Amazon Simple Queue Service

什么是SQS

Amazon Simple Queue Service (Amazon SQS) 提供安全、持久且可用的托管队列，可让您集成和分离分布式软件系统和组件。Amazon SQS 提供常见的构造，例如死信队列和成本分配标签。它提供了一个通用 Web 服务 API，并且可通过 AWS 开发工具包支持的任何编程语言访问。

标准队列

Amazon SQS 提供 标准 作为默认队列类型。 标准队列的每个 API 操作（SendMessage、ReceiveMessage 或 DeleteMessage）每秒支持接近无限的 API 调用。标准队列支持至少一次消息传递。但是,偶尔(由于高度分布式架构允许几乎无限的吞吐量),可能会乱序发送多个消息副本。标准队列会尽最大努力进行排序，保证了消息大致按其发送的顺序进行传递。

FIFO队列

FIFO 队列具有标准队列的所有功能。

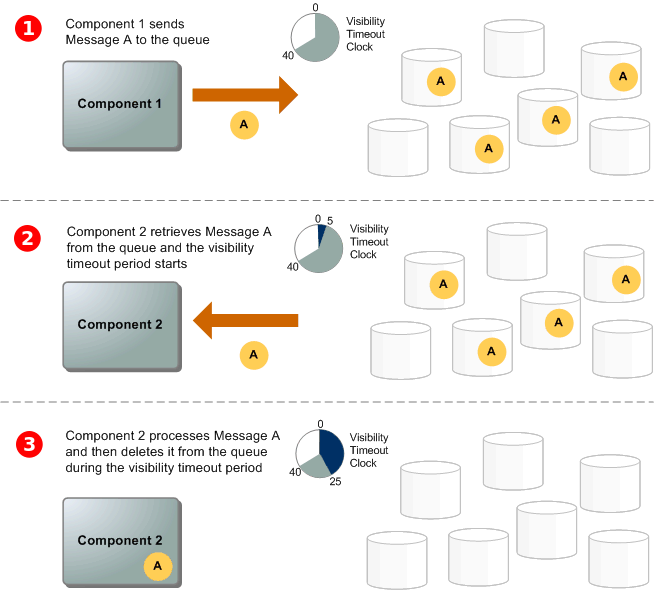
FIFO (先进先出) 队列能在操作和事件的顺序很重要或者不能容忍重复消息的情况下改善应用程序之间的消息收发。

FIFO 队列在标准队列的基础上改进和补充此队列。此队列类型最重要的特性是 FIFO (先进先出)传送和一次性处理。

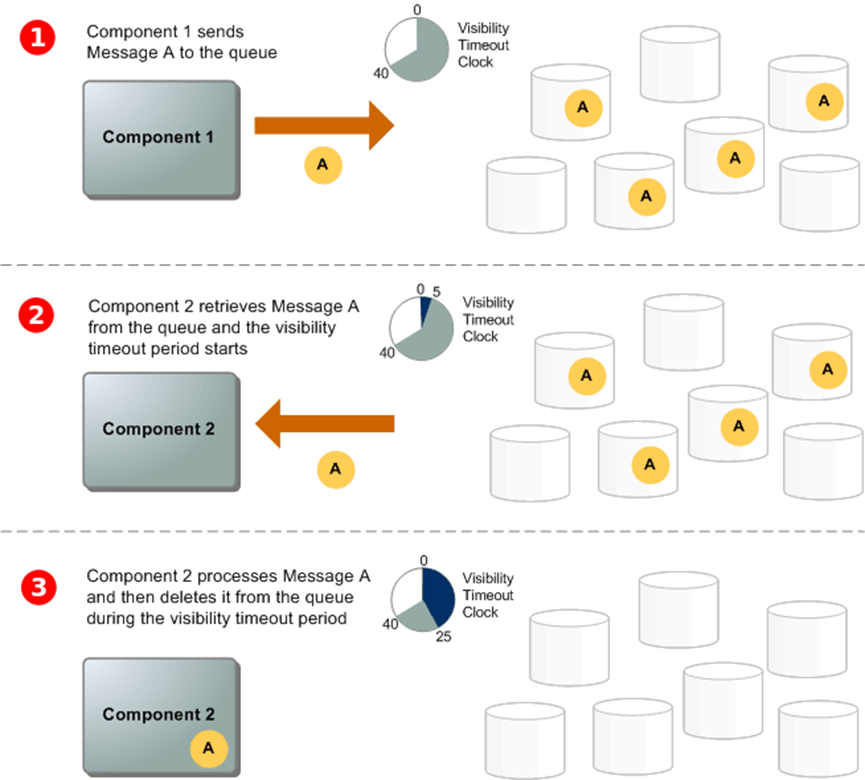
FIFO队列的特性

* 发送和接收消息的顺序严格保持一致；一条消息传送一次后就保持可用，直到使用者处理并删除它为止。
* 不会将重复项引入到队列中。
* FIFO 队列支持消息组,该组允许单个队列中存在多个有序的消息组。对于 FIFO 队列中的消息组数量没有配额限制。

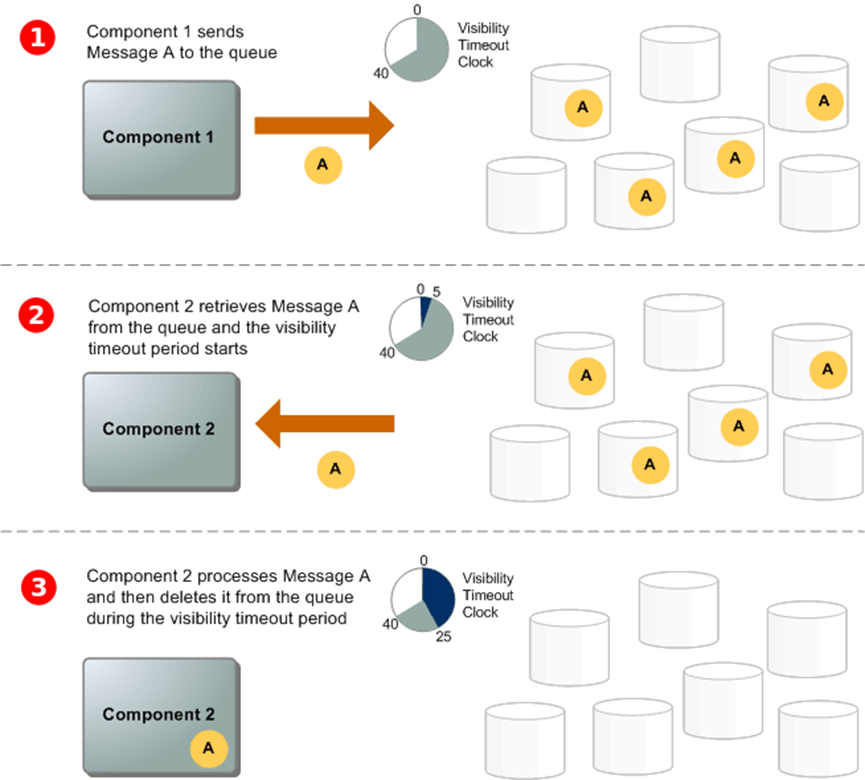
Amazon SQS 消息在队列中从创建到删除的整个生命周期



生产者(组件1)将消息A发送到队列,消息分布在 Amazon SQS 服务器冗余。



当消费者(组件2)准备好处理消息时,它会消耗来自队列的消息,消息A返回。在处理 Message A 期间，它仍保留在队列中，并且在可见性超时期间不返回至后续接收请求。



使用者(组件2)从队列中删除消息A,以防止在可见性超时到期时再次接收和处理消息。

死信队列

Amazon SQS 支持死信队列，其他队列（源队列）可将其作为无法成功处理（使用）的消息的目标。死信队列有助于调试您的应用程序或消息传递系统，因为它们可让您隔离有问题的消息以确定其处理失败的原因。

有时会因各种可能的问题 (例如，创建者应用程序或使用者应用程序内的错误条件或导致您的应用程序代码出现问题的意外状态更改) 而导致无法处理消息。例如，如果用户使用某特定产品 ID 下达 Web 订单，但产品 ID 已被删除，则 Web 商店的代码会失败并显示错误，而且包含订单请求的消息将发送到死信队列。

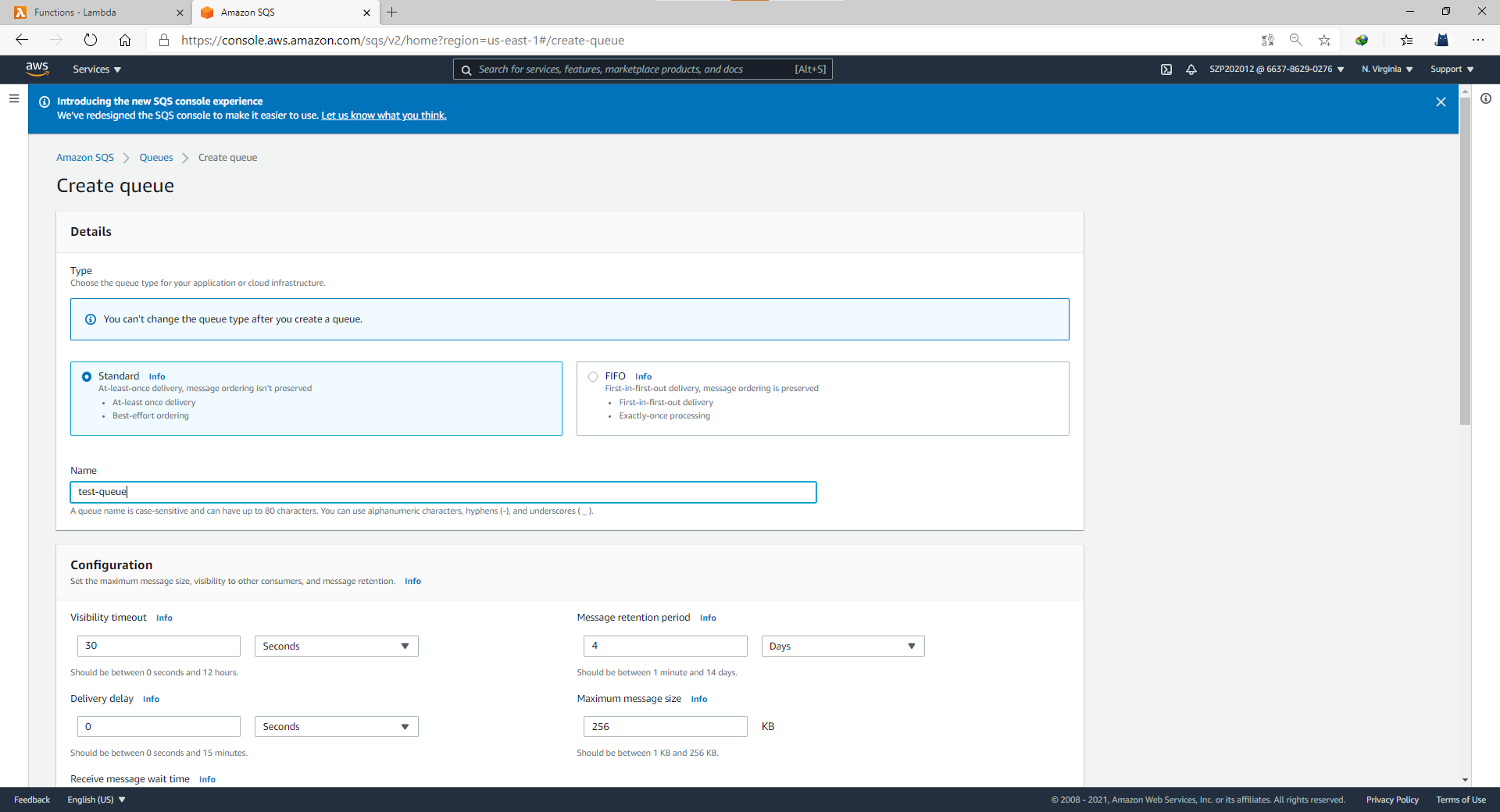
有时，创建器和使用器可能无法解释其用于通信的协议的各个方面，从而导致消息中断或丢失。此外，使用器的硬件错误可能会中断消息负载。

重新驱动策略 指定源队列、死信队列 以及 Amazon SQS 将消息从前者移至后者的条件（如果源队列的使用者无法处理消息指定次数）。当消息的 ReceiveCount 超出队列的 maxReceiveCount 时，Amazon SQS 会将该消息移到死信队列（带有其原始消息 ID）。例如，如果源队列的重新驱动策略已将 maxReceiveCount 设置为 5，并且源队列的使用器收到一条消息 6 次而从未将其删除，则 Amazon SQS 会将该消息移至死信队列。

实验：创建一个队列并用Lambda处理事件

1. 创建一个SQS队列

点击Create queue，输入队列名字，其他参数保持默认，点击创建



1. 创建一个Lambda函数

import json

def lambda\_handler(event, context):

# TODO implement

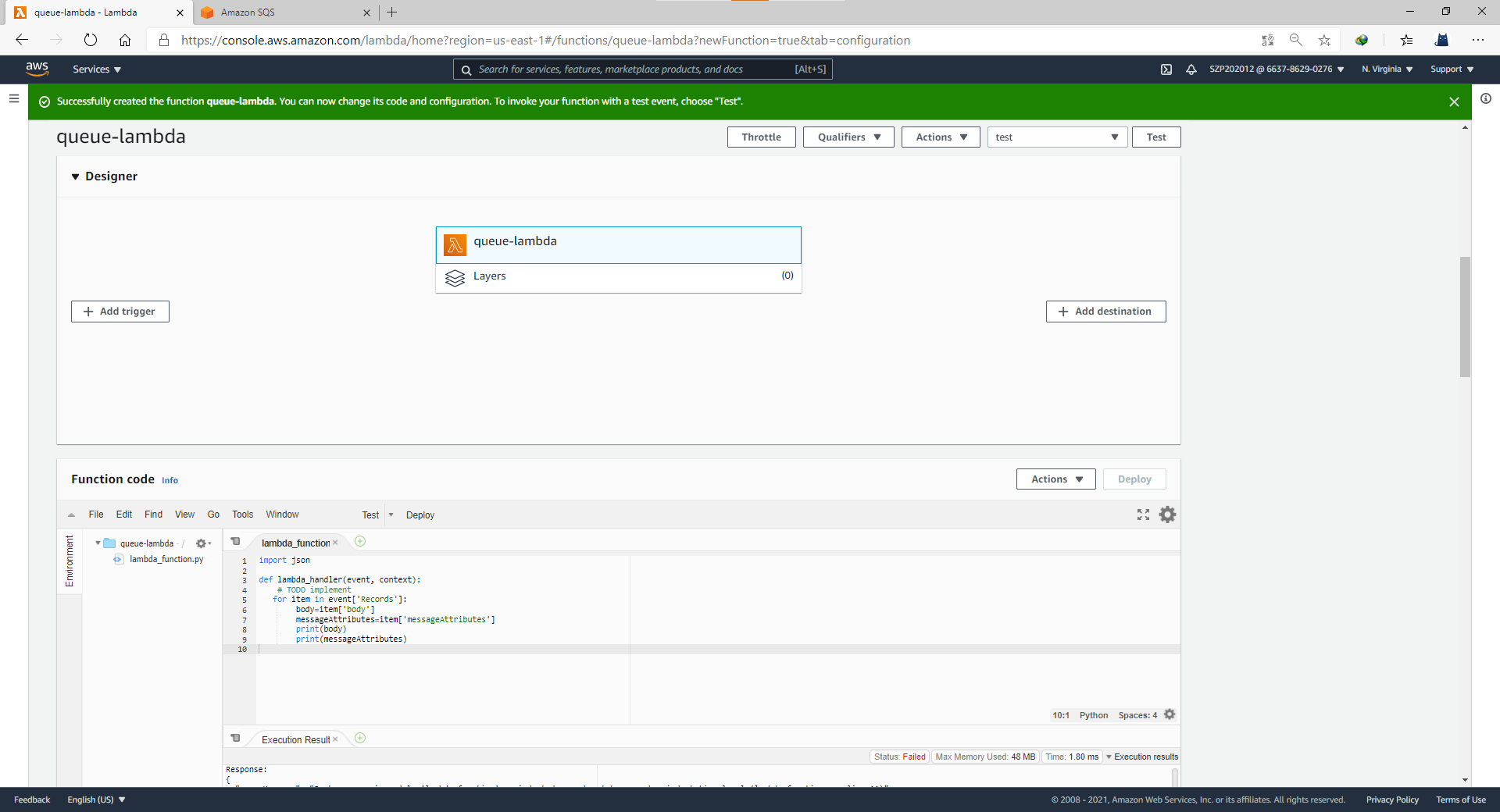
for item in event['Records']:

body=item['body']

messageAttributes=item['messageAttributes']

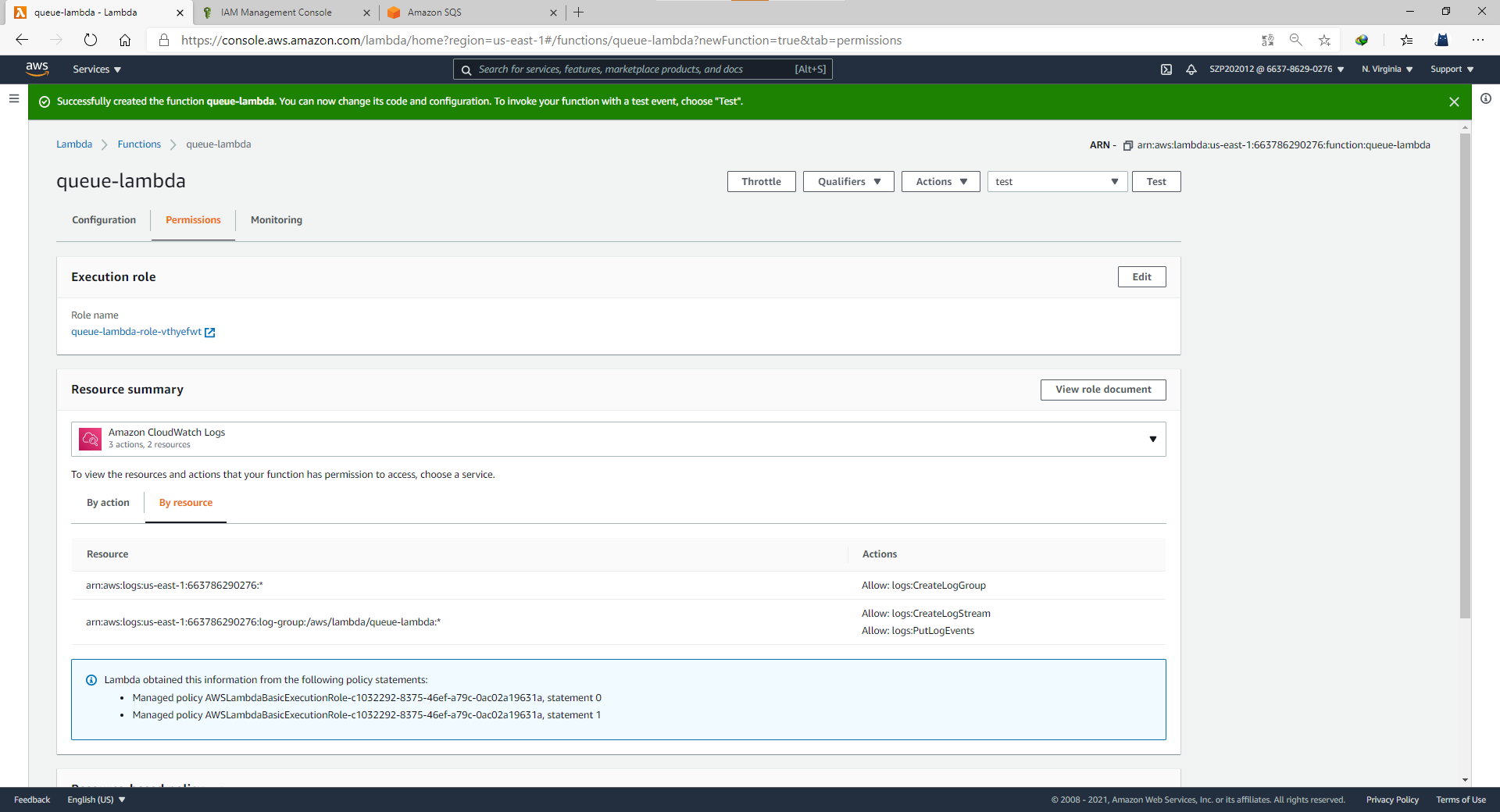
print(body)

print(messageAttributes)



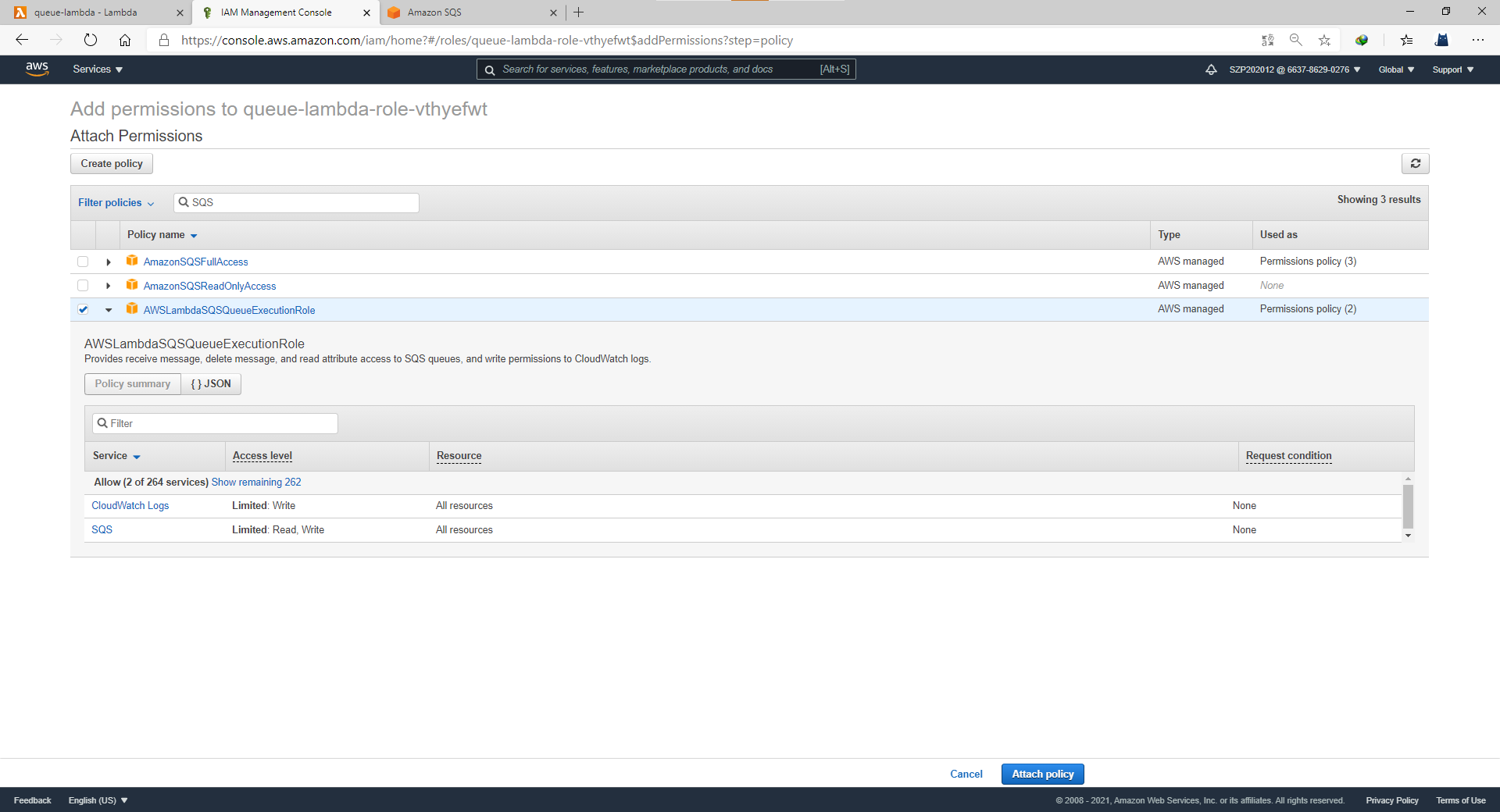
1. 修改Lambda函数的权限

点击Permissions，选择那个角色



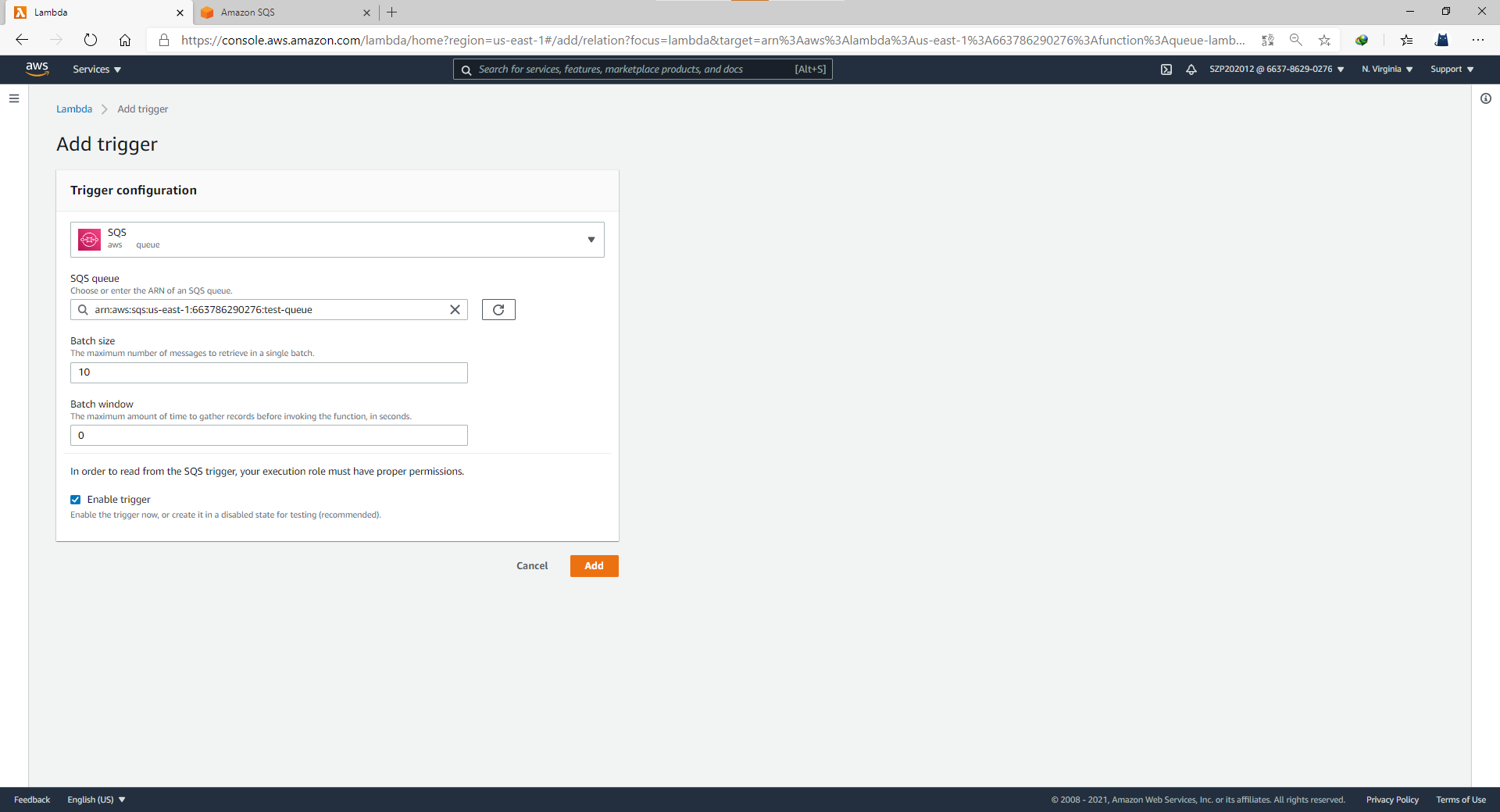
点击Attach Permissions

搜索AWSLambdaSQSQueueExecutionRole，添加这个策略



1. 给Lambda添加触发器

点击Add trigger，将队列添加为触发器



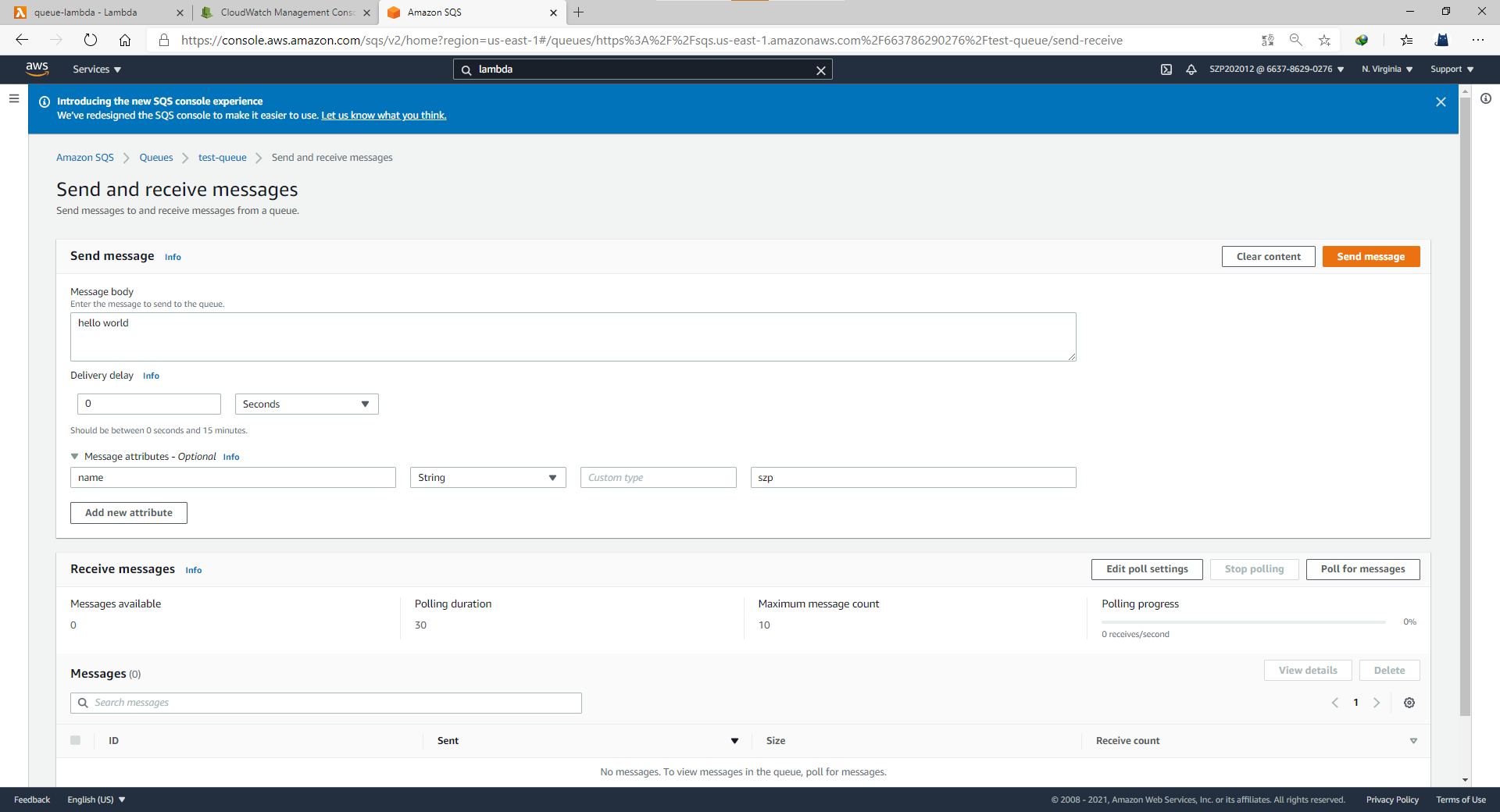
1. 回到SQS，我们发消息试试

点击Send and receive messages

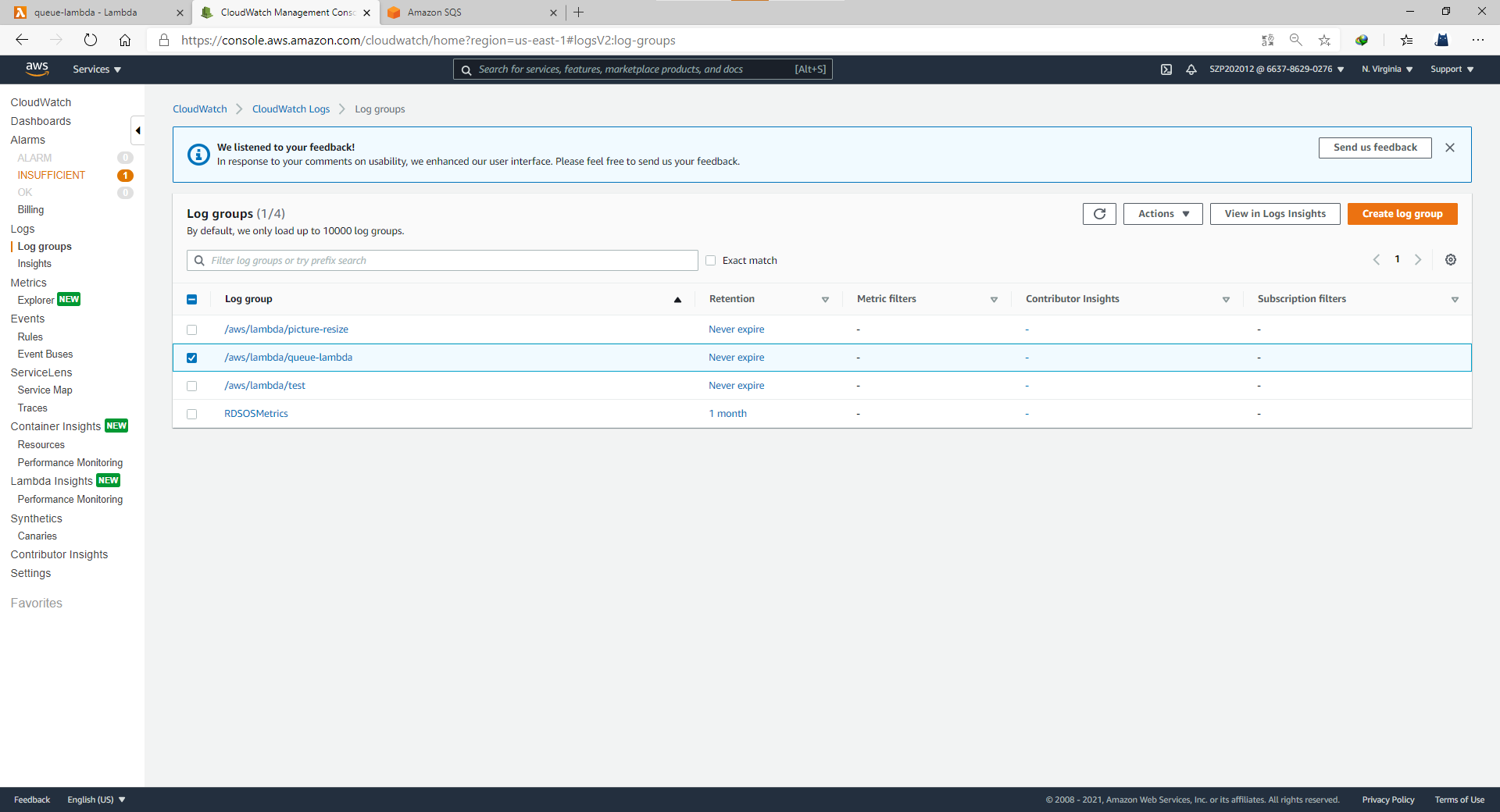
输入Message body，它的类型是文本

再添加Message attributes，它的类型是键值对

然后点击Send messages



1. 我们打开Cloudwatch查看Lambda的日志



打开日志流发现Lambda已经打印了刚刚发送的消息，恭喜！实验成功

