



## Workshop

*Sa'am Wanderlust, aplicación Serverless con  
reconocimiento de imágenes*

---

Febrero 2020

V 1.7

# Sa'am Wanderlust Serverless Workshop

En este workshop usted creará una aplicación web que le permitirá a los usuarios cargar fotos desde su celular para guardar las memorias de sus viajes a sus playas preferidas en México. Esta WebApp contará con una interfaz en HTML donde los usuarios podrán registrarse, crear álbumes automáticamente, subir sus fotografías y se clasificarán en alguno de los álbumes de acuerdo al tipo de fotografía.

La arquitectura de esta aplicación utiliza [AWS Lambda](#), [Amazon API Gateway](#), [Amazon S3](#), [Amazon DynamoDB](#), [Amazon Rekognition](#) y [Amazon Cognito](#).

Los archivos estáticos como HTML, CSS, JavaScript, imágenes y videos, se guardarán en un bucket de S3 que configuraremos como sitio web estático, esta interfaz web se conectará a un backend basado en APIs públicas, desarrolladas en AWS Lambda y API Gateway.

Utilizaremos Amazon Cognito para el registro y autenticación de los usuarios para asegurar el acceso a los APIs del backend.

DynamoDB será nuestro almacén de datos para todas las operaciones de los usuarios, como guardar la metadata de las imágenes y la ubicación de las mismas.

El servicio de Rekognition lo utilizaremos para analizar las imágenes cargadas por los usuarios y detectar rostros en las mismas, también para clasificarlas en selfies, imágenes de grupo y otras.

## Requisitos

### Cuenta de AWS

Para poder completar este workshop, usted necesitará una cuenta de AWS con el nivel de permisos necesarios para crear:

- Usuarios de IAM
- Buckets de S3
- Tablas de DynamoDB
- Funciones Lambda
- APIs y recursos de API Gateway
- Cognito user y identity pools

Todos los recursos que se creen como parte de este workshop entran en la capa gratuita de AWS, si su cuenta tiene menos de un año de haberse creado. Para más detalle consulte la página de la [capa gratuita de AWS](#).

## Navegador de Internet

Para este workshop recomendamos utilizar la última versión de Chrome.

## Editor de texto

Será necesario contar con un editor de texto para realizar cambios en algunas funciones de código y archivos de configuración de la aplicación, así como modificaciones opcionales a archivos HTML, CSS y JS. Una opción gratuita y popular de editor de código es Atom el cual se puede descargar desde [aquí](#).

## Módulos

Este workshop está dividido en los siguientes módulos:

1. Sitio web estático en S3
2. Registro y Log-in de usuarios
3. Carga y análisis de fotos
4. Implementar el Rest API Backend
5. Prueba de la aplicación

# Módulo 1: Sitio web estático en S3

## Instrucciones de implementación

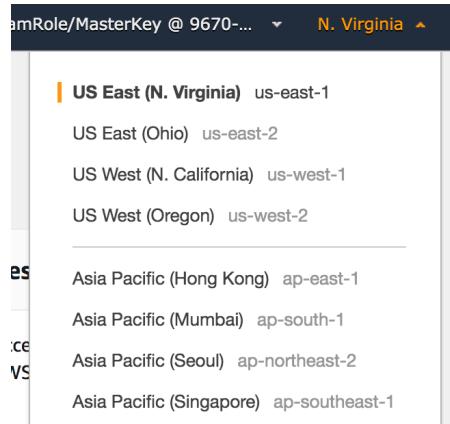
### Selección de Región

Este Workshop se puede crear en cualquier región que soporte los siguientes servicios:

- Amazon Cognito
- AWS Lambda
- Amazon API Gateway
- Amazon S3
- Amazon DynamoDB
- Amazon Rekognition

Para consultar los servicios soportados por región, consulte la siguiente [URL](#).

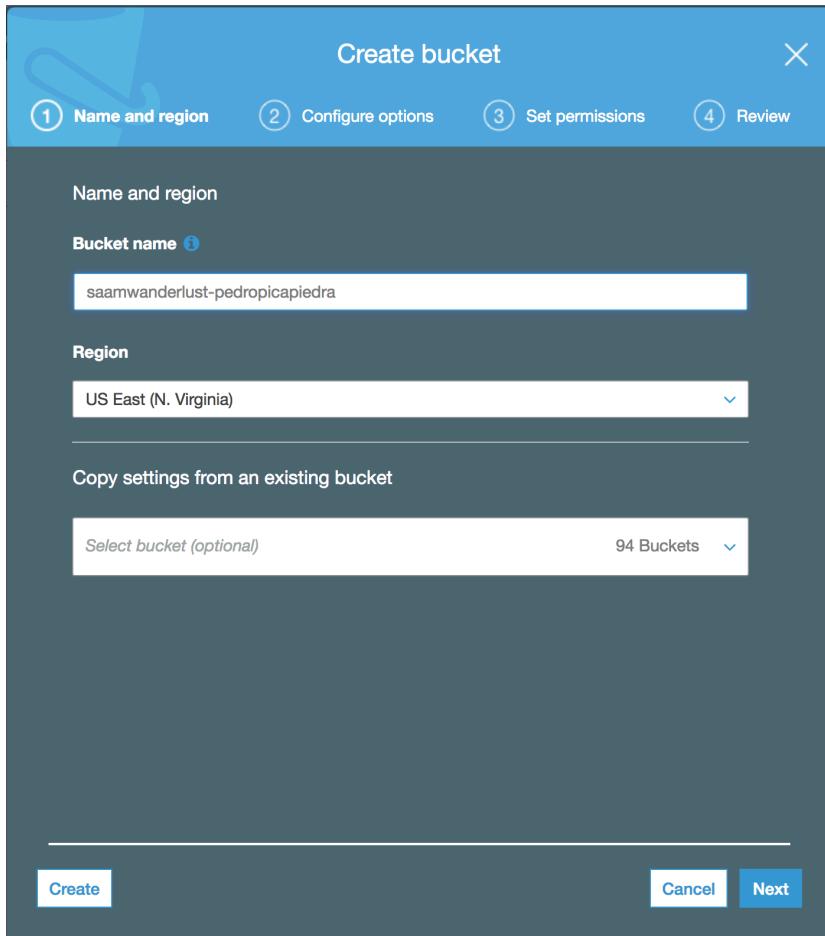
Una vez que haya seleccionado la región deberá lanzar todos los recursos en la misma región, seleccione la región del menú de regiones en la parte superior derecha de la consola.



## 1. Crear un bucket de S3

Usted puede crear un bucket de S3 y configurarlo para hospedar un sitio web estático, en este paso vamos a crear un bucket para almacenar los archivos estáticos de la aplicación web y volverlos públicos para que los usuarios puedan acceder a la aplicación.

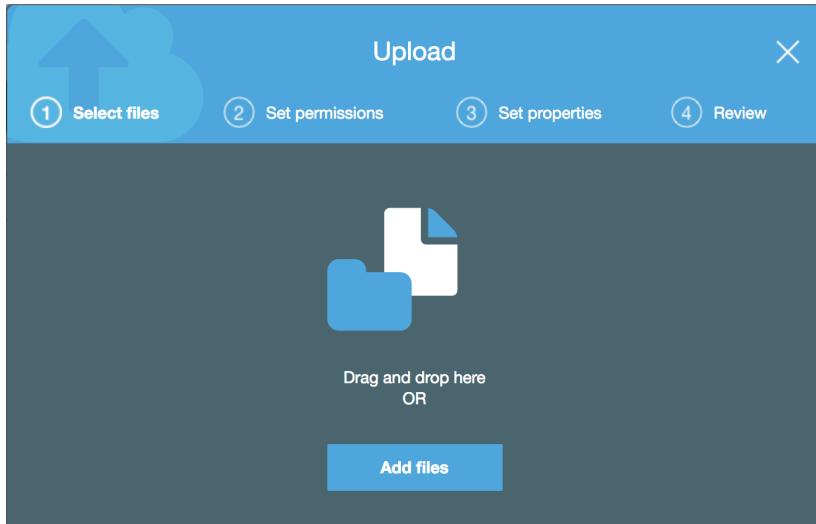
1. En la consola de administración de AWS seleccione **Services** y luego seleccione S3 de la categoría de **Storage**. También puede escribir S3 en el buscador y seleccionar la opción del resultado de búsqueda
2. Seleccione **+ Create bucket**
3. Escriba un nombre único para el bucket como **saamwanderlust-pedropicapedra** (*con su nombre al final, pedropicapedra es un ejemplo y no se puede usar*)
4. Seleccione la región con la que va a trabajar el resto del workshop de la lista de regiones
5. Seleccione **Create**



## 2. Copiar el sitio estático

A continuación copiaremos los archivos estáticos de la aplicación web al bucket de S3, para esto es necesario descargar los archivos estáticos comprimidos en un archivo ZIP de la URL que el instructor proporcionará, descomprimirlo, y arrastrar todo el contenido de la carpeta descomprimida a la sección de **Drag and drop** de la consola de S3 (*No copie la carpeta que se crea al descomprimir, sólo el contenido de la misma!*). Asegúrese de utilizar la versión más reciente de Chrome.

1. De la consola de S3 seleccione el bucket creado en la sección anterior
2. Haga clic en **Upload**
3. Arrastre todos los archivos y carpetas del directorio descomprimido en el área **Drag and drop**



Una vez terminado el proceso de copiado de archivos, verifique que el proceso se haya completado correctamente, los archivos deben aparecer listados dentro del bucket de la siguiente manera:

Name	Last modified	Size	Storage
css	--	--	--
js	--	--	--
uploads	--	--	--
confirm.html	Sep 20, 2018 3:58:38 PM GMT-0500	2.7 KB	Standard
index.html	Dec 4, 2018 10:49:27 AM GMT-0600	4.1 KB	Standard
login.html	Sep 20, 2018 3:58:46 PM GMT-0500	3.5 KB	Standard
photos.html	Sep 20, 2018 3:58:46 PM GMT-0500	3.5 KB	Standard
register.html	Sep 20, 2018 3:58:47 PM GMT-0500	3.0 KB	Standard
upload.html	Sep 20, 2018 3:58:47 PM GMT-0500	9.9 KB	Standard

### 3. Agregar una política al bucket para permitir acceso al público

Mediante el uso de políticas de acceso de S3, usted puede controlar quién tiene acceso a los archivos, por defecto los buckets no permiten el acceso a usuarios no autenticados, en este caso agregaremos una política que le permita a cualquier usuario anónimo descargar los archivos que se encuentren en el bucket.

1. De la consola de S3 seleccione el bucket creado en la sección anterior
2. Seleccione la pestaña de **Permissions**, luego haga clic en **Block public Access**
3. Haga clic en la opción **Edit**
4. Deshabilite las cuatro opciones que aparecen marcadas como **Recommended**, la ventana debería verse de la siguiente manera:



#### Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, or both. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block *all* public access. These settings apply only to this bucket. 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 your buckets 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)**  
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket policies**  
S3 will block new bucket policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket policies**  
S3 will ignore public and cross-account access for buckets with policies that grant public access to buckets and objects.

Cancel Save

5. Haga clic en **Save**
6. En la ventana de confirmación escriba la palabra “confirm” en el recuadro y haga clic en **confirm**
7. Haga clic en el botón de **Bucket policy**
8. Copie el siguiente JSON en el **Bucket policy editor**, asegúrese de reemplazar **SU\_BUCKET** por el nombre del bucket creado en paso anterior

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "s3:GetObject",  
            "Resource": "arn:aws:s3:::SU_BUCKET/*"  
        }  
    ]  
}
```

La política debe verse de la siguiente manera:

Amazon S3 > saamwanderlust-pedropicapedra

Overview	Properties	Permissions	Management
<a href="#">Access Control List</a>	<a href="#">Bucket Policy</a>	<a href="#">CORS configuration</a>	
Bucket policy editor ARN: arn:aws:s3:::saamwanderlust-pedropicapedra Type to add a new policy or edit an existing policy in the text area below.			
1	{		
2	"Version": "2012-10-17",		
3	"Statement": [		
4	{		
5	"Effect": "Allow",		
6	"Principal": "*",		
7	>Action": "s3:GetObject",		
8	"Resource": "arn:aws:s3:::saamwanderlust-pedropicapedra/*"		
9	}		
10	]		
11	}		

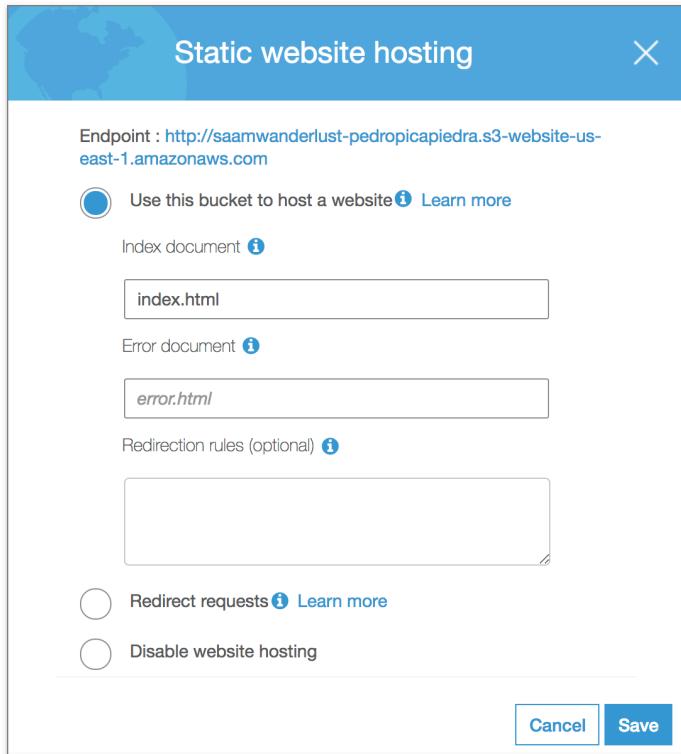
Haga clic en **Save**

## 4. Habilitar el bucket como sitio web

Ahora es necesario habilitar el bucket para poder hospedar sitios web y cualquier usuario desde Internet pueda entrar a la parte estática de nuestra aplicación. Cuando habilitamos un bucket como sitio web podemos definir un nombre de documento como índice, además de que podemos definir un documento por defecto como página de error y contaremos con un URL público para acceder a nuestro sitio.

Es posible utilizar un dominio personalizado como [www.sudominio.com](http://www.sudominio.com) y mediante el uso de Route 53 direccionarlo al bucket publico.

1. De la consola de S3 seleccione el bucket creado en la sección anterior
2. Seleccione la pestaña de **Properties**
3. Haga clic en el recuadro **Static Web Hosting**
4. Seleccione **Use this bucket to host a website** y escriba `index.html` en el campo de **Index document**
5. Copie el URL que viene en la sección de **Endpoint**, guárdelo en un archivo de texto ya que lo utilizaremos posteriormente
6. Haga clic en **Save**



## 5. Agregar la configuración CORS al bucket

Para poder cargar archivos a S3 desde nuestra aplicación web necesitamos agregarle la configuración CORS para permitirle al API de S3 recibir llamadas desde dominios diferentes, aquí le agregaremos los headers Access-Control-Allow-Origin que responderá S3 a cada carga de imágenes.

1. Haga clic en la pestaña de **Permissions**
2. Haga clic en el botón de **CORS configuration**
3. En el área de **CORS Configuration editor** escriba el siguiente código XML

```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>
    <AllowedMethod>HEAD</AllowedMethod>
    <AllowedHeader>*</AllowedHeader>
</CORSRule>
</CORSConfiguration>
```

4. Haga clic en **Save**

## Validación

Pegue el Endpoint que copio anteriormente en el navegador, el URL debe ser similar a este:

**[http://SU\\_BUCKET.s3-website-us-east-1.amazonaws.com](http://SU_BUCKET.s3-website-us-east-1.amazonaws.com)**

Si el sitio web se abre correctamente en el navegador y puede ver la página de "Mis Álbumes" del sitio web, puede continuar con el siguiente módulo

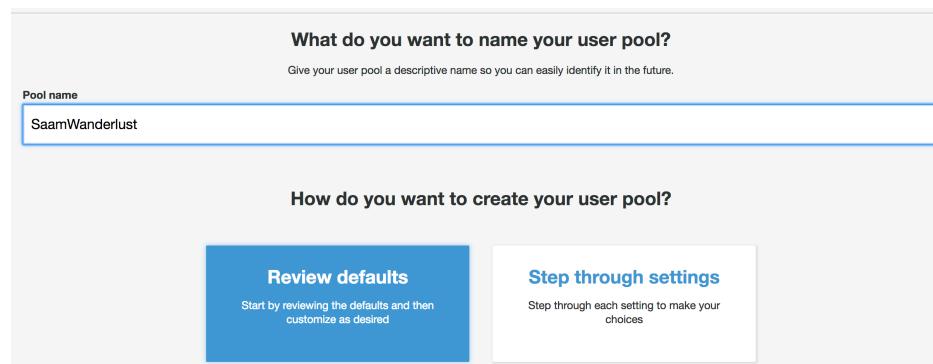
# Módulo 2: Registro y Log-in de usuarios

## Instrucciones de implementación

### 1. Crear un User Pool de Cognito

Para el registro de usuarios y la autenticación con los APIs del backend vamos a utilizar el servicio de Amazon Cognito, con este servicio podremos agregar la funcionalidad de registro y autenticación de usuarios a nuestra aplicación Serverless sin necesidad de desarrollarla desde cero.

1. En la consola de administración de AWS seleccione **Services** y luego seleccione **Cognito** de la categoría de **Security, Identity & Compliance**. También puede escribir Cognito en el buscador y seleccionar la opción del resultado de búsqueda
2. Seleccione la opción de **Manage your User Pools** y haga clic en **Create a user pool**
3. Escriba un nombre para su user pool como **SaamWanderlust** y haga clic en **Review defaults**



4. Haga clic en **Create pool**

- Copie el nuevo **Pool Id** que aparece en la pantalla de confirmación y péguelo en un archivo de texto, lo necesitaremos posteriormente.

Your user pool was created successfully.

Pool Id us-east-1\_tniBvEx9D

## 2. Crear un cliente de aplicación (App Client)

Ahora crearemos un cliente de aplicación desde nuestro User Pool, esto para poder conectar nuestra aplicación web con el servicio de Cognito.

**Importante:** Los Client secrets no están soportados en el SDK de JavaScript, al crear la aplicación debe asegurarse que la opción de Generate Client Secret debe estar desmarcada, de lo contrario la autenticación no va a funcionar. En caso de omitir este paso es necesario borrar la aplicación y crear una nueva.

- De la pantalla de confirmación de la creación del User Pool seleccione la opción de **App clients** dentro de la sección **General Settings**. Si ya salió de esa pantalla, en la sección de User Pools haga clic en el que tiene el nombre que creó en pasos anteriores.
- Haga clic en **Add an App client**
- Escriba un nombre la aplicación como **SaamWanderlustWebApp**
- Asegúrese de que la opción **Generate client secret** no esté seleccionada
- Seleccione **Create app client**

The sidebar menu includes: General settings, Users and groups, Attributes, Policies, MFA and verifications, Advanced security, Message customizations, Tags, Devices, **App clients** (selected), Triggers, Analytics, App integration, App client settings, Domain name, UI customization, Resource servers, Federation, Identity providers, and Attribute mapping.

**Which app clients will have access to this user pool?**

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

**App client name:** SaamWanderlustWebApp

**Refresh token expiration (days):** 30

**Generate client secret**

**Auth Flows Configuration:**

- Enable username password auth for admin APIs for authentication (ALLOW\_ADMIN\_USER\_PASSWORD\_AUTH) [Learn more.](#)
- Enable lambda trigger based custom authentication (ALLOW\_CUSTOM\_AUTH) [Learn more.](#)
- Enable username password based authentication (ALLOW\_USER\_PASSWORD\_AUTH) [Learn more.](#)
- Enable SRP (secure remote password) protocol based authentication (ALLOW\_USER\_SRP\_AUTH) [Learn more.](#)
- Enable refresh token based authentication (ALLOW\_REFRESH\_TOKEN\_AUTH) [Learn more.](#)

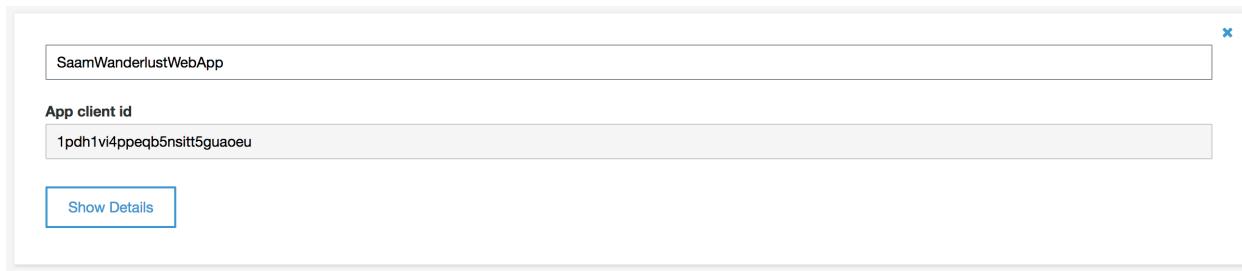
**Prevent User Existence Errors:** [Learn more.](#)

Legacy  
 Enabled (Recommended)

**Set attribute read and write permissions**

**Create app client**

6. De la pantalla de confirmación copie el App client id y péguelo en un archivo de texto, lo ocuparemos más adelante.



### 3. Actualice el archivo de configuración

Vamos a actualizar el archivo de configuración del sitio que se encuentra en el siguiente path:

**/js/config.js**

Descargue el archivo del bucket S3 creado anteriormente, o del archivo .zip si es que siguió este paso en la sección “copiar el sitio”. Abra el archivo para edición y escriba los valores para el **Pool Id** y **App client id** que se generaron en los pasos anteriores, su archivo debe quedar como el siguiente con usando sus valores obtenidos:

```
window._config = {
    cognito: {
        userPoolId: 'us-east-1_uXboG5pAb',
        userPoolClientId: '25ddkmj4v6hfsfvruhpfi7n4hv',
        region: 'us-east-1', // e.g. us-east-1
        identityPoolId: ''
    },
    api: {
        invokeUrl: ''
    },
    s3: {
        bucket : '',
        hostUrl : '',
        uploadPath : 'uploads/'
    },
    dynamodb: {
        tableName: 'SaamWanderlustPhotos'
    }
};
```

Copie el archivo modificado al bucket de S3 reemplazando el que estaba anteriormente, asegúrese de copiar al archivo dentro de la carpeta **/js**.

## Validación

Una vez que haya reemplazado el archivo config.js puede validar que los pasos anteriores hayan sido exitosos, para esto deberá entrar al sitio estático de su aplicación, para esto usaremos el navegador de Internet, copie el URL de su sitio web estático en S3 obtenido del paso 4 del módulo 1. Agregue al URL la página de registro en el navegador como se muestra en el siguiente ejemplo y presione *Enter*

**[http://SU\\_BUCKET.s3-website-us-east-1.amazonaws.com/register.html](http://SU_BUCKET.s3-website-us-east-1.amazonaws.com/register.html)**

Cree un usuario nuevo utilizando una cuenta de correo válida, esta cuenta se utilizará para mandar el código de confirmación del registro. Si no cuenta con una dirección de correo válida tendrá que realizar la confirmación del usuario de forma manual desde la consola de Cognito User Pools.

Siga las indicaciones del registro hasta que el flujo lo redirija a la página de "Mis Álbumes", en esta página si los pasos anteriores se ejecutaron de manera correcta aparecerá un recuadro verde con la información del token actual:

```
Token:eyJraWQiOiJQXC9mTHBFQTVaVXFxZW1mZ01YT2p6Q0tcLzIJamFBaDfMHNIYWhVbFBLVjg9liwiYWxnIjoiUIMyNTYifQ.eJBFbhWGZ8CZHRHuicBqWumft42Fz0zb5GBaKw1-MQWZAvdN2MHJrwq15or5gnu1kOKtJ8RMFkh4H6410rYLeroFY9ZIFbEPYfiypymuC6aEqGZ-LDKBZj3TWN6jOXuQPvSqlqaMjwQj1hdlrgdSYsDo9gNiGzPpHfvRomNhymNsrxcnezm3HXd6n2kHMH4KlmdXBjvMVnYLzs16O5c
```

## Módulo 3: Carga y análisis de fotos

### Instrucciones de implementación

#### 1. Crear un Cognito Identity Pool

Con el Identity pool de Cognito podremos invocar servicios como S3 para subir las fotografías a un bucket y ejecutar los métodos de Rekognition para analizar imágenes.

1. Desde la consola, en la sección de servicios seleccione **Cognito** de la sección de **Security, Identity & Compliance**
2. Haga clic en el botón **Manage Identity Pools** y luego haga clic en **Create new identity pool** si es que no es redireccionado al **Getting started wizard**

3. En la siguiente pantalla escriba **SaamWanderlustIdentityPool** en el campo de **Identity pool name**
4. Expanda la sección de **Unauthenticated identities** y seleccione la opción **Enable Access to unauthenticated identities**
5. Haga clic en **Create Pool**

Create new identity pool

Identity pools are used to store end user identities. To declare a new identity pool, enter a unique name.

Identity pool name\*  (Valid)  
Example: My App Name

▼ Unauthenticated identities ⓘ

Amazon Cognito can support unauthenticated identities by providing a unique identifier and AWS credentials for users who do not authenticate with an identity provider. If your application allows customers to use the application without logging in, you can enable access for unauthenticated identities. [Learn more about unauthenticated identities.](#)

Enable access to unauthenticated identities  
Enabling this option means that anyone with internet access can be granted AWS credentials. Unauthenticated identities are typically users who do not log in to your application. Typically, the permissions that you assign for unauthenticated identities should be more restrictive than those for authenticated identities.

▼ Authentication flow settings ⓘ

A user authenticating with Amazon Cognito will go through a multi-step process to bootstrap their credentials. Amazon Cognito has two different flows for authentication with public providers: enhanced and basic. Cognito recommends the use of enhanced authentication flow. However, if you still wish to use the basic flow, you can enable it here. [Learn more about authorization flows.](#)

Allow Basic (Classic) Flow

▶ Authentication providers ⓘ

\* Required Cancel Create Pool

6. En la siguiente pantalla haga clic en **Allow**
7. En la pantalla de **Getting started with Amazon Cognito** en la sección de **Get AWS Credentials** copie el **identity pool Id** que aparece en rojo y péguelo en un archivo de texto, lo ocuparemos más adelante
8. Haga clic en **Go To Dashboard**

## 2. Crear una tabla de DynamoDB

Vamos a crear una Tabla de DynamoDB desde la consola, en esta tabla vamos a guardar todas la imágenes que se suban desde la aplicación.

- 1 Desde la consola, en la sección de servicios seleccione **DynamoDB** de la sección de **Databases**
- 2 Haga clic en la opción **Create table**
- 3 En el campo de **Table Name** escriba el nombre **SaamWanderlustPhotos**
- 4 En el campo **Partition Key** escriba **userId** y seleccione **String** como tipo de dato
- 5 Haga clic en el checkbox **Add sort key**
- 6 En el campo que aparece escriba **PhotoId** y seleccione **String** como tipo de dato
- 7 Seleccione la opción **Use default settings** y haga clic en **Create**

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*	SaamWanderlustPhotos	<a href="#">?</a>
Primary key*	Partition key	
	userid	Type: String
	<input checked="" type="checkbox"/> Add sort key	
	Photoid	Type: String

**Table settings**

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes.
- Encryption at Rest with DEFAULT encryption type.

[+ Add tags NEW!](#)

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

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

De la página de la nueva Tabla creada, busque el ARN de la misma y cópielo en un archivo de texto, lo utilizaremos más tarde.

### 3. Agregar los permisos de S3 y Rekognition a los roles creados por Cognito

Estos permisos los utilizaremos para autenticar llamados desde la aplicación web hacia los servicios de S3 y DynamoDB al momento de cargar nuevas imágenes.

1. Desde la consola, en la sección de servicios seleccione **IAM** de la sección de **Security, Identity & Compliance**
2. Del menú de la izquierda haga clic en **Roles**
3. En la caja de búsqueda escriba **SaamWanderlust** y seleccione el rol que se llama **Cognito\_SaamWanderlustIdentityPoolUnauth\_Role**, asegúrese que sea el que dice **Unauth** y no **Auth**
4. En la pantalla de **Summary**, en la pestaña **Permissions policies**, haga clic en la opción **Add inline policy** del lado derecho
5. En la pantalla de **Create policy** haga clic en la pestaña de **JSON** y copie el siguiente código de la política, asegúrese de reemplazar el **NOMBRE\_DE\_BUCKET** por el nombre de su bucket y **ARN\_DE\_TABLA\_DE\_DYNAMO** por el ARN que se generó al crear la tabla de DynamoDB, un ejemplo del ARN sería: **arn:aws:dynamodb:us-east-1:234571279823:table/SaamWanderlustPhotos**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1537420073230",
      "Action": [
        "s3:PutObject*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::NOMBRE_DE_BUCKET/*"
    },
    {
      "Sid": "Stmt1537420125923",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Effect": "Allow",
      "Resource": "ARN_DE_TABLA_DE_DYNAMODB"
    },
    {
      "Sid": "Stmt1537420145342",
      "Action": [
        "rekognition:DetectFaces",
        "rekognition:DetectLabels"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

6. Haga clic en ***Review policy***
7. En la siguiente pantalla escriba **SaamWanderlust-CognitoPolicy** en el campo de **Name**, haga clic en ***Create policy***

#### 4. Actualice el archivo de configuración

Vamos a actualizar el archivo de configuración del sitio que se encuentra en el siguiente path:

**/js/config.js**

Descargue el archivo del bucket S3 creado anteriormente, o del archivo .zip si es que siguió este paso en la sección “copiar el sitio”. Abra el archivo para edición y escriba los valores para ***identityPoolId*** y ***bucket*** que se generaron en el paso anterior al crear el Cognito identity pool, su archivo debe quedar como el siguiente usando sus valores obtenidos:

```
window._config = {
  cognito: {
    userPoolId: 'us-east-1_ uXboG5pAb',
    userPoolClientId: '25ddkmj4v6hfsfvruhpfi7n4hv',
    region: 'us-east-1', // e.g. us-east-1
    identityPoolId: 'us-east-1:643eda5c-9039-5b22-2654-8de31aab4ca2'
  },
  api: {
    invokeUrl: ''
  },
  s3: {
    bucket : 'saamwanderlust-pedropicapedra',
    hostUrl : '',
    uploadPath : 'uploads/'
  },
  dynamodb: {
    tableName: 'SaamWanderlustPhotos'
  }
};
```

Copie el archivo modificado al bucket de S3 reemplazando el que estaba anteriormente, asegúrese de copiar al archivo dentro de la carpeta **/js**.

## Validación

Ahora probaremos que la carga de imágenes y la inserción de las nuevas imágenes en la tabla de DynamoDB funcione correctamente.

Escriba en el navegador el URL de la aplicación web:

```
http://saamwanderlust-pedropicapedra.s3-website-us-east-1.amazonaws.com/upload.html
```

En la página haga clic en el botón de Tomar foto, seleccione una fotografía desde su computadora y espere a que la imagen se analice y se cargue. Deberá ver un mensaje que diga **"La foto se cargo correctamente en el álbum: XXX"**



# Módulo 4: Implementar el REST API backend

## Instrucciones de implementación

### 1. Crear un rol de IAM para la función Lambda

Cada función Lambda tiene un rol asociado, este rol es el que define los permisos hacia los diferentes servicios a los que su función Lambda tendrá acceso.

1. De la consola en la opción de Services seleccione **IAM** del área de **Security, Identity & Compliance**
2. En la barra de navegación de la izquierda seleccione **Roles** y luego seleccione **Create new role**
3. Seleccione **Lambda** como el tipo de rol del grupo de AWS Service, en la sección **Common use cases**, haga clic en **Next:permissions**
4. Escriba **AWSLambdaBasicExecutionRole** en el campo de búsqueda y seleccione el rol que aparece como resultado, haga clic en **Next:Tags** y luego en **Next:Review**
5. Escriba **SaamWanderlustLambda** en el campo de **Role name**, evite los espacios en blanco
6. Haga clic en **Create role**
7. De la lista de roles, ubique el rol que acaba de crear, haga clic en el nombre del rol
8. En la pestaña de **Permissions** haga clic en la opción **Add inline policy**

The screenshot shows the 'Permissions' tab selected in the AWS IAM console. A single policy named 'AWSLambdaBasicExecutionRole' is listed under 'Permissions policies (1 policy applied)'. This policy is an 'AWS managed policy'. There is also a note about a 'Permissions boundary (not set)'.

9. Haga clic en ***Choose a service***

10. En el buscador escriba **DynamoDB** y seleccione la opción con ese título

The screenshot shows the 'Actions' section of the AWS IAM Policy Editor. A search bar has 'DynamoDB' typed into it, and the result 'Scan' is selected. Other options like 'DynamoDB' and 'DynamoDBAccelerator' are also visible.

11. En el buscador escriba **Scan** y selecciona la opción con el mismo nombre,

**Actions** Specify the actions allowed in DynamoDB ?

close

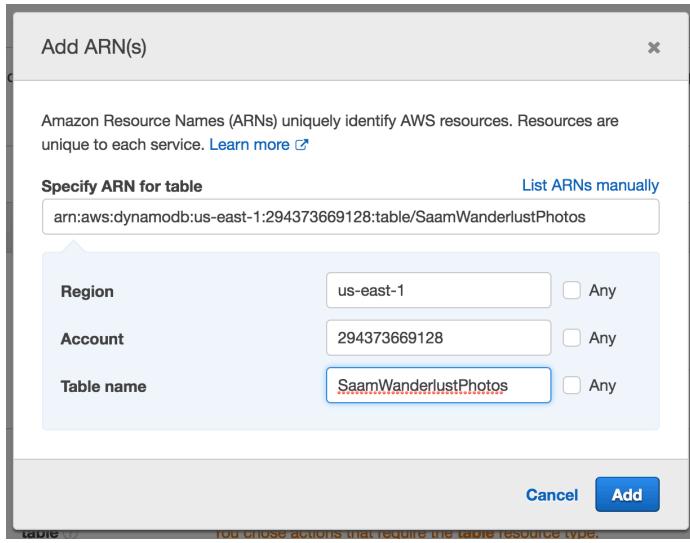
Scan

Scan ?

12. Repita la misma operación ahora con la palabra **Query** y seleccione la acción con el mismo nombre

13. Haga clic en la opción de **Resources**

14. Haga clic en la opción **Add ARN** de **table** (no en **index**) y en el campo **Specify ARN for table** copie el ARN de la tabla de DynamoDB que creó en el paso anterior, haga clic en **Add**



15. Haga clic en ***Review policy***

16. Escriba **SaamWanderlustWritePolicy** en campo de **Name** y haga clic en **Create policy**

#### Review policy

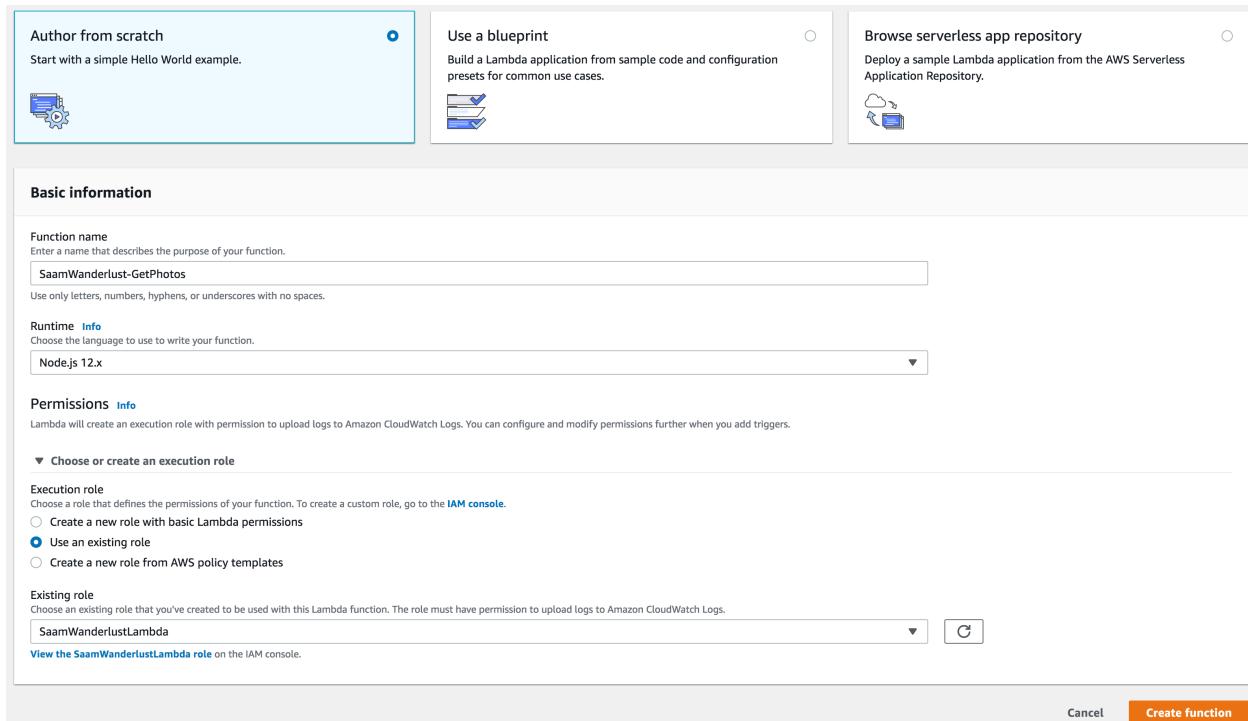
Before you create this policy, provide the required information and review this policy.

Name*	SaamWanderlustWritePolicy										
Maximum 128 characters. Use alphanumeric and '+,-,.,@-_ ' characters.											
Summary	<input type="text"/> Filter <table border="1"> <thead> <tr> <th>Service</th> <th>Access level</th> <th>Resource</th> <th>Request condition</th> </tr> </thead> <tbody> <tr> <td>Allow (1 of 146 services) Show remaining 145</td> <td>DynamoDB</td> <td>Limited: Write</td> <td>TableName   string like   SaamWanderlustPhotos</td> </tr> </tbody> </table>			Service	Access level	Resource	Request condition	Allow (1 of 146 services) Show remaining 145	DynamoDB	Limited: Write	TableName   string like   SaamWanderlustPhotos
Service	Access level	Resource	Request condition								
Allow (1 of 146 services) Show remaining 145	DynamoDB	Limited: Write	TableName   string like   SaamWanderlustPhotos								

## 2. Crear una función Lambda para listar los álbumes y las fotos

La función lambda se encargará de ejecutar los llamados de la aplicación para consultar todas las fotos creadas en la tabla de DynamoDB.

1. Desde la consola en la opción de Services seleccione **Lambda**
2. Haga clic en **Create function**
3. Deje la opción **Author from scratch** seleccionada
4. Escriba **SaamWanderlust-GetPhotos** en el campo **Name**
5. En el campo de **Runtime** en la sección **Other Supported** seleccione **Node.js 12.x**
6. En **Permissions** haga clic en la opción **Choose or create an execution role**
7. En el drop down **Execution Role** seleccione la opción **Use an existing role**
8. En el drop down **Existing role**, seleccione el rol creado anteriormente **SaamWanderlustLambda**
9. Haga clic en **Create function**



10. Abajo en la sección **Fuction code** reemplace todo el código de la pestaña index.js del editor de código por el código de la función lambda que se muestra a continuación (se recomienda copiar el código y pegarlo en algún editor de texto antes para asegurarse que no se copien caracteres especiales):

```
const AWS = require('aws-sdk');
const dynamo = new AWS.DynamoDB.DocumentClient();

exports.handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));
    var params = {
        TableName : "SaamWanderlustPhotos",
        KeyConditionExpression: "#id = :i",
        ExpressionAttributeNames:{
            "#id": "userId"
        },
        ExpressionAttributeValues: {
            ":i": event.params.querystring.uid
        }
    };
    dynamo.query(params, function(err, data) {
        if (err) {
            console.error("Unable to query. Error:", JSON.stringify(err, null, 2));
        } else {
            console.log("Query succeeded.");
            console.log(data);
            callback(null, data);
        }
    });
}
```

```
    });
};
```

11. Asegúrese que en la línea de TableName esté escrito el nombre de la tabla de

Dinamo como la creó, si es necesario modifique esa línea

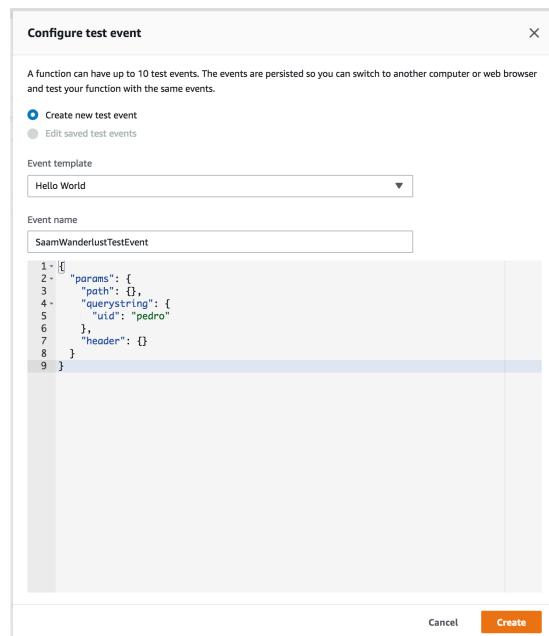
12. Haga clic en **Save** en la esquina superior derecha

## Validación

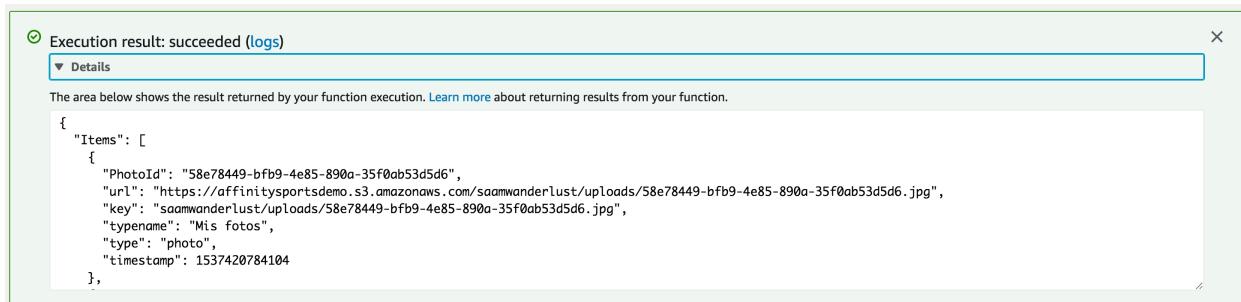
Del menú superior seleccione la lista **Select a test event**, haga clic en **configure test event**, en el campo de **Event name** escriba **SaamWanderlustTestEvent**, en el campo de texto reemplace el código JSON que aparece por el siguiente:

```
{
  "params": {
    "path": {},
    "querystring": {
      "uid": "pedro"
    },
    "header": {}
  }
}
```

Reemplace el nombre “pedro” por el nombre de usuario con el que se registró previamente, el evento de pruebas debe quedar como se muestra a continuación:



Haga clic en **Create**, después haga clic en el botón de **Test**. Aparecerá el resultado de la ejecución como “**succeeded**”, expanda el detalle y podrá ver el resultado de la ejecución que debería ser el detalle de la foto que subió anteriormente.



```
Execution result: succeeded (logs)
▼ Details
The area below shows the result returned by your function execution. Learn more about returning results from your function.

{
  "Items": [
    {
      "PhotoId": "58e78449-bfb9-4e85-890a-35f0ab53d5d6",
      "url": "https://affinitysportsdemo.s3.amazonaws.com/saamwanderlust/uploads/58e78449-bfb9-4e85-890a-35f0ab53d5d6.jpg",
      "key": "saamwanderlust/uploads/58e78449-bfb9-4e85-890a-35f0ab53d5d6.jpg",
      "typename": "Mis fotos",
      "type": "photo",
      "timestamp": 1537420784104
    }
  ]
}
```

### 3. Crear un API para conectarse desde la interfaz web a la función para listar las fotos/álbumes

El end point de API Gateway nos servirá para hacer llamadas AJAX desde la aplicación web hacia el backend construido con Lambda y DynamoDB

1. Desde la consola en la opción de **Services** seleccione **API Gateway** de la sección de **Networking & Content delivery**, si le aparece la opción **Getting started** haga clic en ella
2. En la sección **Choose an API type**, busque la opción **REST API** (sin la palabra private) y haga clic en **Build**, en la ventana de **Create your first API** haga clic en **OK**
3. En la sección **Choose the protocol** seleccione **REST**, en **Create new API** seleccione **New API**
4. Escriba **SaamWanderlust** en el campo de **API Name**, deje el resto de los valores por defecto
5. Haga clic en **Create API**

Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

REST     WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

New API     Import from Swagger or Open API 3     Example API

Settings

Choose a friendly name and description for your API.

API name\*

SaamWanderlust

Description

(empty field)

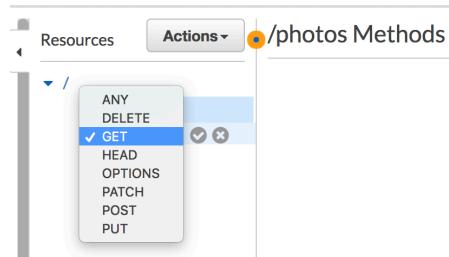
Endpoint Type

Regional



**Create API**

6. En el botón de Action seleccione la opción **Create resource**
7. En el campo de **Resource name** escriba **photos**
8. Haga clic en **Create resource**
9. Seleccione de la sección de Resources el recurso **/photos** nuevamente del botón de Actions seleccione la opción **Create Method**
10. Aparecerá un nuevo campo de lista abajo del recurso, seleccione el método **GET** de la lista y haga clic en la pequeña paloma gris que está a un lado del campo



11. En la siguiente pantalla del método de integración, deje seleccionada la opción **lambda function** en **Integration type**
12. En el campo de **Lambda function** escriba **SaamWanderlust-GetPhotos**

Choose the integration point for your new method.

The screenshot shows the 'Create Integration' dialog. Under 'Integration type', the 'Lambda Function' option is selected. Below it, there are other options: 'HTTP', 'Mock', 'AWS Service', and 'VPC Link'. Under 'Lambda Function', the text 'SaamWanderlustUploadPhoto' is entered. There are also fields for 'Lambda Region' (set to 'us-east-1') and 'Use Default Timeout' (with a checked checkbox).

13. Si aparece una pantalla de confirmación para agregar permisos a API Gateway para ejecutar la función Lambda, haga clic en **Ok**
14. En el diagrama de flujo de **Method Execution** haga clic en el recuadro de **Integration Request**
15. En la parte inferior del **Integration Request**, identifique la sección **Mapping Templates** en la parte inferior, haga clic en la flecha para expandir esa sección
16. En la opción de **Request body passthrough**, haga clic en la segunda opción: **When there are no templates defined (recommended)**
17. Haga clic en la opción de **Add mapping template**
18. En el campo de texto escriba **application/json**

19. Haga clic en la paloma para guardar los cambios
20. Del campo **Generate template** que aparece más abajo seleccione la opción **Method Request passthrough**
21. El campo se llenará con el código JSON para hacer el mapeo

▼ Mapping Templates 

Request body passthrough  When no template matches the request Content-Type header   
 When there are no templates defined (recommended)   
 Never 

Content-Type	
application/json	

 [Add mapping template](#)

application/json

Generate template: Method Request passthrough 

```

1 ## See http://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-mapping-template-reference.html
2 ## This template will pass through all parameters including path, querystring, header, stage variables, and context through to the integration endpoint via the body/payload
3 #set($allParams = $input.params())
4 {
5   "body-json" : $input.json('$'),
6   "params" : {
7     #foreach($type in $allParams.keySet())
8       #set($params = $allParams.get($type))
9       "$type" : {
10         #foreach($paramName in $params.keySet())
11           "$paramName" : '$util.escapeJavaScript($params.get($paramName))"
12           #if($foreach.hasNext),#end
13         #end
14       }
15     #if($foreach.hasNext),#end
16   #end
}

```

[Cancel](#) [Save](#)

22. Haga clic en Save
23. Para validar que la integración con Lambda está funcionando correctamente haga clic en el botón de test de la pantalla de **Method Execution**



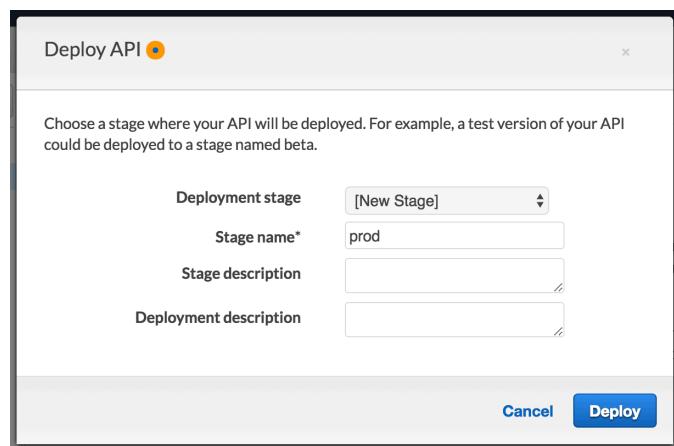
24. En la sección de **Query Strings** en el campo debajo de **{photos}** escriba **uid=nombre-de-usuario**, donde nombre-de-usuario es el usuario con el que se registró en la web app previamente
25. Nuevamente haga clic en el botón **Test**, En el campo de **Response Body** aparecerá un JSON con la información de la foto que cargó anteriormente
26. De nuevo seleccione de la sección de **Resources** el recurso **/photos**, del botón de **Actions** seleccione la opción **Enable CORS**
27. Haga clic en la opción **Enable CORS and replace existing CORS headers**
28. Aparecerá una ventana de confirmación sobre los cambios a ejecutarse, haga clic en **Yes, replace existing values**

29. Aparecerá una lista de los métodos agregados marcados con una paloma verde, además en el recurso **/photos** se agregó un método **OPTIONS**

The screenshot shows the AWS API Gateway resource configuration for the '/photos' endpoint. Under the 'Actions' section, the 'OPTIONS' method is listed with a green checkmark next to it, indicating successful configuration. The 'Method Request' section shows 'GET' and 'OPTIONS' methods defined. The 'Integration Response' section lists several successful steps: 'Create OPTIONS method', 'Add 200 Method Response with Empty Response Model to OPTIONS method', 'Add Mock Integration to OPTIONS method', 'Add 200 Integration Response to OPTIONS method', 'Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers to OPTIONS method', 'Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings to OPTIONS method', 'Add Access-Control-Allow-Origin Method Response Header to GET method', and 'Add Access-Control-Allow-Origin Integration Response Header Mapping to GET method'. A note at the bottom states: 'Your resource has been configured for CORS. If you see any errors in the resulting output above please check the error message and if necessary attempt to execute the failed step manually via the Method Editor.'

30. Vuelva a hacer clic en el botón de **Actions**, ahora haga clic en la opción **Deploy API**

31. En el campo **Deployment stage**, seleccione **[New Stage]**, en **Stage name** escriba **prod**



32. Haga clic en **Deploy**

33. En la pantalla de confirmación copie el URL que aparece en la sección **Invoke URL** y péguelo en un archivo de texto, lo ocuparemos más adelante

The screenshot shows the 'prod Stage Editor' interface. It includes a 'Stages' list with 'prod' selected, a 'Create' button, and a main area with a light blue background containing the 'Invoke URL' information.

#### 4. Actualice el archivo de configuración

Vamos a actualizar el archivo de configuración del sitio que se encuentra en el siguiente path:

**/js/config.js**

Descargue el archivo del bucket S3 creado anteriormente, o del archivo .zip si es que siguió este paso en la sección "copiar el sitio". Abra el archivo para edición y escriba

los valores para el ***invokeUrl*** y ***hostUrl*** que se generaron en los pasos anteriores al crear el bucket de S3 y el invocation URL de API Gateway, en el caso del ***invokeUrl*** debe asegurarse de agregarle al final **/photos**, su archivo debe quedar como el siguiente ejemplo usando sus valores obtenidos:

```
window._config = {
  cognito: {
    userPoolId: 'us-east-1_uXboG5pAb',
    userPoolClientId: '25ddkmj4v6hfsfvruhpfi7n4hv',
    region: 'us-east-1', // e.g. us-east-1
    identityPoolId: 'us-east-1:643eda5c-9039-5b22-2654-8de31aab4ca2'
  },
  api: {
    invokeUrl: 'https://rc3nyt7x.execute-api.us-west-2.amazonaws.com/prod/photos'
  },
  s3: {
    bucket : 'saamwanderlust-pedropicapedra',
    hostUrl : 'http://saamwanderlust-pedropicapedra.s3-website-us-east-1.amazonaws.com',
    uploadPath : 'uploads/'
  },
  dynamodb: {
    tableName: 'SaamWanderlustPhotos'
  }
};
```

Copie el archivo modificado al bucket de S3 reemplazando el que estaba anteriormente, asegúrese de copiar al archivo dentro de la carpeta **/js**.

## Validación

Para validar el funcionamiento del backend entraremos a la aplicación web, nos autenticaremos y listaremos las imágenes que cargamos anteriormente.

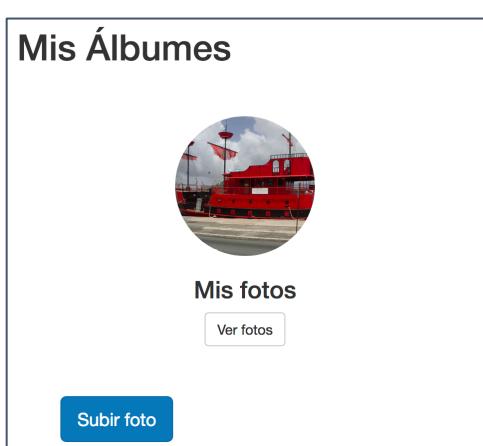
*Nota: Se recomienda borrar caché del navegador, y de preferencia cerrar completamente del navegador y abrirlo nuevamente, esto para asegurarse que la sesión anterior se elimine y podamos entrar a la aplicación con una sesión completamente nueva. Otra opción es abrir el siguiente URL para cerrar la sesión:*

**`http://saamwanderlust-pedropicapedra.s3-website-us-east-1.amazonaws.com/logout.html`**

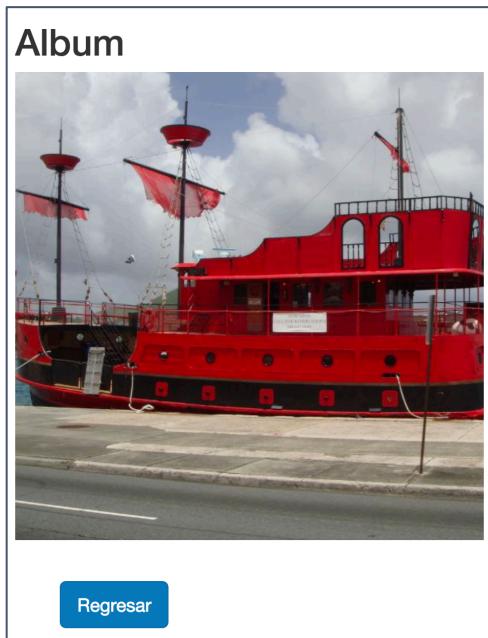
Escriba en el navegador el URL de la aplicación web:

**`http://saamwanderlust-pedropicapedra.s3-website-us-east-1.amazonaws.com`**

La aplicación debe redireccionarlo automáticamente a la página de Login, escriba su nombre de usuario y contraseña que creó anteriormente. Si la información de Login es correcta la aplicación redirigirá a la página de ***Mis Álbumes***, ahí deberá ver la imagen que cargó anteriormente.



Haga clic en la opción ***Ver fotos***, ahora aparecerá la página donde se muestra el listado de todas las fotos cargadas en ese álbum como se muestra a continuación:



## 5. Configurar la autenticación de los llamados del API a través de Cognito User Pool

El end point de API Gateway nos servirá para hacer llamadas AJAX desde la aplicación web hacia el backend construido con Lambda y DynamoDB

1. Desde la consola en la opción de **Services** seleccione **API Gateway** de la sección de **Networking & Content delivery**
2. Haga clic en el API **SaamWanderlust**
3. De las opciones que de despliegan, seleccione **Authorizers**
4. Haga clic en **+ Create New Authorizer**
5. En el campo de **Name** escriba **SaamWanderlustAuthorizer**
6. En Type marque la opción de **Cognito**
7. En el campo de **Cognito User Pool**, asegúrese que la región es correcta y seleccione el Pool con el nombre **SaamWanderlust**
8. En el campo de **Token source**, escriba **Authorization**

**Create Authorizer**

**Name \***  
SaamWanderlustAuthorizer

**Type \* ⓘ**  
 Lambda  Cognito

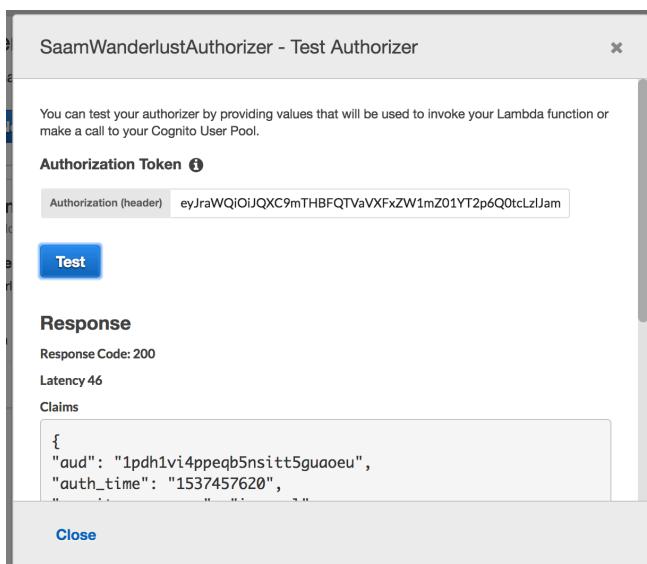
**Cognito User Pool \* ⓘ**  
us-east-1  SaamWanderlust

**Token Source \* ⓘ**  
Authorization

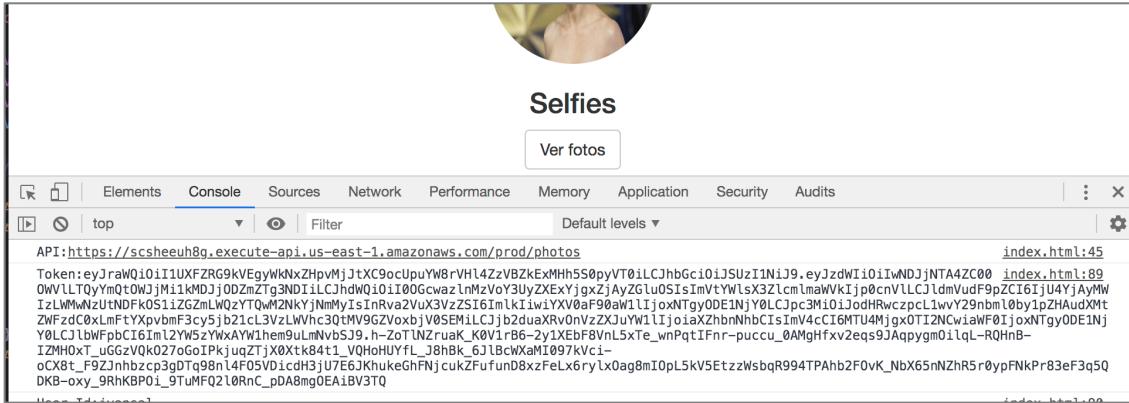
**Token Validation ⓘ**

**Create** **Cancel**

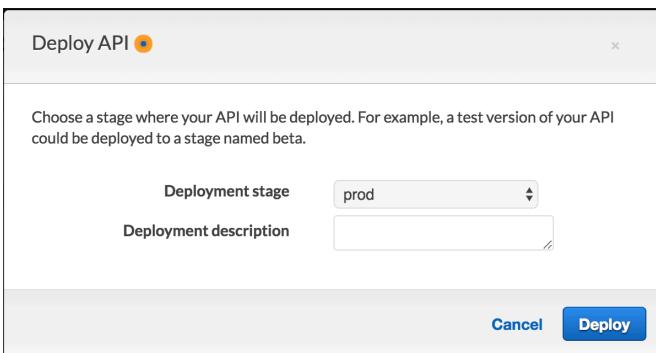
9. Haga clic en **Create**
10. Puede probar el authorizer haciendo clic en **Test** y pegando el Token que aparece en la consola de Javascript del navegador cada vez que hace Login desde la aplicación



El token se puede obtener abriendo la consola de Javascript del browser



11. Si la autorización funciona correctamente verá un mensaje **Response code: 200** mostrando los datos del usuario
12. Haga clic en **Close**
13. De las opciones de la izquierda haga clic en **Resources**, luego haga clic en el recurso **/photos** y seleccione el método **GET**
14. Haga clic en la opción de **Method Request**
15. En la sección de Settings haga clic en el ícono de un lápiz para seleccionar el Authorizer con el nombre **SaamWanderlustAuthorizer**, Nota: Si no le aparece la opción en la lista recargue la página completa y vuelva a intentar, la opción debería aparecer ahora.
16. Haga clic en la pequeña paloma que está a un lado del campo
17. En la pantalla de GET – Method Execution, vuelva a hacer clic en el botón de **Actions**, y haga clic en la opción **Deploy API**
18. En el campo **Deployment stage**, seleccione **prod**



19. Haga clic en **Deploy**

## Validación

Para validar el funcionamiento del backend entraremos a la aplicación web, de preferencia cerrando el browser e iniciando una nueva sesión. Ahora cada vez que se

consulten los álbumes y las fotografías el API Gateway tomará el token de sesión y lo usará para autenticar los llamados.

## 6. Comprobación de la capacidad de tener múltiples usuarios registrados en la aplicación

La aplicación web está diseñada para que se puedan registrar múltiples usuarios y cada uno vea sus fotografías, vamos a validar esto creando un usuario nuevo en la aplicación y subiremos nuevas imágenes:

1. Cierre el navegador de internet para asegurarse que la cookie de sesión se elimine, también puede borrar las cookies del navegador o abra la página de **logout.html** paraerrar la sesión.
2. Vuelva a ejecutar el navegador que estaba utilizando en las pruebas
3. Tecleé de nuevo el URL de su aplicación web, esto deberá redireccionarlo a la página de **Login**, haga clic en la opción **¿Usuario nuevo? Regístrate aquí**
4. Registre un nuevo usuario con un correo electrónico diferente, si no cuenta con una cuenta de correo adicional, escriba cualquier correo y realice la confirmación de la cuenta de forma manual desde la consola de **Cognito User Pools**
5. Una vez que se haya registrado, e iniciado sesión, aparecerá la pantalla de **Mis álbumes**, sin ningún álbum creado
6. Suba algunas imágenes, en esta nueva cuenta
7. Cierre de nuevo el navegador y vuélvalo a abrir
8. Inicie sesión con la cuenta anterior y verifique que siga viendo las imágenes que cargó anteriormente

## 7. Configurar la autenticación del Identity Pool con el token de sesión del User Pool

Actualmente la página de upload.html utiliza las credenciales del usuario no autenticado de **Cognito Identity Pool**, esto quiere decir que cualquier usuario anónimo podría subir archivos al bucket de **S3** o hacer modificaciones a las tablas de **DynamoDB**, esto es inseguro y debe modificarse.

A continuación modificaremos la política de acceso para el usuario autenticado del Identity Pool creando una política administrada en lugar de un **inline policy**, le asignaremos los permisos que tenía el usuario no autenticado y se los removeremos al mismo, luego configuraremos el User Pool como **Authentication provider** del identity pool para mandar el token de sesión válido y permitir la carga de archivos a S3.

1. Desde la consola en la opción de **Services** seleccione **IAM** de la sección de **Networking & Content delivery**
2. Del menú de la izquierda haga clic en **Policies** y haga clic en el botón **Create policy**

3. En la pantalla de **Create policy** haga clic en la pestaña de **JSON** y copie el siguiente código de la política, asegúrese de reemplazar el **NOMBRE\_DE\_BUCKET** por el nombre de su bucket y **ARN\_DE\_TABLA\_DE\_DYNAMO** por el ARN que se generó al crear la tabla de DynamoDB

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "Stmt1537420073230",  
      "Action": [  
        "s3:PutObject"  
      ],  
      "Effect": "Allow",  
      "Resource": "arn:aws:s3:::NOMBRE_DE_BUCKET/*"  
    },  
    {  
      "Sid": "Stmt1537420125923",  
      "Action": [  
        "dynamodb:DescribeTable",  
        "dynamodb:GetItem",  
        "dynamodb:PutItem",  
        "dynamodb:Query",  
        "dynamodb:Scan"  
      ],  
      "Effect": "Allow",  
      "Resource": "ARN_DE_TABLA_DE_DYNAMODB"  
    },  
    {  
      "Sid": "Stmt1537420145342",  
      "Action": [  
        "rekognition:DetectFaces",  
        "rekognition:DetectLabels"  
      ],  
      "Effect": "Allow",  
      "Resource": "*"  
    }  
  ]  
}
```

4. Haga clic en **Review policy**  
5. En la siguiente pantalla escriba **SaamWanderlust-CognitoPolicy** en el campo de **Name**, haga clic en **Create policy**  
6. Del menú de la izquierda haga clic en **Roles**  
7. En la caja de búsqueda escriba **SaamWanderlust** y seleccione el rol que se llama **Cognito\_SaamWanderlustIdentityPoolAuth\_Role**, asegúrese que sea el que dice **Auth** y no **Unauth**  
8. En la pantalla de **Summary**, en la pestaña **Permissions policies**, haga clic en la el botón **Attach policies**

9. Expanda el menú **Filter policies** y marque la opción **Customer managed**, en la lista aparecerá la política que acabamos de crear con el nombre **SaamWanderlust-CognitoPolicy**, selecciónela y haga clic en **Attach policy**
10. La sección de **Permissions** se verá de la siguiente manera:

The screenshot shows the 'Permissions' tab selected in the top navigation bar. Below it, a table lists two policies: 'SaamWanderlust-CognitoPolicy' (Managed policy) and 'oneClick\_Cognito\_SaamWanderlustIdentityPoolAuth\_Role\_1582780129067' (Inline policy). There is also a note about 'Permissions boundary (not set)'.

11. Del menú de la izquierda haga clic en **Roles**
12. En la caja de búsqueda escriba **SaamWanderlust** y seleccione el rol que se llama **Cognito\_SaamWanderlustIdentityPoolUnauth\_Role**, asegúrese que sea el que dice **Unauth** y no **Auth**
13. En la pestaña de **Permissions** elimine la **Inline policy** con el nombre **SaamWanderlust-CognitoPolicy** haciendo clic en la pequeña "x" que se encuentra del lado derecho
14. En la ventana de confirmación haga clic en **Remove**
15. Desde la consola, en la sección de servicios seleccione **Cognito** de la sección de **Security, Identity & Compliance**
16. Haga clic en el botón **Manage Identity Pools** y luego haga clic en el pool con el nombre **SaamWanderlustIdentityPool**
17. Haga clic en la opción **Edit identity pool** en la esquina superior derecha
18. Expanda la sección **Authentication providers**
19. En la pestaña de **Cognito** escriba el **User Pool Id** y el **App client id** que generamos en los módulos anteriores, la sección debe quedar como se muestra:

The screenshot shows the 'Authentication providers' section. It includes fields for 'User Pool ID' (us-east-1\_FeZin5tHC) and 'App client id' (48g0k9g35hcu2eq1b81f02din9). Below these, there's a note about 'Authenticated role selection' and a dropdown menu for 'Use default role'. At the bottom, there's a link to 'Add Another Provider'.

**20.**Haga clic en *Save changes*

## 8. Actualice el archivo de configuración

Vamos a actualizar el archivo de carga de fotografías del sitio que se encuentra en el siguiente path:

**/upload.html**

Descargue el archivo del bucket S3 creado anteriormente, o del archivo .zip si es que siguió este paso en la sección “copiar el sitio”. Abra el archivo para edición y localice la siguiente sección en el código (líneas 23 a 31)

```
23      // Modulo 7
24      //var cognitoSessionMap = {};
25      //cognitoSessionMap[ 'cognito-idp.' + _config.cognito.region + '.amazonaws.com/' + _config.cognito.userPoolId ] = sessionToken;
26
27      AWS.config.region = _config.cognito.region;
28      AWS.config.credentials = new AWS.CognitoIdentityCredentials({
29          IdentityPoolId: _config.cognito.identityPoolId //,
30          //Logins: cognitoSessionMap
31      });

```

Descomente las líneas comentadas con “//” para que el código quede de la siguiente manera:

```
23      // Modulo 7
24      var cognitoSessionMap = {};
25      cognitoSessionMap[ 'cognito-idp.' + _config.cognito.region + '.amazonaws.com/' + _config.cognito.userPoolId ] = sessionToken;
26
27      AWS.config.region = _config.cognito.region;
28      AWS.config.credentials = new AWS.CognitoIdentityCredentials({
29          IdentityPoolId: _config.cognito.identityPoolId,
30          Logins: cognitoSessionMap
31      });

```

Copie el archivo modificado al bucket de S3 reemplazando el que estaba anteriormente, asegúrese de copiar al archivo dentro de la carpeta raíz del bucket, donde se encuentran todos archivos HTML.

## Validación

Para validar la autenticación de usuarios a través del User Pool este funcionando cierre la sesión haciendo clic en el siguiente URL (utilice el URL de su aplicación):

**<http://saamwanderlust-pedropicapedra.s3-website-us-east-1.amazonaws.com/logout.html>**

Escriba en el navegador el URL de la aplicación web:

**<http://saamwanderlust-pedropicapedra.s3-website-us-east-1.amazonaws.com>**

La aplicación debe re-direccionarlo automáticamente a la página de Login, escriba su nombre de usuario y contraseña que creó anteriormente.

Haga clic en el botón **Subir foto** y luego en botón **Tomar foto**, la aplicación deberá cargar la foto al álbum correspondiente sin ningún contratiempo, la única diferencia es que ahora **Cognito Identity Pool** está validando el token de sesión del usuario contra el **User Pool**, nosotros le pasamos el token de sesión de la siguiente manera:

Logins: 'cognito-idp.us-east-1.amazonaws.com/us-east-1\_FeZ1n5tHC:[Token]'

## **¡Felicidades! ha concluido el workshop**

Esto concluye el workshop, si terminó antes que el resto de la clase, podrá solicitar los retos adicionales al laboratorio al instructor.

---