

INTEGRACIÓN DE AMAZON SQS CON UNA APLICACIÓN WEB ESTÁTICA EN AMAZON S3

Uno de los objetivos del módulo de “Desarrollo Web en Entorno Cliente” es la utilización de mecanismos de comunicación asíncrona para el envío y recepción de información.

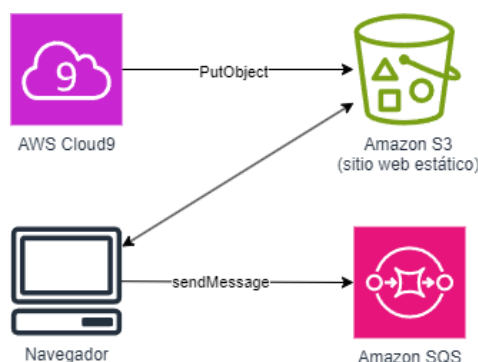
Amazon Simple Queue Service (SQS) es un servicio de mensajería completamente administrado que facilita el desacoplamiento de los diferentes componentes de una aplicación. Esto significa que podemos enviar, almacenar y recibir mensajes entre servicios o aplicaciones, sin necesidad de que estos interactúen directamente entre sí. Por otro lado, Amazon S3 es un servicio de almacenamiento de objetos que nos permite alojar contenido web de forma segura, escalable y de bajo costo.

En esta práctica, exploraremos cómo implementar una cola de Amazon SQS y cómo acceder a ella desde una aplicación web estática alojada en un bucket de Amazon S3, que se desarrollará en un entorno de AWS Cloud9. Esta integración permitirá que nuestra aplicación web envíe mensajes a la cola SQS utilizando el AWS SDK de JavaScript para el navegador, lo que es útil en numerosos escenarios de arquitectura de aplicaciones distribuidas.

Requerimientos:

- Disponer de acceso a los recursos de AWS a través de un *sandbox* de AWS Academy

Arquitectura propuesta:



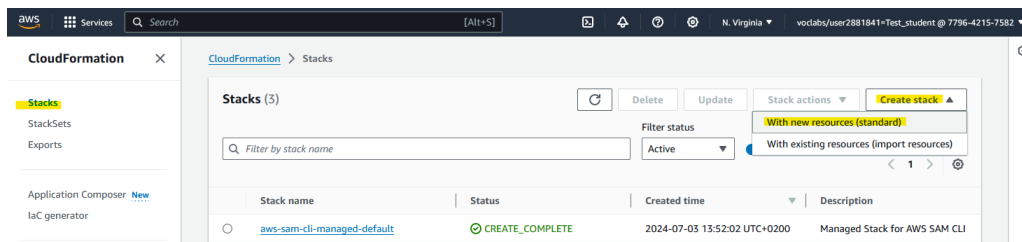
Realización:

DESPLIEGUE DE LA INFRAESTRUCTURA NECESARIA

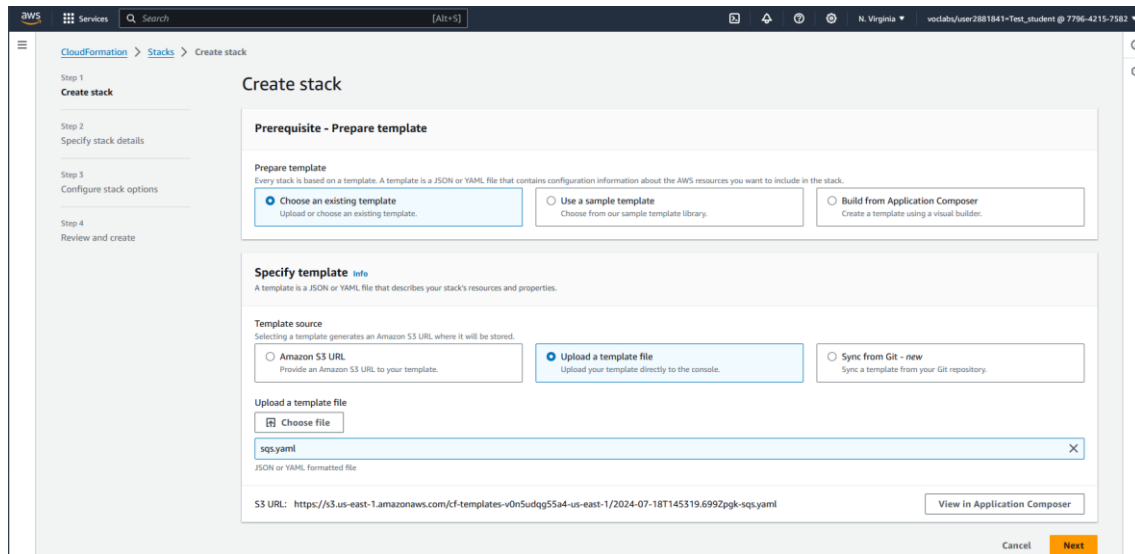
- 1) En primer lugar, hay que desplegar la infraestructura del entorno de desarrollo en AWS Cloud9. Como el objetivo de esta práctica no es el despliegue de la infraestructura, se utilizará una plantilla de AWS CloudFormation que puede obtenerse desde el siguiente enlace:

<https://raw.githubusercontent.com/jose-emilio/aws-academy-fp-daw/main/resources/sqs/sqs.yaml>

- 2) A continuación, abrimos la consola del servicio de AWS CloudFormation y presionamos en el menú lateral la opción **Stacks** y desde el botón **Create stack** seleccionamos la opción **With new resources (standard)**.

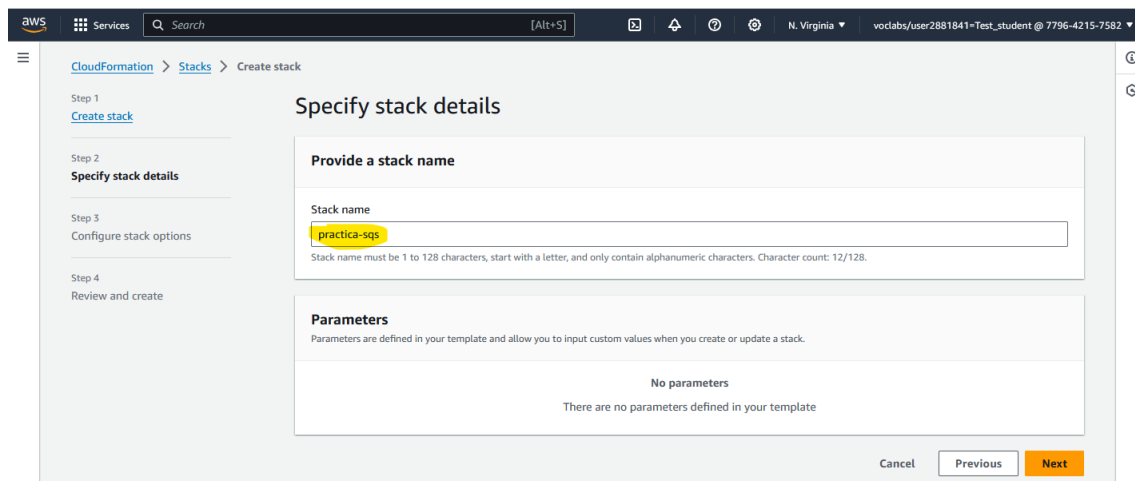


- 3) Desde el asistente siguiente, en la sección **Prerequisite – Prepare template** se elige la opción **Choose an existing template**. En el apartado **Specify template**, se elige la opción **Upload a template file** y se presiona el botón **Choose file** para elegir el archivo *sqs.yaml* descargado en el apartado 1):



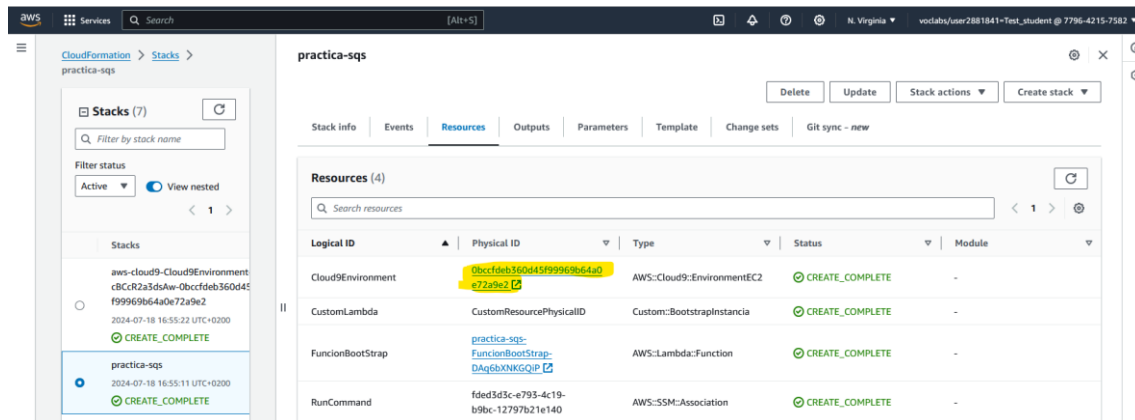
Por último, se presiona el botón **Next**

- 4) En el paso 2 del asistente, introducimos el nombre de la pila de recursos que creará AWS CloudFormation. Indicamos en el campo **Stack name** el valor *practica-sqs* y presionamos el botón **Next**:

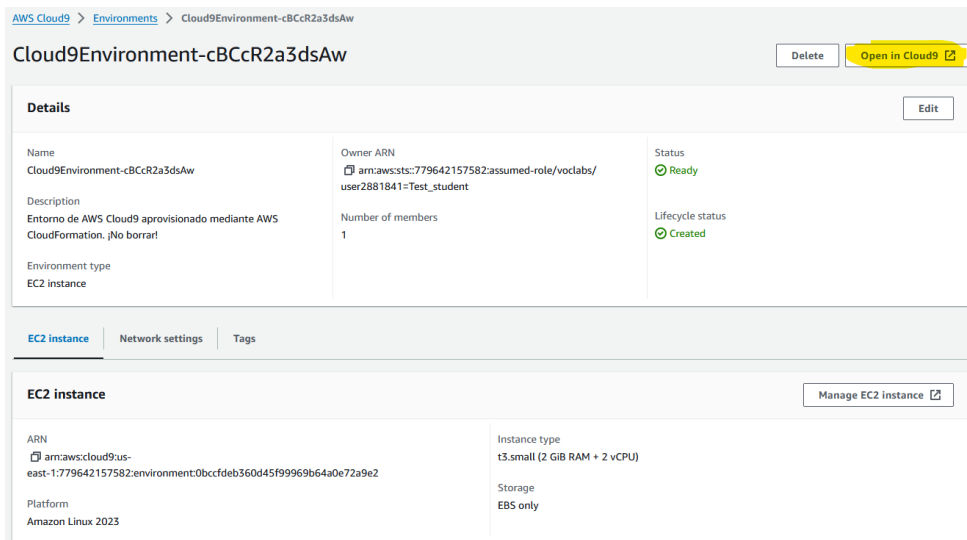


- 5) En el paso 3 del asistente, dejamos los valores por defecto y presionamos el botón **Next**.

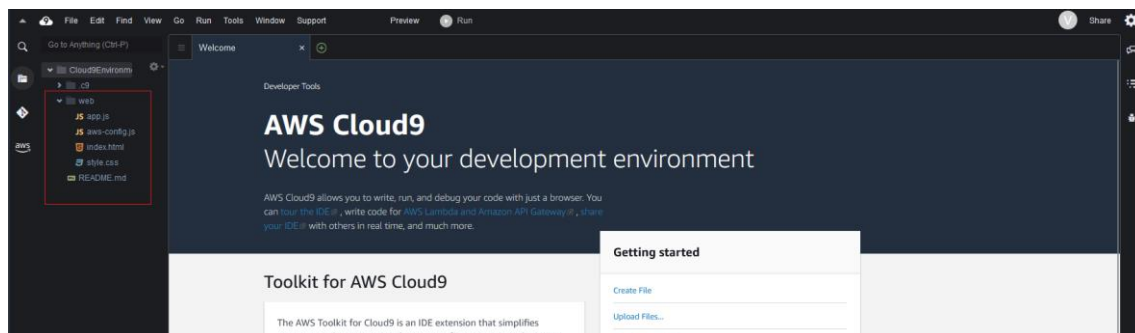
- 6) En el paso final, presionamos el botón **Submit** que se encuentra al final de la página. Tras un par de minutos, la pila ya estará creada. Presionamos la pestaña **Resources** y podremos visualizar la infraestructura que se ha creado como parte del proceso. Entre los recursos se debe encontrar el entorno de desarrollo aprovisionado en AWS Cloud9. Seguimos el enlace marcado con el ratón:



- 7) Desde la siguiente ventana, se pueden visualizar las características del entorno. Presionamos el botón **Open in Cloud9** para acceder al IDE:

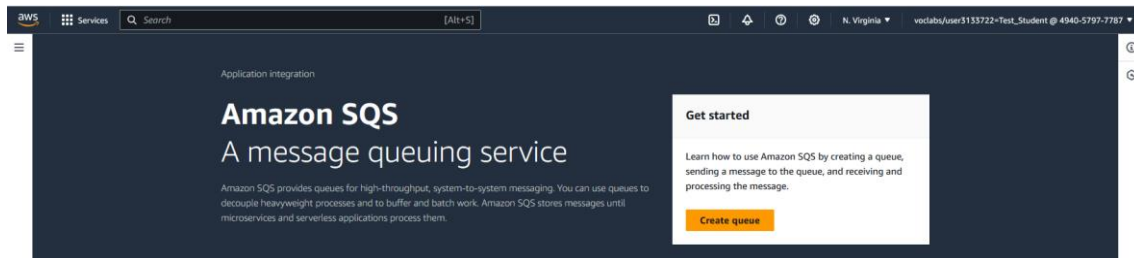


- 8) Si todo ha ido bien, podremos visualizar el entorno de AWS Cloud9, donde podremos comprobar que se ha creado una carpeta llamada **web** con el contenido de nuestra aplicación web estática:



CREACIÓN DE LA COLA DE AMAZON SQS

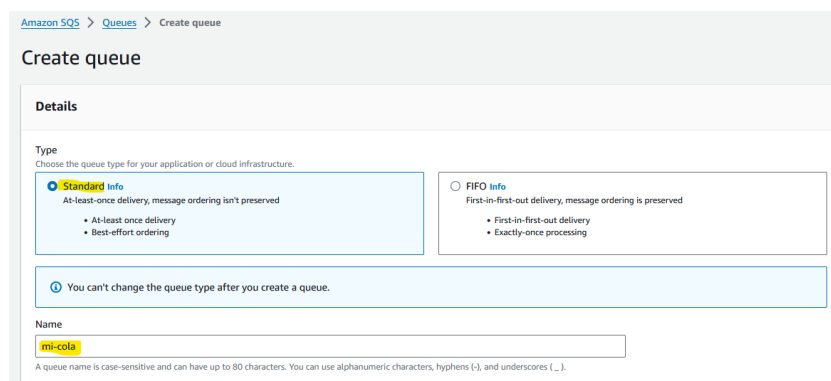
- 9) Para crear nuestra cola de mensajes, accedemos a la consola del servicio Amazon SQS y presionamos el botón **Create queue**:



- 10) Las colas de Amazon SQS pueden ser de dos tipos:

- Standard.** Garantizan la entrega de los mensajes de al menos una vez (por debe implementarse idempotencia en la lógica del consumidor del mensaje). Los mensajes se ordenan según un algoritmo *best effort*, por lo que no está garantizada. La productividad de la cola es virtualmente ilimitada
- FIFO.** Garantizan la entrega de los mensajes una única vez. Los mensajes se entregan ordenadamente según una estrategia FIFO. La productividad de la cola está limitada a 300 operaciones *ReceiveMessage* por segundo (hasta un total de 3000 mensajes por segundo, asumiendo que en cada operación se recupera un lote máximo de 10 mensajes)

Indicaremos el tipo *Standard* e introduciremos el nombre *mi-cola* en el campo **Name**



A continuación, configuraremos los parámetros de la cola:

- Visibility timeout.** Indica el número de segundos que, tras una operación *ReceiveMessage*, el mensaje estará invisible para el resto de los consumidores de la cola. Este parámetro debería ser lo suficientemente grande para permitir que el consumidor pueda procesar el mensaje y realizar una operación *DeleteMessage* para eliminarlo de la cola.
- Message retention period.** Indica el tiempo que un mensaje podrá permanecer en la cola antes de que se elimine automáticamente de la cola.
- Delivery delay.** Indica el tiempo que un mensaje, inmediatamente tras ser introducido en la cola en una operación *SendMessage*, estará invisible para los consumidores.
- Maximum message size.** Tamaño máximo de un mensaje en la cola. Puede ser hasta 256 KBytes.
- Receive message wait time.** Indica, por defecto, el tiempo que un consumidor esperará tras realizar una operación *ReceiveMessage* a recuperar los mensajes de la cola. Un valor de 0 indica

que el servicio Amazon SQS encuestará a un subconjunto de servidores que alojan los mensajes de la cola, por lo que hay riesgo de obtener respuestas vacías o falsas respuestas vacías. Un valor mayor que 0 indicará que el servicio Amazon SQS podrá encuestar un subconjunto de servidores mayor (a mayor tiempo, mayor será el subconjunto), y por tanto se reducirá el riesgo de respuestas vacías, legítimas o falsas.

En nuestra cola configuraremos los valores indicados en la imagen

Configuration [Info](#)
Set the maximum message size, visibility to other consumers, and message retention.

Visibility timeout [Info](#)
10 Seconds
Should be between 0 seconds and 12 hours.

Message retention period [Info](#)
1 Days
Should be between 1 minute and 14 days.

Delivery delay [Info](#)
0 Seconds
Should be between 0 seconds and 15 minutes.

Maximum message size [Info](#)
256 KB
Should be between 1 KB and 256 KB.

Receive message wait time [Info](#)
5 Seconds
Should be between 0 and 20 seconds.

Dejamos el resto de los parámetros por defecto y presionamos el botón **Create queue** al final de la ventana

- 11) Tras ello, podremos comprobar que nuestra cola se ha creado satisfactoriamente. A continuación, realizaremos una prueba de envío y recepción de mensajes desde la consola. Para ello, presionamos el botón **Send and receive messages**:

Amazon SQS > Queues > mi-cola

mi-cola [Edit](#) [Delete](#) [Purge](#) [Send and receive messages](#) [Start DLQ redrive](#)

Details [Info](#)

Name	Type	ARN
mi-cola	Standard	arn:aws:sqs-east-1:494057977787:mi-cola
Encryption	URL	Dead-letter queue
Amazon SQS key (SSE-SQS)	https://sqs.us-east-1.amazonaws.com/494057977787/mi-cola	-

[More](#)

- 12) En la siguiente ventana, en el apartado **Send message**, introduciremos en el campo **Message body** el valor *Prueba de mensaje* y, desplegando el menú **Message attributes**, añadimos los atributos indicados en la imagen:

Send and receive messages
Send messages to and receive messages from a queue.

Send message [Info](#) [Clear content](#) [Send message](#)

Message body
Enter the message to send to the queue.
Prueba de mensaje

Delivery delay [Info](#)
0 Seconds
Should be between 0 seconds and 15 minutes.

Message attributes - Optional [Info](#)

Entorno	String	Custom type	Producción
Prioridad	String	Custom type	Alta

[Add new attribute](#) [Remove](#)

Por último, presionamos el botón **Send Message**. Envía adicionalmente 3-4 mensajes adicionales a la cola con diferentes cuerpos y atributos.

- 13) A continuación, vamos a encuestar a la cola para recuperar los mensajes de ella. En la sección **Receive messages**, podemos visualizar el número de mensajes disponibles en la cola (*Messages available*). Previamente, presionamos el botón **Edit poll settings** para configurar el proceso de encuesta de la cola:

The screenshot shows the 'Receive messages' interface. At the top, there are buttons for 'Edit poll settings', 'Stop polling', and 'Poll for messages'. Below these, a summary section shows 'Messages available' as 4, 'Polling duration' as 30, and 'Maximum message count' as 10. A 'Polling progress' bar shows 0% completion. Below this is a search bar for messages and a table with columns: ID, Sent, Size, and Receive count. The table is currently empty, with a message 'No messages. To view messages in the queue, poll for messages.' and a 'Poll for messages' button at the bottom.

- 14) Configuramos un tiempo de encuesta de 5 segundos y un lote de mensajes a recuperar de 2, tal como muestra la figura y presionamos el botón **Save**:

The 'Edit poll settings' dialog box is shown. It has two input fields: 'Polling duration' set to 5 seconds and 'Maximum message count' set to 2. Below each field is a note: 'Should be between 1 and 999.' At the bottom of the dialog is a 'Save' button.

- 15) Presionamos el botón **Poll for messages** y podremos comprobar que se recuperan dos de los mensajes de la cola:

The screenshot shows the 'Receive messages' interface after polling. The 'Poll for messages' button is now highlighted. The summary section shows 'Messages available' as 4, 'Polling duration' as 5, and 'Maximum message count' as 2. The 'Polling progress' bar now shows 2 receives/second with a green checkmark. Below the search bar, the table now contains two messages:

ID	Sent	Size	Receive count
394358ac-1e68-47f2-9b53-cb614a67019a	2024-07-19T09:22+02:00	60 bytes	1
32348ead-bffc-4afc-b855-7df767cb292c	2024-07-19T09:23+02:00	86 bytes	1

- 16) Cada mensaje tiene un campo **ID** que lo identifica; además hay un campo **Receive count** que indica el número de veces que el mensaje se ha recuperado (se ha procesado y no se ha eliminado de la cola). Si presionamos sobre el enlace de uno de los mensajes, podremos visualizar su contenido y atributos:

Message: 394358ac-1e68-47f2-9b53-cb614a67019a

Body

Attributes

Details

Prueba de mensaje

Done

Attributes (2)

Name	Type	Value
Entorno	String	Producción
Prioridad	String	Alta

Done

Message: 394358ac-1e68-47f2-9b53-cb614a67019a

Body

Attributes

Details

ID 394358ac-1e68-47f2-9b53-cb614a67019a	Size 60 bytes	MD5 of message body 3d8ca9243be56b2da8712b141a8ed102	Sender account ID AROAXGCBGO652ZR7GQDWN:user3133722=Test_Student
Sent 2024-07-19T09:22+02:00	First received 2024-07-19T09:33+02:00	Receive count 1	Message attributes count 2
Message attributes size 43 bytes	MD5 of message attributes 5142def25d4e4b3b0c32cd128244f011		

Done

17) Un valor alto en el campo **Receive count** de un mensaje puede implicar varias causas:

- El umbral del tiempo de visibilidad del mensaje es demasiado bajo y por tanto el consumidor no tiene tiempo de procesar el mensaje y eliminarlo de la cola.
- El mensaje puede tener un formato erróneo (mensaje envenenado) y el consumidor falla una y otra vez al procesar el mensaje.

Para el funcionamiento correcto de la aplicación y evitar este tipo de problemas, es posible crear otra cola de mensajes no procesados (DLQ, *Dead-Letter Queue*) y, cuando el valor del campo **Receive count** supere un umbral especificado, reconducir el mensaje a la cola DLQ para un procesamiento posterior.

Para configurar una cola DLQ, vamos de nuevo a la consola de Amazon SQS y, desde el menú lateral **Queues** presionamos el botón **Create queue**:

18) En la ventana de configuración de la cola, indicamos como tipo el valor *Standard* y en el campo **Name** el valor *mi-cola-dlq*. El resto de los parámetros los dejamos en su valor por defecto y presionamos el botón **Create queue** al final de la ventana:

Amazon SQS > Queues > Create queue

Create queue

Details

Type
Choose the queue type for your application or cloud infrastructure.

☒ **Standard** Info
 At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

☐ **FIFO** Info
 First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

You can't change the queue type after you create a queue.

Name
mi-cola-dlq

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

- 19) De nuevo, volviendo a la consola de Amazon SQS en la sección **Queues** del menú lateral podremos visualizar las dos colas creadas. Seleccionamos la cola llamada *mi-cola* y presionamos el botón **Edit**:

Amazon SQS > Queues

Queues (2) Edit Delete Send and receive messages Actions Create queue

Search queues by prefix

Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based deduplication
mi-cola	Standard	2024-07-19T09:16+02:00	4	0	Amazon SQS key (SSE-SQS)	-
mi-cola-dlq	Standard	2024-07-19T09:47+02:00	0	0	Amazon SQS key (SSE-SQS)	-

- 20) En la sección **Dead-letter queue**, seleccionamos la opción **Enabled** y elegimos la cola *mi-cola-dlq* como destino de los mensajes no procesados y establecemos en 3 el valor del campo **Maximum receives**, de forma que, si el valor **ReceiveCount** de un mensaje supera el valor 3, se sacará de la cola original y se introducirá en la cola DLQ (manteniendo su ID original y su valor **ReceiveCount**):

Dead-letter queue - Optional Info
Send undeliverable messages to a dead-letter queue.

Set this queue to receive undeliverable messages.

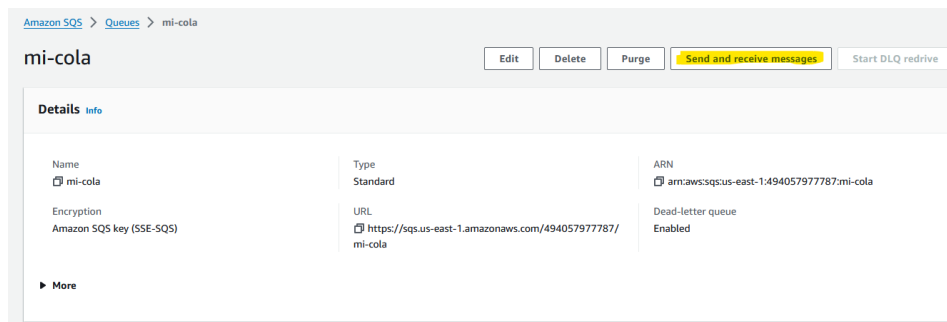
☐ Disabled
☒ **Enabled**

Choose queue
arn:aws:sqs:us-east-1:494057977787:mi-cola-dlq

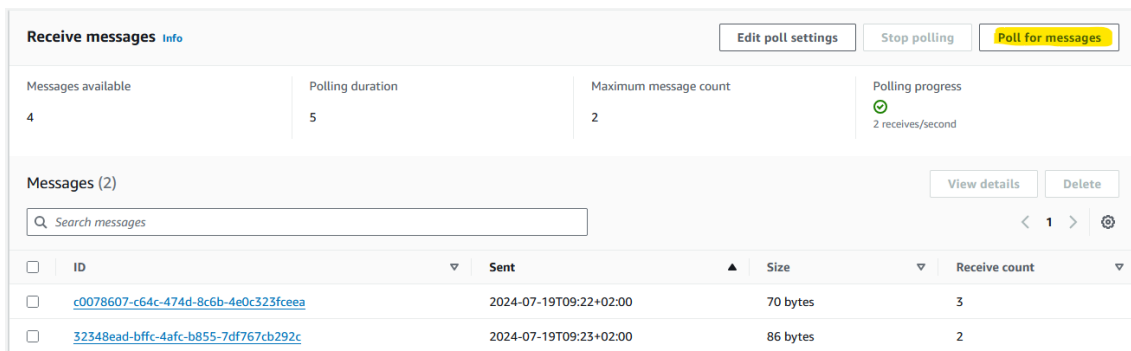
Maximum receives
3
Should be between 1 and 1000

Por último, presionamos el botón **Save** para confirmar los cambios en la configuración.

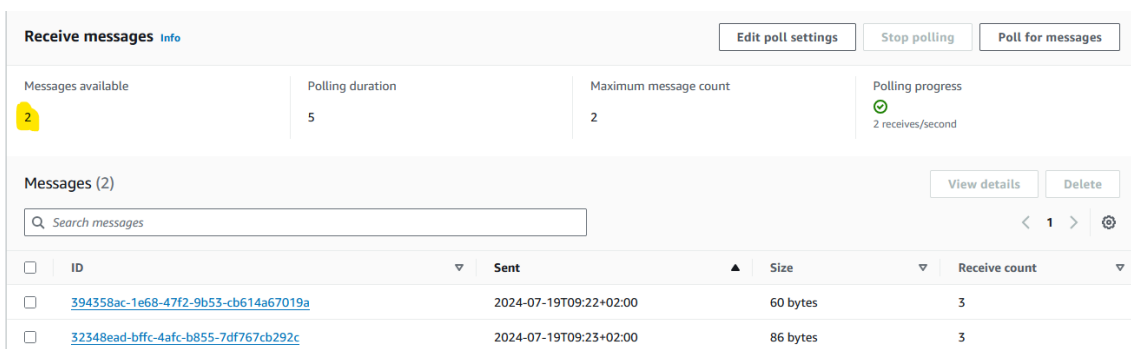
- 21) A continuación, comprobaremos los cambios realizando sucesivas operaciones de recepción de mensajes para comprobar el funcionamiento de la configuración de la DLQ. Para ello, presionamos el botón **Send and receive messages** de la cola *mi-cola*:



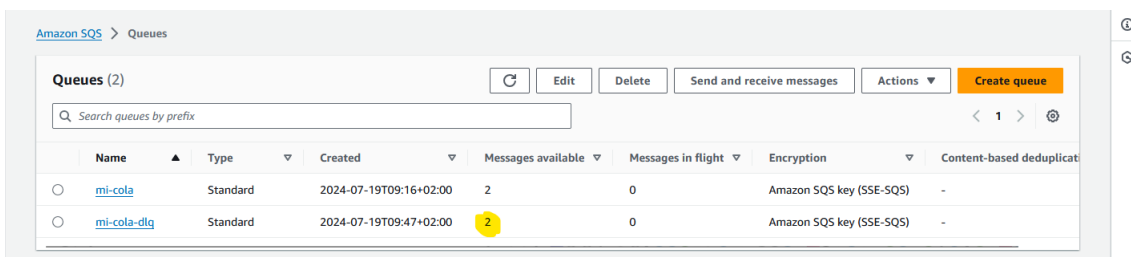
- 22) Desde la sección **Receive messages**, presionamos el botón **Poll for messages** sucesivas veces para realizar encuestas a la cola. Comprueba que, en cada operación de recuperación, los valores del campo **Receive count** van incrementándose a medida que se realizan dichas operaciones:



- 23) Eventualmente, el número de mensajes disponible (**Messages available**) irá decreciendo, ya que los mensajes se derivarán a la DLQ:



- 24) Para terminar de comprobar el funcionamiento, volvemos a la consola de Amazon SQS y comprobamos que la cola *mi-cola-DLQ* tiene mensajes disponibles:



Si accedemos a la cola *mi-cola-dlq* y presionamos el botón **Send and receive messages** e intentamos encuestar a la cola, comprobaremos que se reciben los mensajes:

Receive messages <small>info</small>				Edit poll settings	Stop polling	Poll for messages
Messages available	Polling duration	Maximum message count	Polling progress			
2	5	2	2 receives/second			
Messages (2)				View details Delete		
<input type="text" value="Search messages"/>						
	ID	Sent	Size	Receive count		
<input type="checkbox"/>	c0078607-c64c-474d-8c6b-4e0c323fcea	2024-07-19T09:22+02:00	70 bytes	4		
<input type="checkbox"/>	a35274ce-af7b-4794-8c58-caab2b2c817e	2024-07-19T09:23+02:00	87 bytes	4		

ACCESO A LA COLA DE AMAZON SQS MEDIANTE PROGRAMACIÓN DE LADO DE CLIENTE

En este apartado, implementaremos una aplicación web estática (HTML, CSS y JavaScript) que acceda a la cola de mensajes de Amazon SQS mediante JavaScript. La aplicación estará alojada en un bucket de Amazon S3.

- 25) Para ello, volvemos a la pestaña del entorno de AWS Cloud9 aprovisionado anteriormente (véase apartado 8)). Desde allí hacemos doble clic sobre el archivo `aws-config.js` para configurar nuestra aplicación:

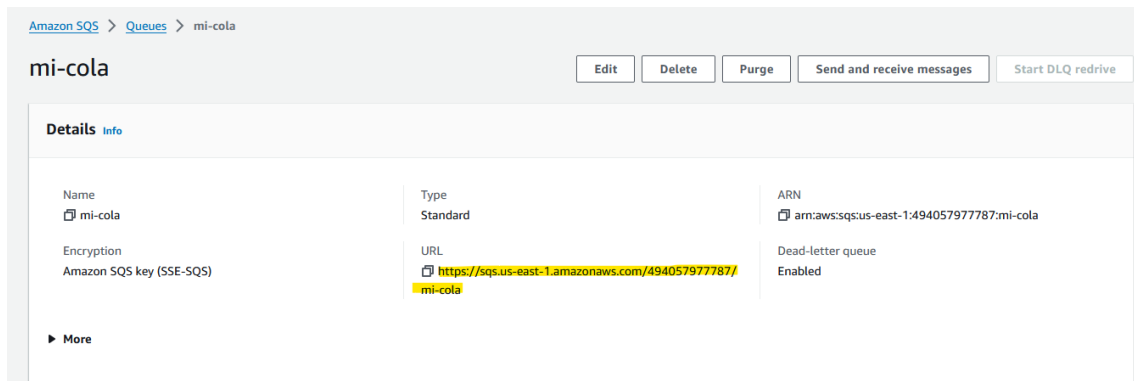
```

1  const awsConfig = {
2    // Cambia esto a tu región, sólo puede ser us-east-1 o us-west-2, las regiones permitidas en los AWS Academy Learner Labs
3    region: 'us-east-1',
4    // Sustituye por tu Access Key ID del AWS Academy Learner Lab
5    accessKeyId: 'ID-Clave-Acceso',
6    // Sustituye por tu Secret Access Key del AWS Academy Learner Lab
7    secretAccessKey: 'Clave-Acceso-Secreta',
8    // Sustituye por tu Session Token del AWS Academy Learner Lab
9    sessionToken: 'Token-Sesion',
10   // Sustituye por tu URL de la cola de SQS creada
11   queueUrl: 'URL-cola'
12 };
13
14 AWS.config.update({
15   region: awsConfig.region,
16   accessKeyId: awsConfig.accessKeyId,
17   secretAccessKey: awsConfig.secretAccessKey,
18   sessionToken: awsConfig.sessionToken
19 });
20

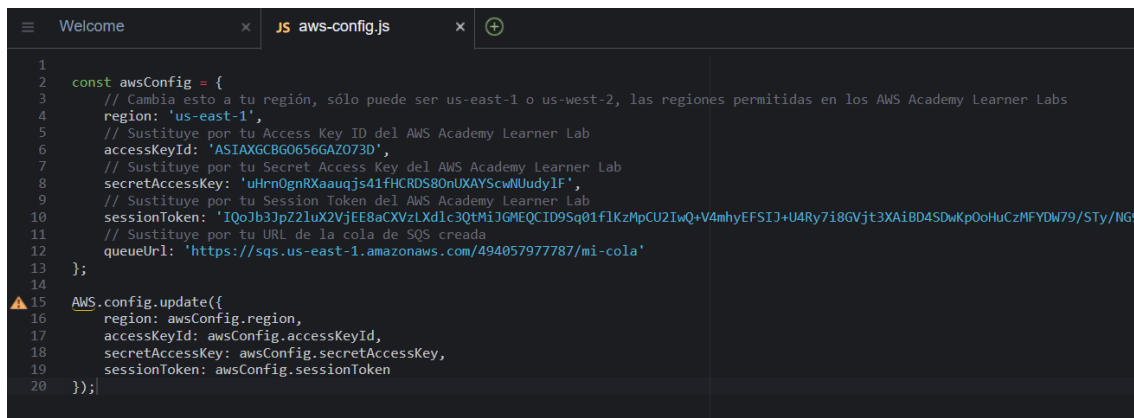
```

- 26) Debemos sustituir los parámetros `accessKeyId`, `secretAccessKey` y `sessionToken` por las credenciales del laboratorio. Estas credenciales se utilizarán para darle permisos a nuestra aplicación en JavaScript para acceder a la cola de Amazon SQS mediante el AWS SDK de JavaScript para el navegador. Estos parámetros los podemos obtener del AWS Academy Learner Lab, accediendo al botón **AWS Details** y posteriormente presionando el botón **AWS CLI**:

El parámetro **queueURL** lo podemos obtener de la consola de Amazon SQS:



Copiamos dichos valores y los sustituimos en el editor del IDE de AWS Cloud9:



Por último, salvamos los cambios del fichero mediante el menú **File / Save**.

NOTA IMPORTANTE: Configurar credenciales de acceso a los servicios de AWS de esta forma es una muy mala práctica, ya que se estarían exponiendo públicamente en el sitio web. La forma más segura sería utilizando un grupo de identidades de Amazon Cognito y acceder desde nuestro código a dicho grupo de identidades para obtener las credenciales de acceso temporales dinámicamente desde nuestra aplicación. Desafortunadamente, en el momento de la elaboración de esta práctica, el servicio de Amazon Cognito Identity Pools no está disponible en los AWS Academy Learner Labs.

27) Si abrimos el archivo *index.html*, podremos ver que se trata de un formulario formado por:

- Un *textarea* para introducir el cuerpo del mensaje
- Dos *textbox* para introducir el nombre y el valor del atributo
- Un botón de tipo **button** para agregar más atributos al mensaje
- Un botón de tipo **submit** para procesar la lógica de envío del mensaje a la cola

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Enviar Mensaje a SQS</title>
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10  <h1>Enviar Mensaje a SQS</h1>
11  <form id="messageForm">
12    <label for="messageBody">Cuerpo del Mensaje:</label>
13    <textarea id="messageBody" required></textarea>
14
15    <h3>Atributos</h3>
16    <div id="attributesContainer">
17      <div class="attribute">
18        <input type="text" class="key" placeholder="Clave">
19        <input type="text" class="value" placeholder="Valor">
20        <button type="button" class="removeAttribute">Eliminar</button>
21      </div>
22    </div>
23    <button type="button" id="addAttribute">Agregar Atributo</button>
24    <button type="submit">Enviar Mensaje</button>
25  </form>
26
27  <script src="https://sdk.amazonaws.com/js/aws-sdk-2.283.1.min.js"></script>
28  <script src="aws-config.js"></script>
29  <script src="app.js"></script>
30 </body>
31 </html>

```

Las últimas marcas *script* permiten que nuestra aplicación utilice las dependencias del AWS SDK de JavaScript para el navegador.

- 28) Si abrimos el archivo *app.js*, podremos ver la lógica asociada a los dos botones del formulario. La lógica del botón **Submit** permite obtener el cuerpo y los atributos del mensaje. A continuación, instancia un cliente para el servicio de Amazon SQS y envía el mensaje mediante la llamada *sendMessage*:

```

1 document.getElementById('messageForm').addEventListener('submit', function(e) {
2   e.preventDefault();
3
4   const messageBody = document.getElementById('messageBody').value;
5   const attributesContainer = document.getElementById('attributesContainer');
6   const attributeElements = attributesContainer.getElementsByClassName('attribute');
7
8   let messageAttributes = {};
9
10  for (let attributeElement of attributeElements) {
11    const key = attributeElement.getElementsByClassName('key')[0].value;
12    const value = attributeElement.getElementsByClassName('value')[0].value;
13
14    if (key && value) {
15      messageAttributes[key] = {
16        DataType: 'String',
17        StringValue: value
18      };
19    }
20  }
21
22  const params = {
23    MessageBody: messageBody,
24    QueueUrl: awsConfig.queueUrl,
25    MessageAttributes: messageAttributes
26  };
27
28  sqs = new AWS.SQS();
29
30  sqs.sendMessage(params, function(err, data) {
31    if (err) {
32      console.error("Error", err);
33    } else {
34      console.log("Success", data.MessageId);
35      alert('Mensaje enviado con éxito!');
36    }
37  });
38 });
39

```

El código asociado al botón de tipo **Button** permite añadir nuevos campos de tipo *textbox* al formulario para introducir más atributos al mensaje:

```

40 document.getElementById('addAttribute').addEventListener('click', function() {
41   const attributesContainer = document.getElementById('attributesContainer');
42   const attributeDiv = document.createElement('div');
43   attributeDiv.className = 'attribute';
44
45   attributeDiv.innerHTML = `
46     <input type="text" class="key" placeholder="Clave">
47     <input type="text" class="value" placeholder="Valor">
48     <button type="button" class="removeAttribute">Eliminar</button>
49   `;
50
51   attributesContainer.appendChild(attributeDiv);
52
53   const removeButtons = attributesContainer.getElementsByClassName('removeAttribute');
54   for (let button of removeButtons) {
55     button.addEventListener('click', function() {
56       this.parentElement.remove();
57     });
58   }
59 });

```

- 29) A continuación, vamos a crear un bucket de Amazon S3 para alojar los ficheros del sitio web. Para ello, desde la terminal de AWS Cloud9 introducimos el siguiente comando sustituyendo las 'X' por números aleatorios (no puede haber dos buckets de Amazon S3 con el mismo nombre a nivel global):

```
$ aws s3 mb s3://practica-sqs-XXXXXXX
```

```

aws - "ip-172-31-16-131.€ x" (+)
voclabs:~/environment $ aws s3 mb s3://practica-sqs-0123456
make_bucket: practica-sqs-0123456
voclabs:~/environment $

```

- 30) A continuación, transferimos los ficheros al bucket de Amazon S3, mediante la siguiente orden, sustituyendo las 'X' por el número elegido en el apartado anterior:

```
$ aws s3 sync web/ s3://practica-sqs-XXXXXXX
```

```

bash - "ip-172-31-16-131. x" (+)
voclabs:~/environment $ aws s3 sync web/ s3://practica-sqs-0123456
upload: web/style.css to s3://practica-sqs-0123456/style.css
upload: web/index.html to s3://practica-sqs-0123456/index.html
upload: web/app.js to s3://practica-sqs-0123456/app.js
upload: web/aws-config.js to s3://practica-sqs-0123456/aws-config.js
voclabs:~/environment $

```

- 31) A continuación, habilitaremos el bucket de Amazon S3 como sitio web estático, para ello ejecutamos la orden, sustituyendo las 'X' por el número elegido:

```
aws s3 website s3://practica-sqs-XXXXXX --index-document index.html
```

```

bash - "ip-172-31-16-131. x" (+)
voclabs:~/environment $ aws s3 website s3://practica-sqs-0123456 --index-document index.html
voclabs:~/environment $

```

- 32) Lo siguiente que debe realizarse es deshabilitar (temporalmente) el bloqueo del acceso público del bucket para poder posteriormente asociar una política de bucket. Para ello, ejecutamos la orden siguiente, sustituyendo las 'X' por el valor correspondiente:

```
$ aws s3api put-public-access-block --bucket practica-sqs-XXXXXXX --public-access-block-configuration BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
```

```
bash - "ip-172-31-16-131.x"
voclabs:~/environment $ aws s3api put-public-access-block --bucket practica-sqs-0123456 --public-access-block-configuration BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false
voclabs:~/environment $
```

- 33) A continuación, establecemos la política del bucket para permitir el acceso a los objetos de nuestro sitio web estático, mediante la siguiente instrucción, sustituyendo las 'X' por los valores correspondientes:

```
$ aws s3api put-bucket-policy --bucket practica-sqs-XXXXXXX --policy '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::practica-sqs-XXXXXXX/*"
    }
  ]
}'
```

```
bash - "ip-172-31-16-131.x"
voclabs:~/environment $ aws s3api put-bucket-policy --bucket practica-sqs-0123456 --policy '{
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Effect": "Allow",
>       "Principal": "*",
>       "Action": "s3:GetObject",
>       "Resource": "arn:aws:s3:::practica-sqs-0123456/*"
>     }
>   ]
> }'
```

- 34) Por último, volvemos a habilitar el bloqueo del acceso público el bucket de S3 mediante la siguiente orden, sustituyendo las 'X' por los valores correspondientes:

```
$ aws s3api put-public-access-block --bucket practica-sqs-XXXXXXX --public-access-block-configuration BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true
```

```
bash - "ip-172-31-16-131.x"
voclabs:~/environment $ aws s3api put-public-access-block --bucket practica-sqs-0123456 --public-access-block-configuration BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true
voclabs:~/environment $
```

- 35) Finalmente, obtenemos la URL del sitio web estático, accediendo a la consola del servicio Amazon S3, seleccionando el bucket *practica-sqs-XXXXXXX*:

Amazon S3

Buckets

Account snapshot - updated every 24 hours

General purpose buckets | Directory buckets

General purpose buckets (2)

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
cf-templates-zowrasangw9s-us-east-1	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 19, 2024, 08:54:35 (UTC+02:00)
practica-sqs-0123456	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 19, 2024, 11:00:25 (UTC+02:00)

Desde la pestaña **Properties**, en el apartado **Static website hosting** podremos visualizar la URL del sitio web estático:

Static website hosting

Use this bucket to host a website or redirect requests.

Static website hosting
Enabled

Hosting type
Bucket hosting

Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket.

<http://practica-sqs-0123456.s3-website-us-east-1.amazonaws.com>

36) Accediendo a la URL anterior desde un navegador, podremos visualizar el frontend de nuestra aplicación, desde donde podremos introducir el cuerpo y los atributos de nuestros mensajes

Cuerpo del Mensaje:

Bienvenido a mi aplicación web

Atributos

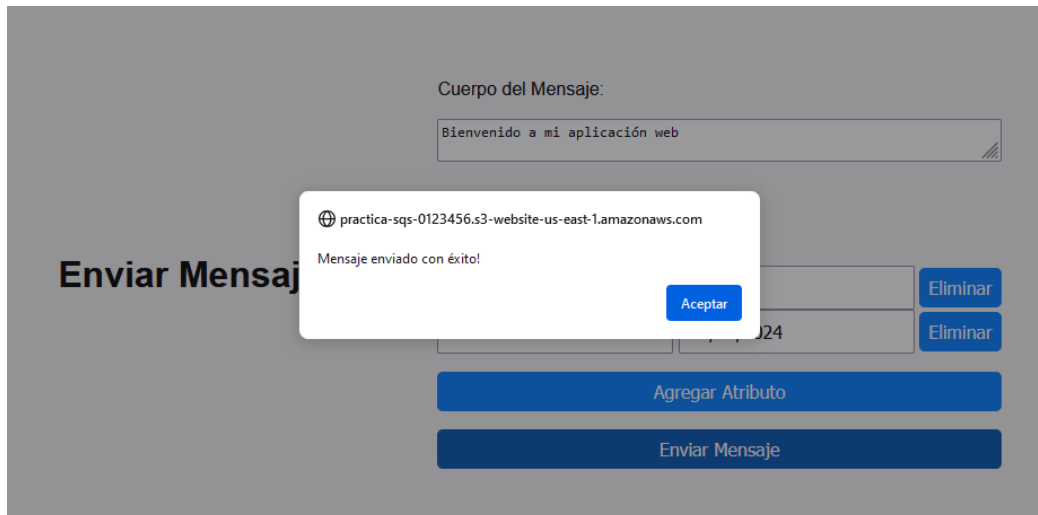
Enviar Mensaje a SQS

Prioridad	Baja	Eliminar
Fecha	19/07/2024	Eliminar

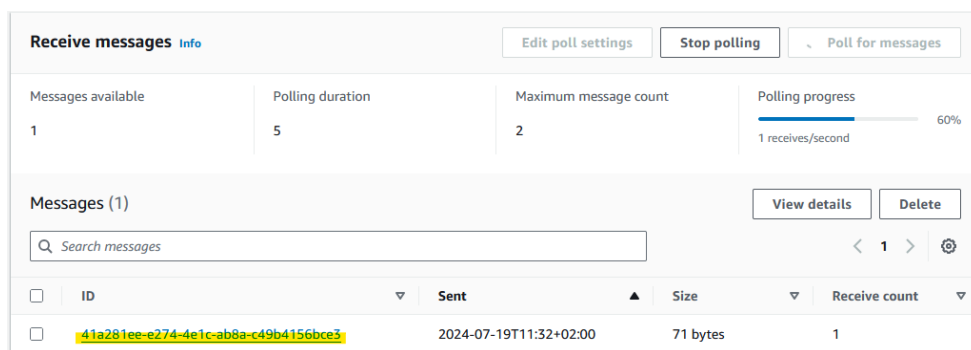
Agregar Atributo

Enviar Mensaje

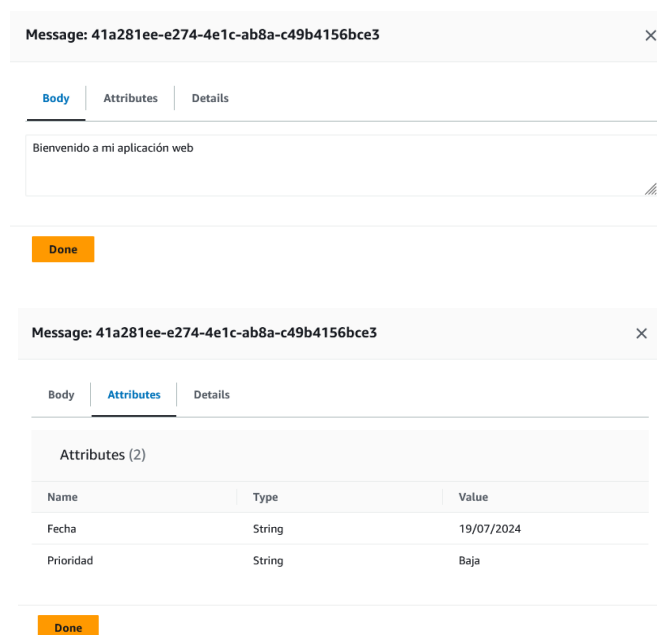
Si presionamos el botón **Enviar mensaje** nos aparecerá un mensaje de éxito:



- 37) Si volvemos a la consola de Amazon SQS y encuestamos la cola *mi-cola* podremos comprobar que recibimos el mensaje:



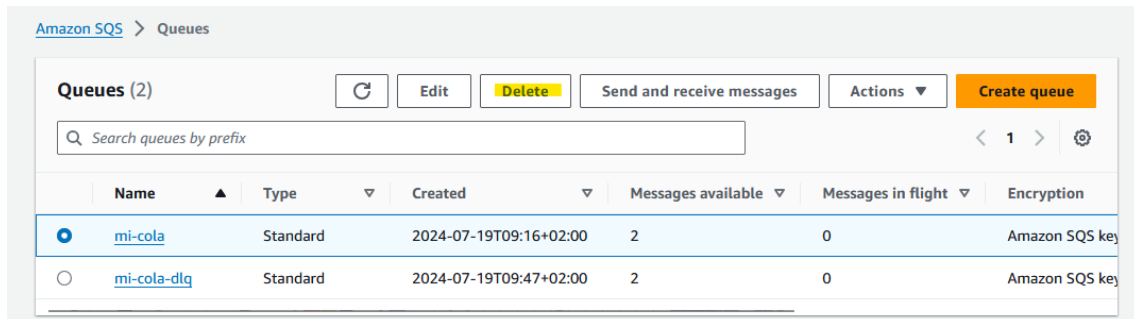
- 38) Si abrimos el mensaje, veremos el contenido y los atributos que hemos enviado desde nuestra aplicación:



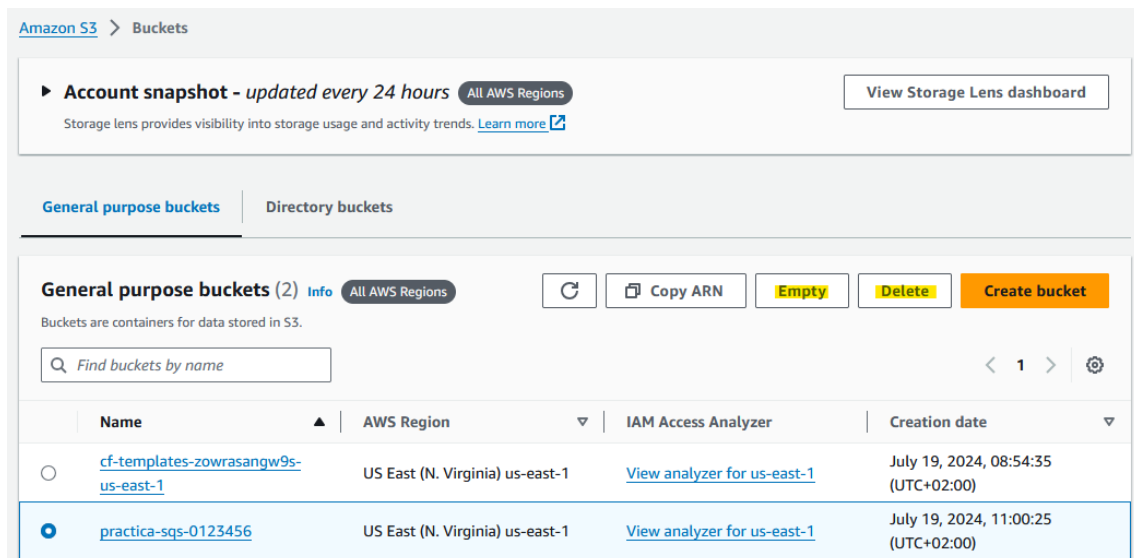
Limpieza de la Práctica:

Para terminar esta práctica y liberar los recursos creados, simplemente debemos dar los siguientes pasos:

- **Eliminación de las colas de Amazon SQS creadas.** Desde la consola de Amazon SQS seleccionamos las colas individualmente y presionamos el botón **Delete**:



- **Eliminación del bucket de Amazon S3.** Desde la consola de Amazon S3, seleccionamos el bucket creado y presionamos primero el botón **Empty** y, tras eliminar su contenido, presionamos el botón **Delete**:



- **Eliminación de la infraestructura del entorno de desarrollo.** Lo haremos desde **Cloudshell**, ejecutando la siguiente orden

```
$ aws cloudformation delete-stack --stack-name practica-sqs
```