-crud-dev-hello (1.7 kB)

Stack Outputs:
HelloLambdaFunctionQualifiedArn: arn:aws:lambda:us-east-1:175452962618:function:aws-lambda-crud-dev-hello:1
HttpApiId: yv5amscicj
ServerlessDeploymentBucketName: aws-lambda-crud-dev-serverlessdeploymentbucket-tqjt8tai7yx0
HttpApiUrl: https://yv5amscicj.execute-api.us-east-1.amazonaws.com

Need a better logging experience than CloudWatch? Try our Dev Mode in console: run "serverless --console"
PS C:\Users\WillYer\Desktop\AWS\aws-lambda-crud>

```

Al finalizar nos va a retornar el endpoint que acabamos de crear.



```

{
  "message": "Hello World!",
  "input": {
    "version": "2.0",
    "routeKey": "GET /",
    "rawPath": "/",
    "rawQueryString": "",
    "headers": {
      "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
      "accept-encoding": "gzip, deflate, br",
      "accept-language": "es-ES,es;q=0.9",
      "content-length": "0",
      "host": "yv5amscicj.execute-api.us-east-1.amazonaws.com",
      "sec-ch-ua": "\"Google Chrome\";v=\"107\"\", \"Chromium\";v=\"107\"\", \"Not-A?Brand\";v=\"24\"",
      "sec-ch-ua-mobile": "?0",
      "sec-ch-ua-platform": "\"Windows\"",
      "sec-fetch-dest": "document",
      "sec-fetch-mode": "navigate",
      "sec-fetch-site": "none",
      "sec-fetch-user": "?1",
      "upgrade-insecure-requests": "1",
      "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36",
      "x-amzn-trace-id": "Root=1-638939ba-52f69be740f4d7ae72c7916d",
      "x-forwarded-for": "179.13.9.144",
      "x-forwarded-port": "443",
      "x-forwarded-proto": "https"
    },
    "requestContext": {
      "accountId": "175452962618",
      "apiId": "yv5amscicj",
      "domainName": "yv5amscicj.execute-api.us-east-1.amazonaws.com",
      "domainPrefix": "yv5amscicj",
      "http": {
        "method": "GET",
        "path": "/",
        "protocol": "HTTP/1.1",
        "sourceIp": "179.13.9.144",
        "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36"
      },
      "requestId": "cfX1Ji44IAMEbqQ=",
      "routeKey": "GET /",
      "stage": "$default",
      "time": "01/Dec/2022:23:33:14 +0000",
      "timeEpoch": 1669937594372
    },
    "isBase64Encoded": false
  }
}

```

## Configuración DynamoDB:

Ahora vamos a comenzar la creación del CRUD, conectándonos a la base de datos en este caso utilizando DYNAMO DB..

```
resources:
  Resources:
    TaskTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: TaskTable
        BillingMode: PAY_PER_REQUEST
        AttributeDefinitions:
          - AttributeName: id
            AttributeType: S
        KeySchema:
          - AttributeName: id
            KeyType: HASH
```

Para esto configuramos un apartado de recursos en nuestro serverless.yml

Después vamos a volver a ejecutar el comando de serverless para subir los cambios...

```
PS C:\Users\WillYer\Desktop\AWS\aws-lambda-crud> serverless deploy --verbose

Deploying aws-lambda-crud to stage dev (us-east-1)

Packaging
Excluding development dependencies for service package
Retrieving CloudFormation stack
Uploading
Uploading CloudFormation file to S3
Uploading State file to S3
Uploading service aws-lambda-crud.zip file to S3 (1.69 kB)
Updating CloudFormation stack
Creating new change set
Waiting for new change set to be created
Change Set did not reach desired state, retrying
Change Set did not reach desired state, retrying
Executing created change set
  UPDATE_IN_PROGRESS - AWS::CloudFormation::Stack - aws-lambda-crud-dev
  CREATE_IN_PROGRESS - AWS::DynamoDB::Table - TaskTable
  UPDATE_IN_PROGRESS - AWS::Lambda::Function - HelloLambdaFunction
  CREATE_IN_PROGRESS - AWS::DynamoDB::Table - TaskTable
  UPDATE_COMPLETE - AWS::Lambda::Function - HelloLambdaFunction
  CREATE_COMPLETE - AWS::DynamoDB::Table - TaskTable
  UPDATE_COMPLETE_CLEANUP_IN_PROGRESS - AWS::CloudFormation::Stack - aws-lambda-crud-dev
  UPDATE_COMPLETE - AWS::CloudFormation::Stack - aws-lambda-crud-dev
Retrieving CloudFormation stack
Removing old service artifacts from S3

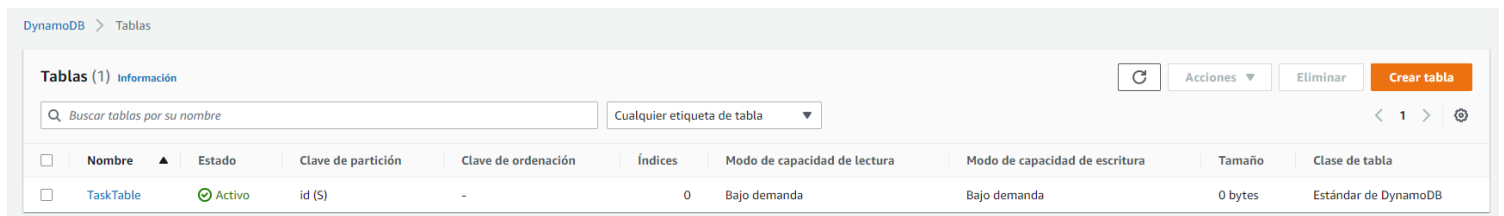
✓Service deployed to stack aws-lambda-crud-dev (48s)

endpoint: GET - https://yv5amsicj.execute-api.us-east-1.amazonaws.com/
functions:
  hello: aws-lambda-crud-dev-hello (1.7 kB)

Stack Outputs:
HelloLambdaFunctionQualifiedArn: arn:aws:lambda:us-east-1:175452962618:function:aws-lambda-crud-dev-hello:1
HttpApiId: yv5amsicj
ServerlessDeploymentBucketName: aws-lambda-crud-dev-serverlessdeploymentbucket-tqjt8tai7yx0
HttpApiUrl: https://yv5amsicj.execute-api.us-east-1.amazonaws.com

Need a better logging experience than CloudWatch? Try our Dev Mode in console: run "serverless --console"
PS C:\Users\WillYer\Desktop\AWS\aws-lambda-crud>
```

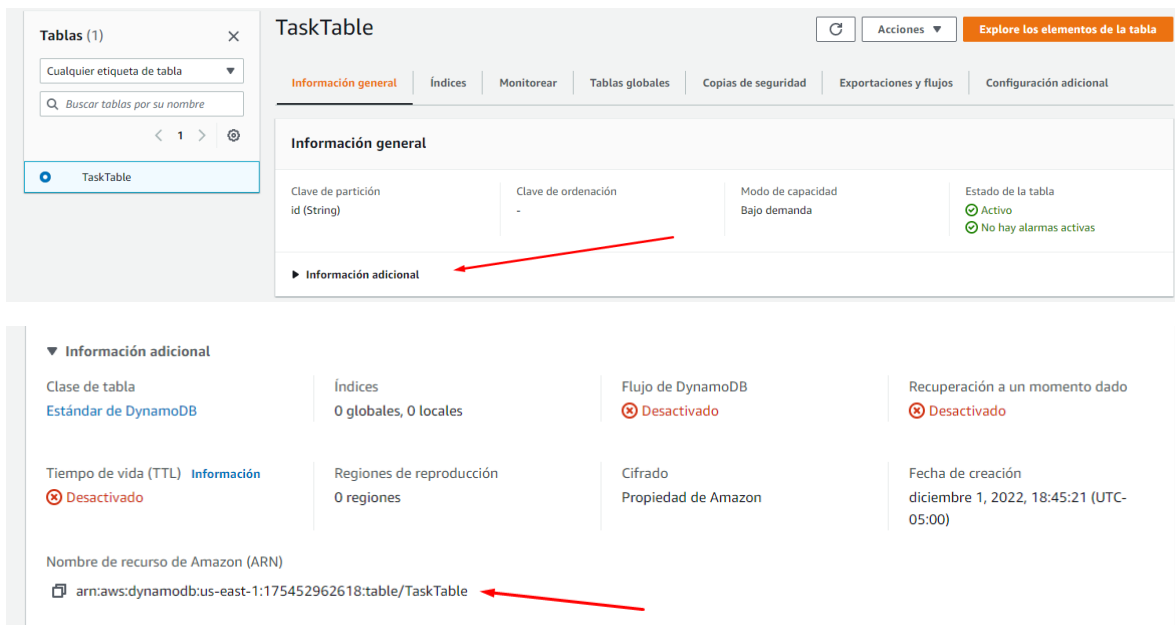
Con estos pasos quedaría creada nuestra tabla dentro de DynamoDB...



The screenshot shows the AWS DynamoDB console interface. At the top, there's a header with 'DynamoDB' and 'Tablas'. Below it, a search bar and a dropdown menu are visible. A table lists the existing tables, with 'TaskTable' being the only one. The table has columns for Nombre, Estado, Clave de partición, Clave de ordenación, Índices, Modo de capacidad de lectura, Modo de capacidad de escritura, Tamaño, and Clase de tabla. The 'TaskTable' row shows it is 'Activo' (Active) with a partition key 'id (S)' and no sort key.

	Nombre	Estado	Clave de partición	Clave de ordenación	Índices	Modo de capacidad de lectura	Modo de capacidad de escritura	Tamaño	Clase de tabla
<input type="checkbox"/>	TaskTable	Activo	id (S)	-	0	Bajo demanda	Bajo demanda	0 bytes	Estándar de DynamoDB

El siguiente paso es configurar TaskTable en nuestro proyecto...



The screenshot shows the configuration details for the 'TaskTable' in the AWS DynamoDB console. The 'Información general' tab is selected, showing the partition key 'id (String)', sort key '-', and capacity mode 'Bajo demanda'. The table is 'Activo' (Active) with no active alarms. The 'Información adicional' section is expanded, showing the table class 'Estándar de DynamoDB', 0 global and 0 local indexes, disabled DynamoDB stream, disabled point-in-time recovery, disabled TTL, 0 replication regions, Amazon encryption, and the creation date 'diciembre 1, 2022, 18:45:21 (UTC-05:00)'. The Amazon resource name (ARN) is 'arn:aws:dynamodb:us-east-1:175452962618:table/TaskTable'.

**Información general**

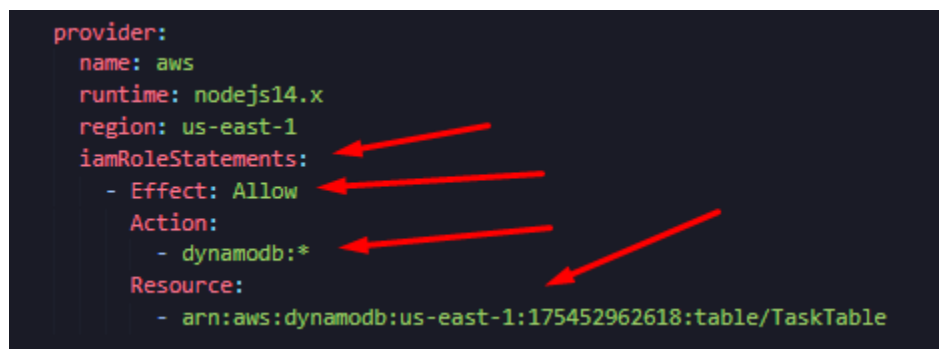
Clave de partición id (String)	Clave de ordenación -	Modo de capacidad Bajo demanda	Estado de la tabla Activo No hay alarmas activas
-----------------------------------	--------------------------	-----------------------------------	--

**Información adicional**

Clase de tabla Estándar de DynamoDB	Índices 0 globales, 0 locales	Flujo de DynamoDB Desactivado	Recuperación a un momento dado Desactivado
Tiempo de vida (TTL) Desactivado	Regiones de reproducción 0 regiones	Cifrado Propiedad de Amazon	Fecha de creación diciembre 1, 2022, 18:45:21 (UTC-05:00)

Nombre de recurso de Amazon (ARN)  
arn:aws:dynamodb:us-east-1:175452962618:table/TaskTable

Copiamos esa información de ARN... y agregamos a nuestro serverless.yml...



```
provider:
  name: aws
  runtime: nodejs14.x
  region: us-east-1
  iamRoleStatements:
    - Effect: Allow
      Action:
        - dynamodb:*
      Resource:
        - arn:aws:dynamodb:us-east-1:175452962618:table/TaskTable
```

The screenshot shows a snippet of a serverless.yml file. It defines the provider as 'aws' with runtime 'nodejs14.x' and region 'us-east-1'. Under 'iamRoleStatements', it specifies an 'Effect' of 'Allow' for the 'Action' 'dynamodb:\*' on the 'Resource' 'arn:aws:dynamodb:us-east-1:175452962618:table/TaskTable'.

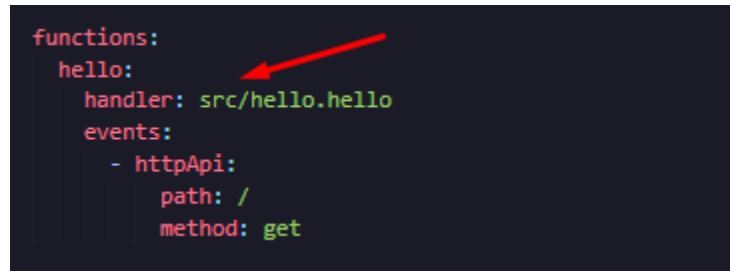
Con esto quedaría configurado el apartado de DynamoDB....



## Configuración de LAMBDA:

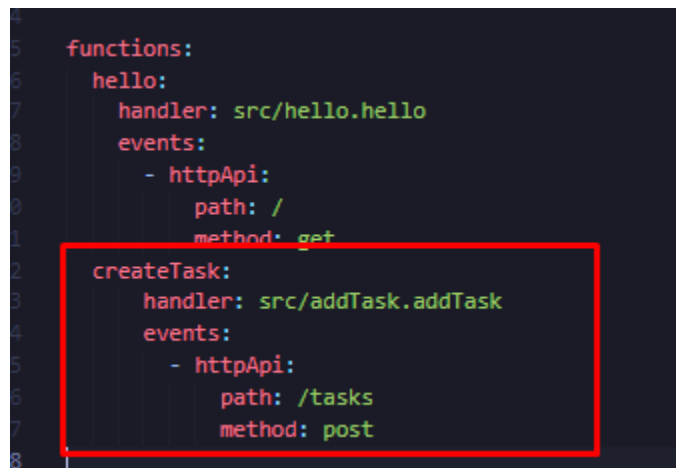
Ahora vamos a reordenar nuestro código, en este caso crearemos una carpeta SRC, pasaremos el archivo handler.js a esta carpeta, a este archivo le actualizaremos el nombre para identificarlo de forma más fácil.

Al momento de realizar este cambio es necesario que también se actualice la ruta dentro del serverless.yml, agregando la carpeta que fue creada “src”



```
functions:
  hello:
    handler: src/hello.hello
    events:
      - httpApi:
          path: /
          method: get
```

Ahora vamos a crear una función nueva llamada addTask.js dentro de la carpeta que acabamos de crear “src”, después de crear la carpeta es necesario configurar esa información en el serverless.yml.



```
functions:
  hello:
    handler: src/hello.hello
    events:
      - httpApi:
          path: /
          method: get
  createTask:
    handler: src/addTask.addTask
    events:
      - httpApi:
          path: /tasks
          method: post
```

En la función vamos a escribir lo siguiente:

```

src > js addTask.js > [e] addTask
1  const { v4 } = require("uuid");
2  const AWS = require("aws-sdk");
3
4  const middy = require("@middy/core");
5  const httpJSONBodyParser = require("@middy/http-json-body-parser");
6
7  const addTask = async (event) => {
8      const dynamodb = new AWS.DynamoDB.DocumentClient();
9
10     const { title, description } = event.body;
11     const createdAt = new Date();
12     const id = v4();
13
14     console.log("created id: ", id);
15
16     const newTask = {
17         id,
18         title,
19         description,
20         createdAt,
21         done: false,
22     };
23
24     await dynamodb
25         .put({
26             TableName: "TaskTable",
27             Item: newTask,
28         })
29         .promise();
30
31     return {
32         statusCode: 200,
33         body: JSON.stringify(newTask),
34     };
35 };
36
37 module.exports = {
38     addTask: middy(addTask).use(httpJSONBodyParser()),
39 };

```

Ahora es necesario crear un ID a través de un módulo, en este caso es uuid.

```

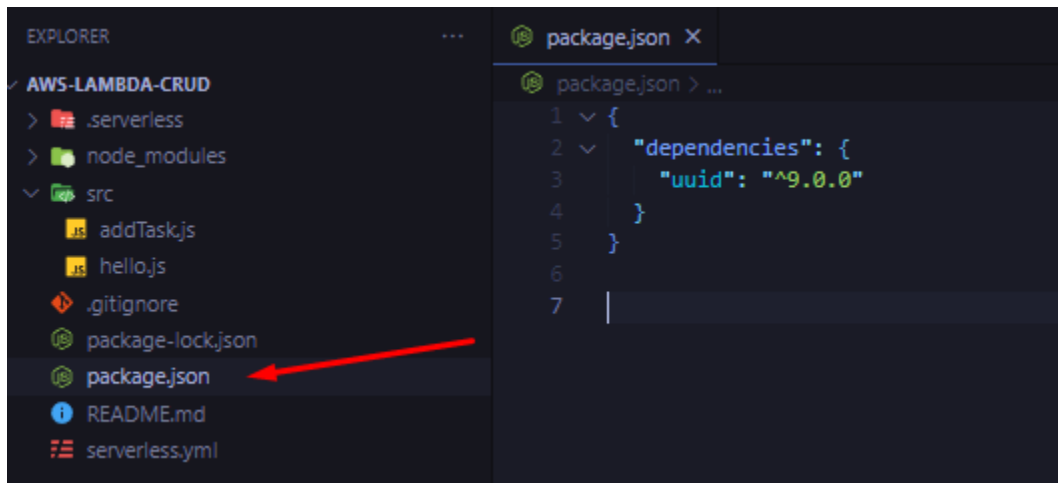
PS C:\Users\WillYer\Desktop\AWS\aws-lambda-crud> npm i uuid
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\WillYer\Desktop\AWS\aws-lambda-crud\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\WillYer\Desktop\AWS\aws-lambda-crud\package.json'
npm WARN aws-lambda-crud No description
npm WARN aws-lambda-crud No repository field.
npm WARN aws-lambda-crud No README data
npm WARN aws-lambda-crud No license field.

+ uuid@9.0.0
added 1 package and audited 1 package in 0.601s
found 0 vulnerabilities

```

Esto debemos referenciarlo como se muestra en la imagen anterior de la función...

Es importante también verificar que este configurado el uuid dentro del Package.json, en caso de que este archivo no este creado, ejecutar el comando “npm init -y” esto solo en caso de que no esté el archivo.



Después de realizar este proceso es necesario instalar otro modulo, en este caso el “npm i aws-sdk”

```
PS C:\Users\WillYer\Desktop\AWS\aws-lambda-crud> npm i aws-sdk
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
npm WARN aws-lambda-crud No repository field.
npm WARN aws-lambda-crud No license field.

+ aws-sdk@2.1266.0
added 30 packages from 69 contributors and audited 31 packages in 6.417s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

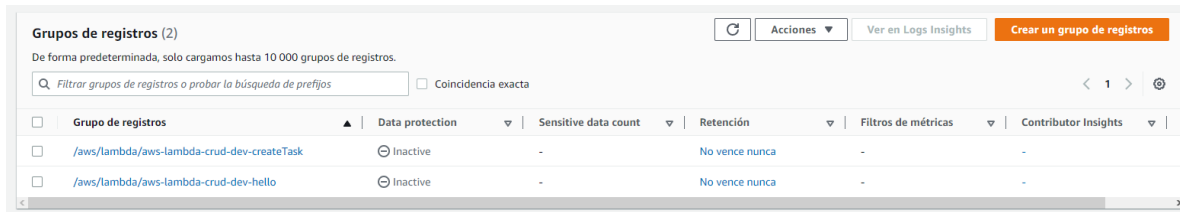
Ahora vamos a actualizar esos cambios en aws...

```
UPDATE_COMPLETE = AWS::CloudFormation::Stack = aws-lambda-crud-dev
Retrieving CloudFormation stack
Removing old service artifacts from S3

✓ Service deployed to stack aws-lambda-crud-dev (85s)

endpoints:
  GET - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/
  POST - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/tasks
functions:
  hello: aws-lambda-crud-dev-hello (14 MB)
  createTask: aws-lambda-crud-dev-createTask (14 MB)
```

Ahora podemos utilizar un servicio de aws llamado CloudWatch, este servicio nos permite monitorear recursos y aplicaciones (LOGS)

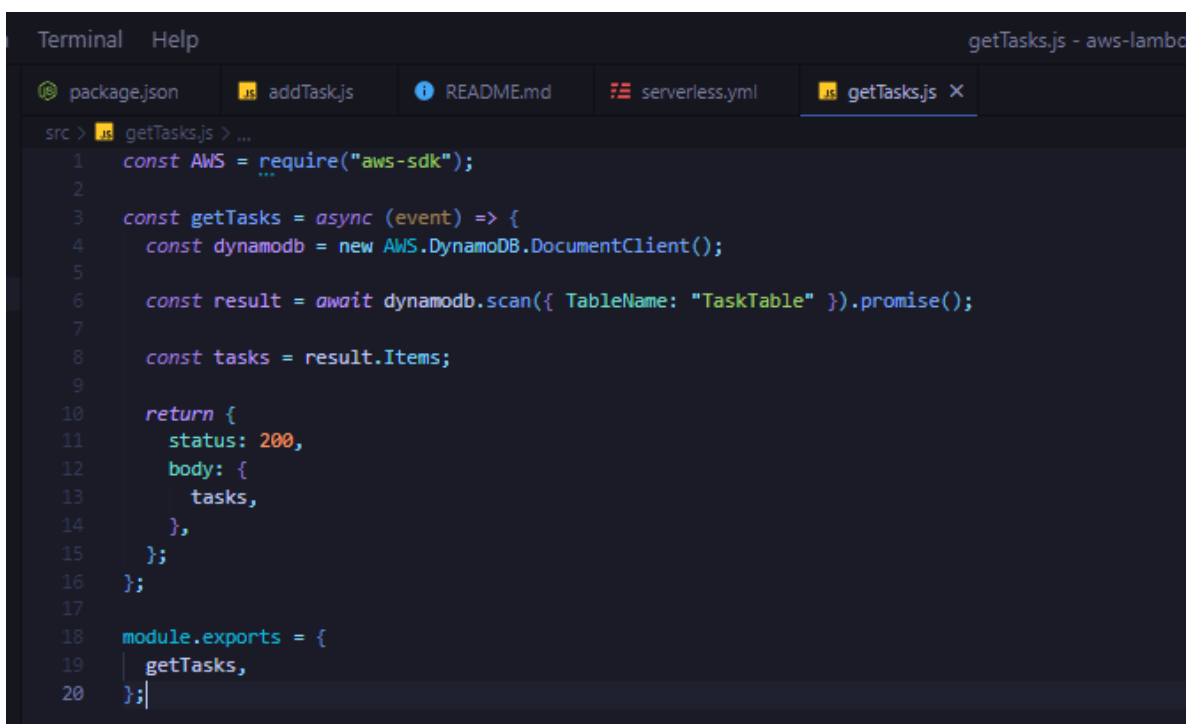


Aquí podremos monitorear todas las solicitudes que hagamos a los endpoints..

## Configurar EndPoint para traer las tareas:

Estos procesos son casi parecidos, siguiendo los mismos pasos del punto anterior para crear la función y llamarla dentro del serverless.yml.

```
getTasks:
  handler: src/getTasks.getTasks
  events:
    - httpApi:
        path: /tasks
        method: get
```



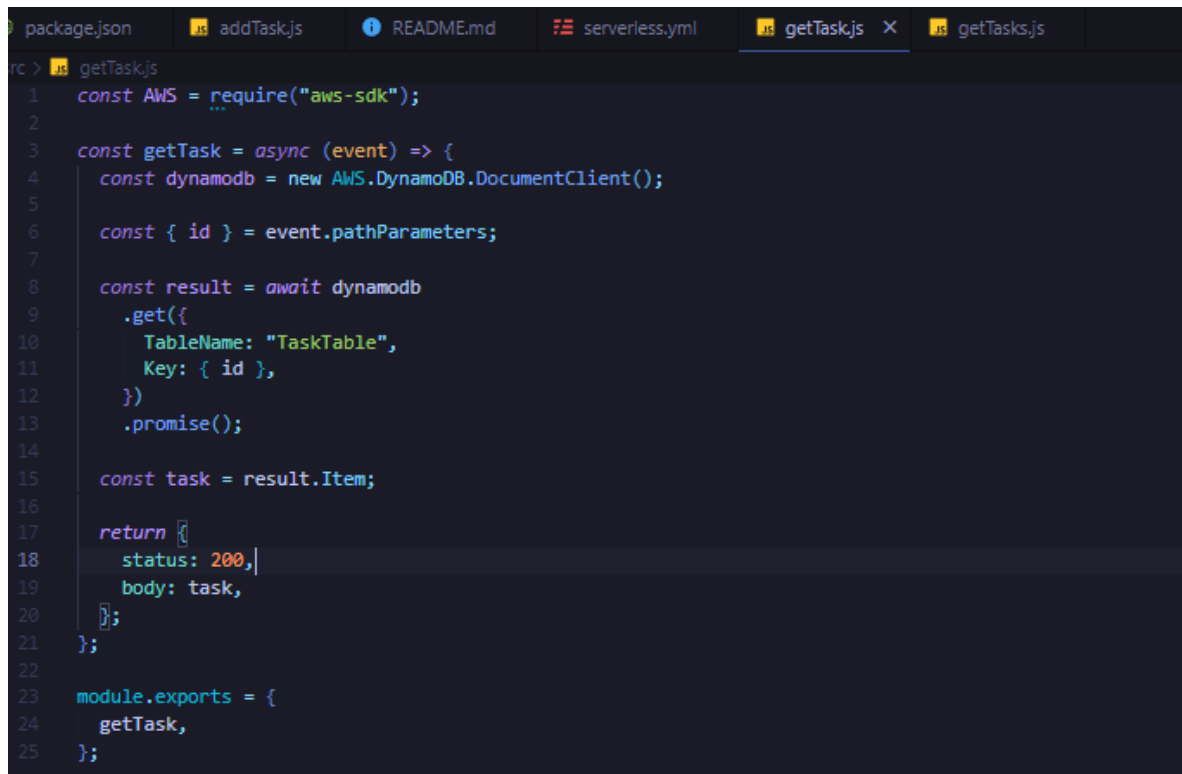
Al actualizar los cambios en aws con serverles, automáticamente va a retornar la url del endpoint el cual puede ser probado con algún recurso para peticiones http.

## Obtener solo las tareas:

Lo que vamos a configurar ahora es como devolver una sola tarea apartir de un ID.

Estos procesos son casi parecidos, siguiendo los mismos pasos del punto anterior para crear la función y llamarla dentro del serverless.yml.

```
getTask:  
  handler: src/getTask.getTask  
  events:  
    - httpApi:  
        path: /tasks/{id}  
        method: get
```



```
rc > getTask.js  
1  const AWS = require("aws-sdk");  
2  
3  const getTask = async (event) => {  
4    const dynamodb = new AWS.DynamoDB.DocumentClient();  
5  
6    const { id } = event.pathParameters;  
7  
8    const result = await dynamodb  
9      .get({  
10       TableName: "TaskTable",  
11       Key: { id },  
12     })  
13     .promise();  
14  
15    const task = result.Item;  
16  
17    return {  
18      status: 200,  
19      body: task,  
20    };  
21  };  
22  
23  module.exports = {  
24    getTask,  
25  };  

```

Al actualizar los cambios en aws con serverless, automáticamente va a retornar la url del endpoint el cual puede ser probado con algún recurso para peticiones http.

## Actualizar una tarea:

Lo que vamos a configurar ahora es como actualizar una sola tarea apartir de un ID.

Estos procesos son casi parecidos, siguiendo los mismos pasos del punto anterior para crear la función y llamarla dentro del serverless.yml.

```
updateTask:
  handler: src/updateTask.updateTask
  events:
    - httpApi:
        path: /tasks/{id}
        method: put
```

```
const uuid = require("uuid");
const AWS = require("aws-sdk");

const updateTask = async (event) => {
  const dynamodb = new AWS.DynamoDB.DocumentClient();
  const { id } = event.pathParameters;

  const { done } = JSON.parse(event.body);

  await dynamodb
    .update({
      TableName: "TaskTable",
      Key: { id },
      UpdateExpression: "set done = :done",
      ExpressionAttributeValues: {
        ":done": done,
      },
      ReturnValues: "ALL_NEW",
    })
    .promise();

  return {
    statusCode: 200,
    body: JSON.stringify({
      message: "task updated",
    }),
  };
};

module.exports = {
  updateTask,
};
```

Al actualizar los cambios en aws con serverless, automáticamente va a retornar la url del endpoint el cual puede ser probado con algún recurso para peticiones http.

## Eliminar Tareas:

Lo que vamos a configurar ahora es como eliminar una sola tarea apartir de un ID.

Estos procesos son casi parecidos, siguiendo los mismos pasos del punto anterior para crear la función y llamarla dentro del serverless.yml.

```
deleteTask:
  handler: src/deleteTask.deleteTask
  events:
    - httpApi:
        path: /tasks/{id}
        method: delete
```

```
src > . deleteTask.js > deleteTask > body
1  const AWS = require("aws-sdk");
2
3  const deleteTask = async (event) => {
4    const dynamodb = new AWS.DynamoDB.DocumentClient();
5    const { id } = event.pathParameters;
6
7    await dynamodb
8      .delete({
9        TableName: "TaskTable",
10       Key: {
11         id,
12       },
13     })
14     .promise();
15
16     return {
17       status: 200,
18       body: {
19         message: 'Deleted Task'
20       }
21     };
22   };
23
24   module.exports = {
25     deleteTask,
26   };
27
```

Al actualizar los cambios en aws con serverles, automáticamente va a retornar la url del endpoint el cual puede ser probado con algún recurso para peticiones http.

## Ya con estas configuraciones, nuestro CRUD, estaría terminado.

Ahora vamos a configurar los Middlewares.

### ¿Qué es un Middleware?

Los middleware son funciones que se ejecutan antes de que llegue una función principal.

Para utilizar Middleware en nuestro proyecto es necesario instalar un módulo llamado middy, para esto vamos a utilizar nuevamente node..

```
npm i @middy/core
```

Esto nos permite procesar datos antes de que llegue la función principal con la palabra clave use.

```
PS C:\Users\WillYer\Desktop\AWS\aws-lambda-crud> npm i @middy/core
npm WARN notsup Unsupported engine for @middy/core@4.0.0: wanted: {"node": ">=16"} (current: {"node": "14.16.0", "npm": "6.14.11"})
npm WARN notsup Not compatible with your version of node/npm: @middy/core@4.0.0
npm WARN aws-lambda-crud No repository field.
npm WARN aws-lambda-crud No license field.

+ @middy/core@4.0.0
added 1 package from 1 contributor and audited 32 packages in 2.023s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Ahora instalamos `npm i @middy/http-json-body-parser`

```
PS C:\Users\WillYer\Desktop\AWS\aws-lambda-crud> npm i @middy/http-json-body-parser
npm WARN notsup Unsupported engine for @middy/http-json-body-parser@4.0.0: wanted: {"node": ">=16"} (current: {"node": "14.16.0", "npm": "6.14.11"})
npm WARN notsup Not compatible with your version of node/npm: @middy/http-json-body-parser@4.0.0
npm WARN notsup Unsupported engine for @middy/util@4.0.0: wanted: {"node": ">=16"} (current: {"node": "14.16.0", "npm": "6.14.11"})
npm WARN notsup Not compatible with your version of node/npm: @middy/util@4.0.0
npm WARN aws-lambda-crud No repository field.
npm WARN aws-lambda-crud No license field.

+ @middy/http-json-body-parser@4.0.0
added 2 packages from 1 contributor and audited 34 packages in 5.743s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Configuración:

```
const middy = require("@middy/core");
const httpJSONBodyParser = require("@middy/http-json-body-parser");
```

```
35 };
36
37 module.exports = {
38   addTask: middy(addTask).use(httpJSONBodyParser()),
39 };

```



## Actualizamos serverless en aws....

```
✓ Service deployed to stack aws-lambda-crud-dev (78s)

endpoints:
GET - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/
POST - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/tasks
GET - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/tasks
GET - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/tasks/{id}
PUT - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/tasks/{id}
DELETE - https://yv5amscicj.execute-api.us-east-1.amazonaws.com/tasks/{id}

functions:
hello: aws-lambda-crud-dev-hello (14 MB)
createTask: aws-lambda-crud-dev-createTask (14 MB)
```

Lambda > Funciones

Funciones (6) Última obtención ahora Acciones [Crear una función](#)

<input type="checkbox"/>	Nombre de la función	Descripción	Tipo de paquete	Tiempo de ejecución	Última modificación
<input type="checkbox"/>	aws-lambda-crud-dev-getTasks	-	Zip	Node.js 12.x	hace 3 minutos
<input type="checkbox"/>	aws-lambda-crud-dev-hello	-	Zip	Node.js 12.x	hace 3 minutos
<input type="checkbox"/>	aws-lambda-crud-dev-deleteTask	-	Zip	Node.js 12.x	hace 3 minutos
<input type="checkbox"/>	aws-lambda-crud-dev-updateTask	-	Zip	Node.js 12.x	hace 3 minutos
<input type="checkbox"/>	aws-lambda-crud-dev-createTask	-	Zip	Node.js 12.x	hace 3 minutos
<input type="checkbox"/>	aws-lambda-crud-dev-getTask	-	Zip	Node.js 12.x	hace 3 minutos

Grupos de registros (6) Acciones [Ver en Logs Insights](#) [Crear un grupo de registros](#)

De forma predeterminada, solo cargamos hasta 10 000 grupos de registros.

☐ Coincidencia exacta

<input type="checkbox"/>	Grupo de registros	Data protection	Sensitive data count	Retención	Filtros de métricas	Contributor Insights
<input type="checkbox"/>	/aws/lambda/aws-lambda-crud-dev-createTask	Inactive	-	No vence nunca	-	-
<input type="checkbox"/>	/aws/lambda/aws-lambda-crud-dev-deleteTask	Inactive	-	No vence nunca	-	-
<input type="checkbox"/>	/aws/lambda/aws-lambda-crud-dev-getTask	Inactive	-	No vence nunca	-	-
<input type="checkbox"/>	/aws/lambda/aws-lambda-crud-dev-getTasks	Inactive	-	No vence nunca	-	-
<input type="checkbox"/>	/aws/lambda/aws-lambda-crud-dev-hello	Inactive	-	No vence nunca	-	-
<input type="checkbox"/>	/aws/lambda/aws-lambda-crud-dev-updateTask	Inactive	-	No vence nunca	-	-

DynamoDB > Tablas

Tablas (1) Información Acciones

<input type="checkbox"/>	Nombre	Estado	Clave de partición	Clave de ordenación	Índices	Modo de capacidad de lectura	Modo de capacidad de escritura
<input type="checkbox"/>	TaskTable	Activo	id (S)	-	0	Bajo demanda	Bajo demanda