

# Crud Api Rest y MicroServicio

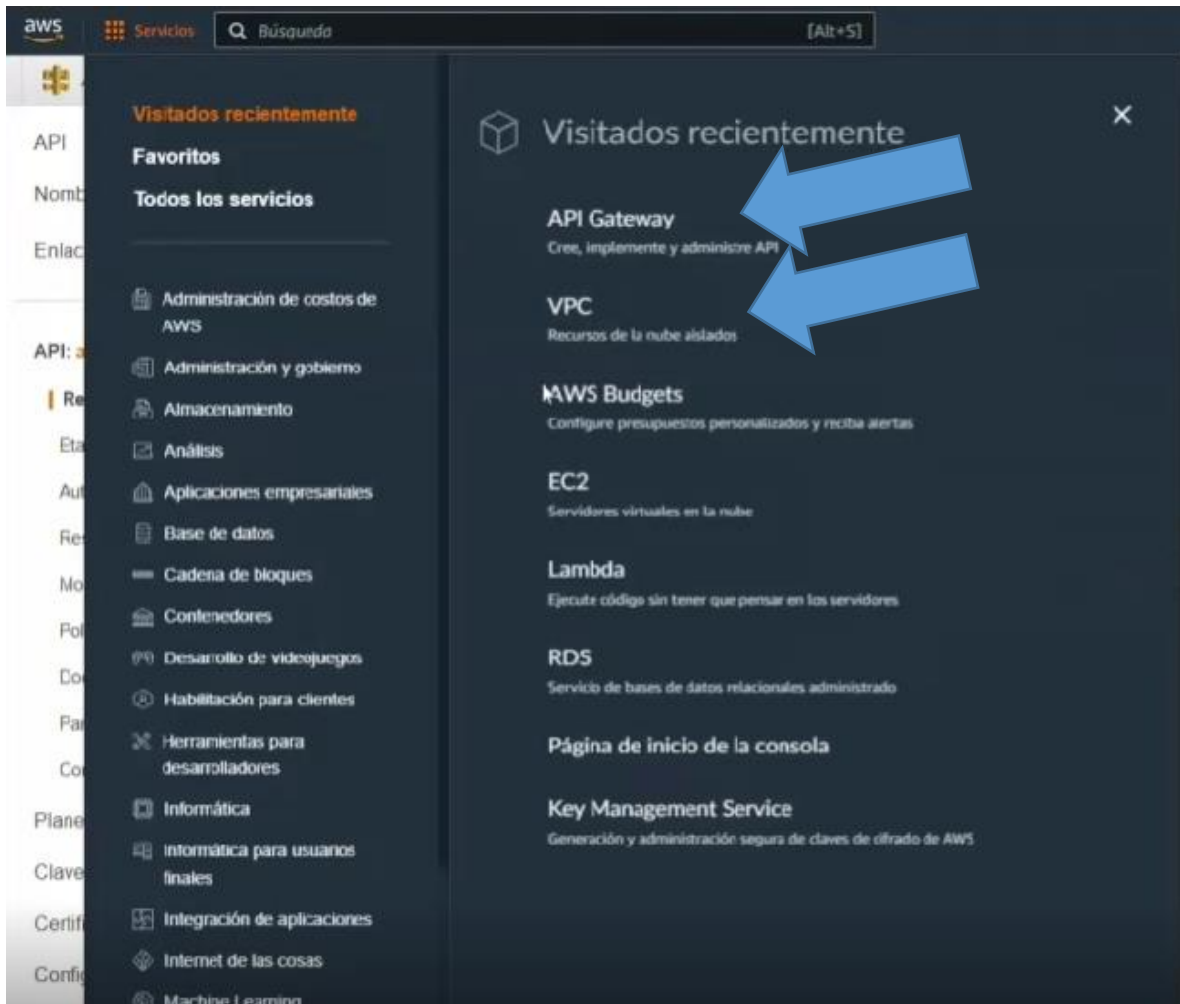
Manual Desarrollado por:

Heriberto Cabas Rocha  
Javier Andres Alba Tellez  
Jonathan Diaz Ramirez  
Libardo Andres Lopez Cala  
Norberto Enrique Cardenas Gomez

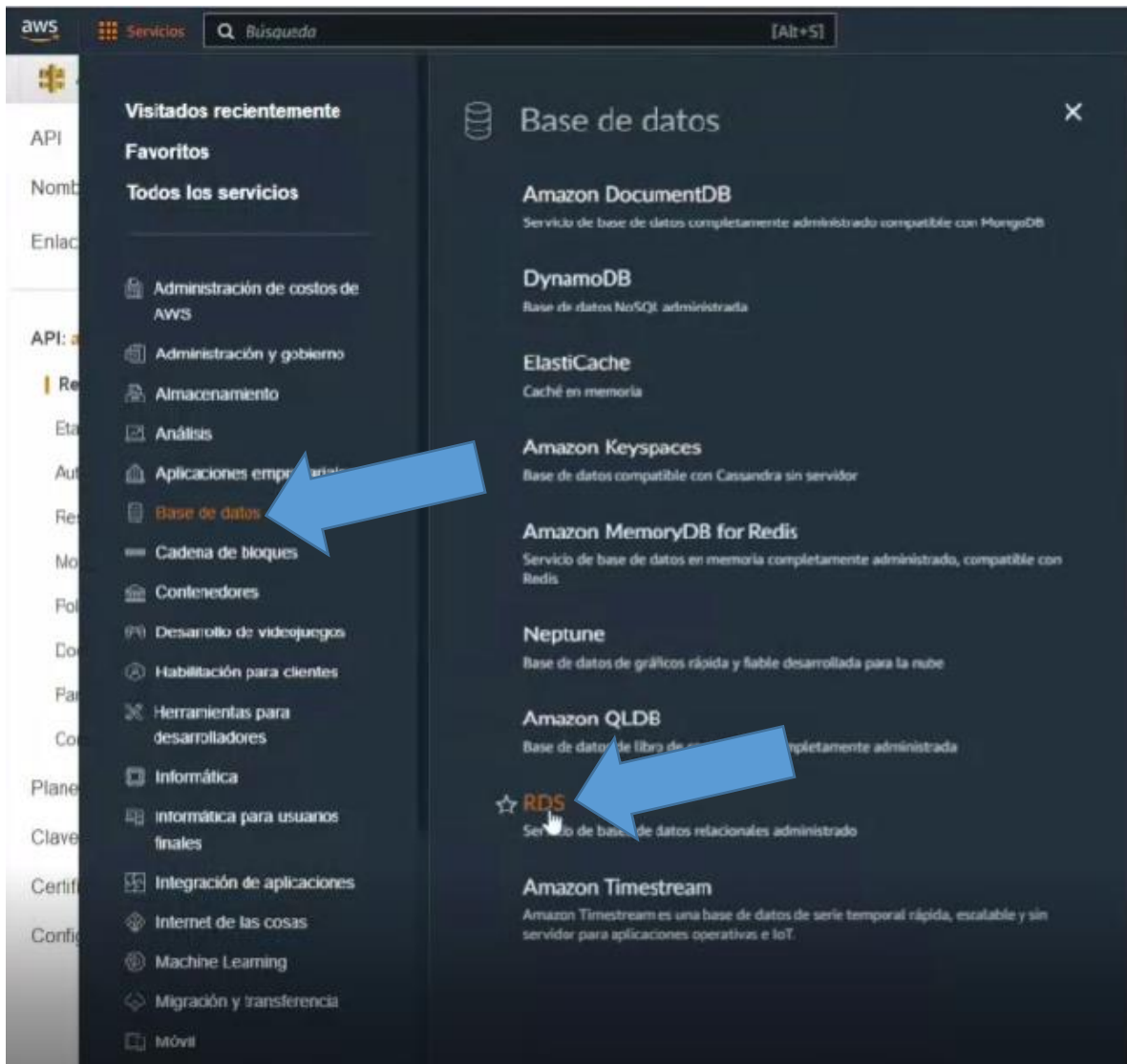
**INGENIERIA DE SISTEMAS**  
**2022**



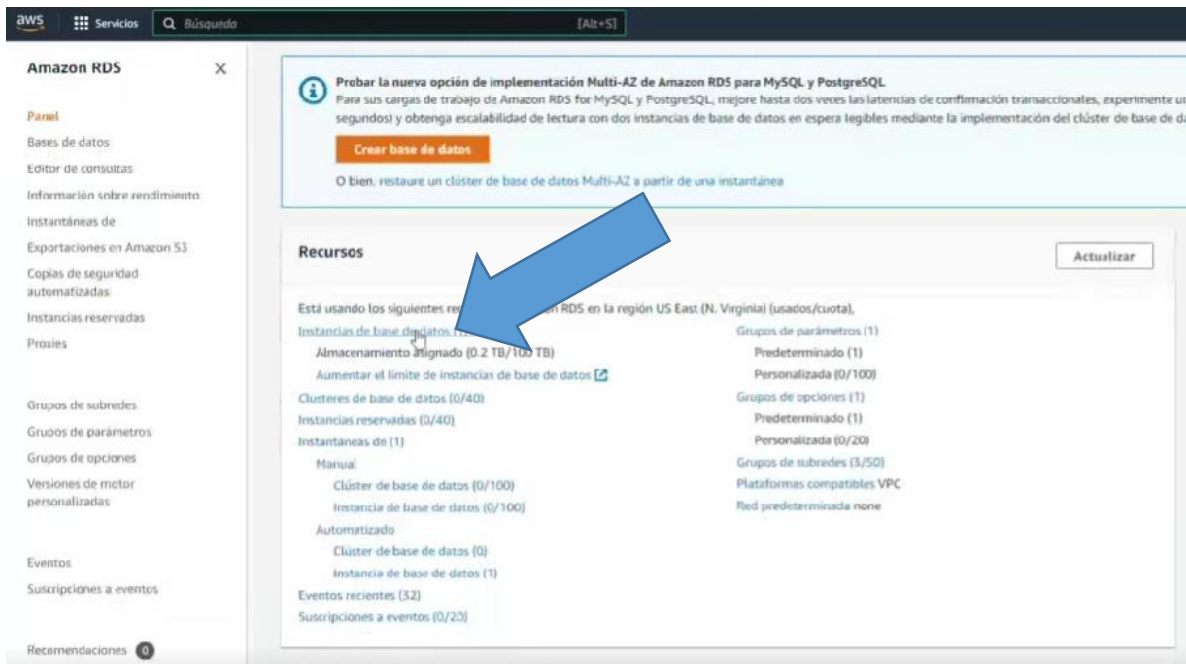
En este proyecto usaremos AWS(Amazon Web Services), en el cual usaremos dos servicios, API Gateway y VPC(Virtual Private Cloud)



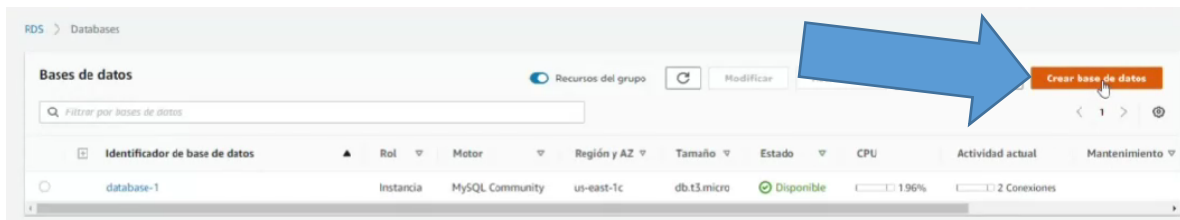
En el caso de la base de datos se usará RDS, servicio de base de datos relacionales administrado



Debemos dar clic en RCD para crear instancias de base de datos



A continuación, dar clic en Crear base de datos



Debemos seleccionar las siguientes opciones, que el motor de base de datos sea PostgreSQL y la versión sea 13.7-R1

The screenshot shows the AWS Management Console interface for creating a new RDS database. The breadcrumb navigation at the top indicates 'RDS > Create database'. The main heading is 'Crear base de datos'. Below this, there is a section titled 'Elegir un método de creación de base de datos' with two options: 'Creación estándar' (selected) and 'Creación sencilla'. The 'Creación estándar' option is highlighted with a blue border and a blue arrow pointing to it. Below this is the 'Opciones del motor' section, which includes a 'Tipo de motor' dropdown set to 'Información'. There are six database engine options displayed in a grid: Amazon Aurora, MySQL, MariaDB, PostgreSQL (selected), Oracle, and Microsoft SQL Server. The 'PostgreSQL' option is highlighted with a blue border and a blue arrow pointing to it. At the bottom, there is a 'Versión' dropdown menu set to 'PostgreSQL 13.7-R1', which is also highlighted with a blue arrow pointing to it.

aws Servicios Búsqueda [Alt+S]

RDS > Create database

## Crear base de datos


**Elegir un método de creación de base de datos** [Información](#)


☒ **Creación estándar**  
Puede definir todas las opciones de configuración, incluidas las de disponibilidad, seguridad, copias de seguridad y mantenimiento.


☐ **Creación sencilla**  
Utilice las configuraciones recomendadas. Algunas opciones de configuración se pueden cambiar después de crear la base de datos.


**Opciones del motor**


Tipo de motor [Información](#)


☐ Amazon Aurora  


☐ MySQL  


☐ MariaDB  


☒ PostgreSQL  


☐ Oracle  


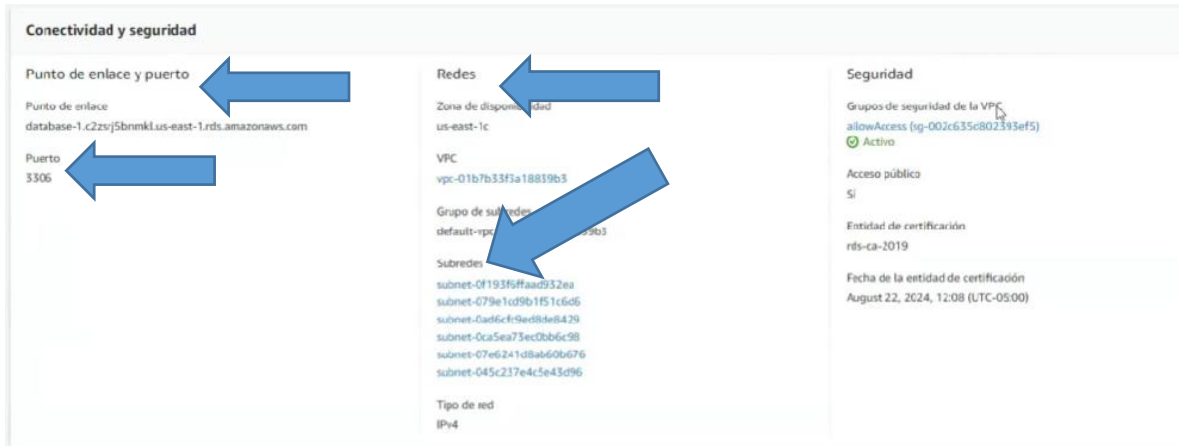
☐ Microsoft SQL Server  


**Versión**

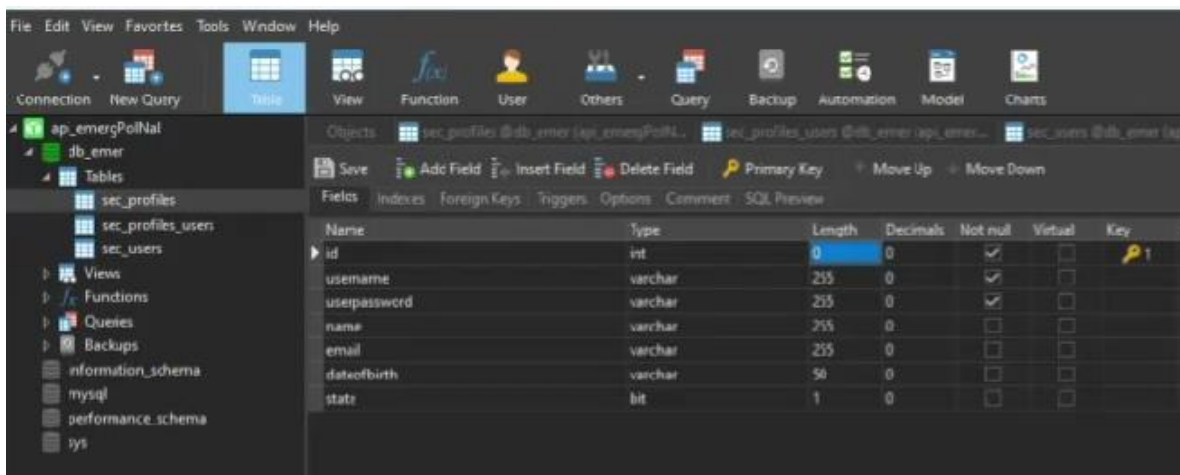
PostgreSQL 13.7-R1

Además, la conectividad debe contar con sus respectivas redes y subredes, en el punto de enlace encontraremos el link con el que nos podremos comunicar al proyecto, junto con su respectivo puerto de conexión.

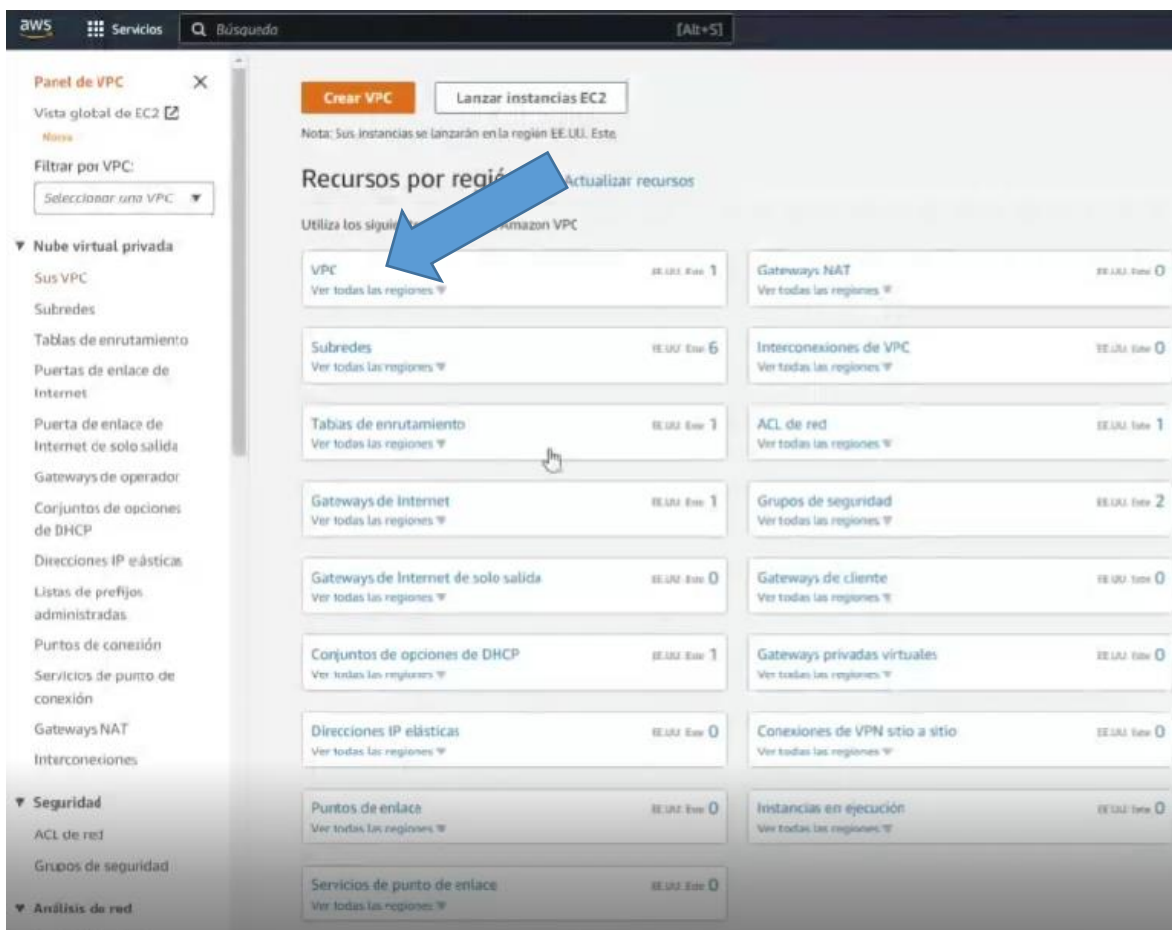
En conclusión, se implementó la instancia de la base de datos en el servidor



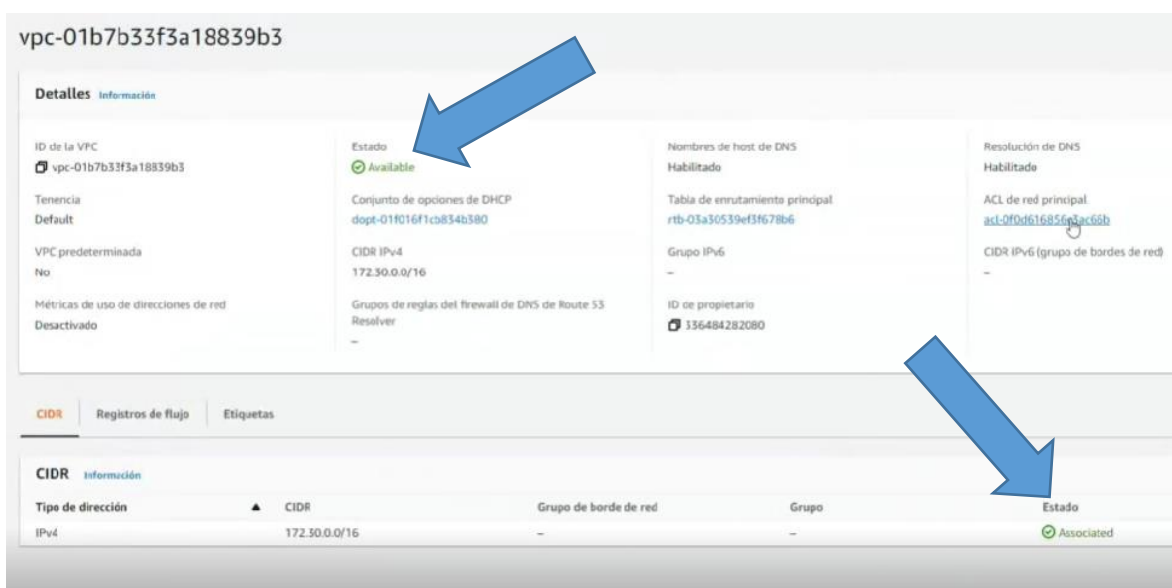
Una vez montada la instancia de la base de datos, se hace la conexión, en este caso con Navicat el cual es nuestro motor de base de datos



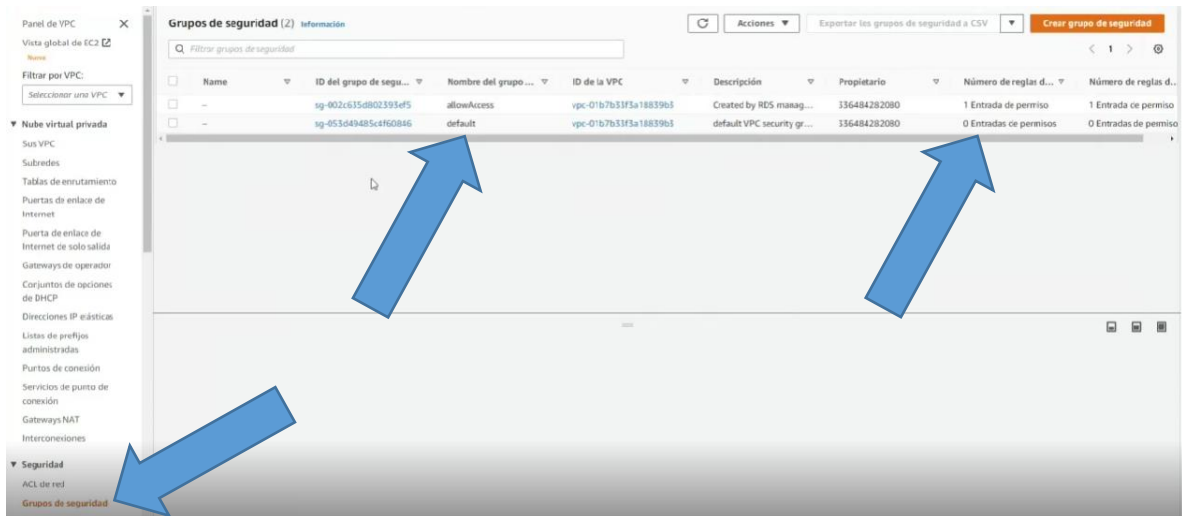
A continuación, debemos configurar el VPC para tener conexión con nuestro motor de base de datos



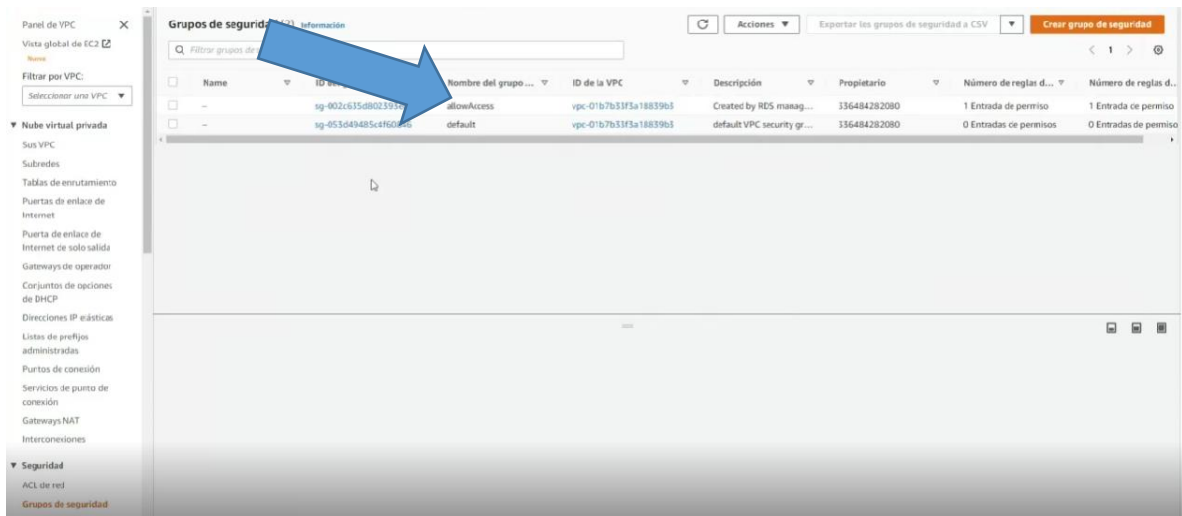
Se deberá configurar las siguientes opciones, además al finalizar el CIDR debe aparecer como Associated y el estado del VPC como Available



En el grupo de seguridad aparecerá el que se crea por default, si se puede observar aparece 0 entradas de permisos ya que no lo estamos usando, además sirve para las conexiones entrantes y salientes



En este proyecto se usará la que llamamos allowAccess, en la cual se dejara libre para todas las personas, la cual todos se podrán conectar sin problema alguno





sg-002c635d802393ef5 - allowAccess

Acciones

**Detalles**

Nombre del grupo de seguridad allowAccess	ID del grupo de seguridad sg-002c635d802393ef5	Descripción Created by RDS management console	ID de la VPC vpc-01b7b33f3a18639b3
Propietario 336484282090	Número de reglas de entrada 1 Entrada de permiso	Número de reglas de salida 1 Entrada de permiso	

Reglas de entrada Reglas de salida Etiquetas

Ahora puede comprobar la conectividad de red con Reachability Analyzer [Ejecutar Reachability Analyzer](#)

**Reglas de salida (1/1)**

Filtrar reglas de grupo de seguridad

	Name	ID de la regla del g...	Versión de IP	Tipo	Protocolo	Intervalo de puertos	Destino	Descripción
<input checked="" type="checkbox"/>	-	sg-01b1734d91a76cd...	IPv4	Todo el tráfico	Todo	Todo	0.0.0.0/0	-

A continuación, usaremos Lambda, el cual es un gestor de funciones que se encarga de crear las funciones internas en la nube, variando el lenguaje que se quiera usar

**AWS Lambda**

Panel  
Aplicaciones  
Funciones

▼ Recursos adicionales  
Configuraciones de la firma de código  
Capas  
Réplicas

▼ Recursos de AWS relacionados  
Máquinas de estado de Step Functions

**Recursos para EE.UU. Este (Norte de Virginia)**

[Crear una función](#)

Función(es) de Lambda: 3

Almacenamiento de código: 329,5 kB (D % de 75,0 GB)

Simultaneidad de cuenta completa: 10

Simultaneidad de cuenta no reservada: 10

**Métricas de nivel de cuenta**

Los siguientes gráficos muestran las métricas de todas las Funciones de Lambda en esta región de AWS.

1h 3h 12h 1d 3d 1sem. Personalizado

**Error count and success rate (%)**

Count Sin unidad

18:00 19:00 20:00

Errors Success rate (%)

**Throttles**

Count

18:00 19:00 20:00

Throttles

**Invocations**

Count

18:00 19:00 20:00

Invocations

**Duration**

Milliseconds

10.3k 5.16k

**ConcurrentExecutions**

Count

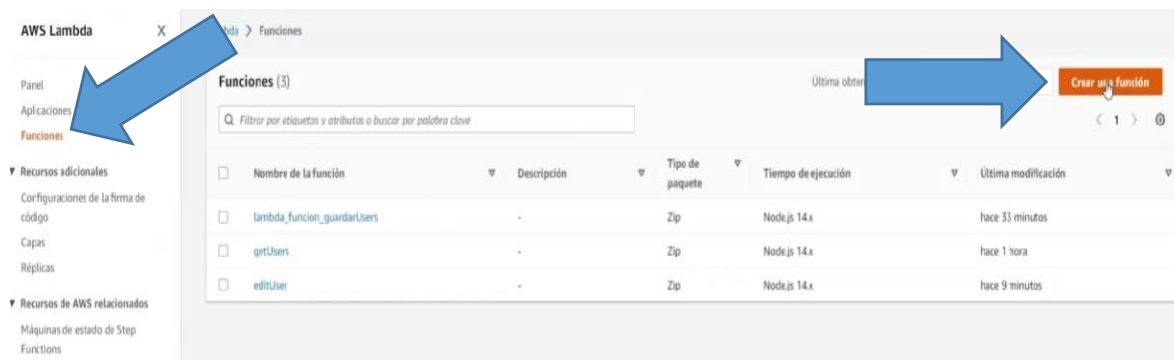
2 1

**UnreservedConcurrentExecutions**

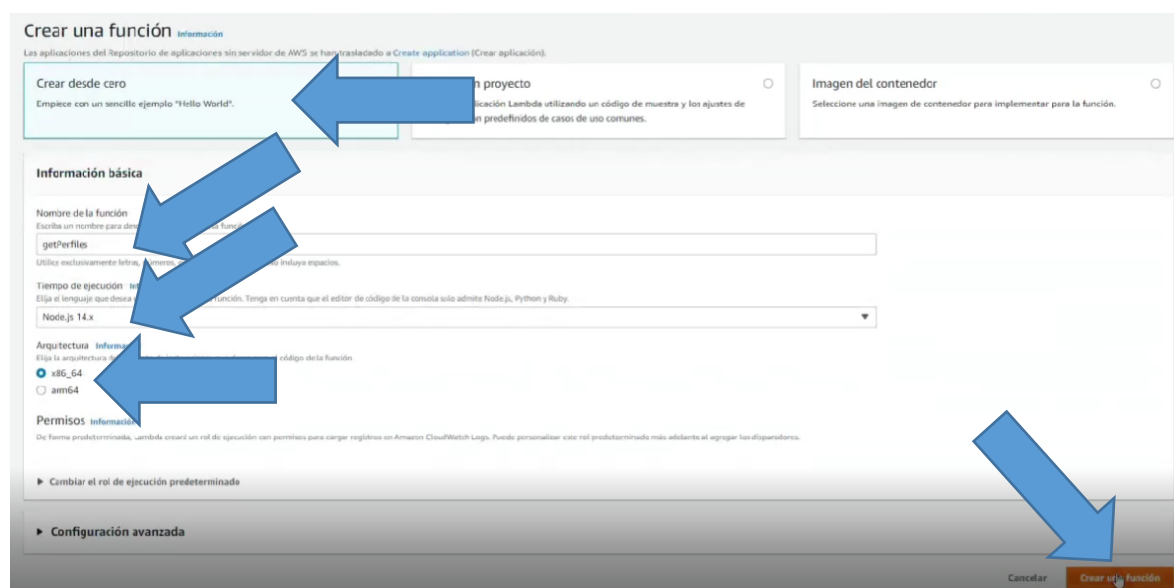
Count

2 1

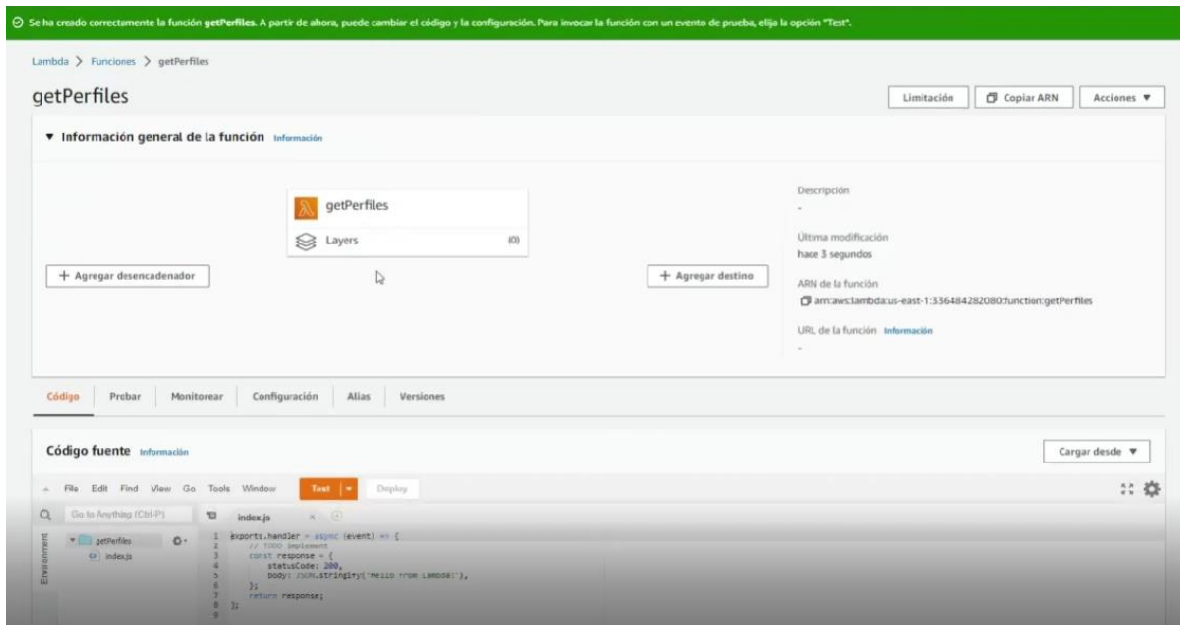
Por ejemplo, damos clic en Funciones y luego en crear una función



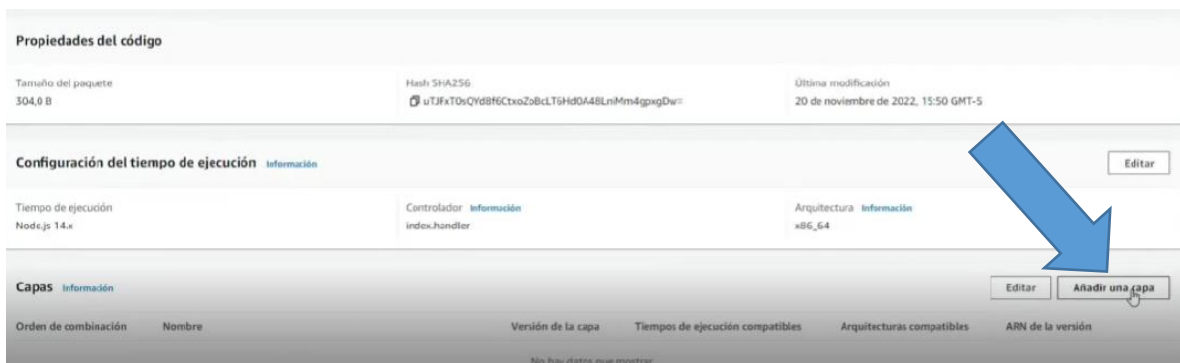
Seleccionamos las siguientes opciones, desde cero, en este caso el nombre de la función será “getPerfiles”, en tiempo de ejecución usaremos “Node js 14x”, la arquitectura que se usara es “x86\_64” y para finalizar, clic en crear función.



Recordar que estamos usando MySQL, por lo tanto, debemos instarle MySQL a la nueva función creada



Debemos dar clic en Añadir una capa



Luego seleccionar “Capas personalizadas”, en Capas personalizadas aparecerá un desplegable para importar la librería de MySQL, junto con su versión “1”, para finalizar daremos clic en Agregar.

## Agregar capa

### Configuración del tiempo de ejecución de la función

Tiempo de ejecución Node.js 14.x	Arquitectura x86_64
-------------------------------------	------------------------

### Elija una capa

**Fuente de capa** [Información](#)  
Elija entre capas con un tiempo de ejecución de Amazon (ARN) de una versión de capa. También puede crear una nueva capa.

☐ Capas de AWS  
Elija una capa de una lista de capas proporcionadas por AWS.

☒ Capas personalizadas  
Elija una capa de una lista de capas creadas por la cuenta u organización de AWS.

☐ Especificar un ARN  
Proporcione el ARN y especifique una capa.

### Capas personalizadas

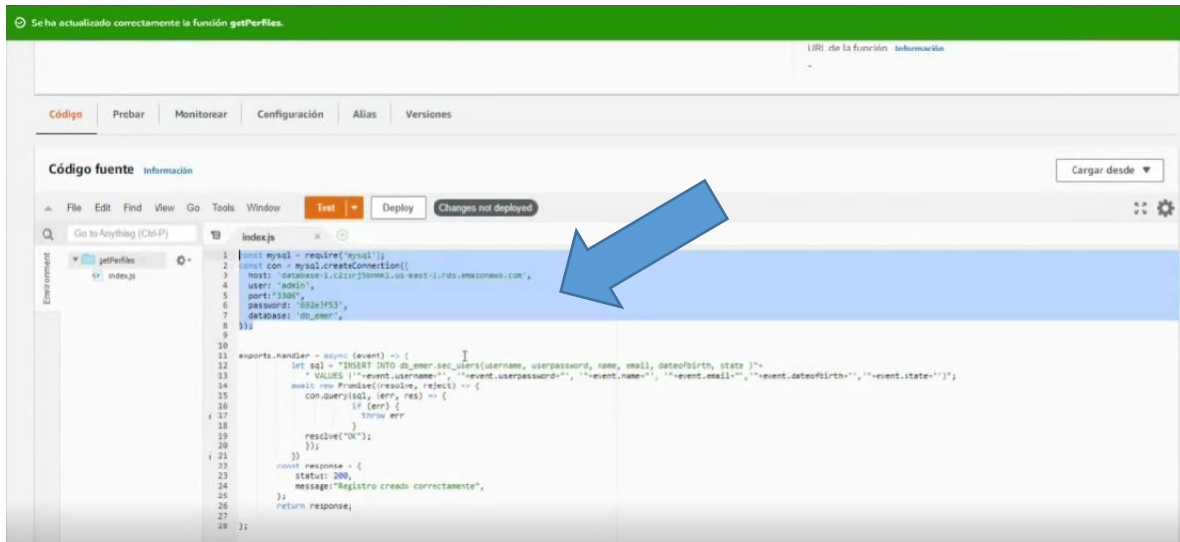
Capas creadas por la cuenta u organización de AWS que son compatibles con el tiempo de ejecución de la función

lib-common-mysql ▼

Versión  
1 ▼

Cancelar Agregar

La parte resalta en azul, nos permitira la conexion con MySQL



Se ha actualizado correctamente la función `getPerfiles`.

[URL de la función](#) `informacion`

[Código](#) [Prebar](#) [Monitorear](#) [Configuración](#) [Alias](#) [Versiones](#)

**Código fuente** [Información](#) Cargar desde ▾

File Edit View Go Tools Window **Test** Changes not deployed

Go to Anything (Ctrl-F)

`index.js`

```
1 const mysql = require('mysql');
2 const con = mysql.createConnection({
3   host: 'database-1.c2irj3bnnk1.us-east-1.rds.amazonaws.com',
4   user: 'admin',
5   port: '3306',
6   password: 'G2p2Y53',
7   database: 'db_emer'
8 });
9
10 exports.handler = async (event) => {
11   let sql = "INSERT INTO db_emer_sec_users(username, userpassword, name, email, dateofbirth, state) "
12     + "VALUES ('" + event.username + "', '" + event.userpassword + "', '" + event.name + "', '" + event.email + "', '" + event.dateofbirth + "', '" + event.state + "')";
13   await new Promise((resolve, reject) => {
14     con.query(sql, (err, res) => {
15       if (err) {
16         throw err
17       }
18       resolve("OK");
19     });
20   });
21   const response = {
22     status: 200,
23     message: "Registro creado correctamente",
24   };
25   return response;
26 };
27
28 ;
```

Como podemos observar, estamos usando “index\_handler”, el cual es el controlador de Node js, lo que nos permite en AWS recibir la información para su utilización

Propiedades del código

Tamaño del paquete

304,0 B

Hash SHA256

uTJfXt0xQY08f6CxoZoBcLT6hd0A48LnM4gpgDw=

Última modificación

20 de noviembre de 2022, 15:50 GMT-5

Configuración del tiempo de ejecución

Información

Editar

Tiempo de ejecución

Node.js 14.x

Controlador

index.handler

Información

Arquitectura

x86\_64

Información

Capas

Información

Editar

Añadir una capa

Orden de combinación

Nombre

Versión de la capa

Tiempos de ejecución compatibles

Arquitecturas compatibles

ARN de la versión

1

lib-common-mysql

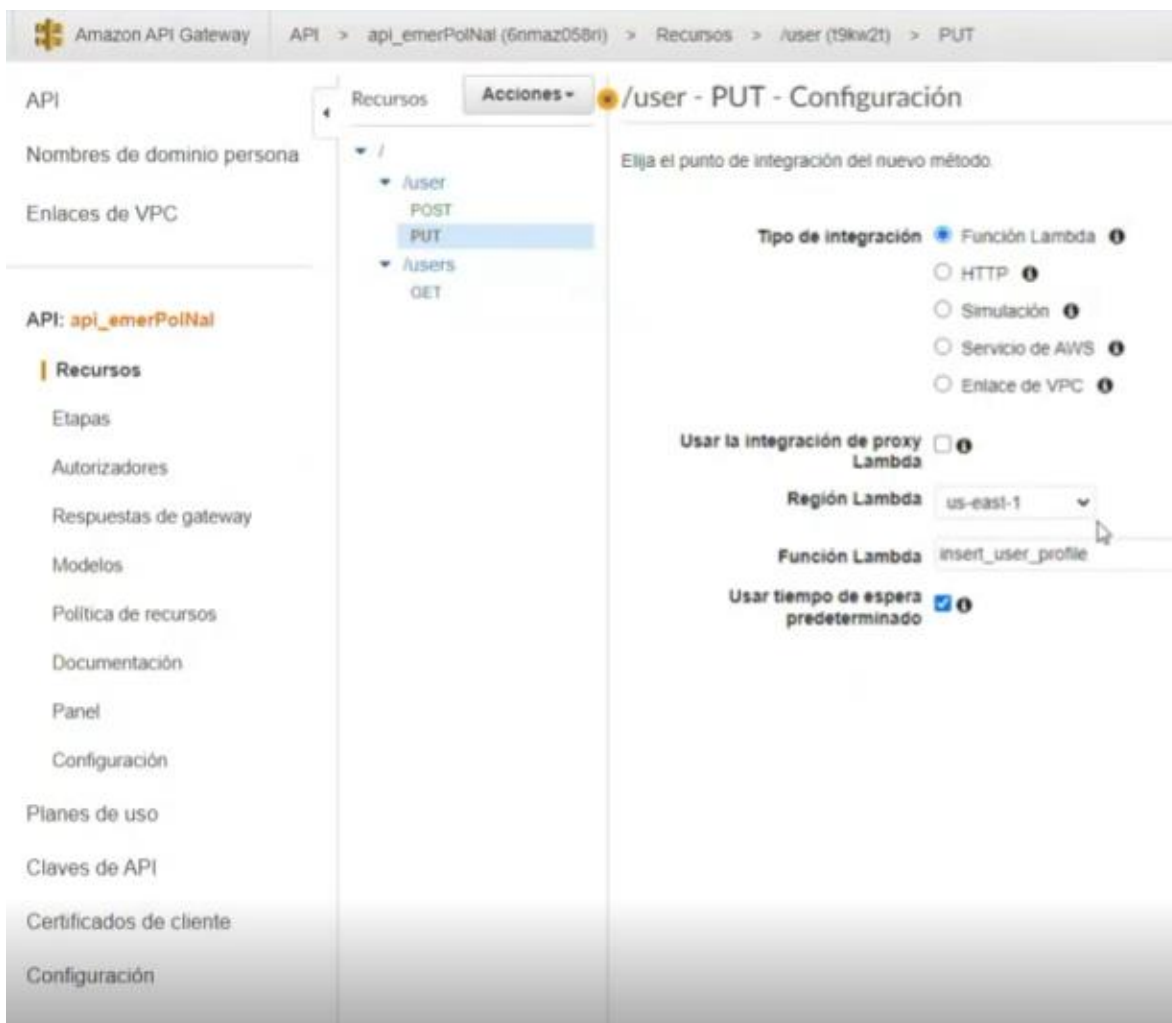
1

nodejs18.x, nodejs12.x, nodejs14.x, nodejs16.x

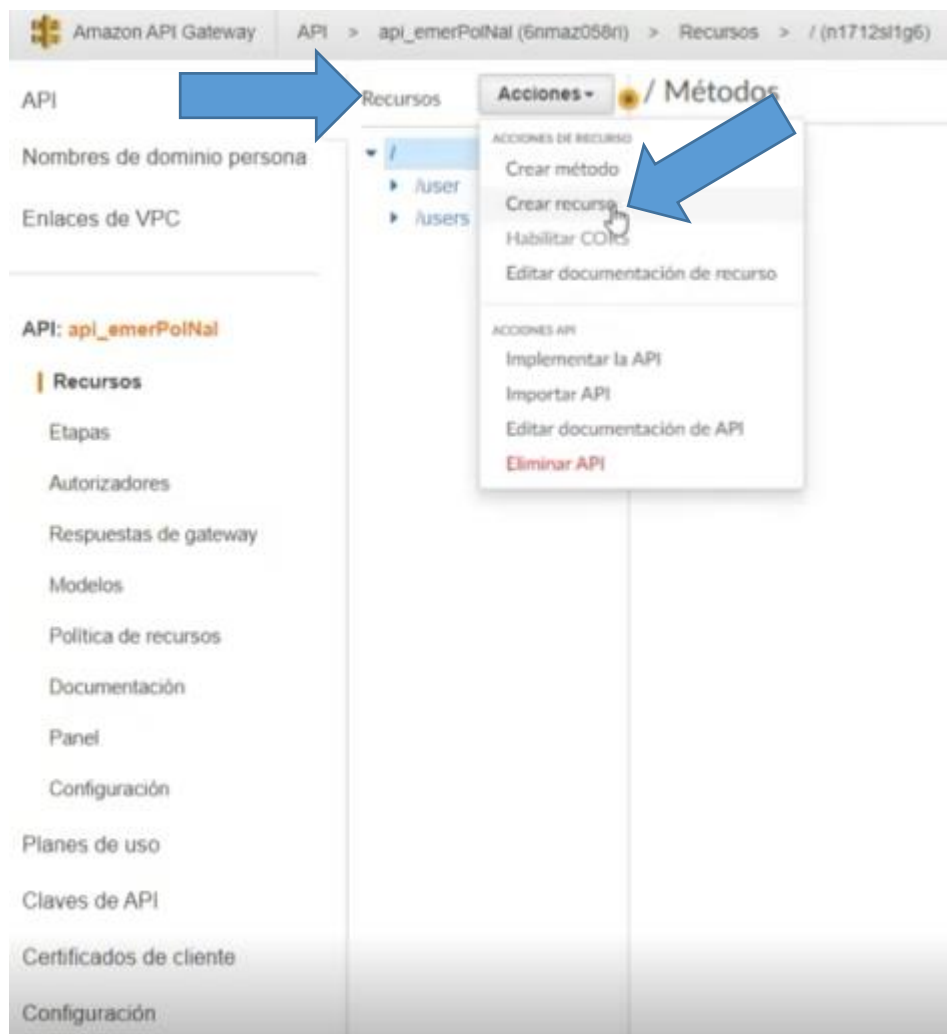
-

arn:aws:lambda:us-east-1:336484282080:layer:lib-common-mysql:1

Una vez creada la función de Lambda, debemos usar nuestro API Gateway



A continuación, debemos crear un recurso, damos clic “Acciones” y Crear recurso



En este caso, el nombre de nuestro recurso será “profiles” y clic en crear recurso



El recurso “profiles”, va a tener el método “GET”, el tipo de integración será “Función Lambda” y en Función Lambda ejecutaremos “getProfiles” y clic en guardar

Elige el punto de integración del nuevo método.

Tipo de integración: ☒ Función Lambda

☐ HTTP

☐ Simulación

☐ Servicio de AWS

☐ Enlace de VPC

Usar la integración de proxy Lambda: ☐

Región Lambda: us-east-1

Función Lambda: getProfiles

Usar tiempo de espera predeterminado: ☒

guardar

Saldrá un cuadro, en el cual le concedemos permiso al API Gateway dando clic en Aceptar

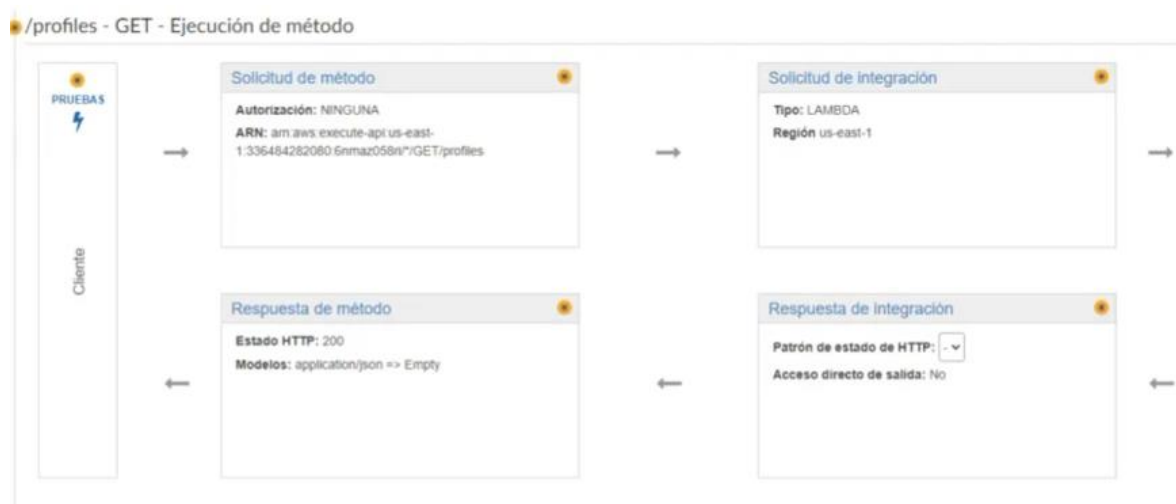
Agregar permiso a la función Lambda

Va a conceder permiso a API Gateway para invocar la función Lambda:  
arn:aws:lambda:us-east-1:336484282080:function:getProfiles

Cancelar Aceptar

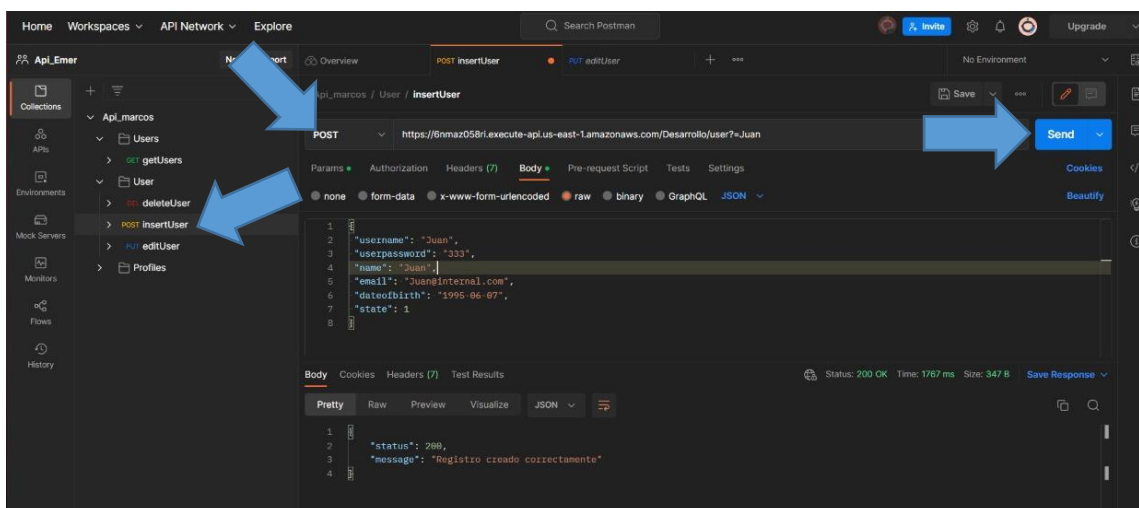


Con eso tendremos nuestro endpoint






A continuación, mostraremos el crear un nuevo usuario.

Utilizaremos Postman para hacer pruebas, lo cual debemos tener en cuenta que para crear un usuario debemos usar el método POST y para finalizar dar clic en "Send"

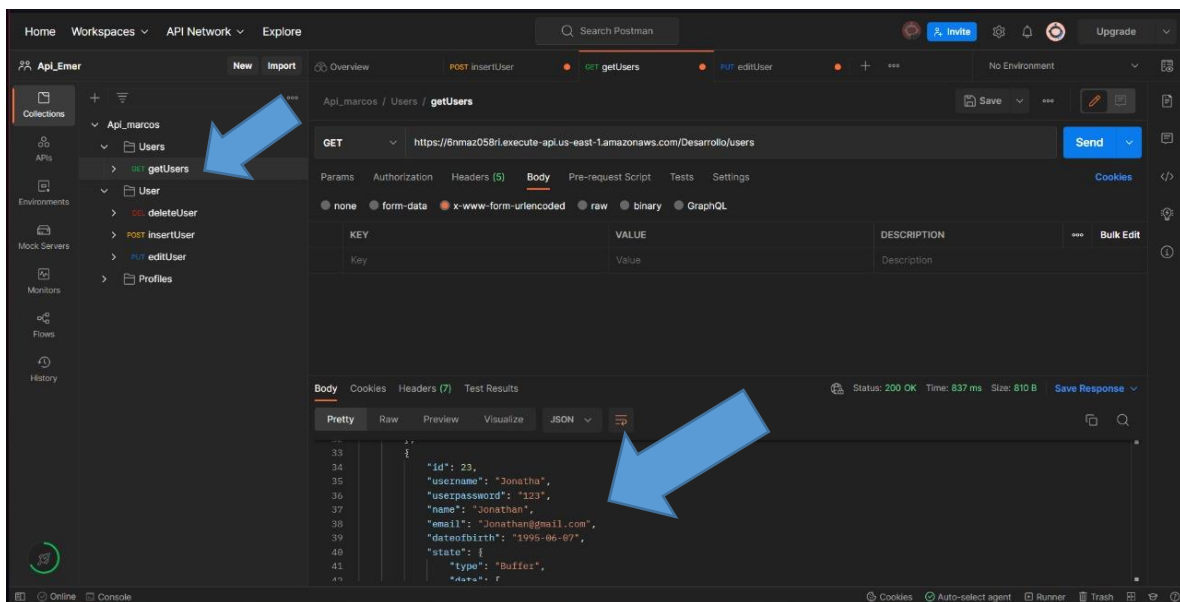


Como podemos observar, ya se puede visualizar el nuevo usuario recién creado, junto con sus dos botones de editar y eliminar



ID	Nombre	Correo	Acciones
21	Kike	Kike@internal.com	 
22	Juan	Juan@internal.com	 

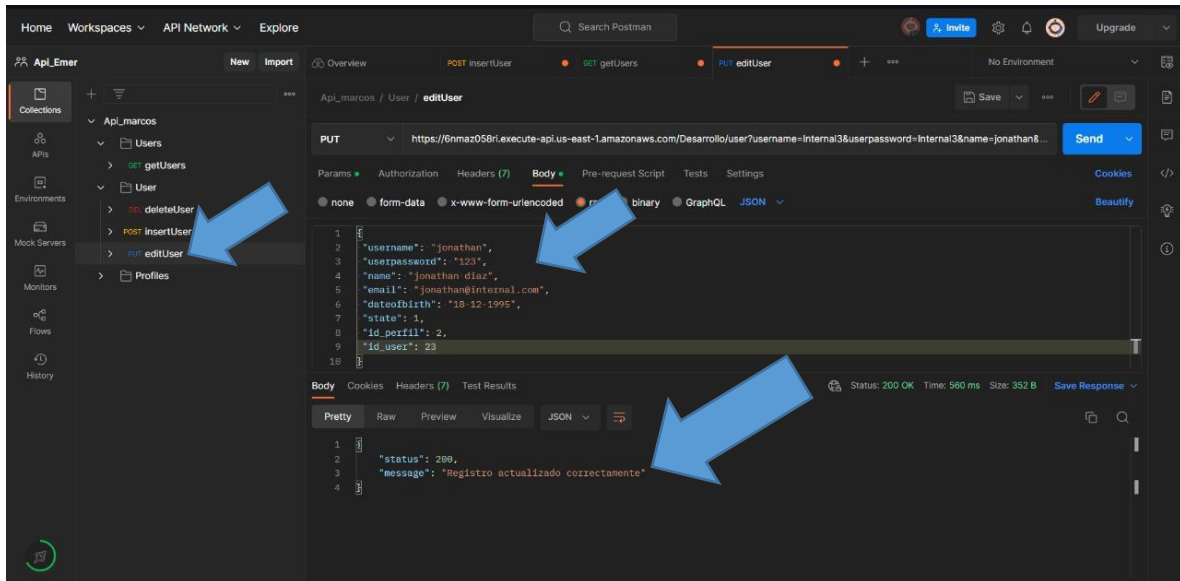
A continuación, crearemos un nuevo usuario para hacer la prueba de actualizar, en el cual el nombre será Jonathan



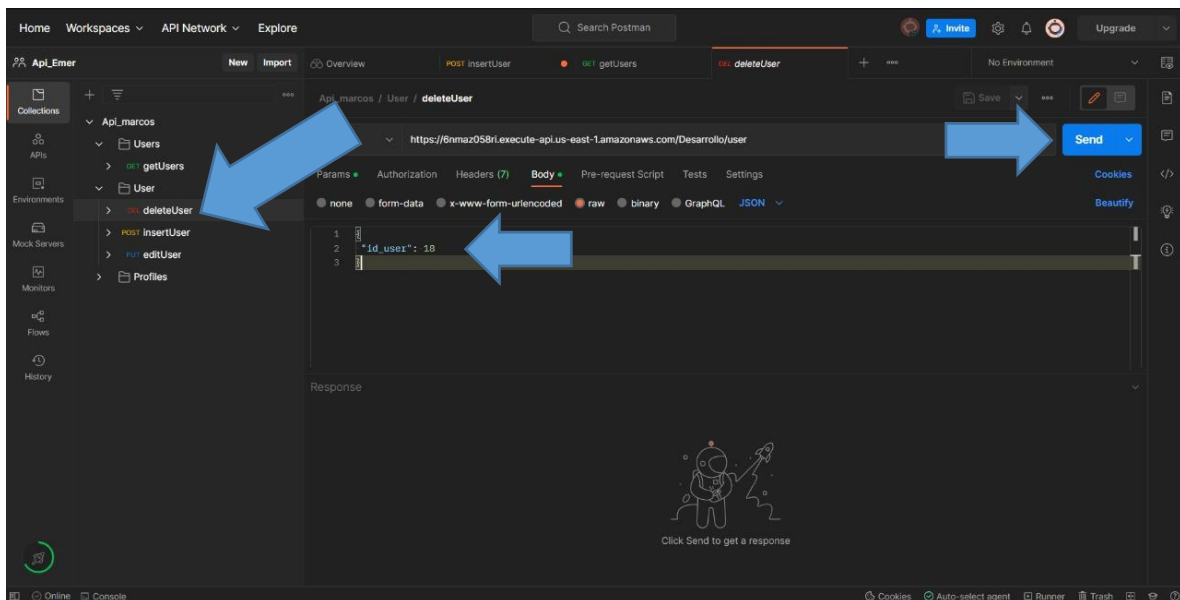
Ahora cambiaremos el nombre, y colocaremos “Jonathan Diaz”.

Recordar que para editar un registro debemos usar el método “PUT”

Como podemos observar, el registro fue actualizado correctamente.

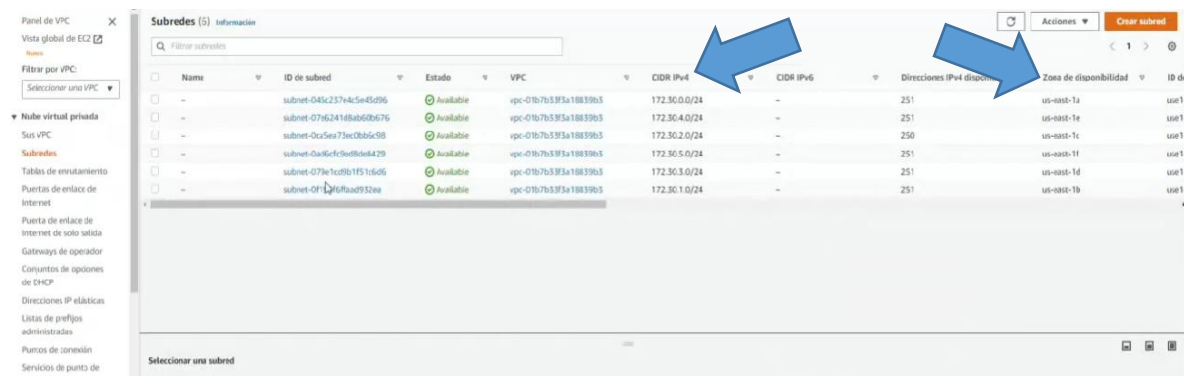


Ahora para eliminar el nuevo usuario ya creado, usamos el método “DEL”, junto con su código, y para finalizar, clic en el botón “Send”



El microservicio es el re direccionamiento de las peticiones en diferentes IP, lo cual se encarga de eso el VPC, en el cual se encuentran todas las subredes.

En CIDR Ipv4 podemos observar las conexiones que hace, además en cada dirección IP la zona de disponibilidad cambia, lo cual me permite que, si algún servidor de AWS se cae, el aplicativo seguirá funcionando, ya que hay otros 5 más disponibles, o si se caen 3 servidores, aún quedan 3 más disponibles.



	Name	ID de subred	Estado	VPC	CIDR IPv4	CIDR IPv6	Direcciones IPv4 disponibles	Zona de disponibilidad	ID de
<input type="checkbox"/>	-	subnet-043c2374dc5e43d96	available	vpc-07b7b33f3a18839b3	172.30.0.0/24	-	251	us-east-1a	us-east-1a
<input type="checkbox"/>	-	subnet-07b624188ab60b676	available	vpc-07b7b33f3a18839b3	172.30.4.0/24	-	251	us-east-1e	us-east-1e
<input type="checkbox"/>	-	subnet-0ca5ea73ec0bb5c98	available	vpc-07b7b33f3a18839b3	172.30.2.0/24	-	250	us-east-1c	us-east-1c
<input type="checkbox"/>	-	subnet-0a68c4c19d8d4e429	available	vpc-07b7b33f3a18839b3	172.30.5.0/24	-	251	us-east-1f	us-east-1f
<input type="checkbox"/>	-	subnet-073a1c0fb1f515d05	available	vpc-07b7b33f3a18839b3	172.30.3.0/24	-	251	us-east-1d	us-east-1d
<input type="checkbox"/>	-	subnet-0f16f8a2d35ea	available	vpc-07b7b33f3a18839b3	172.30.1.0/24	-	251	us-east-1b	us-east-1b

Y listo, ya tendríamos funcional nuestro **Tercer Parcial Marcos de Trabajo** y con esto daríamos fin a la creación del aplicativo. **Muchas gracias, hasta pronto.**