



ITESO

UNIVERSIDAD JESUITA DE GUADALAJARA

GAMING HUB

EQUIPO NÚMERO: 5

PRESENTADO POR: <LUIS ANGEL SANTANA HERNANDEZ, JUAN BERNARDO OROZCO QUIRARTE, LUIS DANIEL ARELLANO NÚÑEZ>

CORREO ELECTRÓNICO: <LUIS.SANTANA, BERNARDO.OROZCO, LUIS.ARELLANO>@ITESO.MX

PARA EL CURSO DE: CLOUD COMPUTING (DEVELOPMENT)

IMPARTIDA POR: MTRO. OJEDA OROZCO MIGUEL ANGEL

FECHA: <31/29/04/2025>

NOMBRE DEL PROYECTO

Gaming Hub

MARCO TEÓRICO

Amazon, junto con su alojamiento de servidores AWS, ha creado Amazon Gamelift, un método de alojamiento de servidores de videojuegos. Gamelift leyendo un poco son instancias de EC2 y EC2 Graviton, con las cuales levantan servidores de videojuegos [1].

A partir de esto surge la pregunta: ¿podemos hacer un servicio similar a este?

Durante este proyecto, hemos decidido hacer un sistema el cual te ayude a alojar servidores e información de estos dentro de servicios de EC2, servicios de almacenamiento, etc. Debido a las necesidades del proyecto que se encuentran en el área de requisitos, decidimos usar las siguientes herramientas:

Amazon EC2: Sera la base para el despliegue y la administración de servidores de videojuegos en la nube, con la ayuda de EC2 podremos ejecutar instancias virtuales con la capacidad de cómputo necesaria para poder correr juegos con baja latencia y buen rendimiento, además EC2 nos brinda flexibilidad en la selección de los tipos de instancia, cosa que nos va a permitir optimizar los costos y el rendimiento dependiendo de la demanda del juego, y bueno para esta propuesta. Para el Demo incluso con un tipo de instancia sin muchos recursos podemos testear el funcionamiento de nuestro servidor [2].

Amazon S3: Para el almacenamiento de archivos y datos que se relacionen con nuestros servidores. S3 nos permite guardar datos, en este caso siendo más específicos los tipos de archivos y datos que podemos guardar son configuraciones de juegos, logs, backups de servidores, entre otras cosas. Su alta durabilidad y también escalabilidad nos garantizan que la información esté disponible 24/7 sin comprometer el rendimiento de nuestro sistema [3].

Amazon CloudWatch: Para monitoreo y generación de alertas sobre el estado de los servidores, CloudWatch es una herramienta estrechamente relacionada con el pilar de Excelencia Operativa, nos permitirá monitorear métricas como el uso del CPU, memoria y tráfico de red a las instancias, automatizar acciones en caso de fallas [4].

Amazon Cloud Front: La latencia en la comunicación es un factor de enorme importancia, Cloud Front nos permitirá optimizar la distribución de contenido, como lo pueden ser actualizaciones del juego, datos de configuración, animaciones como videos, asegurando que los jugadores puedan acceder de forma rápida a los recursos desde diferentes regiones [5].

Amazon DYNAMO: Lo utilizaremos para gestionar la base de datos, lo que nos permitira almacenar y administrar información de las cuentas de cada usuario [6].

AWS Lambda: Para automatizar tareas dentro de nuestro sistema, Lambda nos permitirá ejecutar algunas funciones sin necesidad de mantener servidores adicionales, cosa que nos reducirá costos y facilitará la implementación de procesos como el reinicio automático de servidores y procesamiento de logs y métricas [7].

API Gateway: Nos permitirá exponer una API REST segura y escalable, por medio de esta API los usuarios podrán interactuar con los servidores de manera sencilla y eficiente sin preocuparse por el acceso directo a la consola de AWS o configuraciones [8].

AWS IAM: La seguridad en la nube es un aspecto fundamental, un pilar, con IAM definiremos roles y permisos específicos para controlar el acceso a los servicios, de esta forma podremos garantizar que cada usuario tenga acceso a los recursos que le correspondan, aplicaremos el principio del menor privilegio para minimizar riesgos de seguridad [9].

Amazon VPC: Para garantizar un entorno de producción seguro, con Amazon VPC podemos gestionar dominios y resolver nombres de servidores de juegos que operen en una infraestructura separada de otros recursos públicos, nos permitirá configurar reglas de firewall para mejorar la seguridad y rendimiento [10].

Route 53: Para facilitar la conexión e los jugadores a sus servidores Route 53 nos permitirá gestionar dominios y resolver nombres de servidor, esto asegura que los usuarios puedan conectarse a sus servicios favoritos y servidores sin la necesidad de recordar direcciones IP directamente [11].

REQUERIMIENTOS

Principales

- Página web para que un usuario pueda generar y crear servidores de juegos (por el momento pensamos Minecraft, left for death, etc. [En si juegos con bajos recursos para no acabarnos los recursos de la plataforma educativa de AWS])
- El usuario puede crear máximo 3 servidores
- Panel para parar, eliminar, actualizar servicios en la nube

Secundarios

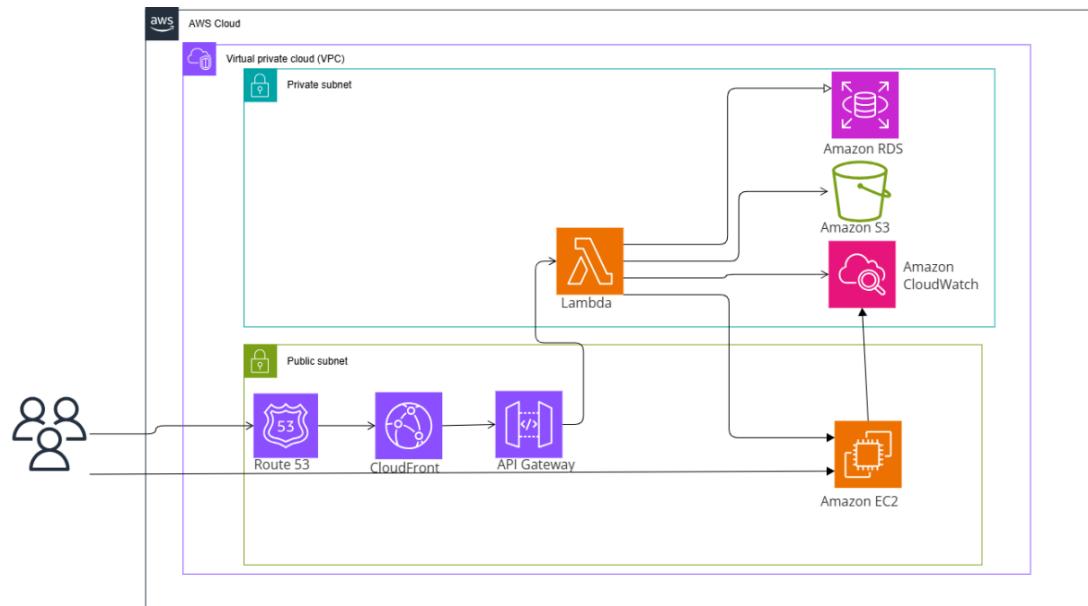
- Se pueden generar servidores públicos (visibles en la página web)
- Alguna forma de que el usuario pueda ver en la página web el porcentaje de uso de los recursos
- Validación de login del usuario

Opcionales u otros

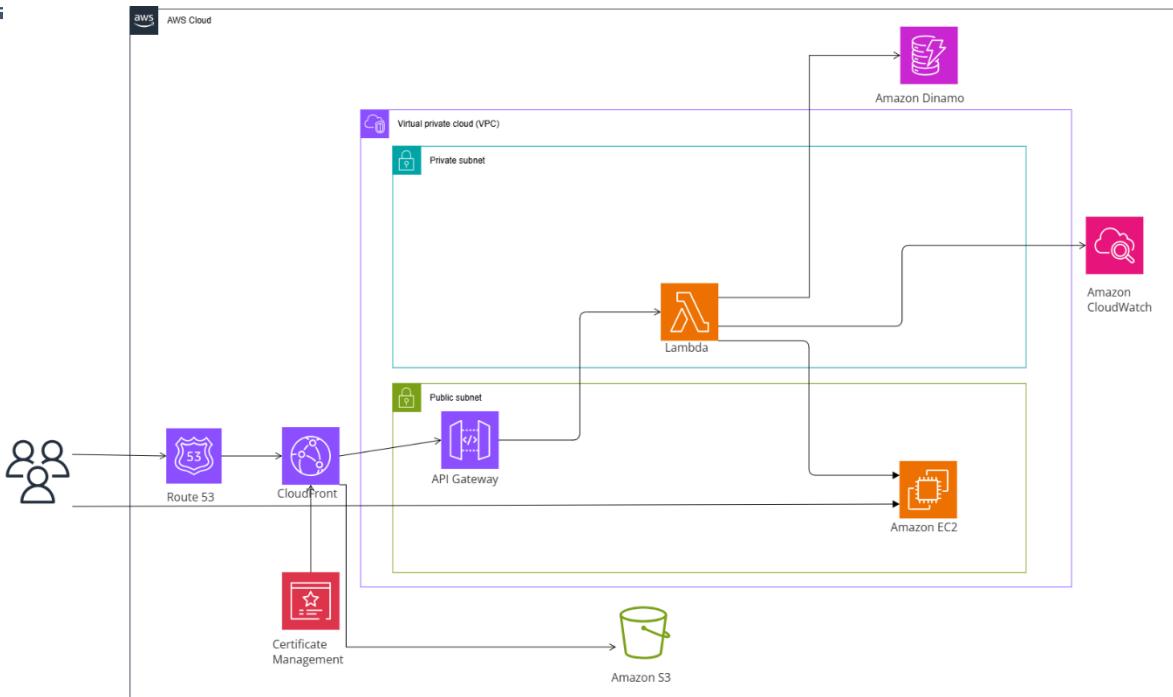
- Diseño de la página bonita
- Auto escalable

DIAGRAMA

ANTES:



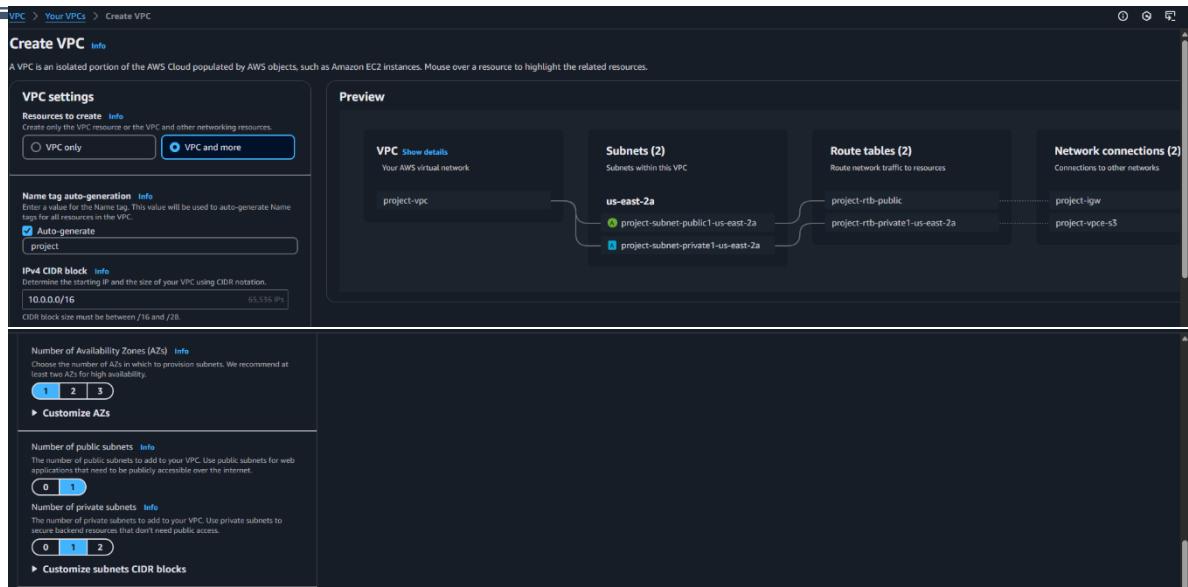
DESPUES:



DESARROLLO:

VPC:

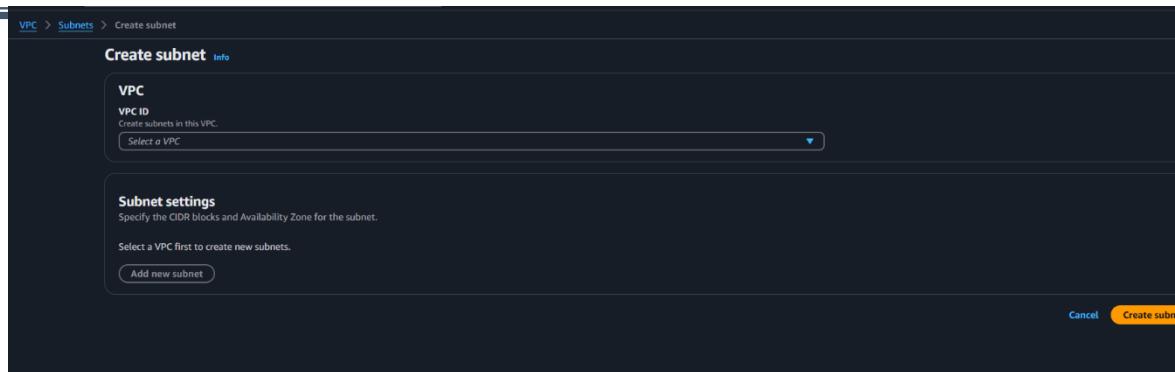
1. Desde la consola de AWS acceder al servicio VPC
2. Creamos un VPC, cambiando Availability Zones en 1 en vez de 2(default). Antes de continuar debo de explicar que debes de dejar 2, porque varias aplicaciones te avisan que necesitan 2 availability zones, entonces luego vamos a poner el segundo AZ



3. No le ponemos NAT Gateway



4. Una vez creada nos vamos a subnets y vamos a crear una subnet(COn esto ya deberamos cumplir con 2 AZ, que te piden aplicaciones como amazon RDS (Al final no lo usamos pero en su momento era necesario))

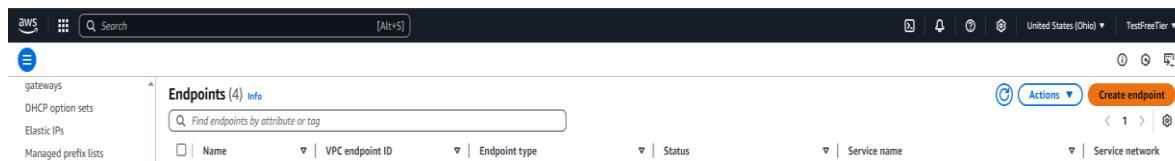


5. Añadir endpoints de conexión para comunicación entre subnets privadas y públicas

▼ **PrivateLink and Lattice**

- Getting started [Updated](#)
- Endpoints [Updated](#)**
- Endpoint services
- Service networks [Updated](#)
- Lattice services
- Resource configurations [New](#)
- Resource gateways [New](#)
- Target groups

6. Dar clic en crear endpoint



7. Agregar los siguientes endpoints 1 por servicio:

Servicio s3

- i) buscar el servicio: *com.amazonaws.us-east-2.s3* de tipo gateway



Service Name	Owner	Type	Service Region
com.amazonaws.us-east-2.s3	amazon	Gateway	us-east-2
com.amazonaws.us-east-2.s3	amazon	Interface	us-east-2

- ii) Añadir el VPC creado y en la IP table seleccionar la subnet privada

Network settings

Select the VPC in which to create the endpoint

VPC

Create the VPC endpoint in the VPC in the same AWS Region from which you will access a resource.

vpc-0d3f0273d571819b0 (Gaming-vpc)


Route tables (1/3) Info

Name	Route Table ID	Main	Associated Id
<input checked="" type="checkbox"/> Gaming-rtb-private1-us-east-2a	rtb-0e472142f51771ee3 (Gaming-rtb-...)	No	subnet-0a8af7c26cbe433c6 (Gaming-subnet-private1-us-east-2a)
<input type="checkbox"/> Gaming-rtb-public	rtb-0dca6a264018b8ac9 (Gaming-rtb-...)	No	subnet-02b8b5064491aec02 (Gaming-subnet-public1-us-east-2a)
<input type="checkbox"/> -	rtb-08677c55ca30ff4a5	Yes	subnet-03acbf4aa34d4d18f (gaming-Subnet-2)

iii) Creamos el endpoint
Tags

No tags associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#)
[Create endpoint](#)
DYNAMO DB
i) Buscar el servicio *com.amazonaws.us-east-2.dynamodb* de tipo Gateway
Services (1/2)

Service Name = com.amazonaws.us-east-2.dynamodb				Clear filters
Service Name	Owner	Type	Service Region	
<input checked="" type="radio"/> com.amazonaws.us-east-2.dynamodb	amazon	Gateway	us-east-2	
<input type="radio"/> com.amazonaws.us-east-2.dynamodb	amazon	Interface	us-east-2	

ii) Añadir VPC y en la IP table seleccionar la subnet privada
Network settings

Select the VPC in which to create the endpoint

VPC

Create the VPC endpoint in the VPC in the same AWS Region from which you will access a resource.

vpc-0d3f0273d571819b0 (Gaming-vpc)


Route tables (1/3) Info

Name	Route Table ID	Main	Associated Id
<input checked="" type="checkbox"/> Gaming-rtb-private1-us-east-2a	rtb-0e472142f51771ee3 (Gaming-rtb-...)	No	subnet-0a8af7c26cbe433c6 (Gaming-subnet-private1-us-east-2a)
<input type="checkbox"/> Gaming-rtb-public	rtb-0dca6a264018b8ac9 (Gaming-rtb-...)	No	subnet-02b8b5064491aec02 (Gaming-subnet-public1-us-east-2a)
<input type="checkbox"/> -	rtb-08677c55ca30ff4a5	Yes	subnet-03acbf4aa34d4d18f (gaming-Subnet-2)

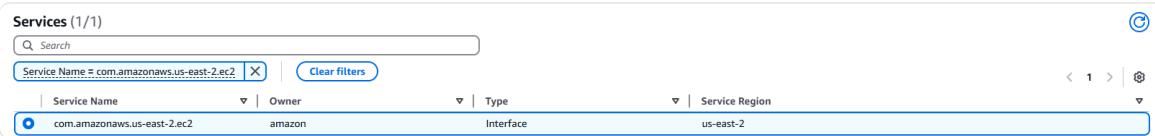
iii) Creamos el endpoint
Tags

No tags associated with the resource.

[Add new tag](#)

You can add 50 more tags.

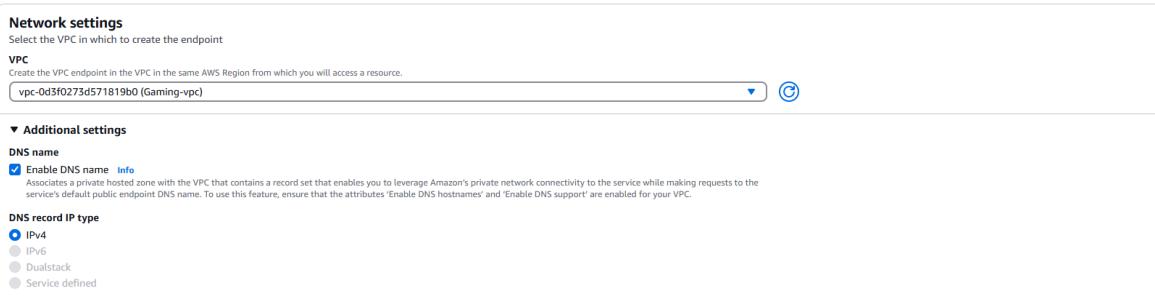
[Cancel](#)
[Create endpoint](#)
Servicio EC2

i) Buscamos el servicio *com.amazonaws.us-east-2.ec2*


Services (1/1)

Search: com.amazonaws.us-east-2.ec2

Service Name	Owner	Type	Service Region
com.amazonaws.us-east-2.ec2	amazon	Interface	us-east-2

ii) Seleccionamos el VPC, habilitamos DNS y ipv4


Network settings
Select the VPC in which to create the endpoint

VPC
Create the VPC endpoint in the VPC in the same AWS Region from which you will access a resource.
vpc-0d3f0273d571819b0 (Gaming-vpc)

Additional settings

DNS name
 Enable DNS name [Info](#)
Associates a private hosted zone with the VPC that contains a record set that enables you to leverage Amazon's private network connectivity to the service while making requests to the service's default public endpoint DNS name. To use this feature, ensure that the attributes 'Enable DNS hostnames' and 'Enable DNS support' are enabled for your VPC.

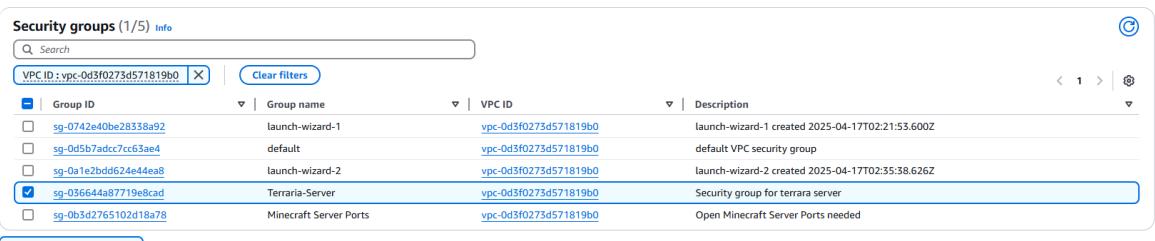
DNS record IP type
 IPv4
 IPv6
 Dualstack
 Service defined

iii) Seleccionamos la zona y la subnet privada


Subnets (1/3) [Info](#)

Availability Zone	Subnet ID	Designate IP addresses	IPv4 address	IPv6 address
<input checked="" type="checkbox"/> us-east-2a (use2-az1)	subnet-0a8af7c26be433c6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> us-east-2b (use2-az2)	Select a subnet	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> us-east-2c (use2-az3)	No subnet available	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

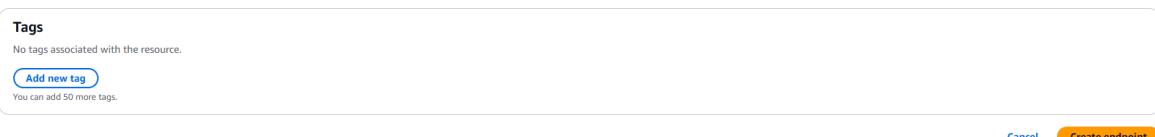
IP address type
 IPv4
 IPv6
 Dualstack

iv) Seleccionamos el SG Terraria-server


Security groups (1/5) [Info](#)

VPC ID: vpc-0d3f0273d571819b0

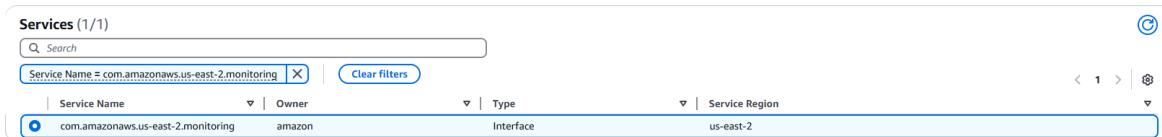
Group ID	Group name	VPC ID	Description
<input type="checkbox"/> sg-0742e40be28338a92	launch-wizard-1	vpc-0d3f0273d571819b0	launch-wizard-1 created 2025-04-17T02:21:53.600Z
<input type="checkbox"/> sg-0d5b7adcc7cc63ae4	default	vpc-0d3f0273d571819b0	default VPC security group
<input type="checkbox"/> sg-0a1e2bdd624e44ea8	launch-wizard-2	vpc-0d3f0273d571819b0	launch-wizard-2 created 2025-04-17T02:35:38.626Z
<input checked="" type="checkbox"/> sg-036644a87719ebcad	Terraria-Server	vpc-0d3f0273d571819b0	Security group for terrara server
<input type="checkbox"/> sg-0b3d2765102d18a78	Minecraft Server Ports	vpc-0d3f0273d571819b0	Open Minecraft Server Ports needed

v) Creamos el endpoint


Tags
No tags associated with the resource.
[Add new tag](#)
You can add 50 more tags.

[Cancel](#) [Create endpoint](#)

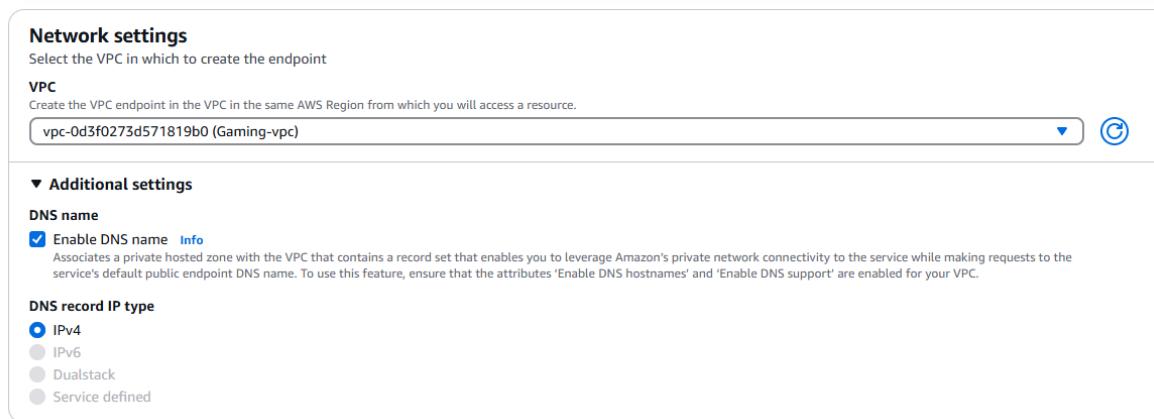
Servicio CloudWatch

i) Buscamos el servicio *com.amazonaws.us-east-2.monitoring*


Services (1/1)

Service Name = com.amazonaws.us-east-2.monitoring

Service Name	Owner	Type	Service Region
com.amazonaws.us-east-2.monitoring	amazon	Interface	us-east-2

ii) Elejimos el VPC, DNS y ipv4


Network settings

Select the VPC in which to create the endpoint

VPC

Create the VPC endpoint in the VPC in the same AWS Region from which you will access a resource.

vpc-0d3f0273d571819b0 (Gaming-vpc)

Additional settings

DNS name

Enable DNS name [Info](#)

Associates a private hosted zone with the VPC that contains a record set that enables you to leverage Amazon's private network connectivity to the service while making requests to the service's default public endpoint DNS name. To use this feature, ensure that the attributes 'Enable DNS hostnames' and 'Enable DNS support' are enabled for your VPC.

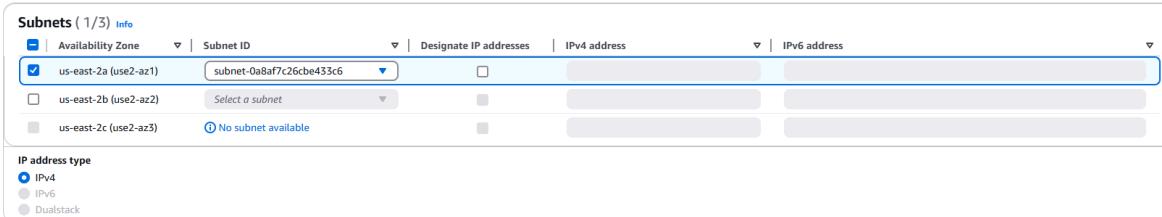
DNS record IP type

IPv4

IPv6

Dualstack

Service defined

iii) Elejimos la subnet 2a y su rama privada


Subnets (1/3)

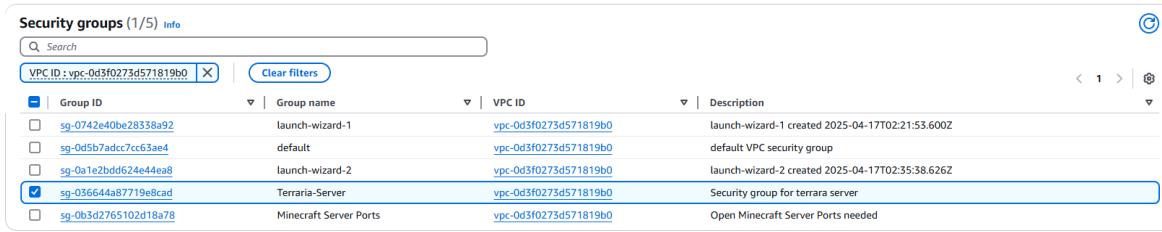
Availability Zone	Subnet ID	Designate IP addresses	IPv4 address	IPv6 address
<input checked="" type="checkbox"/> us-east-2a (use2-az1)	subnet-0a8af7c26cbe433c6	<input type="checkbox"/>		
<input type="checkbox"/> us-east-2b (use2-az2)	Select a subnet	<input type="checkbox"/>		
<input type="checkbox"/> us-east-2c (use2-az3)	No subnet available	<input type="checkbox"/>		

IP address type

IPv4

IPv6

Dualstack

iv) Elejimos el security group Terraria Server


Security groups (1/5)

Group ID	Group name	VPC ID	Description
<input type="checkbox"/> sg-0742e40be28338a92	launch-wizard-1	vpc-0d3f0273d571819b0	launch-wizard-1 created 2025-04-17T02:21:53.600Z
<input type="checkbox"/> sg-0d5b7adc7cc63ae4	default	vpc-0d3f0273d571819b0	default VPC security group
<input type="checkbox"/> sg-0a1e2bdd524e44ea8	launch-wizard-2	vpc-0d3f0273d571819b0	launch-wizard-2 created 2025-04-17T02:35:38.626Z
<input checked="" type="checkbox"/> sg-036644a87719e8cad	Terraria-Server	vpc-0d3f0273d571819b0	Security group for terraria server
<input type="checkbox"/> sg-0b3d2765102d18a78	Minecraft Server Ports	vpc-0d3f0273d571819b0	Open Minecraft Server Ports needed

v) Creamos el endpoint

Tags

No tags associated with the resource.

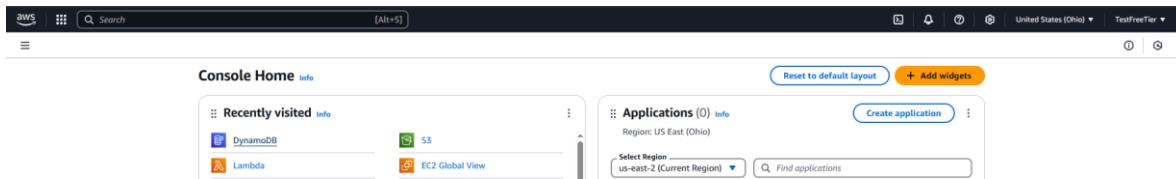
[Add new tag](#)

You can add 50 more tags.

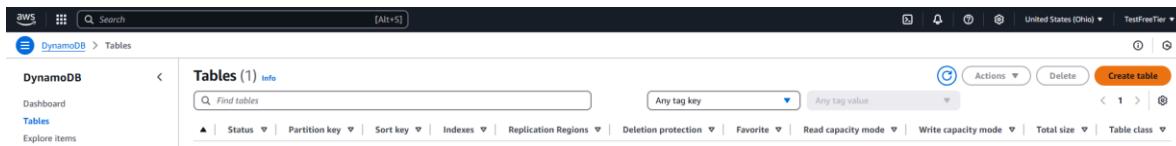
[Cancel](#) [Create endpoint](#)

AMAZON DYNAMO:

1.- Desde la consola de AWS acceder al servicio de DYNAMO DB



2.- Dentro del menú de tablas dar clic en crear tabla



3.- Especificamos el nombre de la tabla, su llave primaria y su sort key

Create table

Table details [Info](#)
Dynamodb is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

4.- Creamos la tabla

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

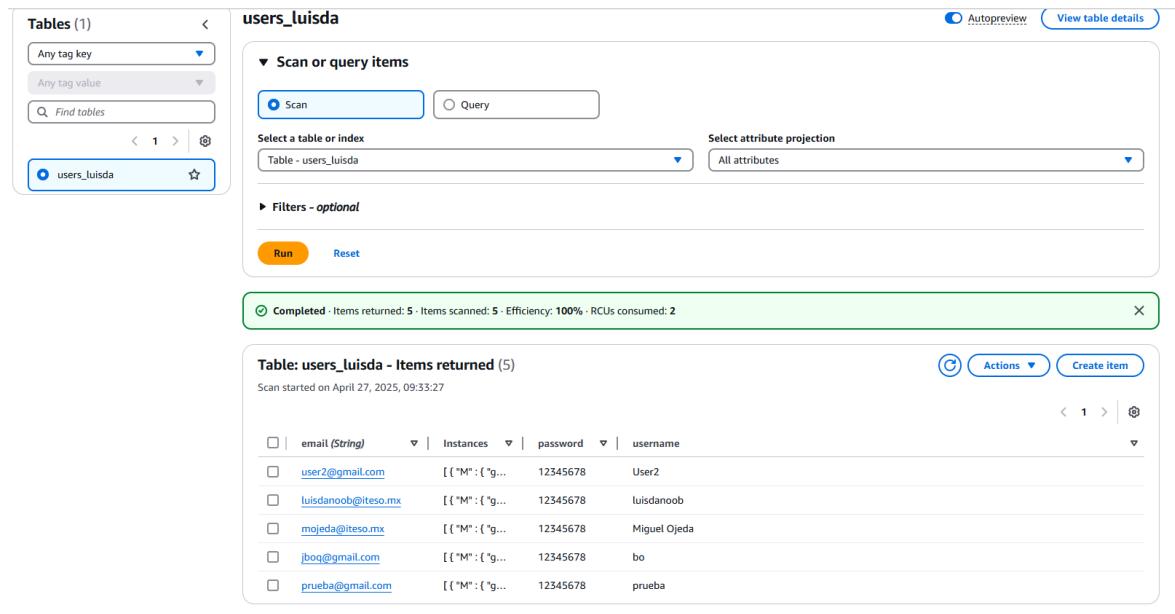
No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#)
[Create table](#)

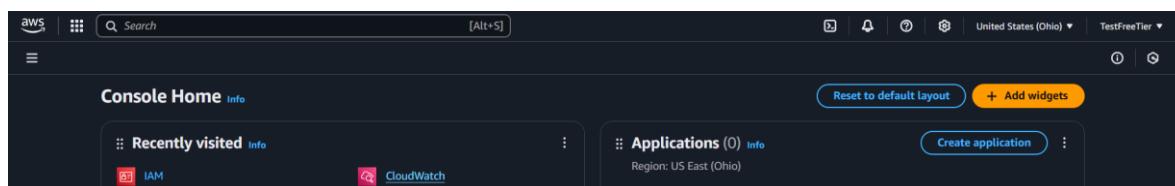
5.- Podemos acceder a los ítems de nuestra nueva tabla



The screenshot shows the AWS Lambda function configuration page. The 'Handler' field is set to 'lambda_function.lambda_handler'. Other fields like 'Runtime', 'Memory size', and 'Timeout' are also visible.

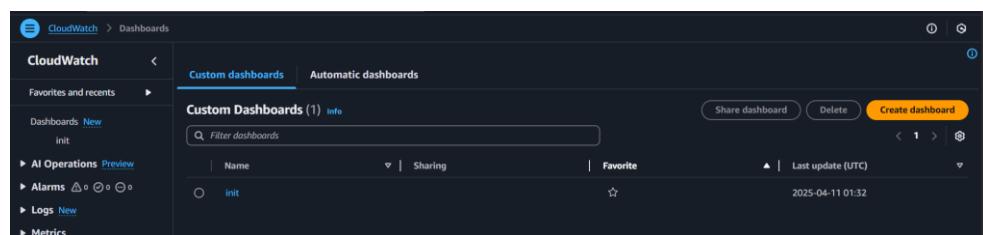
CLOUD WATCH:

1.- Desde la consola de AWS acceder al servicio CloudWatch



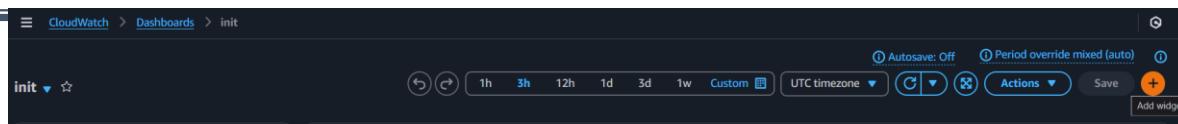
The screenshot shows the AWS CloudWatch console home page. It features sections for 'Recently visited' (IAM, CloudWatch Metrics), 'Applications (0)', and 'Logs (0)'. A sidebar on the left provides navigation for CloudWatch services like AI Operations, Alarms, Logs, and Metrics.

2.- Dar create Dashboard

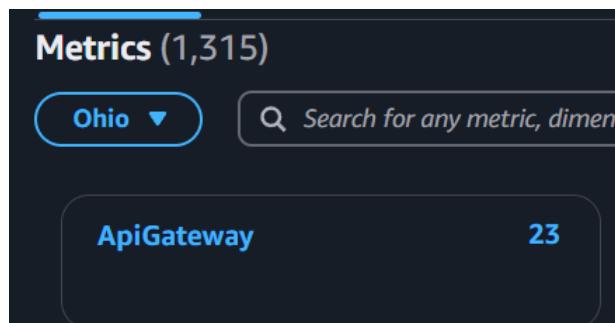


The screenshot shows the AWS CloudWatch Dashboards page. It displays a list of custom dashboards under 'Custom Dashboards'. The 'init' dashboard is selected, showing its last update time as '2025-04-11 01:32'.

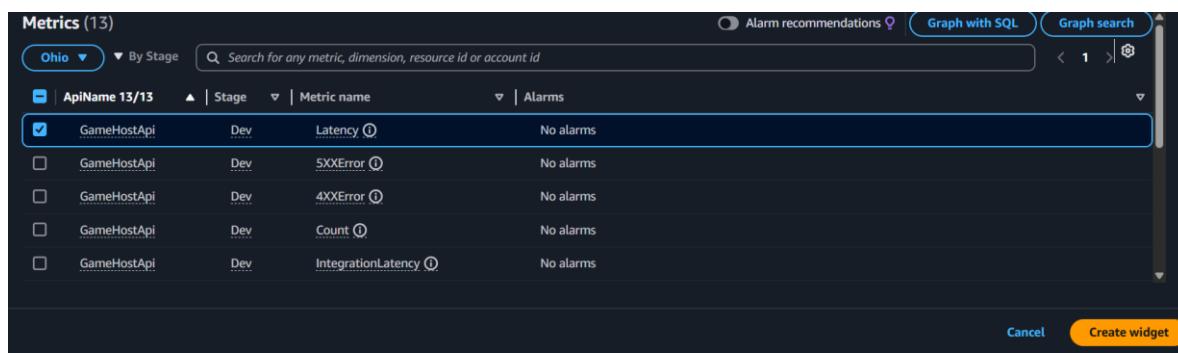
3.- Añadir métricas



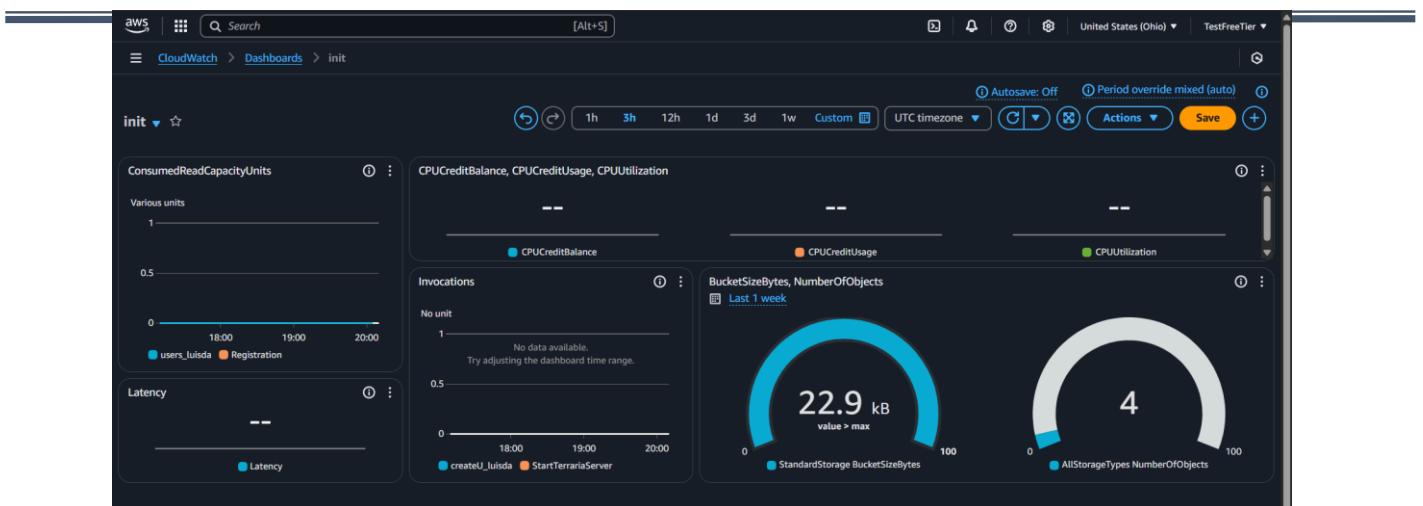
4.- Elegir la instancia y lo que se va a medir y el tipo de grafica

5.- Elegir métrica y creamos widget

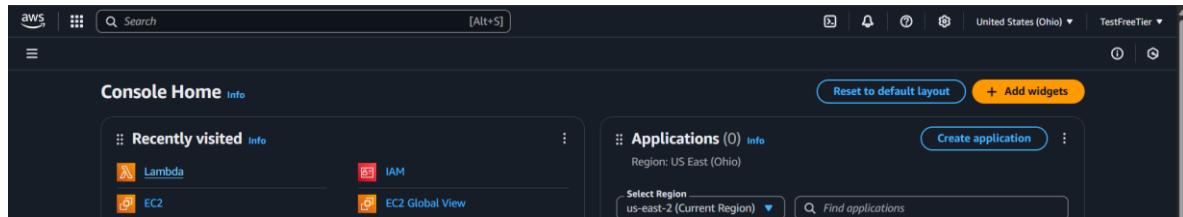


6.- Mostrar el dashboard

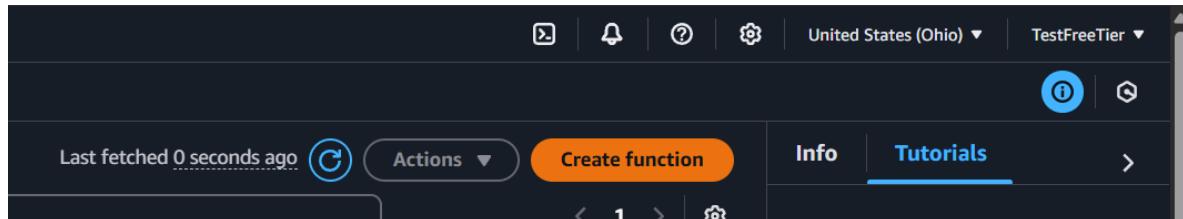


LAMBDA:

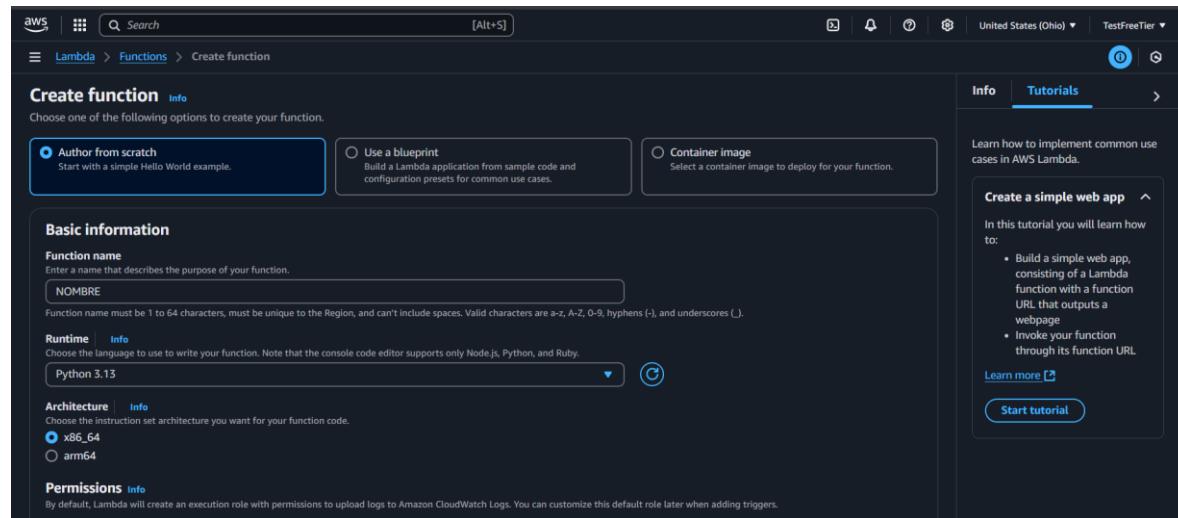
1.- Desde la consola de AWS acceder al servicio LAMBDA



2.- Dentro del servicio LAMBDA crear una función nueva



3.- Dejamos el autor from scratch, el nombre que queramos de la función, usando Python 3.13, y con un rol nuevo



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.

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

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.

Tutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

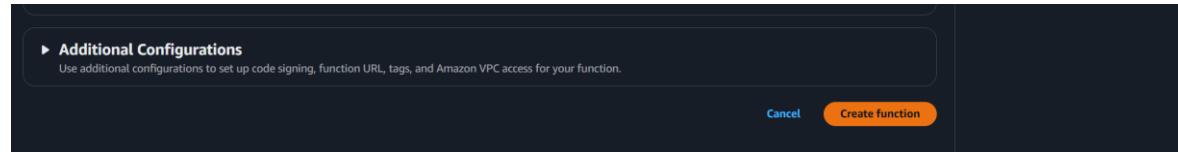
In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

[Learn more](#) ^

[Start tutorial](#)

4.- Creamos la función:



Additional Configurations

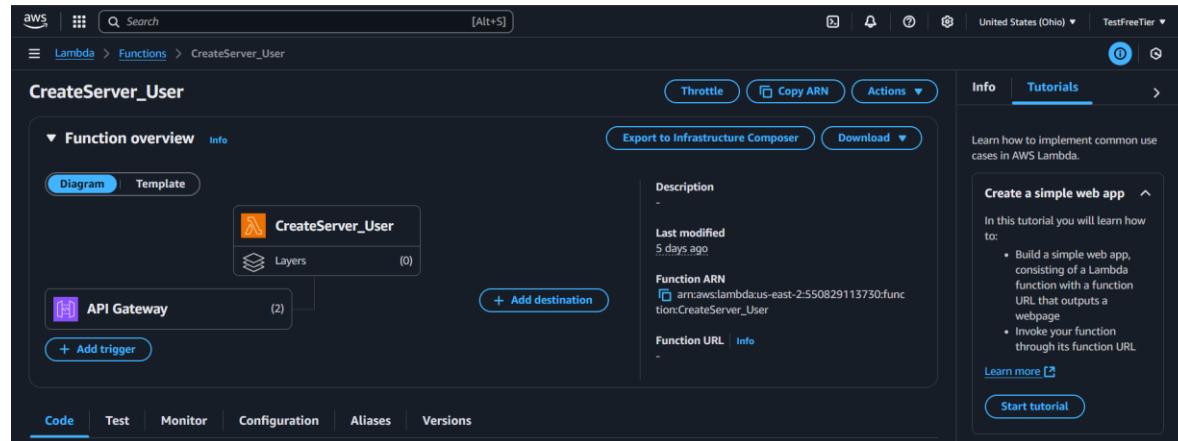
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

[Cancel](#) [Create function](#)

5.- Funciones:

FUNCIÓN Crear Instancias desde lambda:

i) En el menú principal de nuestra función



CreateServer_User

Function overview Info

[Diagram](#) [Template](#)

Description
-

Last modified
5 days ago

Function ARN
[arn:aws:lambda:us-east-2:550829113730:function:CreateServer_User](#)

Function URL Info
-

Throttle [Copy ARN](#) [Actions](#)

[Export to Infrastructure Composer](#) [Download](#)

Code [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

Tutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

[Learn more](#) ^

[Start tutorial](#)

 ii) Código

```

import boto3
import time
import json
from botocore.exceptions import ClientError

dynamodb = boto3.resource('dynamodb') #conexión con DYNAMO
tabla = dynamodb.Table('users_luisda') #conexión con la tabla de DYNAMO

def lambda_handler(event, context):
    # Manejo de CORS preflight
    if event['httpMethod'] == 'OPTIONS':
        return {
            'statusCode': 200,
            'headers': {           # Estos headers nos sirven para permitir el
                'Access-Control-Allow-Origin': "*",
                'Access-Control-Allow-Headers': "*",
                'Access-Control-Allow-Methods': "OPTIONS,POST"
            },
            'body': json.dumps('CORS preflight OK')
        }

    # Obtener el cuerpo de la solicitud (ahora el cuerpo es un JSON)
    body = json.loads(event["body"])

    if 'User' not in body: #Manejo de error en la solicitud
        return {
            'statusCode': 400,
            'headers': {
                'Access-Control-Allow-Origin': "*",
                'Access-Control-Allow-Headers': "*",
                'Access-Control-Allow-Methods': "OPTIONS,POST"
            },
            'body': json.dumps({'error': 'No se proporcionó un cuerpo en la
solicitud'})
        }

    email = body["User"]["email"] #obtenemos el email
    password = body["User"]["password"] #obtenemos el password
  
```

```

#Manejamos error de falta de parámetros
if not email or not password:
    return {
        'statusCode': 400,
        'headers': {
            "Access-Control-Allow-Origin": "*",
            "Access-Control-Allow-Headers": "*",
            "Access-Control-Allow-Methods": "OPTIONS,POST"
        },
        'body': json.dumps({'error': 'Faltan datos requeridos'})
    }

game = body["game"] #Obtenemos el juego que se va a levantar
try:
    response = tabla.get_item(Key={'email': email})
    if 'Item' not in response or response['Item']['password'] != password: #Validamos contraseña
        return {
            'statusCode': 400,
            'headers': {
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Headers": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST"
            },
            'body': json.dumps({'error': 'El usuario no existe o la contraseña es incorrecta'})
        }

    #Dependiendo el juego se mapea con el ID de la imagen con el servidor ya instalado
    ami_map = {
        "terraria": "ami-02af08be27104ae31"
    }

    #Si el juego no existe en nuestros servidores
    if game not in ami_map:
        return {
            'statusCode': 400,
            'headers': {
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Headers": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST"
            },

```

```

        'body': json.dumps({'error': f"Juego '{game}' no
soportado"})
    }

    nombre_instancia = f"Servidor-{game}-{int(time.time())}" #Creamos el
nombre del servidor
    ec2 = boto3.client('ec2') #Nos conectamos con EC2

    #Ejecutamos run_instances que crea las instancias
    response = ec2.run_instances(
        ImageId=ami_map[game], #Imagen
        InstanceType='t2.micro', #Tipo de instancia
        MinCount=1,
        MaxCount=1,
        KeyName='voockey', #LLave
        SecurityGroupIds=['sg-036644a87719e8cad'],
        SubnetId='subnet-02b8b5064491aec02',
        TagSpecifications=[{
            'ResourceType': 'instance',
            'Tags': [{'Key': 'Name', 'Value': nombre_instancia}]
        }]
    )

    #Obtenemos los parámetros de la instancia para mandarsela al usuario
    instance_id = response['Instances'][0]['InstanceId']
    ec2_resource = boto3.resource('ec2')
    instance = ec2_resource.Instance(instance_id)
    instance.wait_until_running()
    instance.load()
    ip = instance.public_ip_address

    # Actualizar la base de datos con la nueva instancia y asociar el
juego
    tabla.update_item(
        Key={'email': email},
        UpdateExpression='SET Instances = list_append(Instances,
:new_instance)',
        ExpressionAttributeValues={
            ':new_instance': [
                {'instance_id': instance_id,
                 'game': game, # Aquí guardamos el nombre del juego
junto con la instancia
```
```

```

 'public_ip': ip
 }]
}
)

#Regresamos la información vital para el usuario
return {
 "statusCode": 200,
 "headers": {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "*",
 "Access-Control-Allow-Methods": "OPTIONS,POST",
 "Content-Type": "application/json"
 },
 "body": json.dumps({
 "game": game, #El juego con el servidor
 "instance_id": instance_id, #El ID de la instancia
 "public_ip": ip #Su ip para poderse conectar
 })
}

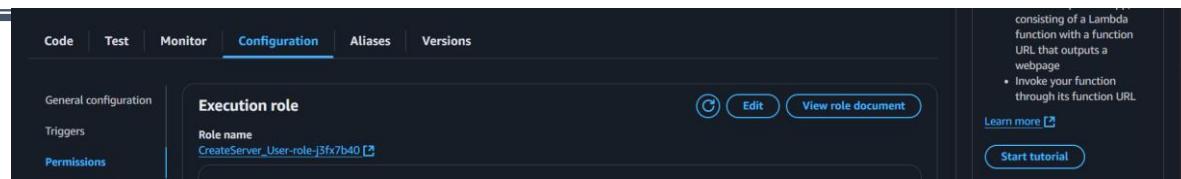
#Manejamos exception de error con el cliente
except ClientError as e:
 return {
 'statusCode': 500,
 'headers': {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "*",
 "Access-Control-Allow-Methods": "OPTIONS,POST"
 },
 'body': json.dumps({'error': 'Error en la base de datos',
'details': str(e)})
 }

```

iii) Para el test

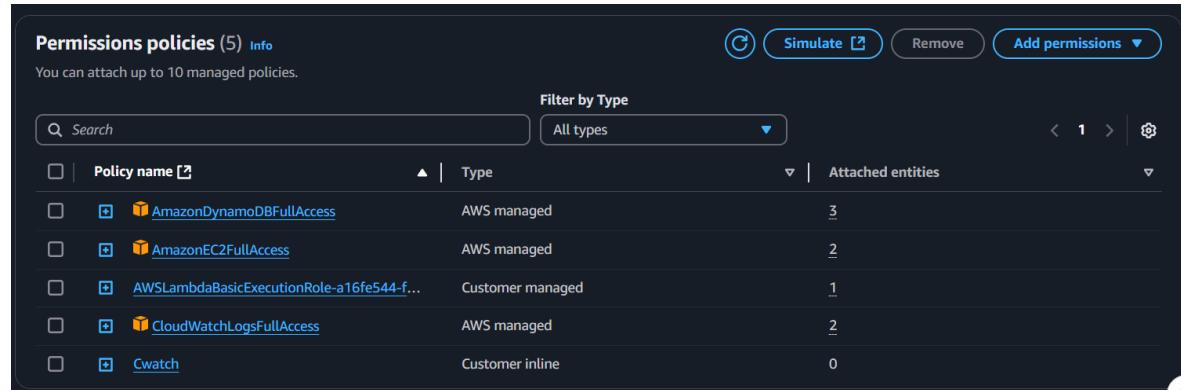
```
{
 "httpMethod": "POST",
 "body": "{\"User\":{\"email\":\"user2@gmail.com\",\"password\":\"12345678\"},\"game\":\n \"terrraria\"}"
}
```

iv) Configurar permisos



The screenshot shows the 'Execution role' section of the AWS Lambda function configuration. It displays the role name 'CreateServer\_User-role-j3fx7b40'. There are buttons for 'Edit', 'View role document', 'Learn more', and 'Start tutorial'. A sidebar on the right provides information about Lambda functions.

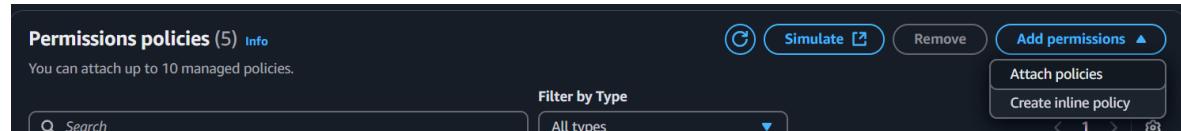
Agregar permisos desde IAM en el rol



The screenshot shows the 'Permissions policies' list with 5 items. The policies are:

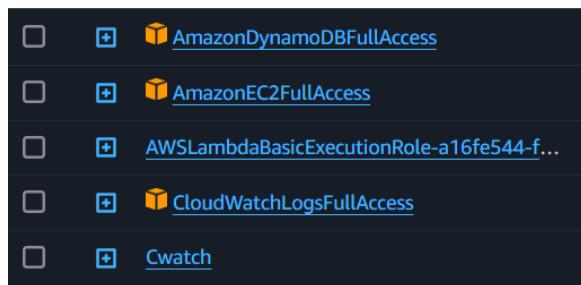
| Policy name                                               | Type             | Attached entities |
|-----------------------------------------------------------|------------------|-------------------|
| <a href="#">AmazonDynamoDBFullAccess</a>                  | AWS managed      | 3                 |
| <a href="#">AmazonEC2FullAccess</a>                       | AWS managed      | 2                 |
| <a href="#">AWSLambdaBasicExecutionRole-a16fe544-f...</a> | Customer managed | 1                 |
| <a href="#">CloudWatchLogsFullAccess</a>                  | AWS managed      | 2                 |
| <a href="#">Cwatch</a>                                    | Customer inline  | 0                 |

Dar click en Add permissions y Attach policies



The screenshot shows the same 'Permissions policies' list as before, but with the 'Add permissions' button highlighted. This button is located at the top right of the list.

Agregar estos permisos:



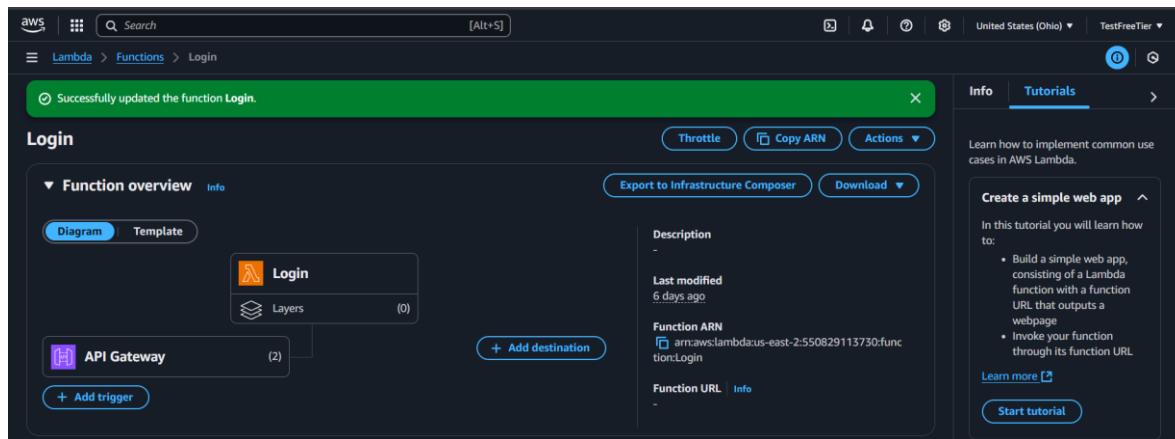
The list includes the following policies:

- [AmazonDynamoDBFullAccess](#)
- [AmazonEC2FullAccess](#)
- [AWSLambdaBasicExecutionRole-a16fe544-f...](#)
- [CloudWatchLogsFullAccess](#)
- [Cwatch](#)

## FUNCTION Login:

---

 i) Consola principal de nuestra función



## ii) Código

```

import boto3
import json
from botocore.exceptions import ClientError
import time

Inicializar recurso DynamoDB
dynamodb = boto3.resource('dynamodb')
tabla = dynamodb.Table('users_luisda')

Cabeceras CORS comunes
cors_headers = {
 'Access-Control-Allow-Origin': '*',
 'Access-Control-Allow-Headers': 'Content-Type',
 'Access-Control-Allow-Methods': 'OPTIONS,POST'
}

def lambda_handler(event, context):
 start_time = time.time() # Registrar el tiempo de inicio
 print("Evento recibido:", event)

 # Validar entrada
 if 'User' not in event:
 print("Falta el campo 'User'")
 return {
 'statusCode': 400,
 'headers': cors_headers,
 'body': json.dumps({'error': 'No se proporcionó el campo "User" en la solicitud'})
 }

```

```

}

email = event["User"].get("email") #Obtenemos email
password = event["User"].get("password") #Obtenemos password

#Validamos que si existan
if not email or not password:
 print("Faltan email o password")
 return {
 'statusCode': 400,
 'headers': cors_headers,
 'body': json.dumps({'error': 'Faltan datos requeridos: email o password'})
 }

try:
 print(f"Consultando usuario con email: {email}")
 response = tabla.get_item(Key={'email': email}) #Obtenemos el email de la tabla
 print("Consulta a DynamoDB completada en", time.time() - start_time, "segundos")

 #Si no encuentra el email
 if 'Item' not in response:
 print("Usuario no encontrado")
 return {
 'statusCode': 404,
 'headers': cors_headers,
 'body': json.dumps({'error': 'El usuario no existe'})
 }

 user_data = response['Item'] #Si encuentra el email regresa sus datos
 print("Datos obtenidos del usuario:", user_data)

 if user_data.get('password') != password: #Ahora validamos si la contraseña es correcta
 print("Contraseña incorrecta")
 return {
 'statusCode': 401,
 'headers': cors_headers,
 'body': json.dumps({'error': 'Contraseña incorrecta'})
 }

```

```

 print("Autenticación exitosa") #Validamo el login
 return {
 'statusCode': 200,
 'headers': cors_headers,
 #Regresamos la información personal del usuario para las
variables globales
 'body': json.dumps({
 'username': user_data.get('username'),
 'email': user_data.get('email'),
 'instances': user_data.get('Instances', []),
 'message': 'Usuario autenticado correctamente'
 })
 }

#Manejamos excepciones de conexión

except ClientError as e:
 print("Error al consultar DynamoDB:", str(e))
 return {
 'statusCode': 500,
 'headers': cors_headers,
 'body': json.dumps({'error': 'Error en la base de datos',
'details': str(e)})
 }

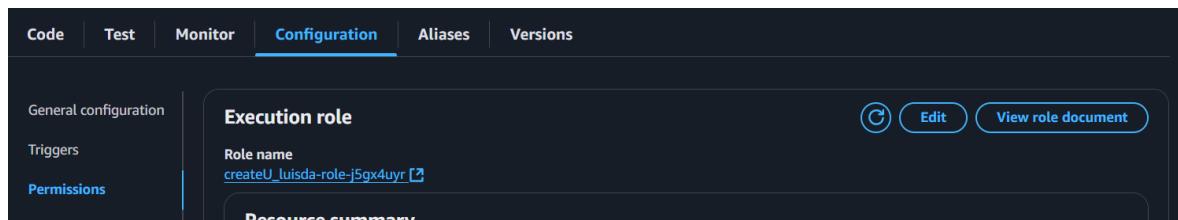
except Exception as e:
 print("Error inesperado:", str(e))
 return {
 'statusCode': 500,
 'headers': cors_headers,
 'body': json.dumps({'error': 'Error inesperado', 'details':
str(e)})
 }

```

iii) Test

```
{
 "User": {
 "email": "user2@gmail.com",
 "password": "12345678"
 }
}
```

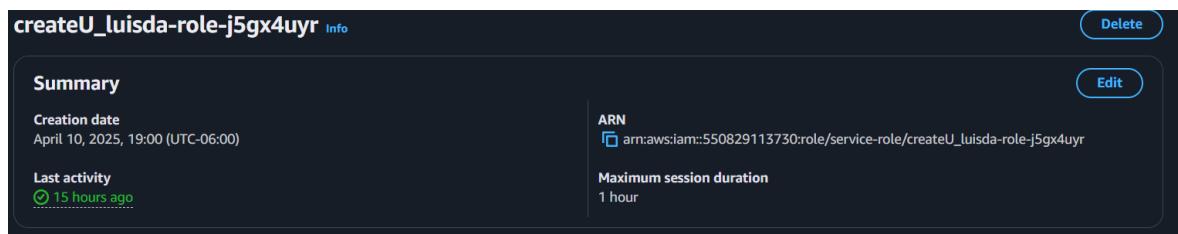
**iv) Configurar permisos**



The screenshot shows the AWS Lambda Configuration tab. The 'Code' tab is selected. In the main area, there is a JSON object representing a user:

```
{
 "User": {
 "email": "user2@gmail.com",
 "password": "12345678"
 }
}
```

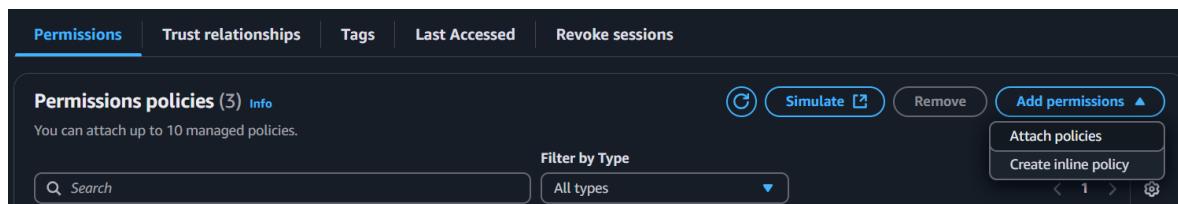
**v) Ir al rol de IAM**



The screenshot shows the AWS IAM Role details page for the role 'createU\_luisda-role-j5gx4uyr'. The 'Summary' section includes:

- Creation date:** April 10, 2025, 19:00 (UTC-06:00)
- Last activity:** 15 hours ago
- ARN:** arn:aws:iam::550829113730:role/service-role/createU\_luisda-role-j5gx4uyr
- Maximum session duration:** 1 hour

**vi) Agregar permisos y attach policies**



The screenshot shows the AWS IAM Permissions tab. The 'Permissions policies' section lists three managed policies:

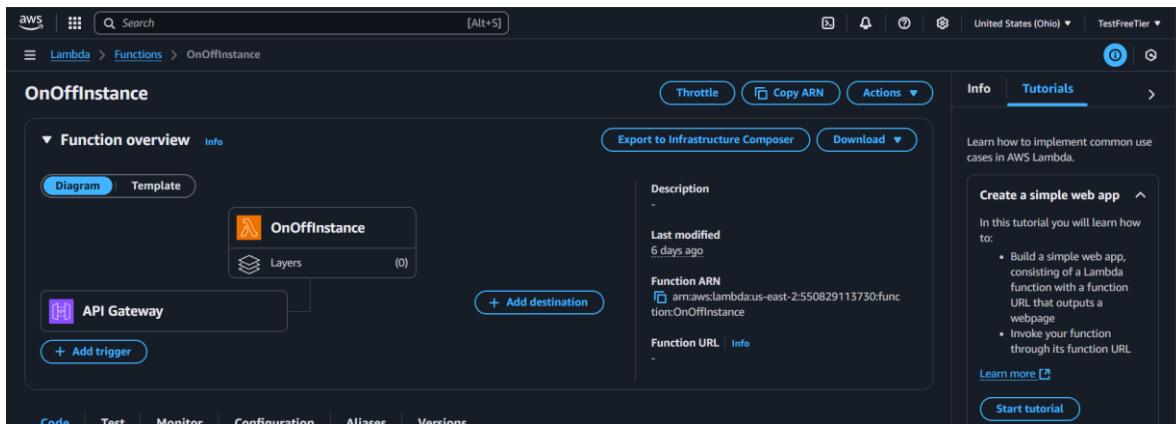
- AmazonDynamoDBFullAccess
- AWSLambdaBasicExecutionRole-f360045c-...
- AWSLambdaVPCAccessExecutionRole

**vii) Agregar estos permisos**

| <input type="checkbox"/> | Policy name                              | Type             |
|--------------------------|------------------------------------------|------------------|
| <input type="checkbox"/> | AmazonDynamoDBFullAccess                 | AWS managed      |
| <input type="checkbox"/> | AWSLambdaBasicExecutionRole-f360045c-... | Customer managed |
| <input type="checkbox"/> | AWSLambdaVPCAccessExecutionRole          | AWS managed      |

## FUNCTION Apagar y Prender Servidor

---

i) Menú principal de nuestra función


## ii) Código

```

import boto3
import json

#Conexión con EC2
ec2 = boto3.client('ec2')

def lambda_handler(event, context):
 try:
 body = json.loads(event["body"])

 #Si el parametro instance no esta en el JSON
 if 'Instance' not in body:
 return {
 'statusCode': 400,
 'headers': {
 'Access-Control-Allow-Origin': '*'
 },
 'body': json.dumps({'error': 'No se proporcionó una
instancia'})
 }

 instance_id = body["Instance"]["id"] #Obtenemos el id de la
instancia
 status = body["Instance"]["status"] # Obtenemos su estatus al que se
va a cambiar

 #Dependiendo de su estatus start o stop instance
 if status == "ON":

```

```

 ec2.start_instances(InstanceIds=[instance_id])
 message = f'Instancia {instance_id} encendida'
 elif status == "OFF":
 ec2.stop_instances(InstanceIds=[instance_id])
 message = f'Instancia {instance_id} apagada'
 else:
 return {
 'statusCode': 400,
 'headers': {
 'Access-Control-Allow-Origin': '*'
 },
 'body': json.dumps({'error': 'Estado inválido. Usa ON u
OFF'})
 }

 #Regresamos si prendimos o apagamos la instancia
 return {
 'statusCode': 200,
 'headers': {
 'Access-Control-Allow-Origin': '*',
 'Access-Control-Allow-Headers': '*'
 },
 'body': json.dumps({'message': message})
 }

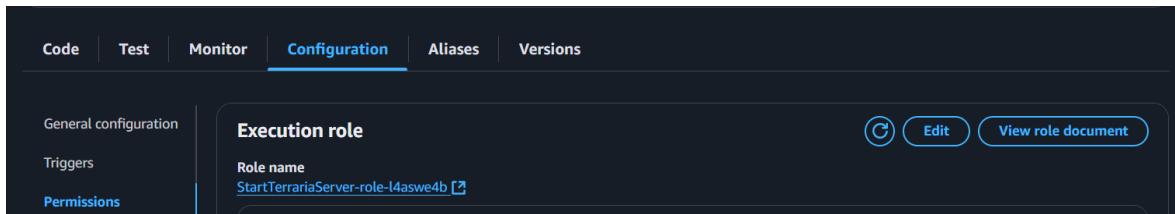
#Mala conexión

except Exception as e:
 return {
 'statusCode': 500,
 'headers': {
 'Access-Control-Allow-Origin': '*'
 },
 'body': json.dumps({'error': str(e)})
 }

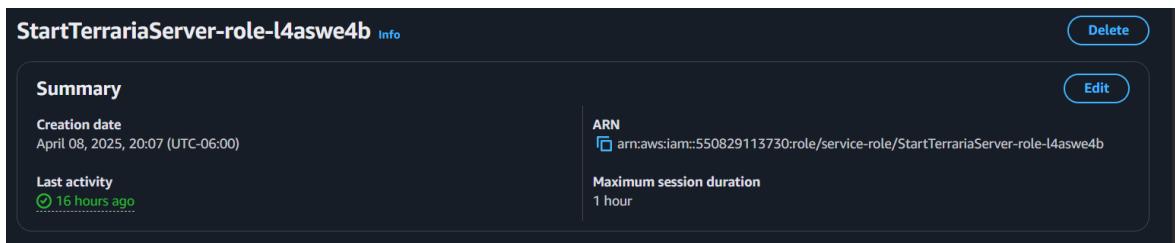
```

```
{
 "body": [
 {"\Instance": {"id": "i-0415d3cfdae1048f1", "status": "OFF"}}
]
}
```

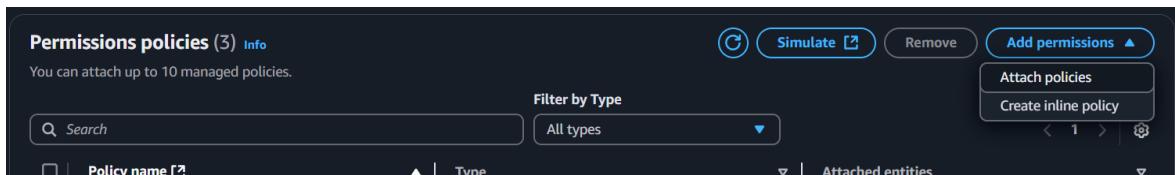
iv) En la configuración ir a permisos



v) Ir al rol de IAM



vi) Add permissions y Attach policies



vii) Añadir estos permisos

| <input type="checkbox"/> Policy name                                              | Type             |
|-----------------------------------------------------------------------------------|------------------|
| <input type="checkbox"/> <a href="#">AmazonEC2FullAccess</a>                      | AWS managed      |
| <input type="checkbox"/> <a href="#">AWSLambdaBasicExecutionRole-b317ba3e-...</a> | Customer managed |
| <input type="checkbox"/> <a href="#">CloudWatchLogsFullAccess</a>                 | AWS managed      |

## FUNCTION Estatus de la instancia

---

**i) ESTA FUNCIÓN USA EL MISMO ROL CreateServer\_User-role**

**Execution role**

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role

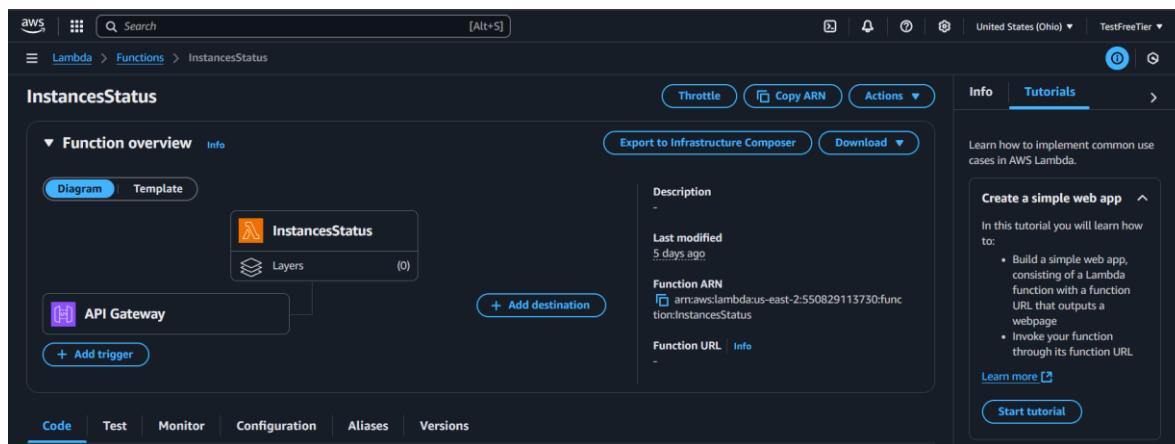
Create a new role from AWS policy templates

**Existing role**

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/CreateServer\_User-role-j3fx7b40

[View the CreateServer\\_User-role-j3fx7b40 role](#) on the IAM console.

**ii) Menú principal de nuestra función**

**iii) Código**

```

import boto3
import json
import time
from datetime import datetime, timedelta
from botocore.exceptions import ClientError

Clientes de AWS
dynamodb = boto3.resource('dynamodb') #DYNAMO
tabla = dynamodb.Table('users_luisda') #Tabla de dynamo
ec2 = boto3.client('ec2') #EC2
cloudwatch = boto3.client('cloudwatch') #nos conectamos a CLOUD WATCH para métricas

Función para obtener promedio de CPU
def get_metric_average(metric_name, namespace, dimensions):
 end_time = datetime.utcnow()

```

```

 start_time = end_time - timedelta(hours=12) #Especificamos un tiempo
 (ultimas 12 horas)

 #Llamamos a CloudWatch para que nos regrese las metricas deseadas
 response = cloudwatch.get_metric_statistics(
 Namespace=namespace,
 MetricName=metric_name,
 Dimensions=dimensions,
 StartTime=start_time,
 EndTime=end_time,
 Period=300,
 Statistics=['Average']
)

 #filtramos los puros datos de la respuesta
 datapoints = response.get('Datapoints', [])
 if not datapoints:
 return None

 #Retornamos los datos
 return round(sorted(datapoints, key=lambda x: x['Timestamp'],
 reverse=True)[0]['Average'], 2)

def lambda_handler(event, context):
 try:
 body = json.loads(event['body'])
 user = body.get('User') #Obtenemos el usuario

 #Si no estamos logeados
 if not user:
 return {
 "statusCode": 400,
 "body": json.dumps({"error": "No se proporcionó un usuario
válido"}),
 "headers": {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "Content-Type",
 "Access-Control-Allow-Methods": "OPTIONS,POST"
 }
 }

 email = user.get("email") #Obtenemos el email
 password = user.get("password") #Obtenemos password

```

```

#Validamos la existencia del email y password
if not email or not password:
 return {
 'statusCode': 400,
 'body': json.dumps({'error': 'Faltan datos requeridos'}),
 'headers': {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "Content-Type",
 "Access-Control-Allow-Methods": "OPTIONS,POST"
 }
 }

response = tabla.get_item(Key={'email': email}) #Obtenemos si el
usuario si esta en la tabla
 if 'Item' not in response or response['Item']['password'] != password: #Validamos si la contraseña es correcta y el usuario existe
 return {
 'statusCode': 400,
 'body': json.dumps({'error': 'El usuario no existe o la
contraseña es incorrecta'}),
 'headers': {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "Content-Type",
 "Access-Control-Allow-Methods": "OPTIONS,POST"
 }
 }

Obtenemos las instancias del usuario
isinstances = response['Item'].get('Instances', [])
if not isinstances:
 return {
 'statusCode': 400,
 'body': json.dumps({'error': 'No hay instancias asociadas al
usuario'}),
 'headers': {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "Content-Type",
 "Access-Control-Allow-Methods": "OPTIONS,POST"
 }
 }

```

```

 instance_ids = [inst['instance_id'] for inst in instances]
#Obtenemos su id
 response = ec2.describe_instances(InstanceIds=instance_ids)

 instances_info = []
 for reservation in response['Reservations']:
 for instance in reservation['Instances']:
 instance_id = instance['InstanceId']
 game_name = next((inst['game'] for inst in instances if
inst['instance_id'] == instance_id), 'No Game') #Obtenemos el servidor

 # Obtener rendimiento CPU
 cpu_utilization = get_metric_average(
 metric_name='CPUUtilization',
 namespace='AWS/EC2',
 dimensions=[{'Name': 'InstanceId', 'Value':
instance_id}]
)

 #Obtenemos la información completa de la instancia
 info = {
 'InstanceId': instance_id,
 'game': game_name,
 'PublicIpAddress': instance.get('PublicIpAddress', 'No
Public IP'),
 'State': instance['State']['Name'],
 'cpuUtilization': f"{cpu_utilization}%" if
cpu_utilization is not None else 'No data'
 }
 instances_info.append(info)

 #Si todo función regresamos 200 y la información de todas las
 #instancias del usuario
 return {
 'statusCode': 200,
 'headers': {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "Content-Type",
 "Access-Control-Allow-Methods": "OPTIONS,POST"
 },
 'body': json.dumps({'instances': instances_info})
 }

```

```

#Excepción de conexión con DYNAMO
except ClientError as e:
 return {
 'statusCode': 500,
 'body': json.dumps({'error': 'Error en la base de datos',
'details': str(e)}),
 'headers': {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "Content-Type",
 "Access-Control-Allow-Methods": "OPTIONS,POST"
 }
 }

#Error de conexión
except Exception as e:
 return {
 "statusCode": 500,
 "body": json.dumps({"error": str(e)}),
 "headers": {
 "Access-Control-Allow-Origin": "*",
 "Access-Control-Allow-Headers": "Content-Type",
 "Access-Control-Allow-Methods": "OPTIONS,POST"
 }
 }

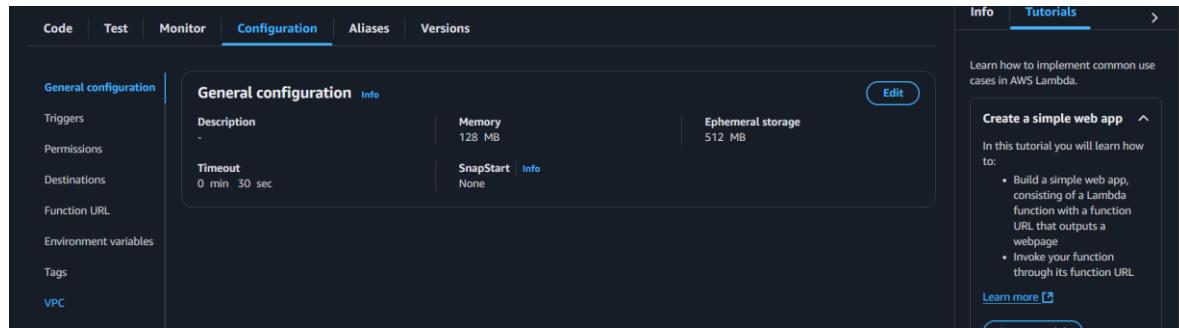
```

## iv) Test

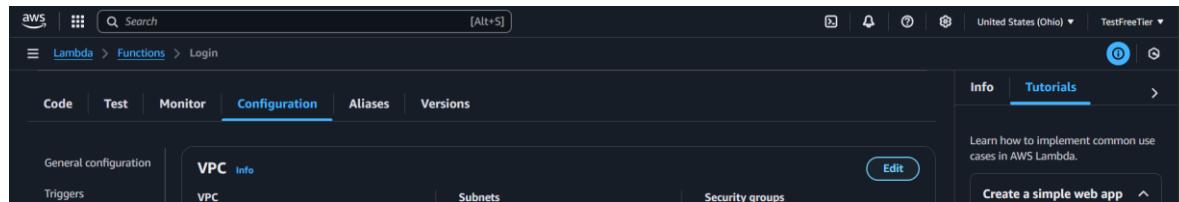
```
{
"body":
 "{\"User\":{\"email\":\"user2@gmail.com\",\"password\":\"12345678\"}}"
```

## 6.- Conectar funciones lambda con el VPC subnet privada

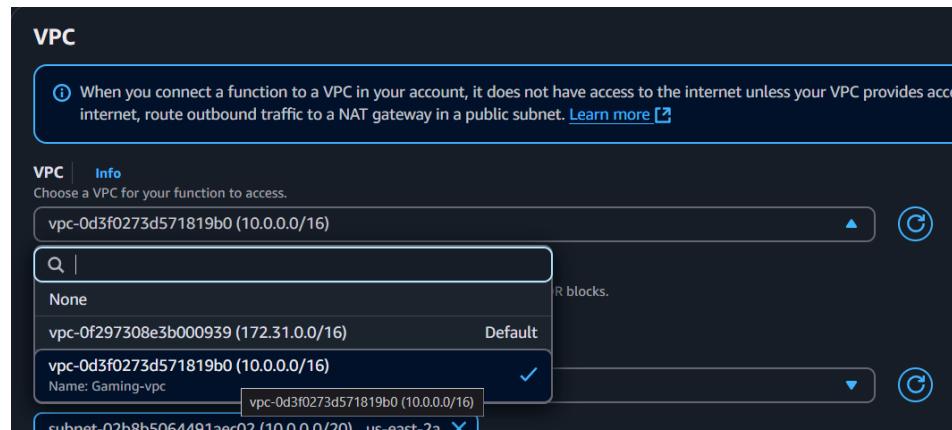
### i.- En la configuración de la función irnos al apartado del VPC



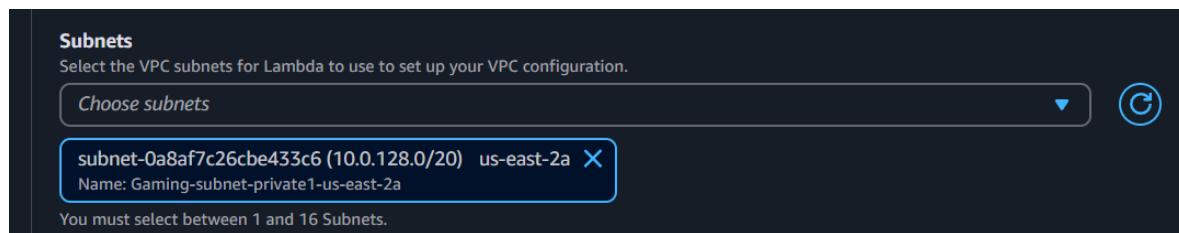
### ii.- Dar click a editar



### iv.- Elegimos el VPC

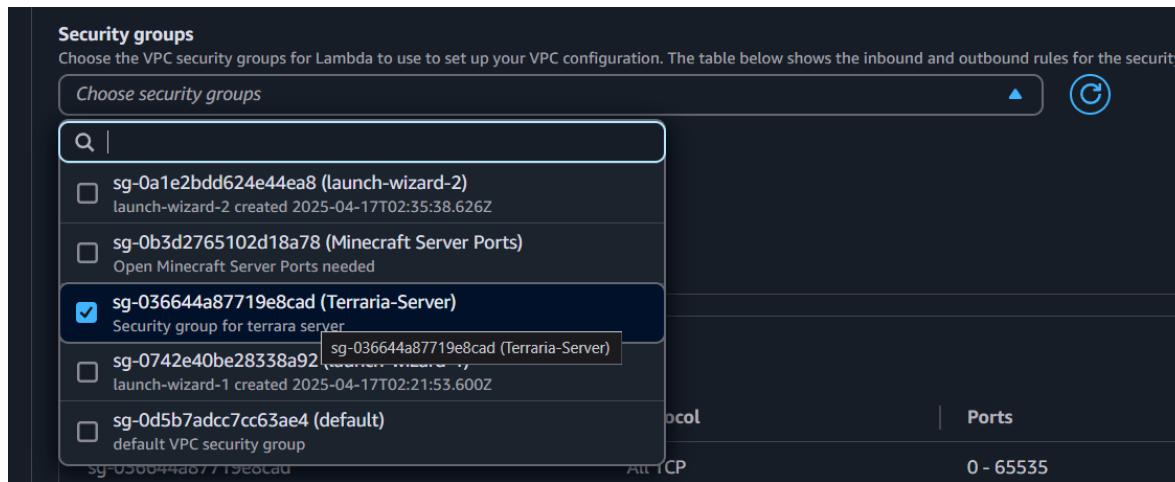


### v.- Agregamos la subnet privada del VPC



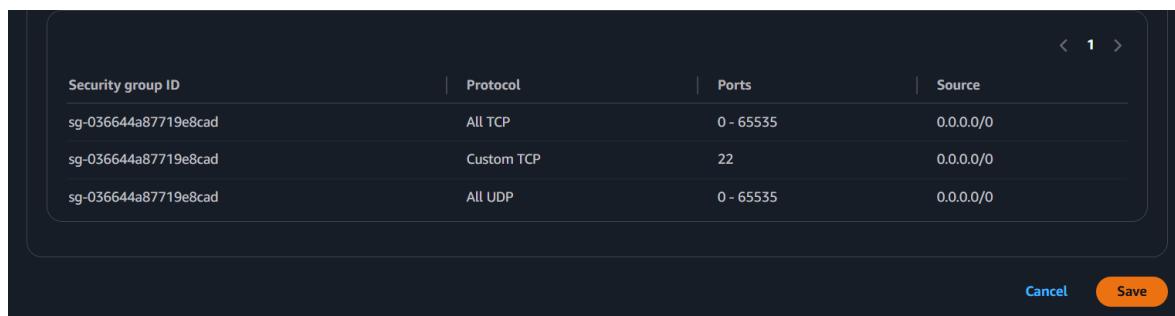
---

vi.- Elegir el security group establecido



| Protocol | Ports     |
|----------|-----------|
| All TCP  | 0 - 65535 |

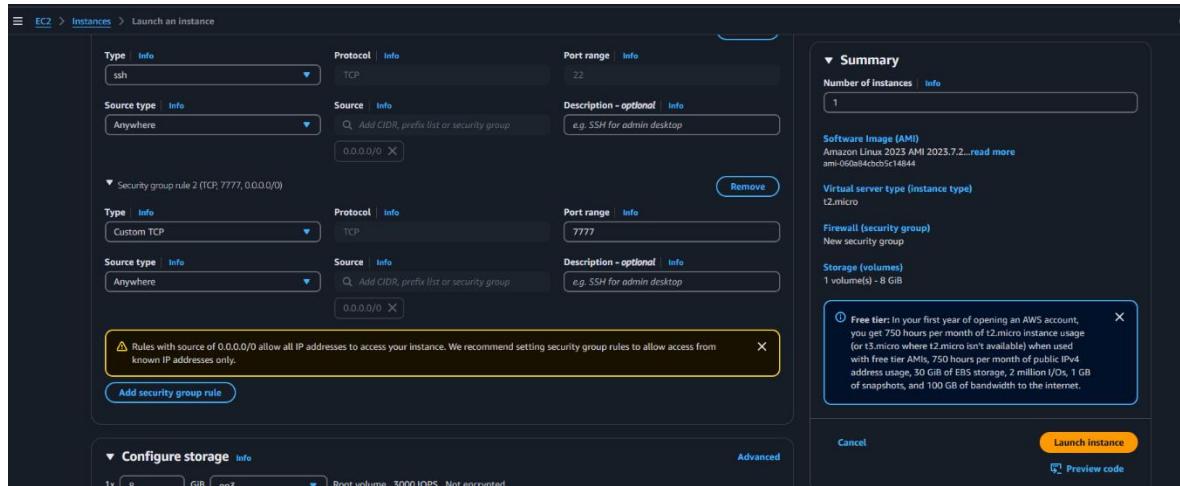
## vii.- Damos click en save



| Security group ID    | Protocol   | Ports     | Source    |
|----------------------|------------|-----------|-----------|
| sg-036644a87719e8cad | All TCP    | 0 - 65535 | 0.0.0.0/0 |
| sg-036644a87719e8cad | Custom TCP | 22        | 0.0.0.0/0 |
| sg-036644a87719e8cad | All UDP    | 0 - 65535 | 0.0.0.0/0 |

## AMAZON EC2:

- Vamos a levantar un EC2 t2.micro, con amazon Linux y en las reglas de seguridad, aparte de permitir ssh, vamos a permitir la entrada el puerto 7777 TCP (Puerto que se necesita para el servidor). Esta instnacia se pondra en el VPC creado anteriormente, en la subnet publica



- Script de creación y hacer que el servidor funcione, aunque se reinicie

```
descargamos los componentes necesarios
sudo yum update -y
sudo yum install -y unzip wget screen tmux libcurl

#descargamos el servidor y descomprimimos
wget https://terraria.org/api/download/pc-dedicated-server/terraria-server-1449.zip

unzip terraria-server-1449.zip

entramos al archivo descomprimido
cd 1449/Linux

#damos permisos de ejecucion a los archivos necesarios
chmod +x TerrariaServer.bin.x86_64

#Corremos y creamos un mundo
./TerrariaServer.bin.x86_64
```



ITESO

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Diplomado – Cloud Computing (Development)

Intel Partnership

```
#new World
n

#small
1

#classic
1

#Random
1

#Name
TerrariaEC2

#enter para llave random

#Cerramos servidor con CRTL+C
vamos a hacer un serverconfig file para que no tengamos que volver a hacer
el mundo cada vez que reiniciamos el servidor
nano serverconfig.txt

#codigo a escribir adentro
Nombre del mundo (usa el nombre exacto del archivo .wld)
world=/home/ec2-user/.local/share/Terraria/Worlds/TerrariaEC2.wld

guardamos el achivo y salimos de nano con CTRL+X, Y, ENTER

ahora vamos a generar un script para que levante el servidor automaticamente
cd /home/ec2-user/1449/Linux
nano startserver.sh

en si solamente corre el servidor pero con el archivo de configuracion que
hicimos antes
#!/bin/bash
cd /home/ec2-user/1449/Linux
export TERM=xterm
```



ITESO

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Diplomado – Cloud Computing (Development)

Intel Partnership

```
./TerrariaServer.bin.x86_64 -config /home/ec2-
user/1449/Linux/serverconfig.txt

guardamos el archivo y salimos de nano con CTRL+X, Y, ENTER

#damos permisos de ejecucion al script
chmod +x startserver.sh

ahora vamos a levantar el servicio para que el servidor se inicie
automaticamente al reiniciar la instancia
sudo nano /etc/systemd/system/terraria.service

lo que vas a escribir adentro

[Unit]
Description=Servidor de Terraria
After=network.target

[Service]
User=ec2-user
WorkingDirectory=/home/ec2-user/1449/Linux
ExecStart=/home/ec2-user/1449/Linux/startserver.sh
Restart=on-failure
RestartSec=10
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target

guardamos el archivo y salimos de nano con CTRL+X, Y, ENTER

#ahora agregamos el servicio a systemd para que lo reconozca y lo inicie al
reiniciar la instancia
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable terraria.service
sudo systemctl start terraria.service
```

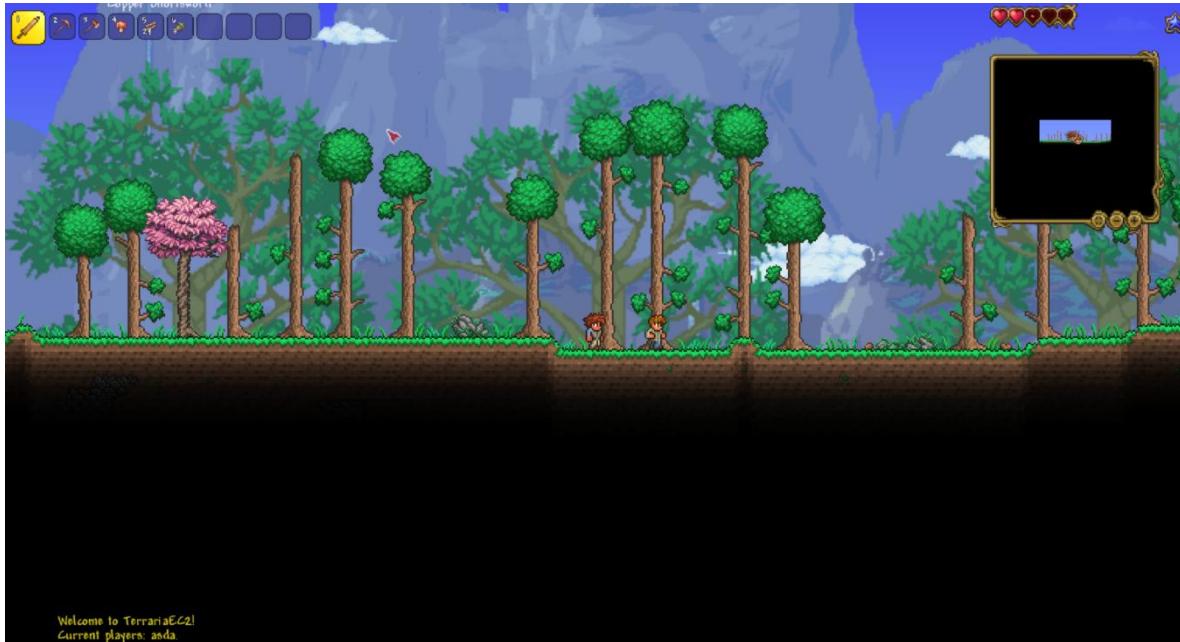
```
revisamos el estado del servicio
sudo systemctl status terraria.service
```

### 3. Revisamos el estatus

```
[root@ip-172-31-9-144 Linux]# sudo systemctl status terraria
● terraria.service - Servidor de Terraria
 Loaded: loaded (/etc/systemd/system/terraria.service; enabled; preset: disabled)
 Active: active (running) since Sat 2025-04-26 02:27:05 UTC; 5s ago
 Main PID: 4912 (startserver.sh)
 Tasks: 11 (limit: 1111)
 Memory: 514.3M
 CPU: 5.920s
 CGroup: /system.slice/terraria.service
 └─4912 /bin/bash /home/ec2-user/1449/Linux/startserver.sh
 ├─4914 ./TerrariaServer.bin.x86_64 -config /home/ec2-user/1449/Linux/serverconfig.txt

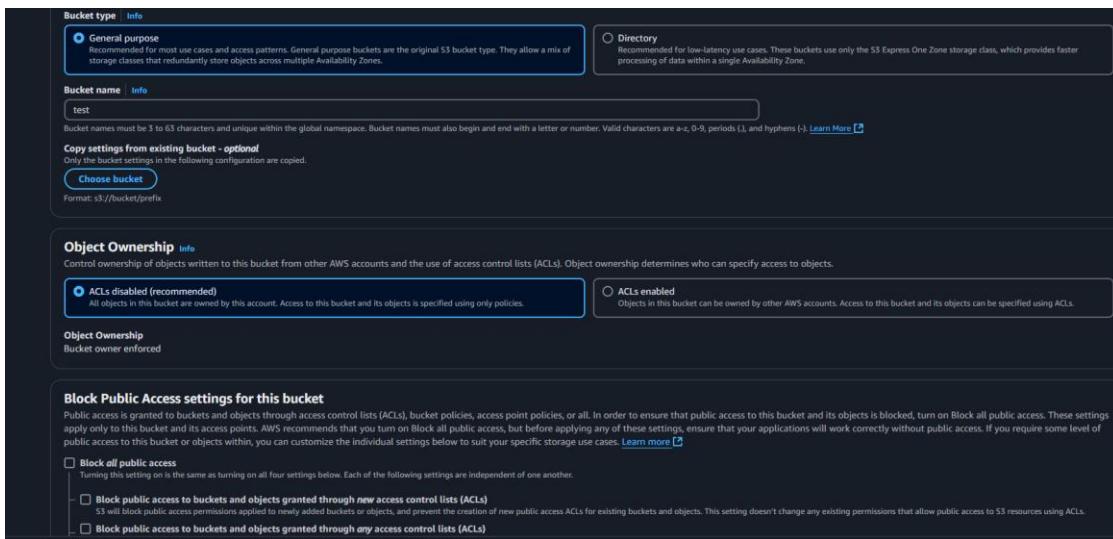
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Settling liquids 39%
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Settling liquids 40%
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Settling liquids 41%
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Settling liquids 44%
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Settling liquids 48%
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Settling liquids 50%
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Terraria Server v1.4.4.9
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Listening on port 7777
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: Type 'help' for a list of commands.
Apr 26 02:27:11 ip-172-31-9-144.us-east-2.compute.internal startserver.sh[4914]: : Server started
```

### 4. Ahora podemos entrar al servidor sin problemas



**AMAZON S3:**

- 
1. Generamos un bucket eliminando la prohibición a TODO acceso público de esta bucket, ya que va a ser referenciada por Cloud Front



**Bucket type** | [Info](#)

General purpose Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

**Bucket name** | [Info](#)

test

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn More](#)

**Copy settings from existing bucket - optional**  
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: `es3://bucket/prefix`

**Object Ownership** | [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended) All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

**Object Ownership**  
Bucket owner enforced

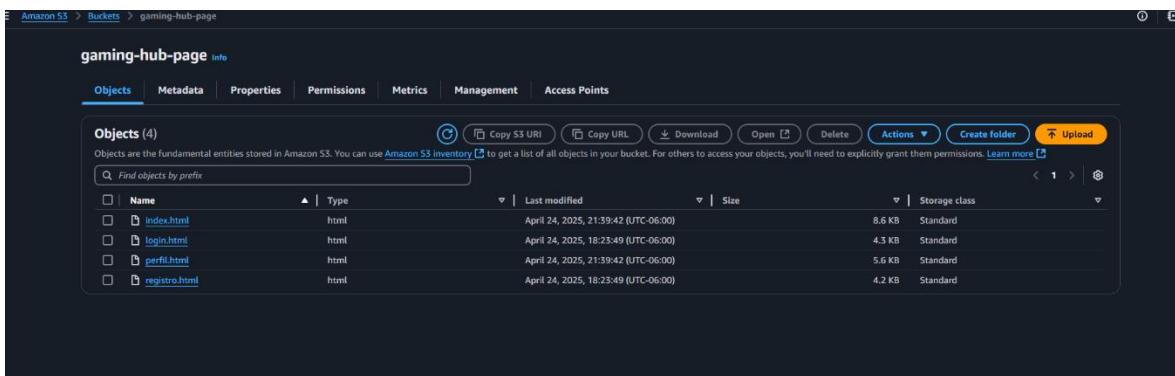
**Block Public Access settings for this bucket**

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

**Block all public access** Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)** S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**

2. Metemos los archivos html al bucket



Amazon S3 > Buckets > gaming-hub-page | [Info](#)

**gaming-hub-page** | [Info](#)

[Objects](#) | [Metadata](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

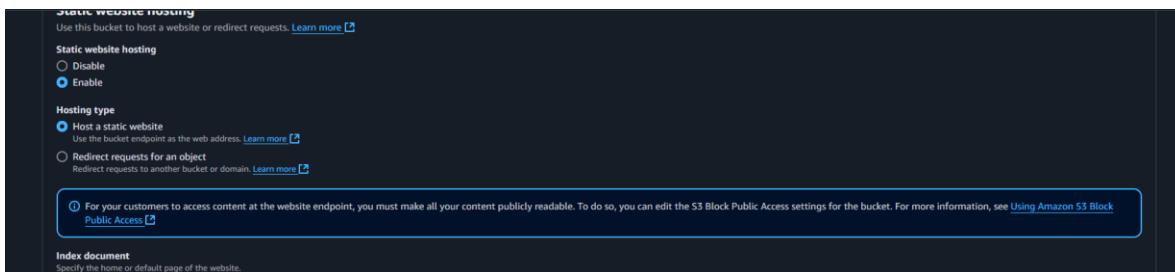
**Objects (4)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

| Name                          | Type | Last modified                        | Size   | Storage class |
|-------------------------------|------|--------------------------------------|--------|---------------|
| <a href="#">index.html</a>    | html | April 24, 2025, 21:39:42 (UTC-06:00) | 8.6 KB | Standard      |
| <a href="#">login.html</a>    | html | April 24, 2025, 18:23:49 (UTC-06:00) | 4.3 KB | Standard      |
| <a href="#">perfil.html</a>   | html | April 24, 2025, 21:39:42 (UTC-06:00) | 5.6 KB | Standard      |
| <a href="#">registro.html</a> | html | April 24, 2025, 18:23:49 (UTC-06:00) | 4.2 KB | Standard      |

3. Nos metemos a properties y configuramos el static web hosting para el EC2



**Static Website Hosting**

Use this bucket to host a website or redirect requests. [Learn more](#)

**Static website hosting**

Disable

Enable

**Hosting type**

Host a static website Use the bucket endpoint as the web address. [Learn more](#)

Redirect requests for an object Redirect requests to another bucket or domain. [Learn more](#)

**For your customers to access content at the website endpoint, you must make all your content publicly readable.** To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

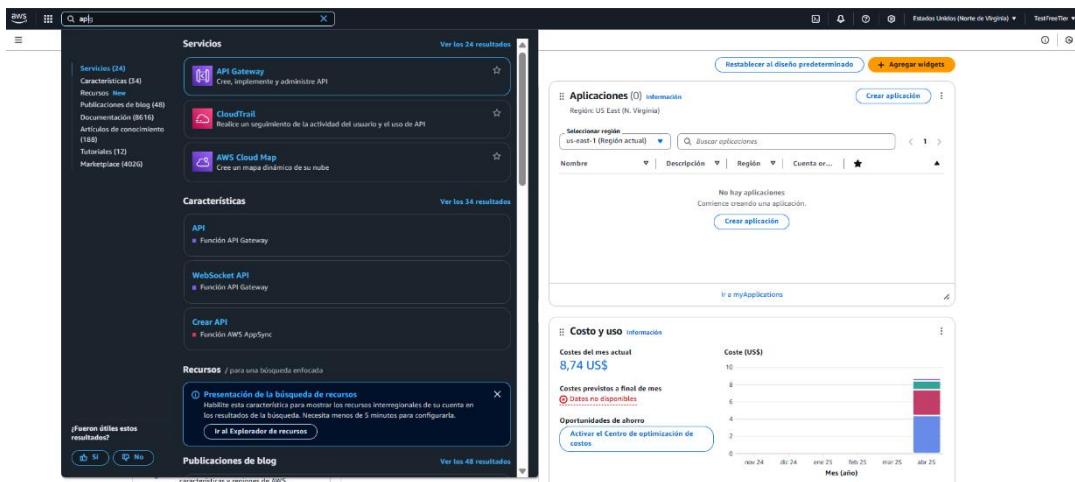
**Index document**

Specify the home or default page of the website.

**API GATEWAY:**

## 1. Crear API Gateway

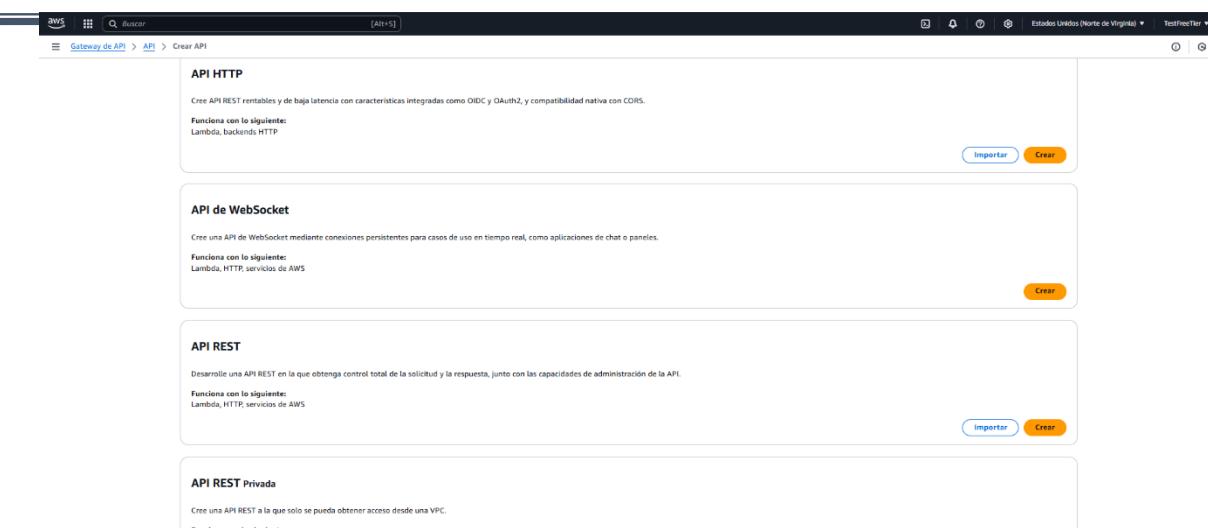
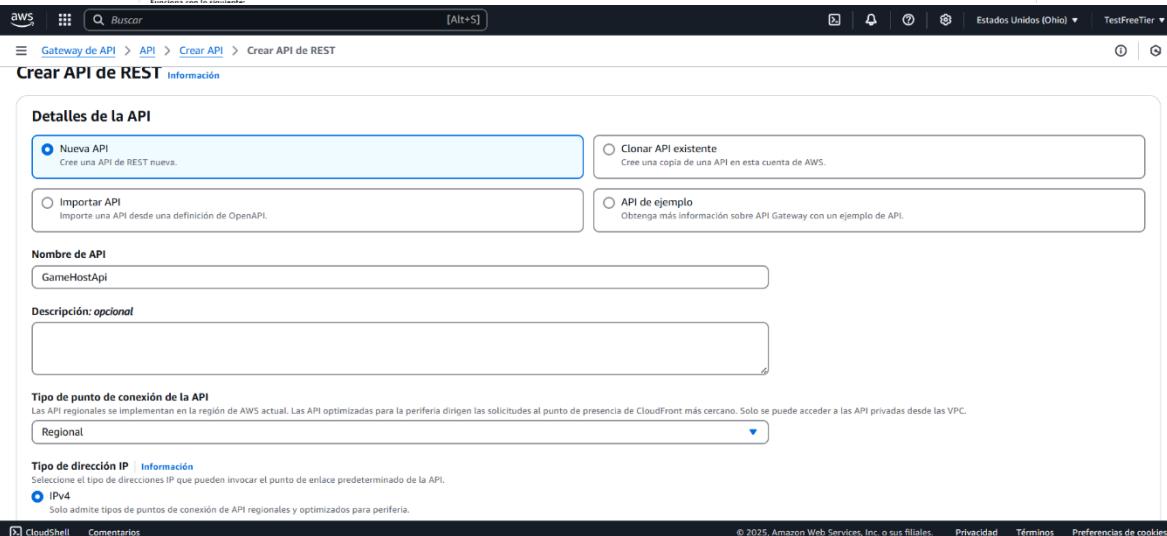
- En la barra de búsqueda de AWS buscar API Gateway.



- Entra a **API Gateway**, selecciona **Crear API**.

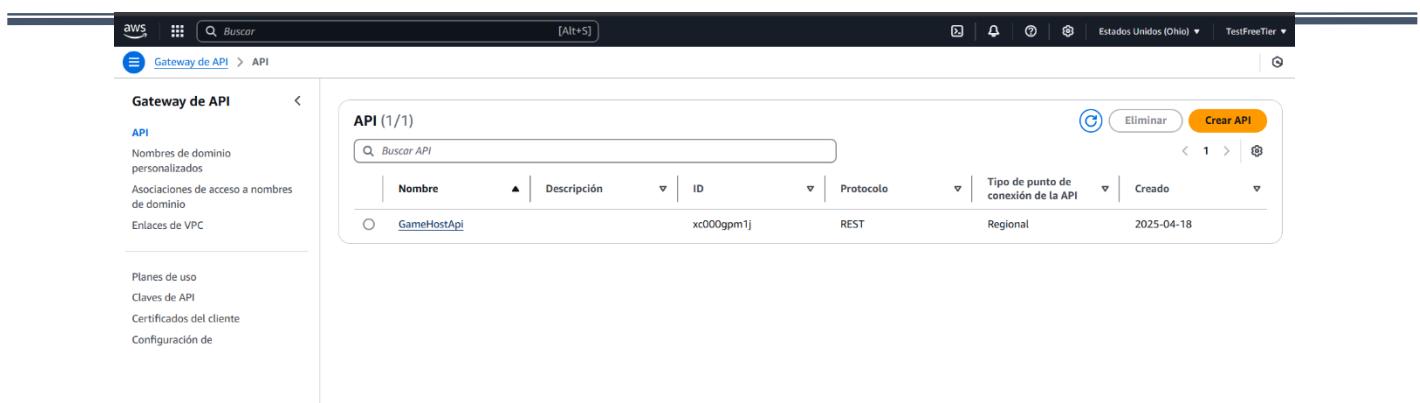


Elegir **REST API pública**, Poner en nombre: GameHostApi, tipo Regional, y crea la API.

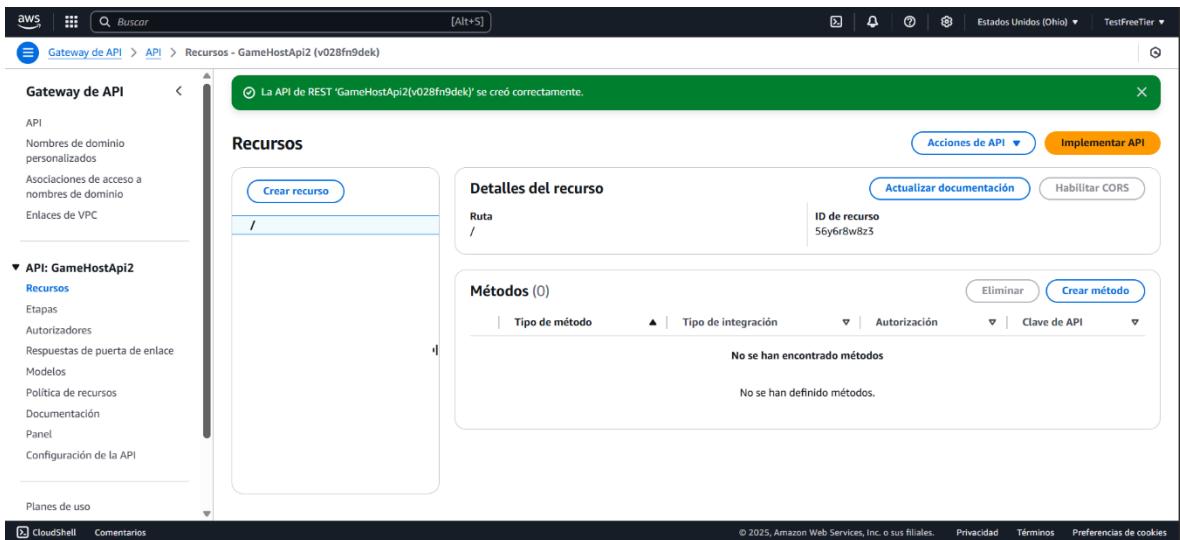
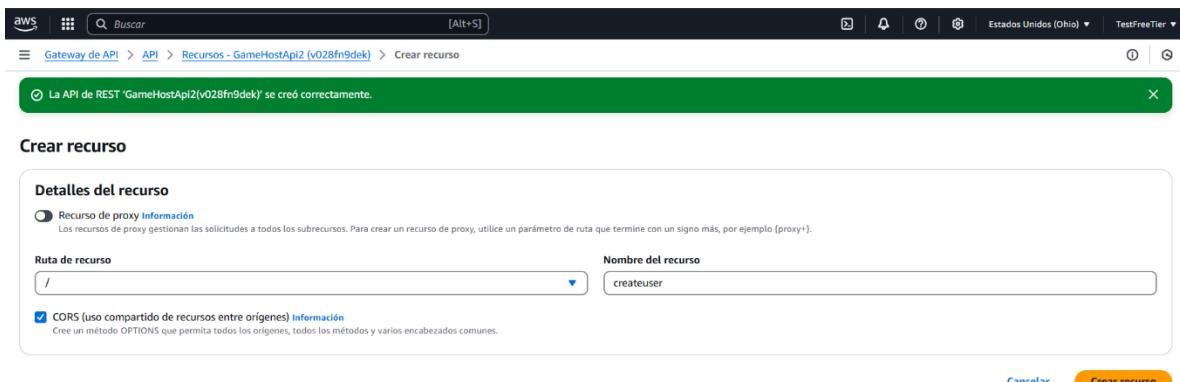

  


## 2. Crear Recursos y Métodos

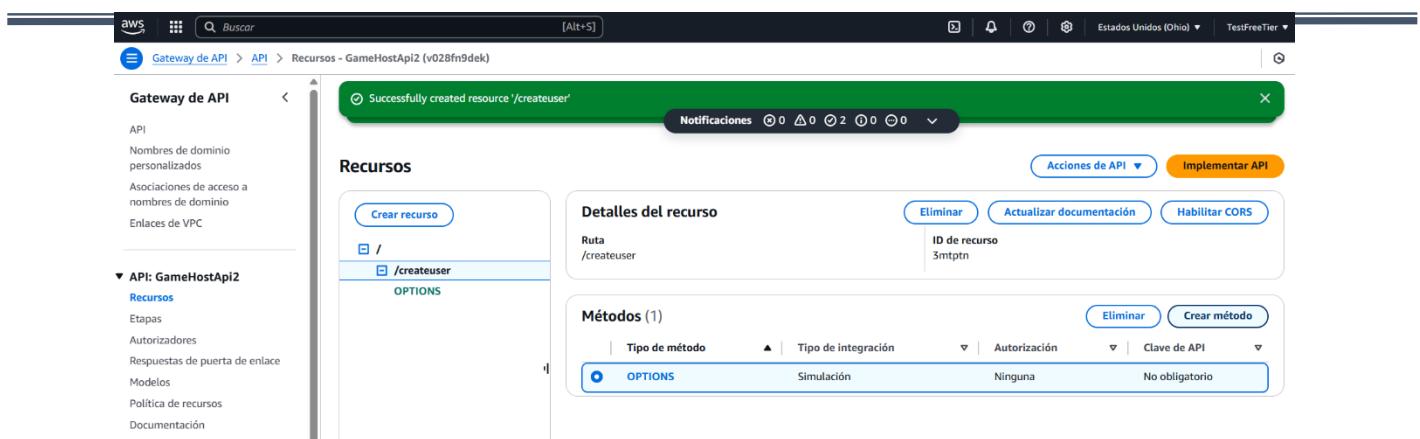
- En nuestra API GameHostApi, dar clic derecho en para configurarla, podremos crear recursos y añadir configuraciones adicionales.



- Damos click en **Crear recurso** (le asignamos el nombre /createuser), habilita **CORS** si sale la opción, y crea.

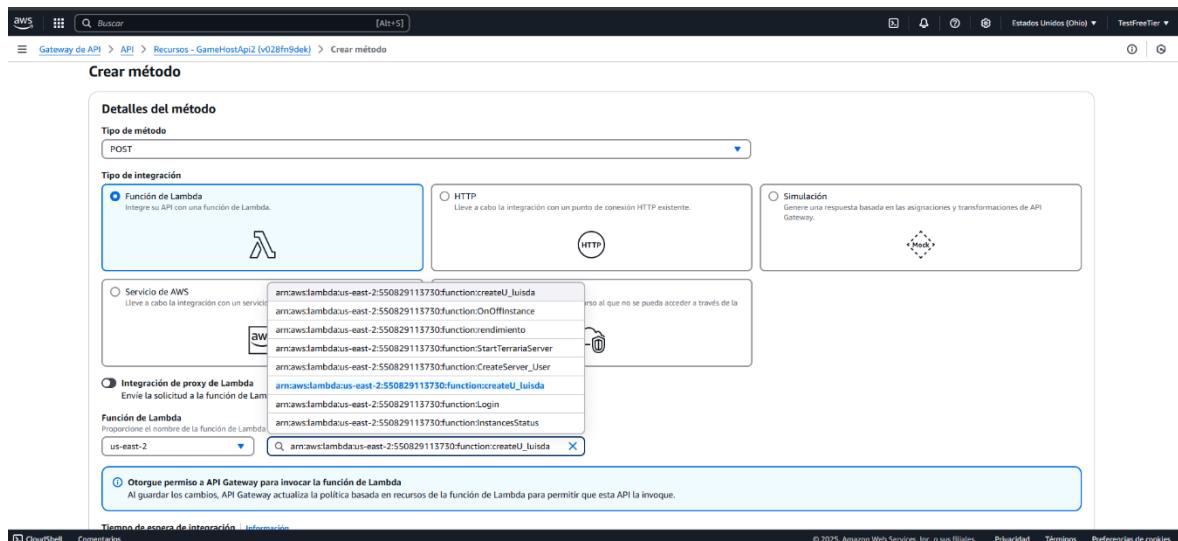



- Luego en /createuser, damos click en Crear método en el apartado **Métodos**.



The screenshot shows the AWS API Gateway interface. On the left, there's a sidebar for 'Gateway de API' and 'API'. Under 'API', it lists 'Nombres de dominio personalizados', 'Asociaciones de acceso a nombres de dominio', and 'Enlaces de VPC'. Below that is a section for 'API: GameHostApi2' with 'Recursos', 'Etapas', 'Autorizadores', 'Respuestas de puerta de enlace', 'Modelos', 'Política de recursos', and 'Documentación'. The main area is titled 'Recursos' and shows a list with a single item: '/createuser'. To the right, there's a 'Detalles del recurso' panel with 'Ruta /createuser' and an 'ID de recurso' of '3mptn'. Below it is a 'Métodos (1)' panel showing an 'OPTIONS' method with 'Simulación' and 'Ninguna' for 'Autorización'.

- Creamos un método de tipo post, lo integramos con Función de Lambda y especificamos la función de lambda para crear usuario, en este caso CreateU\_Luisda. Aquí por como manejamos nuestra función lambda, no usamos Lambda Proxy porque mandaremos el JSON directo y sin envoltorio, lo mismo en login.

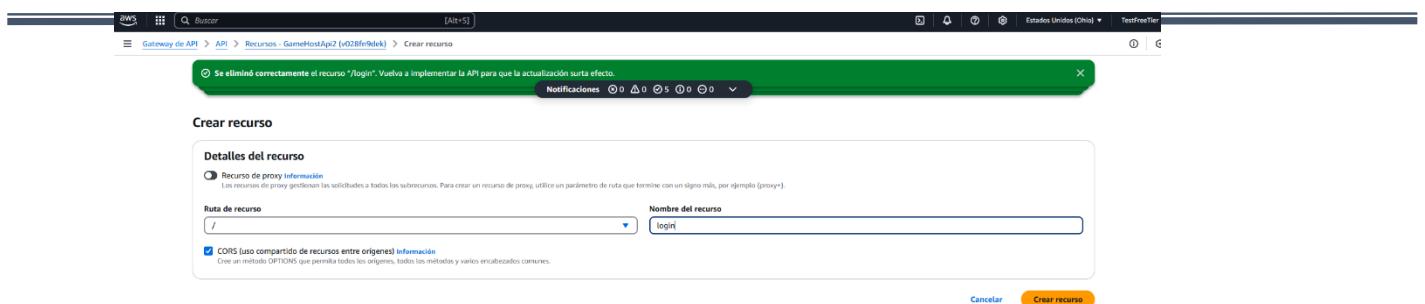


The screenshot shows the 'Crear método' (Create Method) dialog in AWS API Gateway. It has a 'Detalles del método' section with 'Tipo de método' set to 'POST'. Below it is a 'Tipo de integración' section with three options: 'Función de Lambda' (selected), 'HTTP', and 'Simulación'. The 'Función de Lambda' section shows a dropdown menu with several Lambda function names, and a search bar at the bottom containing 'arn:aws:lambda:us-east-2:2550829113730:function:createU\_luisda'. At the bottom of the dialog, there's a checkbox for 'Otorgue permiso a API Gateway para invocar la función de Lambda' (Grant permission to API Gateway to invoke the Lambda function).

- **Repite** este proceso para:
  - /login → vinculado a Login
  - /onoff → vinculado a OnOffInstance
  - /createserver → vinculado a CreateServer\_User
  - /instancestatus → vinculado a InstancesStatus.

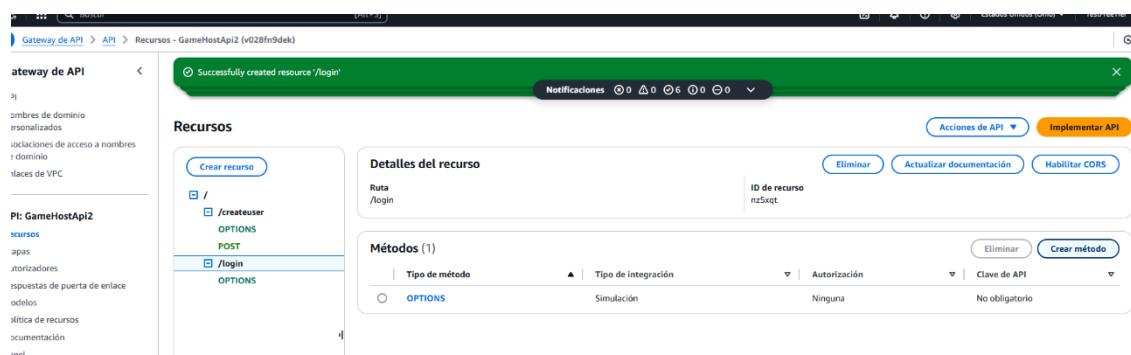
*(todos usando POST y Lambda Proxy menos en login)*

- Proceso para login: Creamos recurso /login habilitando CORS.



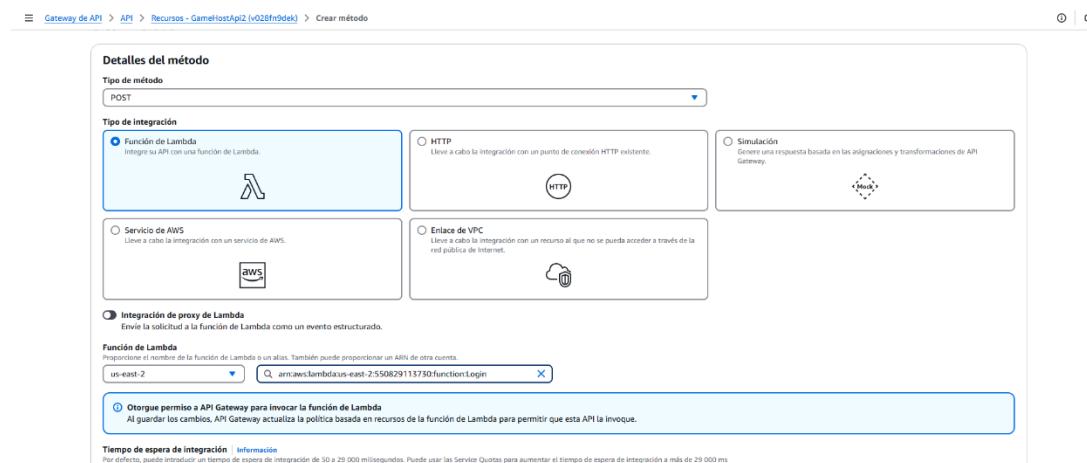
The screenshot shows the AWS API Gateway interface. A success message at the top says: "Se creó correctamente el recurso 'login'. Vuelva a implementar la API para que la actualización surta efecto." Below it, a form for creating a new resource is displayed. The 'Nombre del recurso' field is filled with 'login'. The 'Ruta de recurso' dropdown shows '/'. There is a checked checkbox for 'CORS (uso compartido de recursos entre orígenes)'. At the bottom right are 'Cancelar' and 'Crear recurso' buttons.

- Damos Click en crear método para login.



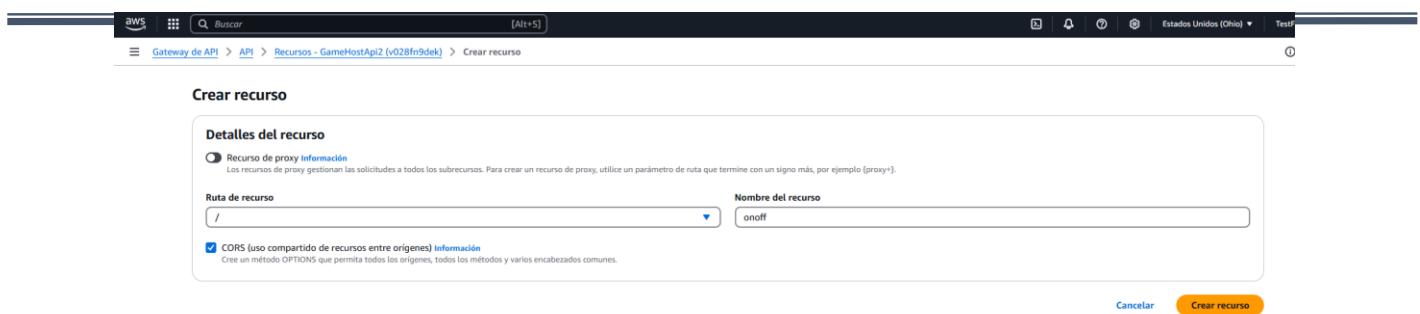
The screenshot shows the AWS API Gateway interface with the newly created 'login' resource listed under 'Recursos'. A success message at the top says: "Successfully created resource '/login'". The 'Métodos' section shows a single POST method. The 'Tipo de integración' dropdown is set to 'Función de Lambda'. The 'Función de Lambda' input field contains 'arn:aws:lambda:us-east-2:550629113730:function:Login'.

- Tipo Post, integración con Lambda y lo ligamos a la función Login, creamos método.



The screenshot shows the configuration of the POST method for the '/login' resource. In the 'Detalles del método' section, the 'Tipo de método' is set to 'POST' and the 'Tipo de integración' is set to 'Función de Lambda'. The 'Función de Lambda' input field contains 'arn:aws:lambda:us-east-2:550629113730:function:Login'. A note below says: "Al guardar los cambios, API Gateway actualiza la política basada en recursos de la función de Lambda para permitir que esta API la invoque." The 'Tiempo de espera de integración' field is set to 200 ms.

- Proceso para onoff: Creamos recurso /onoff habilitando CORS.



**Crear recurso**

**Detalles del recurso**

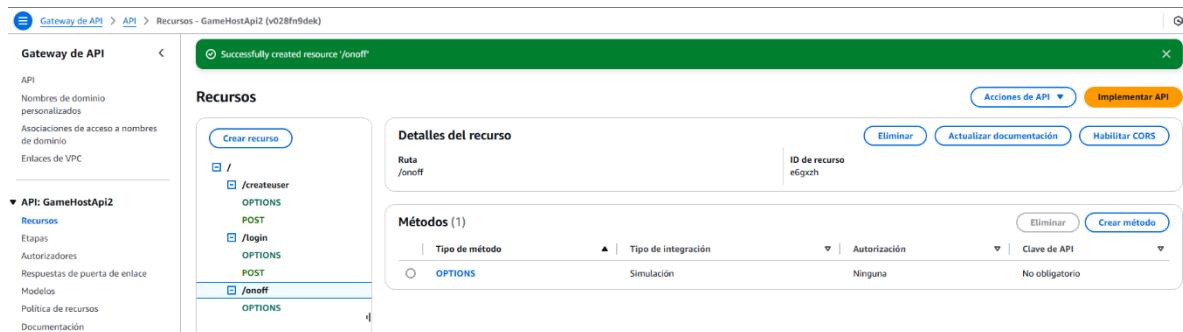
Recurso de proxy [Información](#)  
Los recursos de proxy gestionan las solicitudes a todos los subrecursos. Para crear un recurso de proxy, utilice un parámetro de ruta que termine con un signo más, por ejemplo [proxy+].

Ruta de recurso: / Nombre del recurso: onoff

CORS (uso compartido de recursos entre orígenes) [Información](#)  
Cree un método OPTIONS que permita todos los orígenes, todos los métodos y varios encabezados comunes.

[Cancelar](#) [Crear recurso](#)

- Damos Click en crear método para onoff.



**Gateway de API** > API > Recursos - GameHostApi2 (v028fn9dek)

**Recursos**

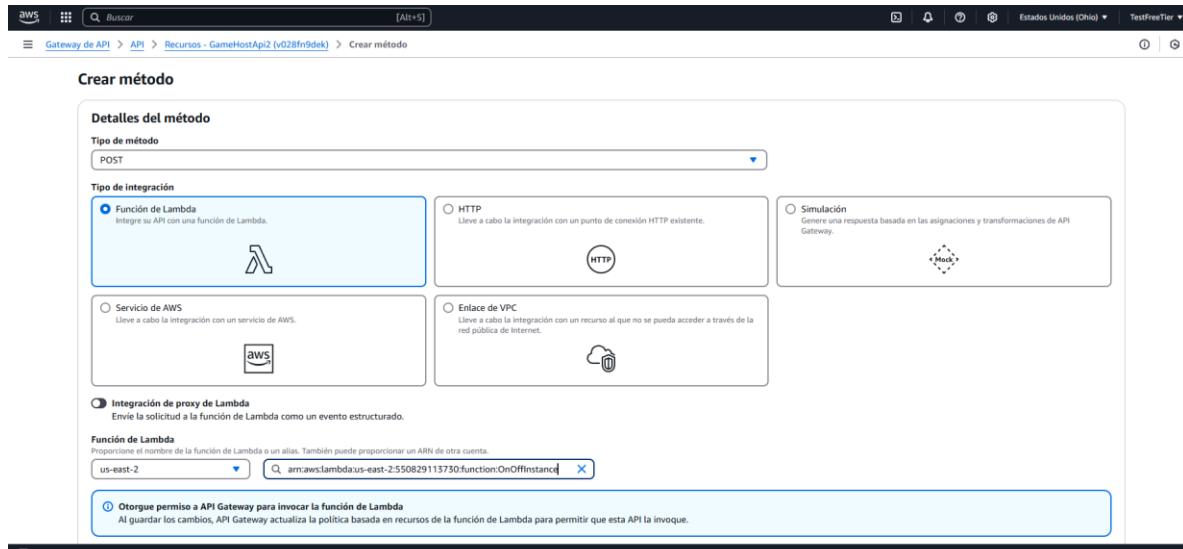
**Detalles del recurso**

Ruta: /onoff

**Métodos (1)**

| Tipo de método | Tipo de integración | Autorización | Clave de API   |
|----------------|---------------------|--------------|----------------|
| OPTIONS        | Simulación          | Ninguna      | No obligatorio |

- Tipo Post, integración con Lambda, activamos integración de proxy de Lambda y lo ligamos a la función OnOffInstance, creamos método.



**Crear método**

**Detalles del método**

**Tipo de método**: POST

**Tipo de integración**

- Función de Lambda: Integre su API con una función de Lambda. 
- HTTP: Lleve a cabo la integración con un punto de conexión HTTP existente. 
- Simulación: Genera una respuesta basada en las asignaciones y transformaciones de API Gateway. 

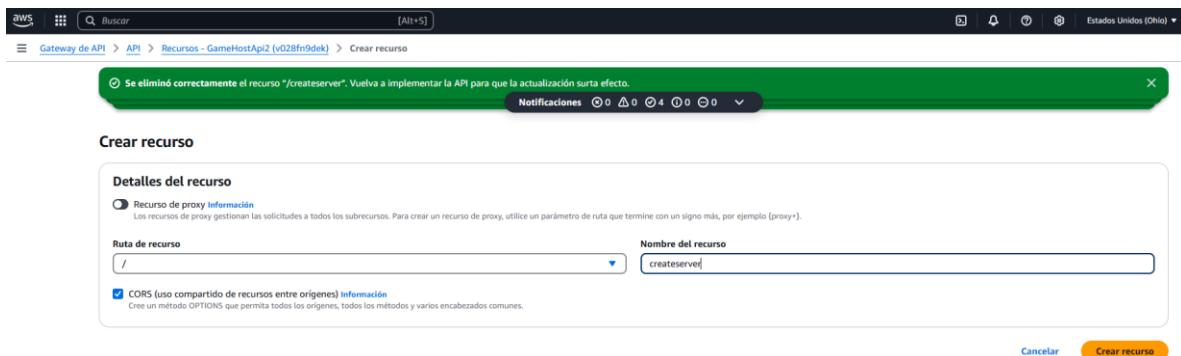
**Integración de proxy de Lambda**: Envíe la solicitud a la función de Lambda como un evento estructurado.

Proporcione el nombre de la función de Lambda o un alias. También puede proporcionar un ARN de otra cuenta.  
us-east-2 Q arn:aws:lambda:us-east-2:550829115730:function:OnOffInstance X

**Otorgue permiso a API Gateway para invocar la función de Lambda**: Al guardar los cambios, API Gateway actualiza la política basada en recursos de la función de Lambda para permitir que esta API la invoque.

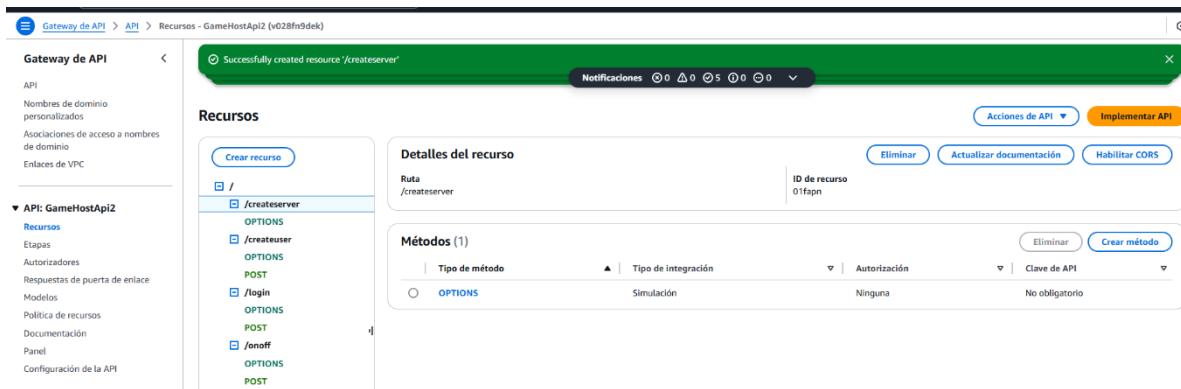
**Integración de proxy de Lambda**  
Envíe la solicitud a la función de Lambda como un evento estructurado.

- Proceso para Create Server: Creamos recurso `/createserver` habilitando CORS.



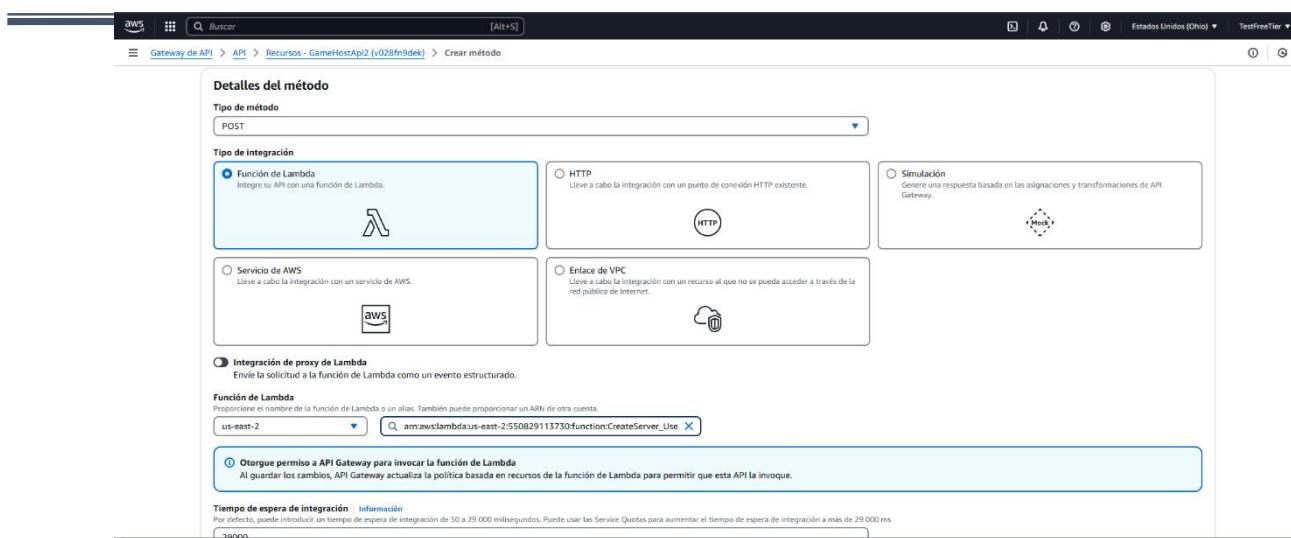
The screenshot shows the AWS API Gateway interface. In the top navigation bar, it says 'Gateway de API > API > Recursos - GameHostApi2 (v028fn9dek) > Crear recurso'. A green notification bar at the top indicates that the resource was successfully deleted: 'Se eliminó correctamente el recurso "/createserver": Vuelva a implementar la API para que la actualización surta efecto.' Below this, the 'Crear recurso' dialog is open, showing the 'Detalles del recurso' section. It includes fields for 'Nombre del recurso' (set to 'createserver') and 'Ruta de recurso' (set to '/'). There is also a checkbox for 'CORS (uso compartido de recursos entre orígenes)' which is checked. At the bottom right of the dialog is a yellow 'Crear recurso' button.

- Damos Click en crear método para createserver.



The screenshot shows the AWS API Gateway 'Resources' list. On the left, there's a sidebar with 'Gateway de API' and 'API: GameHostApi2' sections. Under 'API: GameHostApi2', there are several resources listed: '/createuser', '/login', '/onoff', and '/createserver'. The '/createserver' resource is expanded, showing its details: 'Ruta /createserver' and 'ID de recurso 01fapn'. Below this, the 'Métodos (1)' table is shown, listing a single 'OPTIONS' method for the '/createserver' endpoint. The table includes columns for 'Tipo de método', 'Tipo de integración', 'Autorización', and 'Clave de API'. The 'Tipo de integración' is listed as 'Simulación' and 'Autorización' as 'Ninguna'. A green success message at the top of the main content area says 'Successfully created resource "/createserver"'.

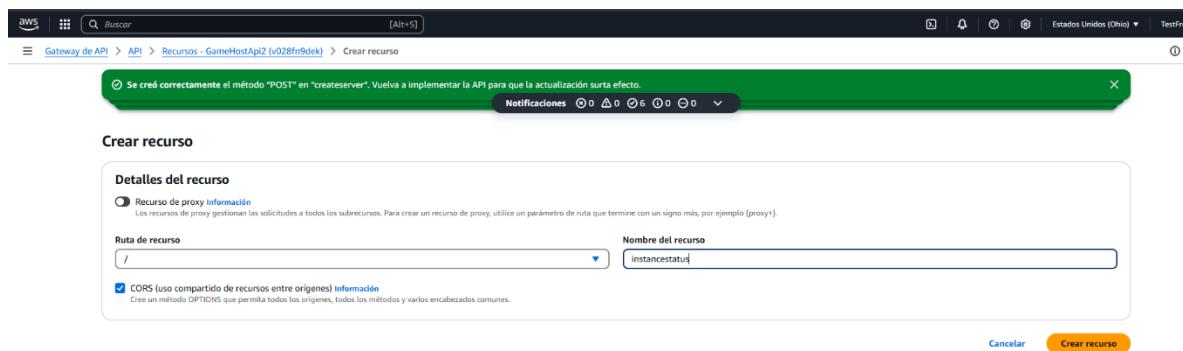
- Tipo Post, integración con Lambda, activamos integración de proxy de Lambda y lo ligamos a la función CreateServer\_User, creamos el método.



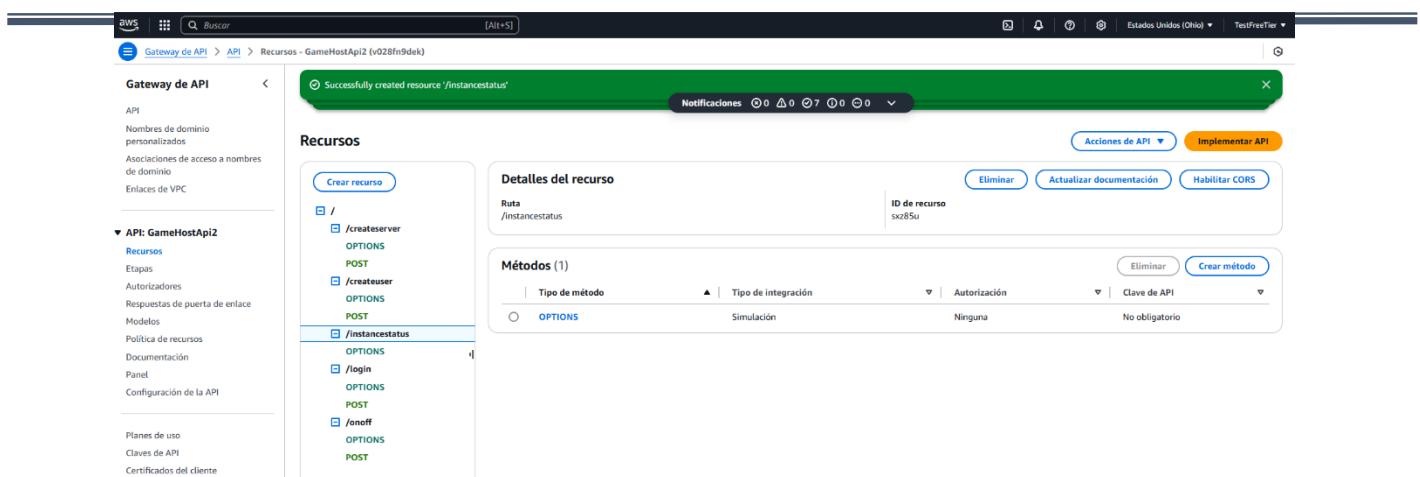
**Integración de proxy de Lambda**

Envíe la solicitud a la función de Lambda como un evento estructurado.

- Proceso para Instance Status: Creamos recurso /instancestatus habilitando CORS.

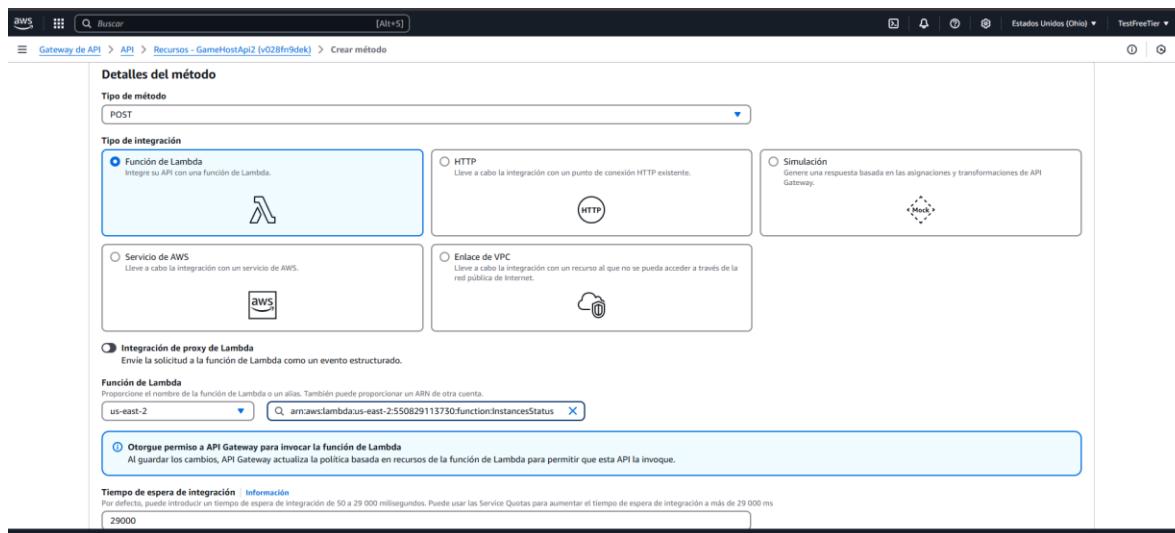


- Damos Click en crear método para instancestatus.



The screenshot shows the AWS API Gateway interface. In the left sidebar, under the 'API' section, there is a tree view of resources. One node under 'GameHostApi2' is expanded, showing methods for '/createserver', '/createuser', and '/instancestatus'. The '/instancestatus' node has four methods listed: 'OPTIONS', 'POST', 'OPTIONS', and 'POST'. On the right, a detailed view of the '/instancestatus' resource is shown. The 'Detalles del recurso' (Resource Details) panel shows the path '/instancestatus' and an ID 'sx285u'. The 'Métodos' (Methods) panel lists the four methods with their respective types ('OPTIONS', 'POST') and integration types ('Simulación', 'Ninguna'). Buttons for 'Eliminar' (Delete), 'Actualizar documentación' (Update Documentation), and 'Habilitar CORS' (Enable CORS) are visible.

- Tipo Post, integración con Lambda, activamos integración de proxy de Lambda y lo ligamos a la función InstancesStatus, creamos método.



The screenshot shows the 'Create method' dialog for the '/instancestatus' resource. Under 'Tipo de método' (Method Type), 'POST' is selected. Under 'Tipo de integración' (Integration Type), 'Función de Lambda' (Lambda Function) is selected. Below it, 'Integración de proxy de Lambda' (Proxy Integration) is also selected. A note says 'Envíe la solicitud a la función de Lambda como un evento estructurado.' (Send the request to the Lambda function as a structured event.) At the bottom, a note says 'Al otorgar permiso a API Gateway para invocar la función de Lambda' (When granting permission to API Gateway to invoke the Lambda function) and 'Al guardar los cambios, API Gateway actualiza la política basada en recursos de la función de Lambda para permitir que esta API la invoque.' (When you save changes, API Gateway updates the Lambda function's resource-based policy to allow this API to invoke it.) A note at the bottom left says 'Tiempo de espera de integración' (Integration timeout) with a value of '29000'.

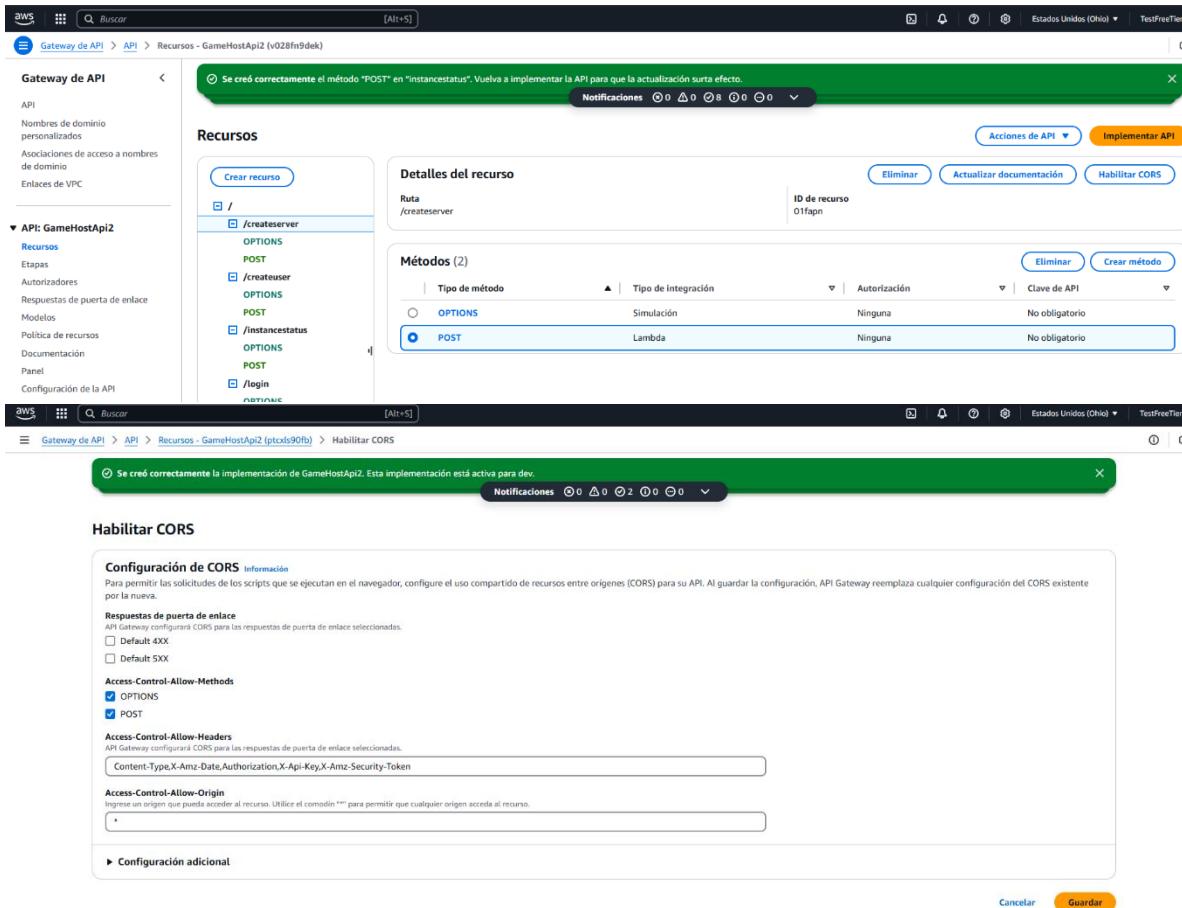
## Integración de proxy de Lambda

Envíe la solicitud a la función de Lambda como un evento estructurado.

### 3. Habilitar CORS en cada recurso

- En cada recurso (/instancestatus, /createserver, /login, /onoff, /createuser):
  - Selecciona el POST, luego **Detalles del recurso > Habilitar CORS**. (Enable CORS).
  - Configura:
    - Allow-Headers: Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token
    - Allow-Methods: OPTIONS, POST

- Allow-Origin: \*
- Aplica los cambios.



**Recursos**

**Detalles del recurso**

**Métodos (2)**

**Habilitar CORS**

**Configuración de CORS**

**Resuestas de puerta de enlace**

**Access-Control-Allow-Methods**

**Access-Control-Allow-Headers**

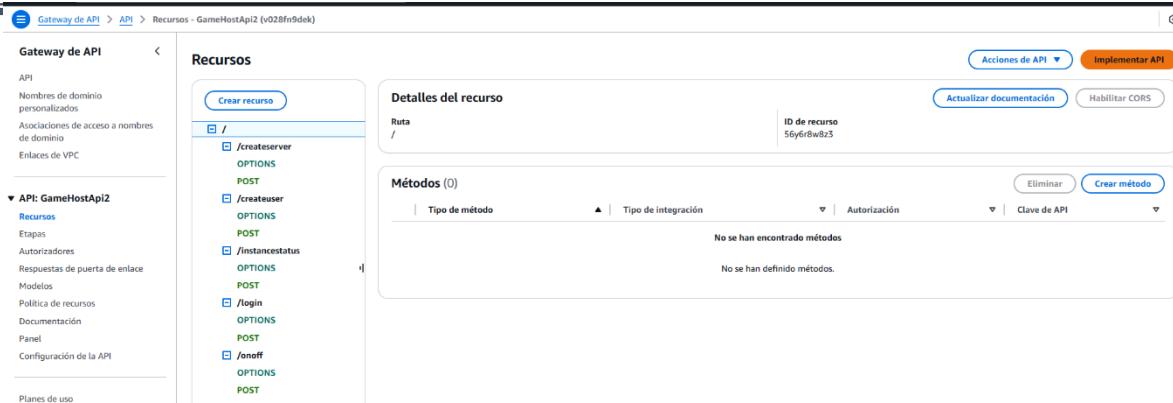
**Access-Control-Allow-Origin**

**Configuración adicional**

*Repetimos el proceso para cada recurso como se mencionó previamente.*

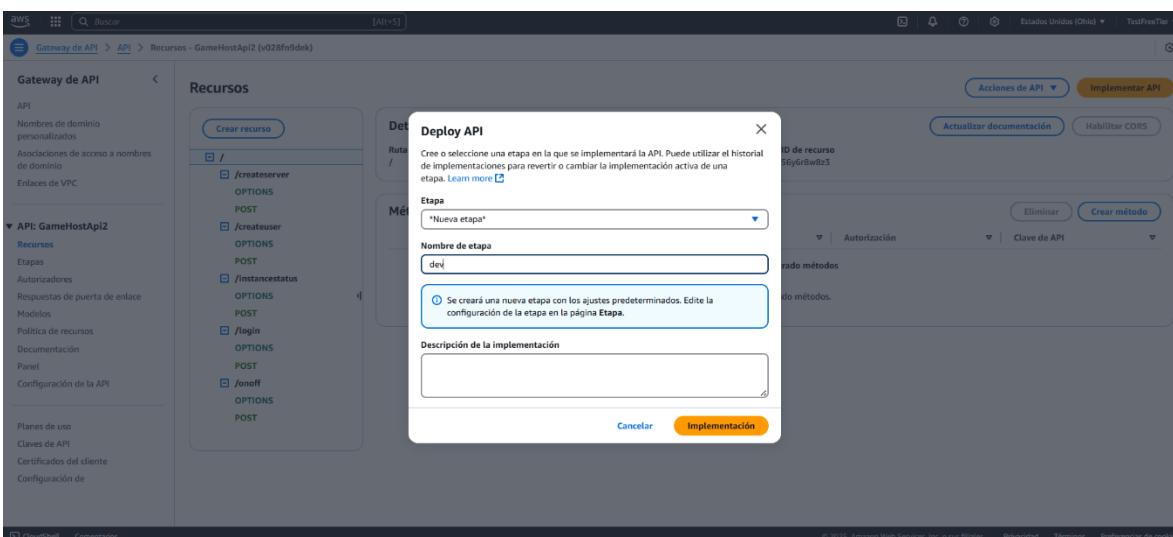
#### 4. Deploy del API

- En GameHostoApi, ve a **Acciones > Implementar API** (el botón amarillo en la esquina superior derecha).

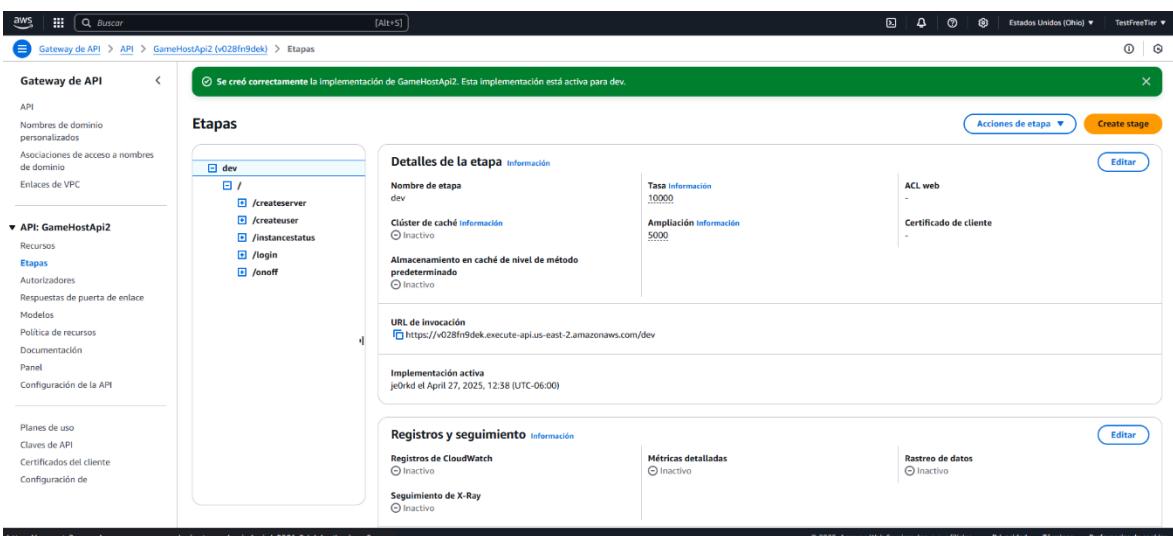


The screenshot shows the AWS API Gateway console. On the left, the navigation sidebar includes sections for API, Recursos, and API: GameHostApi2. Under Recursos, there are links for Etapas, Autorizadores, Respuestas de puerta de enlace, Modelos, Política de recursos, Documentación, Panel, and Configuración de la API. The main content area displays the 'RECURSOS' section for GameHostApi2. It lists several resources: /createuser (OPTIONS, POST), /login (OPTIONS, POST), /onoff (OPTIONS, POST), /createserver (OPTIONS, POST), and /instancestatus (OPTIONS, POST). The 'Detalles del recurso' panel shows the resource path '/' and its ID '56y6rwBz3'. The 'Métodos' panel indicates that no methods have been defined.

- Crea un nuevo stage o etapa (por ejemplo dev en este caso), y despliega.



The screenshot shows the 'Deploy API' dialog box. In the 'Etapa' field, 'Nueva etapa' is selected and labeled 'dev'. A note says: 'Se creará una nueva etapa con los ajustes predeterminados. Edite la configuración de la etapa en la página Etapa.' The 'Implementación' button is highlighted.



The screenshot shows the 'Etapas' section for GameHostApi2. The 'dev' stage is listed under the '/' resource. The 'Detalles de la etapa' panel shows the stage name 'dev', cache clustering status (Inactivo), and URL of invocation 'https://v028fn9dek.execute-api.us-east-2.amazonaws.com/dev'. The 'Registros y seguimiento' panel shows CloudWatch logs (Inactivo) and X-Ray tracing (Inactivo). The 'Rastreo de datos' section is also present.

---

## 5. Implementación

- En GameHostoApi, ve a **Etapas o Stages**> damos click al método que queramos invocar, copiamos la URL de invocación, y ya podemos manejarla y enviarle solicitudes de tipo POST desde nuestro script de JS.
- Para el login copiamos el URL de invocación de /login POST, y lo ponemos en el siguiente FETCH en nuestro login.html.

```

120
121 // Post para registrar usuario
122 try {
123 const response = await fetch('https://xc000gpm1j.execute-api.us-east-2.amazonaws.com/Dev/login', {
124 method: 'POST',
125 headers: {
126 'Content-Type': 'application/json'
127 },
128 body: JSON.stringify({
129 User: { email, password }
130 })
131 });

```

Código

completo de login (HTML, CSS) con Script JS para manejar las solicitudes y respuestas:

```

<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <title>GameHost - Iniciar Sesión</title>
 <link
 href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap"
 rel="stylesheet">
 <style>
 body {
 background-color: #0f0f0f;
 color: #00ffcc;
 font-family: 'Press Start 2P', cursive;
 display: flex;
 justify-content: center;
 align-items: center;
 height: 100vh;
 margin: 0;
 }

 .login-container {
 background-color: #1a1a1a;
 border: 2px solid #00ffcc;
 border-radius: 12px;
 padding: 40px;
 }
 </style>
</head>
<body>
 <div class="login-container">
 <form>
 <input type="text" placeholder="Email" />
 <input type="password" placeholder="Contraseña" />
 <button type="submit">Iniciar Sesión</button>
 </form>
 </div>
</body>

```

```
width: 400px;
box-shadow: 0 0 15px #00ffcc;
animation: fadeIn 1s ease-in-out;
}

.login-container h2 {
 text-align: center;
 color: #fff;
 margin-bottom: 30px;
}

label {
 display: block;
 margin: 15px 0 5px;
 font-size: 10px;
 color: #00ffff;
}

input[type="text"],
input[type="password"] {
 width: 100%;
 padding: 12px;
 background-color: #0f0f0f;
 border: 2px solid #00ffcc;
 border-radius: 5px;
 color: #00ffcc;
 font-family: 'Press Start 2P', cursive;
 font-size: 10px;
 box-sizing: border-box;
}

.btn {
 margin-top: 25px;
 width: 100%;
 padding: 15px;
 background-color: #00ffcc;
 color: #000;
 border: none;
 font-size: 12px;
 border-radius: 5px;
 cursor: pointer;
 transition: transform 0.2s, background 0.3s;
```

```

 }

 .btn:hover {
 background-color: #00ffff;
 transform: scale(1.05);
 }

 @keyframes fadeIn {
 from { opacity: 0; transform: scale(0.9); }
 to { opacity: 1; transform: scale(1); }
 }

 .back-link {
 display: block;
 text-align: center;
 margin-top: 20px;
 font-size: 10px;
 }

 .back-link a {
 color: #00ffff;
 text-decoration: none;
 }

 .back-link a:hover {
 text-decoration: underline;
 }
 </style>
</head>
<body>
 <div class="login-container">
 <h2>👤 Iniciar Sesión</h2>
 <form id="login-form">
 <label for="username">Usuario</label>
 <input type="text" id="username" name="username" required>

 <label for="password">Contraseña</label>
 <input type="password" id="password" name="password" required>

 <button type="submit" class="btn">Conectar</button>
 </form>
 </div>
</body>
</html>

```

```

<div class="back-link">
 ← Volver al inicio |


```

```

 if (data.statusCode === 200) {
 const parsedBody = JSON.parse(data.body);
 localStorage.setItem('email', parsedBody.email);
 localStorage.setItem('password', password);
 localStorage.setItem('username', parsedBody.username);
 alert('Usuario autenticado correctamente');
 window.location.href= 'index.html';

 } else {
 const errorMessage = data.body ? JSON.parse(data.body).error :
 'Error al iniciar sesión';
 alert(errorMessage);
 }
 } catch (err) {

}
});
</script>
</body>
</html>

```

- Para el registro copiamos el URL de invocación de /registroPOST, y lo ponemos en el siguiente FETCH en nuestro registro.html.

```

139 // Post a API gateway
140 try {
141 const response = await fetch('https://xc000gpm1j.execute-api.us-east-2.amazonaws.com/Dev/createuser', {
142 method: 'POST',
143 headers: { 'Content-Type': 'application/json' },
144 body: JSON.stringify(payload)
145 });
146

```

Código completo de registro (HTML, CSS) con Script JS para manejar las solicitudes y respuestas:

```

<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <title>GameHost - Registrarse</title>
 <link
 href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap"
 rel="stylesheet">
 <style>
 body {
 background-color: #0f0f0f;

```



```
color: #00ffcc;
font-family: 'Press Start 2P', cursive;
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
margin: 0;
}

.register-container {
background-color: #1a1a1a;
border: 2px solid #00ffcc;
border-radius: 12px;
padding: 40px;
width: 400px;
box-shadow: 0 0 15px #00ffcc;
animation: fadeIn 1s ease-in-out;
}

.register-container h2 {
text-align: center;
color: #fff;
margin-bottom: 30px;
}

label {
display: block;
margin: 15px 0 5px;
font-size: 10px;
color: #00ffff;
}

input[type="text"] {
width: 100%;
padding: 12px;
background-color: #0f0f0f;
border: 2px solid #00ffcc;
border-radius: 5px;
color: #00ffcc;
font-family: 'Press Start 2P', cursive;
font-size: 10px;
box-sizing: border-box;
```

```
}

.btn {
 margin-top: 25px;
 width: 100%;
 padding: 15px;
 background-color: #00ffcc;
 color: #000;
 border: none;
 font-size: 12px;
 border-radius: 5px;
 cursor: pointer;
 transition: transform 0.2s, background 0.3s;
}

.btn:hover {
 background-color: #00ffff;
 transform: scale(1.05);
}

@keyframes fadeIn {
 from { opacity: 0; transform: scale(0.9); }
 to { opacity: 1; transform: scale(1); }
}

.back-link {
 display: block;
 text-align: center;
 margin-top: 20px;
 font-size: 10px;
}

.back-link a {
 color: #00ffff;
 text-decoration: none;
}

.back-link a:hover {
 text-decoration: underline;
}

</style>
```



ITESO

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Diplomado – Cloud Computing (Development)

Intel Partnership

```
</head>
<body>
 <div class="register-container">
 <h2>📝 Registrarse</h2>
 <form id="register-form">
 <label for="username">Usuario</label>
 <input type="text" id="username" name="username" required>

 <label for="email">Correo electrónico</label>
 <input type="text" id="email" name="email" required>

 <label for="password">Contraseña</label>
 <input type="text" id="password" name="password" required>

 <button type="submit" class="btn">Crear Cuenta</button>
 </form>
 <div class="back-link">
 ⬅ Volver al inicio
 </div>
 </div>

 <script>
 document.getElementById('register-form').addEventListener('submit',
 async function(e) {
 e.preventDefault();

 const username = document.getElementById('username').value.trim();
 const email = document.getElementById('email').value.trim();
 const password = document.getElementById('password').value.trim();

 // Validacion de email
 const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
 if (!emailRegex.test(email)) {
 alert('⚠ El correo electrónico no es válido');
 return;
 }

 // Validar longitud password
 if (password.length < 8) {
 alert('⚠ La contraseña debe tener al menos 8 caracteres');
 return;
 }
 }

```

```

 }

 const payload = {
 User: { username, email, password }
 };

 // Post a API gateway
 try {
 const response = await fetch('https://xc000gpm1j.execute-api.us-
east-2.amazonaws.com/Dev/createuser', {
 method: 'POST',
 headers: { 'Content-Type': 'application/json' },
 body: JSON.stringify(payload)
 });

 const data = await response.json();

 if (response.ok && data.statusCode === 201) {
 alert('✓ Usuario registrado con éxito');
 window.location.href = 'index.html';
 } else {
 alert('⚠ Error: ' + JSON.parse(data.body).error);
 }
 } catch (err) {
 console.error(err);
 alert('✗ Error al conectar con el servidor');
 }
 });
</script>
</body>
</html>

```

- Para el index copiamos el URL de invocación POST de /createserver, /instancestatus y /onoff y lo ponemos en las siguientes variables declaradas en nuestro código index.html ya que por medio de estas se hacen las solicitudes requeridas para el proceso de crear servidor, checar su estatus y apagarlo o encenderlo en nuestro index.html.

```

137 // API Gateways
138 const API_URL = "https://v028fn9dek.execute-api.us-east-2.amazonaws.com/dev/createserver";
139 const CONSULTAR_ESTADO_URL = "https://v028fn9dek.execute-api.us-east-2.amazonaws.com/dev/instancestatus";
140 const ENCENDER_URL = "https://v028fn9dek.execute-api.us-east-2.amazonaws.com/dev/onoff";
141

```

Código completo de index (HTML, CSS) con Script JS para manejar las solicitudes y respuestas:

```

<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1" />
 <title>GameHUB - Servidores Minecraft y Terraria</title>
 <link
 href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap"
 rel="stylesheet">
 <style>
 body {
 margin: 0;
 padding: 0;
 background: #0f0f0f;
 color: #00ffcc;
 font-family: 'Press Start 2P', cursive;
 text-shadow: 1px 1px #000;
 }
 header {
 background: linear-gradient(90deg, #00ffcc, #0066ff);
 padding: 30px;
 text-align: center;
 color: #000;
 }
 nav {
 background: #111;
 padding: 15px;
 text-align: center;
 }
 nav a {
 color: #00ffcc;
 text-decoration: none;
 margin: 0 20px;
 font-size: 12px;
 }
 nav a:hover {
 color: #fff;
 }
 </style>

```



ITESO

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Diplomado – Cloud Computing (Development)

Intel Partnership

```
}

section {
 padding: 40px 20px;
 max-width: 1000px;
 margin: auto;
}

.card {
 background: #1a1a1a;
 border: 2px solid #00ffcc;
 border-radius: 10px;
 padding: 30px;
 margin-bottom: 40px;
 box-shadow: 0 0 10px #00ffcc;
}

.card h2 {
 color: #fff;
}

.btn {
 display: inline-block;
 background: #00ffcc;
 color: #000;
 padding: 15px 25px;
 font-size: 12px;
 border: none;
 border-radius: 5px;
 margin-top: 15px;
 cursor: pointer;
 transition: transform 0.2s, background 0.3s;
}

.btn:hover {
 background: #00ffff;
 transform: scale(1.05);
}

footer {
 background: #111;
 text-align: center;
 padding: 30px;
 font-size: 10px;
 color: #555;
}

</style>
</head>
```



```
<body>
 <header>
 <h1>🎮 GameHUB</h1>
 <p>Servidores dedicados para Minecraft y Terraria</p>
 </header>

 <nav>
 Minecraft
 Terraria
 Precios
 Contacto
 Log In
 Profile
 </nav>

 <section id="minecraft">
 <div class="card">
 <h2>🌐 Servidores Minecraft</h2>
 <p id="pMinecraft">Juega con amigos en mundos personalizados. Java y Bedrock, soporte 24/7.</p>
 <button class="btn" id="minecraftButton">Levantar Nuevo Servidor</button>
 <button class="btn reload" onclick="location.reload()">⟳</button>
 <div id="minecraftInstances"></div>
 </div>
 </section>

 <section id="terraria">
 <div class="card">
 <h2>⌚ Servidores Terraria</h2>
 <p id="p1">Explora, construye y sobrevive con mods o clásicos. Servidores estables y rápidos.</p>
 <button class="btn" id="terrariaButton">Levantar Nuevo Servidor</button>
 <button class="btn reload" onclick="location.reload()">⟳</button>
 <div id="terrariaInstances"></div>
 </div>
 </section>

 <section id="precios">
 <div class="card">
```

```

<h2>⌚ Planes</h2>

 Básico: 2GB RAM - $5 USD/mes

</div>
</section>

<section id="contacto">
 <div class="card">
 <h2>✉️ Contáctanos</h2>
 <p>¿Dudas? Escríbenos: soporte@gamehub.com</p>
 </div>
</section>

<footer>
 <p>© 2025 GameHUB. Powered by gamers, for gamers.</p>
</footer>

<script>
 const user = localStorage.getItem("email");
 const passwd = localStorage.getItem("password");

 // API Gateways
 const API_URL = "https://xc000gpm1j.execute-api.us-east-2.amazonaws.com/Dev/createserver";
 const CONSULTAR_ESTADO_URL = "https://xc000gpm1j.execute-api.us-east-2.amazonaws.com/Dev/instancestate";
 const ENCENDER_URL = "https://xc000gpm1j.execute-api.us-east-2.amazonaws.com/Dev/onoff";

 const terrariaBtn = document.getElementById("terrariaButton");
 const minecraftBtn = document.getElementById("minecraftButton");
 const p1 = document.getElementById("p1");
 const pMinecraft = document.getElementById("pMinecraft");

 const terrariaInstances = document.getElementById("terrariaInstances");
 const minecraftInstances =
 document.getElementById("minecraftInstances");

 // Consultar estado al cargar

```

```

consultarEstado("minecraft");
consultarEstado("terraria");

// Funcion para consultar el estado de los servidores minecraft o
terraria
async function consultarEstado(game) {
 try {
 const res = await fetch(CONSULTAR_ESTADO_URL, {
 method: "POST",
 headers: { "Content-Type": "application/json" },
 body: JSON.stringify({ User: { email: user, password: passwd } })
 });

 const data = await res.json();
 const instances = data.instances?.filter(i => i.game === game);

 const container = game === "minecraft" ? minecraftInstances :
terrariaInstances;
 container.innerHTML = '';// Limpiar el contenedor

 const instanceDiv = document.createElement("div");

 if (!instances || instances.length === 0) {
 instanceDiv.innerHTML = `<p>⚠️ No hay instancias activas.</p>`;
 container.appendChild(instanceDiv);
 } else {
 instances.forEach(instancia => {
 const instanceDiv = document.createElement("div");
 instanceDiv.innerHTML =
 `<p>>ID: ${instancia.InstanceId} - ${instancia.State ===
'running' ? '☑️ En ejecución' : '🔴 Detenido'}

 🌐 IP: ${instancia.PublicIpAddress}

 ⚡ Uso del CPU: ${instancia.cpuUtilization}</p>
 <button class="btn"
 onclick="toggleServer('${instancia.InstanceId}', '${game}')">
 ${instancia.State === 'running' ? 'Detener' : 'Levantar'}
 Servidor
 </button>
 `;
 container.appendChild(instanceDiv);
 });
 }
 }
}

```

```

 }

} catch (err) {
 console.error(err);
 const container = game === "minecraft" ? minecraftInstances :
terriaInstances;
 container.innerHTML = "<p>X Error al consultar estado.</p>";
}
}

// Funcion para levantar o apagar servidores
async function toggleServer(id, game) {
 const boton = event.target;
 const container = game === "minecraft" ? minecraftInstances :
terriaInstances;
 const instanciaDiv = boton.parentNode;

 boton.disabled = true;
 boton.textContent = "Procesando...";

 try {
 const estado = boton.textContent.includes("Levantar") ? "ON" :
"OFF";
 const res = await fetch(ENCENDER_URL, {
 method: "POST",
 headers: { "Content-Type": "application/json" },
 body: JSON.stringify({
 Instance: { id: id, status: estado }
 })
 });

 if (res.ok) {
 consultarEstado(game); // Actualizar el estado
 } else {
 instanciaDiv.innerHTML = "<p>X Error al cambiar el estado del
servidor.</p>";
 }
 } catch (err) {
 console.error(err);
 instanciaDiv.innerHTML = "<p>X Error de red.</p>";
 }
}

```

```

 }

}

// Levantar nuevo servidor
async function levantarServidor(game) {
 const container = game === "minecraft" ? minecraftInstances :
terriaInstances;
 const nuevoServidorDiv = document.createElement("div");
 nuevoServidorDiv.innerHTML = `<p>☒ Levantando nuevo servidor...</p>`;
 container.appendChild(nuevoServidorDiv);

 try {
 const res = await fetch(API_URL, {
 method: "POST",
 headers: { "Content-Type": "application/json" },
 body: JSON.stringify({
 User: { email: user, password: passwd },
 game: game
 })
 });
 });

 const data = await res.json();

 if (res.ok && data.public_ip) {
 nuevoServidorDiv.innerHTML = `
 ☑ Servidor de ${game} activo
🌐 IP: ${data.public_ip}

 <button class="btn" onclick="toggleServer('${data.instance_id}', '${game}')">Detener Servidor</button>
 `;
 } else {
 nuevoServidorDiv.innerHTML = `<p>☒ Error al levantar el
 servidor.</p>`;
 }
 } catch (err) {
 console.error(err);
 nuevoServidorDiv.innerHTML = `<p>☒ Error de red.</p>`;
 }
}

minecraftBtn.addEventListener("click", () =>
levantarServidor("minecraft"));

```



ITESO

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Diplomado – Cloud Computing (Development)

Intel Partnership

```
 terrariaBtn.addEventListener("click", () =>
levantarServidor("terraria"));

</script>
</body>
</html>
```

- Para el perfil copiamos el URL de invocación de /instancestatus POST, y lo ponemos en la siguiente variable (CONSULTAR\_ESTADO\_URL) en nuestro perfil.html.

```
document.addEventListener('DOMContentLoaded', ()=>{
 const CONSULTAR_ESTADO_URL = "https://v028fn9dek.execute-api.us-east-2.amazonaws.com/dev/instancestatus";
 emailText=document.getElementById('email');
 perfilText=document.getElementById('username');
 instancesText=document.getElementById('instances');
 emailText.textContent=localStorage.getItem('email');
 perfilText.textContent=localStorage.getItem('username');
```

Código completo de perfil (HTML, CSS) con Script JS para manejar las solicitudes y respuestas:

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <title>GameHost - Perfil de Usuario</title>
 <link
 href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap"
 rel="stylesheet">
 <style>
 body {
 background-color: #0f0f0f;
 color: #00ffcc;
 font-family: 'Press Start 2P', cursive;
 margin: 0;
 padding: 0;
 }

 header {
 padding: 20px;
 text-align: center;
 background-color: #1a1a1a;
 border-bottom: 2px solid #00ffcc;
 }
```

```
header h1 {
 font-size: 20px;
 margin: 0;
 text-shadow: 0 0 5px #00ffcc;
}

.logout {
 position: absolute;
 top: 20px;
 right: 20px;
 font-size: 10px;
 text-decoration: none;
 color: #ff0066;
}

.section {
 padding: 30px;
}

.section h2 {
 font-size: 14px;
 margin-bottom: 20px;
 color: #00ffff;
}

.gallery {
 display: flex;
 flex-wrap: wrap;
 gap: 20px;
 justify-content: center;
}

.gallery img,
.gallery video {
 border: 2px solid #00ffcc;
 border-radius: 10px;
 max-width: 200px;
 max-height: 150px;
 object-fit: cover;
}
```



ITESO

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Diplomado – Cloud Computing (Development)

Intel Partnership

```
.upload-form {
 text-align: center;
 margin-top: 40px;
}

.upload-form input[type="file"] {
 margin: 10px 0;
 background: #1a1a1a;
 border: 1px solid #00ffcc;
 color: #00ffcc;
 font-family: 'Press Start 2P', cursive;
 font-size: 10px;
}

.btn {
 margin-top: 10px;
 padding: 10px 20px;
 background-color: #00ffcc;
 color: #000;
 border: none;
 font-size: 10px;
 border-radius: 5px;
 cursor: pointer;
 transition: transform 0.2s, background 0.3s;
}

.btn:hover {
 background-color: #00ffff;
 transform: scale(1.05);
}

footer {
 text-align: center;
 padding: 20px;
 font-size: 8px;
 color: #555;
}
 </style>
</head>
<body>
 <header>
```



```
<h1>👤 Email: </h1>
<h1>👤 Username: </h1>
<hr>
Home
</header>

<div class="section">
 <h2>💻 Instancias del usuario</h2>
 <div class="instan">
 <p id="instances"></p>
 </div>
 <h2>🖼 Galería de Imágenes</h2>
 <div class="gallery">
 <!-- Imágenes subidas -->

 </div>

 <div class="upload-form">
 <form action="/upload-image" method="post" enctype="multipart/form-data">
 <label for="image">Subir imagen:</label>

 <input type="file" name="image" id="image" accept="image/*" required>

 <button type="submit" class="btn">Subir Imagen</button>
 </form>
 </div>
</div>

<div class="section">
 <h2>🎥 Galería de Videos</h2>
 <div class="gallery">
 <!-- Videos subidos -->
 <video controls>
 <source src="video/sample1.mp4" type="video/mp4">
 Tu navegador no soporta videos.
 </video>
 <video controls>
 <source src="video/sample2.mp4" type="video/mp4">
 Tu navegador no soporta videos.
 </video>
 </div>
</div>
```

```

 </div>

 <div class="upload-form">
 <form action="/upload-video" method="post" enctype="multipart/form-data">
 <label for="video">Subir video:</label>

 <input type="file" name="video" id="video" accept="video/*" required>

 <button type="submit" class="btn">Subir Video</button>
 </form>
 </div>
 </div>

 <footer>
 GameHost © 2025 - Servidores épicos para gamers épicos 🎮💧
 </footer>

 <script>

 document.addEventListener('DOMContentLoaded', ()=>{
 const CONSULTAR_ESTADO_URL = "https://xc000gpm1j.execute-api.us-east-2.amazonaws.com/Dev/instancestate";
 emailText=document.getElementById('email');
 perfilText=document.getElementById('username');
 instancesText=document.getElementById('instances');
 emailText.textContent=localStorage.getItem('email');
 perfilText.textContent=localStorage.getItem('username');

 const user= localStorage.getItem('email');
 const passwd= localStorage.getItem('password');
 async function consultarEstado() {
 try {
 const res = await fetch(CONSULTAR_ESTADO_URL, {
 method: "POST",
 headers: { "Content-Type": "application/json" },
 body: JSON.stringify({ User: { email: user, password: passwd } })
 });

 const data = await res.json();
 console.log(JSON.stringify(data));
 const instances=data.instances;
 }
 }
 });
 </script>

```

```

 if (!instances || instances.length === 0) {
 instancesText.textContent = "💤 No hay instancias activas de
Terraria.";
 return;
 }

 let instancesContent = '';
 instances.forEach(instancia => {
 if (instancia.State === "running") {
 instancesContent += `✅ Servidor en ejecución
>ID:
${instancia.InstanceId}
🌐 IP: ${instancia.PublicIpAddress}
📊 Uso
del CPU: ${instancia.cpuUtilization}

`;
 } else {
 instancesContent += "🔴 Servidor detenido

";
 }
 });

 instancesText.innerHTML = instancesContent;

 } catch (err) {
 console.error(err);
 instancesText.textContent = "❌ Error al consultar estado.";
 }
};

consultarEstado();

});
</script>
</body>
</html>

```

## CLOUD FRONT:

1. Nos metemos a crear un cloud front y en origin ponemos el bucket de ec2

**Create distribution**

**Origin**

**Origin domain**  
Choose an AWS origin, or enter your origin's domain name. [Learn more](#)

Enter a valid DNS domain name, such as an S3 bucket, HTTP server, or VPC origin ID.

**Origin path - optional**  
Enter a URL path to append to the origin domain name for origin requests.

**Name**  
Enter a name for this origin.

**Origin access** | [Info](#)

- Public**  
Bucket must allow public access.
- Origin access control settings (recommended)**  
Bucket can restrict access to only CloudFront.
- Leave access identities**

2. Aceptamos al uso del website endpoint

**Origin**

**Origin domain**  
Choose an AWS origin, or enter your origin's domain name. [Learn more](#)

Enter a valid DNS domain name, such as an S3 bucket, HTTP server, or VPC origin ID.

**Protocol** | [Info](#)

- HTTP only**
- HTTPS only**
- Match viewer**

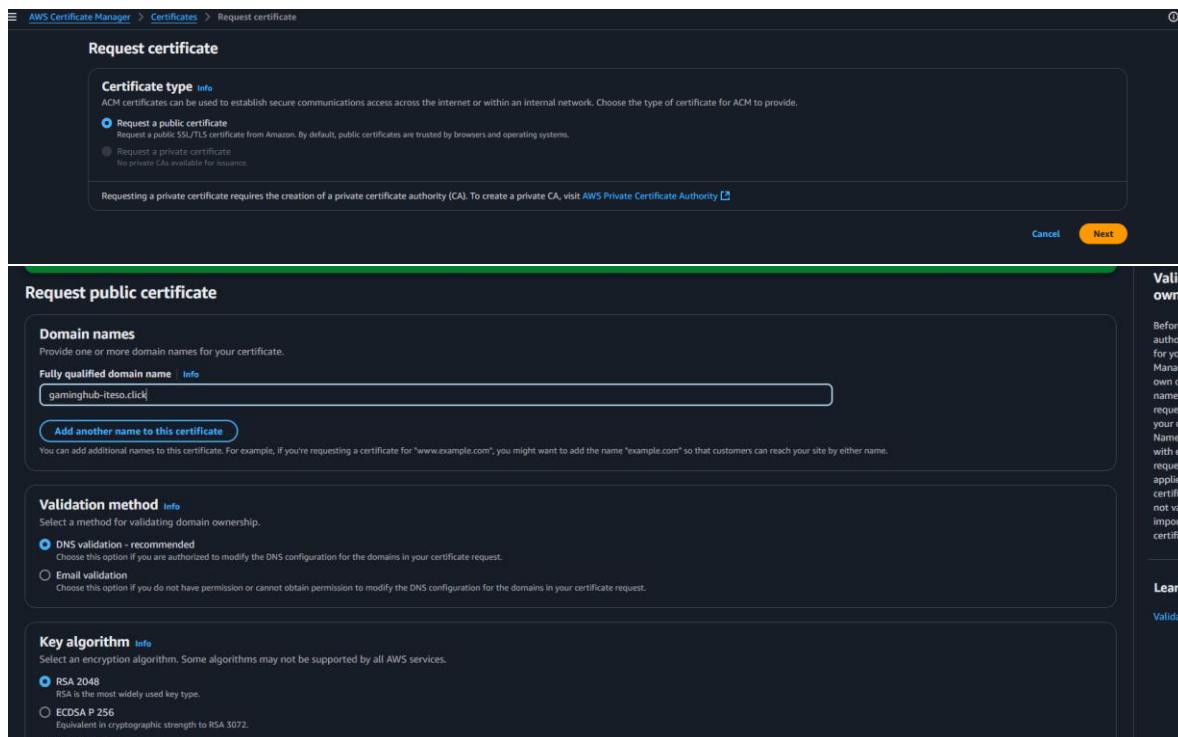
**HTTP port**  
Enter your origin's HTTP port. The default is port 80.

**Origin path - optional**  
Enter a URL path to append to the origin domain name for origin requests.

**Name**  
Enter a name for this origin.

**Add custom headers - optional**

3. Tenemos que ingresar a certificate para generar un certificado de ACM para el dominio con el mismo nombre que le vas a poner al route 53



**Request certificate**

**Certificate type** Info  
ACM certificates can be used to establish secure communications across the internet or within an internal network. Choose the type of certificate for ACM to provide.

Request a public certificate  
Request a public SSL/TLS certificate from Amazon. By default, public certificates are trusted by browsers and operating systems.

Request a private certificate  
No private CAs available for issuance.

Requesting a private certificate requires the creation of a private certificate authority (CA). To create a private CA, visit [AWS Private Certificate Authority](#).

**Request public certificate**

**Domain names**  
Provide one or more domain names for your certificate.

**Fully qualified domain name** Info

**Add another name to this certificate**  
You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name.

**Validation method** Info  
Select a method for validating domain ownership.

DNS validation - recommended  
Choose this option if you are authorized to modify the DNS configuration for the domains in your certificate request.

Email validation  
Choose this option if you do not have permission or cannot obtain permission to modify the DNS configuration for the domains in your certificate request.

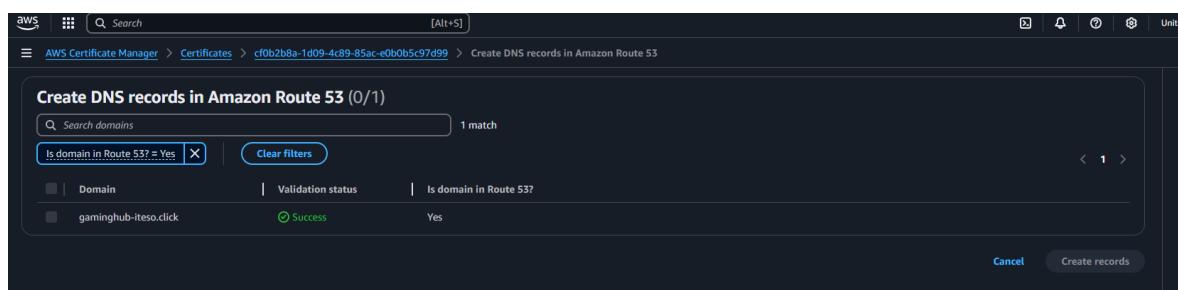
**Key algorithm** Info  
Select an encryption algorithm. Some algorithms may not be supported by all AWS services.

RSA 2048  
RSA is the most widely used key type.

ECDSA P 256  
Equivalent in cryptographic strength to RSA 3072.

ECDSA P 384

4. Vamos a pedir un record para mostrar que de echo somos dueños del dominio de Route 53



**Create DNS records in Amazon Route 53 (0/1)**

Search domains	1 match	
<input type="text" value="Is domain in Route 53 = Yes"/>	<input type="button" value="Clear filters"/>	
<input type="checkbox"/> Domain	<input type="checkbox"/> Validation status	<input type="checkbox"/> Is domain in Route 53?
<input type="checkbox"/> gaminghub-iteso.click		Yes

**Create records**

5. Con eso deberíamos tener el certificado

**cf0b2b8a-1d09-4c89-85ac-e0b0b5c97d99**

**Certificate status**

Identifier	cf0b2b8a-1d09-4c89-85ac-e0b0b5c97d99	Status	Issued
ARN	arn:aws:acm:us-east-1:550829113730:certificate/cf0b2b8a-1d09-4c89-85ac-e0b0b5c97d99		
Type	Amazon Issued		

**Domains (1)**

Domain	Status	Renewal status	Type	CNAME name
gaminghub-iteso.click	Success	-	CNAME	!_b64fc7626aa9740afb15d0116d6c07b1.gaminghub-iteso.click.

**Details**

## 6. Ahora podemos ponerle el nombre alternativo al cloudfront

- Use all edge locations (best performance)
- Use only North America and Europe
- Use North America, Europe, Asia, Middle East, and Africa

**Alternate domain name (CNAME) - optional**  
Add the custom domain names that you use in URLs for the files served by this distribution.

Remove
  
Add item

(i) To add a list of alternative domain names, use the [bulk editor](#).

## 7. Y ponemos el certificado para mostrar que tenemos el dominio

(i) To add a list of alternative domain names, use the [bulk editor](#).

**Custom SSL certificate - optional**  
Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

gaminghub-iteso.click (cf0b2b8a-1d09-4c89-85ac-e0b0b5c97d99)

gaminghub-iteso.click [ Request certificate ]

**Legacy clients support - \$600/month prorated charge applies. Most customers do not need this.**  
CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS.

CloudFront > Distributions > E2D0XEQZLAZYFQ

Successfully created new distribution.

To get in-depth monitoring information for your distribution's internet traffic, create an Internet Monitor.

**E2D0XEQZLAZYFQ**

**General** **Security** **Origins** **Behaviors** **Error pages** **Invalidations** **Tags** **Logging**

**Details**

Distribution domain name: d16is277qdu77l.cloudfront.net

ARN: arn:aws:cloudfront:us-east-1:550829113730:distribution/E2D0XEQZLAZYFQ

Last modified: Deploying

**Settings**

Description: Use all edge locations (best performance)

Price class: Standard

Supported HTTP versions: HTTP/2, HTTP/1.1, HTTP/1.0

Alternate domain names: gaminghub-iteso.click

Custom SSL certificate: gaminghub-iteso.click

Security policy: TLSv1.2\_2021

Standard logging: Off

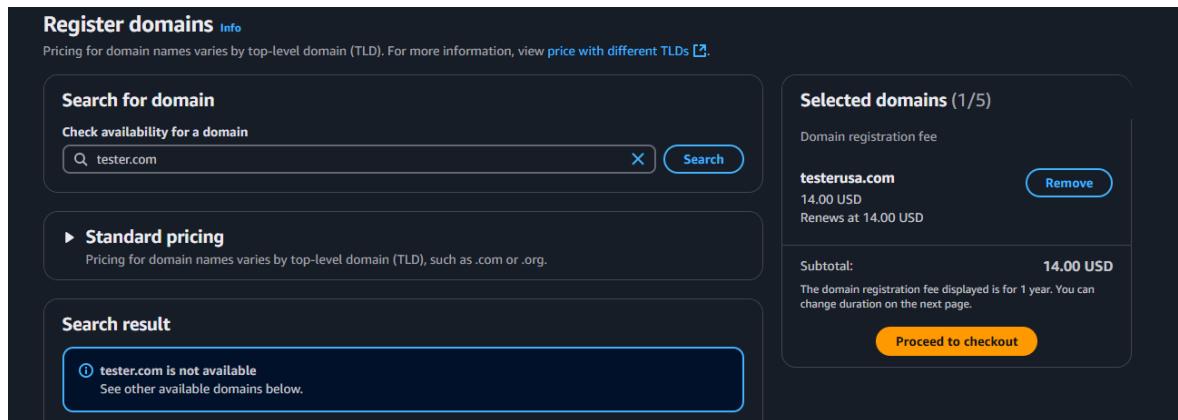
Cookie logging: Off

Default root object: -

**Continuous deployment** [Create staging distribution](#)

## ROUTE 53:

1. Registramos un dominio y pagamos por su uso



**Register domains Info**  
Pricing for domain names varies by top-level domain (TLD). For more information, view [price with different TLDs](#).

**Search for domain**  
Check availability for a domain  
tester.com

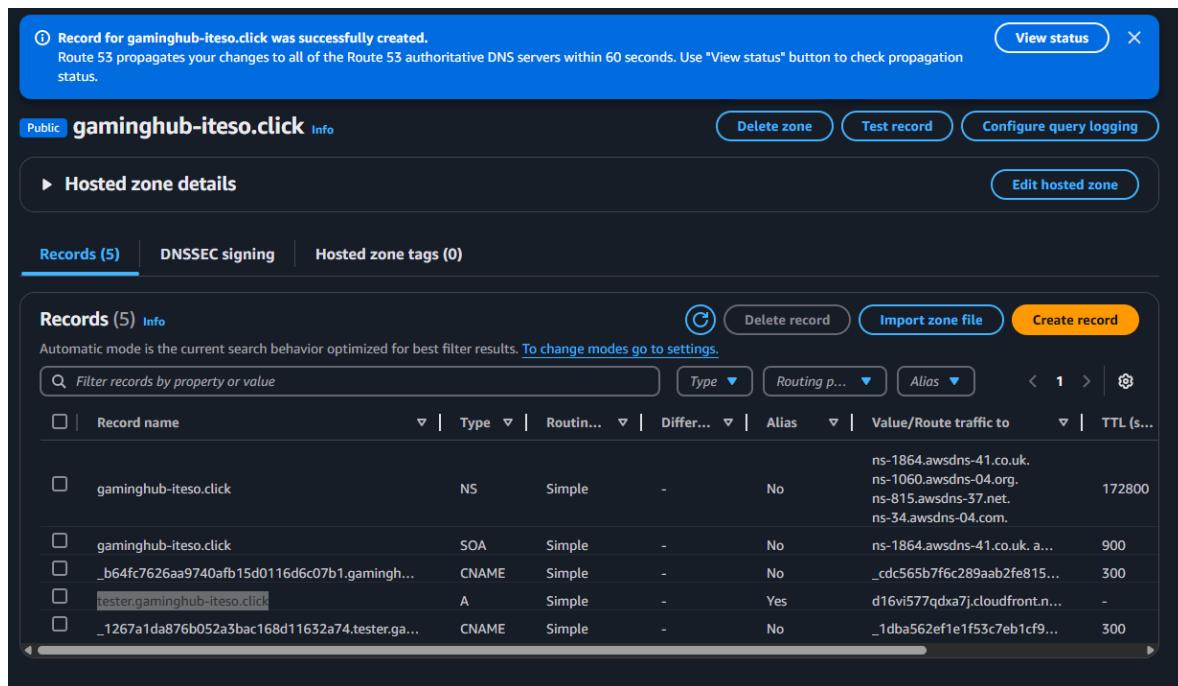
► **Standard pricing**  
Pricing for domain names varies by top-level domain (TLD), such as .com or .org.

**Search result**  
tester.com is not available  
See other available domains below.

**Selected domains (1/5)**  
Domain registration fee  
**testerusa.com** 14.00 USD   
Renews at 14.00 USD

**Subtotal:** **14.00 USD**  
The domain registration fee displayed is for 1 year. You can change duration on the next page.

2. Dentro de hosted zones, nos metemos al dominio y crearemos un nuevo récord



Record for gaminghub-iteso.click was successfully created.  
Route 53 propagates your changes to all of the Route 53 authoritative DNS servers within 60 seconds. Use "View status" button to check propagation status.

**gaminghub-iteso.click Info**

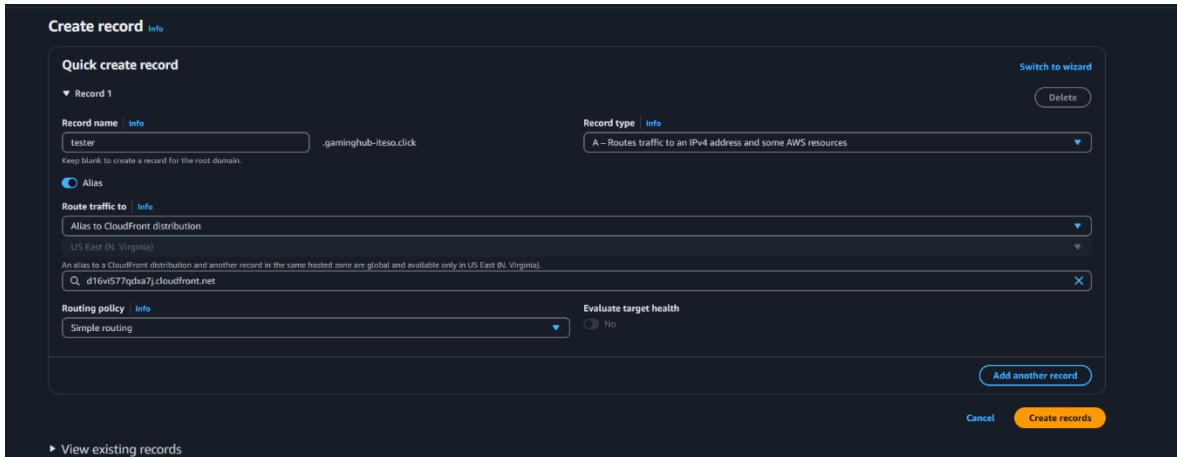
► **Hosted zone details**

**Records (5) Info**

Automatic mode is the current search behavior optimized for best filter results. [To change modes go to settings.](#)

<input type="checkbox"/>	Record name	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...)
<input type="checkbox"/>	gaminghub-iteso.click	NS	Simple	-	No	ns-1864.awsdns-41.co.uk. ns-1060.awsdns-04.org. ns-815.awsdns-37.net. ns-34.awsdns-04.com.	172800
<input type="checkbox"/>	gaminghub-iteso.click	SOA	Simple	-	No	ns-1864.awsdns-41.co.uk. a...	900
<input type="checkbox"/>	_b64fc7626aa9740afb15d0116d6c07b1.gamingh...	CNAME	Simple	-	No	_cdc565b7f6c289aab2fe815...	300
<input type="checkbox"/>	tester.gaminghub-iteso.click	A	Simple	-	Yes	d16vi577qda7j.cloudfront.n...	-
<input type="checkbox"/>	_1267a1d876b052a3bac168d11632a74.tester.ga...	CNAME	Simple	-	No	_1dba562ef1e1f153c7eb1cf9...	300

### 3. Creamos el récord con la instancia de CloudFront



The screenshot shows the 'Create record' interface in the AWS Route 53 console. A new record named 'tester' is being created for the domain '.gaminghub-iteso.click'. The record type is set to 'A' (IPv4 address and some AWS resources), which routes traffic to an IP address or specific AWS resources. The target for this record is an alias pointing to a CloudFront distribution located in the US East (N. Virginia) region, specifically the endpoint 'd16w577qdxs7.cloudfront.net'. The routing policy is set to 'Simple routing'. There are buttons for 'Switch to wizard', 'Delete', 'Add another record', 'Cancel', and 'Create records'.

### 4. Listo



**NOTA:** CloudFront(sobre todo el certificado) y Route 53 se deben de hacer al mismo tiempo y pensando en el nombre del subdominio desde el inicio. si la página es tester.gaminghub.com, tienes que llamarle así en el Certificate y ponerlo exactamente igual en el Route53 como récord o te va a dar error 53[yo cometí ese error y tuve que repetir el proceso del certificado con tester(en Route 53 era tester.gaminghub-iteso.click y en el certificado era gaminghub-iteso.click)]

## COSTO

Gracias a aws y su cuenta gratis en total este proyecto nos ha costado 13.64 dólares mensuales(avg), lo cual es muy bajo a comparación de los valores que sacamos de la calculadora. AWS nos regaló 750 horas de EC2, 5 GB de almacenamiento en los s3, 25 GB de almacenamiento de DynamoDB, 1 millón de llamadas a él API Gateway y 1TB gratis de CloudFront [13]. Aparte de los precios de servicios, pagamos 3 dólares por el dominio de gaming Hub Iteso, por lo cual cree que si utilizamos bien la versión gratis de aws. Obviamente este precio va a escalar con el tiempo ya que se levantarían más instancias de EC2, aumentando el precio mensual del producto.



## Checando con la calculadora de precios

My Estimate									Duplicate	Delete	Move to	Create group	Add support	Add service
									< 1 >					⚙️
	Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary							
<input type="checkbox"/>	Amazon EC2	0 -	0.00 USD	7.13 USD	-	US East (Ohio)	Tenancy (Shared Inst...							
<input type="checkbox"/>	Amazon Route 53	0 -	0.00 USD	0.50 USD	-	US East (Ohio)	Hosted Zones (1)							
<input type="checkbox"/>	Amazon CloudFront	0 -	0.00 USD	0.11 USD	-	US East (Ohio)	Data transfer out to i...							
<input type="checkbox"/>	AWS Lambda	0 -	0.00 USD	0.00 USD	-	US East (Ohio)	Architecture (x86, Ar...							
<input type="checkbox"/>	Amazon DynamoDB	0 -	180.00 USD	12.35 USD	-	US East (Ohio)	Table class (Standard)...							
<input type="checkbox"/>	Amazon Virtual Priva...	0 -	0.00 USD	46.18 USD	-	US East (Ohio)	Number of network a...							
<input type="checkbox"/>	Amazon API Gateway	0 -	0.00 USD	14.60 USD	-	US East (Ohio)	HTTP API requests un...							

## CONCLUSIONES PARTE 1

La propuesta de Gaming Hub presenta un enfoque innovador para la gestión y alojamiento de servidores de videojuegos en la nube utilizando los servicios de AWS. A través de herramientas como Amazon EC2, S3, CloudWatch, Lambda y API Gateway, se plantea una solución escalable, segura y eficiente para que los usuarios puedan crear y administrar sus propios servidores. Además, se consideran aspectos clave como el monitoreo del rendimiento, el almacenamiento de datos y la distribución de tráfico con Elastic Load Balancing, garantizando una experiencia óptima.

Este proyecto busca ofrecer una plataforma accesible e intuitiva, con opciones de personalización y seguridad mediante AWS IAM y Amazon VPC. La propuesta incluye características como la escalabilidad automática y la gestión flexible de servidores, con posibilidades de crecimiento a futuro. Como siguiente paso, se evaluará la viabilidad técnica y se definirán los detalles de implementación para desarrollar un sistema funcional que responda a las necesidades de los jugadores y optimice el uso de los recursos en la nube.

## CONCLUSIONES PARTE 2

Este proyecto demuestra cómo una función Lambda puede integrarse con Amazon services para controlar instancias EC2, y conexión con otros servicios como DYNAMO, CloudWatch, etc., enfocándose en ofrecer una solución accesible y automatizada para obtener métricas clave como el uso de CPU y memoria. La implementación considera buenas prácticas de seguridad mediante roles IAM, así como la posibilidad de extender la solución dentro de una arquitectura escalable en Amazon VPC.

A través de API Gateway, logramos integrar funciones Lambda para acciones como el despliegue de instancias y el registro de usuarios, entre otras funcionalidades. Configuramos el método de solicitud (en este caso, principalmente POST) para facilitar el envío de solicitudes utilizando JavaScript. En el código de JavaScript manejamos correctamente las respuestas devueltas por el API para de esta forma poder mostrar mensajes a los usuarios o actualizar divs con información relevante como lo puede ser el uso del CPU, instancias asociadas al usuario y muchas cosas más.

A parte de esto, nos gustaría hablar un poco de Route 53, ya que esto se complicó un poco más de lo que se creía. Route 53 trabaja en conjunto con AWS Certificate Manager, ya que se necesita otorgar certificados a CloudFront para que este pueda reconocer su nombre en la web ([pruebas.gaminhub-iteso.click](http://pruebas.gaminhub-iteso.click)), lo cual no teníamos claro al momento de crear nuestro CloudFront. Por esta razón, tuvimos que recrear todo el CloudFront y generar varios certificados hasta que entendimos cómo funcionaban y cómo se relacionaban con Route 53.

Finalmente, algo que nos impactó de forma negativa y al mismo tiempo positiva fue el firewall de AWS, el cual se configuró automáticamente sin avisar cuando creamos el CloudFront con Route 53. Sin embargo, fue el responsable del 30 % de nuestro costo total.

## BIBLIOGRAFÍA

- [1] Amazon Web Services, "Amazon GameLift – Dedicated Game Server Hosting," 2024. [Online]. Available: <https://aws.amazon.com/gamelift/>
- [2] Amazon Web Services, Inc. "¿Qué es Amazon EC2?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/AWSEC2/latest/UserGuide/concepts.html](https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html). [Accedido: 24-mar-2025].
- [3] Amazon Web Services, Inc. "¿Qué es Amazon S3?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/AmazonS3/latest/userguide>Welcome.html](https://docs.aws.amazon.com/es_es/AmazonS3/latest/userguide>Welcome.html). [Accedido: 24-mar-2025].
- [4] Amazon Web Services, Inc. "¿Qué es Amazon CloudWatch?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html](https://docs.aws.amazon.com/es_es/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html). [Accedido: 24-mar-2025].
- [5] Amazon Web Services, Inc. "¿Qué es Amazon CloudFront?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/AmazonCloudFront/latest/DeveloperGuide/introduction.html](https://docs.aws.amazon.com/es_es/AmazonCloudFront/latest/DeveloperGuide/introduction.html). [Accedido: 24-mar-2025].
- [6] Amazon Web Services, Inc. "¿Qué es Amazon RDS?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/AmazonRDS/latest/UserGuide>Welcome.html](https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide>Welcome.html). [Accedido: 24-mar-2025].
- [7] Amazon Web Services, Inc. "¿Qué es AWS Lambda?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/lambda/latest/dg/welcome.html](https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html). [Accedido: 24-mar-2025].
- [8] Amazon Web Services, Inc. "¿Qué es Amazon API Gateway?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/apigateway/latest/developerguide/welcome.html](https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/welcome.html). [Accedido: 24-mar-2025].
- [9] Amazon Web Services, Inc. "¿Qué es AWS IAM?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/IAM/latest/UserGuide/introduction.html](https://docs.aws.amazon.com/es_es/IAM/latest/UserGuide/introduction.html). [Accedido: 24-mar-2025].
- [10] Amazon Web Services, Inc. "¿Qué es Amazon VPC?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/vpc/latest/userguide/what-is-amazon-vpc.html](https://docs.aws.amazon.com/es_es/vpc/latest/userguide/what-is-amazon-vpc.html). [Accedido: 24-mar-2025].
- [11] Amazon Web Services, Inc. "¿Qué es Amazon Route 53?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/Route53/latest/DeveloperGuide>Welcome.html](https://docs.aws.amazon.com/es_es/Route53/latest/DeveloperGuide>Welcome.html). [Accedido: 24-mar-2025].

[12] Amazon Web Services, Inc. "¿Qué es Elastic Load Balancing?" [En línea]. Disponible: [https://docs.aws.amazon.com/es\\_es/elasticloadbalancing/latest/application/introduction.html](https://docs.aws.amazon.com/es_es/elasticloadbalancing/latest/application/introduction.html). [Accedido: 24-mar-2025].

[13] "Capa gratuita de AWS | Cloud computing gratis |AWS," *Amazon Web Services, Inc.* [https://aws.amazon.com/es/free/?trk=2590dac4-c7c1-4106-8878-5344251a0baf&sc\\_channel=ps&ef\\_id=Cj0KCQjw8cHABhC-ARIsAJnY12xHcKEhsN4rRT4IHntsTu1ChC0ohGXLb37dM3LTQuyer43IrIQussaAlOJEALw\\_wcB:G:s&s\\_kwcid=AL!4422!3!646879396464!e!!g!!aws%20free%20tier!19636893018!145054767999&gad\\_campaignid=19636893018&gbraid=0AAAAAADjHtp\\_tKvL5v7h-zyn9dz82\\_p7TG&gclid=Cj0KCQjw8cHABhC-ARIsAJnY12xHcKEhsN4rRT4IHntsTu1ChC0ohGXLb37dM3LTQuyer43IrIQussaAlOJEALw\\_wcB&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=\\*all&awsf.Free%20Tier%20Categories=\\*all](https://aws.amazon.com/es/free/?trk=2590dac4-c7c1-4106-8878-5344251a0baf&sc_channel=ps&ef_id=Cj0KCQjw8cHABhC-ARIsAJnY12xHcKEhsN4rRT4IHntsTu1ChC0ohGXLb37dM3LTQuyer43IrIQussaAlOJEALw_wcB:G:s&s_kwcid=AL!4422!3!646879396464!e!!g!!aws%20free%20tier!19636893018!145054767999&gad_campaignid=19636893018&gbraid=0AAAAAADjHtp_tKvL5v7h-zyn9dz82_p7TG&gclid=Cj0KCQjw8cHABhC-ARIsAJnY12xHcKEhsN4rRT4IHntsTu1ChC0ohGXLb37dM3LTQuyer43IrIQussaAlOJEALw_wcB&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all)