Api GW + lambdas (CRUD) + DB + Serverless

Estudiantes:

Johan Calixto Ramírez Acosta

Gian Lucas Figueroa Felizola

Leandro Alfonso Maldonado Ospino

Docente:

Salomón Segundo de la hoz

Electiva:

Cloud Computing

Facultad de Ingenierías y Tecnológicas

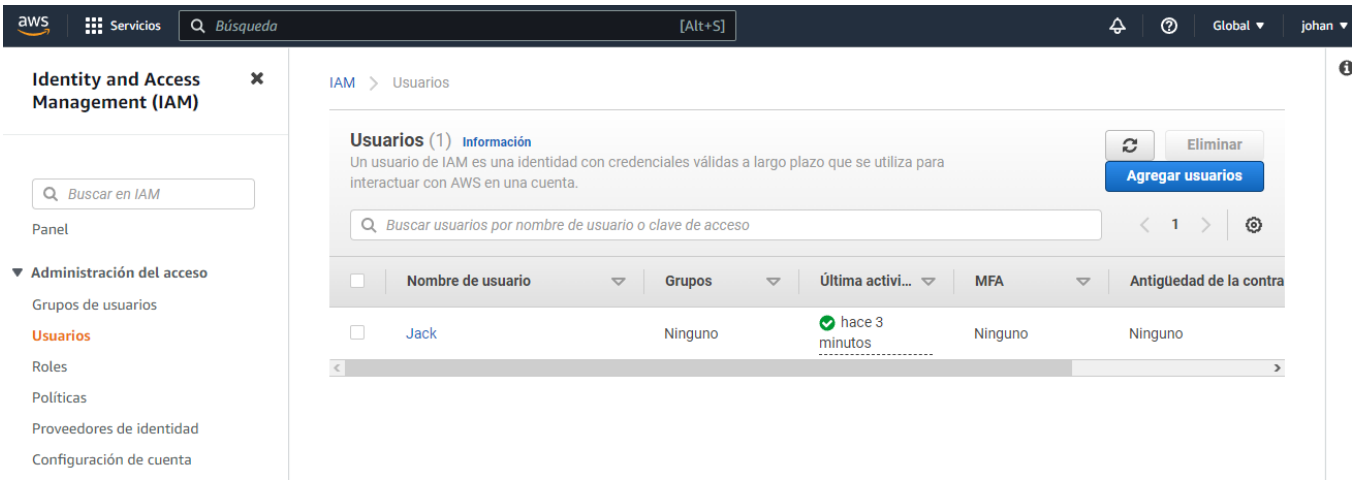
Universidad Popular del Cesar

Valledupar – Cesar

2022

Api GW + lambdas (CRUD) + DB + Serverless

En AWS es importante crear un usuario en Identity and Access Management (IAM)

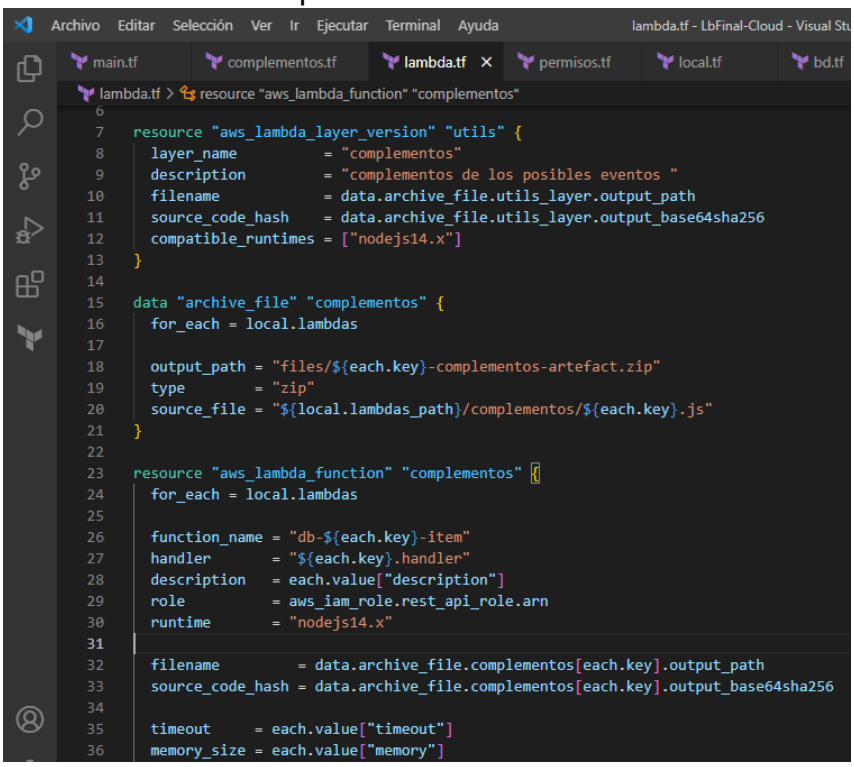


Escribimos el siguiente código en Visual Code

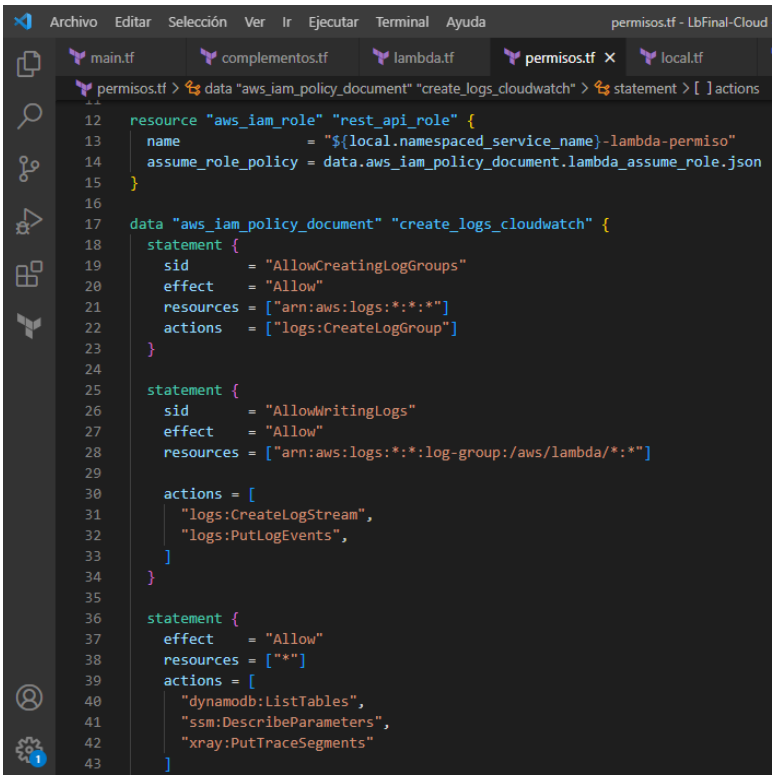
main.tf > ...

```
1
2   provider "aws" {
3     region  = var.aws_region
4     profile = var.aws_profile
5
6     default_tags {
7       tags = {
8         Project    = "Rest Api GW"
9         CreatedAt  = "2022-12-08"
10        ManagedBy  = "Terraform"
11        Owner      = "Johan Ramirez"
12        Env        = var.env
13      }
14    }
15  }
```

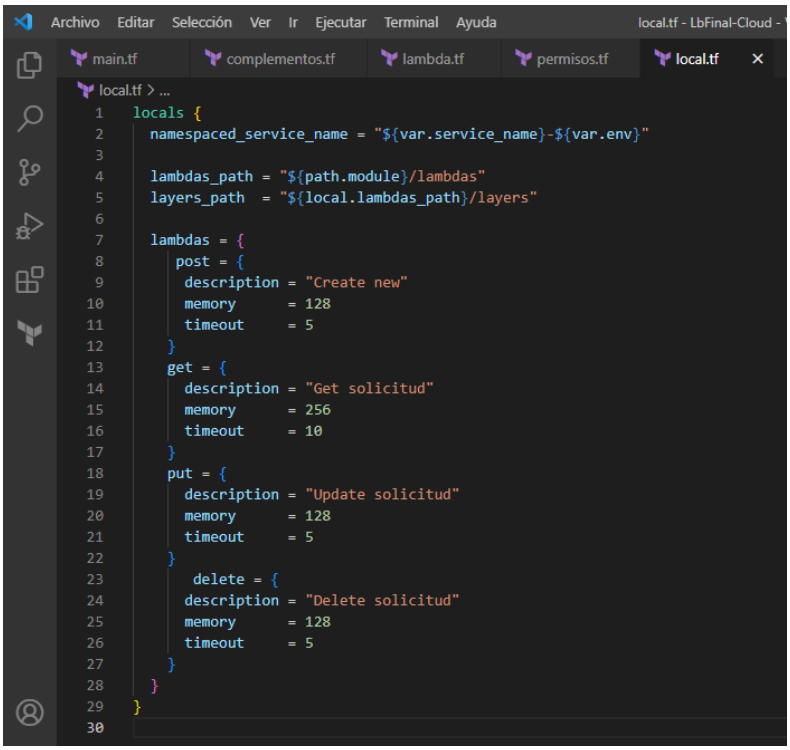
Creamos el archivo que tiene informacion de las lambdas



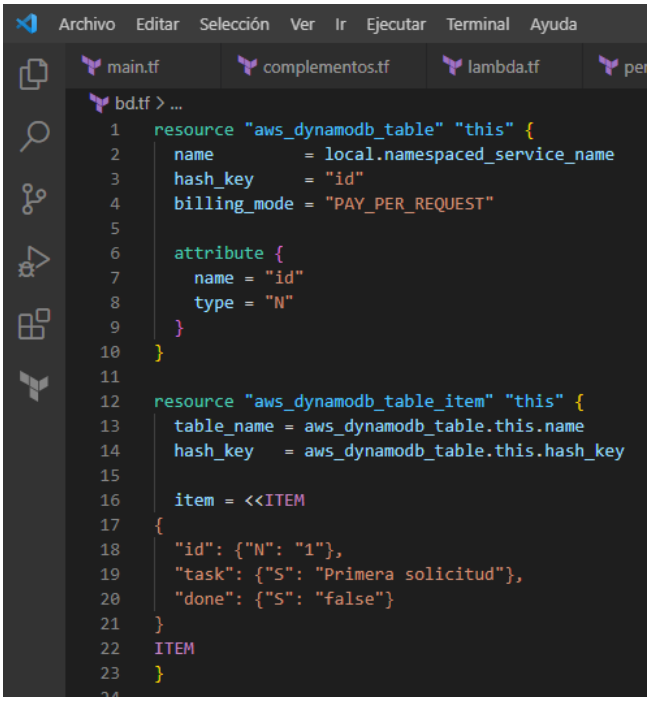
Definimos todos los permisos que utilizaremos



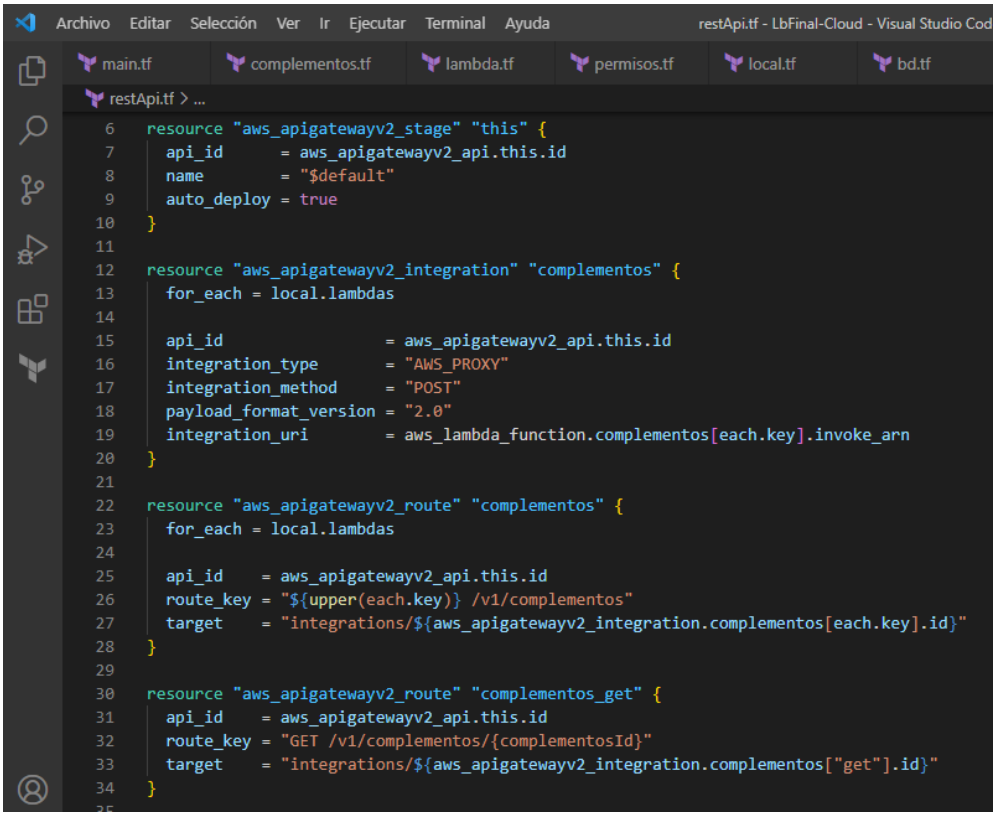
Creamos las solicitudes que se trabajan de manera local



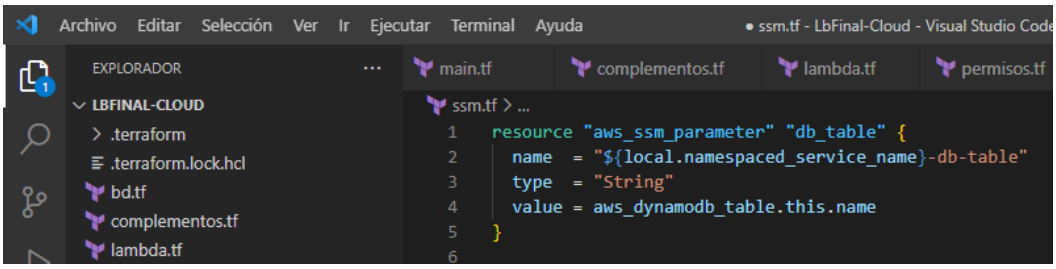
Archivo para la creación de la base de datos



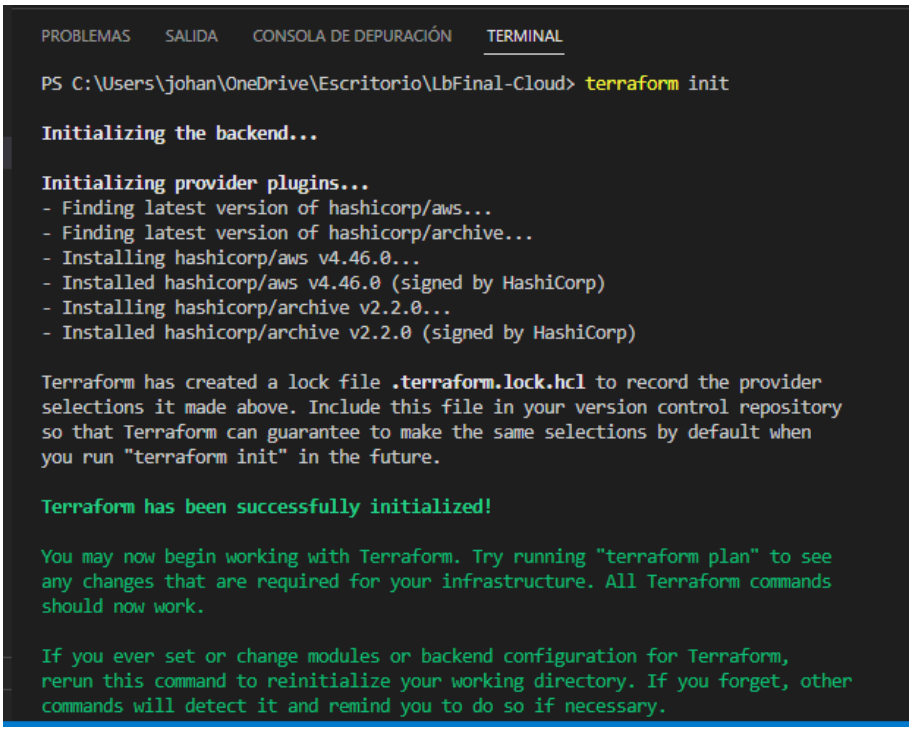
Definimos la configuración del archivo el cual va a completar e integral los servicios

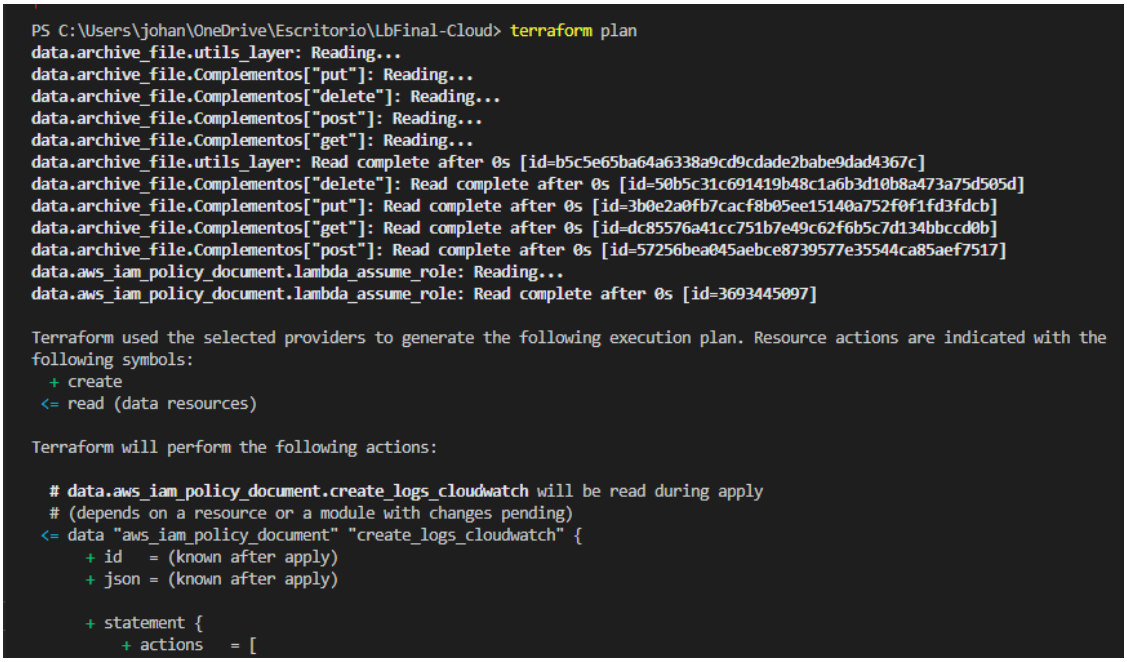


Creamos un archivo para la configuración del Parameter Store



Inicializamos el directorio de trabajo que contiene la configuración de Terraform



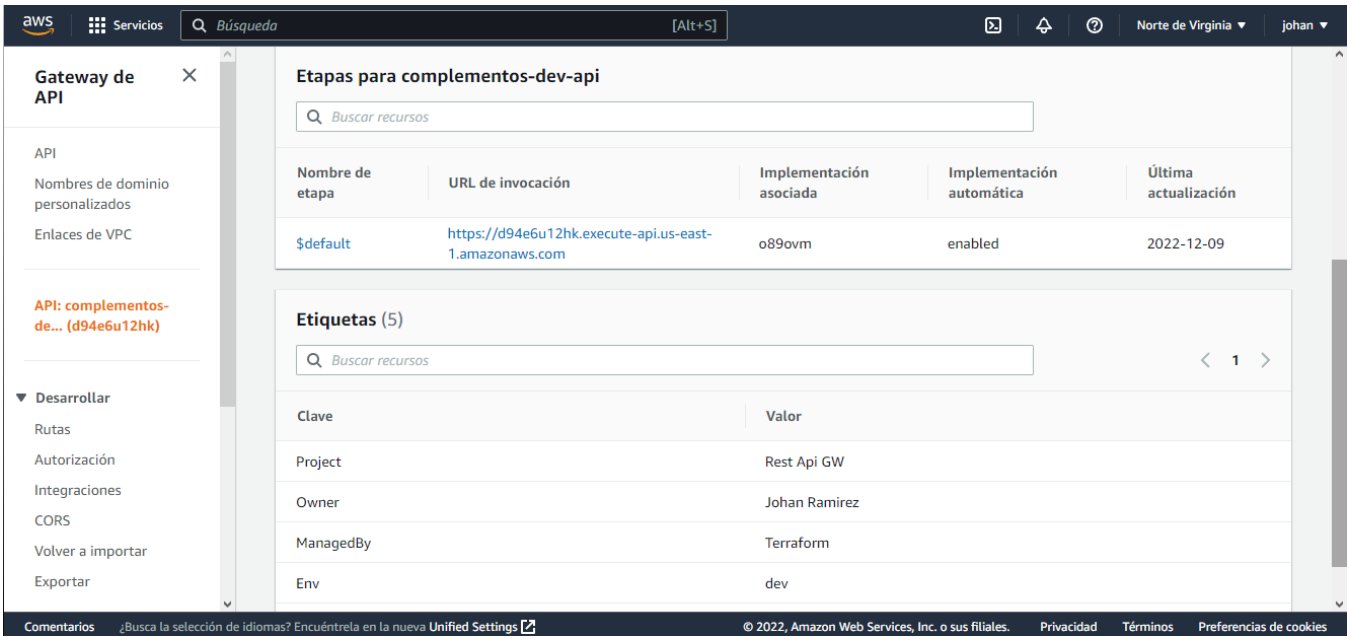


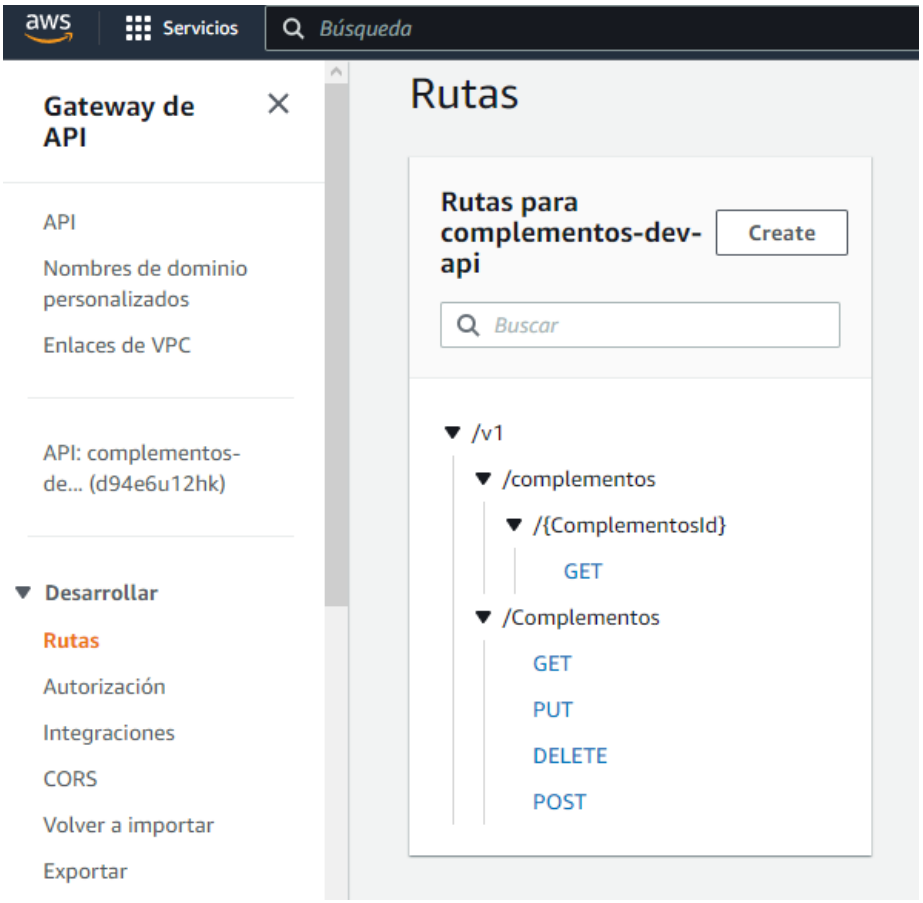
Por último, aplicamos

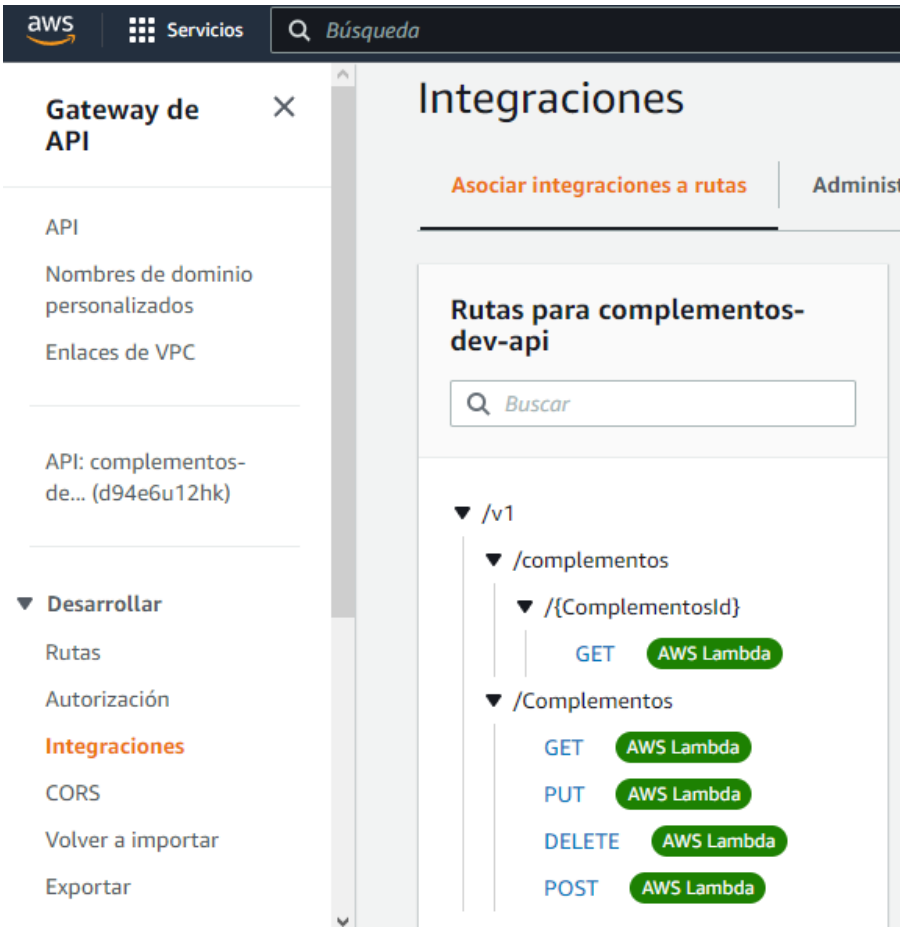
```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run
"terraform apply" now.
PS C:\Users\johan\OneDrive\Escritorio\LbFinal-Cloud> terraform apply -auto-approve
data.archive_file.Complementos["put"]: Reading...
data.archive_file.utils_layer: Reading...
data.archive_file.Complementos["post"]: Reading...
data.archive_file.Complementos["get"]: Reading...
data.archive_file.Complementos["delete"]: Reading...
data.archive_file.Complementos["put"]: Read complete after 0s [id=3b0e2a0fb7cacf8b05ee15140a752f0f1fd3fdcb]
data.archive_file.utils_layer: Read complete after 0s [id=b5c5e65ba64a6338a9cd9cdade2babe9dad4367c]
data.archive_file.Complementos["post"]: Read complete after 0s [id=57256bea045aebce8739577e35544ca85aef7517]
data.archive_file.Complementos["get"]: Read complete after 0s [id=dc85576a41cc751b7e49c62f6b5c7d134bbccd0b]
data.archive_file.Complementos["delete"]: Read complete after 0s [id=50b5c31c691419b48c1a6b3d10b8a473a75d505d]
data.aws_iam_policy_document.lambda_assume_role: Reading...
aws_lambda_layer_version.utils: Refreshing state... [id=arn:aws:lambda:us-east-1:476786095040:layer:Complementos:2]
aws_apigatewayv2_api.this: Refreshing state... [id=d94e6u12hk]
aws_dynamodb_table.this: Refreshing state... [id=complementos-dev]
data.aws_iam_policy_document.lambda_assume_role: Read complete after 0s [id=3693445097]
aws_iam_role.rest_api_role: Refreshing state... [id=complementos-dev-lambda-permiso]
aws_apigatewayv2_stage.this: Refreshing state... [id=$default]
aws_dynamodb_table_item.this: Refreshing state... [id=complementos-dev|id|||1]
aws_ssm_parameter.db_table: Refreshing state... [id=complementos-dev-db-table]
data.aws_iam_policy_document.create_logs_cloudwatch: Reading...
```

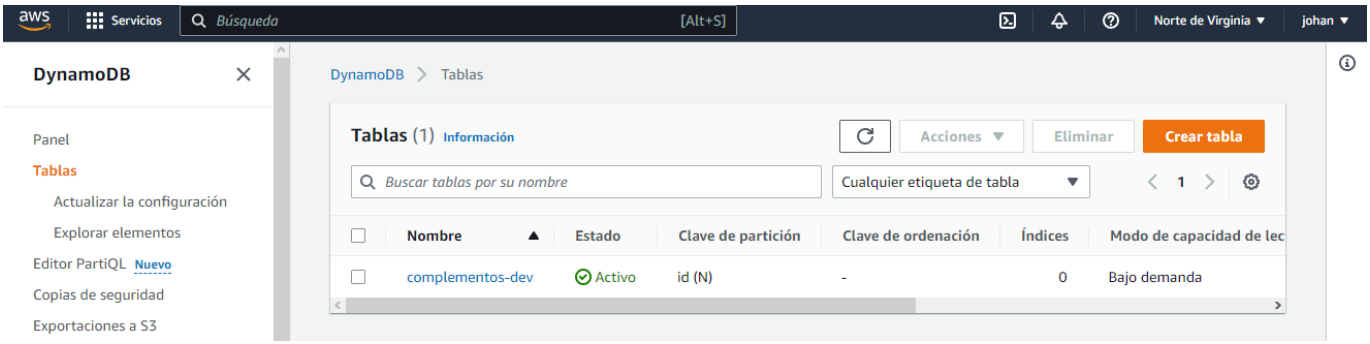
Luego podemos verificar la creación en AWS

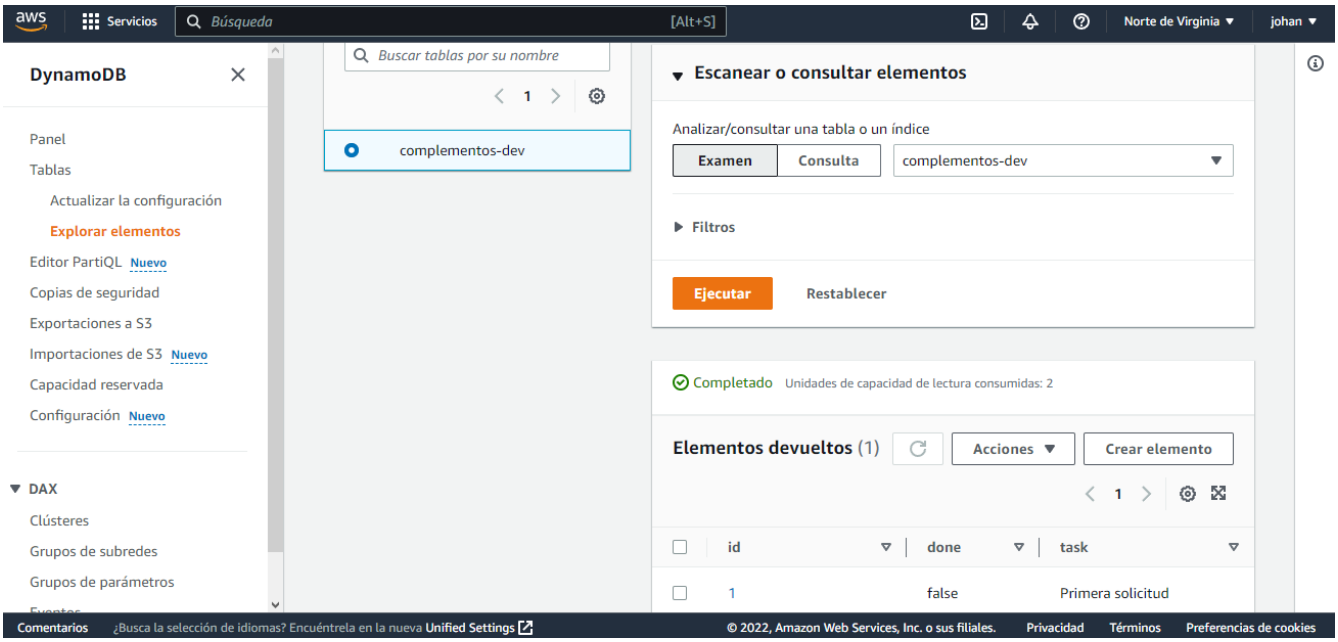

aws
Servicios
Búsqueda
[Alt+S]
Norte de Virginia
johan
Gateway de API
API
Nombres de dominio personalizados
Enlaces de VPC
API: complementos-de... (d94e6u12hk)
Desarrollar
Rutas
Autorización
Integraciones
CORS
Volver a importar
Exportar
Etapas para complementos-dev-api
Buscar recursos
Nombre de etapa
URL de invocación
Implementación asociada
Implementación automática
Última actualización
$default
https://d94e6u12hk.execute-api.us-east-1.amazonaws.com
o89ovm
enabled
2022-12-09
Etiquetas (5)
Buscar recursos
1
Clave
Valor
Project
Rest Api GW
Owner
Johan Ramirez
ManagedBy
Terraform
Env
dev
Comentarios
¿Busca la selección de idiomas? Encuéntrela en la nueva Unified Settings
© 2022, Amazon Web Services, Inc. o sus filiales.
Privacidad
Términos
Preferencias de cookies


aws
Servicios
Búsqueda
Gateway de API
API
Nombres de dominio personalizados
Enlaces de VPC
API: complementos-de... (d94e6u12hk)
Desarrollar
Rutas
Autorización
Integraciones
CORS
Volver a importar
Exportar
Rutas
Rutas para complementos-dev-api
Create
Buscar
/v1
/complementos
/{ComplementosId}
GET
/Complementos
GET
PUT
DELETE
POST



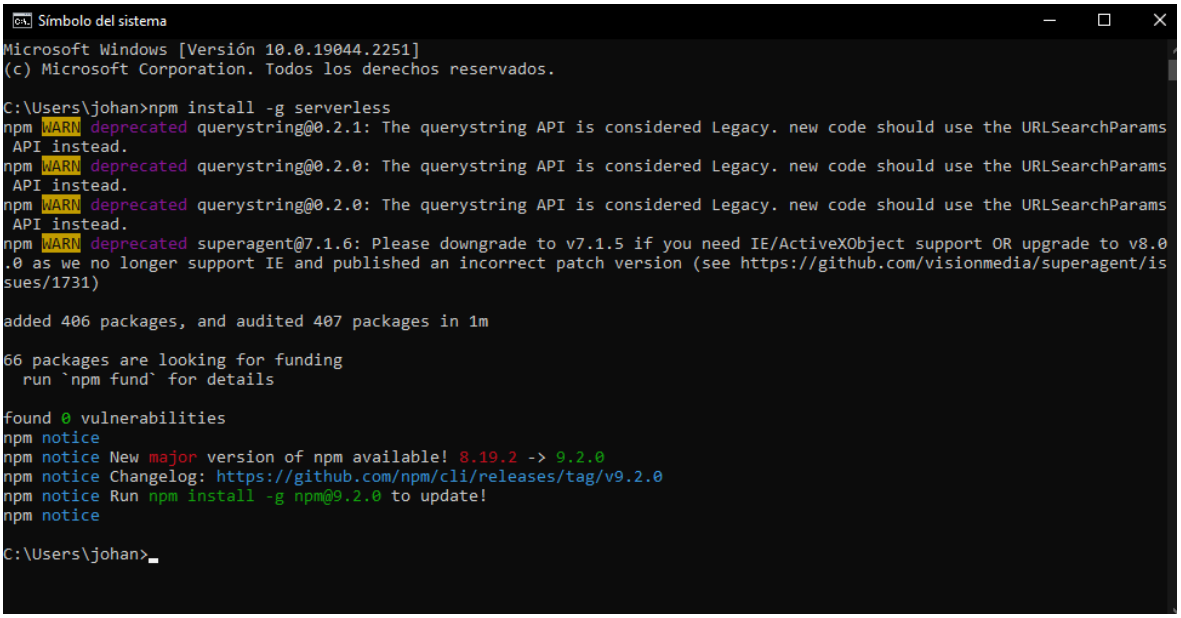
También podemos observar la tabla que creamos en la base de datos



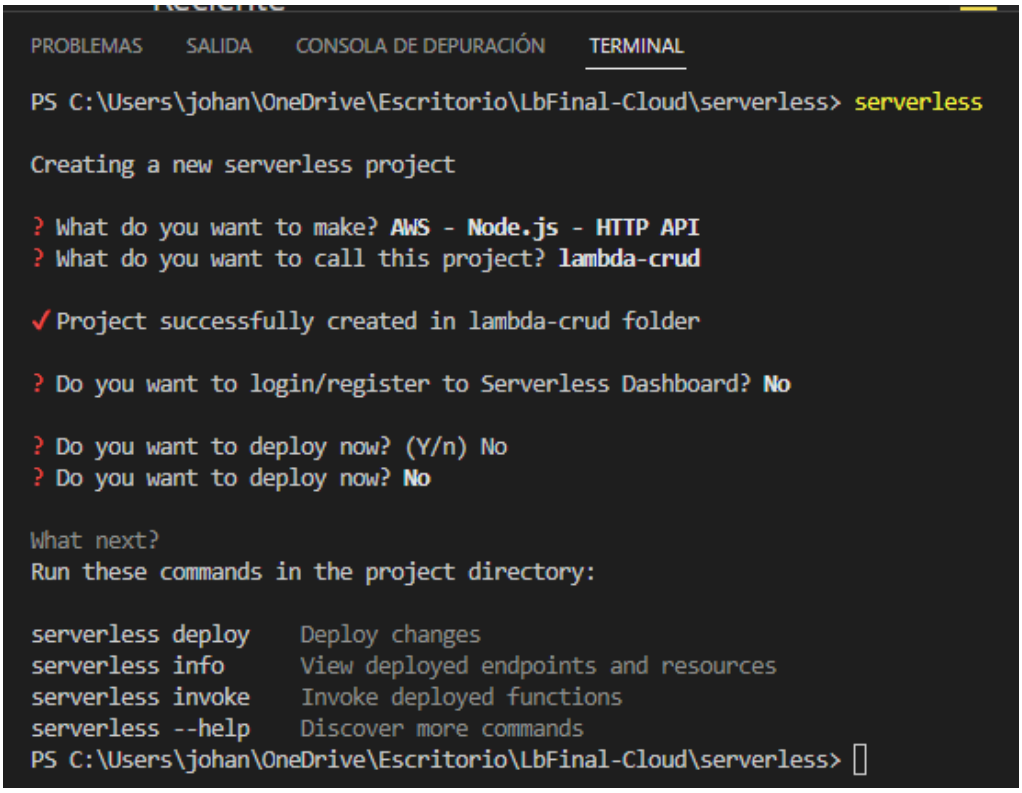


Serverless

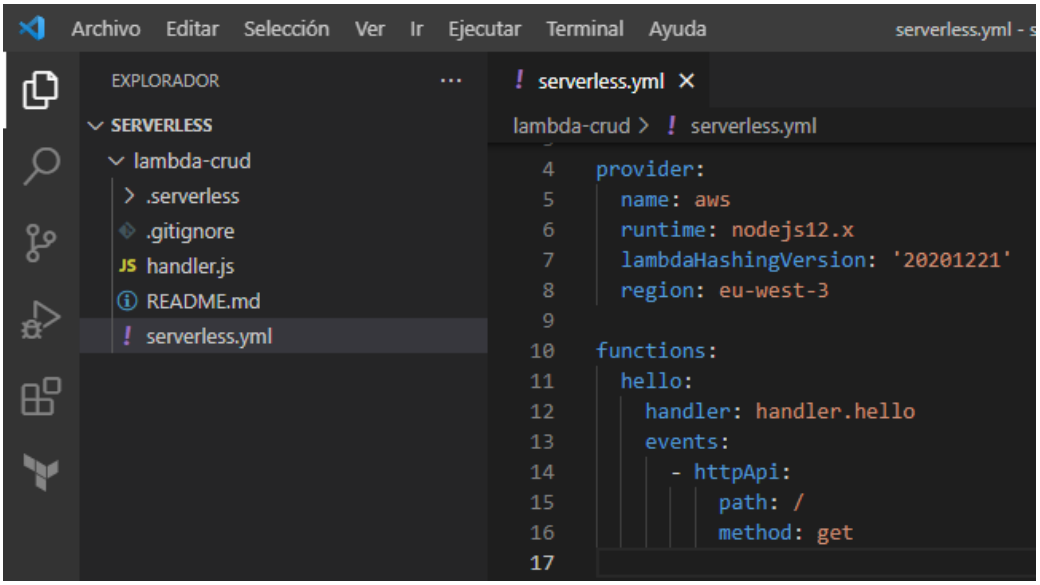
Procedemos a instalar serverless



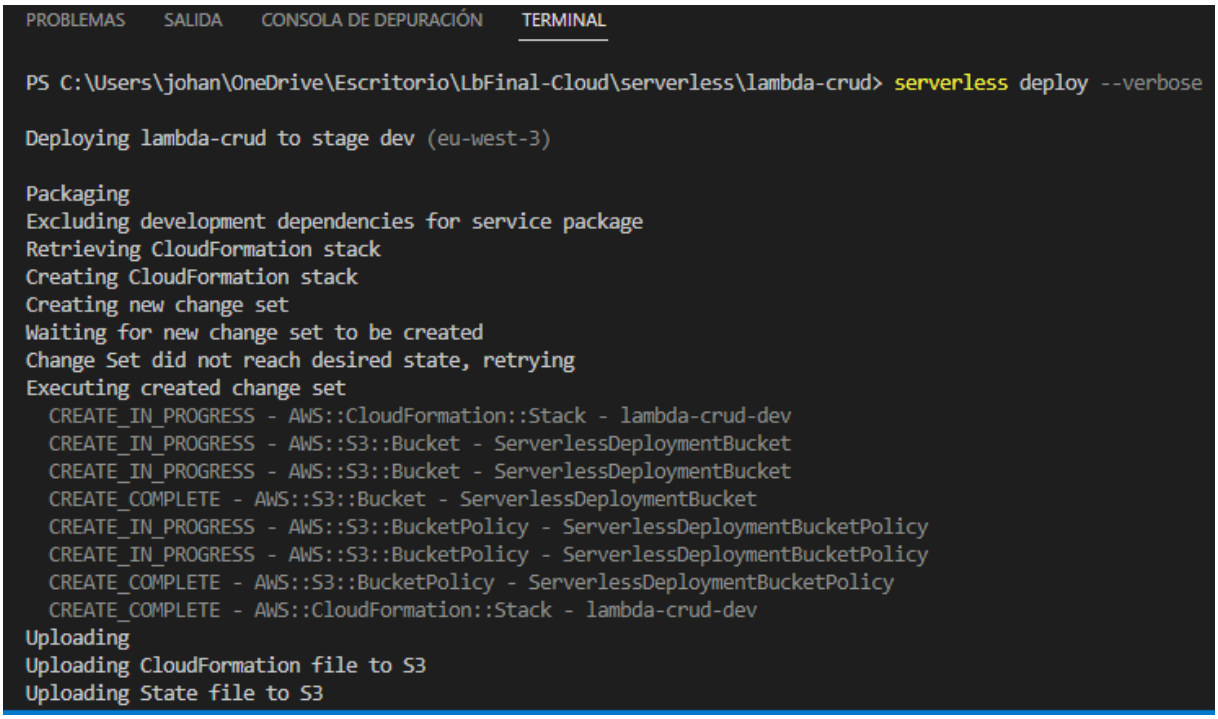
Ya teniendo Serverless podemos crear nuestro proyecto

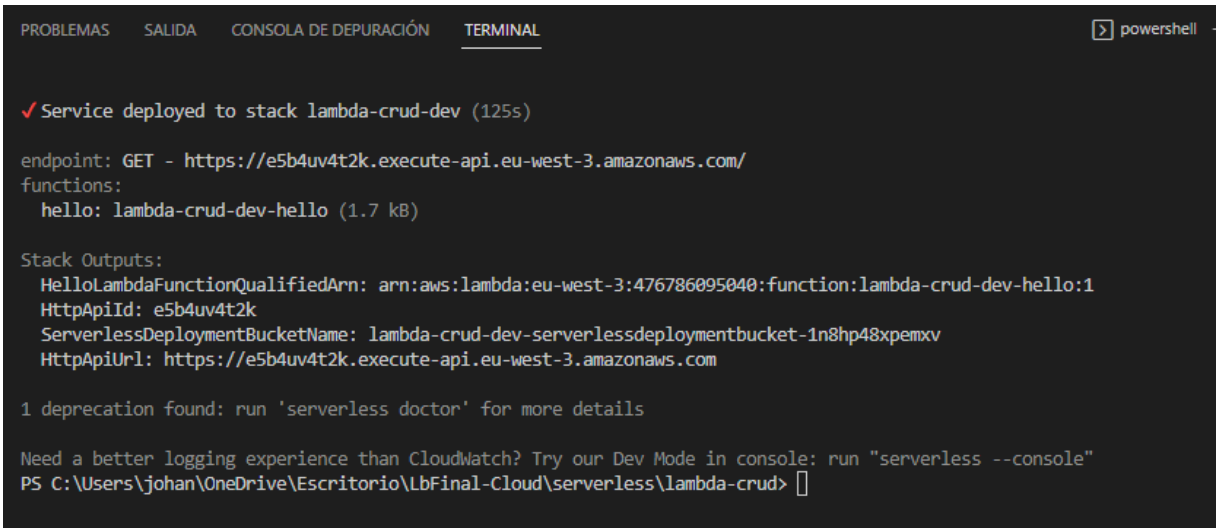


Definimos la región en el archivo serverless.yml

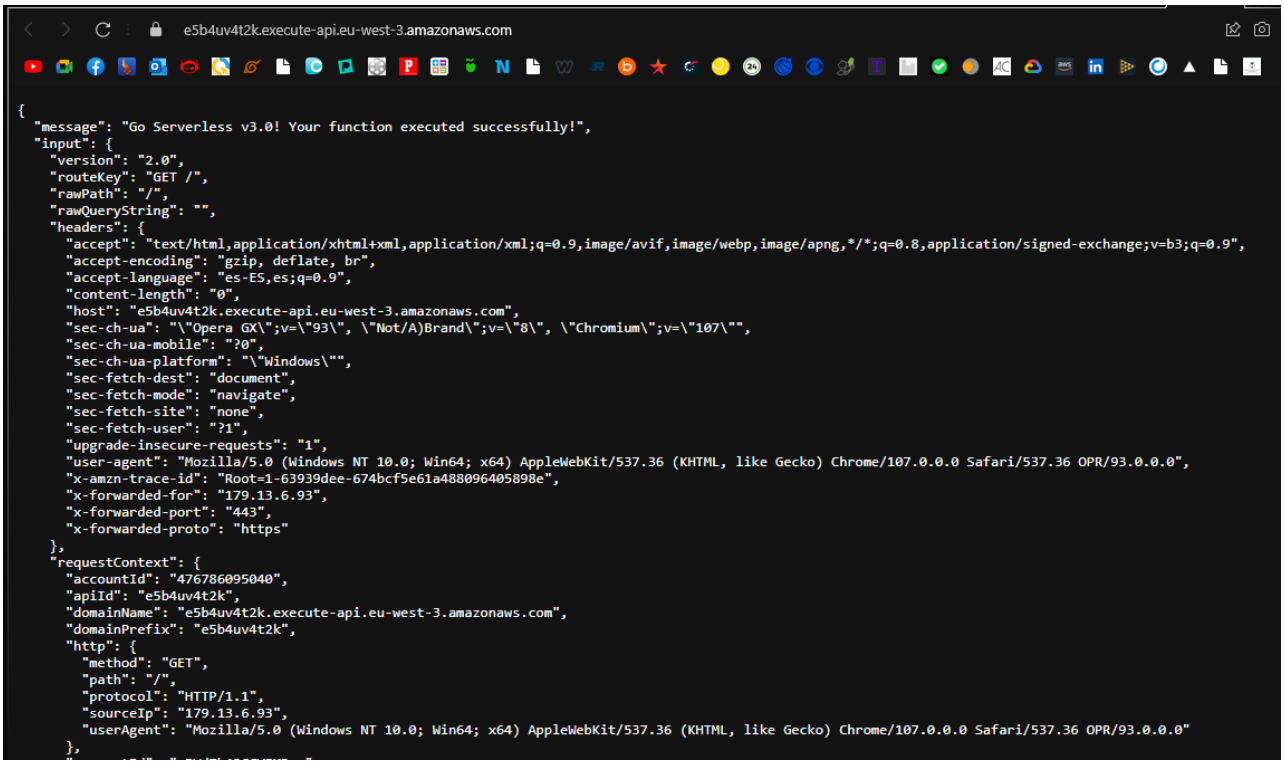


Ejecutamos el serverless

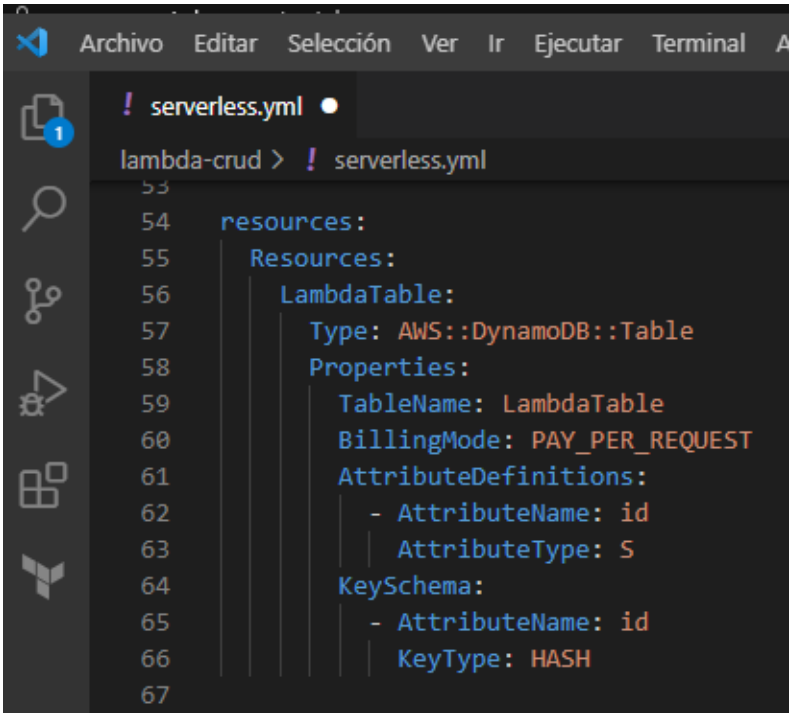




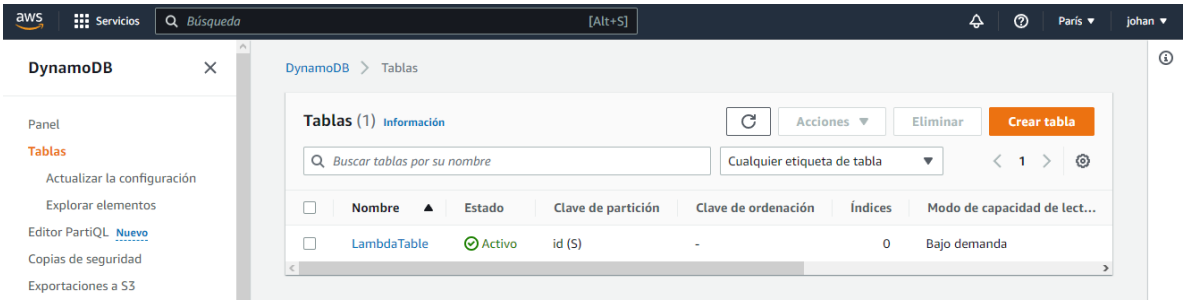
Podemos observar el objeto Json



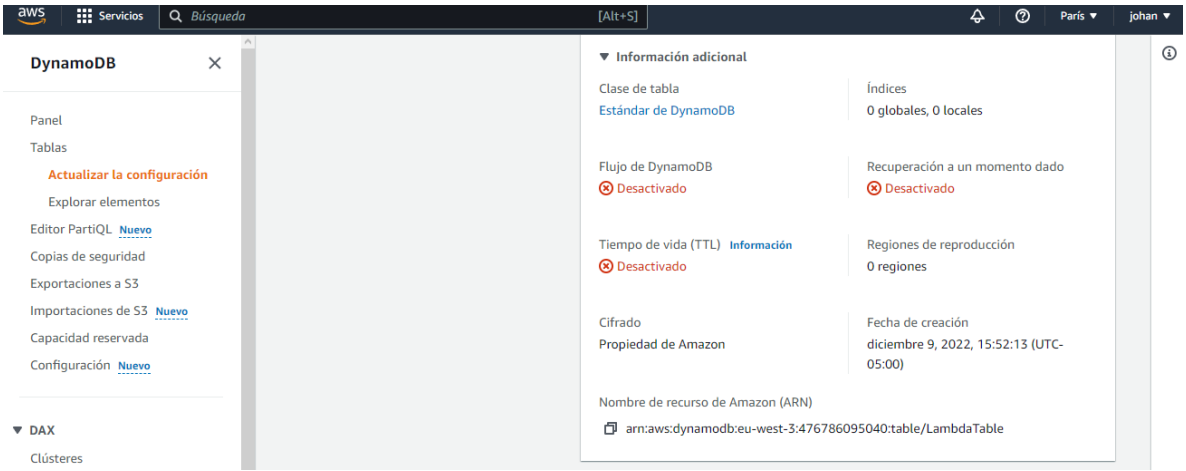
Configuramos el DynamoBd en el mismo archivo

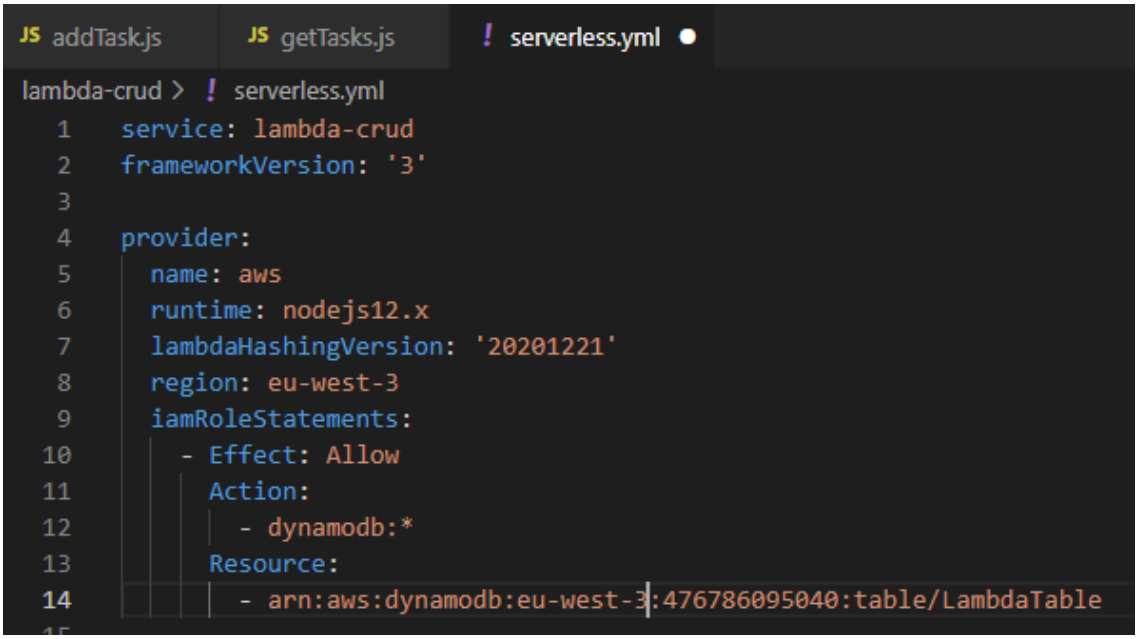


Luego de subir los cambios podemos observa que en AWS nuestra tabla fue creada

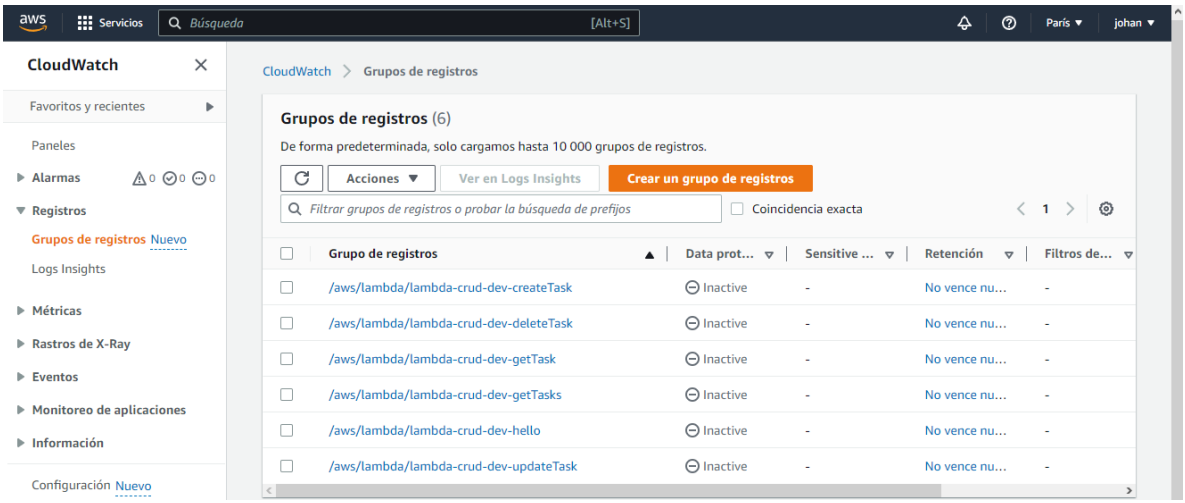


Con la información que nos proporciona aws completamos nuestro archivo Serveless.yml

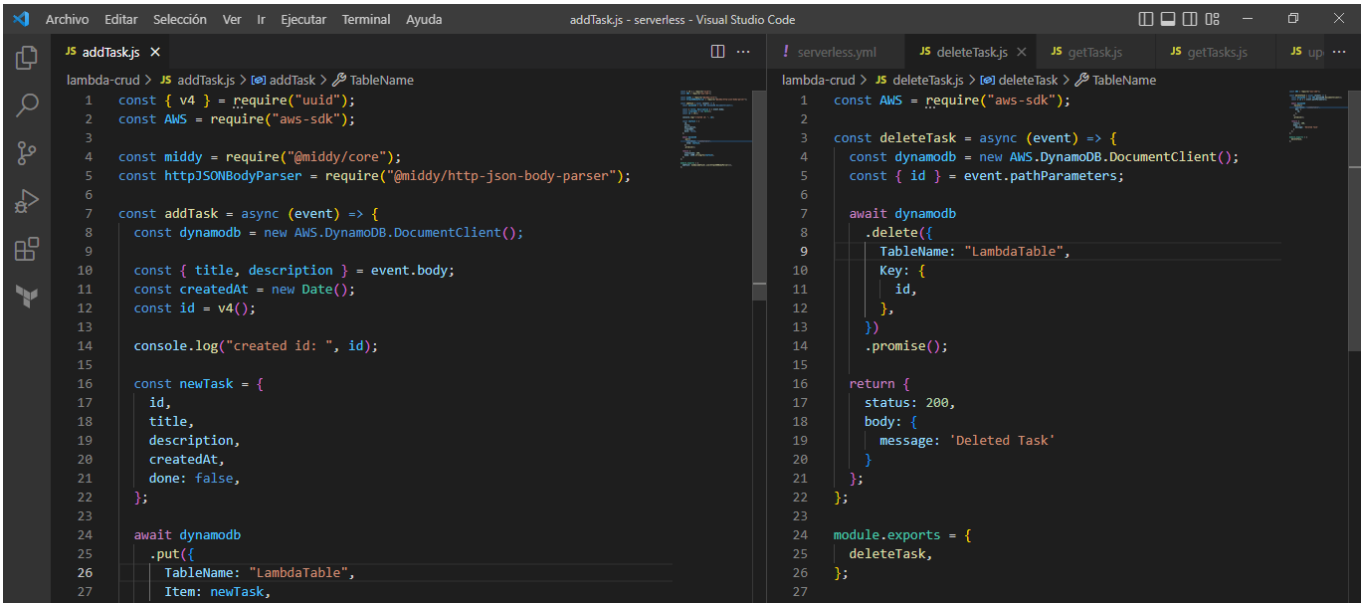


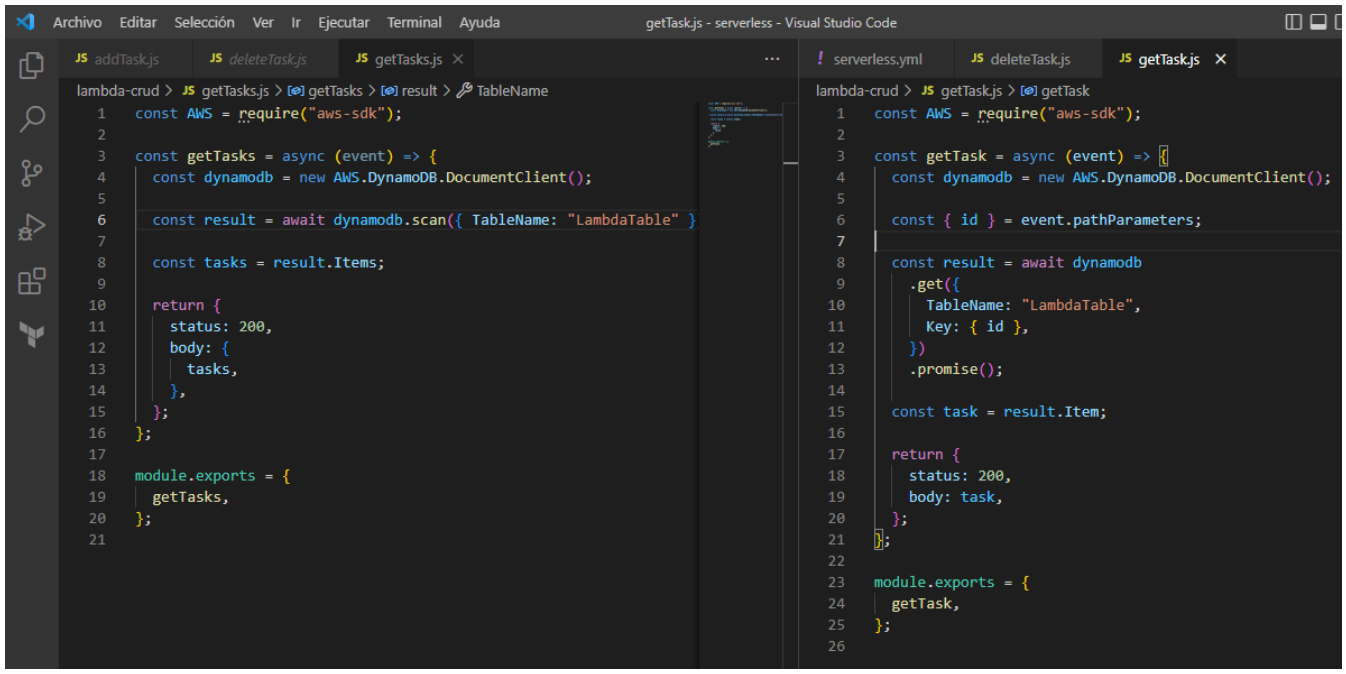


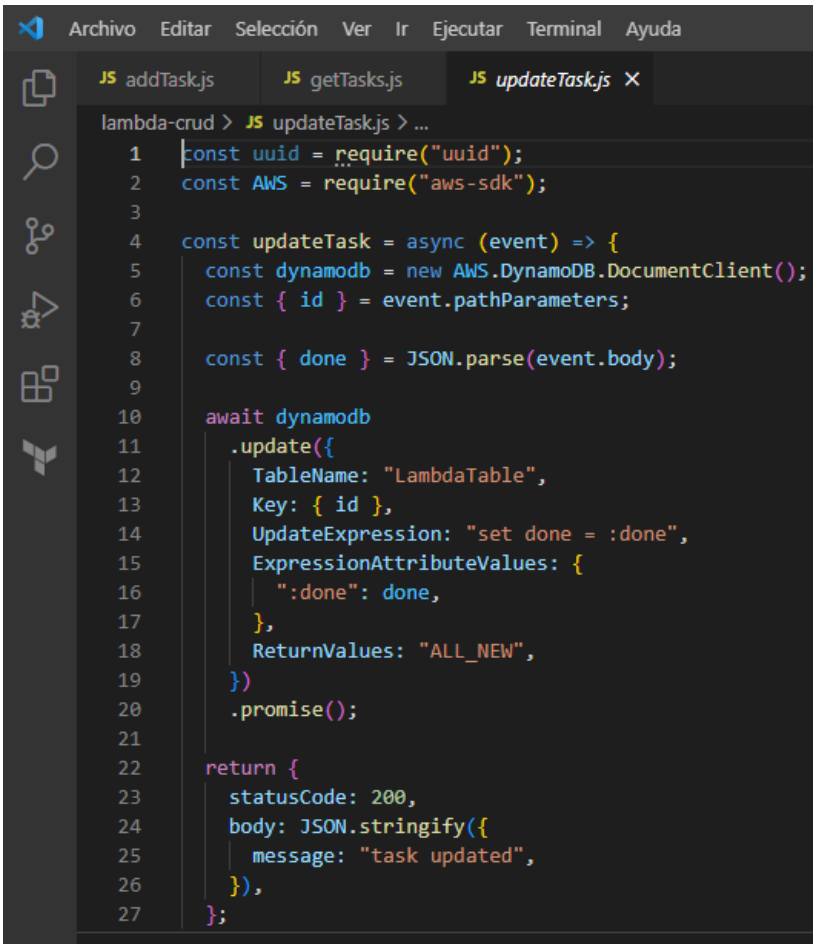
Ahora con el servicio CloudWatch podemos ver los mensajes que salen de nuestro servidor



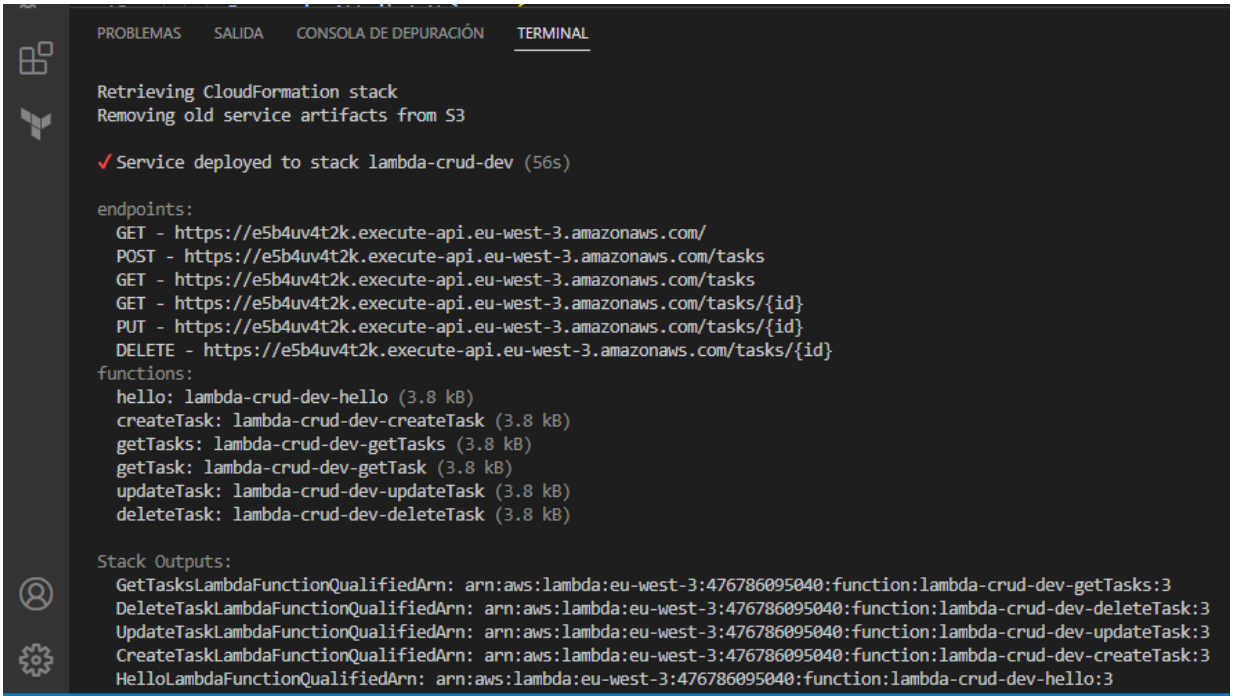
Procedemos a codificar el crud



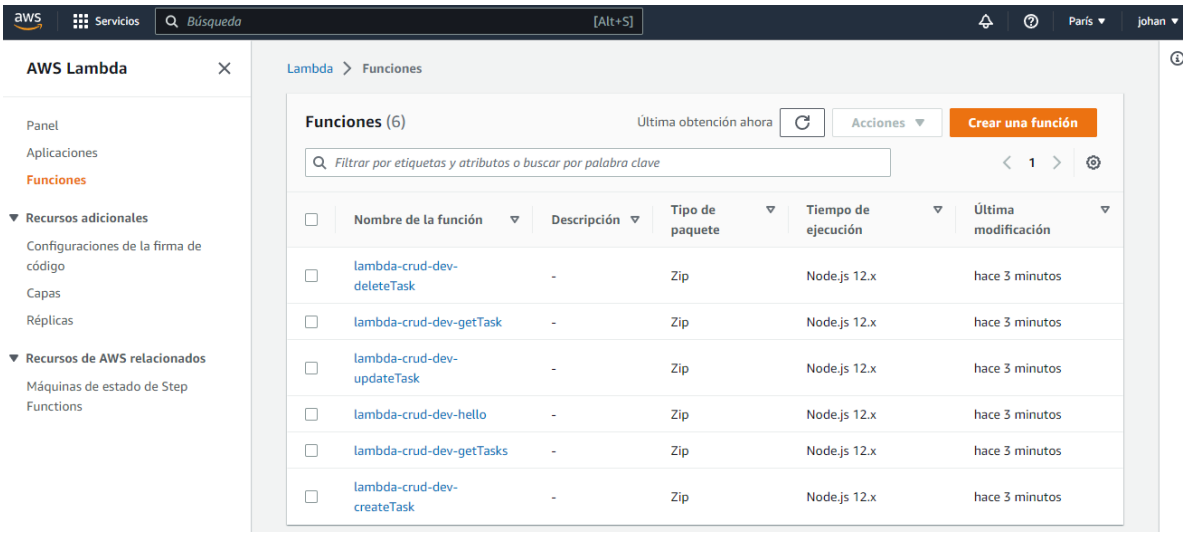

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
getTask.js - serverless - Visual Studio Code
addTask.js
deleteTask.js
getTasks.js
lambda-crud > getTasks.js > getTasks > result > TableName
const AWS = require("aws-sdk");
const getTasks = async (event) => {
const dynamodb = new AWS.DynamoDB.DocumentClient();
const result = await dynamodb.scan({ TableName: "LambdaTable" }
const tasks = result.Items;
return {
status: 200,
body: {
tasks,
},
};
};
module.exports = {
getTasks,
};
serverless.yml
deleteTask.js
getTask.js
lambda-crud > getTask.js > getTask
const AWS = require("aws-sdk");
const getTask = async (event) => {
const dynamodb = new AWS.DynamoDB.DocumentClient();
const { id } = event.pathParameters;
const result = await dynamodb
.get({
TableName: "LambdaTable",
Key: { id },
})
.promise();
const task = result.Item;
return {
status: 200,
body: task,
};
};
module.exports = {
getTask,
};


Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
addTask.js
getTasks.js
updateTask.js
lambda-crud > updateTask.js > ...
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: "LambdaTable",
Key: { id },
UpdateExpression: "set done = :done",
ExpressionAttributeValues: {
":done": done,
},
ReturnValues: "ALL_NEW",
})
.promise();
return {
statusCode: 200,
body: JSON.stringify({
message: "task updated",
}),
};

Finalmente volvemos a ejecutar el serverless para subir los cambios y con esto tendríamos el crud completo para las peticiones http



Resultados finales



Bibliografía

https://www.youtube.com/watch?v=da6E0R2grfo

https://github.com/hashicorp/terraform-provider-aws/issues/14873

https://repost.aws/questions/QU_m8virJwQKacrrt6YZa-Pg/access-denied-to-ssm-parameter-but-policy-in-place

https://www.youtube.com/watch?v=SkbcSbF0uq4&t=938s

https://www.youtube.com/watch?v=wvux4WOU5dc