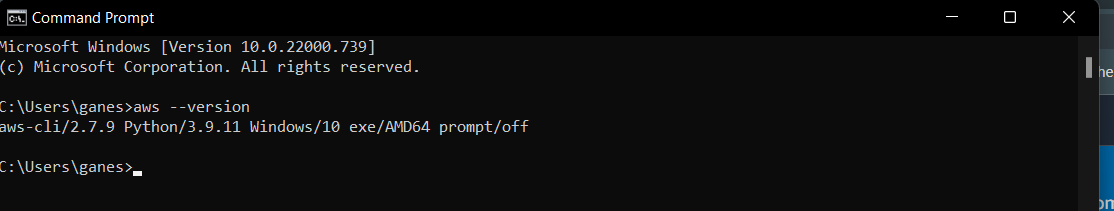
Handson 05 - **Solution**

Demonstrate the functioning of asynchronous messaging between 2 software components using AWS SQS. Create 2 Spring Boot RESTful services one acting as the producer of the message which will put the message in AWS SQS Queue, while the other acting as the consumer of the message from AWS SQS.

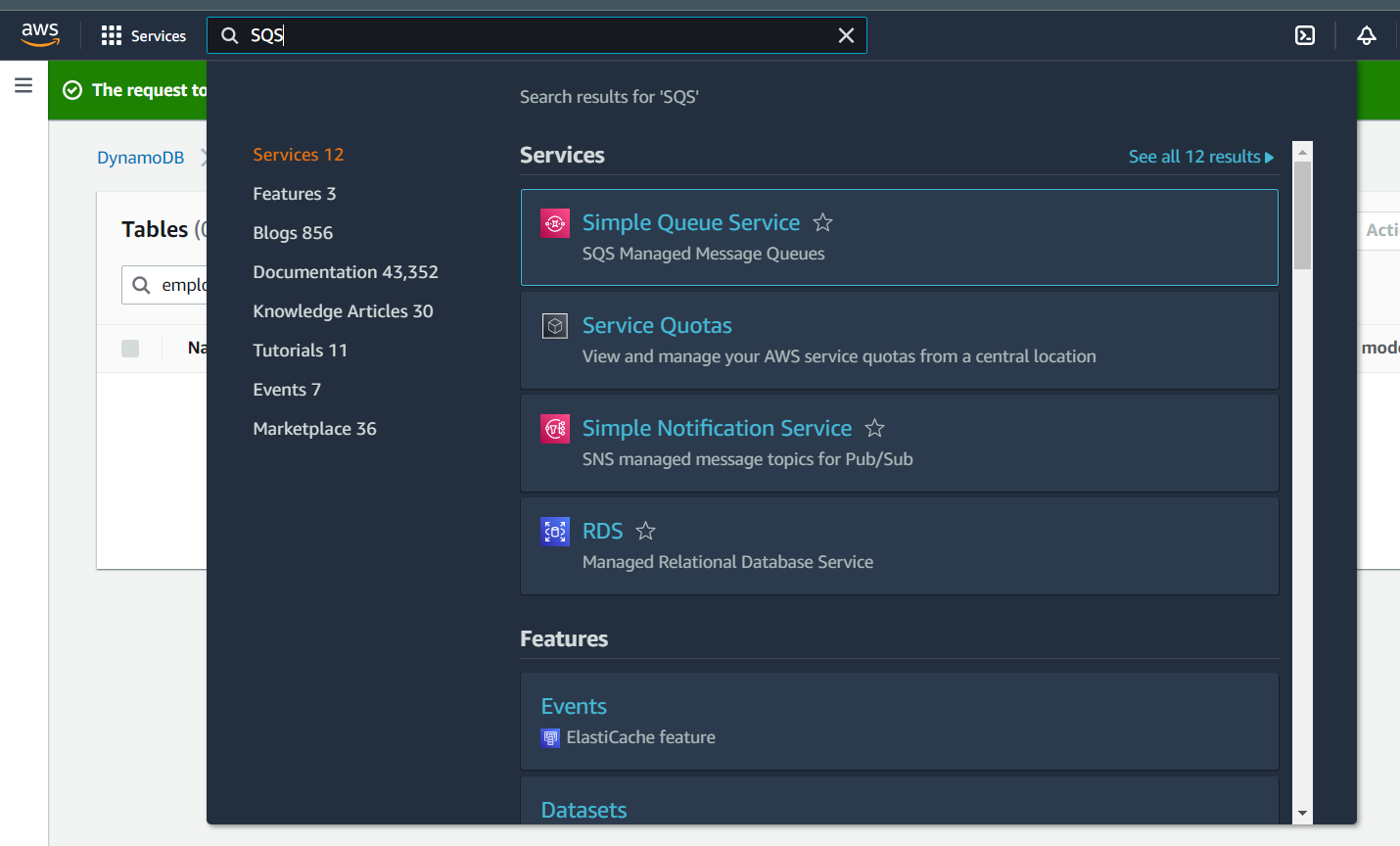
Steps:

1. Please ensure that AWS CLI is installed on your system. If not already installed, please visit the below link for downloading and installing AWS CLI (for windows).

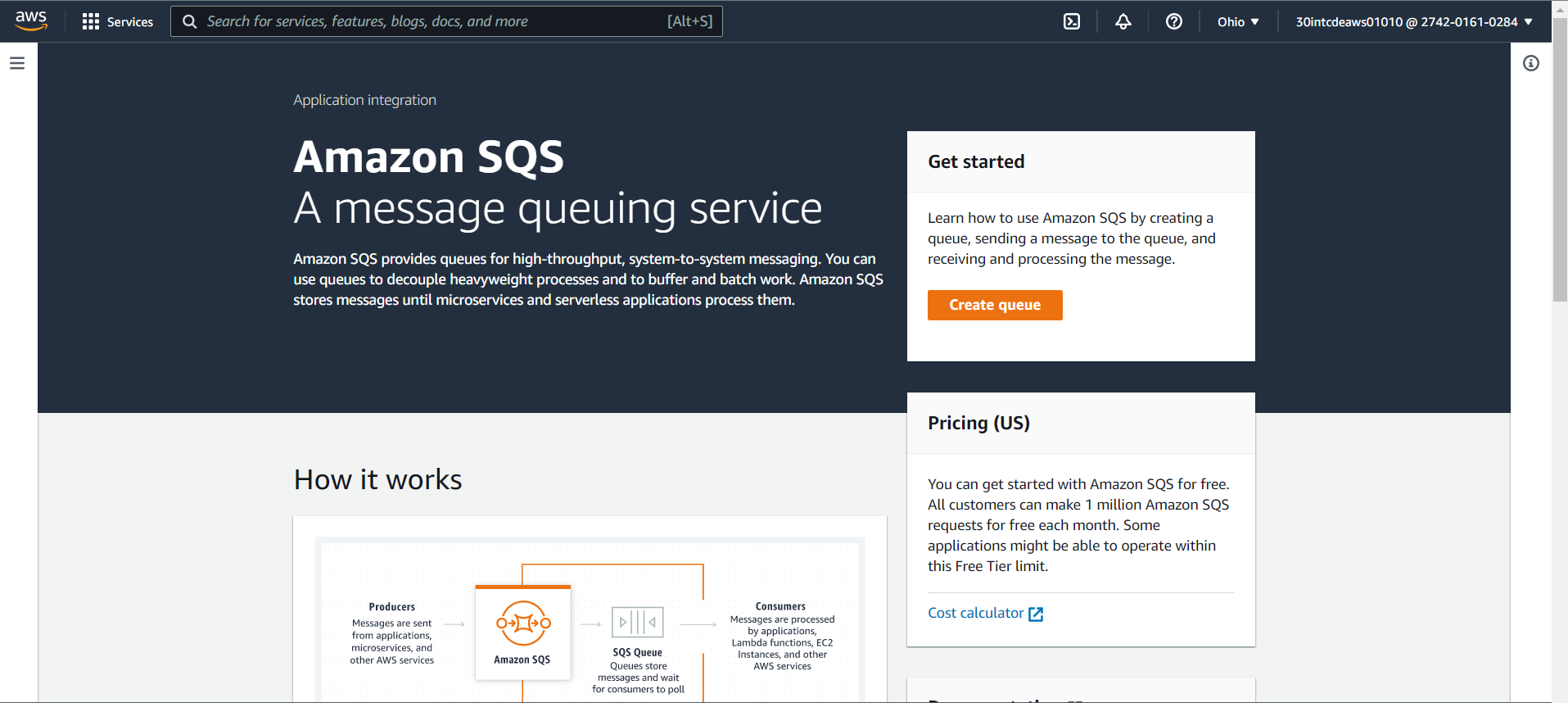
<https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-windows.html>



1. Make sure that the User / Role trying to access the SQS queue should have “AmazonSQSFullAccess” policy attached.
2. Create an SQS Queue.
   1. Go to the AWS console and search for SQS



* 1. Click on the “Create queue” button



* 1. You will be navigated to “Create queue screen” as shown below. You can create 2 types of queues in SQS.
     1. **Standard**

Standard queues support a nearly unlimited number of transactions per second (TPS) per API action (SendMessage, ReceiveMessage, or DeleteMessage).

A message is delivered at least once, but occasionally more than one copy of a message is delivered. Standard queues provide best-effort ordering. Occasionally, messages might be delivered in an order different from which they were sent.

Standard queues are useful when very high throughput is important

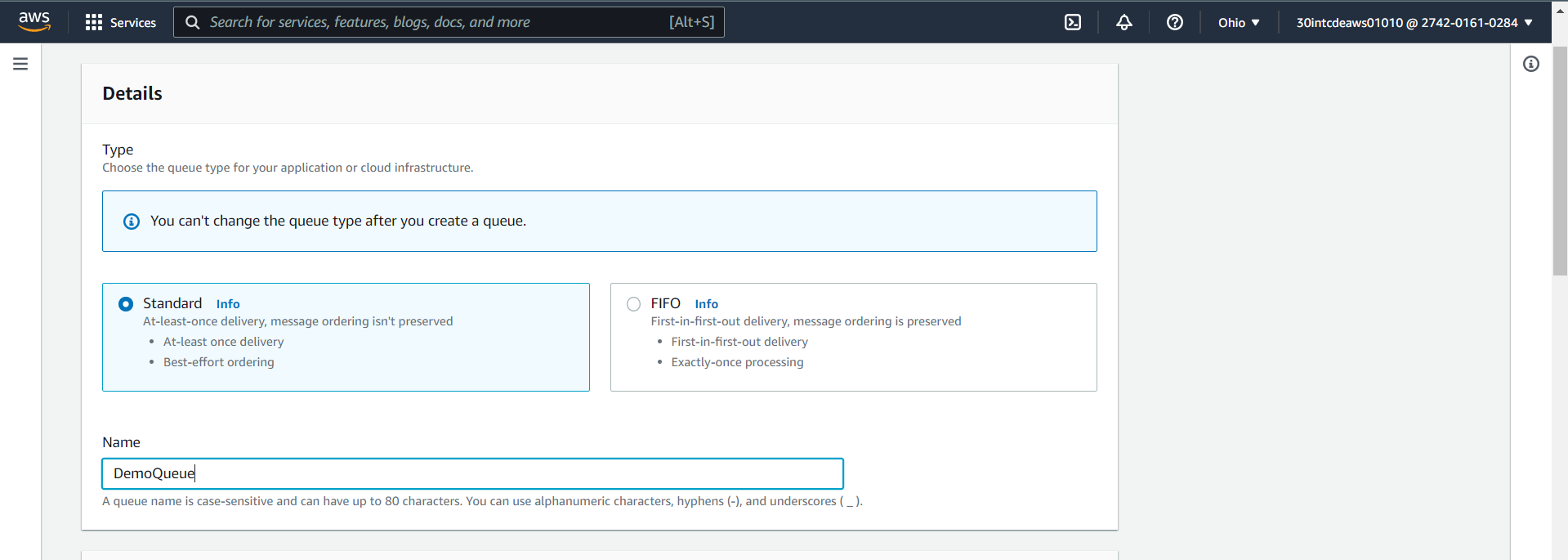
Standard queue is the default queue.

* + 1. **FIFO**

FIFO queues support up to 3000 messages per second, each message is delivered exactly once, and message order is preserved. FIFO queues are designed to enhance messaging between applications when the order of operations and events is critical, or where duplicates can't be tolerated.

The name of a FIFO queue must end with the .fifo suffix.

* 1. Select the standard queue and give a name for the queue.



* 1. You can specify additional configuration for your queue
     1. **Visibility timeout**, range🡺 0 seconds to 12 hours.

Visibility timeout sets the length of time that a message received from a queue (by one consumer) will not be visible to the other message consumers.

The visibility timeout begins when Amazon SQS returns a message. If the consumer fails to process and delete the message before the visibility timeout expires, the message becomes visible to other consumers. If a message must be received only once, your consumer must delete it within the duration of the visibility timeout.

* + 1. **Message retention period**, range🡺 1 minute to 14 days.

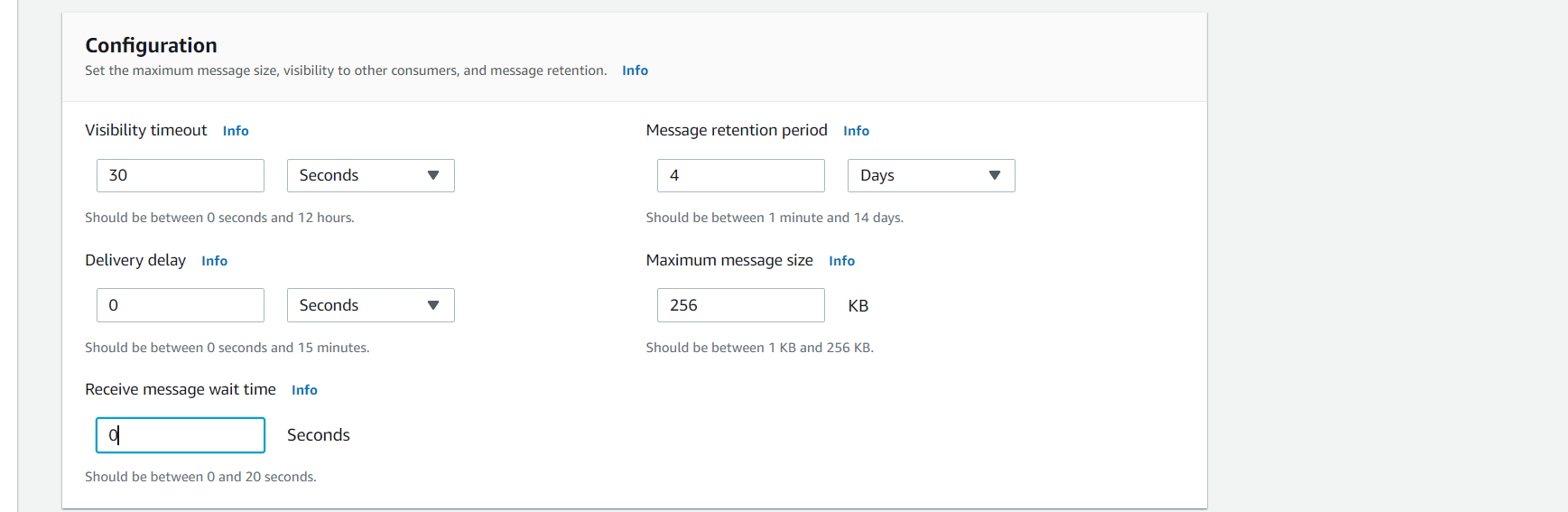
The message retention period is the amount of time that Amazon SQS retains a message that does not get deleted. Amazon SQS automatically deletes messages that have been in a queue for more than the maximum message retention period. The default retention period is 4 days.

* + 1. **Delivery delay**, range🡺 0 seconds to 15 minutes.

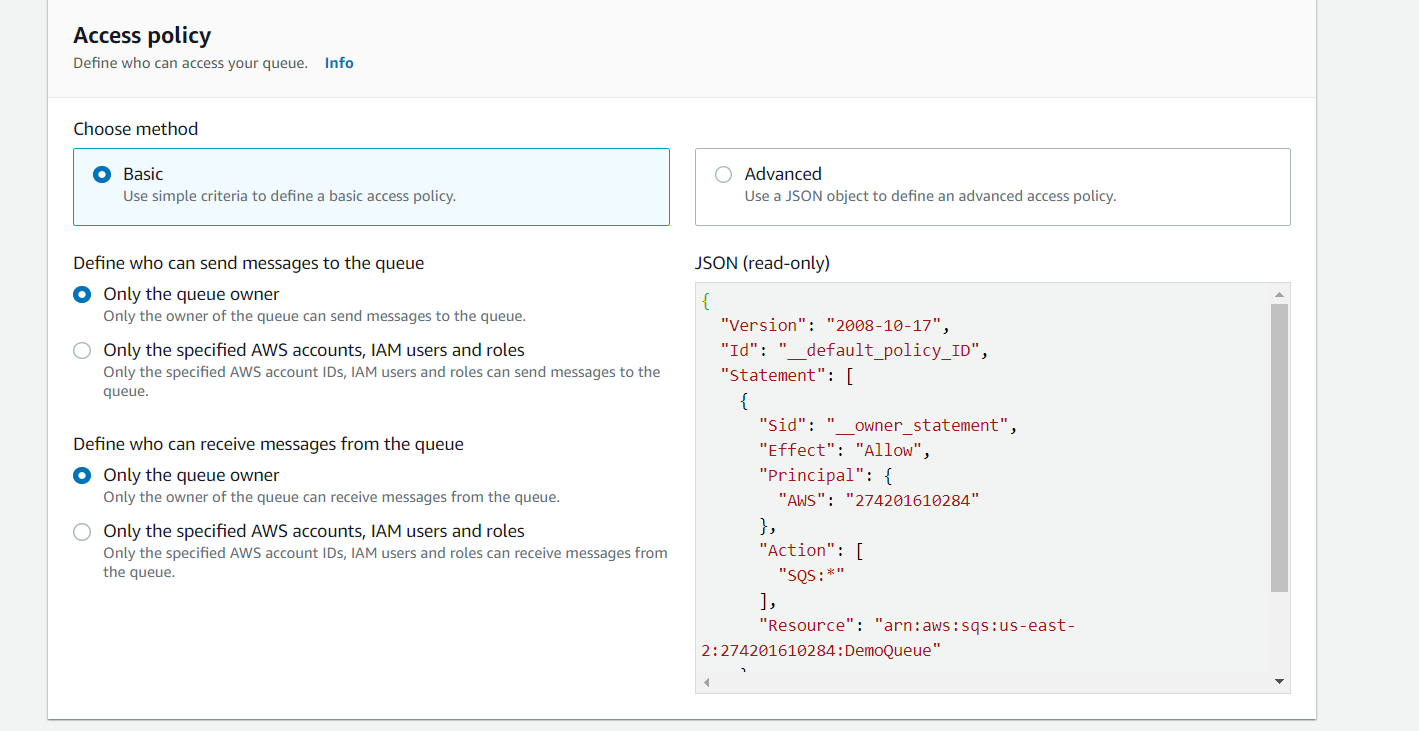
The delivery delay is the amount of time to delay the first delivery of each message added to the queue. Any messages that you send to the queue remain invisible to consumers for the duration of the delay period

* + 1. **Maximum message size**, range🡺 1 KB to 256 KB
    2. Receive message wait time, range🡺 0 to 20 seconds

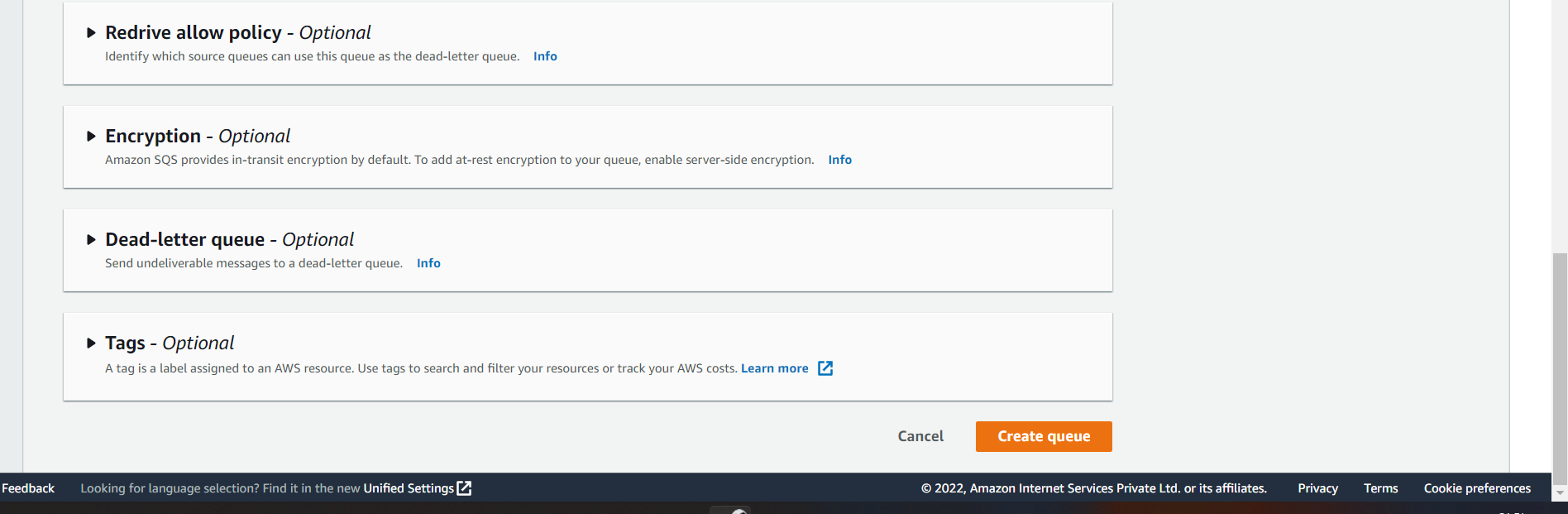
The receive message wait time is the maximum amount of time that polling will wait for messages to become available to receive. The minimum value is zero seconds, and the maximum value is 20 seconds.



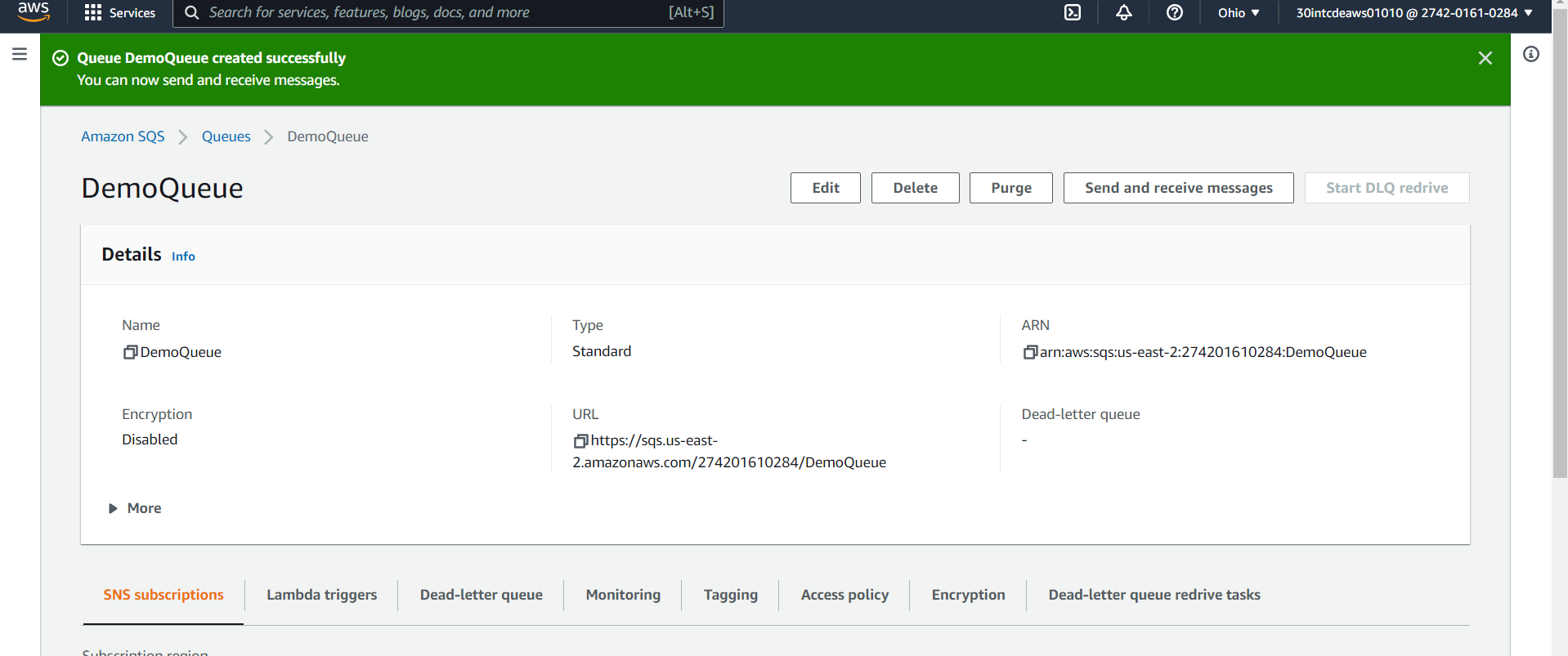
* 1. Select “Basic” as the Access policy and specify who can send and receive messages to the queue based on your application’s requirements.



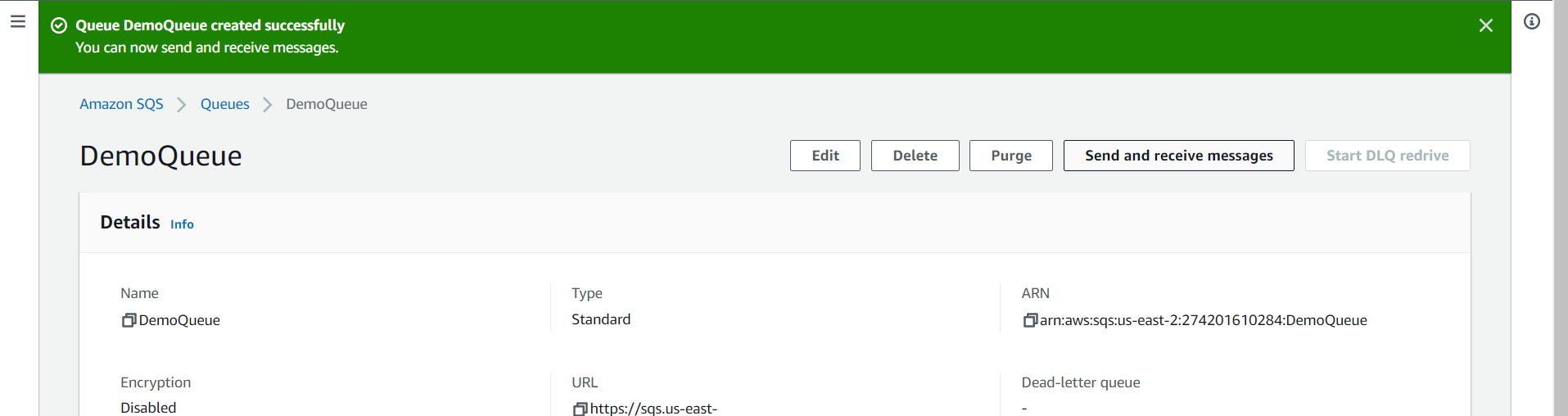
* 1. Click on the “Create queue” button.



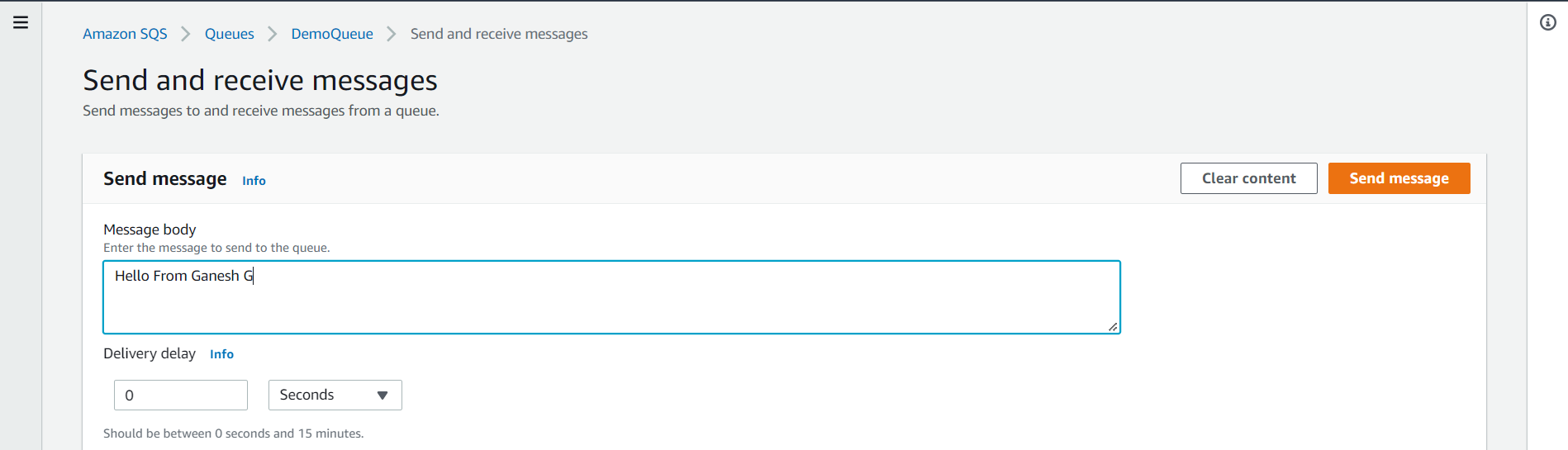
* 1. You will be redirected to “Success” screen as shown below.



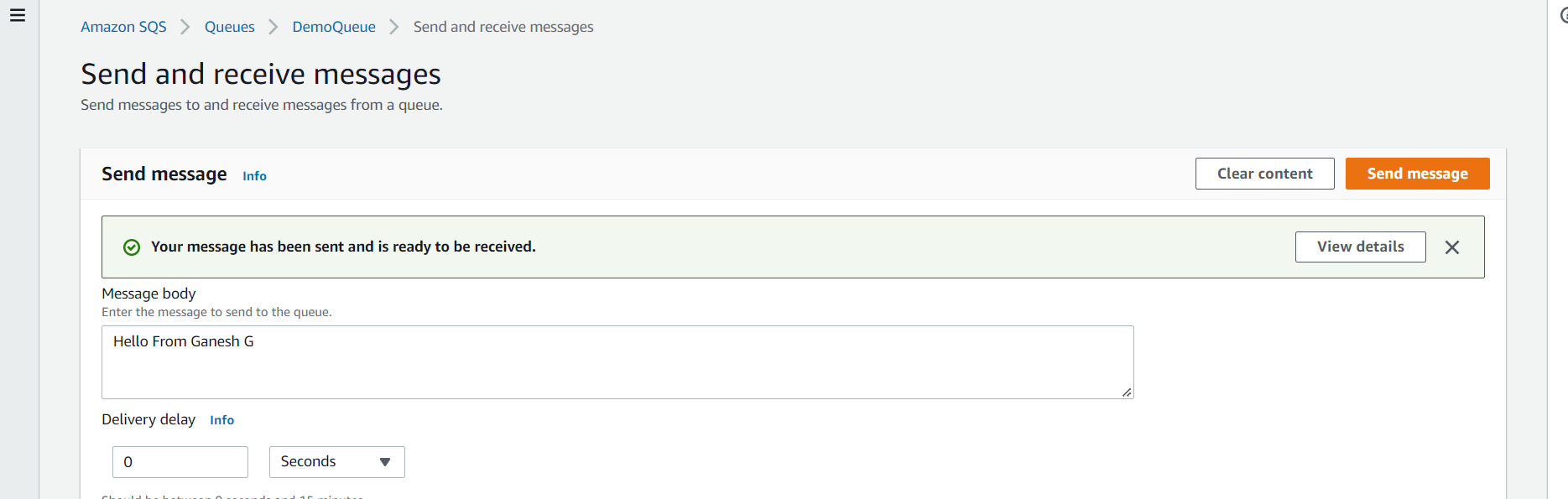
1. Click on the “Send and receive messages” button



1. Enter your message in the “Message body” and click on “Send message” button.



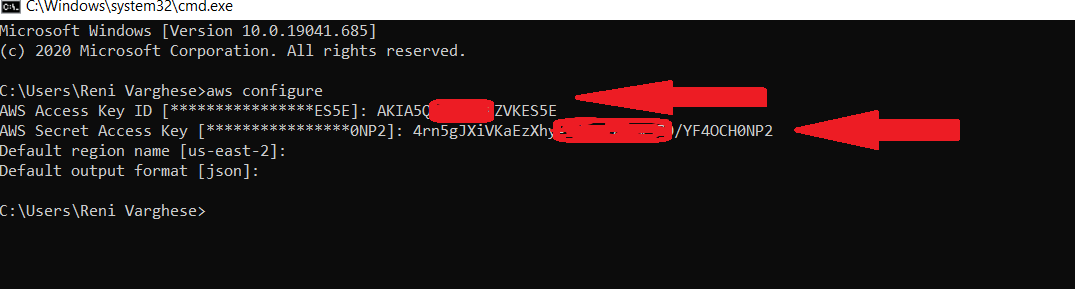
1. You can see the success message and also you can view that the messages available in the queue is 1 as shown in the screen below.



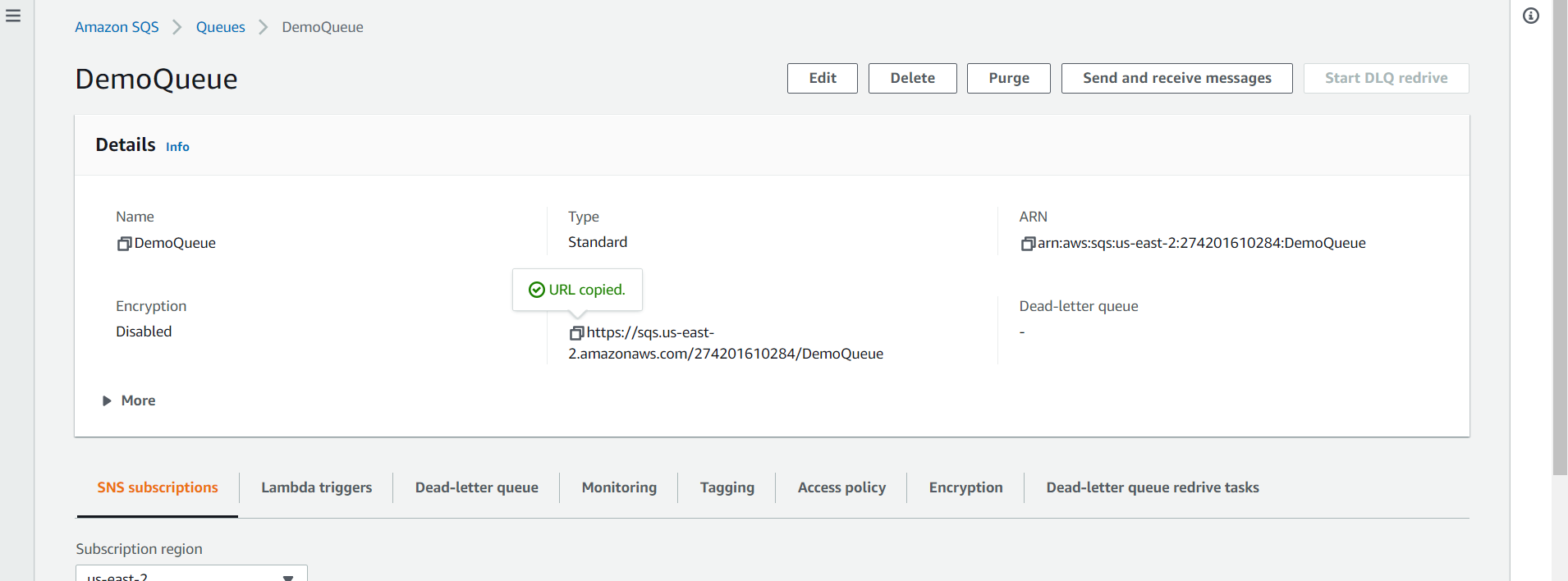
1. You can now consume this message from command prompt using AWS CLI buy issuing the below commands.
   1. If you have not already configured AWS CLI, you have to configure it by issuing the below command.

**aws configure**.

* 1. Specify your access key, secret key and your region as shown in the screen below.



* 1. Copy the URL to the SQS queue as shown in the screen below.

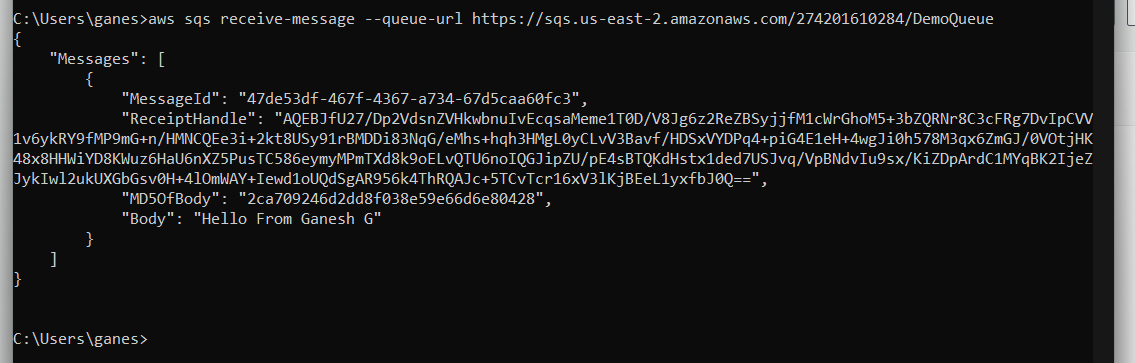


* 1. Issue the below mentioned AWS CLI command to consume the message.

aws sqs receive-message --queue-url <https://sqs.us-east-2.amazonaws.com/929084036303/DemoQueue>

queue-url 🡺 specify the URL of your queue.

* 1. You can see that the message has been consumed by the consumer as shown by the screen below in JSON format.



1. Create one Spring REST Application which is going to act as the producer of messages.
2. Add the following dependency to the application

<dependency>

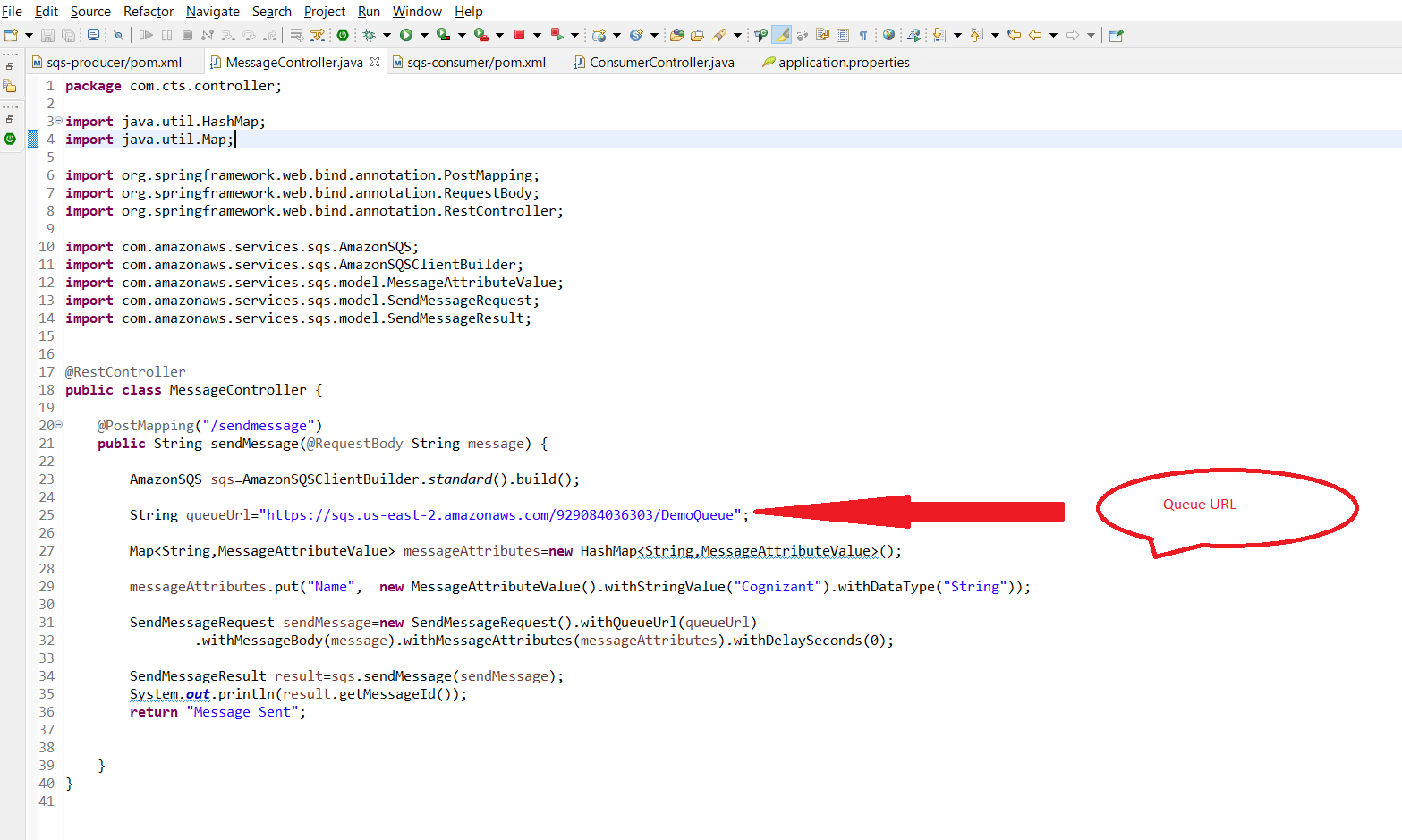
<groupId>com.amazonaws</groupId>

<artifactId>aws-java-sdk</artifactId>

<version>1.11.918</version>

</dependency>

1. Create a REST Controller class with the following code snippets



1. Create another Spring REST application which is going to act as the consumer of the SQS Queue.
2. Include the following dependency in the application.

<dependency>

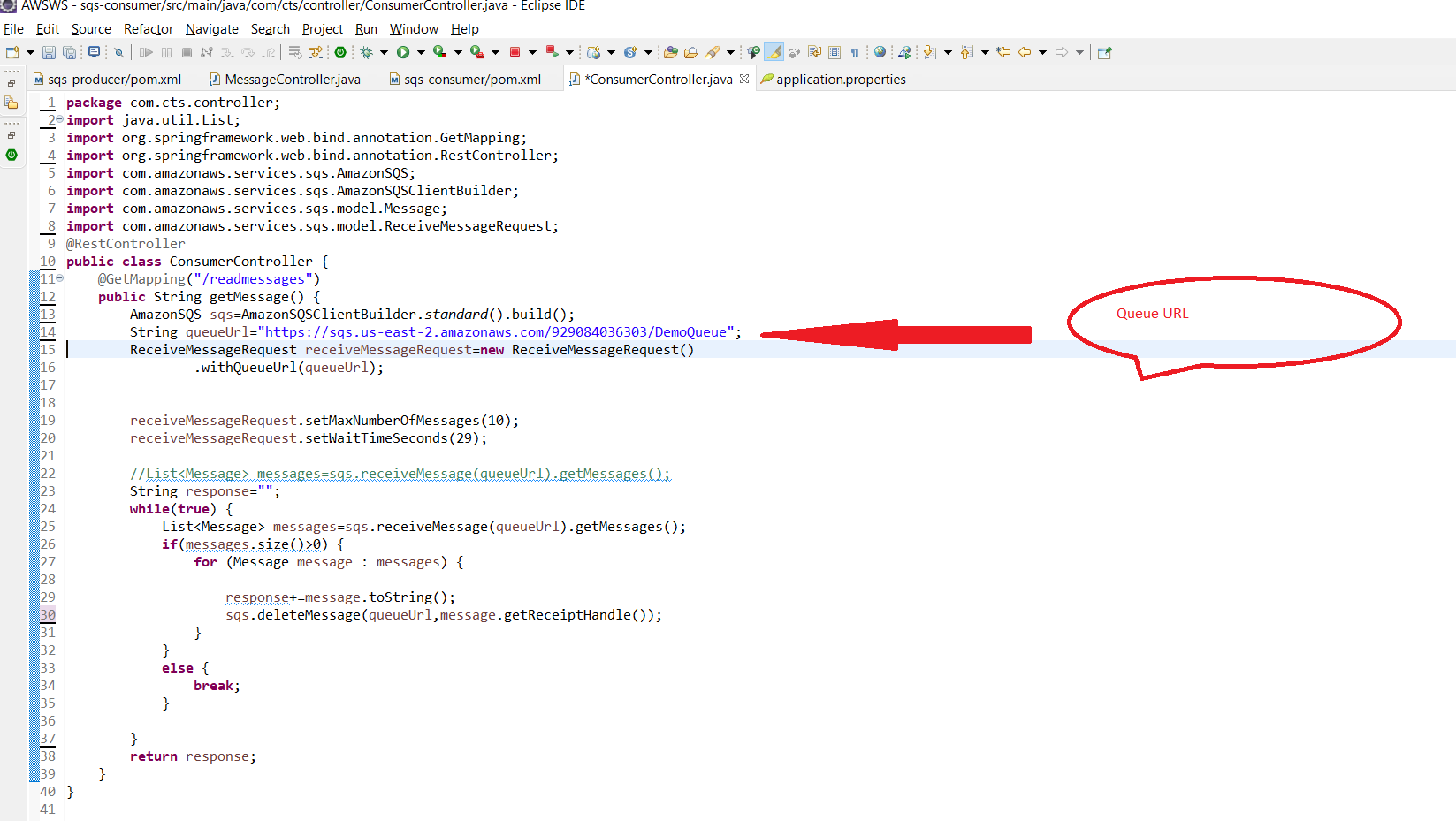
<groupId>com.amazonaws</groupId>

<artifactId>aws-java-sdk</artifactId>

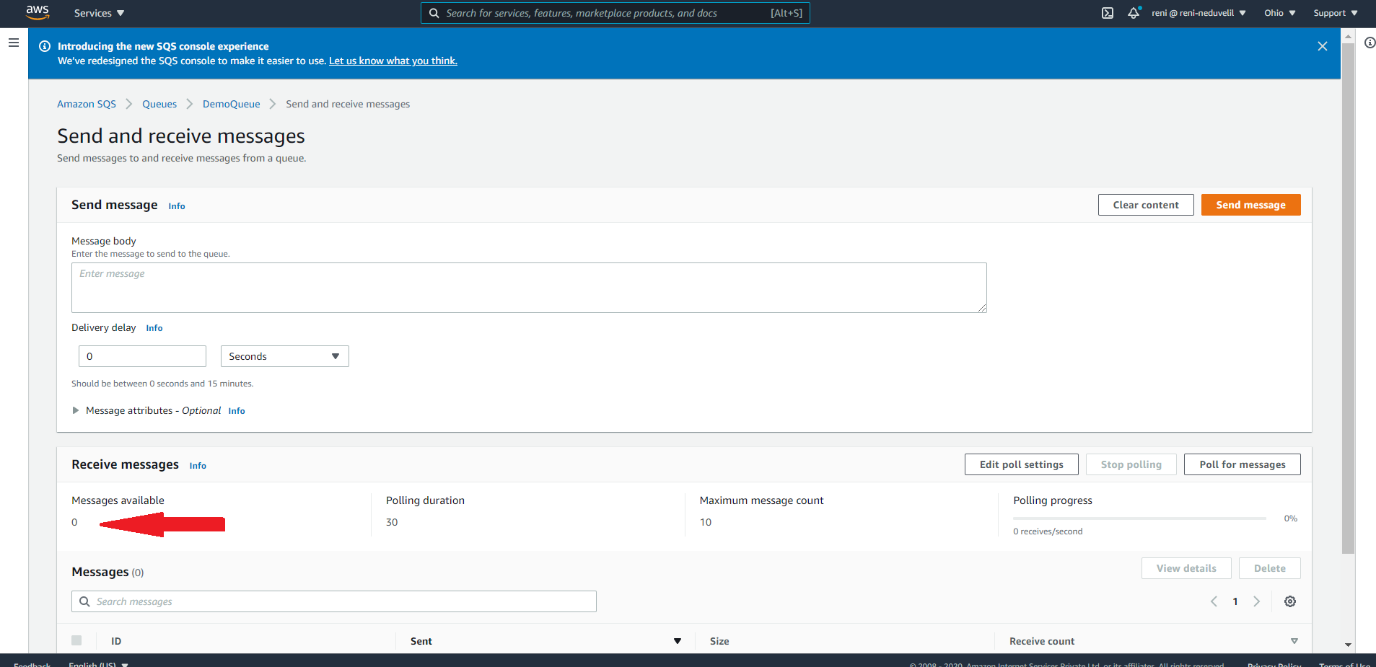
<version>1.11.918</version>

</dependency>

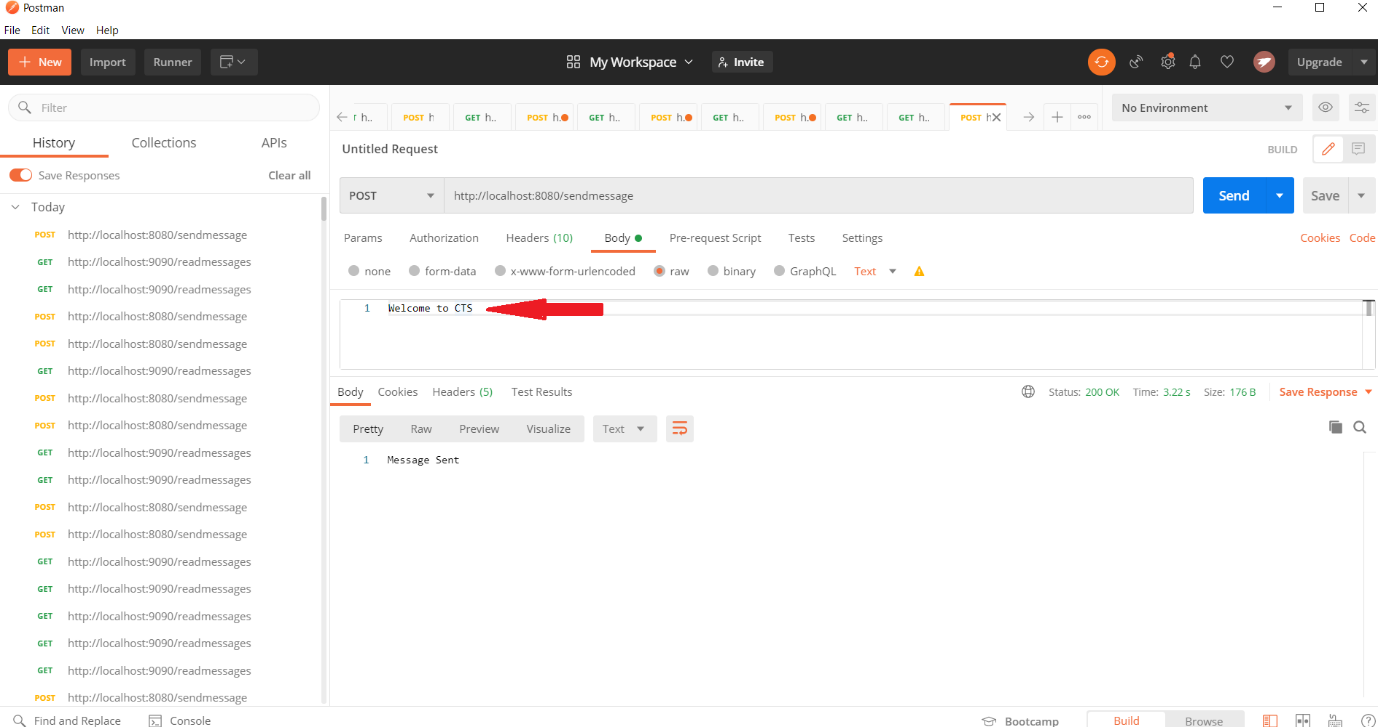
1. Include a REST controller with the following code snippets.

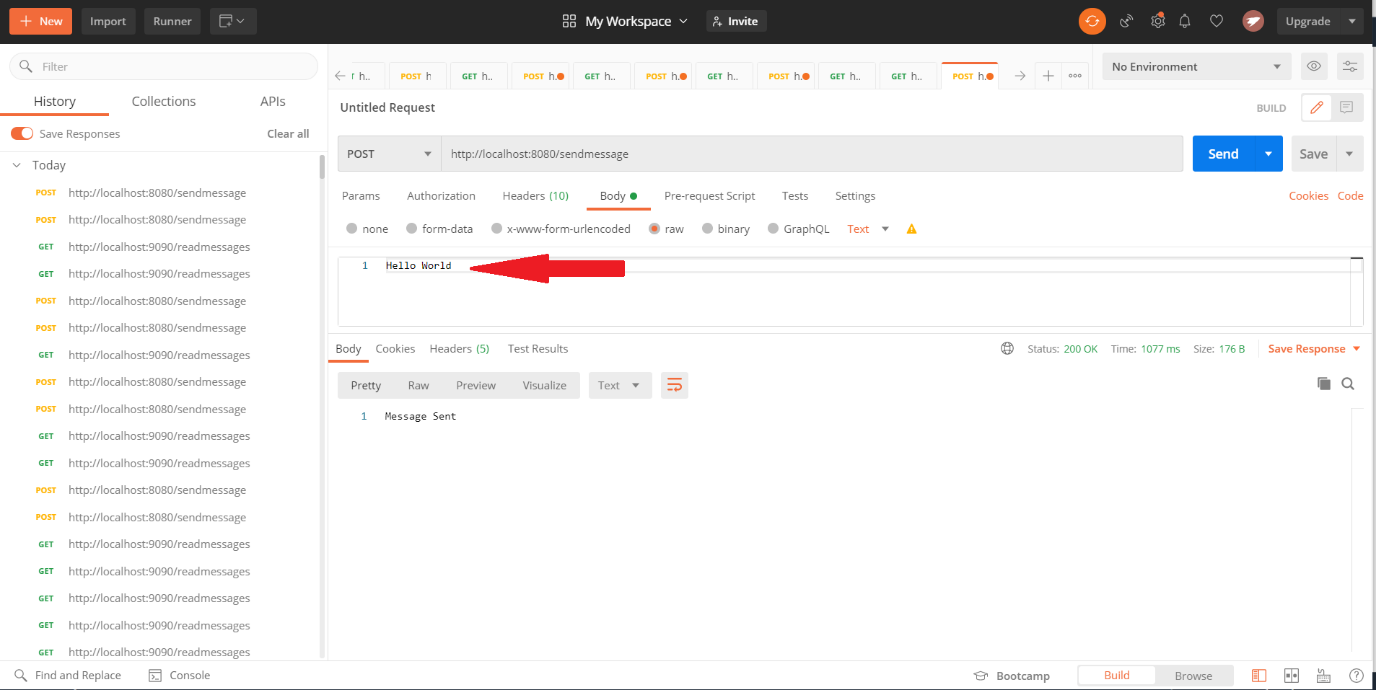


1. Run both the producer and consumer API services
2. Check for any messages in the queue. You can see that there are not messages in the queue.

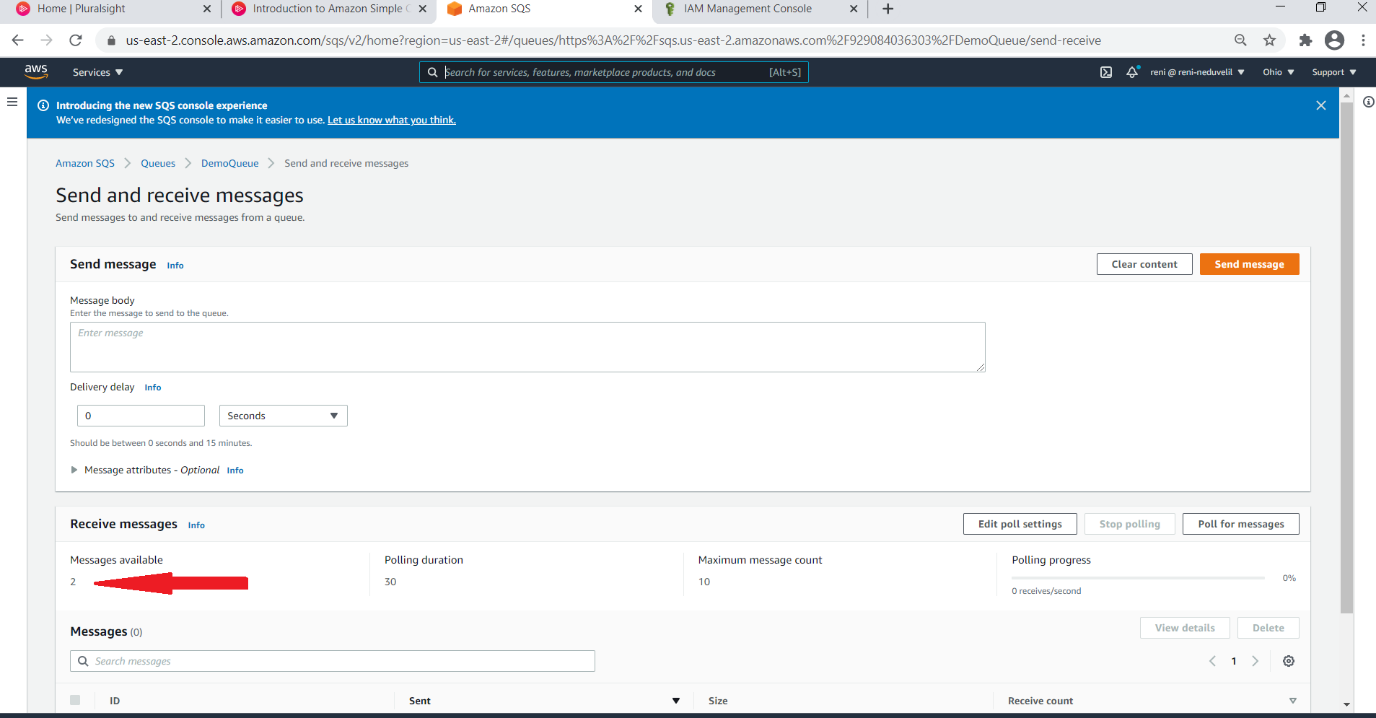


1. Invoke the Producer API endpoint to produce a couple of messages as shown by the screenshots below.

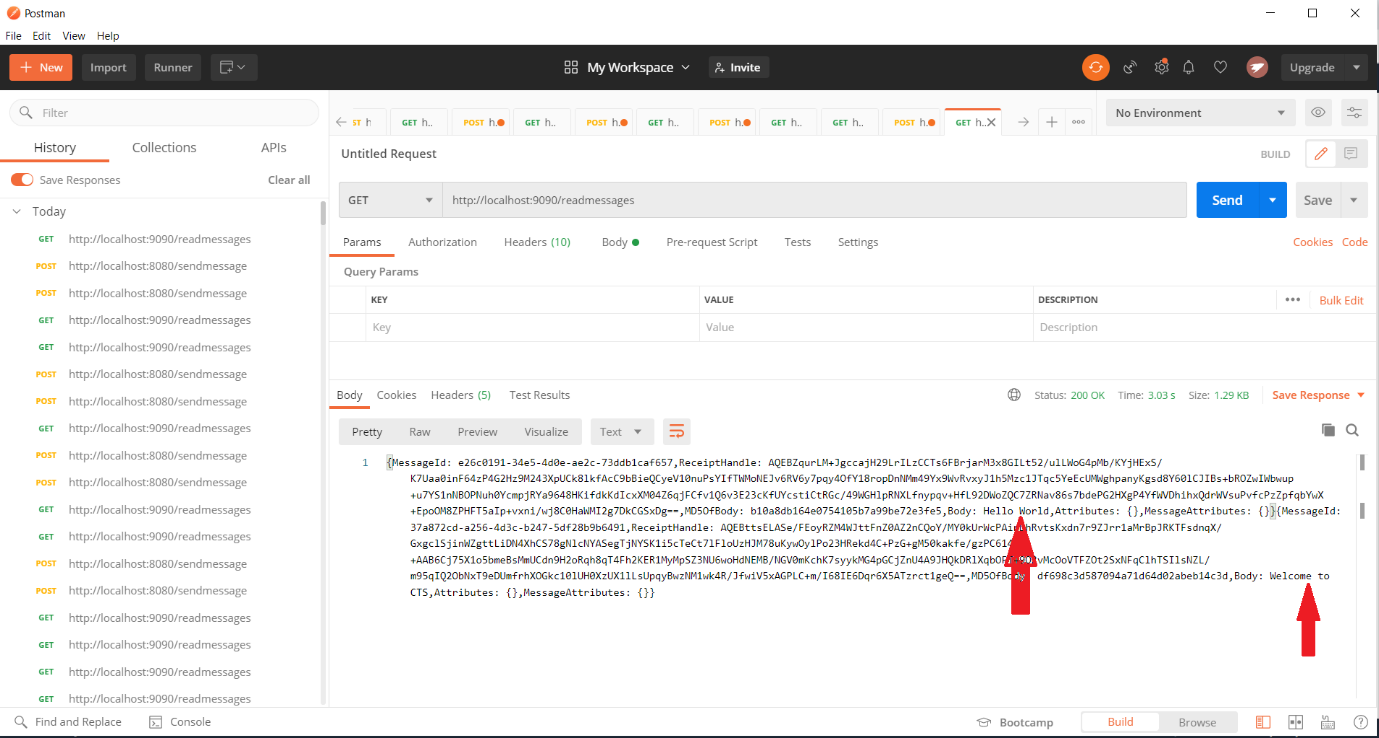




1. Check whether the messages has been delivered in the queue. You can see that 2 messages are already there in the queue.



1. Invoke the Consumer REST API endpoint to consume the messages from the SQS queue.



1. You can see that both messages have been consumed by the consumer application.
2. This is how we can create services which are decoupled from each other which promotes fault tolerance in a distributed system using AWS managed SQS service.

Note: Please make sure that the SQS queue is deleted once you have done with the hands-on.

