



Guia do desenvolvedor

Amazon Simple Queue Service



Amazon Simple Queue Service: Guia do desenvolvedor

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não são propriedade da Amazon pertencem aos respectivos proprietários, os quais podem ou não ser afiliados, estar conectados ou ser patrocinados pela Amazon.

Table of Contents

O que é o Amazon SQS?	1
Benefícios do uso do Amazon SQS	1
Arquitetura básica	1
Filas distribuídas	2
Ciclo de vida de mensagens	2
Diferenças entre o Amazon SQS, Amazon MQ e Amazon SNS	4
Começar	6
Configurar	6
Etapa 1: criar um usuário Conta da AWS e IAM	6
Etapa 2: conceder acesso programático	8
Etapa 3: preparar-se para usar o código de exemplo	10
Próximas etapas	11
Noções básicas sobre o console do Amazon SQS	11
Tipos de fila	12
Implementar sistemas de solicitação-resposta no Amazon SQS	16
Criar uma fila padrão	16
Criação de uma fila	16
Como enviar uma mensagem usando uma fila padrão	18
Criar uma fila FIFO	19
Criar uma fila	19
Como enviar uma mensagem usando uma fila FIFO	22
Tarefas comuns	22
Gerenciamento de fila	24
Editar uma fila	24
Receber e excluir uma mensagem	24
Confirmar se uma fila está vazia	26
Excluir uma fila	27
Limpar uma fila	28
Filas padrão	30
Entrega de pelo menos uma vez no Amazon SQS	31
Identificadores de filas e mensagens	31
Identificadores de filas padrão	31
Filas FIFO	33
Termos-chave de fila FIFO	34

Lógica de entrega de FIFO	36
Enviar mensagens	36
Recebimento de mensagens	37
Repetir várias vezes	38
Notas adicionais sobre o comportamento de FIFO	38
Exemplos para uma melhor compreensão	39
Processamento exatamente uma vez	39
Migração de uma fila padrão para uma fila FIFO	40
Comportamento de simultaneidade do Lambda e de filas FIFO	41
Agrupamento de mensagens de fila FIFO	42
Simultaneidade do Lambda com filas FIFO	42
Exemplos de casos de uso	43
Alta taxa de transferência para filas FIFO	43
Casos de uso	44
Partições e distribuição de dados	44
Habilitar throughput alto para filas FIFO	47
Identificadores de filas e mensagens	48
Identificadores de filas FIFO	31
Identificadores adicionais para filas FIFO	50
Cotas	51
Cotas de fila FIFO	51
Cotas do Amazon SQS	51
Cotas de fila padrão	53
Cotas de mensagens	55
Cotas de política	61
Recursos e capacidades	62
Filas de mensagens não entregues	62
Usar políticas para filas de mensagens não entregues	63
Compreender os períodos de retenção de mensagens para filas de mensagens não entregues	63
Configurar uma fila de mensagens não entregues	64
Configurar redirecionamento de fila de mensagens não entregues	65
CloudTrail requisitos de atualização e permissão	75
Como criar alarmes para filas de mensagens não entregues usando o Amazon CloudWatch	80
Metadados de mensagens para o Amazon SQS	80

Atributos de mensagens	80
Atributos do sistema de mensagens	84
Recursos necessários para processar mensagens	85
Listar paginação de filas	86
Tags de alocação de custos	86
Sondagem curta e longa	87
Consumo de mensagens usando sondagem curta	87
Consumo de mensagens usando a sondagem longa	88
Diferenças entre as sondagens longa e curta	89
Tempo limite de visibilidade	89
Casos de uso do tempo limite de visibilidade	90
Como definir e ajustar o tempo limite	90
Cotas e mensagens em trânsito	91
Compreender o tempo limite de visibilidade em filas FIFO e padrão	92
Tratamento de falhas	92
Alterar e encerrar o tempo limite de visibilidade	92
Práticas recomendadas	93
Filas justas	94
Diferença com filas FIFO	96
Uso de filas justas	96
Métricas do CloudWatch de filas justas	96
Filas de atraso	97
Filas temporárias	98
Filas virtuais	99
Padrão de mensagens de resposta a solicitação (filas virtuais)	100
Cenário de exemplo: processar uma solicitação de login	101
Limpeza das filas	103
Temporizadores de mensagens	104
Acessar pipes do EventBridge	104
Gerenciamento de mensagens grandes	106
Usar a biblioteca do cliente em versão ampliada para Java	106
Usar a biblioteca do cliente em versão ampliada para Python	113
Configurando o Amazon SQS	117
ABAC para o Amazon SQS	117
O que é ABAC?	117
Por que devo usar o ABAC no Amazon SQS?	118

Marcação para controle de acesso	119
Criação de usuários do IAM e filas do Amazon SQS	119
Testar o controle de acesso por atributo	122
Configurar parâmetros de fila	123
Configurar uma política de acesso	125
Configurar SQS de SSE para uma fila	126
Configurar a SSE-KMS para uma fila	127
Configurar tags para uma fila	129
Inscrever uma fila em um tópico	130
Assinaturas entre contas	131
Assinaturas entre regiões	132
Configurar um acionador do Lambda	132
Pré-requisitos	133
Automação de notificações usando o EventBridge	134
Atributos de mensagens	134
Práticas recomendadas	136
Gerenciamento de erros e mensagens problemáticas	136
Gerenciamento de erros de solicitação no Amazon SQS	136
Capturar mensagens problemáticas no Amazon SQS	137
Configurar a retenção da fila de mensagens não entregues no Amazon SQS	137
Desduplicação e agrupamento de mensagens	138
Evitar o processamento inconsistente de mensagens no Amazon SQS	138
Uso do ID de eliminação de duplicação de mensagens	138
Uso do ID do grupo de mensagens	141
Uso do ID de tentativa de solicitação de recebimento	143
Tempo e processamento de mensagens	143
Processar mensagens em tempo hábil no Amazon SQS	144
Configurar a sondagem longa no Amazon SQS	145
Usar o modo de sondagem apropriado no Amazon SQS	146
Exemplos de SDK do Java	147
Usar criptografia no lado do servidor	147
Adicionar SSE a uma fila existente	147
Desabilitar a SSE para uma fila	149
Criar uma fila com SSE	149
Recuperar atributos de SSE	150
Configurar tags	150

Listar tags	150
Adicionar ou atualizar tags	151
Remover tags	151
Enviar atributos de mensagens	152
Definir atributos	152
Enviar uma mensagem com atributos	154
Usar APIs	155
Fazendo solicitações de API de consulta usando o AWS protocolo JSON	156
Criar um endpoint	156
Como fazer uma solicitação POST	157
Interpretar as respostas da API JSON do Amazon SQS	158
Protocolo Amazon SQS JSON AWS FAQs	159
Fazer solicitações de API de consulta usando o protocolo de consulta da AWS	163
Criar um endpoint	163
Como fazer uma solicitação GET	164
Como fazer uma solicitação POST	157
Interpretar as respostas da API XML do Amazon SQS	165
Autenticação de solicitações	167
Processo de autenticação básica com HMAC-SHA	167
Parte 1: a solicitação do usuário	169
Parte 2: a resposta da AWS	169
Ações em lote	170
Agrupar ações de mensagem em lotes	171
Habilitar o buffer no lado do cliente e o agrupamento de solicitações em lote com o Amazon SQS	172
Aumento do throughput usando escalabilidade horizontal e processamento de ações em lote com o Amazon SQS	185
Como trabalhar com AWS SDKs	197
Usar o JMS	199
Pré-requisitos	199
Usar a Biblioteca de Mensagens Java	200
Criação de uma conexão do JMS	201
Criar uma fila do Amazon SQS	202
Envio de mensagens de forma síncrona	203
Recebimento de mensagens de forma síncrona	204
Recebimento de mensagens de forma assíncrona	206

Uso do modo de reconhecimento do cliente	207
Uso do modo de reconhecimento não ordenado	208
Usar o cliente JMS com outros clientes do Amazon SQS	209
Exemplos de Java funcionais para usar o JMS com filas padrão	210
ExampleConfiguration.java	210
TextMessageSender.java	213
SyncMessageReceiver.java	215
AsyncMessageReceiver.java	217
SyncMessageReceiverClientAcknowledge.java	219
SyncMessageReceiverUnorderedAcknowledge.java	222
SpringExampleConfiguration.xml	226
SpringExample.java	227
ExampleCommon.java	230
Implementações JMS 1.1 com suporte	231
Interfaces comuns com suporte	231
Tipos de mensagens com suporte	232
Modos de reconhecimento de mensagens com suporte	232
Cabeçalhos definidos pelo JMS e propriedades reservadas	232
Tutoriais	234
Criar uma fila do Amazon SQS usando o CloudFormation	234
Enviar uma mensagem a partir de uma VPC	236
Etapa 1: criar um par de chaves do Amazon EC2	237
Etapa 2: criar recursos da AWS	237
Etapa 3: confirmar que sua instância do EC2 não é acessível publicamente	238
Etapa 4: criar um endpoint da Amazon VPC para o Amazon SQS	239
Etapa 5: enviar uma mensagem para sua fila do Amazon SQS	241
Exemplos de código	242
Conceitos básicos	243
Olá, Amazon SQS	244
Ações	256
Cenários	433
Crie um aplicativo de mensagem	434
Criar uma aplicação de messageiro	435
Criar uma aplicação de exploração do Amazon Textract	436
Criar e publicar em um tópico FIFO	437
Detectar pessoas e objetos em um vídeo	450

Gerenciar mensagens grandes usando o S3	451
Processar notificações de eventos do S3	455
Publicar mensagens em filas	459
Enviar e receber lotes de mensagens	596
Use a estrutura de processamento de AWS mensagens para.NET com o Amazon SQS	628
Utilizar a biblioteca de mensagens Java do Amazon SQS para trabalhar com a interface do JMS	628
Trabalhar com tags de fila	650
Exemplos sem servidor	654
Invocar uma função do Lambda em um trigger do Amazon SQS	654
Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS ..	663
Solução de problemas	674
Erro de acesso negado	674
Política de filas do Amazon SQS e política do IAM	675
Permissões do AWS Key Management Service (AWS KMS)	676
Política de endpoint da VPC	677
Política de controle de serviço da organização	678
Erros de API	678
Erro QueueDoesNotExist	678
Erro InvalidAttributeValue	679
Erro ReceiptHandle	679
Problemas de DLQ e redirecionamento de DLQ	680
Problemas de DLQ	680
Problemas de redirecionamento de DLQ	682
Problemas de controle de utilização de FIFO	684
Mensagens não retornadas para uma chamada de API ReceiveMessage	685
Fila vazia	685
Limite de trânsito atingido	685
Atraso de mensagens	685
A mensagem está em trânsito	686
Método de sondagem	686
Erros de rede	686
ETIMEOUT error	686
UnknownHostException error	687
Solução de problemas de filas usando o X-Ray	688
Segurança	690

Proteção de dados	690
Criptografia de dados	691
Privacidade do tráfego entre redes	704
Como usar endpoints de pilha dupla para conectividade	706
Gerenciamento de identidade e acesso	706
Público	706
Autenticação com identidades	707
Gerenciar o acesso usando políticas	708
Visão geral do	710
Como o Amazon Simple Queue Service funciona com o IAM	717
AWS políticas gerenciadas	723
Solução de problemas	726
Usar políticas do	728
Registro em log e monitoramento	777
Registrar em log chamadas de API	780
Monitoramento de filas	784
Validação de conformidade	804
Resiliência	805
Filas distribuídas	805
Segurança da infraestrutura	806
Práticas recomendadas	807
Garantir que as filas não sejam acessíveis ao público	807
Implemente o privilégio de acesso mínimo	807
Usar funções do IAM para aplicações e produtos da AWS que exigem acesso ao Amazon SQS	808
Implemente a criptografia no lado do servidor	808
Aplique a criptografia de dados em trânsito	809
Considere usar endpoints da VPC para acessar o Amazon SQS	809
Recursos relacionados	810
Histórico de documentação	811
.....	dcccxx

O que é o Amazon Simple Queue Service?

O Amazon Simple Queue Service (Amazon SQS) oferece uma fila hospedada segura, durável e disponível que permite integrar e desacoplar sistemas de software e componentes distribuídos. O Amazon SQS oferece constructos comuns, como [filas de mensagens mortas](#) e [tags de alocação de custos](#). Ele fornece uma API de serviço da Web genérica e pode ser acessado por qualquer linguagem de programação compatível com o AWS SDK.

Benefícios do uso do Amazon SQS

- **Segurança:** [você controla](#) quem pode enviar e receber mensagens em uma fila do Amazon SQS. Você pode optar pela transmissão de dados sigilosos protegendo o conteúdo das mensagens em filas por meio da criptografia do lado do servidor (SSE) gerenciada pelo Amazon SQS ou das chaves de [SSE](#) gerenciadas no AWS Key Management Service (AWS KMS).
- **Durabilidade:** para garantir a segurança de suas mensagens, o Amazon SQS as armazena em vários servidores. As filas comuns são compatíveis com a [entrega de mensagens pelo menos uma vez](#), e as filas FIFO são compatíveis com o [processamento de mensagens exatamente uma vez](#) e com o modo de [alto throughput](#).
- **Disponibilidade:** o Amazon SQS usa [infraestrutura redundante](#) para fornecer acesso altamente simultâneo às mensagens, além de alta disponibilidade para produzir e consumir mensagens.
- **Escalabilidade:** o Amazon SQS pode processar cada [solicitação em buffer](#) de forma independente, escalando de forma transparente para lidar com qualquer aumento ou pico de carga sem nenhuma instrução de provisionamento.
- **Confiabilidade:** o Amazon SQS bloqueia suas mensagens durante o processamento, para que vários produtores possam enviar, e vários consumidores possam receber, mensagens ao mesmo tempo.
- **Personalização:** suas filas não precisam ser exatamente iguais. [Você pode definir um atraso padrão em uma fila](#), por exemplo. Você pode armazenar o conteúdo de mensagens maiores que 1 MiB [usando o Amazon Simple Storage Service \(Amazon S3\)](#) ou o Amazon DynamoDB, com o Amazon SQS mantendo um ponteiro no objeto do Amazon S3, ou pode dividir uma mensagem grande em mensagens menores.

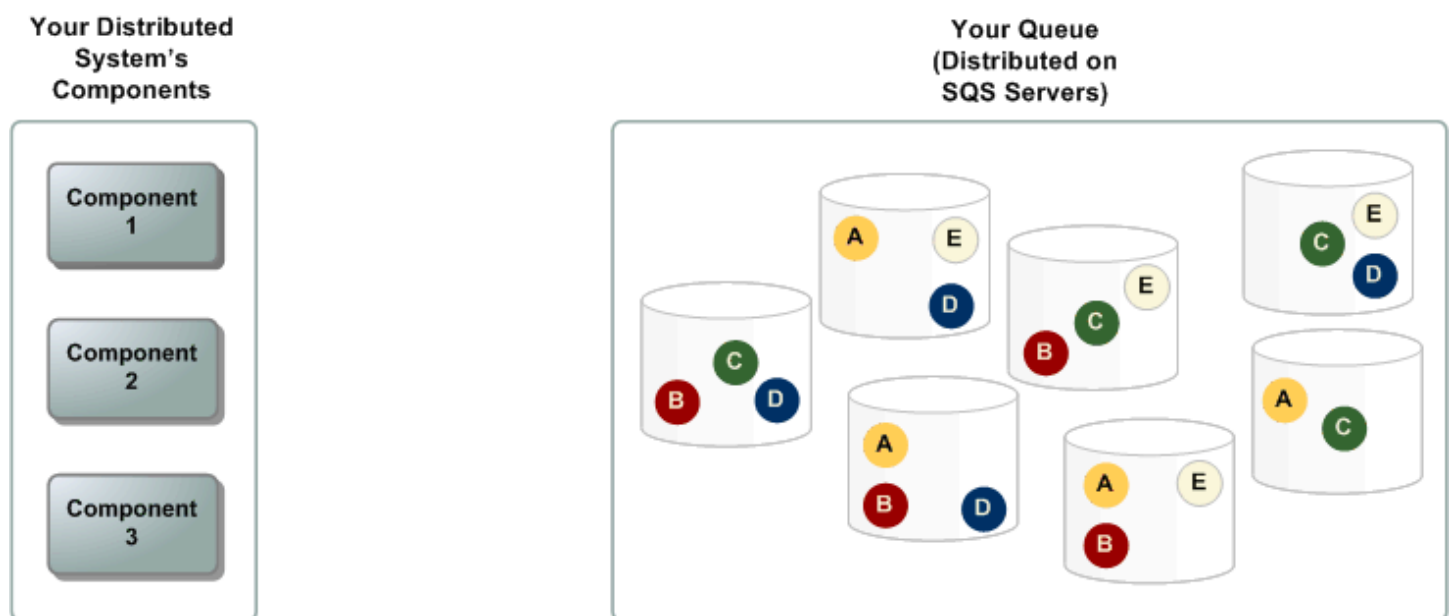
Arquitetura básica do Amazon SQS

Esta seção descreve os componentes de um sistema de mensagens distribuído e explica o ciclo de vida de uma mensagem do Amazon SQS.

Filas distribuídas

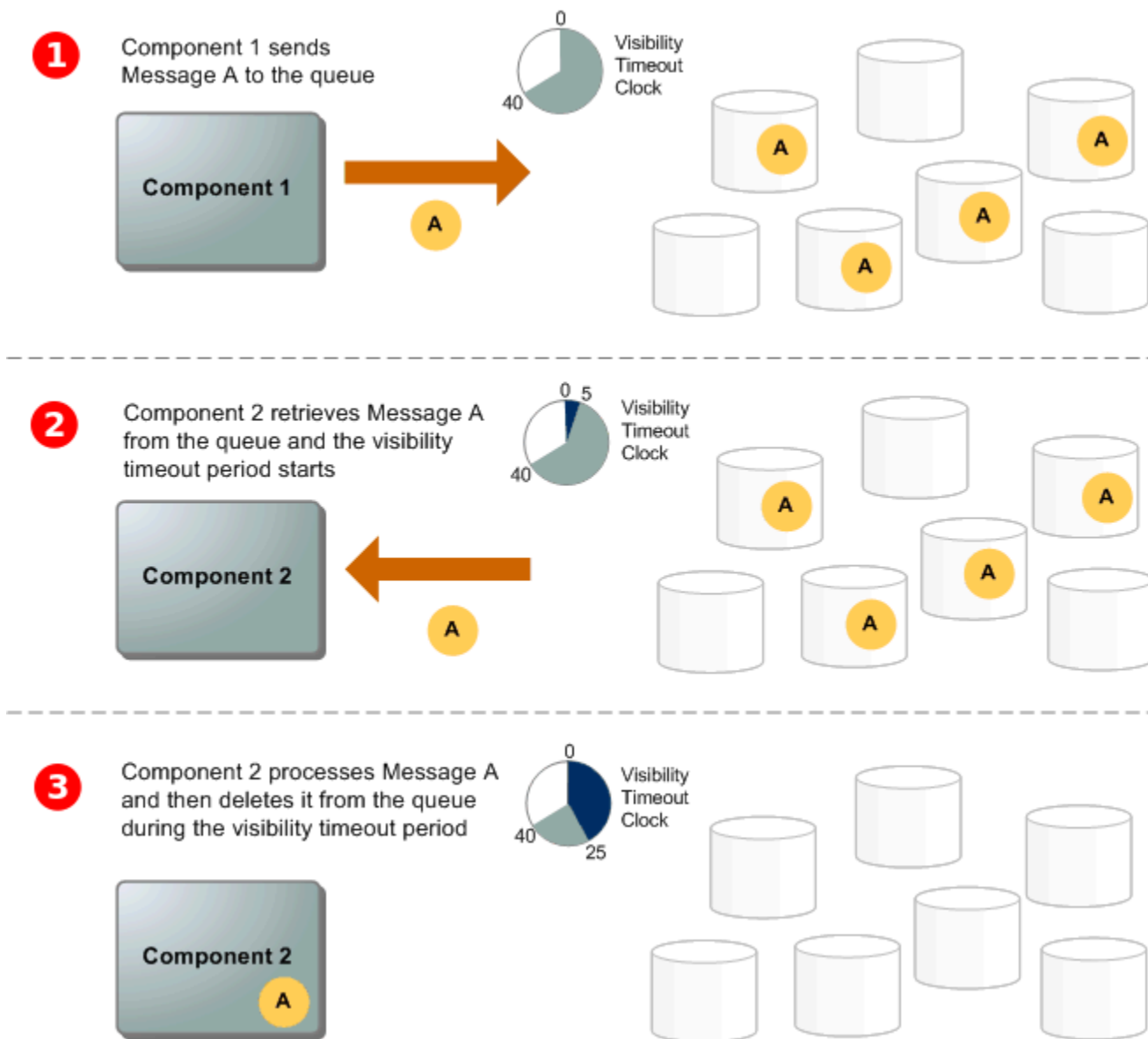
Há três partes principais em um sistema de mensagens distribuído: os componentes do sistema distribuído, a fila (distribuída em servidores do Amazon SQS) e as mensagens na fila.

No cenário a seguir, o sistema tem vários produtores (componentes que enviam mensagens para a fila) e consumidores (componentes que recebem mensagens da fila). A fila (que contém as mensagens A a E) armazena as mensagens de forma redundante em vários servidores do Amazon SQS.



Ciclo de vida de mensagens

O cenário a seguir descreve o ciclo de vida de uma mensagem do Amazon SQS em uma fila, da criação à exclusão.



1 Um produtor (componente 1) envia a mensagem A para uma fila, e a mensagem é distribuída pelos servidores do Amazon SQS de forma redundante.

2 Quando um consumidor (componente 2) está pronto para processar mensagens, ele consome as mensagens da fila, e a mensagem A é devolvida. Enquanto a mensagem A está sendo processada, ela permanece na fila e não é devolvida para as solicitações de recebimento subsequentes durante todo o [tempo limite de visibilidade](#).

3

O consumidor (componente 2) exclui a mensagem A da fila para impedir que a mensagem seja recebida e processada novamente quando o tempo limite de visibilidade for esgotado.

Note

O Amazon SQS exclui automaticamente as mensagens que estiverem em uma fila por mais tempo que o período de retenção máximo de mensagens. O período de retenção de mensagens padrão é de quatro dias. No entanto, você pode configurar o período de retenção de mensagens em um valor de 60 segundos a 1.209.600 segundos (14 dias) usando a ação [SetQueueAttributes](#)

Diferenças entre o Amazon SQS, Amazon MQ e Amazon SNS

O Amazon SQS, o [Amazon SNS](#) e o [Amazon MQ](#) oferecem serviços gerenciados de mensagens altamente escaláveis e fáceis de usar, cada um projetado para funções específicas em sistemas distribuídos. Veja abaixo uma visão geral aprimorada das diferenças entre esses serviços:

O Amazon SQS desacopla e escala sistemas e componentes de software distribuídos como um serviço de fila. Normalmente, ele processa mensagens por meio de um único assinante, ideal para fluxos de trabalho em que a ordem e a prevenção de perdas é fundamental. Para uma distribuição mais ampla, a integração do Amazon SQS com o Amazon SNS permite um [padrão de mensagens de fanout](#), enviando mensagens de forma eficaz para vários assinantes ao mesmo tempo.

O Amazon SNS permite que os publicadores enviem mensagens a vários assinantes por meio de tópicos, que servem como canais de comunicação. Os assinantes recebem mensagens publicadas usando um tipo de endpoint compatível, como [Amazon Data Firehose](#), [Amazon SQS](#), [Lambda](#), HTTP, e-mail, notificações por push em dispositivos móveis e mensagens de texto em dispositivos móveis (SMS). Esse serviço é ideal para cenários que exigem notificações imediatas, como engajamento de usuários em tempo real ou sistemas de alarme. Para evitar a perda de mensagens quando os assinantes estão offline, a integração do Amazon SNS com as mensagens de fila do Amazon SQS garante uma entrega consistente.

O Amazon MQ é ideal para empresas que desejam migrar de agentes de mensagens tradicionais, oferecendo suporte a protocolos de mensagens padrão, como AMQP e MQTT, com o [Apache ActiveMQ](#) e o [RabbitMQ](#). Ele oferece compatibilidade com sistemas legados que precisam de mensagens estáveis e confiáveis sem reconfiguração significativa.

O seguinte gráfico fornece uma visão geral dos tipos de recurso de cada serviço:

Tipo de recurso	Amazon SNS	Amazon SQS	Amazon MQ
Síncrona	Não	Não	Sim
Assíncrona	Sim	Sim	Sim
Filas	Não	Sim	Sim
Sistema de publicado r e assinante de mensagens	Sim	Não	Sim
Agente de mensagens	Não	Não	Sim

O Amazon SQS e o Amazon SNS são recomendados para novas aplicações que podem se beneficiar de uma escalabilidade praticamente ilimitada e de APIs simples. Eles geralmente oferecem soluções mais econômicas para aplicações de alto volume com preço conforme o uso. Recomendamos o Amazon MQ para migrar aplicações de agentes de mensagens em vigor que dependem da compatibilidade com APIs, como o JMS, ou protocolos, como Advanced Message Queuing Protocol (AMQP), MQTT, OpenWire e Simple Text Oriented Message Protocol (STOMP).

Conceitos básicos do Amazon SQS

Este tópico traz orientações para uso do console do Amazon SQS para criar e gerenciar filas padrão e filas FIFO. Você aprenderá a navegar no console, visualizar os atributos da fila e distinguir entre os tipos de fila. As principais tarefas incluem enviar, receber e configurar mensagens, ajustar parâmetros como tempo limite de visibilidade e retenção de mensagens, e gerenciar o acesso à fila por meio de políticas.

Tópicos

- [Configurar o Amazon SQS](#)
- [Noções básicas sobre o console do Amazon SQS](#)
- [Tipos de fila do Amazon SQS](#)
- [Criar uma fila padrão do Amazon SQS e enviar uma mensagem](#)
- [Criar uma fila FIFO do Amazon SQS e enviar uma mensagem](#)
- [Tarefas comuns para começar a usar o Amazon SQS](#)

Configurar o Amazon SQS

Antes de usar o Amazon SQS pela primeira vez, siga as etapas abaixo:

Etapa 1: criar um usuário Conta da AWS e IAM

Para acessar qualquer AWS serviço, primeiro você precisa criar uma [Conta da AWS](#) conta da Amazon.com que possa usar AWS produtos. Você pode usar o seu Conta da AWS para visualizar seus relatórios de atividade e uso e para gerenciar a autenticação e o acesso.

Para evitar o uso do usuário Conta da AWS raiz para ações do Amazon SQS, é uma prática recomendada criar um usuário do IAM para cada pessoa que precisa de acesso administrativo ao Amazon SQS.

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra a <https://portal.aws.amazon.com/billing/inscrição>.

2. Siga as instruções online.

Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar [tarefas que exigem acesso de usuário-raiz](#).

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando <https://aws.amazon.com/e> escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS Centro de Identidade do AWS IAM, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login [Console de gerenciamento da AWS](#) como proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira a senha.

Para obter ajuda ao fazer login usando o usuário-raiz, consulte [Fazer login como usuário-raiz](#) no Guia do usuário do Início de Sessão da AWS.

2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

Para obter instruções, consulte [Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz \(console\) no Guia](#) do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilita o Centro de Identidade do IAM.

Para obter instruções, consulte [Habilitar o Centro de Identidade do AWS IAM](#) no Guia do usuário do Centro de Identidade do AWS IAM.

2. No Centro de Identidade do IAM, conceda o acesso administrativo a um usuário.

Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte [Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM](#) no Guia Centro de Identidade do AWS IAM do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com o seu usuário do Centro de Identidade do IAM, use o URL de login enviado ao seu endereço de e-mail quando o usuário do Centro de Identidade do IAM foi criado.

Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte [Como fazer login no portal de AWS acesso](#) no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

Para obter instruções, consulte [Criar um conjunto de permissões](#) no Guia do usuário do Centro de Identidade do AWS IAM.

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de logon único ao grupo.

Para obter instruções, consulte [Adicionar grupos](#) no Guia do usuário do Centro de Identidade do AWS IAM.

Etapa 2: conceder acesso programático

Para usar ações do Amazon SQS (por exemplo, usando Java ou por meio do AWS Command Line Interface), você precisa de um ID de chave de acesso e uma chave de acesso secreta.

Note

O ID da chave de acesso e a chave de acesso secreta são específicos de AWS Identity and Access Management. Não os confunda com credenciais de outros AWS serviços, como pares de EC2 chaves da Amazon.

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do Console de gerenciamento da AWS. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Recomendado) Use as credenciais do console como credenciais temporárias para assinar solicitações programáticas para o AWS CLI, AWS SDKs ou AWS APIs	Siga as instruções da interface que deseja utilizar. - Para o AWS CLI, consulte Login para desenvolvimento AWS local no Guia AWS Command Line Interface do usuário. - Para AWS SDKs isso, consulte Login para desenvolvimento AWS local no Guia de referência de ferramentas AWS SDKs e ferramentas.
Identidade da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou AWS APIs	Siga as instruções da interface que deseja utilizar. - Para o AWS CLI, consulte Configurando o AWS CLI para uso Centro de Identidade do AWS IAM no Guia do AWS Command Line Interface usuário. - Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referênci

Qual usuário precisa de acesso programático?	Para	Por
		a de ferramentas AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou AWS APIs	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou AWS APIs	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para ferramentas AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de ferramentas AWS SDKs e ferramentas. • Para isso AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Etapa 3: preparar-se para usar o código de exemplo

Este guia inclui exemplos que usam o AWS SDK for Java. Para executar o código de exemplo, siga as instruções de configuração em [Conceitos básicos do AWS SDK for Java 2.0](#).

Você pode desenvolver AWS aplicativos em outras linguagens de programação, como GoJavaScript, Python e Ruby. Para obter mais informações, consulte [Ferramentas para desenvolver AWS](#).

Note

Você pode explorar o Amazon SQS sem escrever código com ferramentas como o AWS Command Line Interface(AWS CLI) ou o Windows. PowerShell Você pode encontrar AWS CLI exemplos na [seção Amazon SQS](#) da Referência de AWS CLI Comandos. Você pode encontrar PowerShell exemplos do Windows na seção Amazon Simple Queue Service da Referência de [Ferramentas da AWS para PowerShell Cmdlet](#).

Próximas etapas

Agora, você já está com tudo pronto para [começar](#) a gerenciar filas e mensagens do Amazon SQS usando o Console de gerenciamento da AWS.

Noções básicas sobre o console do Amazon SQS

Ao abrir o console do Amazon SQS, escolha Filas no painel de navegação. A página Queues (Filas) fornece informações sobre todas as filas na região ativa.

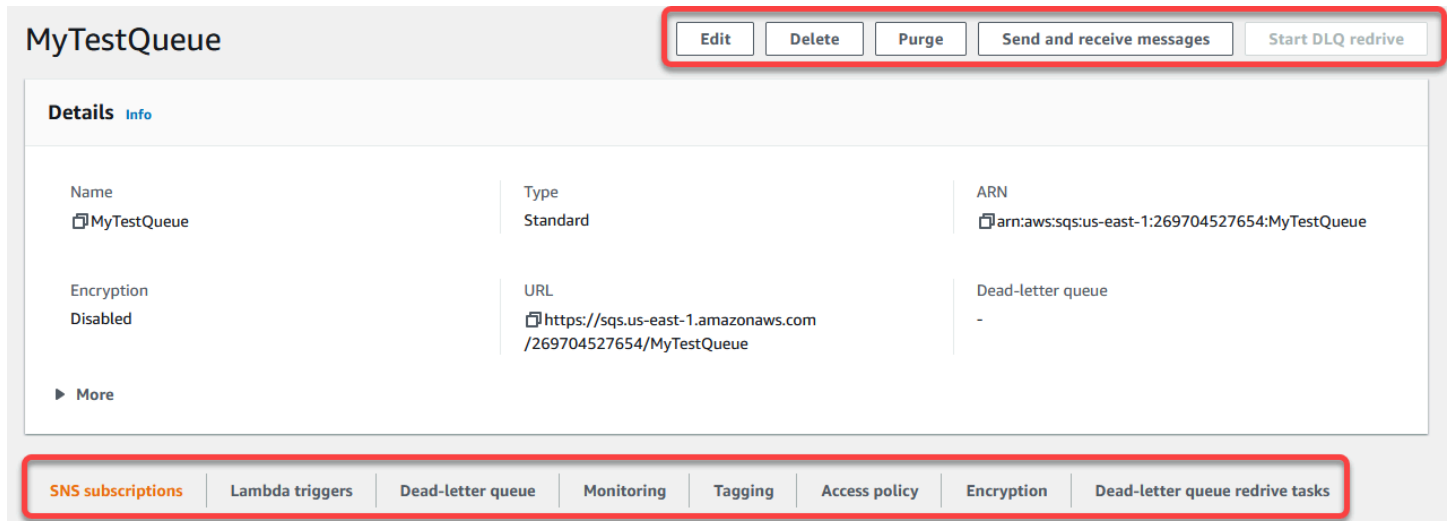
Cada entrada da fila fornece informações essenciais sobre ela, incluindo seu tipo e atributos principais. As [filas padrão](#), otimizadas para throughput máximo e melhor ordenação de mensagens, são diferenciadas das filas [primeiro a entrar, primeiro a sair \(FIFO\)](#), que priorizam a ordenação e a exclusividade das mensagens para aplicações que exigem um sequenciamento estrito de mensagens.

Queues (2)			Edit	Delete	Send and receive messages	Actions ▾	Create queue
Search queues by prefix		< 1 >					
	Name ▲	Type ▼	Created ▼	Messages available ▼	Messages in flight ▼	Encryption ▼	Content-based deduplication ▼
☐	MyTestQueue	Standard	6/27/2022, 13:03:08 EDT	0	0	Disabled	-
☐	testFifo1.fifo	FIFO	6/27/2022, 13:03:41 EDT	0	0	Disabled	Disabled

Elementos interativos e ações

Na página “Filas”, você tem várias opções para gerenciar suas filas:

1. **Ações rápidas:** ao lado de cada nome de fila, um menu suspenso oferece acesso rápido a ações comuns, como enviar mensagens, visualizar ou excluir mensagens, configurar gatilhos e excluir a própria fila.
2. **Visualização e configuração detalhadas:** clicar no nome de uma fila abre a página de detalhes, onde você pode se aprofundar nas configurações e definições da fila. Aqui, você pode ajustar parâmetros como período de retenção de mensagens, tempo limite de visibilidade e tamanho máximo da mensagem para adaptar a fila aos requisitos da aplicação.



Seleção de região e tags de recursos

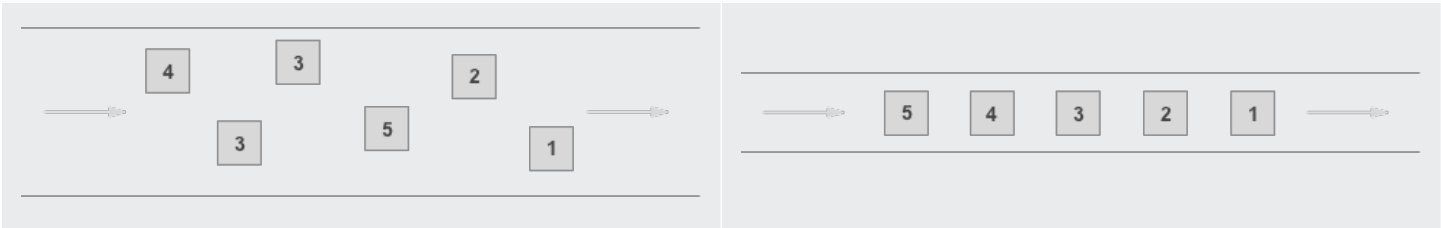
Certifique-se de que você está na Região da AWS certa para acessar e gerenciar suas filas de forma eficaz. Além disso, considere a utilização de tags de recursos para organizar e categorizar suas filas, permitindo melhor gerenciamento de recursos, alocação de custos e controle de acesso em seu ambiente compartilhado da AWS.

Ao aproveitar os recursos e funcionalidades oferecidos no console do Amazon SQS, você pode gerenciar com eficiência sua infraestrutura de mensagens, otimizar o desempenho das filas e garantir a entrega confiável de mensagens para as aplicações.

Tipos de fila do Amazon SQS

O Amazon SQS é compatível com dois tipos de fila: [filas padrão](#) e [filas FIFO](#). Use a tabela a seguir para determinar qual fila melhor atende às suas necessidades.

Filas padrão	Filas FIFO
Throughput ilimitado: as filas padrão são compatíveis com um número muito alto e quase ilimitado de chamadas de API por segundo, por ação (SendMessage , ReceiveMessage ou DeleteMessage). Esse throughput alto as torna ideais para casos de uso que exigem o processamento rápido de grandes volumes de mensagens, como fluxo de dados em tempo real ou aplicações de grande escala. Embora as filas padrão sejam escaladas automaticamente de acordo com a demanda, é essencial monitorar os padrões de uso para garantir o desempenho ideal, especialmente em regiões com workloads mais altas. Entrega pelo menos uma vez: entrega garantida pelo menos uma vez, o que significa que cada mensagem é entregue pelo menos uma vez, mas, em alguns casos, uma mensagem pode ser entregue mais de uma vez devido a novas tentativas ou atrasos na rede. Você deve projetar a aplicação para lidar com possíveis mensagens duplicadas usando operações idempotentes, que garantem que o processamento da mesma mensagem várias vezes não afete o estado do sistema. Ordenação com melhor esforço: isso significa que, embora o Amazon SQS tente entregar as mensagens na ordem em que foram enviadas, ele não garante isso. Em alguns casos, as mensagens podem chegar fora de ordem, principalmente em situações de throughput alto ou recuperação de falhas. Para aplicação	Throughput alto: com o agrupamento em lote, as filas FIFO processam até 3.000 mensagens por segundo, por método de API (SendMessageBatch , ReceiveMessage ou DeleteMessageBatch). Esse throughput depende de 300 chamadas de API por segundo, em que cada uma processe um lote de 10 mensagens. Ao habilitar o modo de throughput alto, é possível aumentar a escala verticalmente para até 30 mil transações por segundo (TPS) com ordenação relaxada em grupos de mensagens. Sem o agrupamento em lote, as filas FIFO permitem até 300 chamadas de API por segundo, por método de API (SendMessage , ReceiveMessage ou DeleteMessage). Se precisar de mais throughput, solicite um aumento de cota no AWS Support Center. Consulte como habilitar o modo de throughput alto em Habilitar throughput alto para filas FIFO no Amazon SQS. Processamento exatamente uma vez: as filas FIFO entregam cada mensagem uma vez e a mantêm disponível até que ela seja processada e excluída. Ao usar recursos como MessageDeduplicationId ou a deduplicação baseada em conteúdo, você evita mensagens duplicadas, mesmo em novas tentativas devido a problemas de rede ou tempo limite. Entrega de primeiro a entrar, primeiro a sair: as filas FIFO garantem que você receba as mensagens na ordem em que são enviadas em cada grupo de mensagens. Ao distribui

Filas padrão	Filas FIFO
s em que a ordem do processamento das mensagens é crucial, você deve lidar com a lógica de reordenação dentro da aplicação ou usar filas FIFO para garantir uma ordenação estrita. Durabilidade e redundância: as filas padrão garantem alta durabilidade ao armazenar várias cópias de cada mensagem em várias zonas de disponibilidade da AWS. Isso garante que as mensagens não sejam perdidas, mesmo em caso de falhas na infraestrutura. Tempo limite de visibilidade: o Amazon SQS permite configurar um tempo limite de visibilidade para controlar por quanto tempo uma mensagem permanece oculta após ser recebida, garantindo que outros consumidores não processem a mensagem até que ela tenha sido totalmente gerenciada ou que o tempo limite expire.	r mensagens em vários grupos, você pode processá-las paralelamente, mantendo a ordem em cada grupo. 

Filas padrão	Filas FIFO
Use filas padrão para enviar dados entre aplicações quando o throughput for crucial, por exemplo: • Desacoplar solicitações do usuário em tempo real de trabalhos intensos em segundo plano. Permita que os usuários façam upload de mídia rapidamente enquanto você processa tarefas como redimensionamento ou codificação em segundo plano, garantindo um tempo de resposta rápido sem sobrecarregar o sistema. • Alocar tarefas para vários nós de processamento. Distribua um alto número de solicitações de validação de cartão de crédito em vários nós de processamento e gerencie mensagens duplicadas com operações idempotentes para evitar erros de processamento. • Agrupar mensagens em lote para processamento futuro. Coloque várias entradas na fila para adições em lote a um banco de dados. Como a ordem das mensagens não é garantida, projete o sistema para gerenciar o processamento fora de ordem, se necessário.	Use filas FIFO para enviar dados entre aplicações quando a ordem dos eventos for importante, por exemplo: • Garantir que os comandos inseridos pelo usuário sejam executados na ordem correta. Esse é um caso de uso importante para filas FIFO, em que a ordem dos comandos é crucial. Por exemplo, se um usuário executa uma sequência de ações em uma aplicação, as filas FIFO garantem que as ações sejam processadas na mesma ordem em que foram inseridas. • Exibir o preço do produto correto enviando modificações de preço na ordem correta. As filas FIFO garantem que várias atualizações do preço de um produto cheguem e sejam processadas sequencialmente. Sem as filas FIFO, uma redução de preço pode ser processada após um aumento de preço, fazendo com que dados incorretos sejam exibidos. • Impedir que um aluno se inscreva em um curso antes de criar uma conta. Ao usar filas FIFO, você garante que o processo de inscrição ocorra na sequência certa. O sistema processa primeiro o registro da conta e depois a inscrição no curso, evitando que a solicitação de inscrição seja executada prematuramente.

Implementar sistemas de solicitação-resposta no Amazon SQS

Ao implementar um sistema de solicitação-resposta ou chamada de procedimento remoto (RPC), lembre-se das seguintes melhores práticas:

- Crie filas de resposta na inicialização: em vez de criar filas de resposta por mensagem, crie-as na inicialização, por produtor. Use um atributo de mensagem de ID de correlação para mapear respostas às solicitações de forma eficiente.
- Evite compartilhar filas de resposta entre produtores: garanta que cada produtor tenha sua própria fila de respostas. Compartilhar filas de resposta pode fazer com que um produtor receba mensagens de resposta destinadas a outro.

Consulte mais informações sobre a implementação do padrão de solicitação-resposta usando o Temporary Queue Client em [Padrão de mensagens de resposta a solicitação \(filas virtuais\)](#).

Criar uma fila padrão do Amazon SQS e enviar uma mensagem

É possível criar uma [fila padrão](#) e enviar mensagens usando o console do Amazon SQS. Este tópico também enfatiza as práticas recomendadas, incluindo evitar informações confidenciais em nomes de filas e utilizar criptografia gerenciada do lado do servidor.

Como criar uma fila padrão usando o console do Amazon SQS

Important

Em 17 de agosto de 2022, a criptografia do lado do servidor (SSE) padrão foi aplicada a todas as filas do Amazon SQS.

Não inclua informações de identificação pessoal (PII) nem outras informações confidenciais ou sigilosas em nomes de filas. Os nomes de filas são acessíveis para vários Amazon Web Services, incluindo logs de faturamento e o CloudWatch Logs. Os nomes de filas não devem ser usados para dados privados ou sigilosos.

Para criar uma fila padrão do Amazon SQS

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. Selecione Criar fila.

3. Para o tipo, a fila do tipo padrão é definida por padrão.

 Note

Não é possível alterar o tipo de uma fila depois de criá-la.

4. Insira um Name (Nome) para a fila.
5. (Opcional) O console define valores padrão para os [parâmetros de configuração](#) da fila. Em Configuration (Configuração), você pode definir novos valores para os seguintes parâmetros:
 - a. Em Visibility timeout (Tempo limite de visibilidade), insira a duração e as unidades. O intervalo é de 0 segundo a 12 horas. O valor padrão de é 30 segundos.
 - b. Em Message retention period (Período de retenção de mensagens), insira a duração e as unidades. O intervalo é de 1 minuto a 14 dias. O valor padrão é 4 dias.
 - c. Em Delivery delay (Atraso de entrega), insira a duração e as unidades. O intervalo é de 0 segundo a 15 minutos. O valor de padrão é 0 segundos.
 - d. Em Maximum message size (Tamanho máximo da mensagem), insira um valor. O intervalo é de 1 KiB a 1024 KiB. O valor padrão é 1024 KiB.
 - e. Em Receive message wait time (Tempo de espera da mensagem), insira um valor. O intervalo é de 0 a 20 segundos. O valor padrão é 0 segundo, o que define uma [sondagem curta](#). Qualquer valor diferente de zero define uma sondagem longa.
6. (Opcional) Defina uma política de acesso. A [política de acesso](#) define as contas, usuários e funções que podem acessar a fila. A política de acesso também define as ações (como [SendMessage](#), [ReceiveMessage](#) ou [DeleteMessage](#)) que os usuários podem acessar. A política padrão permite que apenas o proprietário da fila envie e receba mensagens.

Para definir a política de acesso, realize um dos seguintes procedimentos:

- Escolha Basic (Básico) para configurar quem pode enviar mensagens para a fila e quem pode receber mensagens dela. O console cria a política com base em suas escolhas e exibe a política de acesso resultante no painel JSON somente leitura.
 - Escolha Advanced (Avançado) para modificar a política de acesso JSON diretamente. Isso permite que você especifique um conjunto personalizado de ações que cada entidade (conta, usuário ou função) pode executar.
7. Em Redrive allow policy (Política de permissão de redirecionamento), escolha Enabled (Habilitada). Selecione uma das seguintes opções: Allow all (Permitir tudo), By queue (Por fila)

ou Deny all (Negar tudo). Ao escolher By queue (Por fila), especifique uma lista de até 10 filas de origem pelo nome do recurso da Amazon (ARN).

8. O Amazon SQS fornece criptografia do lado do servidor gerenciada por padrão. Para escolher um tipo de chave de criptografia ou desabilitar a criptografia do lado do servidor gerenciada pelo Amazon SQS, expanda Encryption (Criptografia). Para obter mais informações sobre os tipos de chave de criptografia, consulte [Configurar a criptografia do lado do servidor para uma fila usando chaves de criptografia gerenciadas pelo SQS](#) e [Configurar a criptografia do lado do servidor para uma fila usando o console do Amazon SQS](#).

 Note

Com a SSE habilitada, as solicitações anônimas SendMessage e ReceiveMessage à fila criptografada serão rejeitadas. As práticas recomendadas de segurança do Amazon SQS não aconselham o uso de solicitações anônimas. Se você quiser enviar solicitações anônimas a uma fila do Amazon SQS, desabilite o SSE.

9. (Opcional) Para configurar uma [fila de mensagens mortas](#) para receber mensagens que não podem ser entregues, expanda Dead-letter queue (Fila de mensagens mortas).
10. (Opcional) Para adicionar [tags](#) à fila, expanda Tags.
11. Selecione Criar fila. O Amazon SQS cria a fila e exibe a página Details (Detalhes) da fila.

O Amazon SQS propaga as informações sobre a nova fila pelo sistema. Como o Amazon SQS é um sistema distribuído, você pode enfrentar um pequeno atraso antes que o console exiba a fila na página Queues (Filas).

Como enviar uma mensagem usando uma fila padrão

Depois de criar a fila, você pode enviar uma mensagem para ela.

1. No painel de navegação à esquerda, escolha Queues (Filas). Na lista de filas, selecione a fila que você criou.
2. Em Actions (Ações), escolha Send and receive messages (Enviar e receber mensagens).

O console exibe a página Send and receive messages (Enviar e receber mensagens).

3. Em Message (Mensagem), insira o texto da mensagem.

4. Para uma fila padrão, é possível inserir um valor para Atraso de entrega e escolher as unidades. Por exemplo, insira 60 e escolha seconds (segundos). Para obter mais informações, consulte [Temporizadores de mensagens do Amazon SQS](#).
5. Escolha Send Message (Enviar mensagem).

Quando a mensagem é enviada, o console exibe uma mensagem de sucesso. Escolha View details (Visualizar os detalhes) para exibir informações sobre a mensagem enviada.

Criar uma fila FIFO do Amazon SQS e enviar uma mensagem

Você pode criar uma fila FIFO do Amazon SQS e enviar mensagens usando o console. Este tópico explica como configurar parâmetros de fila, incluindo tempo limite de visibilidade, retenção de mensagens e deduplicação, seguindo as práticas recomendadas de segurança, como evitar informações confidenciais em nomes de filas e ativar a criptografia do lado do servidor. Também abrange a definição de políticas de acesso, a configuração de filas de mensagens não entregues e o envio de mensagens com atributos específicos FIFO, como ID do grupo de mensagens e ID de deduplicação.

Como criar uma fila FIFO usando o console do Amazon SQS

É possível usar o console do Amazon SQS para criar [filas FIFO](#). O console fornece valores padrão para todas as configurações, exceto para o nome da fila.

Important

Em 17 de agosto de 2022, a criptografia do lado do servidor (SSE) padrão foi aplicada a todas as filas do Amazon SQS.

Não inclua informações de identificação pessoal (PII) nem outras informações confidenciais ou sigilosas em nomes de filas. Os nomes de filas são acessíveis para vários Amazon Web Services, incluindo logs de faturamento e o CloudWatch Logs. Os nomes de filas não devem ser usados para dados privados ou sigilosos.

Para criar uma fila FIFO do Amazon SQS

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. Selecione Criar fila.

3. Para o tipo, a fila do tipo padrão é definida por padrão. Para criar uma fila FIFO, escolha FIFO.

 Note

Não é possível alterar o tipo de uma fila depois de criá-la.

4. Insira um Name (Nome) para a fila.

O nome de uma fila FIFO deve terminar com o sufixo `.fifo`. O sufixo conta para a cota de 80 caracteres do nome da fila. Para determinar se uma fila é [FIFO](#), você pode conferir se o nome da fila termina com o sufixo.

5. (Opcional) O console define valores padrão para os [parâmetros de configuração](#) da fila. Em Configuration (Configuração), você pode definir novos valores para os seguintes parâmetros:
 - a. Em Visibility timeout (Tempo limite de visibilidade), insira a duração e as unidades. O intervalo é de 0 segundo a 12 horas. O valor padrão de é 30 segundos.
 - b. Em Message retention period (Período de retenção de mensagens), insira a duração e as unidades. O intervalo é de 1 minuto a 14 dias. O valor padrão é 4 dias.
 - c. Em Delivery delay (Atraso de entrega), insira a duração e as unidades. O intervalo é de 0 segundo a 15 minutos. O valor de padrão é 0 segundos.
 - d. Em Maximum message size (Tamanho máximo da mensagem), insira um valor. O intervalo é de 1 KiB a 1024 KiB. O valor padrão é 1024 KiB.
 - e. Em Receive message wait time (Tempo de espera da mensagem), insira um valor. O intervalo é de 0 a 20 segundos. O valor padrão é 0 segundo, o que define uma [sondagem curta](#). Qualquer valor diferente de zero define uma sondagem longa.
 - f. Para uma fila FIFO, escolha Content-based deduplication (Eliminação de duplicação baseada em conteúdo) para habilitar a eliminação de duplicação baseada em conteúdo. Por padrão, essa configuração está desabilitada.
 - g. (Opcional) Em uma fila FIFO, para habilitar um throughput mais alto a fim de enviar e receber mensagens na fila, escolha Enable high throughput FIFO (Habilitar FIFO de alto throughput).

Escolher esta opção altera as opções relacionadas (Deduplication scope [Escopo de eliminação de duplicação] e FIFO throughput limit [Limite de transferência FIFO]) para as configurações necessárias a fim de habilitar a alta taxa de transferência para filas FIFO. Se você alterar qualquer uma das configurações necessárias para usar FIFO de alta taxa de transferência, a taxa de transferência normal permanecerá em vigor para a fila e a

eliminação de duplicação ocorrerá conforme especificado. Para obter mais informações, consulte [Throughput alto para filas FIFO no Amazon SQS](#) e [Cotas de mensagens do Amazon SQS](#).

6. (Opcional) Defina uma política de acesso. A [política de acesso](#) define as contas, usuários e funções que podem acessar a fila. A política de acesso também define as ações (como [SendMessage](#), [ReceiveMessage](#) ou [DeleteMessage](#)) que os usuários podem acessar. A política padrão permite que apenas o proprietário da fila envie e receba mensagens.

Para definir a política de acesso, realize um dos seguintes procedimentos:

- Escolha Basic (Básico) para configurar quem pode enviar mensagens para a fila e quem pode receber mensagens dela. O console cria a política com base em suas escolhas e exibe a política de acesso resultante no painel JSON somente leitura.
 - Escolha Advanced (Avançado) para modificar a política de acesso JSON diretamente. Isso permite que você especifique um conjunto personalizado de ações que cada entidade (conta, usuário ou função) pode executar.
7. Em Redrive allow policy (Política de permissão de redirecionamento), escolha Enabled (Habilitada). Selecione uma das seguintes opções: Allow all (Permitir tudo), By queue (Por fila) ou Deny all (Negar tudo). Ao escolher By queue (Por fila), especifique uma lista de até 10 filas de origem pelo nome do recurso da Amazon (ARN).
 8. O Amazon SQS fornece criptografia do lado do servidor gerenciada por padrão. Para escolher um tipo de chave de criptografia ou desabilitar a criptografia do lado do servidor gerenciada pelo Amazon SQS, expanda Encryption (Criptografia). Para obter mais informações sobre os tipos de chave de criptografia, consulte [Configurar a criptografia do lado do servidor para uma fila usando chaves de criptografia gerenciadas pelo SQS](#) e [Configurar a criptografia do lado do servidor para uma fila usando o console do Amazon SQS](#).

 Note

Com a SSE habilitada, as solicitações anônimas `SendMessage` e `ReceiveMessage` à fila criptografada serão rejeitadas. As práticas recomendadas de segurança do Amazon SQS não aconselham o uso de solicitações anônimas. Se você quiser enviar solicitações anônimas a uma fila do Amazon SQS, desabilite o SSE.

9. (Opcional) Para configurar uma [fila de mensagens mortas](#) para receber mensagens que não podem ser entregues, expanda Dead-letter queue (Fila de mensagens mortas).
10. (Opcional) Para adicionar [tags](#) à fila, expanda Tags.

11. Selecione Criar fila. O Amazon SQS cria a fila e exibe a página Details (Detalhes) da fila.

O Amazon SQS propaga as informações sobre a nova fila pelo sistema. Como o Amazon SQS é um sistema distribuído, você pode enfrentar um pequeno atraso antes que o console exiba a fila na página Queues (Filas).

Depois de criar uma fila, você pode [enviar mensagens](#) para ela e [receber e excluir mensagens](#). Você também pode [editar](#) qualquer uma das definições de configuração de fila, exceto o tipo de fila.

Como enviar uma mensagem usando uma fila FIFO

Depois de criar sua fila, você pode enviar uma mensagem para ela.

1. No painel de navegação à esquerda, escolha Queues (Filas). Na lista de filas, selecione a fila que você criou.
2. Em Actions (Ações), escolha Send and receive messages (Enviar e receber mensagens).

O console exibe a página Send and receive messages (Enviar e receber mensagens).

3. Em Message (Mensagem), insira o texto da mensagem.
4. Para uma fila FIFO (primeiro a entrar, primeiro a sair), insira um ID do grupo de mensagens. Para obter mais informações, consulte [Lógica de entrega de filas FIFO no Amazon SQS](#).
5. (Opcional) Para uma fila FIFO, você pode inserir um ID de eliminação de duplicação de mensagens. Se você habilitou a eliminação de duplicação baseada em conteúdo para a fila, o ID de eliminação de duplicação de mensagens não será necessário. Para obter mais informações, consulte [Lógica de entrega de filas FIFO no Amazon SQS](#).
6. As filas FIFO não são compatíveis com temporizadores em mensagens individuais. Para obter mais informações, consulte [Temporizadores de mensagens do Amazon SQS](#).
7. Escolha Send Message (Enviar mensagem).

Quando a mensagem é enviada, o console exibe uma mensagem de sucesso. Escolha View details (Visualizar os detalhes) para exibir informações sobre a mensagem enviada.

Tarefas comuns para começar a usar o Amazon SQS

Agora que você criou uma fila e aprendeu a enviar, receber e excluir mensagens, assim como excluir uma fila, experimente o seguinte:

- Acione uma [função do Lambda](#) para processar mensagens recebidas automaticamente, permitindo fluxos de trabalho orientados por eventos sem a necessidade de sondagem contínua.
- [Configurar filas, incluindo SSE e outros recursos](#).
- [Enviar uma mensagem com atributos](#).
- [Enviar uma mensagem a partir de uma VPC](#).
- Conheça a [funcionalidade](#) e a [arquitetura](#) do Amazon SQS.
- Conheça as [diretrizes e advertências](#) que ajudarão você a aproveitar ao máximo o Amazon SQS.
- Explore os exemplos do Amazon SQS para um AWS SDK, como o [Guia do desenvolvedor do AWS SDK for Java 2.x](#).
- Saiba mais sobre os [comandos da AWS CLI do Amazon SQS](#).
- Saiba mais sobre as [ações da API do Amazon SQS](#).
- Saiba como interagir com o Amazon SQS de forma programática. Consulte [Trabalhando com APIs](#) e explore o [Centro desenvolvimento da AWS](#):
 - [Java](#)
 - [JavaScript](#)
 - [PHP](#)
 - [Python](#)
 - [Ruby](#)
 - [Windows e .NET](#)
- Saiba como monitorar [custos e recursos](#).
- Saiba como [proteger seus dados](#).
- Saiba mais sobre o [fluxo de trabalho do Amazon SQS](#).

Gerenciar uma fila do Amazon SQS

Aprenda a gerenciar filas do Amazon SQS usando o console, incluindo edição de configurações de filas, recebimento e exclusão de mensagens, confirmação de vazios de fila e exclusão ou eliminação de filas. Entenda as práticas recomendadas para o tratamento eficiente de mensagens, como o uso de pesquisas longas, o gerenciamento de tempos limite de visibilidade e a verificação de métricas por meio de painéis de monitoramento ou da AWS CLI. Siga as etapas práticas para manter as filas e lidar com as mensagens de forma eficaz, minimizando as interrupções.

Editar uma fila do Amazon SQS usando o console

Você pode usar o console do Amazon SQS para editar parâmetros de configuração de fila (exceto o tipo de fila) e modificar ou remover recursos conforme necessário.

Para editar uma fila do Amazon SQS (console)

1. Abra a [página Queues](#) (Filas) do console do Amazon SQS.
2. Selecione uma fila e escolha Edit (Editar).
3. (Opcional) Em Configuration (Configuração), atualize os [parâmetros de configuração](#) da fila.
4. (Opcional) Para atualizar a [política de acesso](#), em Política de acesso, modifique a política JSON.
5. (Opcional) Para atualizar a [política de permissão de redirecionamento](#) de uma fila de mensagens não entregues, expanda Redrive allow policy (Política de permissão de redirecionamento).
6. (Opcional) Para atualizar ou remover a [criptografia](#), expanda Encryption (Criptografia).
7. (Opcional) Para adicionar, atualizar ou remover uma [fila de mensagens mortas](#) (que permite receber mensagens que não podem ser entregues), expanda Dead-letter queue (Fila de mensagens mortas).
8. (Opcional) Para adicionar, atualizar ou remover as [tags](#) da fila, expanda Tags.
9. Escolha Salvar.
 - O console exibe a página Details (Detalhes) da fila.

Receber e excluir uma mensagem no Amazon SQS

Depois de enviar mensagens para uma fila do Amazon SQS, você pode recuperá-las e excluí-las para processar o fluxo de trabalho da sua aplicação. Esse processo garante o tratamento seguro

e confiável das mensagens. Este tópico explica como recuperar e excluir mensagens usando o console do Amazon SQS e explica as principais configurações para otimizar essa operação. A seguir estão os principais conceitos para receber e excluir mensagens:

1. Recebimento de mensagens

- Ao recuperar mensagens de uma fila do Amazon SQS, não é possível direcionar mensagens específicas. Em vez disso, você deve especificar o número máximo de mensagens a serem recuperadas em uma única solicitação (até 10).
- Devido à natureza distribuída do Amazon SQS, a recuperação de uma fila com poucas mensagens pode retornar uma resposta vazia. Para mitigar isso:
 - Use uma sondagem longa, que espera até que uma mensagem esteja disponível ou que a sondagem atinja o tempo limite. Essa abordagem reduz os custos desnecessários de sondagem e melhora a eficiência.
 - Reemita a solicitação, se necessário.

2. Visibilidade e exclusão de mensagens

- As mensagens não são excluídas automaticamente após a recuperação. Esse recurso garante que você possa reprocessar mensagens em caso de falhas na aplicação ou interrupções na rede.
- Após o processamento, você deve enviar explicitamente uma solicitação de exclusão para remover a mensagem permanentemente. Essa ação confirma o sucesso do tratamento.
- As mensagens recuperadas usando o console do Amazon SQS permanecem visíveis para recuperação. Ajuste a configuração de tempo limite de visibilidade para ambientes automatizados para ocultar temporariamente as mensagens de outros consumidores enquanto estão sendo processadas.

3. Tempo limite de visibilidade

- Essa configuração determina por quanto tempo uma mensagem permanece oculta após a recuperação. Defina um tempo limite apropriado para garantir que as mensagens sejam processadas somente uma vez e para evitar duplicações durante o processamento distribuído.

Como receber e excluir uma mensagem usando console

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.

3. Na página Filas, selecione a fila da qual deseja receber mensagens e escolha Enviar e receber mensagens.
4. Na página Enviar e receber mensagens, escolha Pesquisar mensagens.

O Amazon SQS exibe uma barra de progresso indicando a duração da sondagem. As mensagens recuperadas aparecerão na seção Mensagens, mostrando:

- ID de mensagem
 - Data do envio
 - Tamanho
 - Contagem de recebimentos
5. Para excluir mensagens, escolha aquelas que deseja remover e selecione Excluir.

Confirme a exclusão na caixa de diálogo Excluir mensagens selecionando Excluir.

Para obter mais detalhes sobre operações avançadas, incluindo recuperação e exclusão de mensagens baseadas em API, consulte o [Guia de referência de API do Amazon SQS](#).

Confirmar se uma fila do Amazon SQS está vazia

Na maioria dos casos, você pode usar a [sondagem longa](#) para determinar se uma fila está vazia. Em casos raros, você pode receber respostas vazias mesmo quando uma fila ainda contém mensagens, especialmente se você especificar um valor baixo para o tempo de espera da mensagem quando criar a fila. Esta seção descreve como confirmar se uma fila está vazia.

Para confirmar se uma fila está vazia (console)

1. Interrompa o envio de mensagens por todos os produtores.
2. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
3. No painel de navegação, escolha Queues.
4. Na página Queues (Filas), escolha uma fila.
5. Escolha a guia Monitoring (Monitoramento).
6. No canto superior direito dos painéis de monitoramento, escolha a seta para baixo ao lado do símbolo Refresh (Atualizar). No menu suspenso, escolha Auto refresh (Atualização automática). Deixe Refresh interval (Atualização do intervalo) como 1 Minute (1 minuto).
7. Observe os seguintes painéis:

- Número aproximado de mensagens atrasadas
- Número aproximado de mensagens não visíveis
- Número aproximado de mensagens visíveis

Quando todos eles mostram valores 0 por vários minutos, a fila está vazia.

Para confirmar se uma fila está vazia (API da AWS CLI)

1. Interrompa o envio de mensagens por todos os produtores.
2. Execute repetidamente um dos seguintes comandos:
 - AWS CLI: [get-queue-attributes](#)
 - AWS API: [GetQueueAttributes](#)
3. Observe as métricas dos seguintes atributos:
 - `ApproximateNumberOfMessagesDelayed`
 - `ApproximateNumberOfMessagesNotVisible`
 - `ApproximateNumberOfMessagesVisible`

Quando todos eles são 0 por vários minutos, a fila está vazia.

Se você confiar nas métricas do Amazon CloudWatch, certifique-se de ver vários pontos de dados zero consecutivos antes de considerar essa fila vazia. Para obter mais informações sobre métricas do CloudWatch, consulte [Métricas disponíveis do CloudWatch para o Amazon SQS](#).

Excluir uma fila do Amazon SQS

Se você não usa mais uma fila do Amazon SQS e não planeja usá-la em um futuro próximo, faça a exclusão.

Tip

Se você quiser verificar se uma fila está vazia antes de excluí-la, consulte [Confirmar se uma fila do Amazon SQS está vazia](#).

Você pode excluir uma fila mesmo quando ela não estiver vazia. Para excluir as mensagens em uma fila, mas não a própria fila, [limpe a fila](#).

Para excluir uma fila (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Na página Queues (Filas), escolha a fila a ser excluída.
4. Escolha Excluir.
5. Na caixa de diálogo Delete queue (Excluir fila), confirme a exclusão inserindo **delete**.
6. Escolha Excluir.

Como excluir uma fila (AWS CLI e API)

Escolha o método apropriado para excluir sua fila com base em suas necessidades:

- AWS CLI: [aws sqs delete-queue](#)
- AWS API: [DeleteQueue](#)

Limpar mensagens de uma fila usando o console do Amazon SQS

Para manter uma fila do Amazon SQS, mas remover todas as suas mensagens, você pode limpar a fila. Isso excluirá todas as mensagens, incluindo aquelas que estão atualmente invisíveis (em trânsito). O processo de limpeza pode levar até 60 segundos; portanto, aguarde os 60 segundos completos, independentemente do tamanho da fila.

Important

Quando você limpar uma fila, não poderá recuperar nenhuma mensagem excluída.

Para limpar uma fila (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Na página Queues (Filas), escolha a fila a ser limpa.

4. Em Actions (Ações), escolha Purge (Limpar).
5. Na caixa de diálogo Purge queue (Limpar fila), confirme a limpeza inserindo **purge** e escolhendo Purge (Limpar).
 - Todas as mensagens são removidas da fila. O console exibe uma banner de confirmação.

Filas padrão do Amazon SQS

O Amazon SQS fornece filas padrão como o tipo de fila padrão, permitindo um número quase ilimitado de chamadas de API por segundo para ações como [SendMessage](#), [ReceiveMessage](#) e [DeleteMessage](#). Filas padrão garantem a entrega de mensagens pelo menos uma vez, porém, por causa da arquitetura altamente distribuída, mais de uma cópia de uma mensagem pode ser entregue e as mensagens às vezes podem chegar fora de ordem. Apesar disso, as filas padrão fazem o possível para manter a ordem em que as mensagens são enviadas.

Quando você envia uma mensagem usando `SendMessage`, o Amazon SQS armazena redundantemente a mensagem em várias zonas de disponibilidade (AZs) antes de confirmá-la. Essa redundância garante que nenhuma falha em um único computador, rede ou AZ possa tornar as mensagens inacessíveis.

É possível criar e configurar filas usando o console do Amazon SQS. Para ter instruções detalhadas, consulte [Como criar uma fila padrão usando o console do Amazon SQS](#). Consulte exemplos específicos de Java em [Exemplos de SDK do Java do Amazon SQS](#).

Casos de uso para filas padrão

Filas de mensagens padrão são adequadas para vários cenários, contanto que a aplicação possa lidar com as mensagens que cheguem mais de uma vez ou fora de ordem. Os exemplos incluem:

- Separar as solicitações de usuários ao vivo do trabalho intensivo em segundo plano: os usuários podem fazer upload de mídia enquanto o sistema a redimensiona ou codifica em segundo plano.
- Alocar tarefas para vários nós de processamento: por exemplo, lidar com um alto volume de solicitações de validação de cartão de crédito.
- Agrupar mensagens em lote para processamento futuro: programação de várias entradas para adicioná-las a um banco de dados posteriormente.

Consulte informações sobre cotas relacionadas a filas padrão em [Cotas de fila padrão do Amazon SQS](#).

Para as práticas recomendadas ao trabalhar com filas padrão, consulte [Práticas recomendadas do Amazon SQS](#).

Entrega de pelo menos uma vez no Amazon SQS

O Amazon SQS armazena cópias de suas mensagens em vários servidores para obter redundância e alta disponibilidade. Em raras ocasiões, um dos servidores que armazena a cópia de uma mensagem poderá ficar indisponível quando você receber ou excluir uma mensagem.

Se isso acontecer, a cópia da mensagem não será excluída no servidor que está indisponível, e você poderá obter a cópia da mensagem novamente quando receber mensagens. Projete aplicativos para serem idempotentes (para não serem afetados quando a mesma mensagem é processada mais de uma vez).

Identificadores de mensagens e filas do Amazon SQS

Este tópico descreve os identificadores de filas padrão e FIFO. Esses identificadores podem ajudar a localizar e manipular filas e mensagens específicas.

Identificadores de filas padrão do Amazon SQS

Para obter mais informações sobre os seguintes identificadores, consulte a [Referência da API do Amazon Simple Storage Service](#).

Nome e URL da fila

Ao criar uma nova fila, você deve especificar o nome de uma fila exclusivo para sua conta e região da AWS. O Amazon SQS atribui a cada fila que você cria um identificador chamado URL da fila, que inclui o nome da fila e outros componentes do Amazon SQS. Sempre que você desejar executar uma ação em uma fila, forneça o URL da fila.

O seguinte URL é de uma fila chamada MyQueue, de propriedade de um usuário com o número de conta da AWS 123456789012.

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue
```

É possível recuperar o URL de uma fila programaticamente listando as filas e analisando a string que segue o número da conta. Para obter mais informações, consulte [ListQueues](#).

ID de mensagem

Cada mensagem recebe um ID da mensagem atribuído pelo sistema que o Amazon SQS retorna para você na resposta [SendMessage](#). Esse identificador é útil para identificar mensagens. O tamanho máximo de um ID de mensagem é 100 caracteres.

Identificador de recebimento

Toda vez que você recebe uma mensagem de uma fila, recebe um identificador de recebimento dessa mensagem. Esse identificador é associado à ação de recebimento da mensagem, e não à mensagem. Para excluir a mensagem ou alterar a visibilidade da mensagem, você deve fornecer o identificador de recebimento (não o ID de mensagem). Desse modo, você sempre deve receber uma mensagem antes de excluí-la (você não pode colocar uma mensagem na fila e, em seguida, recuperá-la). O tamanho máximo de um identificador de recebimento é 1024 caracteres.

Important

Se você receber uma mensagem mais de uma vez, cada vez que recebê-la, obterá um identificador de recebimento diferente. Você deve fornecer o identificador de recebimento recebido mais recentemente ao solicitar a exclusão da mensagem (caso contrário, a mensagem pode não ser excluída).

Veja a seguir um exemplo de um identificador de recebimento quebrado em três linhas.

```
MbZj6wDWli+JvwWJaBV+3dcjk2YW2vA3+STFF1jTM8tJJg6HRG6PYSasuWXPJB+Cw
Lj1FjgXUv1uSj1gUPAWV66FU/WeR4mq20KpEGYWbnLmpRCJVAyeMjeU5ZBdtcQ+QE
auMZc8ZRv37sIW2iJKq3M9MFx1YvV11A2x/KSbkJ0=
```

Filas FIFO do Amazon SQS

As filas FIFO (primeiro a entrar, primeiro a sair) têm todos os recursos das [filas padrão](#), mas são projetadas para aprimorar o sistema de mensagens entre aplicações quando a ordem das operações e dos eventos é crucial ou quando duplicatas não podem ser toleradas.

Os recursos mais importantes de filas FIFO são [entrega FIFO \(primeiro a entrar, primeiro a sair\)](#) e [processamento exatamente uma vez](#):

- A ordem em que as mensagens são enviadas e recebidas é preservada estritamente, e uma mensagem é entregue uma vez e permanece indisponível até que um consumidor a processe e a exclua.
- As duplicações não são introduzidas na fila.

Além disso, as filas FIFO também são compatíveis com grupos de mensagens, que permitem diversos grupos de mensagens ordenadas em uma única fila. Não há cota para o número de grupos de mensagens dentro de uma fila FIFO.

Exemplos de situações em que você pode usar filas FIFO incluem os seguintes:

1. Sistema de gerenciamento de pedidos de comércio eletrônico em que o pedido é essencial
2. Integração com sistemas de terceiros em que os eventos precisam ser processados em ordem
3. Processamento de entradas inseridas pelo usuário no pedido inserido
4. Comunicações e redes: envio e recebimento de dados e informações na mesma ordem
5. Sistemas de computador: garantir que os comandos inseridos pelo usuário sejam executados na ordem correta.
6. Institutos educacionais: impedir que um aluno se matricule em um curso antes de criar uma conta.
7. Sistema de emissão de tíquetes online: no qual os tíquetes são distribuídos por ordem de chegada

Note

As filas FIFO também fornecem processamento exatamente uma vez, mas têm um número limitado de transações por segundo (TPS). É possível usar o modo de alto throughput do Amazon SQS com a fila FIFO para aumentar o limite de transações. Para obter detalhes sobre como usar o modo de alto throughput, consulte [Throughput alto para filas FIFO no](#)

[Amazon SQS](#). Para obter mais informações sobre cotas de taxa de transferência, consulte [the section called “Cotas de mensagens”](#).

As filas FIFO do Amazon SQS estão disponíveis em todas as regiões em que o Amazon SQS está disponível

Para saber mais sobre como usar filas FIFO com pedidos complexos, consulte [Solving Complex Ordering Challenges with Amazon SQS FIFO Queues](#) (Resolver desafios complexos de pedidos com filas FIFO do Amazon SQS).

Para obter informações sobre como criar e configurar filas usando o console do Amazon SQS, consulte [Como criar uma fila padrão usando o console do Amazon SQS](#). Para exemplos de Java, consulte [Exemplos de SDK do Java do Amazon SQS](#).

Consulte as práticas recomendadas ao trabalhar com filas FIFO em [Práticas recomendadas do Amazon SQS](#).

Termos-chave de fila FIFO do Amazon SQS

Os seguintes termos-chave podem ajudar você a entender melhor a funcionalidade das filas FIFO. Para obter mais informações, consulte a [Referência da API do Amazon Simple Queue Service](#).

Cientes

Atualmente, o cliente assíncrono no buffer do Amazon SQS não oferece suporte a filas FIFO.

ID de eliminação de duplicação de mensagens

Um token usado nas filas FIFO do Amazon SQS para identificar mensagens de forma exclusiva e evitar duplicações. Se várias mensagens com o mesmo ID de desduplicação forem enviadas dentro de um intervalo de desduplicação de cinco minutos, elas serão tratadas como duplicatas e somente uma cópia será entregue. Se você não especificar um ID de desduplicação e a desduplicação baseada em conteúdo estiver habilitada, o Amazon SQS gerará um ID de desduplicação por meio do hashing do corpo da mensagem. Esse mecanismo garante a entrega exatamente uma vez, eliminando mensagens duplicadas dentro do prazo especificado.

 Note

O Amazon SQS continua acompanhando o ID de desduplicação da mensagem mesmo depois que a mensagem é recebida e excluída.

ID do grupo de mensagens

Em filas FIFO (First-In First-Out — primeira a entrar, primeira a sair), `MessageGroupId` é um atributo que organiza mensagens em grupos distintos. As mensagens dentro do mesmo grupo são sempre processadas uma por vez, em ordem estrita, garantindo que duas mensagens do mesmo grupo não sejam processadas simultaneamente. Em filas padrão, o uso de `MessageGroupId` ativa [filas justas](#). Se for necessária uma ordenação estrita, use uma fila FIFO.

ID de tentativa de solicitação de recebimento

O ID de tentativa de solicitação de recebimento é um token exclusivo usado para desduplicar chamadas de [ReceiveMessage](#) no Amazon SQS.

Número de sequência

O número grande e não consecutivo que o Amazon SQS atribui a cada mensagem.

Serviços

Se sua aplicação usar vários produtos da AWS ou uma combinação de produtos externos e da AWS, é importante compreender quais funcionalidades dos produtos não oferecem suporte a filas FIFO.

Alguns produtos externos ou da AWS que enviam notificações ao Amazon SQS podem não ser compatíveis com filas FIFO, apesar de permitirem que você defina uma fila FIFO como destino.

Atualmente, os seguintes recursos dos produtos da AWS não são compatíveis com filas FIFO:

- [Notificações de eventos do Amazon S](#)
- [Ganchos do ciclo de vida do Auto Scaling](#)
- [AWS IoT Ações de regras de](#)
- [AWS Lambda Filas de mensagens não entregues do](#)

Para obter informações sobre a compatibilidade de outros produtos com filas FIFO, consulte a documentação do seu produto.

Lógica de entrega de filas FIFO no Amazon SQS

Os conceitos a seguir esclarecem como as filas FIFO do Amazon SQS lidam com o envio e o recebimento de mensagens, especialmente quando lidam com ordenação de mensagens e IDs de grupos de mensagens.

Enviar mensagens

As filas FIFO do Amazon SQS preservam a ordem das mensagens usando IDs de deduplicação e IDs de grupos de mensagens exclusivos. Este tópico destaca a importância dos IDs de grupos de mensagens para manter uma ordenação estrita dentro dos grupos e destaca as práticas recomendadas para garantir a entrega confiável e ordenada de mensagens entre vários produtores.

1. Preservação da ordem

- Se várias mensagens forem enviadas em sucessão a uma fila FIFO, cada uma com um ID de deduplicação de mensagem distinto, o Amazon SQS armazenará as mensagens e confirmará a transmissão. Essas mensagens serão, então, recebidas e processadas na ordem exata em que foram transmitidas.

2. ID do grupo de mensagens

- Em filas FIFO, as mensagens são ordenadas com base no ID de grupo de mensagens. Se vários produtores ou threads enviarem mensagens com o mesmo ID de grupo de mensagens, o Amazon SQS garante que elas sejam armazenadas e processadas na ordem em que chegarem.
- Prática recomendada: para garantir uma ordem estrita das mensagens entre vários produtores, atribua um ID de grupo de mensagens exclusivo para todas as mensagens de cada produtor.

3. Ordenar por grupo

- A lógica da fila FIFO aplica-se de acordo com o ID do grupo de mensagens:
 - Cada ID de grupo de mensagens representa um grupo distinto e ordenado.
 - Para cada ID de grupo de mensagens, todas as mensagens são enviadas e recebidas na ordem estrita.
 - Mensagens com IDs de grupo de mensagens diferentes podem chegar ou ser processadas fora de ordem uma em relação à outra.
- Obrigatório: você deve associar um ID de grupo de mensagens a cada mensagem. Se uma mensagem for enviada sem um ID de grupo, a ação falhará.

- Cenário de grupo único: se você precisar que todas as mensagens sejam processadas em ordem estrita, use o mesmo ID de grupo de mensagens para cada mensagem.

Recebimento de mensagens

As filas FIFO do Amazon SQS gerenciam a recuperação de mensagens, incluindo processamento em lote, garantias de ordem FIFO e limitações na solicitação de IDs de grupos de mensagens específicos. Este tópico explica como o Amazon SQS recupera mensagens dentro e entre IDs de grupos de mensagens, mantendo a ordenação estrita e as regras de visibilidade.

1. Recuperação em lote

- Ao receber mensagens de uma fila FIFO com vários IDs de grupo de mensagens, o Amazon SQS:
 - Tenta retornar o máximo de mensagens possível com o mesmo ID de grupo em uma única chamada.
 - Isso permite que outros clientes processem mensagens de IDs de grupo de mensagens diferentes simultaneamente.
- Esclarecimento importante
 - Você pode receber várias mensagens do mesmo ID de grupo de mensagens em um lote (até 10 mensagens em uma única chamada usando o parâmetro `MaxNumberOfMessages`).
 - No entanto, você não poderá receber mensagens adicionais do mesmo ID de grupo de mensagens em solicitações subsequentes até que:
 - As mensagens atualmente recebidas sejam excluídas ou
 - Elas se tornem visíveis novamente (por exemplo, após o tempo limite de visibilidade expirar).

2. Garantia de ordem FIFO

- As mensagens recuperadas em um lote mantêm sua ordem FIFO dentro do grupo.
- Se menos de 10 mensagens estiverem disponíveis para o mesmo ID de grupo de mensagens, o Amazon SQS poderá incluir mensagens de outros IDs de grupo de mensagens no mesmo lote, mas cada grupo manterá a ordem FIFO.

3. Limitações do consumidor

- Você não pode solicitar explicitamente o recebimento de mensagens de um ID de grupo de mensagens específico.

Repetir várias vezes

Produtores e consumidores podem repetir com segurança ações que falharam nas filas FIFO do Amazon SQS sem interromper a ordem das mensagens ou introduzir duplicatas. Este tópico destaca como os IDs de deduplicação e os tempos limite de visibilidade garantem a integridade da mensagem durante novas tentativas.

1. Novas tentativas do produtor

- Se uma ação [SendMessage](#) falhar, o produtor pode tentar novamente enviar a mensagem várias vezes com o mesmo ID de deduplicação de mensagens.
- Desde que o produtor receba pelo menos uma confirmação antes que o intervalo de deduplicação expire, as novas tentativas:
 - Não introduzem mensagens duplicadas.
 - Não interrompem a ordem das mensagens.

2. Novas tentativas do consumidor

- Se a ação [ReceiveMessage](#) falhar, o consumidor pode tentar novamente quantas vezes forem necessárias usando o mesmo ID de tentativa de solicitação de recebimento.
- Desde que o consumidor receba pelo menos uma confirmação antes que o tempo limite de visibilidade expire, as novas tentativas:
 - Não interrompem a ordem das mensagens.

Notas adicionais sobre o comportamento de FIFO

Saiba mais sobre como lidar com os tempos limite de visibilidade, habilitar o processamento paralelo com vários IDs de grupos de mensagens e garantir o processamento sequencial estrito em cenários de grupo único.

1. Gerenciar o tempo limite de visibilidade

- Quando uma mensagem é recuperada, mas não excluída, ela permanece invisível até que o tempo limite de visibilidade expire.
- Nenhuma mensagem adicional do mesmo ID de grupo de mensagens será retornada até que a primeira mensagem seja excluída ou fique visível novamente.

2. Simultaneidade e processamento paralelo

- As filas FIFO permitem o processamento paralelo de mensagens em diferentes IDs de grupos de mensagens.

- Para maximizar a simultaneidade, projete seu sistema com vários IDs de grupos de mensagens para fluxos de trabalho independentes.

3. Cenários de grupo único

- Para o processamento sequencial estrito de todas as mensagens em uma fila FIFO, use um único ID de grupo de mensagens para todas as mensagens na fila.

Exemplos para uma melhor compreensão

A seguir estão cenários práticos que ilustram o comportamento da fila FIFO no Amazon SQS.

1. Cenário 1: ID de grupo único

- Um produtor envia cinco mensagens com o mesmo ID de grupo de mensagens, Grupo A.
- Um consumidor recebe essas mensagens na ordem FIFO. Até que o consumidor exclua essas mensagens ou o tempo limite de visibilidade expire, nenhuma mensagem adicional do Grupo A será recebida.

2. Cenário 2: vários IDs de grupo

- Um produtor envia cinco mensagens para o Grupo A e 5 para o Grupo B.
- O Consumidor 1 processa as mensagens do Grupo A, enquanto o Consumidor 2 processa as mensagens do Grupo B. Isso permite o processamento paralelo com ordenação estrita mantida dentro de cada grupo.

3. Cenário 3: recuperação em lote

- Um produtor envia sete mensagens para o Grupo A e três para o Grupo B.
- Até 10 mensagens podem ser recuperadas por um único consumidor. Se a fila permitir, poderá retornar:
 - Sete mensagens do Grupo A e três do Grupo B (ou menos, se houver menos mensagens disponíveis em um único grupo).

Processamento exatamente uma vez no Amazon SQS

Ao contrário das filas padrão, as filas FIFO não introduzem mensagens duplicadas. As filas FIFO ajudam a evitar o envio de duplicações para uma fila. Se você tentar novamente a ação `SendMessage` dentro do intervalo de eliminação de duplicação de 5 minutos, o Amazon SQS não introduzirá duplicações na fila.

Para configurar a eliminação de duplicação, você deve realizar umas das seguintes ações:

- Ativar a eliminação de duplicação baseada em conteúdo. Isso instrui o Amazon SQS a usar um hash SHA-256 para gerar o ID de eliminação de duplicação de mensagens usando o corpo da mensagem, mas não os atributos dela. Para obter mais informações, consulte a documentação sobre ações [CreateQueue](#), [GetQueueAttributes](#) e [SetQueueAttributes](#) na Referência da API do Amazon Simple Queue Service.
- Forneça explicitamente o ID de eliminação de duplicação da mensagem (ou visualize o número de sequência) para a mensagem. Para obter mais informações, consulte a documentação sobre ações [SendMessage](#), [SendMessageBatch](#) e [ReceiveMessage](#) na Referência da API do Amazon Simple Queue Service.

Migração de uma fila padrão para uma fila FIFO no Amazon SQS

Se a sua aplicação usa filas padrão e você quiser aproveitar os recursos de ordenação ou de processamento exatamente uma vez das filas FIFO, precisará configurar a fila e sua aplicação corretamente.

Considerações importantes

- Criar uma fila FIFO: não é possível converter uma fila padrão existente em uma fila FIFO. É necessário criar uma fila FIFO para a aplicação ou excluir a fila padrão existente e recriá-la como uma fila FIFO.
- Parâmetro de atraso: as filas FIFO não são compatíveis com atrasos por mensagem, apenas com atrasos por fila. Se a aplicação definir o parâmetro `DelaySeconds` em cada mensagem, você deverá modificá-la para definir `DelaySeconds` em toda a fila.
- ID do grupo de mensagens: forneça um [ID do grupo de mensagens](#) para cada mensagem enviada. Esse ID permite o processamento paralelo de mensagens enquanto mantém a respectiva ordem. Use uma dimensão empresarial mais detalhada para o ID do grupo de mensagens a fim de escalar melhor com as filas FIFO. Quanto maior for o número de IDs de grupos de mensagens para os quais você distribui mensagens, maior será o número de mensagens disponíveis para consumo.
- Modo de throughput alto: use o [modo de throughput alto](#) recomendado para filas FIFO a fim de atingir um throughput maior. Consulte mais informações sobre cotas de mensagens em [Cotas de mensagens do Amazon SQS](#).

Lista de verificação de migração para filas FIFO

Antes de enviar mensagens para uma fila FIFO, confirme o seguinte:

1. Defina configurações de atraso

- Modifique a aplicação para remover atrasos por mensagem.
- Defina o parâmetro `DelaySeconds` em toda a fila.

2. Defina IDs do grupo de mensagens

- Organize as mensagens em grupos especificando um ID do grupo de mensagens em uma dimensão empresarial.
- Use dimensões empresariais mais detalhadas para melhorar a escalabilidade.

3. Lide com a deduplicação de mensagens

- Se a aplicação não puder enviar mensagens com corpos de mensagem idênticos, forneça um ID de deduplicação de mensagem exclusivo para cada mensagem.
- Se a aplicação envia mensagens com corpos de mensagem exclusivos, habilite a deduplicação baseada em conteúdo.

4. Configure o consumidor

- Geralmente, nenhuma alteração de código é necessária para o consumidor.
- Se o processamento de mensagens leva muito tempo e o tempo limite de visibilidade está definido como alto, considere adicionar um ID de tentativa de solicitação de recebimento a cada ação `ReceiveMessage`. Isso ajuda a repetir as tentativas de recebimento em caso de falhas de rede e impede que as filas parem devido a tentativas de recebimento com falha.

Seguindo essas etapas, você pode garantir que a aplicação funcione corretamente com filas FIFO, aproveitando ao máximo seus recursos de ordenação e processamento de exatamente uma vez.

Consulte informações mais detalhadas na [Referência de API do Amazon Simple Queue Service](#).

Comportamento de simultaneidade do Lambda e de filas FIFO do Amazon SQS

Ao usar uma fila FIFO (primeiro entrar, primeiro a sair) com o Lambda, você pode garantir o processamento ordenado de mensagens em cada grupo de mensagens. A função do Lambda não executará várias instâncias para o mesmo grupo de mensagens simultaneamente, mantendo assim a ordem. No entanto, é possível aumentar a escala verticalmente para lidar com vários grupos de

mensagens em paralelo, garantindo o processamento eficiente da workload da fila. Os seguintes pontos descrevem o comportamento das funções do Lambda ao processar mensagens de uma fila FIFO do Amazon SQS em relação aos IDs de grupos de mensagens:

- **Instância única por grupo de mensagens:** a qualquer momento, somente uma instância do Lambda processará mensagens de um ID de grupo de mensagens específico. Isso garante que as mensagens dentro do mesmo grupo sejam processadas em ordem, mantendo a integridade da sequência FIFO.
- **Processamento simultâneo de diferentes grupos:** o Lambda pode processar simultaneamente mensagens de diferentes IDs de grupos de mensagens usando várias instâncias. Isso significa que, enquanto uma instância da função do Lambda está processando mensagens de um ID de grupo de mensagens, outras instâncias podem processar simultaneamente mensagens de outros IDs de grupos de mensagens, utilizando os recursos de simultaneidade do Lambda para processar vários grupos em paralelo.

Agrupamento de mensagens de fila FIFO

As filas FIFO garantem que as mensagens sejam processadas na ordem exata em que foram enviadas. Eles usam um ID de grupo de mensagens para agrupar mensagens que devem ser processadas sequencialmente.

As mensagens dentro do mesmo grupo de mensagens são processadas em ordem, e somente uma mensagem de cada grupo é processada por vez para manter essa ordem.

Simultaneidade do Lambda com filas FIFO

Depois de criar sua fila, você pode enviar uma mensagem para ela.

Quando você configura uma função do Lambda para processar mensagens de uma fila FIFO do Amazon SQS, o Lambda respeita as garantias de ordenação fornecidas pela fila FIFO. Os pontos a seguir descrevem o comportamento das funções do Lambda em termos de simultaneidade e escalabilidade ao processar mensagens de uma fila FIFO do Amazon SQS ao usar IDs de grupos de mensagens.

- **Simultaneidade em grupos de mensagens:** somente uma instância do Lambda processa mensagens para determinado ID de grupo de mensagens por vez. Isso garante que as mensagens dentro de um grupo sejam processadas sequencialmente.

- Escalabilidade e vários grupos de mensagens: embora o Lambda possa aumentar a escala verticalmente para processar mensagens de forma simultânea, essa escalabilidade ocorre em diferentes grupos de mensagens. Se você tiver vários grupos de mensagens, o Lambda poderá processar vários grupos em paralelo, com cada grupo sendo tratado por uma instância separada do Lambda.

Consulte mais informações em [Scaling and concurrency in Lambda](#) no AWS Lambda Operator Guide.

Exemplos de casos de uso

Suponha que a fila FIFO receba mensagens com o mesmo ID de grupo de mensagens e a função do Lambda tenha um alto limite de simultaneidade (até 1.000).

Se uma mensagem do ID do grupo “A” estiver sendo processada e outra mensagem do ID do grupo “A” chegar, a segunda mensagem não acionará uma nova instância do Lambda até que a primeira mensagem seja totalmente processada.

No entanto, se as mensagens dos IDs de grupo “A” e “B” chegarem, ambas poderão ser processadas simultaneamente por instâncias do Lambda separadas.

Throughput alto para filas FIFO no Amazon SQS

As filas FIFO de throughput alto no Amazon SQS gerenciam com eficiência o throughput alto de mensagens enquanto mantêm uma ordem rígida de mensagens, garantindo confiabilidade e escalabilidade para aplicações que processam várias mensagens. Essa solução é ideal para cenários que exigem throughput alto e entrega ordenada de mensagens.

As filas FIFO de throughput alto do Amazon SQS não são necessárias em cenários em que a ordenação estrita de mensagens não é crucial e em que o volume de mensagens recebidas é relativamente baixo ou esporádico. Por exemplo, se você tiver uma aplicação de pequena escala que processa mensagens pouco frequentes ou não sequenciais, a complexidade e o custo adicionais associados às filas FIFO de throughput alto podem não ser justificados. Além disso, se a aplicação não exigir os recursos aprimorados de throughput fornecidos pelas filas FIFO de throughput alto, optar por uma fila padrão do Amazon SQS pode ser mais econômico e mais simples de gerenciar.

Para aumentar a capacidade de solicitação em filas FIFO de throughput alto, é recomendável aumentar o número de grupos de mensagens. Para ter mais informações sobre cotas de mensagens

de throughput alto, consulte [Amazon SQS service quotas](#) no Referência geral da Amazon Web Services.

Consulte informações sobre cotas por fila e estratégias de distribuição de dados em [Cotas de mensagens do Amazon SQS](#) e [Partições e distribuição de dados para alta taxa de transferência para filas FIFO do SQS](#).

Casos de uso de throughput alto para filas FIFO do Amazon SQS

Os seguintes casos de uso destacam as diversas aplicações de filas FIFO de throughput alto, mostrando sua eficácia em todos os setores e cenários:

1. **Processamento de dados em tempo real:** aplicações que lidam com fluxos de dados em tempo real, como processamento de eventos ou ingestão de dados de telemetria, podem se beneficiar de filas FIFO de throughput alto para lidar com o fluxo contínuo de mensagens, preservando sua ordem para uma análise precisa.
2. **Processamento de pedidos de comércio eletrônico:** em plataformas de comércio eletrônico em que manter a ordem das transações do cliente é fundamental, as filas FIFO de throughput alto garantem que os pedidos sejam processados sequencialmente e sem atrasos, mesmo durante os períodos de pico de compras.
3. **Serviços financeiros:** instituições financeiras que lidam com dados comerciais ou transacionais de alta frequência dependem de filas FIFO de throughput alto para processar dados e transações de mercado com latência mínima, ao mesmo tempo que cumprem os rígidos requisitos regulatórios para ordenação de mensagens.
4. **Streaming de mídia:** plataformas de streaming e serviços de distribuição de mídia utilizam filas FIFO de throughput alto para gerenciar a entrega de arquivos de mídia e conteúdo de streaming, garantindo experiências de reprodução suaves para os usuários e mantendo a ordem correta de entrega do conteúdo.

Partições e distribuição de dados para alta taxa de transferência para filas FIFO do SQS

O Amazon SQS armazena dados da fila FIFO em partições. Uma partição é uma alocação de armazenamento para uma fila que é automaticamente replicada em várias zonas de disponibilidade dentro de uma região da AWS. Você não gerencia partições. Em vez disso, o Amazon SQS lida com o gerenciamento de partições

Para filas FIFO, o Amazon SQS modifica o número de partições em uma fila nas seguintes situações:

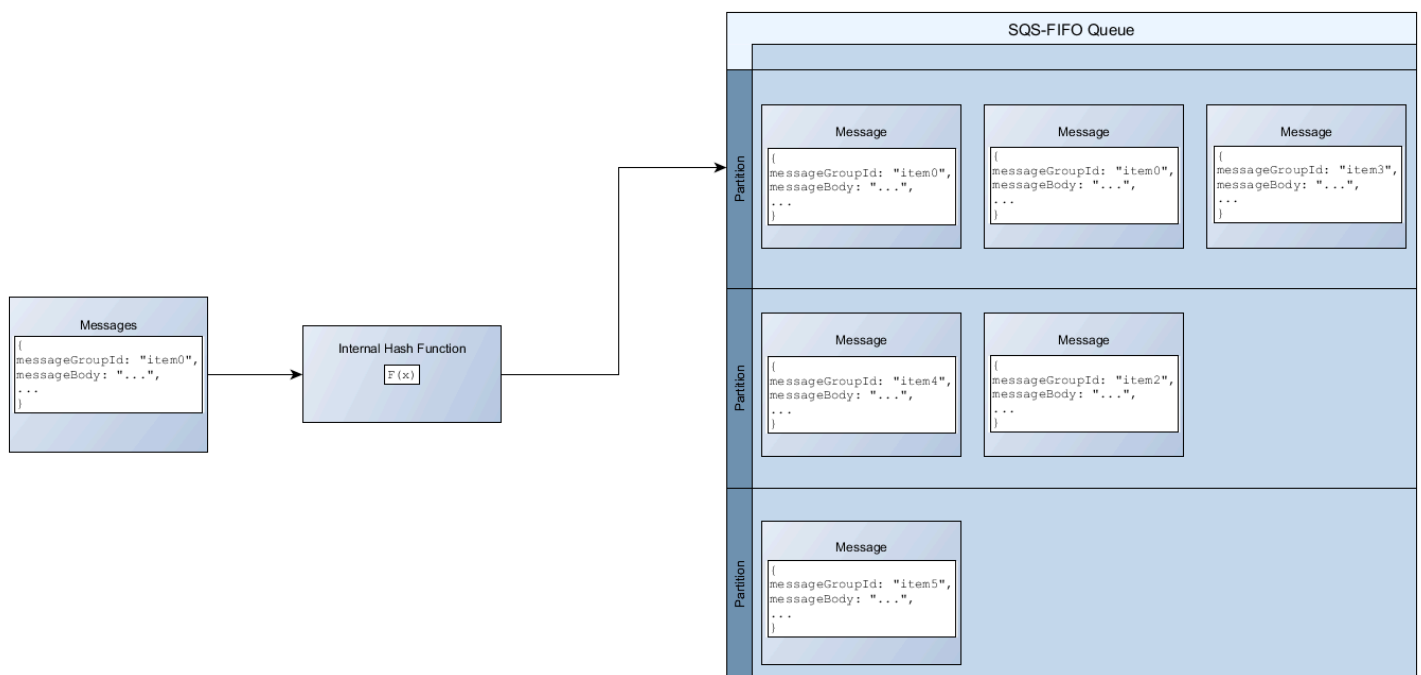
- Se a taxa de solicitação atual se aproximar ou exceder o que as partições existentes podem suportar, partições adicionais serão alocadas até que a fila atinja a cota regional. Para obter informações sobre cotas, consulte [Cotas de mensagens do Amazon SQS](#).
- Se as partições atuais tiverem baixa utilização, o número de partições poderá ser reduzido.

O gerenciamento de partições ocorre automaticamente em segundo plano e é transparente para as aplicações. Sua fila e mensagens estão disponíveis em todos os momentos.

Distribuindo dados por IDs de grupo de mensagens

Para adicionar uma mensagem a uma fila FIFO, o Amazon SQS usa o valor do ID do grupo de mensagens de cada mensagem como entrada para uma função de hash interna. O valor de saída da função de hash determina a partição na qual a mensagem será armazenada.

O diagrama a seguir mostra uma fila que abrange várias partições. O ID do grupo de mensagens da fila é baseado no número do item. O Amazon SQS usa sua função de hash para determinar onde armazenar um novo item, neste caso, com base no valor de hash da string `item0`. Observe que os itens são armazenados na mesma ordem em que são adicionados à fila. A localização de cada item é determinada pelo valor de hash de seu ID de grupo de mensagens.



 Note

O Amazon SQS é otimizado para distribuição uniforme de itens entre partições de uma fila FIFO, independentemente do número de partições. A AWS recomenda que você use IDs de grupo de mensagens que possam ter uma grande quantidade de valores distintos.

Otimizando a utilização de partições

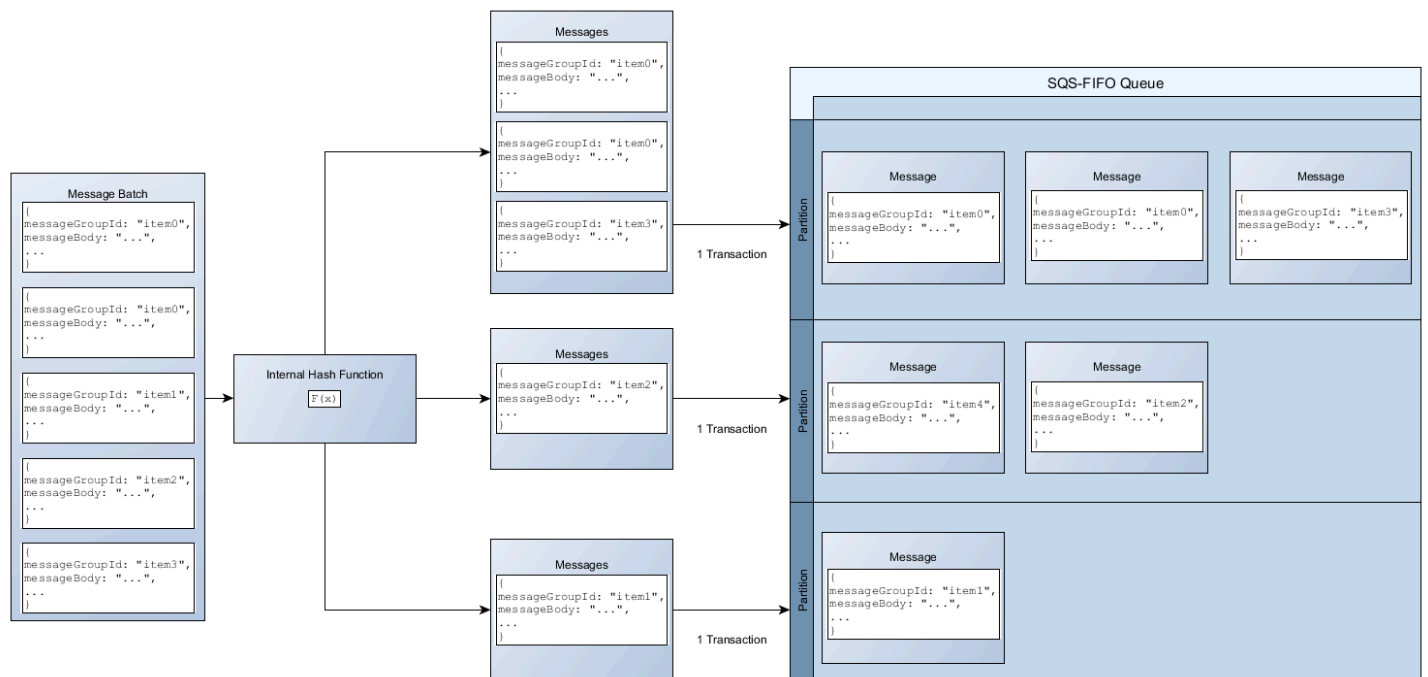
Cada partição suporta até 3.000 mensagens por segundo com processamento em lote ou até 300 mensagens por segundo para operações de envio, recebimento e exclusão em regiões compatíveis. Para ter mais informações sobre cotas de mensagens de throughput alto, consulte [Amazon SQS service quotas](#) no Referência geral da Amazon Web Services.

Ao usar APIs em lote, cada mensagem é encaminhada com base no processo descrito em [Distribuindo dados por IDs de grupo de mensagens](#). Mensagens encaminhadas para a mesma partição são agrupadas e processadas em uma única transação.

Para otimizar a utilização de partições para a API da `SendMessageBatch`, a AWS recomenda o processamento em lotes de mensagens com os mesmos IDs de grupo de mensagens, sempre que possível.

Para otimizar a utilização da partição para as APIs `DeleteMessageBatch` e `ChangeMessageVisibilityBatch`, a AWS recomenda o uso de solicitações `ReceiveMessage` com o parâmetro `MaxNumberOfMessages` definido como 10 e agrupando os identificadores de recebimento retornados por uma única solicitação `ReceiveMessage`.

No exemplo a seguir, um lote de mensagens com vários IDs de grupo de mensagens é enviado. O lote é dividido em três grupos, com cada um contando para a cota da partição.



Note

O Amazon SQS garante somente que mensagens com a mesma função de hash interna do ID de grupo de mensagens sejam agrupadas em uma solicitação em lote. Dependendo da saída da função hash interna e do número de partições, mensagens com diferentes IDs de grupo de mensagens podem ser agrupadas. Como a função hash ou o número de partições pode ser alterado a qualquer momento, as mensagens agrupadas em um ponto podem não ser agrupadas posteriormente.

Habilitar throughput alto para filas FIFO no Amazon SQS

Você pode habilitar a alta taxa de transferência para qualquer fila FIFO nova ou existente. O recurso inclui três novas opções ao criar e editar filas FIFO:

- **Enable high throughput FIFO (Habilitar FIFO de alta taxa de transferência):** aumenta a taxa de transferência disponível para mensagens na fila FIFO atual.
- **Deduplication scope (Escopo de eliminação de duplicação):** especifica se a eliminação de duplicação ocorre no nível da fila ou do grupo de mensagens.
- **FIFO throughput limit (Limite de taxa de transferência FIFO):** especifica se a cota de taxa de transferência em mensagens na fila FIFO está definida no nível da fila ou do grupo de mensagens.

Para habilitar a alta taxa de transferência para uma fila FIFO (console)

1. Inicie [criando](#) ou [editando](#) uma fila FIFO.
2. Ao especificar opções para a fila, escolha Enable high throughput FIFO (Habilitar FIFO de alta taxa de transferência).

Habilitar a alta taxa de transferência para filas FIFO define as opções relacionadas da seguinte maneira:

- Deduplication scope (Escopo de eliminação de duplicação) é definido como Message group (Grupo de mensagens), a configuração necessária para usar a alta taxa de transferência para filas FIFO.
- FIFO throughput limit (Limite de taxa de transferência FIFO) é definido como Per message group ID (Por ID do grupo de mensagens), a configuração necessária para usar a alta taxa de transferência para filas FIFO.

Se você alterar qualquer uma das configurações necessárias para usar a alta taxa de transferência para filas FIFO, a taxa de transferência normal estará em vigor para a fila e a eliminação de duplicação ocorrerá conforme especificado.

3. Continue especificando todas as opções para a fila. Ao concluir, escolha Create queue (Criar fila) ou Save (Salvar).

Depois de criar ou editar a fila FIFO, você pode [enviar mensagens](#) para ela e [receber e excluir mensagens](#), tudo em um TPS mais alto. Para cotas de alto throughput, consulte Throughput de mensagens em [Cotas de mensagens do Amazon SQS](#).

Identificadores de mensagens e filas FIFO no Amazon SQS

Esta seção descreve os identificadores de filas FIFO. Esses identificadores podem ajudar a localizar e manipular filas e mensagens específicas.

Identificadores de filas FIFO no Amazon SQS

Para obter mais informações sobre os seguintes identificadores, consulte a [Referência da API do Amazon Simple Storage Service](#).

Nome e URL da fila

Ao criar uma nova fila, você deve especificar o nome de uma fila exclusivo para sua conta e região da AWS. O Amazon SQS atribui a cada fila que você cria um identificador chamado URL da fila, que inclui o nome da fila e outros componentes do Amazon SQS. Sempre que você desejar executar uma ação em uma fila, forneça o URL da fila.

O nome de uma fila FIFO deve terminar com o sufixo `.fifo`. O sufixo conta para a cota de 80 caracteres do nome da fila. Para determinar se uma fila é [FIFO](#), você pode conferir se o nome da fila termina com o sufixo.

O URL a seguir é de uma fila FIFO chamada MyQueue, pertencente a um usuário com o número de conta da AWS 123456789012.

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue.fifo
```

É possível recuperar o URL de uma fila programaticamente listando as filas e analisando a string que segue o número da conta. Para obter mais informações, consulte [ListQueues](#).

ID de mensagem

Cada mensagem recebe um ID da mensagem atribuído pelo sistema que o Amazon SQS retorna para você na resposta [SendMessage](#). Esse identificador é útil para identificar mensagens. O tamanho máximo de um ID de mensagem é 100 caracteres.

Identificador de recebimento

Toda vez que você recebe uma mensagem de uma fila, recebe um identificador de recebimento dessa mensagem. Esse identificador é associado à ação de recebimento da mensagem, e não à mensagem. Para excluir a mensagem ou alterar a visibilidade da mensagem, você deve fornecer o identificador de recebimento (não o ID de mensagem). Desse modo, você sempre deve receber uma mensagem antes de excluí-la (você não pode colocar uma mensagem na fila e, em seguida, recuperá-la). O tamanho máximo de um identificador de recebimento é 1024 caracteres.

Important

Se você receber uma mensagem mais de uma vez, cada vez que recebê-la, obterá um identificador de recebimento diferente. Você deve fornecer o identificador de recebimento

recebido mais recentemente ao solicitar a exclusão da mensagem (caso contrário, a mensagem pode não ser excluída).

Veja a seguir um exemplo de um identificador de recebimento (quebrado em três linhas).

```
MbZj6wDWli+JvwwJaBV+3dcjk2YW2vA3+STFF1jTM8tJJg6HRG6PYSasuWXPJB+Cw  
Lj1FjgXUv1uSj1gUPAWV66FU/WeR4mq20KpEGYWbnLmpRCJVAyeMjeU5ZBdtcQ+QE  
auMZc8ZRv37sIW2iJKq3M9MFx1YvV11A2x/KSbkJ0=
```

Identificadores adicionais para filas FIFO do Amazon SQS

Para obter mais informações sobre os identificadores a seguir, consulte [Processamento exatamente uma vez no Amazon SQS](#) e a [Referência da API do Amazon Simple Queue Service](#).

ID de eliminação de duplicação de mensagem

Um token usado nas filas FIFO do Amazon SQS para identificar mensagens de forma exclusiva e evitar duplicações. Se várias mensagens com o mesmo ID de deduplicação forem enviadas dentro de um intervalo de deduplicação de cinco minutos, elas serão tratadas como duplicatas e somente uma cópia será entregue. Se você não especificar um ID de deduplicação e a deduplicação baseada em conteúdo estiver habilitada, o Amazon SQS gerará um ID de deduplicação por meio do hashing do corpo da mensagem. Esse mecanismo garante a entrega exatamente uma vez, eliminando mensagens duplicadas dentro do prazo especificado.

ID do grupo de mensagens

O `MessageGroupId` é um atributo usado apenas em filas FIFO (First-In, First-Out — Primeiro a entrar, primeiro a sair) do Amazon SQS para organizar mensagens em grupos distintos. As mensagens dentro do mesmo grupo são sempre processadas uma por vez, em ordem estrita, garantindo que duas mensagens do mesmo grupo não sejam processadas simultaneamente. As filas padrão não usam `MessageGroupId` e não oferecem garantias de ordenação. Se for necessária uma ordenação estrita, use uma fila FIFO.

Número de sequência

O número grande e não consecutivo que o Amazon SQS atribui a cada mensagem.

Cotas do Amazon SQS

Este tópico explica as cotas e limitações das filas FIFO e padrão do Amazon SQS, detalhando como elas afetam a criação, a configuração e o tratamento de mensagens de filas. Saiba mais sobre restrições como limites de retenção de mensagens, limites de mensagens em trânsito e limites de throughput, bem como estratégias para maximizar a eficiência por meio de lotes, otimização de chamadas de API e sondagens longas. Este tópico também aborda convenções de nomenclatura, regras de marcação e métodos para solicitar aumentos de cota para atender workloads de alta demanda, garantindo o gerenciamento eficaz e o desempenho ideal de filas.

Cotas de fila FIFO do Amazon SQS

Cotas do Amazon SQS

A tabela a seguir lista as cotas relacionadas a filas FIFO.

Quota	Descrição
Fila de atraso	O atraso padrão (mínimo) para uma fila é 0 segundo. O máximo é 15 minutos.
Filas listadas	1.000 filas por solicitação ListQueues .
Tempo de espera da sondagem longa	O tempo máximo de espera de sondagem longa é de 20 segundos.
Grupos de mensagens	Não há cota para o número de grupos de mensagens dentro de uma fila FIFO.
Mensagens por fila (backlog)	O número de mensagens que uma fila do Amazon SQS pode armazenar é ilimitado.
Mensagens por fila (em andamento)	As filas FIFO oferecem suporte para um máximo de 120.000 mensagens em trânsito (mensagens recebidas por um consumidor, mas ainda não excluídas). Se esse limite for atingido, o Amazon SQS não retornará um erro, mas o processamento poderá ser afetado. Você pode

Quota	Descrição
	solicitar um aumento desse limite entrando em contato com o AWS Support .
Nome da fila	O nome de uma fila FIFO deve terminar com o sufixo <code>.fifo</code> . O sufixo conta para a cota de 80 caracteres do nome da fila. Para determinar se uma fila é FIFO , você pode conferir se o nome da fila termina com o sufixo.
Tag de fila	Não recomendamos adicionar mais de 50 tags a uma fila. A marcação é compatível com caracteres Unicode em UTF-8.
	A tag Key é necessária, mas a tag Value é opcional.
	A tag Key e a tag Value diferenciam maiúsculas de minúsculas.
	A tag Key e a tag Value podem incluir caracteres alfanuméricos Unicode em UTF-8 e espaços em branco. Os seguintes caracteres especiais são permitidos: _ . : / = + - @
	A tag Key ou Value não pode incluir o prefixo reservado <code>aws:</code> (não é possível excluir chaves ou valores de tag com esse prefixo).
	O comprimento máximo da tag Key é de 128 caracteres Unicode em UTF-8. A tag Key não pode estar vazia nem ser nula.
	O comprimento máximo da tag Value é de 256 caracteres Unicode em UTF-8. A tag Value pode estar vazia ou ser nula.
	As ações de marcação são limitadas a 30 TPS por Conta da AWS. Se a sua aplicação exigir um throughput mais alto, envie uma solicitação .

Cotas de fila padrão do Amazon SQS

A tabela a seguir lista as cotas relacionadas a filas padrão.

Quota	Descrição
Fila de atraso	O atraso padrão (mínimo) para uma fila é 0 segundo. O máximo é 15 minutos.
Filas listadas	1.000 filas por solicitação ListQueues .
Tempo de espera da sondagem longa	O tempo máximo de espera de sondagem longa é de 20 segundos.
Mensagens por fila (backlog)	O número de mensagens que uma fila do Amazon SQS pode armazenar é ilimitado.
Mensagens por fila (em andamento)	Para a maioria das filas padrão (dependendo do tráfego da fila e da lista de pendências), pode haver um máximo de aproximadamente 120.000 mensagens em trânsito (recebidas de uma fila por um consumidor, mas ainda não excluídas da fila). Se você atingir essa cota ao usar a sondagem curta , o Amazon SQS retornará a mensagem de erro <code>OverLimit</code> . Se você usar a sondagem longa , o Amazon SQS não retornará nenhuma mensagem de erro. Para evitar atingir o quota, você deve excluir mensagens da fila depois de serem processadas. Você também pode aumentar o número de filas que usar para processar as mensagens. Para solicitar um aumento, envie um pedido de suporte .
Nome da fila	O nome da fila pode ter até 80 caracteres. Os seguintes caracteres são aceitos: caracteres alfanuméricos, hifens (-) e sublinhados (_).

Quota	Descrição
	 Note Os nomes de fila diferenciam maiúsculas e minúsculas (por exemplo, Test-queue e test-queue são filas diferentes).
Tag de fila	Não recomendamos adicionar mais de 50 tags a uma fila. A marcação é compatível com caracteres Unicode em UTF-8. A tag Key é necessária, mas a tag Value é opcional. A tag Key e a tag Value diferenciam maiúsculas de minúsculas. A tag Key e a tag Value podem incluir caracteres alfanuméricos Unicode em UTF-8 e espaços em branco. Os seguintes caracteres especiais são permitidos: _ . : / = + - @ A tag Key ou Value não pode incluir o prefixo reservado aws: (não é possível excluir chaves ou valores de tag com esse prefixo). O comprimento máximo da tag Key é de 128 caracteres Unicode em UTF-8. A tag Key não pode estar vazia nem ser nula. O comprimento máximo da tag Value é de 256 caracteres Unicode em UTF-8. A tag Value pode estar vazia ou ser nula. As ações de marcação são limitadas a 30 TPS por Conta da AWS. Se a sua aplicação exigir um throughput mais alto, envie uma solicitação.

Cotas de mensagens do Amazon SQS

A tabela a seguir lista as cotas relacionadas a mensagens.

Quota	Descrição
ID de mensagem em lote	O ID da mensagem em lote pode ter até 80 caracteres. Os seguintes caracteres são aceitos: caracteres alfanuméricos, hifens (-) e sublinhados (_).
Atributos de mensagens	Uma mensagem pode conter até 10 atributos de metadados.
Lote de mensagens	Uma única solicitação em lote de mensagens pode incluir um máximo de 10 mensagens. Para obter mais informações, consulte Configurar o AmazonSQSBufferedAsyncClient na seção Ações em lote do Amazon SQS .
Conteúdo da mensagem	Uma mensagem pode incluir apenas XML, JSON e texto não formatado. Os seguintes caracteres Unicode são permitidos: #x9 #xA #xD #x20 até #xD7FF #xE000 até #xFFFD #x10000 até #x10FFFF Os caracteres não incluídos nesta lista serão rejeitados. Para obter mais informações, consulte a Especificação W3C para caracteres.
ID do grupo de mensagens	MessageGroupId é necessário para filas FIFO. Se você não fornecer um MessageGroupId ao enviar uma mensagem para uma fila FIFO, a ação resultará em falha. Em filas padrão, o uso de MessageGroupId ativa filas justas. Recomendamos que você inclua um MessageGroupId em todas as mensagens ao usar filas justas. O tamanho máximo de MessageGroupId é 128 caracteres. Valores válidos: caracteres alfanuméricos e pontuação (!"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~) .

Quota	Descrição
Retenção da mensagem	Por padrão, uma mensagem será retida por 4 dias. A duração mínima é de 60 segundos (1 minuto). A configuração máxima é de 1.209.600 seconds (14 dias).
Taxa de transferência da mensagem	Filas padrão As filas padrão permitem um número muito alto e quase ilimitado de chamadas de API por segundo, por ação (SendMessage , ReceiveMessage ou DeleteMessage). Esse throughput alto as torna ideais para casos de uso que exigem o processamento rápido de grandes volumes de mensagens, como fluxo de dados em tempo real ou aplicações de grande escala. Embora as filas padrão sejam escaladas automaticamente de acordo com a demanda, é essencial monitorar os padrões de uso para garantir o desempenho ideal, especialmente em regiões com workloads mais altas.

Quota	Descrição
	<u>Filas FIFO</u> • Cada partição em uma fila FIFO é limitada a 300 transações por segundo, por ação de API (<code>SendMessage</code>, <code>ReceiveMessage</code>, e <code>DeleteMessage</code>). Esse limite se aplica especificamente ao modo de throughput não alto. Ao alternar para o modo de throughput alto, é possível ultrapassar esse limite padrão. Consulte como habilitar o modo de throughput alto em Habilitar throughput alto para filas FIFO no Amazon SQS. • Se você usa o agrupamento em lote, as filas FIFO de throughput não alto permitem até 3 mil mensagens por segundo, por ação de API (<code>SendMessage</code>, <code>ReceiveMessage</code> e <code>DeleteMessage</code>). As 3 mil mensagens por segundo representam 300 chamadas de API, cada uma com um lote de 10 mensagens. <u>Alta taxa de transferência para filas FIFO</u> Os limites de FIFO do Amazon SQS são baseados no número de solicitações de API, não nos limites de mensagens. Estes são os limites de solicitação de API no modo de throughput alto: Limites de throughput de transações (chamadas de API não agrupadas em lote) Esses limites definem com que frequência cada operação de API (como SendMessage, ReceiveMessage ou DeleteMessage) pode ser executada de forma independente, garantindo um desempenho eficiente do sistema dentro das transações permitidas por segundo (TPS). Os seguintes limites são baseados em chamadas de API não agrupadas em lote:

Quota	Descrição
	• Leste dos EUA (N. da Virgínia), Oeste dos EUA (Oregon) e Europa (Irlanda): até 70 mil transações por second (TPS). • Leste dos EUA (Ohio) e Europa (Frankfurt): até 19 mil TPS. • Ásia-Pacífico (Mumbai), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney) e Ásia-Pacífico (Tóquio): até 9 mil TPS. • Europa (Londres) e América do Sul (São Paulo): até 4.500 TPS. • Todas as outras Regiões da AWS: throughput padrão de 2.400 TPS. Maximizar o throughput com o agrupamento em lote Processa várias mensagens em uma única chamada de API, o que aumenta significativamente a eficiência. Em vez de lidar com cada mensagem individualmente, o processamento em lote permite que você envie, receba ou exclua até dez mensagens em uma única solicitação de API. Isso reduz o número total de chamadas de API, permitindo que você processe mais mensagens por segundo enquanto permanece dentro dos limites de transação (TPS) da região, maximizando o throughput e o desempenho do sistema. Para obter mais informações, consulte Aumento do throughput usando escalabilidade horizontal e processamento de ações em lote com o Amazon SQS. Os seguintes limites são baseados em chamadas de API agrupadas em lote: • Leste dos EUA (N. da Virgínia), Oeste dos EUA (Oregon) e Europa (Irlanda): até 700 mil mensagens por segundo (10 vezes o limite de 70 mil TPS).

Quota	Descrição
	• Leste dos EUA (Ohio) e Europa (Frankfurt): até 190 mil mensagens por segundo. • Ásia-Pacífico (Mumbai), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney) e Ásia-Pacífico (Tóquio): até 90 mil mensagens por segundo. • Europa (Londres) e América do Sul (São Paulo): até 45 mil mensagens por segundo. • Todas as outras Regiões da AWS: até 24 mil mensagens por segundo. Otimizar o throughput além do agrupamento em lote Embora o agrupamento em lote possa aumentar consideravelmente o throughput, é importante considerar outras estratégias para otimizar o desempenho de filas FIFO: • Distribuir mensagens em vários IDs de grupos de mensagens: como as mensagens em um único grupo são processadas sequencialmente, distribuir a workload em vários grupos de mensagens permite um melhor paralelismo e um throughput maior geral. Para obter mais informações, consulte Partições e distribuição de dados para alta taxa de transferência para filas FIFO do SQS. • Usar chamadas de API com eficiência: minimize chamadas de API desnecessárias, como alterações frequentes de visibilidade ou exclusões de mensagens repetidas, para otimizar o uso do TPS disponível e melhorar a eficiência. • Usar recebimentos de sondagem longa: utilize sondagem longa definindo <code>WaitTimeSeconds</code> nas solicitações de recebimento para reduzir respostas vazias quando nenhuma mensagem estiver disponível.

Quota	Descrição
	I, diminuindo chamadas de API desnecessárias e fazendo melhor uso de sua cota de TPS. • Solicitar aumento de throughput: se a aplicação exigir um throughput maior do que os limites padrão, solicite um aumento usando o console do Service Quotas. Isso pode ser necessário para workloads de alta demanda ou em regiões com limites padrão mais baixos. Consulte como habilitar o modo de throughput alto em Habilitar throughput alto para filas FIFO no Amazon SQS.
Temporizador de mensagem	O atraso padrão (mínimo) para uma mensagem é 0 segundo. O máximo é 15 minutos.
Tamanho da mensagem	O tamanho mínimo da mensagem é de 1 byte (1 caractere). O máximo é de 1.048.576 bytes (1 MiB). Para enviar mensagens maiores de 1 MiB, use a biblioteca a do cliente em versão ampliada para Java do Amazon SQS e a biblioteca do cliente em versão ampliada para Python do Amazon SQS. Essa biblioteca permite que você envie uma mensagem do Amazon SQS que contenha uma referência à carga útil de uma mensagem no Amazon S3. O tamanho máximo da carga útil é 2 GB.  Note Essa biblioteca em versão ampliada funciona somente para clientes síncronos.
Tempo limite de visibilidade da mensagem	O tempo limite de visibilidade padrão para uma mensagem é de 30 segundos. O mínimo é 0 segundo. O máximo é 12 horas.

Quota	Descrição
Informações de política	A cota máxima é 8.192 bytes, 20 declarações, 50 principais ou 10 condições. Para obter mais informações, consulte Cotas de política do Amazon SQS .

Cotas de política do Amazon SQS

A tabela a seguir lista as cotas relacionadas a políticas.

Name	Máximo
Bytes	8,192
Condições	10
Entidades principais	50
Declarações	20
Ações por declaração	7

Recursos e funcionalidades do Amazon SQS

Este tópico fornece recursos comumente usados no Amazon SQS para gerenciar filas de mensagens, otimizar o desempenho, garantir a entrega confiável de mensagens e lidar com o processamento de mensagens de forma eficiente.

Usar filas de mensagens não entregues no Amazon SQS

O Amazon SQS oferece suporte a filas de letras mortas (DLQs), que as filas de origem podem direcionar para mensagens que não foram processadas com sucesso. DLQs são úteis para depurar seu aplicativo porque você pode isolar mensagens não consumidas para determinar por que o processamento não foi bem-sucedido. Para um desempenho ideal, é uma prática recomendada manter a fila de origem e o DLQ dentro da mesma região e Conta da AWS. Quando as mensagens estão em uma fila de mensagens não entregues, você pode:

- Examinar logs para encontrar as exceções que podem ter causado a entrega de mensagens a uma fila de mensagens mortas.
- Analisar o conteúdo de mensagens movidas para uma fila de mensagens não entregues para diagnosticar problemas de aplicações.
- Determinar se você concedeu ao consumidor tempo suficiente para processar mensagens.
- Mova as mensagens para fora da fila de mensagens não entregues usando o [redirecionamento da fila de mensagens não entregues](#).

Primeiro, é necessário criar uma fila antes de configurá-la como uma fila de mensagens não entregues. Para obter informações sobre como configurar uma fila de mensagens não entregues usando o console do Amazon SQS, consulte [Como configurar uma fila de mensagens não entregues usando o console do Amazon SQS](#). Para obter ajuda com filas de mensagens não entregues, como configurar um alarme para todas as mensagens movidas para uma fila de mensagens não entregues, consulte [Como criar alarmes para filas de mensagens não entregues usando o Amazon CloudWatch](#).

Note

Não use uma fila de mensagens mortas com uma fila FIFO se você não quiser interromper a ordem exata das mensagens ou operações. Por exemplo, não use uma fila de mensagens

mortas com instruções em uma Edit Decision List (EDL) para um pacote de edição de vídeo, onde a alteração da ordem de edições muda o contexto de edições subsequentes.

Usar políticas para filas de mensagens não entregues

Use uma política de redirecionamento para especificar `maxReceiveCount`. O `maxReceiveCount` é o número de vezes que um consumidor pode receber uma mensagem de uma fila de origem antes de ser movida para uma fila de mensagens não entregues. Por exemplo, se `maxReceiveCount` estiver definido com um valor baixo, como 1, uma falha em receber uma mensagem faria com que a mensagem fosse movida para a fila de mensagens não entregues. Para garantir que seu sistema seja resiliente contra erros, defina o `maxReceiveCount` alto o suficiente para permitir novas tentativas suficientes.

Para filas padrão com uma política de redrive em que `maxReceiveCount` é maior que 3, se uma mensagem for recebida 3 ou mais vezes sem ser excluída, o SQS a moverá para o final da fila. A `ApproximateAgeOfOldestMessage` métrica então reflete a idade da próxima mensagem que não excedeu esse limite.

A política de permissão de redirecionamento especifica quais filas de origem podem acessar a fila de mensagens mortas. É possível escolher se deseja permitir todas as filas de origem, permitir filas de origem específicas ou negar que todas as filas de origem usem a fila de mensagens não entregues. O padrão permite que todas as filas de origem usem a fila de mensagens não entregues. Se você optar por permitir filas específicas usando a opção `byQueue`, você pode especificar até dez filas de origem usando o nome do recurso da Amazon (ARN) da fila de origem. Se você especificar `denyAll`, a fila não pode ser usada como uma fila de mensagens mortas.

Compreender os períodos de retenção de mensagens para filas de mensagens não entregues

Para filas comuns, a validade de uma mensagem é sempre baseada em seu carimbo de data/hora de enfileiramento original. Quando uma mensagem é movida para uma fila de mensagens mortas, o carimbo de data/hora de enfileiramento permanece inalterado. A métrica `ApproximateAgeOfOldestMessage` indica quando a mensagem foi movida para a fila de mensagens não entregues, não quando a mensagem foi originalmente enviada. Por exemplo, suponha que uma mensagem fique um dia na fila original antes de ser movida para uma fila de mensagens mortas. Se o período de retenção da fila de mensagens mortas for

de quatro dias, a mensagem será excluída da fila de mensagens mortas após três dias e a `ApproximateAgeOfOldestMessage` será de três dias. Portanto, é uma prática recomendada definir sempre o período de retenção de uma fila de mensagens mortas para ser maior do que o período de retenção da fila original.

Para filas FIFO, o carimbo de data/hora da fila é redefinido quando a mensagem é movida para uma fila de mensagens não entregues. A métrica `ApproximateAgeOfOldestMessage` indica quando a mensagem foi movida para a fila de mensagens não entregues. No mesmo exemplo acima, a mensagem é excluída da fila de mensagens não entregues após quatro dias e a `ApproximateAgeOfOldestMessage` é de quatro dias.

Como configurar uma fila de mensagens não entregues usando o console do Amazon SQS

Uma fila de mensagens não entregues (DLQ) é uma fila que recebe mensagens que não foram processadas com êxito de outra fila, conhecida como fila de origem. O Amazon SQS não cria fila de mensagens mortas automaticamente. Primeiro, é necessário criar uma fila antes de designar para ela uma fila de mensagens mortas. Ao configurar uma DLQ, o tipo de fila deve corresponder ao tipo de fila de origem. Uma [fila FIFO](#) só pode usar uma DLQ FIFO e uma [fila padrão](#) só pode usar uma DLQ padrão. Você pode configurar uma fila de mensagens não entregues ao criar ou editar uma fila. Consulte mais detalhes em [Usar filas de mensagens não entregues no Amazon SQS](#).

Para configurar uma fila de mensagens mortas para uma fila existente (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Selecione a fila de origem (a fila que enviará mensagens com falha para a fila de mensagens não entregues) e escolha Editar.
4. Role até a seção Fila de mensagens não entregues e escolha Habilitada.
5. Em Configurações da fila de mensagens não entregues, escolha o nome do recurso da Amazon (ARN) de uma fila existente que você deseja usar como fila de mensagens não entregues.
6. Defina o valor de Número máximo de recebimentos, que define quantas vezes uma mensagem pode ser recebida antes de ser enviada à fila de mensagens não entregues (intervalo válido: 1 a 1.000).
7. Escolha Salvar.

Saiba como configurar o redirecionamento de uma fila de mensagens não entregues no Amazon SQS

Use o redirecionamento da fila de mensagens não entregues para mover mensagens não consumidas de uma fila de mensagens não entregues para outro destino para processamento. Por padrão, o redirecionamento da fila de mensagens mortas move as mensagens de uma fila de mensagens mortas para uma fila de origem. No entanto, também será possível configurar qualquer outra fila como o destino de redirecionamento se as filas forem do mesmo tipo. Por exemplo, se a fila de mensagens não entregues for uma fila FIFO, a fila de destino de redirecionamento também deverá ser uma fila FIFO. Além disso, você pode configurar a velocidade de redirecionamento para definir a taxa na qual o Amazon SQS move mensagens.

Note

Quando uma mensagem é movida de uma fila FIFO para uma DLQ FIFO, o ID de eliminação de duplicação da mensagem original será substituído pelo ID da mensagem original. Isso acontece para garantir que a eliminação de duplicação da DLQ não impeça o armazenamento de duas mensagens independentes que, por acaso, compartilham o mesmo ID de eliminação de duplicação.

As filas de mensagens não entregues redirecionam as mensagens na ordem em que são recebidas, começando pela mais antiga. No entanto, a fila de destino ingere as mensagens redirecionadas, bem como as novas mensagens de outros produtores, de acordo com a ordem em que as recebe. Por exemplo, se um produtor enviar mensagens para uma fila FIFO de origem ao receber simultaneamente mensagens redirecionadas de uma fila de mensagens não entregues, as mensagens redirecionadas serão combinadas com as novas mensagens do produtor.

Note

A tarefa de redirecionamento redefine o período de retenção. Todas as mensagens redirecionadas são consideradas novas mensagens com um novo `messageID` e `enqueueTime` são atribuídos a mensagens redirecionadas.

Configurar o redirecionamento de uma fila de mensagens não entregues para uma fila padrão existente usando a API do Amazon SQS

É possível configurar o redirecionamento de uma fila de mensagens não entregues usando as ações de API `StartMessageMoveTask`, `ListMessageMoveTasks` e `CancelMessageMoveTask`:

Ação da API	Description
<code>StartMessageMoveTask</code>	Inicia uma tarefa assíncrona para mover mensagens de uma fila de origem especificada a uma fila de destino especificada.
<code>ListMessageMoveTasks</code>	Exibe as tarefas mais recentes de movimentação de mensagens (até dez) em uma fila de origem específica.
<code>CancelMessageMoveTask</code>	Cancela uma tarefa de movimentação de mensagens especificada. A movimentação de uma mensagem só pode ser cancelada quando o status atual é EM EXECUÇÃO.

Configurar o redirecionamento de uma fila de mensagens não entregues para uma fila padrão existente usando o console do Amazon SQS

1. Abra o console do Amazon SQS em. <https://console.aws.amazon.com/sqs/>
2. No painel de navegação, escolha Queues.
3. Escolha o nome da fila configurada como uma [fila de mensagens não entregues](#).
4. Selecione Start DLQ redrive (Iniciar o redirecionamento DLQ).
5. Em Redrive configuration (Configurações de redirecionamento), em Message destination (Destino da mensagem), execute uma das seguintes ações:
 - Para redirecionar mensagens para a fila de origem, escolha Redrive to source queue(s) (Redirecionar para fila(s) de origem).
 - Para redirecionar mensagens para outra fila, escolha Redrive to custom destination (Redirecionar para um destino personalizado). Em seguida, insira o nome do recurso da Amazon (ARN) de uma fila de destino existente.

6. Em Velocity control settings (Configurações de controle de velocidade), escolha uma das opções a seguir:
 - System optimized (Otimizado para o sistema) — Redirecione mensagens de fila de mensagens não entregues no número máximo de mensagens por segundo.
 - Custom max velocity (Velocidade máxima personalizada) — Redirecione mensagens de fila de mensagens não entregues com uma taxa máxima personalizada de mensagens por segundo. A taxa máxima permitida é de 500 mensagens por segundo.
 - É recomendável começar com um valor pequeno para a velocidade máxima personalizada e verificar se a fila de origem não está sobrecarregada com mensagens. A partir daí, aumente gradualmente o valor de velocidade máxima personalizada, continuando a monitorar o estado da fila de origem.
7. Quando você terminar de configurar o redirecionamento da fila de mensagens não entregues, escolha Save (Salvar).

 Important

O Amazon SQS não oferece suporte à filtragem e modificação de mensagens enquanto as redireciona da fila de mensagens não entregues.

Uma tarefa de redirecionamento de fila de mensagens não entregues pode ser executada no máximo 36 horas. O Amazon SQS oferece suporte a um máximo de 100 tarefas de redirecionamento ativo por conta.

8. Se você quiser cancelar a tarefa de redirecionamento de mensagens, na página Details (Detalhes) da sua fila, escolha Cancel DLQ redrive (Cancelar redirecionamento DLQ). Ao cancelar uma redirecionamento de mensagem em andamento, todas as mensagens que já tenham sido movidas com sucesso para a fila de destino de movimentação permanecerão na fila de destino.

Configurar permissões de fila para o redirecionamento da fila de mensagens não entregues

Você pode conceder ao usuário acesso a ações específicas da fila de mensagens não entregues adicionando permissões à política. As permissões mínimas necessárias para uma fila de mensagens não entregues são as seguintes:

Permissões mínimas	Métodos de API necessários
Como iniciar um redirecionamento de mensagens	- Adicione <code>sqs:StartMessageMoveTask</code> , <code>sqs:ReceiveMessage</code> , <code>sqs:DeleteMessage</code> e <code>sqs:GetQueueAttributes</code> da fila de mensagens não entregues. Se a fila de mensagens não entregues ou a fila de origem forem criptografadas (também conhecida como fila SSE), também será necessário <code>kms:Decrypt</code> para qualquer chave do KMS usada para criptografar as mensagens. - Adicione o <code>sqs:SendMessage</code> da fila de destino. Se a fila de destino estiver criptografada, também será necessário adicionar <code>kms:GenerateDataKey</code> e <code>kms:Decrypt</code> .
Como cancelar um redirecionamento de mensagem em andamento	Adicione <code>sqs:CancelMessageMoveTask</code> , <code>sqs:ReceiveMessage</code> , <code>sqs:DeleteMessage</code> e <code>sqs:GetQueueAttributes</code> da fila de mensagens não entregues. Se a fila de mensagens não entregues estiver criptografada (também conhecida como fila SSE), também será necessário o adicionar <code>kms:Decrypt</code> .
Como exibir o status de movimentação de uma mensagem	Adicione <code>sqs:ListMessageMoveTasks</code> e <code>sqs:GetQueueAttributes</code> da fila de mensagens não entregues.

Para configurar permissões a um par de filas criptografadas (uma fila de origem com uma fila de mensagens não entregues)

Siga as etapas abaixo para configurar permissões mínimas de redirecionamento de fila de mensagens não entregues (DLQ):

1. Abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel de navegação, selecione Políticas.
3. Crie uma [política](#) e anexe as permissões a seguir. Anexe a política ao [usuário](#) ou [perfil](#) do IAM que executará a operação de recondução.

- Permissões para a DLQ (fila de origem):
 - `sqs:StartMessageMoveTask`
 - `sqs:CancelMessageMoveTask`
 - `sqs:ListMessageMoveTasks`
 - `sqs:ReceiveMessage`
 - `sqs>DeleteMessage`
 - `sqs:GetQueueAttributes`
 - `sqs:ListDeadLetterSourceQueues`
 - Especifique o ARN do recurso da DLQ (fila de origem) (por exemplo, "arn:aws:sqs::").
`<DLQ_region> <DLQ_accountId> <DLQ_name>`
- Permissões para a fila de destino:
 - `sqs:SendMessage`
 - Especifique a fila Resource ARN de destino (por exemplo, "arn:aws:sqs:").
`<DestQueue_region>:<DestQueue_accountId>:<DestQueue_name>`
- Permissões para a chave do KMS:
 - `kms:Decrypt` (Necessário para descriptografar mensagens na DLQ.)
 - `kms:GenerateDataKey` (Necessário para criptografar mensagens na fila de destino.)
 - Resource ARNs:
 - O ARN da chave KMS usada para criptografar mensagens na DLQ (fila de origem) (por exemplo, "arn:aws:kms: ::key/ "). `<region> <accountId> <SourceQueueKeyId>`
 - O ARN da chave KMS usada para criptografar mensagens na fila de destino (por exemplo, "arn:aws:kms: ::key/ "). `<region> <accountId> <DestinationQueueKeyId>`

A política de acesso deve ser semelhante a:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

        "Effect": "Allow",
        "Action": [
            "sqs:StartMessageMoveTask",
            "sqs:CancelMessageMoveTask",
            "sqs:ListMessageMoveTasks",
            "sqs:ReceiveMessage",
            "sqs>DeleteMessage",
            "sqs:GetQueueAttributes",
            "sqs:ListDeadLetterSourceQueues"
        ],
        "Resource": "arn:aws:sqs:us-west-1:123456789012:<DLQ_name>",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/QueueRole": "source"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "sqs:SendMessage",
        "Resource": "arn:aws:sqs:us-west-1:123456789012:<DestQueue_name>",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/QueueRole": "destination"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey"
        ],
        "Resource": [
            "arn:aws:kms:us-west-1:123456789012:key/<SourceQueueKeyId>",
            "arn:aws:kms:us-west-1:123456789012:key/<DestQueueKeyId>"
        ]
    }
]
}

```


Para configurar permissões a um par de filas não criptografadas (uma fila de origem com uma fila de mensagens não entregues)

Siga as etapas abaixo para configurar as permissões mínimas necessárias para gerenciar uma fila de mensagens não entregues (DLQ) padrão não criptografada. As permissões mínimas necessárias são para receber, excluir e obter atributos da fila de mensagens não entregues e enviar atributos para a fila de origem.

1. Abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel de navegação, selecione Políticas.
3. Crie uma [política](#) e anexe as permissões a seguir. Anexe a política ao [usuário](#) ou [perfil](#) do IAM que executará a operação de recondução.

- Permissões para a DLQ (fila de origem):
 - `sqs:StartMessageMoveTask`
 - `sqs:CancelMessageMoveTask`
 - `sqs:ListMessageMoveTasks`
 - `sqs:ReceiveMessage`
 - `sqs>DeleteMessage`
 - `sqs:ListDeadLetterSourceQueues`
 - Especifique o ARN do recurso da DLQ (fila de origem) (por exemplo, "arn:aws:sqs::: ").
`<DLQ_region> <DLQ_accountId> <DLQ_name>`
- Permissões para a fila de destino:
 - `sqs:SendMessage`
 - Especifique a fila Resource ARN de destino (por exemplo, "arn:aws:sqs: ").
`<DestQueue_region>:<DestQueue_accountId>:<DestQueue_name>`

A política de acesso deve ser semelhante a:

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {
```

```

        "Effect": "Allow",
        "Action": [
            "sqs:StartMessageMoveTask",
            "sqs:CancelMessageMoveTask",
            "sqs:ListMessageMoveTasks",
            "sqs:ReceiveMessage",
            "sqs>DeleteMessage",
            "sqs:GetQueueAttributes",
            "sqs:ListDeadLetterSourceQueues"
        ],
        "Resource": "arn:aws:sqs:us-west-1:111122223333:<DLQ_name>",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/QueueRole": "source"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "sqs:SendMessage",
        "Resource": "arn:aws:sqs:us-west-1:111122223333:<DestQueue_name>",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/QueueRole": "destination"
            }
        }
    }
]
}

```

Como usar o redirecionamento da fila de mensagens não entregues com controle de acesso ao endpoint de VPC

Ao restringir o acesso à fila a um VPCs uso específico da `aws:sourceVpc` condição, você precisa abrir uma exceção para os AWS serviços para habilitar a funcionalidade de recondução da fila de cartas mortas (DLQ). Isso ocorre porque o serviço Amazon SQS opera fora da sua VPC ao mover mensagens.

Para permitir operações de redirecionamento da DLQ, adicione a condição `aws:CalledViaLast` à sua política de filas. Isso permite que o Amazon SQS faça chamadas de API em seu nome, mantendo restrições de VPC para acesso direto.

Para permitir o acesso restrito à VPC e o redirecionamento de DLQ:

1. Use a condição `aws:CalledViaLast` em sua política de filas.
2. Aplique a política à fila de origem e à DLQ
3. Mantenha as restrições da VPC para acesso direto de outras fontes

Veja um exemplo de política que implementa esses requisitos:

JSON

```
{
  "Version": "2012-10-17",
  "Id": "SQSRedriveWithVpcRestriction",
  "Statement": [
    {
      "Sid": "DenyOutsideVPCUnlessAWSService_DestQueue",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:*:111122223333:DestQueue",
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpc": "vpc-1234567890abcdef0"
        },
        "StringNotEqualsIfExists": {
          "aws:CalledViaLast": "sqs.amazonaws.com"
        }
      }
    },
    {
      "Sid": "DenyOutsideVPCUnlessAWSService_DLQ",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:*:111122223333:Dlq",
      "Condition": {
        "StringNotEquals": {
```

```

        "aws:SourceVpc": "vpc-1234567890abcdef0"
    },
    "StringNotEqualsIfExists": {
        "aws:CalledViaLast": "sqs.amazonaws.com"
    }
}
]
}

```

- Substitua os valores dos espaços reservados por seus valores reais:
- Esta política usa uma instrução “deny” com condições, o que é mais seguro do que usar instruções “allow”
- O operador `StringNotEqualsIfExists` gerencia casos em que a chave de condição pode não estar presente no contexto da solicitação.

Como alternativa, você pode usar a chave de condição `aws:ViaAWSService` para permitir o acesso baseado em serviços e, ao mesmo tempo, manter as restrições da VPC. Essa chave de condição indica se a solicitação vem de um AWS serviço. Aqui está um exemplo de política que usa `aws:ViaAWSService` em vez de `aws:CalledViaLast`:

JSON

```

{
  "Version": "2012-10-17",
  "Id": "SQSRedriveWithVpcRestriction",
  "Statement": [
    {
      "Sid": "DenyOutsideVPCUnlessAWSService_DestQueue",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:*:111122223333:DestQueue",
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpc": "vpc-1234567890abcdef0"
        },
        "BoolIfExists": {

```

```
        "aws:ViaAWSService": "false"
      }
    },
    {
      "Sid": "DenyOutsideVPCUnlessAWSService_DLQ",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs*:111122223333:DLq",
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpc": "vpc-1234567890abcdef0"
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
        }
      }
    }
  ]
}
```

O BoolIfExists operador com `aws:ViaAWSService` condição garante que as solicitações sejam permitidas quando elas vêm de serviços, mantendo as restrições de VPC para acesso direto. Isso pode ser mais simples de entender e manter, pois verifica diretamente se a solicitação foi feita por um AWS serviço, em vez de verificar qual serviço fez a última chamada.

Para obter mais informações sobre chaves de condição usadas no IAM e nas políticas de recursos, consulte Elementos de política JSON do IAM: condição.

CloudTrail requisitos de atualização e permissão para o redrive de fila de cartas mortas do Amazon SQS

Em 8 de junho de 2023, o Amazon SQS introduziu o redrive de fila de letras mortas (DLQ) para AWS SDK e (CLI).AWS Command Line Interface Esse recurso é uma adição ao redrive DLQ já suportado para o AWS console. Se você já usou o AWS console para redirecionar mensagens da fila de mensagens mortas, você pode ser afetado pelas seguintes alterações:

CloudTrail renomeação de eventos

Em 15 de outubro de 2023, os nomes dos CloudTrail eventos para recondução de filas de cartas mortas serão alterados no console do Amazon SQS. Se você definiu alarmes para esses CloudTrail eventos, você deve atualizá-los agora. A seguir estão os novos nomes de CloudTrail eventos para o DLQ redrive:

Antigo nome do evento	Novo nome do evento
CreateMoveTask	StartMessageMoveTask
CancelMoveTask	CancelMessageMoveTask

Permissões atualizadas

Incluído na versão do SDK e da CLI, o Amazon SQS também atualizou as permissões de fila para o redirecionamento da DLQ a fim de aderir às práticas recomendadas de segurança. Use os seguintes tipos de permissão de fila para redirecionar mensagens do seu DLQs

1. Permissões baseadas em ações (atualização para as ações da API da DLQ)
2. Permissões de políticas gerenciadas do Amazon SQS
3. Política de permissão que usa o curinga sqs:*

Important

Para usar o redirecionamento de DLQ para o SDK ou a CLI, é necessário ter uma política de permissão de redirecionamento de DLQ que corresponda a uma das opções acima.

Se as permissões de fila para o redirecionamento da DLQ não corresponderem a uma das opções acima, você deverá atualizá-las até 31 de agosto de 2023. Até 31 de agosto de 2023, sua conta poderá redirecionar mensagens usando as permissões que você configurou por meio do console da AWS apenas nas regiões em que você já usou o redirecionamento de DLQ. Por exemplo, digamos que você tenha a “Conta A” em us-east-1 e eu-west-1. A “Conta A” foi usada para redirecionar mensagens no AWS console em us-east-1 antes de 8 de junho de 2023, mas não em eu-west-1. Entre 8 de junho de 2023 e 31 de agosto de 2023, se as permissões da política da “Conta

A” não corresponderem a uma das opções acima, elas só poderão ser usadas para redirecionar mensagens no AWS console em us-east-1, e não em eu-west-1.

Important

Se as permissões de redirecionamento da DLQ não corresponderem a uma dessas opções após 31 de agosto de 2023, sua conta não poderá mais redirecionar mensagens da DLQ usando o console da AWS.

No entanto, se você usou o recurso de redrive DLQ no AWS console em agosto de 2023, terá uma extensão até 15 de outubro de 2023 para adotar as novas permissões de acordo com uma dessas opções.

Para obter mais informações, consulte [the section called “Identificação de políticas afetadas”](#).

Veja a seguir exemplos de permissões de fila para cada opção de redirecionamento de DLQ. Ao usar [filas criptografadas do lado do servidor \(SSE\)](#), a permissão de AWS KMS chave correspondente é necessária.

Baseado em ação

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:StartMessageMoveTask",
        "sqs:ListMessageMoveTasks",
        "sqs:CancelMessageMoveTask"
      ],
      "Resource": "arn:aws:sqs:us-west-1:123456789012:<DLQ_name>"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
```

```
    "Resource": "arn:aws:sqs:us-west-1:123456789012:<DestQueue_name>"
  }
}
]
```

Política gerenciada

As seguintes políticas gerenciadas contêm as permissões atualizadas exigidas:

- Amazon SQSFull Access — Inclui as seguintes tarefas de recondução de filas sem saída: iniciar, cancelar e listar.
- Amazon SQSRead OnlyAccess — Fornece acesso somente para leitura e inclui a tarefa de recondução da fila de cartas mortas da lista.

Step 1

Add permissions

Step 2

Review

Add permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

☐ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions
Copy all group memberships, attached managed policies, inline policies, and any existing permissions boundaries from an existing user.

☒ Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1051)

2 matches < 1 >

	Policy name	Type	Attached entities
☒	AmazonSQSFullAccess	AWS managed	0
☐	AmazonSQSReadOnly...	AWS managed	0

Cancel

Next

Política de permissão que usa o curinga sqs*

CloudTrail requisitos de atualização e permissão

78

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sqs:*",
      "Resource": "*"
    }
  ]
}
```

Identificação de políticas afetadas

Se você estiver usando políticas gerenciadas pelo cliente (CMPs), poderá usar AWS CloudTrail um IAM para identificar as políticas afetadas pela atualização das permissões da fila.

Note

Se você estiver usando `AmazonSQSFullAccess` e `AmazonSQSReadOnlyAccess`, nenhuma ação será necessária.

1. Faça login no AWS CloudTrail console.
2. Na página Histórico de eventos, em Pesquisar atributos, use o menu suspenso para selecionar Nome do evento. Depois, pesquise `CreateMoveTask`.
3. Escolha um evento para abrir a página Detalhes. Na seção Registros de eventos, recupere o `UserName` ou `RoleName` do ARN de `userIdentity`.
4. Faça login no console do IAM.
 - Para usuários, escolha “Usuários”. Selecione o usuário com o `UserName` identificado na etapa anterior.
 - Para perfis, escolha “Perfis”. Pesquise o usuário com o `RoleName` identificado na etapa anterior.
5. No página Detalhes, na seção Permissões, revise todas as políticas com o prefixo `sqs:` em `Action`, ou revise as políticas que tenham a fila do Amazon SQS definida em `Resource`.

Como criar alarmes para filas de mensagens não entregues usando o Amazon CloudWatch

Configure um alarme do CloudWatch para monitorar mensagens em uma fila de mensagens não entregues usando a métrica [ApproximateNumberOfMessagesVisible](#). Para ter instruções detalhadas, consulte [Criação de alarmes do CloudWatch para métricas do Amazon SQS](#). Quando o alarme é acionado, indicando que as mensagens foram movidas para a fila de mensagens não entregues, você pode [pesquisar](#) na fila para revisar e recuperar as mensagens não entregues.

Metadados de mensagens para o Amazon SQS

Use atributos de mensagem para adicionar metadados personalizados a mensagens do Amazon SQS para suas aplicações. Use atributos do sistema de mensagens para armazenar metadados para outros Serviços da AWS, como o AWS X-Ray.

Atributos de mensagem do Amazon SQS

O Amazon SQS permite a inclusão de metadados estruturados (como carimbos de data e hora, dados geoespaciais, assinaturas e identificadores) com mensagens usando os atributos de mensagem. Cada mensagem pode ter até dez atributos. Os atributos de mensagem são opcionais e separados do corpo da mensagem (no entanto, são enviados junto com o corpo da mensagem). O consumidor pode usar atributos de mensagem para tratar uma mensagem de uma forma específica sem precisar primeiro processar o corpo da mensagem. Para obter informações sobre como enviar mensagens com atributos usando o console do Amazon SQS, consulte [Enviar uma mensagem com atributos usando o Amazon SQS](#).

Note

Não confunda atributos de mensagens com atributos de sistema de mensagens: enquanto é possível usar atributos de mensagens para associar metadados personalizados a mensagens do Amazon SQS para suas aplicações, é possível usar os [atributos de sistema de mensagens](#) a fim de armazenar metadados para outros produtos da AWS, como o AWS X-Ray.

Tópicos

- [Componentes de atributos de mensagem](#)

- [Tipos de dados de atributos de mensagem](#)
- [Cálculo do resumo de mensagens MD5 para atributos de mensagem](#)

Componentes de atributos de mensagem

Important

Todos os componentes de um atributo de mensagem estão incluídos na restrição de tamanho de 1 MiB da mensagem.

O Name, Type, Value e o corpo da mensagem não devem estar vazios ou serem nulos.

Cada atributo de mensagem consiste nos seguintes componentes:

- Nome: o nome do atributo da mensagem pode conter os seguintes caracteres: A-Z, a-z, 0-9, sublinhado (`_`), hífen (`-`), e ponto (`.`). As seguintes restrições são aplicáveis:
 - Pode ter até 256 caracteres
 - Não pode começar com `AWS .` ou `Amazon .` (ou qualquer variação no uso de maiúsculas e minúsculas)
 - Diferencia maiúsculas de minúsculas
 - Deve ser exclusivo entre todos os nomes de atributos da mensagem
 - Não deve começar ou terminar com um ponto
 - Não deve ter pontos em uma sequência
- Tipo: o tipo de dados do atributo da mensagem. Os tipos compatíveis incluem `String`, `Number` e `Binary`. Você também pode adicionar informações personalizadas para qualquer tipo de dados. O tipo de dados tem as mesmas restrições que o corpo da mensagem (para obter mais informações, consulte [SendMessage](#) na Referência da API do Amazon Simple Queue Service). Além disso, aplicam-se as seguintes restrições:
 - Pode ter até 256 caracteres
 - Diferencia maiúsculas de minúsculas
- Valor: o valor do atributo da mensagem. Para tipos de dados `String`, os valores dos atributos têm as mesmas restrições que o corpo da mensagem.

Tipos de dados de atributos de mensagem

Os tipos de dados de atributos de mensagens indicam ao Amazon SQS como tratar valores de atributos de mensagens correspondentes. Por exemplo, se o tipo for `Number`, o Amazon SQS validará valores numéricos.

O Amazon SQS é compatível com os tipos de dados lógicos `String`, `Number` e `Binary` com rótulos de tipos de dados personalizados opcionais com o formato *`.custom-data-type`*

- `String`: os atributos `String` podem armazenar texto Unicode usando quaisquer caracteres XML válidos.
- `Número`: os atributos `Number` podem armazenar valores numéricos positivos ou negativos. Um número pode ter até 38 dígitos de precisão, e pode ser entre 10^{-128} e 10^{+126} .

Note

O Amazon SQS remove zeros iniciais e finais.

- `Binário`: os atributos binários podem armazenar qualquer dado binário, como dados compactados, dados criptografados ou imagens.
- `Personalizado`: para criar um tipo de dado personalizado, acrescente um rótulo de tipo personalizado a qualquer tipo de dado. Por exemplo:
 - `Number.byte`, `Number.short`, `Number.int` e `Number.float` podem ajudar a diferenciar entre tipos numéricos.
 - `Binary.gif` e `Binary.png` podem ajudar a diferenciar entre tipos de arquivos.

Note

O Amazon SQS não interpreta, valida ou usa os dados anexados.

O rótulo de tipo personalizado tem as mesmas restrições que o corpo da mensagem.

Cálculo do resumo de mensagens MD5 para atributos de mensagem

Ao usar o AWS SDK para Java, você pode ignorar esta seção. A classe `MessageMD5ChecksumHandler` do SDK for Java oferece suporte a resumos de mensagens MD5 para atributos de mensagens do Amazon SQS.

Ao usar a API de consulta ou um dos SDKs da AWS que não ofereçam suporte ao resumo de mensagens MD5 para atributos de mensagens do Amazon SQS, use as seguintes diretrizes para realizar o cálculo do resumo de mensagens MD5.

Note

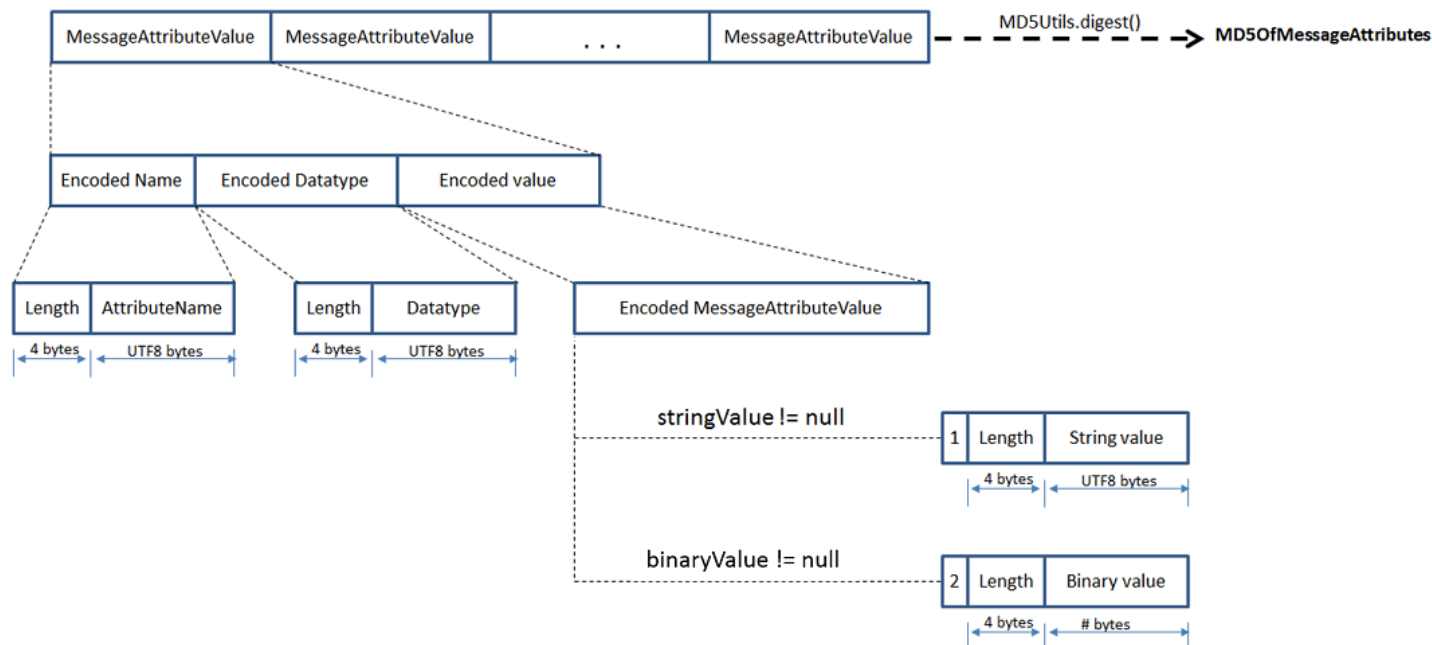
Sempre inclua sufixos de tipos de dados personalizados no cálculo do resumo de mensagens MD5.

Visão geral

O seguinte é uma visão geral do algoritmo de cálculo do resumo de mensagens MD5:

1. Classificar todos os atributos de mensagem por nome em ordem crescente.
2. Codificar as partes individuais de cada atributo (Name, Type e Value) em um buffer.
3. Calcular o resumo de mensagem de todo o buffer.

O seguinte diagrama mostra a codificação do resumo de mensagens MD5 para um único atributo de mensagem:



Para codificar um único atributo de mensagem do Amazon SQS

1. Codifique o nome: o comprimento (4 bytes) e os bytes UTF-8 do nome.

2. Codifique o tipo de dados: o comprimento (4 bytes) e os bytes UTF-8 do tipo de dados.
3. Codifique o tipo de transporte (String ou Binary) do valor (1 byte).

 Note

Os tipos de dados lógicos `String` e `Number` usam o tipo de transporte `String`.
Os tipos de dados lógicos `Binary` usam o tipo de transporte `Binary`.

- a. Para o tipo de transporte `String`, codifique 1.
 - b. Para o tipo de transporte `Binary`, codifique 2.
4. Codifique o valor do atributo.
 - a. Para o tipo de transporte `String`, codifique o valor do atributo: o comprimento (4 bytes) e os bytes UTF-8 do valor.
 - b. Para o tipo de transporte `Binary`, codifique o valor do atributo: o comprimento (4 bytes) e os bytes brutos do valor.

Atributos do sistema de mensagens do Amazon SQS

Enquanto é possível usar [atributos de mensagens](#) para anexar metadados personalizados a mensagens do Amazon SQS para suas aplicações, é possível usar atributos do sistema de mensagens a fim de armazenar metadados para outros produtos da AWS, como o AWS X-Ray. Para obter mais informações, consulte o parâmetro de solicitação `MessageSystemAttribute` das ações de API de [SendMessage](#) e [SendMessageBatch](#), o atributo `AWSTraceHeader` da ação de API [ReceiveMessage](#) e o tipo de dado [MessageSystemAttributeValue](#) na Referência da API do Amazon Simple Queue Service.

Os atributos do sistema de mensagens são estruturados exatamente como atributos de mensagens, com as seguintes exceções:

- Atualmente, o único atributo do sistema de mensagens compatível é `AWSTraceHeader`. Seu tipo deve ser `String`, e seu valor deve ser uma string de cabeçalho de rastreamento do AWS X-Ray formatada corretamente.
- O tamanho de um atributo do sistema de mensagens não entra na contagem para o tamanho total de uma mensagem.

Recursos necessários para processar mensagens do Amazon SQS

O Amazon SQS fornece estimativas do número aproximado de mensagens em atraso, visíveis e não visíveis em uma fila para ajudar a avaliar os recursos necessários para o processamento. Para obter mais informações sobre visibilidade, consulte [Tempo limite de visibilidade do Amazon SQS](#).

Note

Para algumas métricas, o resultado é aproximado por causa da arquitetura distribuída do Amazon SQS. Na maioria dos casos, a contagem deve ser próxima da quantidade real de mensagens na fila.

A tabela a seguir lista o nome do atributo a ser usado com a ação [GetQueueAttributes](#):

Tarefa	Nome do atributo
Obter o número de mensagens disponíveis para recuperação na fila.	<code>ApproximateNumberOfMessagesVisible</code>
Obter o número de mensagens na fila que estão atrasadas e indisponíveis para leitura imediata. Isso pode acontecer quando a fila tem a configuração de fila com atraso ou quando uma mensagem foi enviada com um parâmetro de atraso.	<code>ApproximateNumberOfMessagesDelayed</code>
Obter o número de mensagens que estão em processamento. As mensagens são consideradas como em processamento quando foram enviadas a um cliente, mas ainda não foram excluídas ou ainda não atingiram o final de sua janela de visibilidade.	<code>ApproximateNumberOfMessagesNotVisible</code>

Paginação da fila de listas do Amazon SQS

Os métodos de API [listQueues](#) e [listDeadLetterQueues](#) oferecem suporte a controles opcionais de paginação. Por padrão, esses métodos de API retornam até 1.000 filas na mensagem de resposta. É possível definir o parâmetro `MaxResults` para retornar menos resultados em cada resposta.

Defina o parâmetro `MaxResults` na solicitação `listQueues` ou `listDeadLetterQueues` para especificar o número máximo de resultados a serem retornados na resposta. Se você não definir o `MaxResults`, a resposta incluirá um máximo de 1.000 resultados e o valor de `NextToken` na resposta será nulo.

Se você definir `MaxResults`, a resposta incluirá um valor para `NextToken`, se houver resultados adicionais a serem exibidos. Use `NextToken` como parâmetro na próxima solicitação para `listQueues` a fim de receber a próxima página de resultados. Se não houver resultados adicionais a serem exibidos, o valor de `NextToken` na resposta será nulo.

Tags de alocação de custos do Amazon SQS

Para organizar e identificar as filas do Amazon SQS para alocação de custos, você pode adicionar tags de metadados que identificam o proprietário, o ambiente ou a finalidade de uma fila. Isso é especialmente útil quando você tem várias filas. Para configurar tags usando o console do Amazon SQS, consulte [the section called “Configurar tags para uma fila”](#)

Você pode usar etiquetas de alocação de custos para organizar sua AWS fatura de forma a refletir sua própria estrutura de custos. Para fazer isso, inscreva-se para que sua Conta da AWS fatura inclua chaves e valores de etiquetas. Para obter mais informações, consulte [Configuração de um relatório de alocação de custos mensal](#) no Guia do usuário do AWS Billing .

Toda tag é composta de um par de valores de chave, que são definidos por você. Por exemplo, você pode identificar facilmente suas filas de produção e teste se você marcar suas filas da seguinte forma:

Fila	Chave	Valor
MyQueueA	QueueType	Production
MyQueueB	QueueType	Testing

 Note

Ao usar tags de fila, lembre-se das orientações a seguir:

- Não recomendamos adicionar mais de 50 tags a uma fila. A marcação é compatível com caracteres Unicode em UTF-8.
- As tags não têm significado semântico. O Amazon SQS interpreta as tags como strings.
- As tags diferenciam letras maiúsculas de minúsculas.
- Uma nova tag com uma chave idêntica à de uma tag existente substitui a tag existente.
- As ações de marcação são limitadas a 30 TPS por. Conta da AWS Se a sua aplicação exigir um throughput mais alto, [envie uma solicitação](#).

Para ver uma lista completa de restrições de etiquetas, consulte [Cotas de fila padrão do Amazon SQS](#).

Sondagem curta e longa do Amazon SQS

O Amazon SQS oferece opções de sondagem curta e sondagem longa para receber mensagens de uma fila. Considere os requisitos de capacidade de resposta e eficiência de custos da aplicação ao escolher entre estas duas opções de sondagem:

- Sondagem curta (padrão): a solicitação [ReceiveMessage](#) consulta um subconjunto de servidores (com base em uma distribuição aleatória ponderada) para encontrar as mensagens disponíveis e envia uma resposta imediata, mesmo que nenhuma mensagem seja encontrada.
- Sondagem longa: [ReceiveMessage](#) consulta mensagens em todos os servidores, enviando uma resposta quando pelo menos uma mensagem estiver disponível, até o máximo especificado. Uma resposta vazia será enviada somente se o tempo de espera de sondagem expirar. Essa opção pode reduzir o número de respostas vazias e possivelmente reduzir os custos.

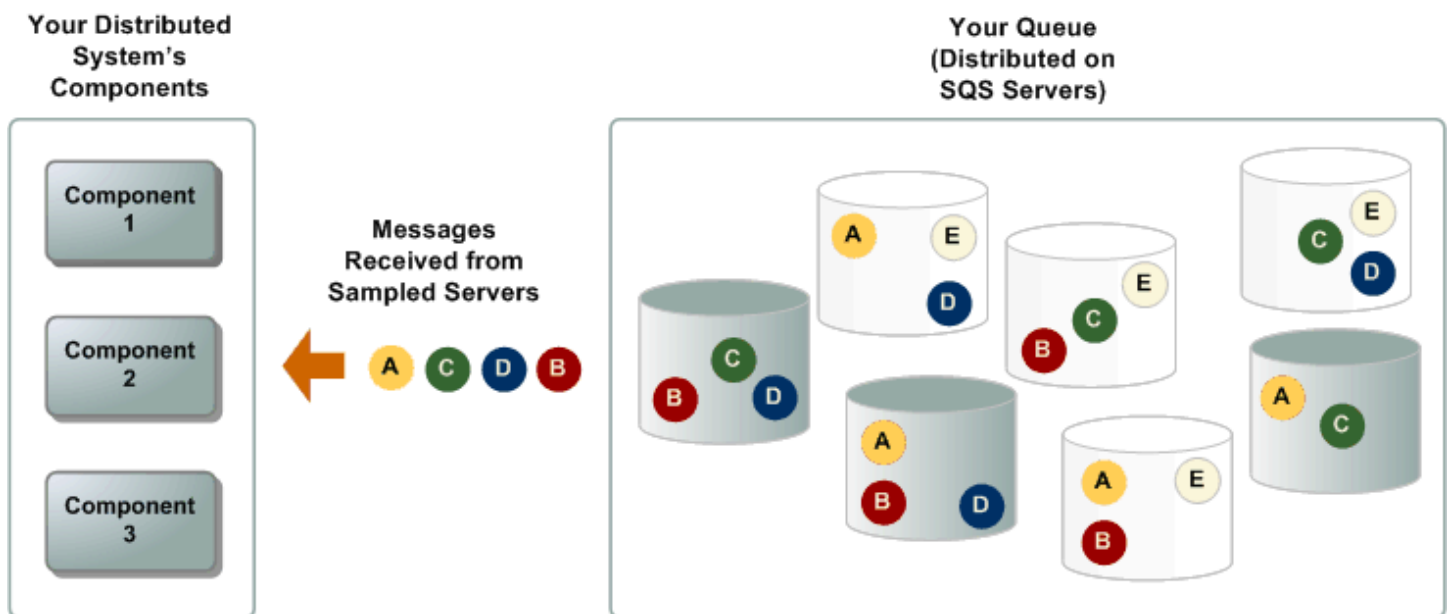
As seções a seguir explicam os detalhes de sondagens curtas e sondagens longas.

Consumo de mensagens usando sondagem curta

Quando você consome as mensagens de uma fila (FIFO ou padrão) usando a sondagem curta, o Amazon SQS obtém amostras de um subconjunto de seus servidores (com base em uma

distribuição aleatória ponderada) e retorna mensagens apenas desses servidores. Assim, uma determinada solicitação [ReceiveMessage](#) pode não retornar todas as suas mensagens. No entanto, se você tiver menos de 1.000 mensagens na fila, uma solicitação subsequente retornará suas mensagens. Se você continuar consumindo em suas filas, o Amazon SQS obterá amostras de todos os seus servidores, e você receberá todas as mensagens.

O diagrama a seguir mostra o comportamento da sondagem curta de mensagens retornadas de uma fila padrão depois que um dos componentes do sistema faz uma solicitação de recebimento. O Amazon SQS analisa vários dos seus servidores (em cinza) e retorna as mensagens A, C, D e B desses servidores. A mensagem E não é retornada para essa solicitação, mas é retornada para uma solicitação subsequente.



Consumo de mensagens usando a sondagem longa

Quando o tempo de espera da ação da API [ReceiveMessage](#) é maior do que 0, a sondagem longa está em vigor. O tempo máximo de espera de sondagem longa é de 20 segundos. A sondagem longa ajuda a reduzir os custos de uso do Amazon SQS eliminando o número de respostas vazias (quando não há mensagens disponíveis para uma solicitação `ReceiveMessage`) e respostas vazias falsas (quando mensagens estão disponíveis, mas não são incluídas em uma resposta). Para obter informações sobre como habilitar a sondagem longa para uma fila nova ou existente usando o console do Amazon SQS, consulte [Configurar parâmetros de filas usando o console do Amazon SQS](#). Para ver as práticas recomendadas, consulte [Configurar a sondagem longa no Amazon SQS](#).

A sondagem longa oferece os seguintes benefícios:

- Reduza as respostas vazias permitindo que o Amazon SQS espere até que uma mensagem esteja disponível em uma fila antes de enviar uma resposta. A menos que uma conexão expire, a resposta à solicitação `ReceiveMessage` contém pelo menos uma das mensagens disponíveis, até o número máximo de mensagens especificado na ação `ReceiveMessage`. Em casos raros, você pode receber respostas vazias mesmo quando uma fila ainda contiver mensagens, especialmente se você especificar um valor baixo para o parâmetro [ReceiveMessageWaitTimeSeconds](#).
- Reduza respostas vazias falsas consultando todos os servidores do Amazon SQS, não apenas um subconjunto deles.
- Retornar mensagens assim que se tornam disponíveis.

Para obter mais informações sobre como confirmar se uma fila está vazia, consulte [Confirmar se uma fila do Amazon SQS está vazia](#).

Diferenças entre as sondagens longa e curta

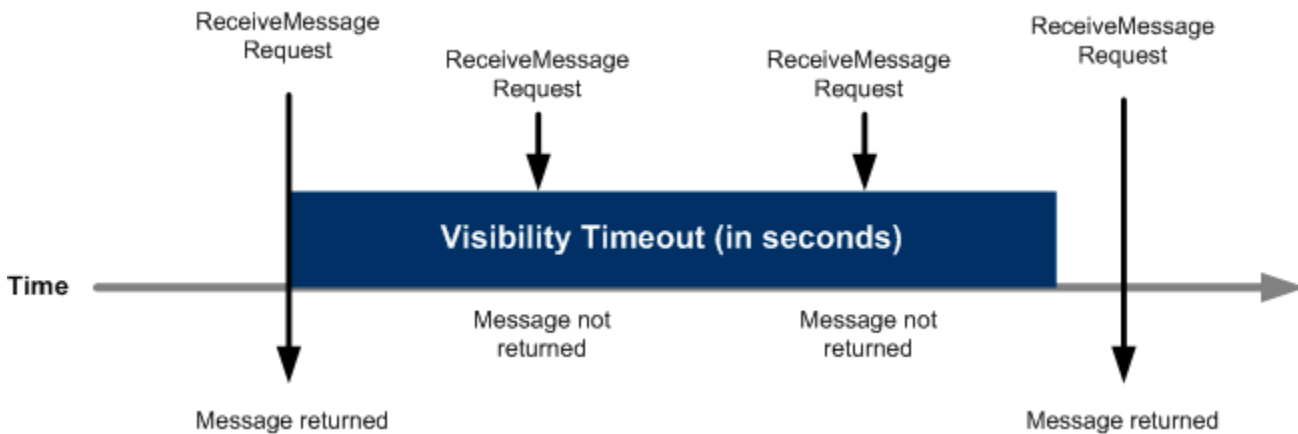
A sondagem curta ocorre quando o parâmetro [WaitTimeSeconds](#) de uma solicitação [ReceiveMessage](#) é definido como 0 de uma de duas maneiras:

- A chamada `ReceiveMessage` define `WaitTimeSeconds` como 0.
- A chamada `ReceiveMessage` não define `WaitTimeSeconds`, mas o atributo da fila [ReceiveMessageWaitTimeSeconds](#) é definido como 0.

Tempo limite de visibilidade do Amazon SQS

Quando você recebe uma mensagem de uma fila do Amazon SQS, ela permanece na fila, mas fica temporariamente invisível para outros consumidores. Essa invisibilidade é controlada pelo tempo limite de visibilidade, o que garante que outros consumidores não possam processar a mesma mensagem enquanto você estiver trabalhando nela. O Amazon SQS oferece duas opções para excluir mensagens após o processamento:

- Exclusão manual: você exclui mensagens explicitamente usando a ação [DeleteMessage](#).
- Exclusão automática: compatível com determinados AWS SDKs, as mensagens são excluídas automaticamente após o processamento bem-sucedido, simplificando os fluxos de trabalho.



Casos de uso do tempo limite de visibilidade

Gerencie tarefas de longa duração: use o tempo limite de visibilidade para lidar com tarefas que exigem tempos de processamento prolongados. Defina um tempo limite de visibilidade adequado para mensagens que exigem um tempo de processamento prolongado. Isso garante que outros consumidores não captem a mesma mensagem enquanto ela está sendo processada, evitando trabalho duplicado e mantendo a eficiência do sistema.

Implemente mecanismos de repetição: estenda o tempo limite de visibilidade programaticamente para tarefas que não foram concluídas dentro do tempo limite inicial. Se uma tarefa não for concluída dentro do tempo limite de visibilidade inicial, você poderá estender o tempo limite programaticamente. Isso permite que seu sistema tente processar novamente a mensagem sem que ela se torne visível para outros consumidores, melhorando a tolerância a falhas e a confiabilidade. Combine com filas de mensagens não entregues (DLQs) para gerenciar falhas persistentes.

Coordene sistemas distribuídos: use o tempo limite de visibilidade do SQS para coordenar tarefas em sistemas distribuídos. Defina tempos limite de visibilidade que se alinhem aos tempos de processamento esperados para diferentes componentes. Isso ajuda a manter a consistência e evita condições de corrida em arquiteturas complexas e distribuídas.

Otimize a utilização de recursos: ajuste os tempos limite de visibilidade do SQS para otimizar a utilização de recursos em sua aplicação. Ao definir tempos limite apropriados, você pode garantir que as mensagens sejam processadas com eficiência sem consumir recursos desnecessariamente. Isso leva a um melhor desempenho geral do sistema e à melhor relação custo-benefício.

Como definir e ajustar o tempo limite

O tempo limite de visibilidade começa assim que uma mensagem é entregue a você. Durante esse período, espera-se que você processe e exclua a mensagem. Se você não excluir a mensagem

antes que o tempo limite de visibilidade expire, ela ficará visível novamente na fila e poderá ser recuperada por outro consumidor. O tempo limite de visibilidade padrão para uma fila é de 30 segundos, mas você pode ajustá-lo para corresponder ao tempo que a aplicação precisa para processar e excluir uma mensagem. Você também pode definir um tempo limite de visibilidade específico para mensagens individuais sem alterar a configuração geral da fila. Use a ação [ChangeMessageVisibility](#) para estender ou reduzir programaticamente o tempo limite conforme necessário.

Cotas e mensagens em trânsito

No Amazon SQS, as mensagens em trânsito são aquelas que foram recebidas por um consumidor, mas ainda não foram excluídas. Para filas padrão, há um limite de aproximadamente 120 mil mensagens em trânsito, dependendo do tráfego da fila e do acúmulo de mensagens. Se esse limite for atingido, o Amazon SQS retornará um erro `OverLimit`, indicando que nenhuma mensagem adicional pode ser recebida até que algumas mensagens em trânsito sejam excluídas. Para filas FIFO, os limites dependem dos grupos de mensagens ativos.

- Ao usar sondagem curta: se esse limite for atingido ao usar a sondagem curta, o Amazon SQS retornará um erro `OverLimit`, indicando que nenhuma mensagem adicional pode ser recebida até que algumas mensagens em trânsito sejam excluídas.
- Ao usar sondagem longa: se você estiver usando a sondagem longa, o Amazon SQS não retornará um erro quando o limite de mensagens em trânsito for atingido. Em vez disso, ele não retornará nenhuma mensagem nova até que o número de mensagens em trânsito fique abaixo do limite.

Para gerenciar as mensagens em trânsito de forma eficaz:

1. Exclusão imediata: exclua as mensagens (manual ou automaticamente) após o processamento para reduzir a contagem em trânsito.
2. Monitore com o CloudWatch: defina alarmes para altas contagens em trânsito para evitar que o limite seja atingido.
3. Distribua a carga: se você estiver processando um grande volume de mensagens, use filas ou consumidores adicionais para equilibrar a carga e evitar gargalos.
4. Solicite um aumento de cota: envie uma solicitação ao [AWS Support](#) se forem necessários limites maiores.

Compreender o tempo limite de visibilidade em filas FIFO e padrão

Tanto nas filas padrão quanto nas FIFO, o tempo limite de visibilidade ajuda a evitar que vários consumidores processem a mesma mensagem simultaneamente. No entanto, devido ao modelo de entrega de pelo menos uma vez do Amazon SQS, não há garantia absoluta de que uma mensagem não seja entregue mais de uma vez durante o período de tempo limite de visibilidade.

- **Filas padrão:** o tempo limite de visibilidade em filas padrão ajuda a evitar que vários consumidores processem a mesma mensagem simultaneamente. No entanto, devido ao modelo de entrega de pelo menos uma vez, o Amazon SQS não garante que uma mensagem não será entregue mais de uma vez durante o período de tempo limite de visibilidade.
- **Filas FIFO:** para filas FIFO, as mensagens com o mesmo ID de grupo de mensagens são processadas em uma sequência estrita. Quando uma mensagem com um ID de grupo de mensagens está em trânsito, as mensagens subsequentes desse grupo não são disponibilizadas até que a mensagem em andamento seja excluída ou o tempo limite de visibilidade expire. No entanto, isso não “bloqueia” o grupo indefinidamente; cada mensagem é processada em sequência e somente quando cada mensagem for excluída ou se tornar visível novamente é que a próxima mensagem desse grupo estará disponível para os consumidores. Essa abordagem garante o processamento ordenado dentro do grupo sem impedir desnecessariamente que o grupo entregue mensagens.

Tratamento de falhas

Se você não processar e excluir uma mensagem antes que o tempo limite de visibilidade expire, devido a problemas como erros na aplicação, falhas ou problemas de conectividade, a mensagem voltará a ficar visível na fila. Então, ela poderá ser recuperada pelo mesmo consumidor ou por um consumidor diferente para outra tentativa de processamento. Isso garante que as mensagens não sejam perdidas mesmo se o processamento inicial falhar. No entanto, definir um tempo limite de visibilidade muito alto poderá atrasar o reaparecimento de mensagens não processadas, possivelmente retardando as novas tentativas. É essencial definir um tempo limite de visibilidade adequado com base no tempo de processamento esperado para o gerenciamento em tempo das mensagens.

Alterar e encerrar o tempo limite de visibilidade

Você pode alterar ou encerrar o tempo limite de visibilidade usando a ação `ChangeMessageVisibility`:

- Alterando o tempo limite: ajuste o tempo limite de visibilidade dinamicamente usando [ChangeMessageVisibility](#). Isso permite que você estenda ou reduza as durações de tempo limite para atender às necessidades de processamento.
- Encerrando o tempo limite: se você decidir não processar uma mensagem recebida, encerre o tempo limite de visibilidade definindo o `VisibilityTimeout` como 0 segundos por meio da ação `ChangeMessageVisibility`. Isso disponibiliza imediatamente a mensagem para que outros consumidores possam processá-la.

Práticas recomendadas

Use as práticas recomendadas a seguir para gerenciar os tempos limite de visibilidade no Amazon SQS, incluindo definir, ajustar e estender os tempos limite, bem como lidar com mensagens não processadas usando filas de mensagens não entregues (DLQs).

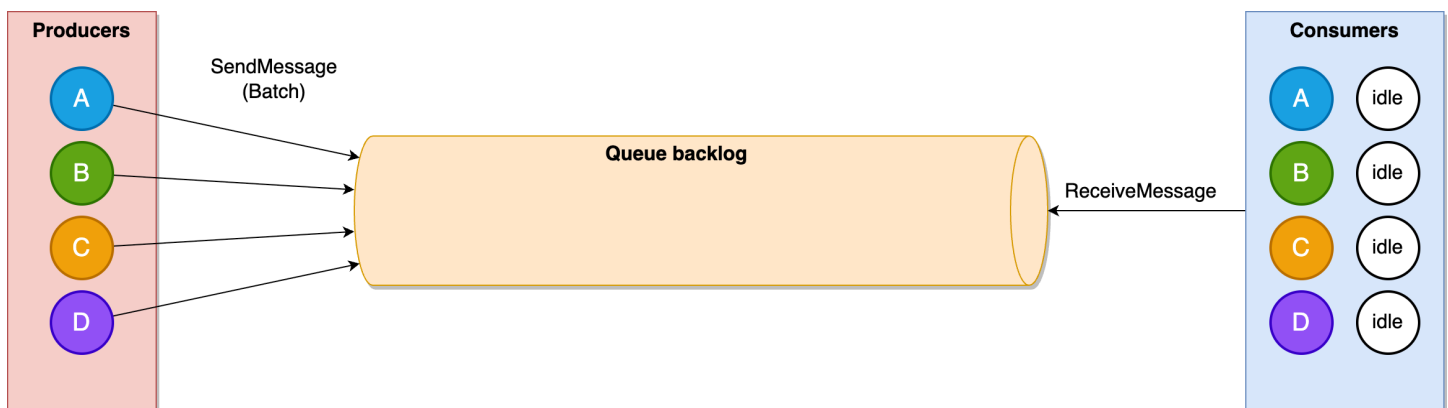
- Defina e ajuste o tempo limite. Comece definindo o tempo limite de visibilidade para corresponder ao tempo máximo que a aplicação geralmente precisa para processar e excluir uma mensagem. Se você não tiver certeza sobre o tempo exato de processamento, comece com um tempo limite menor (por exemplo, dois minutos) e estenda-o conforme necessário. Você pode implementar um mecanismo de pulsação para estender periodicamente o tempo limite de visibilidade, garantindo que a mensagem permaneça invisível até que o processamento seja concluído. Isso minimiza os atrasos no reprocessamento de mensagens não tratadas e evita a visibilidade prematura.
- Extensão do tempo limite e gerenciamento do limite de 12 horas. Se o tempo de processamento variar ou exceder o tempo limite inicialmente definido, use a ação `ChangeMessageVisibility` para estender o tempo limite de visibilidade enquanto processa a mensagem. Lembre-se de que o tempo limite de visibilidade tem um limite máximo de 12 horas a partir do momento em que a mensagem é recebida pela primeira vez. Estender o tempo limite não redefine esse limite de 12 horas. Se o processamento exigir mais tempo do que esse limite, considere usar o AWS Step Functions ou dividir a tarefa em etapas menores.
- Gerenciamento de mensagens não processadas. Para gerenciar mensagens que falham em várias tentativas de processamento, configure uma fila de mensagens não entregues (DLQ). Isso garante que as mensagens que não puderem ser processadas após várias tentativas sejam capturadas separadamente para análise ou gerenciamento adicionais, evitando que elas circulem repetidamente na fila principal.

Filas justas do Amazon SQS

As filas justas do Amazon SQS mitigam automaticamente o impacto de vizinhos barulhentos em filas multilocatário que contenham mensagens de várias entidades lógicas, como clientes, aplicações clientes ou tipos de mensagens. Nesses ambientes de filas compartilhadas, uma métrica crítica de desempenho é o tempo de permanência, que mede o tempo total que as mensagens passam em uma fila desde a chegada até o processamento. Quando um locatário cria acúmulo na fila publicando mais mensagens do que o sistema pode suportar, filas justas minimizam o impacto no tempo de permanência de outros locatários.

Estado estacionário

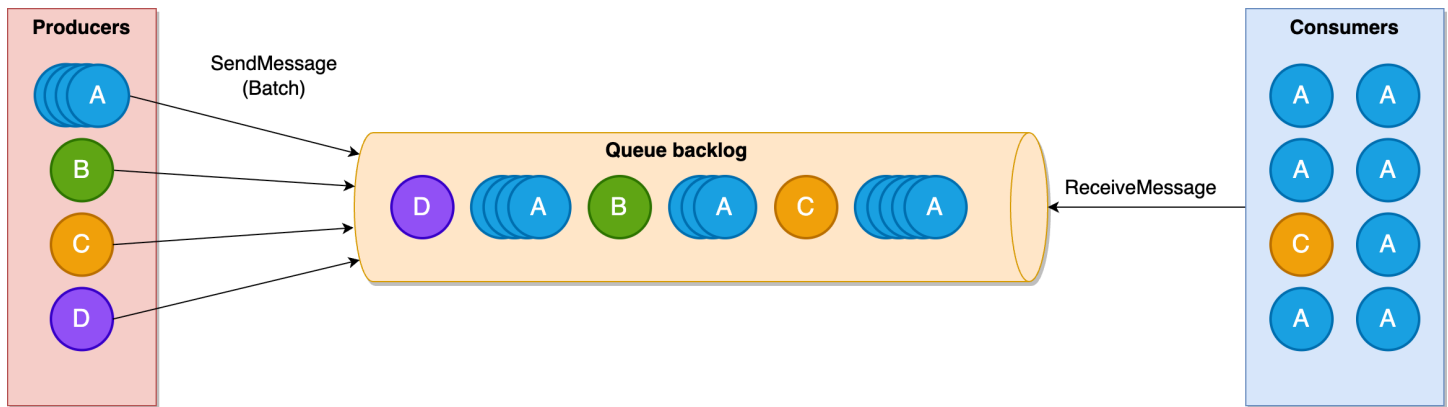
O diagrama a seguir ilustra uma fila multilocatário contendo mensagens de quatro locatários distintos (identificados como A, B, C e D). A fila opera em um estado estável e não há nenhum acúmulo de mensagens, pois os consumidores recebem mensagens assim que elas aparecem na fila. Todos os locatários experimentam tempos de permanência baixos. Nem toda a capacidade do consumidor é totalmente utilizada nesse estado estável.



Impacto de vizinhos barulhentos

O impacto de noisy neighbor (vizinho barulhento) ocorre quando um locatário em uma fila de vários locatários cria um acúmulo, aumentando o tempo de espera das mensagens para todos os outros. Um locatário pode se tornar um vizinho barulhento quando envia um volume maior de mensagens do que outros locatários ou quando os consumidores demoram mais para processar as mensagens desse locatário em particular.

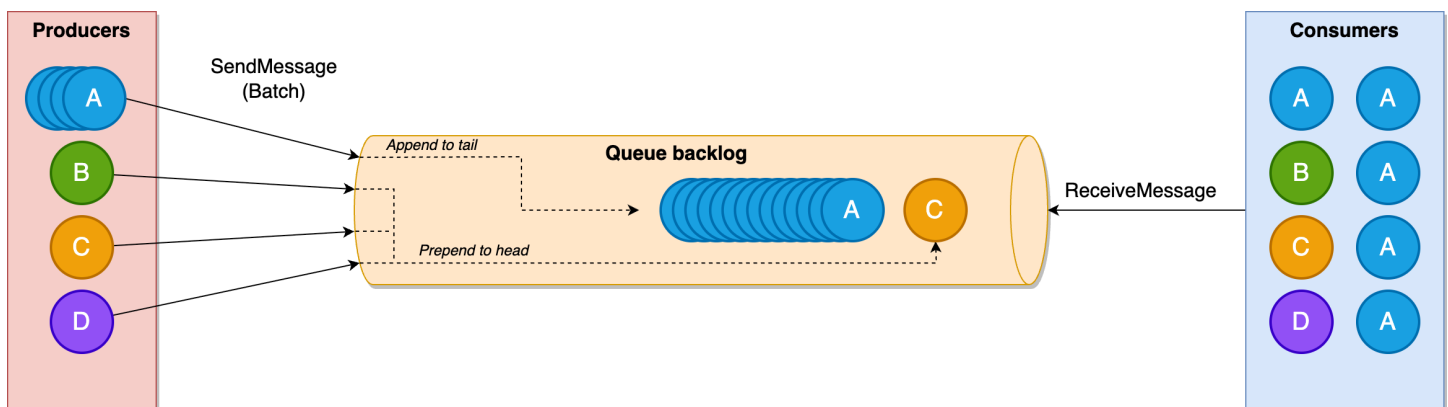
Este diagrama ilustra como o aumento do tráfego do Locatário A cria um acúmulo na fila. Os consumidores estão ocupados processando as mensagens apenas do Locatário A, enquanto as mensagens de outros locatários aguardam, resultando em maiores tempos de espera para todos.



Mitigação com filas justas

O Amazon SQS detecta vizinhos barulhentos monitorando a distribuição de mensagens entre os locatários durante o processamento (o estado “em trânsito”). Quando um locatário tem um número desproporcionalmente grande de mensagens em trânsito em comparação a outros, o Amazon SQS o identifica como um vizinho barulhento e prioriza a entrega de mensagens para outros locatários. Essa abordagem reduz o impacto do tempo de permanência nos outros locatários.

Este diagrama ilustra como as filas justas do Amazon SQS abordam o problema do vizinho barulhento. Quando um locatário (Locatário A) fica barulhento, o Amazon SQS prioriza o retorno de mensagens de outros locatários (B, C e D). Essa priorização ajuda a manter baixos tempos de permanência para os locatários silenciosos — Locatário B, C e D —, enquanto o tempo de permanência das mensagens do Locatário A é elevado até que o acúmulo da fila seja consumido sem afetar outros locatários.



Note

O Amazon SQS não limita a taxa de consumo por locatário. Ele permite que os consumidores recebam mensagens de locatários vizinhos barulhentos quando há capacidade de consumo e a fila não tem outras mensagens para retornar. Assim como as

filas padrão do Amazon SQS, as filas justas permitem throughput praticamente ilimitado, e não há limites para o número de locatários que você pode ter na fila.

Diferença com filas FIFO

As filas FIFO mantêm uma ordenação estrita, limitando o número de mensagens em trânsito de cada locatário. Embora isso evite vizinhos barulhentos, também limita o throughput de cada locatário. As filas justas são projetadas para cenários de vários locatários em que alto throughput, baixo tempo de permanência e alocação justa de recursos são prioridades. Filas justas permitem que vários consumidores processem mensagens do mesmo locatário simultaneamente, ao mesmo tempo em que ajuda todos os inquilinos a manter tempos de permanência consistentes.

Uso de filas justas

Os produtores de mensagens podem adicionar um identificador de locatário definindo um `MessageGroupId` em uma mensagem de saída:

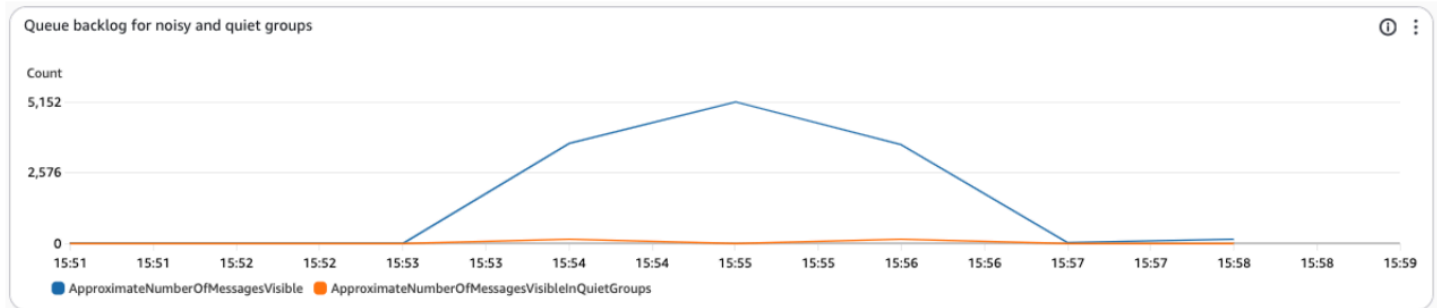
```
// Send message with tenant identifier
SendMessageRequest request = new SendMessageRequest()
    .withQueueUrl(queueUrl)
    .withMessageBody(messageBody)
    .withMessageGroupId("tenant-123"); // Tenant identifier
sqs.sendMessage(request);
```

A capacidade de equidade será aplicada automaticamente em todas as filas padrão do Amazon SQS para mensagens com a propriedade `MessageGroupId`. Nenhuma alteração é necessária no código do consumidor, não há impacto na latência da API e não vem com limitação de taxa de throughput.

Métricas do CloudWatch de filas justas

O Amazon SQS fornece métricas adicionais do CloudWatch que ajudam a monitorar a mitigação do impacto de vizinhos barulhentos. Como exemplo, você pode comparar as métricas `Approximate..InQuietGroups` com métricas padrão em nível de fila. Durante picos de tráfego para um locatário específico, as métricas gerais do nível da fila podem revelar o aumento dos acúmulos ou a idade das mensagens mais antigas. No entanto, ao analisar os grupos silenciosos isoladamente, você pode identificar que a maioria dos grupos de mensagens não ruidosos ou locatários não são afetados.

Abaixo, você pode encontrar um exemplo em que a métrica padrão do acúmulo da fila (`ApproximateNumberOfMessagesVisible`) aumenta devido a um locatário barulhento, enquanto o acúmulo para locatários não barulhentos (`ApproximateNumberOfMessagesVisibleInQuietGroups`) permanece baixo.



Para obter uma lista completa das métricas do Amazon SQS CloudWatch e suas descrições, consulte as [métricas do CloudWatch para o Amazon SQS](#).

Filas de atraso do Amazon SQS

As filas de atraso permitem adiar a entrega de novas mensagens para consumidores por alguns segundos, por exemplo, quando sua aplicação de consumo precisa de tempo adicional para processar mensagens. Se você criar uma fila de atraso, qualquer mensagem enviada para essa fila permanecerá invisível para os consumidores durante o período de atraso. O atraso padrão (mínimo) para uma fila é 0 segundo. O máximo é 15 minutos. Para obter mais informações sobre como configurar filas de atraso usando o console, consulte [Configurar parâmetros de filas usando o console do Amazon SQS](#).

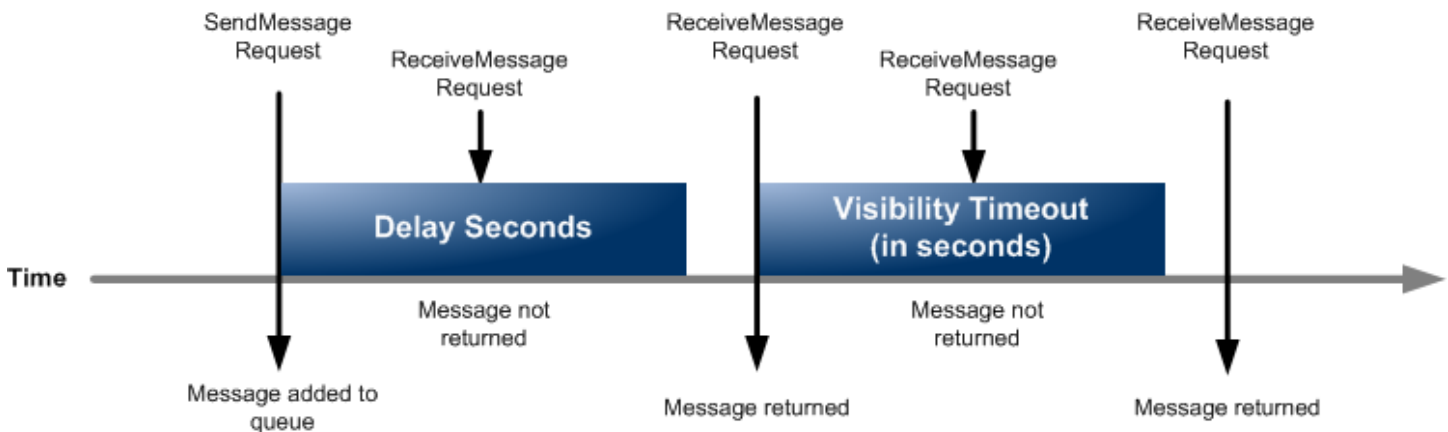
Note

Para filas padrão, a configuração de atraso por fila não é retroativa. A alteração da configuração não afeta o atraso de mensagens que já estão na fila.

Para filas FIFO, a configuração de atraso por fila é retroativa. A alteração da configuração afeta o atraso de mensagens que já estão na fila.

As filas de atraso são semelhantes a [tempos limite de visibilidade](#), pois os dois recursos tornam as mensagens indisponíveis para os consumidores por um período específico. A diferença entre os dois é que, para filas de atraso, uma mensagem é ocultada quando é adicionada à fila pela primeira vez, enquanto que para os tempos limite de visibilidade uma mensagem é ocultada somente depois que

a mensagem é consumida na fila. O diagrama a seguir ilustra a relação entre filas de atraso e os tempos limite de visibilidade.



Opções estendidas de agendamento

Embora as filas de atraso e os temporizadores de mensagens do Amazon SQS permitam programar a entrega de mensagens em até 15 minutos no futuro, você pode precisar de recursos de agendamento mais flexíveis. Nesses casos, considere usar o [EventBridge Scheduler](#), que permite agendar bilhões de ações de API únicas ou recorrentes sem limitações de tempo. O EventBridge Scheduler é a solução recomendada para casos de uso avançados de agendamento de mensagens.

Para definir os segundos de atraso em mensagens individuais, em vez de em uma fila inteira, use [temporizadores de mensagem](#) para permitir que o Amazon SQS use o valor `DelaySeconds` do temporizador de mensagem em vez do valor `DelaySeconds` da fila. O [EventBridge Scheduler](#) também é compatível com o agendamento de mensagens individuais.

Filas temporárias do Amazon SQS

Filas temporárias ajudam você a economizar tempo de desenvolvimento e custos de implantação ao usar padrões comuns de mensagens, como solicitação-resposta. Você pode usar o [Temporary Queue Client](#) para criar filas temporárias de alta taxa de transferência, econômicas e gerenciadas por aplicações.

O cliente mapeia várias filas temporárias (filas gerenciadas pela aplicação criadas sob demanda para um processo específico) em uma única fila do Amazon SQS automaticamente. Isso permite que o aplicativo faça menos chamadas de API e tenha uma taxa de transferência mais alta quando o tráfego é baixo para cada fila temporária. Quando uma fila temporária não está mais em uso, o cliente a limpa automaticamente, mesmo que alguns processos que usam o cliente não estejam encerrados corretamente.

Veja a seguir os benefícios das filas temporárias:

- Elas funcionam como canais de comunicação leves para segmentos ou processos específicos.
- É possível criá-las e excluí-las sem gerar custos adicionais.
- Elas são compatíveis em termos de API com filas estáticas (normais) do Amazon SQS. Isso significa que o código existente que envia e recebe mensagens pode enviar e receber mensagens de filas virtuais.

Filas virtuais

Filas virtuais são estruturas de dados locais criadas pelo Temporary Queue Client. As filas virtuais permitem combinar vários destinos de baixo tráfego em uma única fila do Amazon SQS. Consulte as práticas recomendadas em “Evitar reutilizar o mesmo ID de grupo de mensagens com filas virtuais”.

Note

- Ao criar uma fila virtual, você cria apenas estruturas de dados temporárias nas quais os consumidores receberão as mensagens. Como não fazem chamadas de API para o Amazon SQS, as filas virtuais não geram custo.
- As cotas de TPS se aplicam a todas as filas virtuais em uma única fila de host. Para obter mais informações, consulte [Cotas de mensagens do Amazon SQS](#).

A classe de wrapper `AmazonSQSVirtualQueuesClient` adiciona suporte para atributos relacionados a filas virtuais. Para criar uma fila virtual, você deve chamar a ação de API `CreateQueue` usando o atributo `HostQueueURL`. Esse atributo especifica a fila existente que hospeda as filas virtuais.

O URL de uma fila virtual tem o formato a seguir.

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue#MyVirtualQueueName
```

Quando um produtor chama a ação de API `SendMessage` ou `SendMessageBatch` em um URL de fila virtual, o Temporary Queue Client faz o seguinte:

1. Extrai o nome da fila virtual.

2. Anexa o nome da fila virtual como um atributo de mensagem adicional.
3. Envia a mensagem para a fila de host.

Enquanto o produtor envia mensagens, um thread em segundo plano faz uma sondagem da fila de host e envia as mensagens recebidas para filas virtuais de acordo com os atributos de mensagem correspondentes.

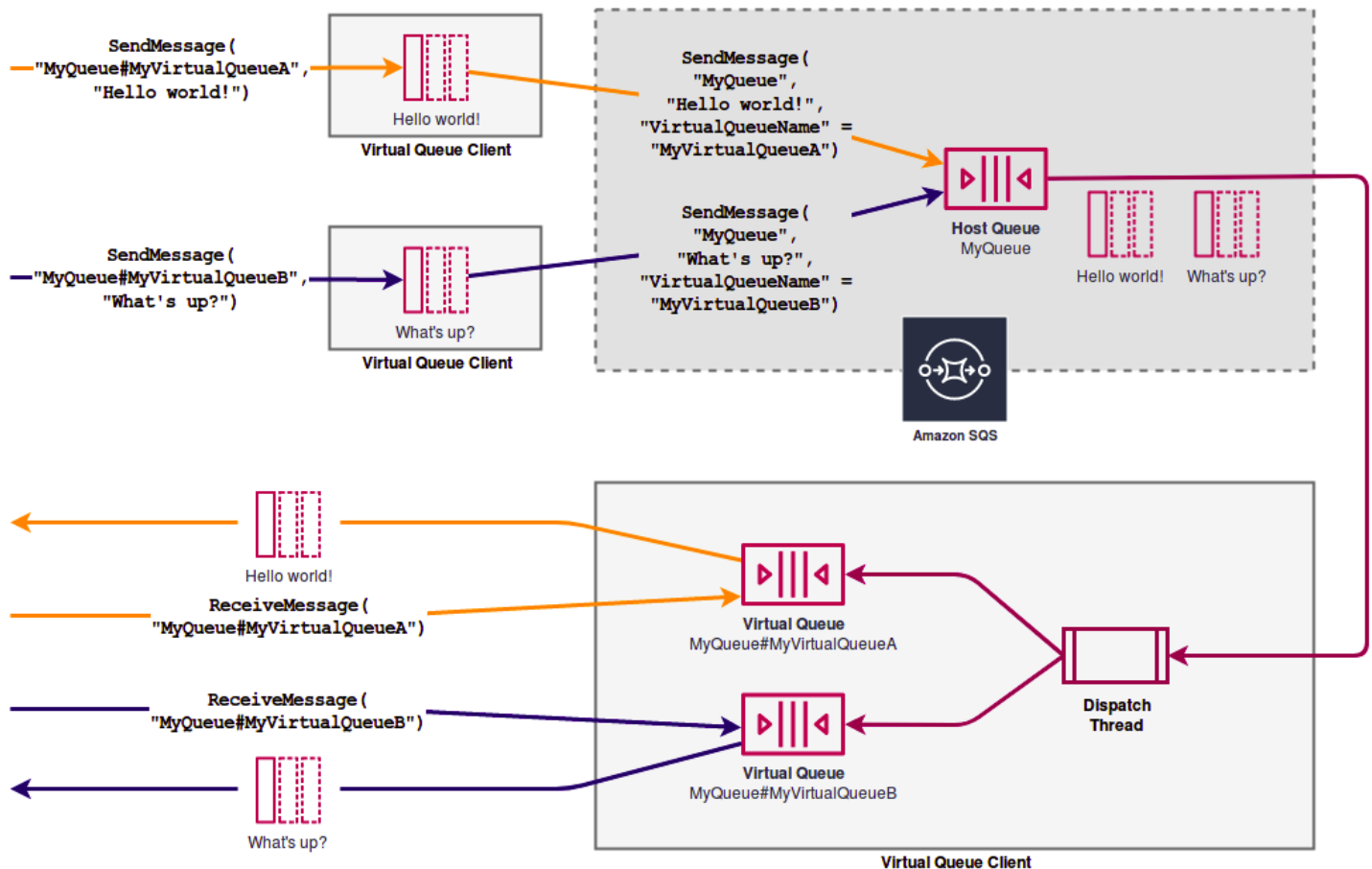
Enquanto o consumidor chama a ação de API `ReceiveMessage` em um URL de fila virtual, o `Temporary Queue Client` bloqueia a chamada localmente até o thread de fundo enviar uma mensagem para a fila virtual. (Esse processo é semelhante à pré-busca de mensagens no [Cliente assíncrono armazenado em buffer](#): uma única ação de API pode fornecer mensagens para até 10 filas virtuais.) A exclusão de uma fila virtual remove todos os recursos do lado do cliente sem chamar o Amazon SQS.

A classe `AmazonSQSTemporaryQueuesClient` transforma automaticamente todas as filas criadas em filas temporárias. Ela também cria filas de host com os mesmos atributos automaticamente, sob demanda. Os nomes dessas filas compartilham o mesmo prefixo configurável (por padrão, `__RequesterClientQueues__`) que as identifica como filas temporárias. Isso permite que o cliente atue como uma substituição inicial que otimiza o código existente que cria e exclui filas. O cliente também inclui as interfaces `AmazonSQSRequester` e `AmazonSQSResponder` que permitem a comunicação bidirecional entre as filas.

Padrão de mensagens de resposta a solicitação (filas virtuais)

O caso de uso mais comum de filas temporárias é o padrão de mensagens de solicitação-resposta no qual um solicitante cria uma fila temporária para receber cada mensagem de resposta. Para evitar a criação de uma fila do Amazon SQS para cada mensagem de resposta, o `Temporary Queue Client` (cliente de fila temporária) permite criar e excluir várias filas temporárias sem fazer chamadas de API ao Amazon SQS. Consulte mais informações em “Implementação de sistemas de solicitação-resposta”.

O diagrama a seguir mostra uma configuração comum usando esse padrão.



Cenário de exemplo: processar uma solicitação de login

O cenário de exemplo a seguir mostra como você pode usar as interfaces `AmazonSQSRequester` e `AmazonSQSResponder` para processar uma solicitação de login do usuário.

No lado do cliente

```
public class LoginClient {

    // Specify the Amazon SQS queue to which to send requests.
    private final String requestQueueUrl;

    // Use the AmazonSQSRequester interface to create
    // a temporary queue for each response.
    private final AmazonSQSRequester sqsRequester =
        AmazonSQSRequesterClientBuilder.defaultClient();

    LoginClient(String requestQueueUrl) {
        this.requestQueueUrl = requestQueueUrl;
    }
}
```

```
}

// Send a login request.
public String login(String body) throws TimeoutException {
    SendMessageRequest request = new SendMessageRequest()
        .withMessageBody(body)
        .withQueueUrl(requestQueueUrl);

    // If no response is received, in 20 seconds,
    // trigger the TimeoutException.
    Message reply = sqsRequester.sendMessageAndGetResponse(request,
        20, TimeUnit.SECONDS);

    return reply.getBody();
}
}
```

O envio de uma solicitação de login faz o seguinte:

1. Cria uma fila temporária.
2. Anexa o URL da fila temporária à mensagem como um atributo.
3. Envia a mensagem.
4. Recebe uma resposta da fila temporária.
5. Exclui a fila temporária.
6. Retorna a resposta.

No lado do servidor

O exemplo a seguir pressupõe que, após a construção, um thread é criado para sondar a fila e chamar o método `handleLoginRequest()` para cada mensagem. Além disso, `doLogin()` é um método presumido.

```
public class LoginServer {

    // Specify the Amazon SQS queue to poll for login requests.
    private final String requestQueueUrl;

    // Use the AmazonSQSResponder interface to take care
    // of sending responses to the correct response destination.
```



```
private final AmazonSQSResponder sqsResponder =
    AmazonSQSResponderClientBuilder.defaultClient();

LoginServer(String requestQueueUrl) {
    this.requestQueueUrl = requestQueueUrl;
}

// Process login requests from the client.
public void handleLoginRequest(Message message) {

    // Process the login and return a serialized result.
    String response = doLogin(message.getBody());

    // Extract the URL of the temporary queue from the message attribute
    // and send the response to the temporary queue.
    sqsResponder.sendResponseMessage(MessageContent.fromMessage(message),
        new MessageContent(response));
}
}
```

Limpeza das filas

Para garantir que o Amazon SQS recupere todos os recursos na memória usados por filas virtuais, quando a aplicação não precisar mais do Temporary Queue Client, ele deverá chamar o método `shutdown()`. Você também pode usar o método `shutdown()` da interface `AmazonSQSRequester`.

O Temporary Queue Client também fornece uma maneira de eliminar filas de host órfãs. Para cada fila que recebe uma chamada de API durante um período (por padrão, cinco minutos), o cliente usa a ação de API `TagQueue` para etiquetar uma fila que permanece em uso.

Note

Qualquer ação de API executada em uma fila será marcada como não ociosa, inclusive uma ação `ReceiveMessage` que não retorne mensagens.

O thread em segundo plano usa as ações de API `ListQueues` e `ListTags` para verificar todas as filas com o prefixo configurado, excluindo as filas que não foram marcadas por pelo menos cinco minutos. Dessa forma, se um cliente não for encerrado corretamente, os outros clientes ativos serão

limpos depois dele. A fim de reduzir a duplicação de trabalho, todos os clientes com o mesmo prefixo se comunicam por meio de uma fila de trabalho interna compartilhada, denominada de acordo com o prefixo.

Temporizadores de mensagens do Amazon SQS

Os temporizadores de mensagem permitem especificar um período de invisibilidade inicial para uma mensagem quando adicionada a uma fila. Por exemplo, se você enviar uma mensagem com um temporizador de 45 segundos, a mensagem ficará invisível para os consumidores nos primeiros 45 segundos. O atraso padrão (mínimo) para uma mensagem é 0 segundo. O máximo é 15 minutos. Para obter informações sobre o envio de mensagens com temporizadores usando o console, consulte [Como enviar uma mensagem usando uma fila padrão](#).

Note

As filas FIFO não são compatíveis com temporizadores em mensagens individuais.

Para definir um período de atraso em uma fila inteira, em vez de mensagens individuais, use [filas de atraso](#). Uma definição de temporizador de mensagem para uma mensagem individual substitui qualquer valor de `DelaySeconds` em uma fila de atraso do Amazon SQS.

Opções estendidas de agendamento

Embora as filas de atraso e os temporizadores de mensagens do Amazon SQS permitam programar a entrega de mensagens em até 15 minutos no futuro, você pode precisar de recursos de agendamento mais flexíveis. Nesses casos, considere usar o [EventBridge Scheduler](#), que permite agendar bilhões de ações de API únicas ou recorrentes sem limitações de tempo. O EventBridge Scheduler é a solução recomendada para casos de uso avançados de agendamento de mensagens.

Acessar pipes do Amazon EventBridge pelo console do Amazon SQS

Os pipes do Amazon EventBridge conectam as origens aos destinos. Os pipes são destinados a funcionar como integrações ponto a ponto entre origens e destinos compatíveis, com suporte para transformações e enriquecimento avançados. Os pipes do EventBridge fornecem uma maneira

altamente escalável de conectar a fila do Amazon SQS a serviços da AWS, como Step Functions, Amazon SQS e API Gateway, além de aplicações de software como serviço (SaaS) de terceiros, como o Salesforce.

Para configurar um pipe, a origem é escolhida, adiciona filtragem opcional, define o enriquecimento opcional e escolhe o destino para os dados do evento.

Na página de detalhes de uma fila do Amazon SQS, você pode ver os pipes que usam essa fila como origem. Nessa página, você também pode:

- Iniciar o console do EventBridge para visualizar os detalhes do pipe.
- Inicie o console do EventBridge para criar um pipe com a fila como origem.

Para ter mais informações sobre a configuração de uma fila do Amazon SQS como origem do pipe, consulte [Amazon SQS queue as a source](#) no Guia do usuário do Amazon EventBridge. Para obter mais informações sobre os pipes do EventBridge em geral, consulte [Pipes do EventBridge](#).

Para acessar os pipes do EventBridge para determinada fila do Amazon SQS

1. Abra a [página Queues](#) (Filas) do console do Amazon SQS.
2. Selecione uma fila.
3. Na página de detalhes da fila, escolha a guia Pipes do EventBridge.

A guia Pipes do EventBridge inclui uma lista de todos os pipes configurados para usar a fila selecionada como fonte, incluindo:

- nome do pipe
 - status atual
 - destino do pipe
 - última modificação do pipe
4. Veja mais detalhes do pipe ou crie um, se desejar:
 - Para acessar mais detalhes sobre um pipe:

Selecione o nome do pipe.

Isso abre a página Detalhes do pipe no console do EventBridge.

- Para criar um pipe:

Escolha Conectar fila do Amazon SQS ao pipe.

Isso abre a página Criar pipe do console do EventBridge, com a fila do Amazon SQS especificada como a origem do pipe. Para obter mais informações, consulte [Criar um pipe do EventBridge](#) no Guia do usuário do Amazon EventBridge.

 Important

Uma mensagem em uma fila do Amazon SQS é lida por um único pipe e excluída da fila após ser processada, correspondendo ou não ao filtro configurado para o pipe. Tenha cuidado ao configurar vários pipes usando a mesma fila como origem.

Gerenciar mensagens grandes do Amazon SQS com a biblioteca do cliente em versão ampliada e o Amazon Simple Storage Service

Use a [biblioteca do cliente em versão ampliada para Java do Amazon SQS](#) e a [biblioteca do cliente em versão ampliada para Python do Amazon SQS](#) para enviar mensagens grandes, principalmente cargas úteis entre 256 KB e 2 GB. Essas bibliotecas armazenam a carga útil da mensagem em um bucket do Amazon S3 e enviam uma referência ao objeto armazenado na fila do Amazon SQS.

 Note

As bibliotecas do cliente em versão ampliada do Amazon SQS são compatíveis com filas comuns e FIFO.

Gerenciar mensagens grandes do Amazon SQS usando Java e Amazon S3

Use a [biblioteca do cliente em versão ampliada do Amazon SQS para Java](#) com o Amazon S3 para gerenciar mensagens do Amazon SQS, especialmente para cargas úteis que variam de 256 KB a 2 GB. A biblioteca armazena a carga útil da mensagem em um bucket do Amazon S3 e envia uma mensagem contendo uma referência do objeto armazenado na fila do Amazon SQS.

Com a biblioteca do cliente em versão ampliada para Java, você pode:

- Especificar se as mensagens são sempre armazenadas no Amazon S3 ou apenas quando uma mensagem tiver mais de 256 KB.
- Enviar uma mensagem que faça referência a um único objeto de mensagem armazenado em um bucket do S3.
- Recuperar o objeto de mensagem de um bucket do Amazon S3
- Excluir o objeto de mensagem de um bucket do Amazon S3

Pré-requisitos

O exemplo a seguir usa o AWS SDK for Java. Para instalar e configurar o SDK, consulte [Configurar o AWS SDK for Java](#) no Guia do desenvolvedor do AWS SDK para Java.

Antes de executar o código de exemplo, configure suas credenciais da AWS. Para mais informações, consulte [Configurar as credenciais e a região da AWS para desenvolvimento](#) no Guia do desenvolvedor do AWS SDK para Java.

O [SDK for Java](#) e a biblioteca cliente Java estendida para o Amazon SQS exigem o J2SE Development Kit 8.0 ou posterior.

Note

Você pode usar a biblioteca cliente Java estendida para o Amazon SQS para gerenciar mensagens do Amazon SQS usando o Amazon S3 somente com o AWS SDK para Java. Você pode fazer isso com a AWS CLI, o console do Amazon SQS, a API HTTP do Amazon SQS ou qualquer um dos outros AWS SDKs.

Exemplo do AWS SDK for Java 2.x: usar o Amazon S3 para gerenciar mensagens grandes do Amazon SQS

O exemplo de SDK para SDK para Java 2.x a seguir usa a biblioteca do cliente em versão ampliada para Java para trabalhar com mensagens grandes. No construtor, o seguinte código:

- Cria um bucket do Amazon S3 com um nome aleatório.
- Cria uma fila do SQS que começa com MyQueue
- Envolve um cliente Amazon S3 padrão do Java SDK em uma instância de um `AmazonSQSExtendedClient`

No método `sendAnReceiveMessage`, o exemplo envia uma mensagem aleatória, que é armazenada em um bucket do Amazon S3 porque é maior que 256 KB (o tamanho máximo padrão da mensagem). Por fim, o método recupera a mensagem e exibe informações sobre ela no console.

Você pode ver o exemplo completo em https://github.com/awsdocs/aws-doc-sdk-examples/blob/94d1b24df12deda0f4fd91433b8231fed6d18b85/javav2/example_code/sqs/src/main/java/com/example/sqs/SqsExtendedClientExample.java#L1.

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 * https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Examples of using Amazon SQS Extended Client Library for Java 2.x
 *
 */
public class SqsExtendedClientExamples {
    // Create an Amazon S3 bucket with a random name.
    private final static String amzn-s3-demo-bucket = UUID.randomUUID() + "-"
        + DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

    public static void main(String[] args) {

        /**
         * Create a new instance of the builder with all defaults (credentials
         * and region) set automatically. For more information, see
         * Creating Service Clients in the AWS SDK for Java Developer Guide.
         */
        final S3Client s3 = S3Client.create();

        /**
         * Set the Amazon S3 bucket name, and then set a lifecycle rule on the
         * bucket to permanently delete objects 14 days after each object's
         * creation date.
         */
        final LifecycleRule lifeCycleRule = LifecycleRule.builder()
            .expiration(LifecycleExpiration.builder().days(14).build())
            .filter(LifecycleRuleFilter.builder().prefix("").build())
            .status(ExpirationStatus.ENABLED)
            .build();
        final BucketLifecycleConfiguration lifecycleConfig =
            BucketLifecycleConfiguration.builder()
```

```
        .rules(lifeCycleRule)
        .build();

// Create the bucket and configure it
s3.createBucket(CreateBucketRequest.builder().bucket(amzn-s3-demo-
bucket).build());

s3.putBucketLifecycleConfiguration(PutBucketLifecycleConfigurationRequest.builder()
    .bucket(amzn-s3-demo-bucket)
    .lifecycleConfiguration(lifecycleConfig)
    .build());
System.out.println("Bucket created and configured.");

// Set the Amazon SQS extended client configuration with large payload support
enabled
final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration().withPayloadSupportEnabled(s3, amzn-s3-demo-bucket);

final SqsClient sqsExtended = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);

// Create a long string of characters for the message object
int stringLength = 300000;
char[] chars = new char[stringLength];
Arrays.fill(chars, 'x');
final String myLongString = new String(chars);

// Create a message queue for this example
final String queueName = "MyQueue-" + UUID.randomUUID();
final CreateQueueResponse createQueueResponse =
sqsExtended.createQueue(CreateQueueRequest.builder().queueName(queueName).build());
final String myQueueUrl = createQueueResponse.queueUrl();
System.out.println("Queue created.");

// Send the message
final SendMessageRequest sendMessageRequest = SendMessageRequest.builder()
    .queueUrl(myQueueUrl)
    .messageBody(myLongString)
    .build();
sqsExtended.sendMessage(sendMessageRequest);
System.out.println("Sent the message.");

// Receive the message
```



```

        final ReceiveMessageResponse receiveMessageResponse =
sqsExtended.receiveMessage(ReceiveMessageRequest.builder().queueUrl(myQueueUrl).build());
        List<Message> messages = receiveMessageResponse.messages();

        // Print information about the message
        for (Message message : messages) {
            System.out.println("\nMessage received.");
            System.out.println("  ID: " + message.messageId());
            System.out.println("  Receipt handle: " + message.receiptHandle());
            System.out.println("  Message body (first 5 characters): " +
message.body().substring(0, 5));
        }

        // Delete the message, the queue, and the bucket
        final String messageReceiptHandle = messages.get(0).receiptHandle();

sqsExtended.deleteMessage(DeleteMessageRequest.builder().queueUrl(myQueueUrl).receiptHandle(me
        System.out.println("Deleted the message.");

sqsExtended.deleteQueue(DeleteQueueRequest.builder().queueUrl(myQueueUrl).build());
        System.out.println("Deleted the queue.");

        deleteBucketAndAllContents(s3);
        System.out.println("Deleted the bucket.");

    }

    private static void deleteBucketAndAllContents(S3Client client) {
        ListObjectsV2Response listObjectsResponse =
client.listObjectsV2(ListObjectsV2Request.builder().bucket(amzn-s3-demo-
bucket).build());

        listObjectsResponse.contents().forEach(object -> {
            client.deleteObject(DeleteObjectRequest.builder().bucket(amzn-s3-demo-
bucket).key(object.key()).build());
        });

        ListObjectVersionsResponse listVersionsResponse =
client.listObjectVersions(ListObjectVersionsRequest.builder().bucket(amzn-s3-demo-
bucket).build());

        listVersionsResponse.versions().forEach(version -> {

```

```
        client.deleteObject(DeleteObjectRequest.builder().bucket(amzn-s3-demo-
bucket).key(version.key()).versionId(version.versionId()).build());
    });

    client.deleteBucket(DeleteBucketRequest.builder().bucket(amzn-s3-demo-
bucket).build());
}
}
```

Você pode [usar o Apache Maven](#) para configurar e compilar a biblioteca do cliente em versão ampliada seu projeto do Java ou compilar o próprio SDK. Especifique módulos individuais do SDK usados na aplicação.

```
<properties>
    <aws-java-sdk.version>2.20.153</aws-java-sdk.version>
</properties>

<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>sqs</artifactId>
        <version>${aws-java-sdk.version}</version>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
        <version>${aws-java-sdk.version}</version>
    </dependency>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>amazon-sqs-java-extended-client-lib</artifactId>
        <version>2.0.4</version>
    </dependency>

    <dependency>
        <groupId>joda-time</groupId>
        <artifactId>joda-time</artifactId>
        <version>2.12.6</version>
    </dependency>
</dependencies>
```

Gerenciar mensagens grandes do Amazon SQS usando Python e Amazon S3

Use a [biblioteca do cliente em versão ampliada para Python do Amazon SQS](#) com o Amazon S3 para enviar mensagens grandes do Amazon SQS, principalmente cargas úteis entre 256 KB e 2 GB. A biblioteca armazena a carga útil da mensagem em um bucket do Amazon S3 e envia uma mensagem contendo uma referência do objeto armazenado na fila do Amazon SQS.

Com a biblioteca do cliente em versão ampliada para Python do Amazon SQS, você pode:

- Especificar se as cargas úteis devem ser sempre armazenadas no Amazon S3 ou apenas no Amazon S3 quando uma carga útil tiver mais de 256 KB.
- Enviar uma mensagem que faça referência a um único objeto de mensagem armazenado em um bucket do Amazon S3.
- Recuperar o objeto de carga útil correspondente de um bucket do Amazon S3
- Excluir o objeto de carga útil correspondente de um bucket do Amazon S3

Pré-requisitos

Veja abaixo os pré-requisitos para usar a biblioteca do cliente em versão ampliada para Python do Amazon SQS:

- Uma conta da AWS com as credenciais necessárias. Para criar uma conta da AWS, acesse a [página inicial da AWS](#) e escolha Criar uma conta da AWS. Siga as instruções. Consulte informações sobre credenciais em [Credentials](#).
- Um SDK da AWS: o exemplo nesta página usa o AWS Python SDK Boto3. Para instalar e configurar o SDK, consulte a documentação do [AWS SDK para Python](#) no Guia do desenvolvedor do AWS SDK para Python
- Python 3.x (ou posterior) e pip.
- A biblioteca do cliente em versão ampliada para Python do Amazon SQS, também disponível no [PyPI](#)

 Note

É possível usar a biblioteca do cliente em versão ampliada para Python do Amazon SQS para gerenciar mensagens do Amazon SQS usando o Amazon S3 somente com o AWS SDK para Python. Você pode fazer isso com a AWS CLI, o console do Amazon SQS, a API HTTP do Amazon SQS ou qualquer um dos outros SDKs da AWS.

Configurar o armazenamento de mensagens

O cliente em versão ampliada do Amazon SQS usa os seguintes atributos de mensagem para configurar as opções de armazenamento de mensagens do Amazon S3:

- `large_payload_support`: o nome do bucket do Amazon S3 para armazenar mensagens grandes.
- `always_through_s3`: se `True`, então todas as mensagens serão armazenadas no Amazon S3. Se `False`, mensagens menores que 256 KB não serão serializadas no bucket do S3. O padrão é `False`.
- `use_legacy_attribute`: se `True`, todas as mensagens publicadas usam o atributo de mensagem reservada legado (`SQSLargePayloadSize`) em vez do atributo de mensagem reservada atual (`ExtendedPayloadSize`).

Gerenciar mensagens grandes do Amazon SQS com a biblioteca do cliente em versão ampliada para Python

O exemplo a seguir cria um bucket do Amazon S3 com um nome aleatório. Depois, ele cria uma fila do Amazon SQS chamada `MyQueue` e envia para ela uma mensagem que é armazenada em um bucket do S3 e é maior que 256 KB. Por fim, o código recupera a mensagem, retorna informações sobre ela e, em seguida, exclui a mensagem, a fila e o bucket.

```
import boto3
import sqs_extended_client

#Set the Amazon SQS extended client configuration with large payload.
sqs_extended_client = boto3.client("sqs", region_name="us-east-1")
sqs_extended_client.large_payload_support = "amzn-s3-demo-bucket"
sqs_extended_client.use_legacy_attribute = False
```

```
# Create an SQS message queue for this example. Then, extract the queue URL.
queue = sqs_extended_client.create_queue(
    QueueName = "MyQueue"
)
queue_url = sqs_extended_client.get_queue_url(
    QueueName = "MyQueue"
)['QueueUrl']

# Create the S3 bucket and allow message objects to be stored in the bucket.
sqs_extended_client.s3_client.create_bucket(Bucket=sqs_extended_client.large_payload_support)

# Sending a large message
small_message = "s"
large_message = small_message * 300000 # Shall cross the limit of 256 KB

send_message_response = sqs_extended_client.send_message(
    QueueUrl=queue_url,
    MessageBody=large_message
)
assert send_message_response['ResponseMetadata']['HTTPStatusCode'] == 200

# Receiving the large message
receive_message_response = sqs_extended_client.receive_message(
    QueueUrl=queue_url,
    MessageAttributeNames=['All']
)
assert receive_message_response['Messages'][0]['Body'] == large_message
receipt_handle = receive_message_response['Messages'][0]['ReceiptHandle']

# Deleting the large message
# Set to True for deleting the payload from S3
sqs_extended_client.delete_payload_from_s3 = True
delete_message_response = sqs_extended_client.delete_message(
    QueueUrl=queue_url,
    ReceiptHandle=receipt_handle
)

assert delete_message_response['ResponseMetadata']['HTTPStatusCode'] == 200

# Deleting the queue
delete_queue_response = sqs_extended_client.delete_queue(
```

```
        QueueUrl=queue_url
    )

    assert delete_queue_response['ResponseMetadata']['HTTPStatusCode'] == 200
```

Configurar filas do Amazon SQS usando o console do Amazon SQS

Use o console do Amazon SQS para configurar e gerenciar filas e recursos do Amazon SQS. Você também pode:

- Habilite a criptografia do lado do servidor para aumentar a segurança.
- Associe uma fila de mensagens não entregues para lidar com mensagens não processadas.
- Configure um gatilho para invocar uma função do Lambda para processamento orientado por eventos.

Controle de acesso por atributo para o Amazon SQS

O que é ABAC?

O controle de acesso baseado em atributos (ABAC) é um processo de autorização que define permissões com base em tags anexadas a usuários e recursos. AWS O ABAC fornece controle de acesso detalhado e flexível com base em atributos e valores, reduz o risco de segurança relacionado a políticas reconfiguradas baseadas em perfis e centraliza a auditoria e o gerenciamento de políticas de acesso. Para obter mais detalhes sobre o ABAC, consulte [O que é a ABAC para a AWS](#), no Guia do usuário do IAM.

O Amazon SQS é compatível com o ABAC, permitindo que você controle o acesso às filas do Amazon SQS com base em tags e aliases associados a uma fila do Amazon SQS. As chaves de condição de tag e alias que ativam o ABAC no Amazon SQS autorizam as entidades principais a usar as filas do Amazon SQS sem editar políticas ou gerenciar concessões. Para saber mais sobre as chaves de condição do ABAC, consulte [Condition keys for Amazon SQS](#) na Referência de autorização do serviço.

Com o ABAC, é possível usar tags para configurar permissões e políticas de acesso do IAM para as filas do Amazon SQS, o que ajuda você a escalar o gerenciamento de permissões. É possível criar uma única política de permissões no IAM usando tags que você adiciona a cada perfil empresarial, sem precisar atualizar a política toda vez em que adicionar um novo recurso. Você também pode anexar tags às entidades principais do IAM para criar uma política de ABAC. É possível criar políticas de ABAC para permitir operações do Amazon SQS quando a tag no perfil do usuário do IAM que

está fazendo a chamada corresponde à tag de fila do Amazon SQS. Para saber mais sobre a marcação AWS, consulte [Estratégias de AWS marcação e Tags de alocação de custos do Amazon SQS](#)

Note

Atualmente, o ABAC para Amazon SQS está disponível em AWS todas as regiões comerciais em que o Amazon SQS está disponível, com as seguintes exceções:

- Ásia-Pacífico (Hyderabad)
- Ásia-Pacífico (Melbourne)
- Europa (Espanha)
- Europa (Zurique)

Por que devo usar o ABAC no Amazon SQS?

Veja alguns benefícios de usar o ABAC no Amazon SQS:

- O ABAC para o Amazon SQS exige menos políticas de permissões. Não é necessário criar políticas diferentes para funções de trabalho diferentes. É possível usar tags de recurso e solicitação que se aplicam a mais de uma fila, o que reduz as despesas operacionais indiretas.
- Use o ABAC para escalar equipes rapidamente. As permissões para novos recursos são concedidas automaticamente com base em tags quando os recursos são devidamente marcados durante a criação.
- Use as permissões na entidade principal do IAM para restringir o acesso aos recursos. É possível criar tags para a entidade principal do IAM principal e usá-las para restringir o acesso a ações específicas que correspondam às tags na entidade principal do IAM. Isso ajuda você a automatizar o processo de concessão de permissões de solicitação.
- Acompanhe quem está acessando seus recursos. Você pode determinar a identidade de uma sessão examinando os atributos do usuário no AWS CloudTrail.

Tópicos

- [Marcação para controle de acesso no Amazon SQS](#)
- [Criação de usuários do IAM e filas do Amazon SQS](#)
- [Testar o controle de acesso por atributo no Amazon SQS](#)

Marcação para controle de acesso no Amazon SQS

Veja a seguir um exemplo do uso de tags para controle de acesso no Amazon SQS. A política do IAM restringe um usuário do IAM a todas as ações do Amazon SQS para todas as filas que incluem uma tag de recurso com a chave “environment” (ambiente) e o valor “production” (produção). Para obter mais informações, consulte [Controle de acesso baseado em atributos com tags e Organizations AWS](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessForProd",
      "Effect": "Allow",
      "Action": "sqs:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/environment": "prod"
        }
      }
    }
  ]
}
```

Criação de usuários do IAM e filas do Amazon SQS

Os exemplos a seguir explicam como criar uma política ABAC para controlar o acesso ao Amazon SQS usando o Console de gerenciamento da AWS CloudFormation

Usando o Console de gerenciamento da AWS

Criar um usuário do IAM

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. Selecione User (Usuário) no painel de navegação à esquerda.

3. Selecione Add Users (Adicionar usuários) e insira um nome na caixa de texto User name (Nome do usuário).
4. Escolha a caixa Access key - Programmatic access (Chave de acesso – Acesso programático) e selecione Next: Permissions (Próximo: permissões).
5. Selecione Next: Tags (Próximo: tags).
6. Adicione beta como a chave da tag e environment como o valor da tag.
7. Selecione Next: Review (Próximo: revisão) e, depois, Create user (Criar usuário).
8. Copie e armazene o ID da chave de acesso e a chave de acesso secreta em um local seguro.

Adicionar permissões de usuário do IAM

1. Selecione o usuário do IAM que você criou.
2. Escolha Add inline policy (Adicionar política em linha).
3. Na guia JSON, cole a seguinte política:
4. Selecione Revisar política.
5. Selecione Criar política.

Usando AWS CloudFormation

Use o CloudFormation modelo de amostra a seguir para criar um usuário do IAM com uma política embutida anexada e uma fila do Amazon SQS:

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "CloudFormation template to create IAM user with custom inline policy"
Resources:
  IAMPolicy:
    Type: "AWS::IAM::Policy"
    Properties:
      PolicyDocument: |
        {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AllowAccessForSameResTag",
              "Effect": "Allow",
              "Action": [
                "sqs:SendMessage",
```

```

        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/environment": "${aws:PrincipalTag/
environment}"
        }
    }
},
{
    "Sid": "AllowAccessForSameReqTag",
    "Effect": "Allow",
    "Action": [
        "sqs:CreateQueue",
        "sqs:DeleteQueue",
        "sqs:SetQueueAttributes",
        "sqs:tagqueue"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/environment": "${aws:PrincipalTag/
environment}"
        }
    }
},
{
    "Sid": "DenyAccessForProd",
    "Effect": "Deny",
    "Action": "sqs:*",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/stage": "prod"
        }
    }
}
]
}

Users:
- "testUser"

```

```
PolicyName: tagQueuePolicy

IAMUser:
  Type: "AWS::IAM::User"
  Properties:
    Path: "/"
    Username: "testUser"
    Tags:
      -
        Key: "environment"
        Value: "beta"
```

Testar o controle de acesso por atributo no Amazon SQS

Os exemplos a seguir mostram como testar o controle de acesso por atributo no Amazon SQS.

Crie uma fila com a chave da tag definida como “ambiente” e o valor da tag definido como “prod”

Execute esse comando da AWS CLI para testar a criação da fila com a chave de tag definida como environment e o valor da tag definido como prod. Se você não tiver a AWS CLI, poderá [baixá-la e configurá-la](#) para sua máquina.

```
aws sqs create-queue --queue-name prodQueue --region us-east-1 --tags "environment=prod"
```

Você recebe um erro AccessDenied do endpoint do Amazon SQS:

```
An error occurred (AccessDenied) when calling the CreateQueue operation: Access to the
resource <queueUrl> is denied.
```

Isso ocorre porque o valor da tag no usuário do IAM não corresponde à tag transmitida na chamada de API [CreateQueue](#). Lembre-se de que aplicamos uma tag ao usuário do IAM com a chave definida como environment e o valor definido como beta.

Crie uma fila com a chave da tag definida como “environment” e o valor da tag definido como “beta”

Execute esse comando da CLI para testar a criação de uma fila com a chave da tag definida como environment e o valor da tag definido como beta.

```
aws sqs create-queue --queue-name betaQueue --region us-east-1 --tags "environment=beta"
```

Você recebe uma mensagem confirmando a criação bem-sucedida da fila, semelhante à indicada abaixo.

```
{
  "QueueUrl": "<queueUrl>"
}
```

Enviar uma mensagem para uma fila

Execute esse comando da CLI para testar o envio de uma mensagem a uma fila.

```
aws sqs send-message --queue-url <queueUrl> --message-body testMessage
```

A resposta mostra uma entrega bem-sucedida de mensagens para a fila do Amazon SQS. A permissão de usuário do IAM permite que você envie uma mensagem para uma fila que tenha uma tag beta. A resposta inclui MD5fMessageBody e MessageId contendo a mensagem.

```
{
  "MD5fMessageBody": "<MD5fMessageBody>",
  "MessageId": "<MessageId>"
}
```

Configurar parâmetros de filas usando o console do Amazon SQS

Ao [criar](#) ou [editar](#) uma fila, você pode configurar os seguintes parâmetros:

- Visibility timeout (Tempo limite de visibilidade): o período de tempo em que uma mensagem recebida de uma fila (por um consumidor) não estará visível para os outros consumidores de mensagens. Para obter mais informações, consulte [Tempo limite de visibilidade](#).

Note

Usar o console para definir o tempo limite de visibilidade configura o valor de tempo limite para todas as mensagens na fila. Para configurar o tempo limite para uma ou várias mensagens, você deve usar um dos SDKs da AWS.

- **Message retention period (Período de retenção de mensagens):** a quantidade de tempo que o Amazon SQS retém as mensagens que permanecem na fila. Por padrão, uma fila retém mensagens por quatro dias. Você pode configurar uma fila para reter as mensagens por até 14 dias. Para obter mais informações, consulte [Período de retenção de mensagens](#).
- **Delivery delay (Atraso de entrega):** quanto tempo o Amazon SQS atrasará antes de enviar uma mensagem adicionada à fila. Para obter mais informações, consulte [Atraso de entrega](#).
- **Maximum message size (Tamanho máximo da mensagem):** tamanho máximo das mensagens para essa fila. Para obter mais informações, consulte [Tamanho máximo da mensagem](#).
- **Receive message wait time (Tempo de espera da mensagem):** a quantidade máxima de tempo que o Amazon SQS espera para que as mensagens fiquem disponíveis depois que a fila recebe uma solicitação de recebimento. Para obter mais informações, consulte [Sondagem curta e longa do Amazon SQS](#).
- **Enable content-based deduplication (Habilitar a eliminação de duplicação baseada em conteúdo):** o Amazon SQS pode criar automaticamente IDs de eliminação de duplicação com base no corpo da mensagem. Para obter mais informações, consulte [Filas FIFO do Amazon SQS](#).
- **Enable high throughput FIFO (Habilitar FIFO de alta taxa de transferência):** use para habilitar a alta taxa de transferência disponível para mensagens na fila. Escolher esta opção altera as opções relacionadas ([Deduplication scope](#) [Escopo de eliminação de duplicação] e [FIFO throughput limit](#) [Limite de transferência FIFO]) para as configurações necessárias a fim de habilitar a alta taxa de transferência para filas FIFO. Para obter mais informações, consulte [Throughput alto para filas FIFO no Amazon SQS](#) e [Cotas de mensagens do Amazon SQS](#).
- **Redrive allow policy (Política de permissão de redirecionamento):** define quais filas de origem podem usar essa fila como a fila de mensagens mortas. Para obter mais informações, consulte [Usar filas de mensagens não entregues no Amazon SQS](#).

Para configurar os parâmetros de uma fila existente (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues. Escolha uma fila e escolha Edit (Editar).
3. Role até a seção Configuration (Configuração).
4. Em Visibility timeout (Tempo limite de visibilidade), insira a duração e as unidades. O intervalo é de 0 segundo a 12 horas. O valor padrão é 30 segundos.
5. Em Message retention period (Período de retenção de mensagens), insira a duração e as unidades. O intervalo é de 1 minuto a 14 dias. O valor padrão é 4 dias.

6. Em uma fila padrão, insira um valor para Receive message wait time (Tempo de espera da mensagem). O intervalo é de 0 a 20 segundos. O valor padrão é 0 segundo, o que define uma [sondagem curta](#). Qualquer valor diferente de zero define uma sondagem longa.
7. Em Delivery delay (Atraso de entrega), insira a duração e as unidades. O intervalo é de 0 segundo a 15 minutos. O valor de padrão é 0 segundos.
8. Em Maximum message size (Tamanho máximo da mensagem), insira um valor. O intervalo é de 1 KiB a 1024 KiB. O valor padrão é 1024 KiB.
9. Para uma fila FIFO, escolha Enable content-based deduplication (Habilitar a eliminação de duplicação baseada em conteúdo) para habilitar a eliminação de duplicação baseada em conteúdo. Por padrão, essa configuração está desabilitada.
10. (Opcional) Em uma fila FIFO, para habilitar um throughput mais alto a fim de enviar e receber mensagens na fila, escolha Enable high throughput FIFO (Habilitar FIFO de alto throughput).

Escolher esta opção altera as opções relacionadas (Deduplication scope [Escopo de eliminação de duplicação] e FIFO throughput limit [Limite de transferência FIFO]) para as configurações necessárias a fim de habilitar a alta taxa de transferência para filas FIFO. Se você alterar qualquer uma das configurações necessárias para usar FIFO de alta taxa de transferência, a taxa de transferência normal permanecerá em vigor para a fila e a eliminação de duplicação ocorrerá conforme especificado. Para obter mais informações, consulte [Throughput alto para filas FIFO no Amazon SQS](#) e [Cotas de mensagens do Amazon SQS](#).

11. Em Redrive allow policy (Política de permissão de redirecionamento), escolha Enabled (Habilitada). Selecione uma das seguintes opções: Allow all (Permitir tudo) (padrão), By queue (Por fila) ou Deny all (Negar tudo). Ao escolher By queue (Por fila), especifique uma lista de até 10 filas de origem pelo nome do recurso da Amazon (ARN).
12. Quando você terminar de configurar os parâmetros da fila, escolha Save (Salvar).

Configurar uma política de acesso no Amazon SQS

Ao [editar](#) uma fila, você pode configurar a política de acesso para controlar quem pode interagir com ela.

- A política de acesso define quais contas, usuários e perfis têm permissão para acessar a fila.
- Ela especifica as ações permitidas, como [SendMessage](#), [ReceiveMessage](#) ou [DeleteMessage](#).
- Por padrão, apenas o proprietário da fila tem permissão para enviar e receber mensagens.

Para configurar a política de acesso para uma fila existente (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Escolha uma fila e escolha Edit (Editar).
4. Role até a seção Access policy (Política de acesso).
5. Edite as instruções da política de acesso na caixa de entrada. Para obter mais informações sobre instruções da política de acesso, consulte [Gerenciamento de identidade e acesso no Amazon SQS](#).
6. Quando terminar de configurar a política de acesso, escolha Save (Salvar).

Configurar a criptografia do lado do servidor para uma fila usando chaves de criptografia gerenciadas pelo SQS

Além da opção [padrão](#) da criptografia do lado do servidor (SSE) gerenciada pelo Amazon SQS, a SSE gerenciada pelo Amazon SQS (SSE-SQS) permite criar uma criptografia do lado do servidor gerenciada pelo cliente que usa chaves de criptografia gerenciadas pelo SQS para proteger dados sigilosos enviados por filas de mensagens. Com o SSE-SQS, você não precisa criar e gerenciar chaves de criptografia ou modificar seu código para criptografar seus dados. O SSE-SQS permite transmitir dados com segurança e ajuda a atender a requisitos regulamentares e conformidade com criptografia rigorosa sem custo adicional.

O SSE-SQS protege os dados em repouso usando a criptografia Advanced Encryption Standard (AES-256) de 256 bits. A SSE criptografa mensagens assim que o Amazon SQS as recebe. O Amazon SQS armazena as mensagens no formato criptografado e as descriptografa apenas quando elas são enviadas a um consumidor autorizado.

Note

- A opção de SSE comum só é efetiva quando você cria uma fila sem especificar atributos de criptografia.
- O Amazon SQS permite que você desative toda a criptografia de filas. Portanto, desativar a KMS-SSE não habilitará automaticamente a SQS-SSE. Se quiser habilitar a SQS-SSE depois de desativar a KMS-SSE, você deverá adicionar uma alteração de atributos na solicitação.

Para configurar a criptografia SSE-SQS para uma fila (console)

Note

Qualquer fila criada usando o endpoint HTTP (não TLS) não habilitará a criptografia SSE-SQS por padrão. É uma prática recomendada de segurança criar filas do Amazon SQS usando endpoints HTTPS ou do [Signature versão 4](#).

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Escolha uma fila e escolha Edit (Editar).
4. Expanda Encryption (Criptografia).
5. Em Server-side encryption (Criptografia do lado do servidor), escolha Enabled (Habilitado) (padrão).

Note

Com a SSE habilitada, as solicitações anônimas SendMessage e ReceiveMessage à fila criptografada serão rejeitadas. As práticas recomendadas de segurança do Amazon SQS não aconselham o uso de solicitações anônimas. Se você quiser enviar solicitações anônimas a uma fila do Amazon SQS, desabilite o SSE.

6. Selecione Amazon SQS key (SSE-SQS) (Chave do Amazon SQS (SSE-SQS)). Não há custo adicional para usar essa opção.
7. Escolha Salvar.

Configurar a criptografia do lado do servidor para uma fila usando o console do Amazon SQS

Para proteger os dados nas mensagens de uma fila, o Amazon SQS tem a criptografia do lado do servidor (SSE) habilitada por padrão para todas as filas recém-criadas. O Amazon SQS integra-se ao Amazon Web Services Key Management Service (Amazon Web Services KMS) para gerenciar [chaves do KMS](#) para criptografia do lado do servidor (SSE). Para obter mais informações sobre o uso de SSE, consulte [Criptografia em repouso no Amazon SQS](#).

A chave do KMS que você atribui à fila deve ter uma política de chave que inclua permissões para todas as entidades autorizadas a usar a fila. Para obter mais informações, consulte [Gerenciamento de chaves](#).

Se você não for o proprietário da chave do KMS ou se fizer login com uma conta que não tenha as permissões `kms:ListAliases` e `kms:DescribeKey`, não será possível visualizar as informações sobre a chave do KMS no console do Amazon SQS. Peça ao proprietário da chave do KMS para conceder essas permissões a você. Para obter mais informações, consulte [Gerenciamento de chaves](#).

Quando você [cria](#) ou [edita](#) uma fila, pode configurar a SSE-KMS.

Para configurar a SSE-KMS para uma fila existente (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Escolha uma fila e escolha Edit (Editar).
4. Expanda Encryption (Criptografia).
5. Em Server-side encryption (Criptografia do lado do servidor), escolha Enabled (Habilitado) (padrão).

 Note

Com a SSE habilitada, as solicitações anônimas `SendMessage` e `ReceiveMessage` à fila criptografada serão rejeitadas. As práticas recomendadas de segurança do Amazon SQS não aconselham o uso de solicitações anônimas. Se você quiser enviar solicitações anônimas a uma fila do Amazon SQS, desabilite o SSE.

6. Selecione AWS Key Management Service key (SSE-KMS) (Chave do serviço de gerenciamento de chaves (SSE-KMS)).

O console exibe a Descrição, a Conta e o ARN da chave do KMS da chave do KMS.

7. Especifique o ID da chave do KMS para a fila. Para obter mais informações, consulte [Principais termos](#).
 - a. Escolha a opção Choose a KMS key alias (Escolha um alias para a chave do KMS).
 - b. A chave padrão é a chave do KMS gerenciada pela Amazon Web Services para o Amazon SQS. Para usar essa chave, escolha-a na lista KMS key (Chave do KMS).

- c. Para usar uma chave do KMS personalizada de sua conta do Amazon Web Services, escolha-a na lista KMS key (Chave do KMS). Para obter instruções sobre como criar chaves do KMS personalizadas, consulte [Creating Keys](#) (Criar chaves) no Amazon Web Services Key Management Service Developer Guide (Guia do desenvolvedor do serviço de gerenciamento de chaves do Amazon Web Services).
 - d. Para usar uma chave do KMS personalizada que não esteja na lista ou uma chave do KMS personalizada de outra conta do Amazon Web Services, escolha Enter the KMS key alias (Inserir o alias da chave do KMS) e insira o nome do recurso da Amazon (ARN) da chave do KMS.
8. (Opcional) Em Data key reuse period (Período de reutilização de chaves de dados), especifique um valor entre 1 minuto e 24 horas. O padrão é 5 minutos. Para obter mais informações, consulte [Entender o período de reutilização de chaves de dados](#).
 9. Quando terminar de configurar a SSE-KMS, escolha Save (Salvar).

Configurar tags de alocação de custos para uma fila usando o console do Amazon SQS

Para organizar e identificar suas filas do Amazon SQS, você pode adicionar tags de alocação de custos. Para obter mais informações, consulte [Tags de alocação de custos do Amazon SQS](#).

- A guia Marcação na página Detalhes exibe as tags da fila.
- Você pode adicionar ou modificar tags ao [criar](#) ou [editar](#) uma fila.

Para configurar tags para uma fila existente (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Escolha uma fila e escolha Edit (Editar).
4. Role até a seção Tags.
5. Adicionar, modificar ou remover tags da fila:
 - a. Para adicionar uma tag, escolha Add new tag (Adicionar nova tag), insira uma Key (Chave) e um Value (Valor) e, em seguida, escolha Add new tag (Adicionar nova tag).
 - b. Para atualizar uma tag, altere sua Key (Chave) e Value (Valor).

- c. Para remover uma tag, escolha Remove tag (Remover tag) ao lado de um par chave-valor.
6. Ao terminar de configurar as tags, escolha Save (Salvar).

Inscriver uma fila em um tópico do Amazon SNS usando o console do Amazon SQS

Você pode inscrever uma ou mais filas do Amazon SQS em um tópico do Amazon SNS. Quando você publica uma mensagem em um tópico, o Amazon SNS envia a mensagem para cada fila inscrita. O Amazon SQS gerencia a assinatura e gerencia as permissões necessárias. Para ter mais informações sobre o Amazon SNS, consulte [O que é o Amazon SNS?](#) no Guia do desenvolvedor do Amazon Simple Notification Service.

Quando você inscreve uma fila do Amazon SQS em um tópico do Amazon SNS, o Amazon SNS usa HTTPS para encaminhar mensagens ao Amazon SQS. Para obter informações sobre como usar o Amazon SNS com filas criptografadas do Amazon SQS, consulte [Configurar permissões do KMS para serviços AWS](#).

Important

O Amazon SQS é compatível com até 20 instruções para cada política de acesso. Assinar um tópico do Amazon SNS adiciona uma instrução desse tipo. Exceder esse valor causará uma falha na entrega da assinatura do tópico.

Como inscrever uma fila em um tópico do Amazon SNS (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Na lista de filas, escolha a fila que você deseja assinar em um tópico do Amazon SNS.
4. Em Actions (Ações), escolha Subscribe to Amazon SNS topic (Inscrever-se no tópico do Amazon SNS).
5. No menu Especificar um tópico do Amazon SNS disponível para esta fila, escolha o tópico do Amazon SNS para a fila.

Se o tópico do SNS não estiver listado, escolha Inserir o ARN do tópico do Amazon SNS e insira o nome do recurso da Amazon (ARN) do tópico.

6. Escolha Salvar.
7. Para verificar a assinatura, publique uma mensagem no tópico e visualize a mensagem na fila. Para ter mais informações, consulte [Publicação de mensagens do Amazon SNS](#) no Guia do desenvolvedor do Amazon Simple Notification Service.

Assinaturas entre contas

Se a fila do Amazon SQS e o tópico do Amazon SNS estiverem em diferentes Contas da AWS, permissões adicionais serão necessárias.

Proprietário do tópico (Conta A)

Modifique a política de acesso do tópico do Amazon SNS para permitir que a fila Conta da AWS do Amazon SQS se inscreva. Exemplo de instrução de política:

```
{
  "Effect": "Allow",
  "Principal": { "AWS": "arn:aws:iam::111122223333:root" },
  "Action": "sns:Subscribe",
  "Resource": "arn:aws:sns:us-east-1:123456789012:MyTopic"
}
```

Esta política permite que a conta 111122223333 se inscreva em MyTopic.

Proprietário da fila (Conta B)

Modifique a política de acesso da fila do Amazon SQS para permitir que o tópico do Amazon SNS envie mensagens. Exemplo de instrução de política:

```
{
  "Effect": "Allow",
  "Principal": { "Service": "sns.amazonaws.com" },
  "Action": "sqs:SendMessage",
  "Resource": "arn:aws:sqs:us-east-1:111122223333:MyQueue",
  "Condition": {
    "ArnEquals": { "aws:SourceArn": "arn:aws:sns:us-east-1:123456789012:MyTopic" }
  }
}
```

Essa política permite que o MyTopic envie mensagens para MyQueue.

Assinaturas entre regiões

Para assinar um tópico do Amazon SNS em outra Região da AWS, certifique-se de que:

- A política de acesso ao tópico do Amazon SNS permita assinaturas entre regiões.
- A política de acesso da fila do Amazon SQS permita que o tópico do Amazon SNS envie mensagens entre regiões.

Para obter mais informações, consulte [Envio de mensagens do Amazon SNS para uma fila do Amazon SQS ou função do AWS Lambda no Guia do desenvolvedor do Amazon Simple Notification Service](#).

Configurar uma fila do Amazon SQS para acionar uma função do AWS Lambda

Você pode usar uma função do Lambda para processar mensagens de uma fila do Amazon SQS. O Lambda sonda a fila e invoca sua função de forma síncrona, transmitindo um lote de mensagens como um evento.

Configuração do tempo limite de visibilidade

Defina o tempo limite de visibilidade da fila para pelo menos seis vezes o valor do [tempo limite da função](#). Isso garante que o Lambda tenha tempo o suficiente para tentar novamente se uma função estiver travada em um lote anterior.

Como usar uma fila de mensagens não entregues (DLQ)

Especifique uma fila de mensagens não entregues para capturar mensagens que a função do Lambda não conseguiu processar.

Como gerenciar várias filas e funções

Uma função do Lambda pode processar várias filas criando uma origem de eventos separada para cada fila. É possível associar várias funções do Lambda à mesma fila.

Permissões para filas criptografadas

Se você associar uma fila criptografada a uma função Lambda, mas o Lambda não sondar as mensagens, adicione a permissão `kms:Decrypt` para sua função de execução do Lambda.

Restrições

A fila e a função do Lambda devem estar na mesma Região da AWS.

Uma [fila criptografada](#) que usa a chave padrão (chave do KMS gerenciada pela AWS para Amazon SQS) não pode invocar uma função Lambda em outra Conta da AWS.

Para obter detalhes de implementação, consulte [Como usar o AWS Lambda com o Amazon SQS](#) no Guia do desenvolvedor do AWS Lambda.

Pré-requisitos

Para configurar os acionadores de função Lambda, você deve atender aos seguintes requisitos:

- Se você usar um usuário, o perfil do Amazon SQS deverá incluir as seguintes permissões:
 - `lambda:CreateEventSourceMapping`
 - `lambda:ListEventSourceMappings`
 - `lambda:ListFunctions`
- A função de execução do Lambda deve incluir as seguintes permissões:
 - `sqs:DeleteMessage`
 - `sqs:GetQueueAttributes`
 - `sqs:ReceiveMessage`
- Se você associar uma fila criptografada a uma função Lambda, adicione a permissão `kms:Decrypt` à função de execução do Lambda.

Para obter mais informações, consulte [Visão geral do gerenciamento de acesso no Amazon SQS](#).

Para configurar uma fila para acionar uma função Lambda (console)

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.
3. Na página Queues (Filas), escolha a fila a ser configurada.
4. Na página da fila, escolha a guia Lambda triggers (Acionadores do Lambda).
5. Na página Lambda triggers (Acionadores do Lambda), escolha um acionador do Lambda.

Se a lista não incluir o acionador do Lambda de que você precisa, escolha Configure Lambda function trigger (Configurar acionador da função Lambda). Insira o nome do recurso da Amazon (ARN) da função Lambda ou escolha um recurso existente. Em seguida, escolha Salvar.

6. Escolha Salvar. O console salva a configuração e exibe a página Details (Detalhes) da fila.

Na página Details (Detalhes), a guia Lambda triggers (Acionadores do Lambda) exibe a função Lambda e seu status. Demora aproximadamente um minuto para a função Lambda se associar à sua fila.

7. Para verificar os resultados da configuração, você pode [enviar uma mensagem à fila](#) e, em seguida, visualizar a função Lambda acionada no console do Lambda.

Automação de notificações de produtos da AWS para o Amazon SQS usando o Amazon EventBridge

O Amazon EventBridge permite que você automatize os Serviços da AWS e responda a eventos, como problemas de aplicações ou alterações de recursos, quase em tempo real.

- Você pode criar regras para filtrar eventos específicos e definir ações automatizadas quando uma regra corresponder a um evento.
- O EventBridge é compatível com vários destinos, incluindo o Amazon SQS padrão e as filas FIFO, que recebem os eventos no formato JSON.

Para obter mais informações, consulte [Destinos do Amazon EventBridge](#) no [Guia do usuário do Amazon EventBridge](#).

Enviar uma mensagem com atributos usando o Amazon SQS

Para filas FIFO e padrão, é possível incluir metadados estruturados em mensagens, incluindo carimbos de data e hora, dados geoespaciais, assinaturas e identificadores. Para obter mais informações, consulte [Atributos de mensagem do Amazon SQS](#).

Como enviar uma mensagem com atributos a uma fila usando o console do Amazon SQS

1. Abra o console do Amazon SQS em <https://console.aws.amazon.com/sqs/>.
2. No painel de navegação, escolha Queues.

3. Na página Queues (Filas), escolha uma fila.
4. Escolha Enviar e receber mensagens.
5. Insira os parâmetros do atributo de mensagem.
 - a. Na caixa de texto de nome, insira um nome exclusivo de até 256 caracteres.
 - b. Para o tipo de atributo, escolha String, Number (Número) ou Binary (Binário).
 - c. (Opcional) Insira um tipo de dado personalizado. Por exemplo, você pode adicionar **byte**, **int** ou **float** como tipos de dados personalizados para Number (Número).
 - d. Na caixa de texto do valor, insira o valor do atributo de mensagem.

The screenshot shows a dialog box titled "Message attributes - Optional" with an "Info" link. It contains a single input field for "Enter name", a dropdown menu set to "String", a "Custom type" input field, and a "Enter value" input field. Below these fields is a button labeled "Add new attribute".

6. Para adicionar outro atributo de mensagem, escolha Add new attribute (Adicionar um atributo).

The screenshot shows the same dialog box as before, but now it contains two identical input fields for "Enter name", "String" dropdown, "Custom type", and "Enter value". A "Remove" button has been added to the right of the second input field. The "Add new attribute" button remains at the bottom.

7. Você pode modificar os valores do atributo a qualquer momento antes de enviar a mensagem.
8. Para excluir um atributo, escolha Remove (Remover). Para excluir o primeiro atributo, feche Message attributes (Atributos de mensagem).
9. Ao concluir a adição de atributos à mensagem, escolha Send Message (Enviar mensagem). Sua mensagem é enviada e o console exibe uma mensagem de sucesso. Para visualizar informações sobre os atributos da mensagem enviada, escolha View details (Visualizar detalhes). Escolha Done (Concluído) para fechar a caixa de diálogo Message details (Detalhes da mensagem).

Práticas recomendadas do Amazon SQS

O Amazon SQS gerencia e processa filas de mensagens, permitindo que diferentes partes de uma aplicação troquem mensagens de forma confiável e escalável. Este tópico aborda as práticas recomendadas operacionais essenciais, incluindo o uso de sondagens longas a fim de reduzir respostas vazias, a implementação de filas de mensagens não entregues para gerenciar erros de processamento e a otimização de permissões de fila para fins de segurança.

Tópicos

- [Gerenciamento de erros e mensagens problemáticas do Amazon SQS](#)
- [Desduplicação e agrupamento de mensagens do Amazon SQS](#)
- [Tempo e processamento de mensagens do Amazon SQS](#)

Gerenciamento de erros e mensagens problemáticas do Amazon SQS

Este tópico fornece instruções detalhadas sobre como gerenciar e mitigar erros no Amazon SQS, incluindo técnicas para lidar com erros de solicitação, capturar mensagens problemáticas e configurar a retenção da fila de mensagens não entregues para garantir a confiabilidade das mensagens.

Tópicos

- [Gerenciamento de erros de solicitação no Amazon SQS](#)
- [Capturar mensagens problemáticas no Amazon SQS](#)
- [Configurar a retenção da fila de mensagens não entregues no Amazon SQS](#)

Gerenciamento de erros de solicitação no Amazon SQS

Para gerenciar erros de solicitação, use uma das seguintes estratégias:

- Se você usa um AWS SDK, já tem a lógica de repetição e recuo automáticos à sua disposição. Para ter mais informações, consulte [Repetições de erro e recuo exponencial na AWS](#) no Referência geral da Amazon Web Services.

- Se você não usar os recursos do AWS SDK para repetição e recuo, permita uma pausa (por exemplo, 200 ms) antes de tentar novamente a ação [ReceiveMessage](#) após receber nenhuma mensagem, um tempo limite ou uma mensagem de erro do Amazon SQS. Para o uso subsequente de `ReceiveMessage` que oferece os mesmos resultados, faça uma pausa maior (por exemplo, 400 ms).

Capturar mensagens problemáticas no Amazon SQS

Para capturar todas as mensagens que não podem ser processadas e coletar métricas do CloudWatch precisas, configure uma [fila de mensagens mortas](#).

- A política de redirecionamento redireciona mensagens para uma dead letter queue depois que a fila de origem falha em processar uma mensagem um número de vezes especificado.
- O uso da dead letter queue diminui o número de mensagens e reduz a possibilidade de exposição a mensagens poison pill (mensagens que podem ser recebidas, mas que não podem ser processadas).
- A inclusão de uma mensagem poison pill em uma fila pode distorcer a métrica do CloudWatch [ApproximateAgeOfOldestMessage](#), fornecendo uma idade incorreta da mensagem poison pill. Configurar uma dead letter queue ajuda a evitar alarmes falsos ao usar essa métrica.

Configurar a retenção da fila de mensagens não entregues no Amazon SQS

Para filas comuns, a validade de uma mensagem é sempre baseada em seu carimbo de data/hora de enfileiramento original. Quando uma mensagem é movida para uma fila de mensagens mortas, o carimbo de data/hora de enfileiramento permanece inalterado. A métrica `ApproximateAgeOfOldestMessage` indica quando a mensagem foi movida para a fila de mensagens não entregues, não quando a mensagem foi originalmente enviada. Por exemplo, suponha que uma mensagem fique um dia na fila original antes de ser movida para uma fila de mensagens mortas. Se o período de retenção da fila de mensagens mortas for de quatro dias, a mensagem será excluída da fila de mensagens mortas após três dias e a `ApproximateAgeOfOldestMessage` será de três dias. Portanto, é uma prática recomendada definir sempre o período de retenção de uma fila de mensagens mortas para ser maior do que o período de retenção da fila original.

Para filas FIFO, o carimbo de data/hora da fila é redefinido quando a mensagem é movida para uma fila de mensagens não entregues. A métrica `ApproximateAgeOfOldestMessage` indica quando a mensagem foi movida para a fila de mensagens não entregues. No mesmo exemplo acima, a mensagem é excluída da fila de mensagens não entregues após quatro dias e `ApproximateAgeOfOldestMessage` é de quatro dias.

Desduplicação e agrupamento de mensagens do Amazon SQS

Este tópico fornece as práticas recomendadas para garantir o processamento consistente de mensagens no Amazon SQS. Aqui, você encontrará explicações para como usar:

- [MessageDeduplicationId](#) para evitar mensagens duplicadas nas filas FIFO.
- [MessageGroupId](#) para gerenciar a ordenação de mensagens em grupos de mensagens distintos.

Tópicos

- [Evitar o processamento inconsistente de mensagens no Amazon SQS](#)
- [Usar o ID de desduplicação de mensagens do Amazon SQS](#)
- [Usar o ID do grupo de mensagens com filas FIFO do Amazon SQS](#)
- [Usar o ID de tentativa de solicitação de recebimento do Amazon SQS](#)

Evitar o processamento inconsistente de mensagens no Amazon SQS

Como o Amazon SQS é um sistema distribuído, é possível que um consumidor não receba uma mensagem mesmo quando o Amazon SQS a marcar como entregue ao retornar com êxito de uma chamada de método da API [ReceiveMessage](#). Nesse caso, o Amazon SQS registra a mensagem como entregue pelo menos uma vez, embora o consumidor nunca a tenha recebido. Como nenhuma tentativa adicional de entregar mensagens é feita sob essas condições, não recomendamos definir o número máximo de recebimentos como 1 para uma [fila de mensagens mortas](#).

Usar o ID de desduplicação de mensagens do Amazon SQS

[MessageDeduplicationId](#) é um token usado apenas em filas FIFO do Amazon SQS para evitar a entrega duplicada de mensagens. Ele garante que, dentro de um intervalo de desduplicação de cinco minutos, somente uma instância de uma mensagem com o mesmo ID de desduplicação seja processada e entregue.

Se o Amazon SQS já aceitou uma mensagem com um ID específico de deduplicação, todas as mensagens subsequentes com o mesmo ID serão reconhecidas, mas não serão entregues aos consumidores.

 Note

O Amazon SQS continua acompanhando o ID de deduplicação da mensagem mesmo depois que a mensagem é recebida e excluída.

Tópicos

- [Quando fornecer o ID de deduplicação de mensagens no Amazon SQS](#)
- [Habilitar a deduplicação para um sistema de produtor/consumidor único no Amazon SQS](#)
- [Cenários de recuperação de interrupção no Amazon SQS](#)
- [Como configurar tempo limite de visibilidade no Amazon SQS](#)

Quando fornecer o ID de deduplicação de mensagens no Amazon SQS

O produtor deve especificar um ID de deduplicação de mensagens nos seguintes cenários:

- Ao enviar corpos de mensagens idênticos que devem ser tratados como exclusivos.
- Ao enviar mensagens com o mesmo conteúdo, mas com atributos de mensagem diferentes, garantindo que cada mensagem seja processada separadamente.
- Ao enviar mensagens com conteúdo diferente (por exemplo, um contador de repetições no corpo da mensagem), mas solicitando que o Amazon SQS as reconheça como duplicações.

Habilitar a deduplicação para um sistema de produtor/consumidor único no Amazon SQS

Se você tiver um único produtor e um único consumidor, e as mensagens forem exclusivas porque incluem um ID de mensagem específico da aplicação incluído no corpo, siga as práticas recomendadas a seguir:

- Ative a eliminação de duplicação baseada em conteúdo para a fila (cada uma de suas mensagens tem um único corpo). O produtor pode omitir o ID de eliminação de duplicação de mensagem.

- Quando a deduplicação baseada em conteúdo é habilitada para uma fila FIFO do Amazon SQS e uma mensagem é enviada com um ID de deduplicação, o ID de deduplicação [SendMessage](#) substitui o ID de deduplicação baseado no conteúdo gerado.
- Embora o consumidor não seja obrigado a fornecer um ID de tentativa de solicitação de recebimento, isso é uma prática recomendada porque permite que sequências de tentativa de recuperação de falhas sejam executadas mais rapidamente.
- Você pode tentar enviar ou receber solicitações novamente, porque elas não interferem na ordenação de mensagens em filas FIFO.

Cenários de recuperação de interrupção no Amazon SQS

O processo de eliminação de duplicação em filas FIFO é dependente do tempo. Ao projetar sua aplicação, garanta que o produtor e o consumidor possam se recuperar de interrupções de cliente ou da rede sem introduzir duplicações ou falhas de processamento.

Considerações sobre o produtor

- O Amazon SQS impõe uma janela de deduplicação de cinco minutos.
- Se um produtor tentar novamente uma solicitação [SendMessage](#) após cinco minutos, o Amazon SQS a tratará como uma nova mensagem, potencialmente criando duplicatas.

Considerações sobre o consumidor

- Se o consumidor não processar uma mensagem antes que o tempo limite de visibilidade expire, outro consumidor poderá recebê-la e processá-la, resultando em um processamento duplicado.
- Ajuste o tempo limite de visibilidade com base no tempo de processamento da sua aplicação.
- Use a API [ChangeMessageVisibility](#) para estender o tempo limite enquanto uma mensagem ainda está sendo processada.
- Se uma mensagem falhar repetidamente no processamento, encaminhe-a para uma [fila de mensagens não entregues \(DLQ\)](#) em vez de permitir que ela seja reprocessada indefinidamente.
- O produtor deve estar ciente do intervalo de eliminação de duplicação da fila. O Amazon SQS tem um intervalo de eliminação de duplicação de cinco minutos. Repetir solicitações [SendMessage](#) após a expiração do intervalo de eliminação de duplicação pode introduzir mensagens duplicadas na fila. Por exemplo, um dispositivo móvel em um carro envia mensagens cuja ordem é importante.

Se o carro perder a conectividade celular por um período antes de receber uma confirmação, tentar novamente a solicitação depois de recuperada a conectividade celular pode criar uma duplicação.

- O consumidor deve ter um tempo limite de visibilidade que minimize o risco de não conseguir processar as mensagens antes que o tempo limite de visibilidade expire. Você pode estender o tempo limite de visibilidade enquanto as mensagens estão sendo processadas chamando a ação `ChangeMessageVisibility`. No entanto, se o tempo limite de visibilidade expirar, outro consumidor poderá começar imediatamente a processar as mensagens, fazendo com que uma mensagem seja processada várias vezes. Para evitar essa situação, configure uma [dead letter queue](#).

Como configurar tempo limite de visibilidade no Amazon SQS

Para garantir o processamento confiável de mensagens, defina o tempo limite de visibilidade como um valor maior do que o tempo limite de leitura do AWS SDK. Isso se aplica ao usar a ação de API [ReceiveMessage](#) com sondagem curta e sondagem longa. Um tempo limite de visibilidade maior impede que as mensagens sejam disponibilizadas para outros consumidores antes que a solicitação original seja concluída, reduzindo o risco de processamento duplicado.

Usar o ID do grupo de mensagens com filas FIFO do Amazon SQS

Em filas FIFO (First-In First-Out — primeira a entrar, primeira a sair), [MessageGroupId](#) é um atributo que organiza mensagens em grupos distintos. As mensagens dentro do mesmo grupo são sempre processadas uma por vez, em ordem estrita, garantindo que duas mensagens do mesmo grupo não sejam processadas simultaneamente. Em filas padrão, o uso de `MessageGroupId` ativa [filas justas](#). Se for necessária uma ordenação estrita, use uma fila FIFO.

Tópicos

- [Intercalar vários grupos de mensagens ordenadas no Amazon SQS](#)
- [Como evitar o processamento de duplicatas em um sistema de vários produtores/consumidores no Amazon SQS](#)
- [Como evitar um grande acúmulo de mensagens com o mesmo ID de grupo de mensagens no Amazon SQS](#)
- [Evitar reutilizar o mesmo ID de grupo de mensagens com filas virtuais no Amazon SQS](#)

Intercalar vários grupos de mensagens ordenadas no Amazon SQS

Para intercalar vários grupos de mensagens ordenadas em uma única fila FIFO, atribua um [MessageGroupId](#) para cada grupo (por exemplo, dados de sessão de diferentes usuários). Isso permite que vários consumidores leiam da fila simultaneamente, ao mesmo tempo em que garante que as mensagens dentro do mesmo grupo sejam processadas em ordem.

Quando uma mensagem com um `MessageGroupId` específico estiver sendo processada e ficar invisível, nenhum outro consumidor poderá processar mensagens desse mesmo grupo até que o tempo limite de visibilidade expire ou a mensagem seja excluída.

Como evitar o processamento de duplicatas em um sistema de vários produtores/consumidores no Amazon SQS

Em um sistema de alto throughput e baixa latência em que a ordenação das mensagens não é uma prioridade, os produtores podem atribuir um [MessageGroupId](#) exclusivo para cada mensagem. Isso garante que as filas FIFO do Amazon SQS eliminem duplicatas, mesmo em uma configuração de vários produtores/vários consumidores. Embora essa abordagem evite mensagens duplicadas, ela não garante a ordem das mensagens, pois cada mensagem é tratada como seu próprio grupo independente.

Em qualquer sistema com vários produtores e consumidores, sempre existe o risco de entrega duplicada. Se um consumidor não processar uma mensagem antes que o tempo limite de visibilidade expire, o Amazon SQS disponibilizará a mensagem novamente, potencialmente permitindo que outro consumidor a pegue. Para mitigar isso, garanta configurações adequadas de reconhecimento de mensagens e tempo limite de visibilidade com base no tempo de processamento.

Como evitar um grande acúmulo de mensagens com o mesmo ID de grupo de mensagens no Amazon SQS

As filas FIFO oferecem suporte para um máximo de 120.000 mensagens em trânsito (mensagens recebidas por um consumidor, mas ainda não excluídas). Se esse limite for atingido, o Amazon SQS não retornará um erro, mas o processamento poderá ser afetado. Você pode solicitar um aumento desse limite entrando em contato com o [AWS Support](#).

Filas FIFO examinam as primeiras 120.000 mensagens para determinar os grupos de mensagens disponíveis. Se houver grande acúmulo em um único grupo de mensagens, as mensagens de outros grupos enviadas posteriormente permanecerão bloqueadas até que o acúmulo seja processado.

 Note

Um acúmulo de mensagens pode ocorrer quando um consumidor não tem sucesso, repetidamente, no processamento de uma mensagem. Isso pode ser devido a problemas de conteúdo da mensagem ou falhas do lado do consumidor. Para evitar atrasos no processamento de mensagens, configure uma [fila de mensagens não entregues](#) para mover mensagens não processadas após várias tentativas malsucedidas. Isso garante que outras mensagens no mesmo grupo possam ser processadas, evitando gargalos no sistema.

Evitar reutilizar o mesmo ID de grupo de mensagens com filas virtuais no Amazon SQS

Ao usar filas virtuais com uma fila de host compartilhado, evite reutilizar o mesmo [MessageGroupId](#) em diferentes filas virtuais. Se várias filas virtuais compartilharem a mesma fila do host e contiverem mensagens com o mesmo MessageGroupId, essas mensagens poderão se bloquear, impedindo o processamento eficiente. Para garantir um processamento sem problemas, atribua valores MessageGroupId exclusivos para mensagens em diferentes filas virtuais.

Usar o ID de tentativa de solicitação de recebimento do Amazon SQS

O ID de tentativa de solicitação de recebimento é um token exclusivo usado para deduplicar chamadas de [ReceiveMessage](#) no Amazon SQS. Durante uma interrupção na rede ou um problema de conectividade entre sua aplicação e o Amazon SQS, a prática recomendada é:

- Forneça um ID de tentativa de solicitação de recebimento ao fazer uma chamada de `ReceiveMessage`.
- Tente novamente usando o mesmo ID de tentativa de solicitação de recebimento se a operação falhar.

Tempo e processamento de mensagens do Amazon SQS

Este tópico fornece orientação abrangente sobre como otimizar a velocidade e a eficiência do gerenciamento de mensagens no Amazon SQS, incluindo estratégias para processamento oportuno de mensagens, seleção do melhor modo de votação e configuração de sondagem longa para melhorar o desempenho.

Tópicos

- [Processar mensagens em tempo hábil no Amazon SQS](#)
- [Configurar a sondagem longa no Amazon SQS](#)
- [Usar o modo de sondagem apropriado no Amazon SQS](#)

Processar mensagens em tempo hábil no Amazon SQS

Definir o tempo limite de visibilidade depende do tempo de que o seu aplicativo precisa para processar e excluir uma mensagem. Por exemplo, se seu aplicativo exigir 10 segundos para processar uma mensagem e você definir o tempo limite de visibilidade como 15 minutos, deverá esperar por um tempo relativamente longo para tentar processar a mensagem novamente se a tentativa de processamento anterior falhar. Como alternativa, se o aplicativo exigir 10 segundos para processar uma mensagem, mas você definir o tempo limite de visibilidade como apenas 2 segundos, uma mensagem duplicada será recebida por outro consumidor enquanto o consumidor original ainda estiver trabalhando na mensagem.

Para garantir que haja tempo suficiente para processar mensagens, use uma das seguintes estratégias:

- Se você souber (ou puder razoavelmente estimar) quanto tempo leva para processar uma mensagem, amplie o tempo limite de visibilidade da mensagem para o máximo de tempo necessário para processar e excluir a mensagem. Para obter mais informações, consulte [Configurar tempo limite de visibilidade](#).
- Se você não souber quanto tempo leva para processar uma mensagem, crie uma pulsação para o processo do consumidor: especifique o tempo limite de visibilidade inicial (por exemplo, 2 minutos) e, desde que o consumidor ainda funcione na mensagem, continue estendendo o tempo limite de visibilidade em 2 minutos a cada minuto.

Important

O tempo limite de visibilidade máximo é de 12 horas a partir do momento em que o Amazon SQS recebe `ReceiveMessage`. Estender o tempo limite de visibilidade não redefine o período máximo de 12 horas.

Além disso, talvez você não consiga definir o tempo limite em uma mensagem individual para as 12 horas completas (por exemplo, 43.200 segundos), pois a solicitação de `ReceiveMessage` inicia o temporizador. Por exemplo, se você receber uma mensagem e definir imediatamente o máximo de 12 horas enviando uma chamada de

`ChangeMessageVisibility` com `VisibilityTimeout` igual a 43.200 segundos, provavelmente ocorrerá uma falha. No entanto, usar um valor de 43.195 segundos funcionará, a menos que haja um atraso significativo entre a solicitação da mensagem via `ReceiveMessage` e a atualização do tempo limite de visibilidade. Se o consumidor precisar de mais de 12 horas, considere usar o Step Functions.

Configurar a sondagem longa no Amazon SQS

Quando o tempo de espera da ação da API [ReceiveMessage](#) é maior do que 0, a sondagem longa está em vigor. O tempo máximo de espera de sondagem longa é de 20 segundos. A sondagem longa ajuda a reduzir os custos de uso do Amazon SQS eliminando o número de respostas vazias (quando não há mensagens disponíveis para uma solicitação `ReceiveMessage`) e respostas vazias falsas (quando mensagens estão disponíveis, mas não são incluídas em uma resposta). Para obter mais informações, consulte [Sondagem curta e longa do Amazon SQS](#).

Para garantir o processamento ideal de mensagens, use as seguintes estratégias:

- Na maioria dos casos, você pode definir o tempo de espera de `ReceiveMessage` como 20 segundos. Se 20 segundos for muito longo para seu aplicativo, defina um tempo de espera `ReceiveMessage` mais curto (no mínimo, 1 segundo). Se você não usar um AWS SDK para acessar o Amazon SQS, ou se tiver configurado um AWS SDK com um tempo limite mais curto, poderá ser necessário modificar o cliente do Amazon SQS para permitir solicitações mais longas ou para usar um tempo limite menor para sondagem longa.
- Se você implementar a sondagem longa para várias filas, use um thread para cada fila, em vez de um único thread para todas as filas. O uso de um único thread para cada fila permite que seu aplicativo processe as mensagens em cada uma das filas conforme se tornam disponíveis, enquanto o uso de um único thread para sondar várias filas pode fazer com que seu aplicativo não possa processar as mensagens disponíveis em outras filas enquanto o aplicativo aguarda (até 20 segundos) por uma fila que não tem mensagens disponíveis.

Important

Para evitar erros de HTTP, certifique-se de que o tempo limite da resposta HTTP para solicitações `ReceiveMessage` é maior do que o parâmetro `WaitTimeSeconds`. Para obter mais informações, consulte [ReceiveMessage](#).

Usar o modo de sondagem apropriado no Amazon SQS

- A sondagem longa permite que você consuma mensagens da fila do Amazon SQS assim que elas se tornam disponíveis.
 - Para reduzir o custo do uso do Amazon SQS e diminuir o número de recebimentos vazios em uma fila vazia (respostas à ação `ReceiveMessage` que não retornam nenhuma mensagem), habilite a sondagem longa. Para obter mais informações, consulte [Sondagem longa do Amazon SQS](#).
 - Para aumentar a eficiência ao sondar vários threads com vários recebimentos, diminua o número de threads.
 - A sondagem longa é melhor do que a sondagem curta na maioria dos casos.
- A sondagem curta retorna respostas imediatamente, mesmo que a fila do Amazon SQS sondada esteja vazia.
 - Para satisfazer os requisitos de um aplicativo que espera respostas imediatas para a solicitação `ReceiveMessage`, use a sondagem curta.
 - A sondagem curta é cobrada pelo mesmo custo de uma sondagem longa.

Exemplos de SDK do Java do Amazon SQS

O AWS SDK para Java permite que você crie aplicações Java que interagem com o Amazon SQS e outros Serviços da AWS.

- Para instalar e configurar o SDK, consulte [Conceitos básicos](#) no Guia do desenvolvedor do AWS SDK for Java 2.x.
- Para obter operações básicas de fila, como criar uma fila ou enviar uma mensagem, consulte [Trabalhar com filas de mensagens do Amazon SQS](#) no Guia do desenvolvedor do AWS SDK for Java 2.x.
- Este guia também inclui exemplos de recursos adicionais do Amazon SQS, como:
 - [Usar criptografia do lado do servidor com filas do Amazon SQS](#)
 - [Configurar tags para uma fila do Amazon SQS](#)
 - [Enviar atributos de mensagem para uma fila do Amazon SQS](#)

Usar criptografia do lado do servidor com filas do Amazon SQS

Use o AWS SDK para Java para adicionar criptografia no lado do servidor (SSE) a uma fila do Amazon SQS. Cada fila usa uma chave do KMS AWS Key Management Service (AWS KMS) para gerar as chaves de criptografia dos dados. Este exemplo usa a chave do KMS gerenciada pela AWS para o Amazon SQS.

Para obter mais informações sobre como usar a SSE e a função da chave do KMS, consulte [Criptografia em repouso no Amazon SQS](#).

Adicionar SSE a uma fila existente

Para habilitar a criptografia no lado do servidor para uma fila existente, use o método [SetQueueAttributes](#) para definir o atributo `KmsMasterKeyId`.

O código de exemplo a seguir define a AWS KMS key como a chave do KMS gerenciada pela AWS para o Amazon SQS. O exemplo também define o [período de reutilização de AWS KMS key](#) como 140 segundos.

Antes de executar o código de exemplo, verifique se você configurou suas credenciais da AWS. Para mais informações, consulte [Configurar as credenciais e a região da AWS para desenvolvimento](#) no Guia do desenvolvedor do AWS SDK for Java 2.x.

```
public static void addEncryption(String queueName, String kmsMasterKeyAlias) {
    SqsClient sqsClient = SqsClient.create();

    GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
        .queueName(queueName)
        .build();

    GetQueueUrlResponse getQueueUrlResponse;
    try {
        getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
    } catch (QueueDoesNotExistException e) {
        LOGGER.error(e.getMessage(), e);
        throw new RuntimeException(e);
    }
    String queueUrl = getQueueUrlResponse.queueUrl();

    Map<QueueAttributeName, String> attributes = Map.of(
        QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
        QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" // Set the
data key reuse period to 140 seconds.
    );
    // This is
how long SQS can reuse the data key before requesting a new one from KMS.

    SetQueueAttributesRequest attRequest = SetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)
        .attributes(attributes)
        .build();

    try {
        sqsClient.setQueueAttributes(attRequest);
        LOGGER.info("The attributes have been applied to {}", queueName);
    } catch (InvalidAttributeNameException | InvalidAttributeValueException e) {
        LOGGER.error(e.getMessage(), e);
        throw new RuntimeException(e);
    } finally {
        sqsClient.close();
    }
}
```

Desabilitar a SSE para uma fila

Para desabilitar a criptografia no lado do servidor para uma fila existente, defina o atributo `KmsMasterKeyId` como uma string vazia usando o método `SetQueueAttributes`.

Important

`null` não é um valor válido para `KmsMasterKeyId`.

Criar uma fila com SSE

Para habilitar a SSE ao criar a fila, adicione o atributo `KmsMasterKeyId` ao método da API [CreateQueue](#).

O exemplo a seguir cria uma fila nova com a SSE habilitada. A fila usa a chave do KMS gerenciada pela AWS para o Amazon SQS. O exemplo também define o [período de reutilização de AWS KMS key](#) como 160 segundos.

Antes de executar o código de exemplo, verifique se você configurou suas credenciais da AWS. Para mais informações, consulte [Configurar as credenciais e a região da AWS para desenvolvimento](#) no Guia do desenvolvedor do AWS SDK for Java 2.x.

```
// Create an SqsClient for the specified Region.
SqsClient sqsClient = SqsClient.builder().region(Region.US_WEST_1).build();

// Create a hashmap for the attributes. Add the key alias and reuse period to the
// hashmap.
HashMap<QueueAttributeName, String> attributes = new HashMap<QueueAttributeName,
String>();
final String kmsMasterKeyAlias = "alias/aws/sqs"; // the alias of the AWS managed KMS
key for Amazon SQS.
attributes.put(QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias);
attributes.put(QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140");

// Add the attributes to the CreateQueueRequest.
CreateQueueRequest createQueueRequest =
    CreateQueueRequest.builder()
        .queueName(queueName)
        .attributes(attributes)
```

```
        .build();  
sqsClient.createQueue(createQueueRequest);
```

Recuperar atributos de SSE

Para obter informações sobre como recuperar atributos da fila, consulte [Exemplos](#) na Referência da API do Amazon Simple Queue Service.

Para recuperar o ID da chave do KMS ou o período de reutilização da chave de dados de uma fila específica, execute o método [GetQueueAttributes](#) e recupere os valores `KmsMasterKeyId` e `KmsDataKeyReusePeriodSeconds`.

Configurar tags para uma fila do Amazon SQS

Use tags de alocação de custos em suas filas do Amazon SQS para ajudar a organizá-las e identificá-las. Os exemplos a seguir mostram como gerenciar tags usando o AWS SDK para Java. Para obter mais informações, consulte [Tags de alocação de custos do Amazon SQS](#).

Antes de executar o código de exemplo, verifique se você configurou suas credenciais da AWS. Para mais informações, consulte [Configurar as credenciais e a região da AWS para desenvolvimento](#) no Guia do desenvolvedor do AWS SDK for Java 2.x.

Listar tags

Para listar as tags de uma fila, use o método `ListQueueTags`.

```
// Create an SqsClient for the specified region.  
SqsClient sqsClient = SqsClient.builder().region(Region.US_WEST_1).build();  
  
// Get the queue URL.  
String queueName = "MyStandardQ1";  
GetQueueUrlResponse getQueueUrlResponse =  
  
    sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());  
String queueUrl = getQueueUrlResponse.queueUrl();  
  
// Create the ListQueueTagsRequest.  
final ListQueueTagsRequest listQueueTagsRequest =
```



```
ListQueueTagsRequest.builder().queueUrl(queueUrl).build();

// Retrieve the list of queue tags and print them.
final ListQueueTagsResponse listQueueTagsResponse =
    sqsClient.listQueueTags(listQueueTagsRequest);
System.out.println(String.format("ListQueueTags: \tTags for queue %s are %s.\n",
    queueName, listQueueTagsResponse.tags() ));
```

Adicionar ou atualizar tags

Para adicionar ou atualizar valores de etiquetas em uma fila, use o método `TagQueue`.

```
// Create an SqsClient for the specified Region.
SqsClient sqsClient = SqsClient.builder().region(Region.US_WEST_1).build();

// Get the queue URL.
String queueName = "MyStandardQ1";
GetQueueUrlResponse getQueueUrlResponse =

    sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();

// Build a hashmap of the tags.
final HashMap<String, String> addedTags = new HashMap<>();
    addedTags.put("Team", "Development");
    addedTags.put("Priority", "Beta");
    addedTags.put("Accounting ID", "456def");

//Create the TagQueueRequest and add them to the queue.
final TagQueueRequest tagQueueRequest = TagQueueRequest.builder()
    .queueUrl(queueUrl)
    .tags(addedTags)
    .build();
sqsClient.tagQueue(tagQueueRequest);
```

Remover tags

Para remover uma ou mais tags da fila, use o método `UntagQueue`. O exemplo a seguir remove a tag `Accounting ID`.

```
// Create the UntagQueueRequest.
final UntagQueueRequest untagQueueRequest = UntagQueueRequest.builder()
    .queueUrl(queueUrl)
    .tagKeys("Accounting ID")
    .build();

// Remove the tag from this queue.
sqsClient.untagQueue(untagQueueRequest);
```

Enviar atributos de mensagem para uma fila do Amazon SQS

É possível incluir metadados estruturados (como carimbos de data e hora, dados geoespaciais, assinaturas e identificadores) com mensagens usando os atributos de mensagem. Para obter mais informações, consulte [Atributos de mensagem do Amazon SQS](#).

Antes de executar o código de exemplo, verifique se você configurou suas credenciais da AWS. Para mais informações, consulte [Configurar as credenciais e a região da AWS para desenvolvimento](#) no Guia do desenvolvedor do AWS SDK for Java 2.x.

Definir atributos

Para definir um atributo para uma mensagem, adicione o código a seguir que usa o tipo de dado [MessageAttributeValue](#). Para obter mais informações, consulte [Componentes de atributos de mensagem](#) e [Tipos de dados de atributos de mensagem](#).

O AWS SDK para Java calcula automaticamente as somas de verificação do corpo da mensagem e dos atributos da mensagem e as compara com os dados retornados pelo Amazon SQS. Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Java 2.x](#) e [Cálculo do resumo de mensagens MD5 para atributos de mensagem](#) para outras linguagens de programação.

String

Este exemplo define um atributo String chamado Name com o valor Jane.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("Name", new MessageAttributeValue()
    .withDataType("String"))
```

```
.withStringValue("Jane"));
```

Number

Este exemplo define um atributo Number chamado `AccurateWeight` com o valor `230.000000000000000001`.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("AccurateWeight", new MessageAttributeValue()
    .withDataType("Number")
    .withStringValue("230.000000000000000001"));
```

Binary

Este exemplo define um atributo Binary chamado `ByteArray` com o valor de uma matriz de 10 bytes não inicializada.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("ByteArray", new MessageAttributeValue()
    .withDataType("Binary")
    .withBinaryValue(ByteBuffer.wrap(new byte[10])));
```

String (custom)

Este exemplo define o atributo personalizado `String.EmployeeId` chamado `EmployeeId` com o valor `ABC123456`.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("EmployeeId", new MessageAttributeValue()
    .withDataType("String.EmployeeId")
    .withStringValue("ABC123456"));
```

Number (custom)

Este exemplo define o atributo personalizado `Number.AccountId` chamado `AccountId` com o valor `000123456`.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("AccountId", new MessageAttributeValue()
    .withDataType("Number.AccountId")
    .withStringValue("000123456"));
```

 Note

Como o tipo de dados base é Number, o método [ReceiveMessage](#) retorna 123456.

Binary (custom)

Este exemplo define um atributo personalizado Binary.JPEG chamado ApplicationIcon com o valor de uma matriz de 10 bytes não inicializada.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("ApplicationIcon", new MessageAttributeValue()
    .withDataType("Binary.JPEG")
    .withBinaryValue(ByteBuffer.wrap(new byte[10])));
```

Enviar uma mensagem com atributos

Este exemplo adiciona os atributos à SendMessageRequest antes de enviar a mensagem.

```
// Send a message with an attribute.
final SendMessageRequest sendMessageRequest = new SendMessageRequest();
sendMessageRequest.withMessageBody("This is my message text.");
sendMessageRequest.withQueueUrl(myQueueUrl);
sendMessageRequest.withMessageAttributes(messageAttributes);
sqs.sendMessage(sendMessageRequest);
```

 Important

Se você enviar uma mensagem para uma fila primeiro a entrar, primeiro a sair (FIFO), verifique se o método sendMessage é executado depois que você fornecer o ID do grupo de mensagens.

Se usar o método [SendMessageBatch](#) em vez de [SendMessage](#), você deverá especificar os atributos da mensagem de cada mensagem no lote.

Usar APIs com o Amazon SQS

Este tópico fornece informações sobre como criar endpoints do Amazon SQS, fazer solicitações da API de consulta com os métodos GET e POST e usar ações em lote da API. Para ter informações detalhadas sobre as [ações](#) do Amazon SQS, inclusive parâmetros, erros, exemplos e [tipos de dados](#), consulte a [Referência da API do Amazon Simple Queue Service](#).

Para acessar o Amazon SQS utilizando várias linguagens de programação, você também pode usar os [AWS SDKs](#) que contêm a seguinte funcionalidade automática:

- Assinar criptograficamente suas solicitações de serviço
- Recuperar solicitações
- Lidar com respostas de erro

Para obter mais informações, consulte [the section called “Como trabalhar com AWS SDKs”](#).

Para ter informações sobre a ferramenta da linha de comando, consulte as seções do Amazon SQS na [Referência de comandos da AWS CLI](#) e na [Referência de Cmdlet do Ferramentas da AWS para PowerShell](#).

APIs do Amazon SQS com protocolo JSON da AWS

O Amazon SQS usa o protocolo JSON da AWS como mecanismo de transporte para todas as APIs do Amazon SQS nas [versões do AWS SDK](#) especificadas. O protocolo JSON fornece uma taxa mais alta de throughput, menor latência e comunicação mais rápida entre aplicações. O protocolo JSON é mais eficiente na serialização/desserialização de solicitações e respostas quando comparado ao protocolo de consulta da AWS. Se você ainda preferir usar o protocolo de consulta da AWS com APIs SQS, consulte [Quais idiomas são compatíveis com o protocolo AWS JSON usado no Amazon APIs SQS?](#) para saber quais versões do AWS SDK são compatíveis com o protocolo de consulta da AWS do Amazon SQS.

O Amazon SQS usa o protocolo JSON da AWS para comunicação entre clientes do AWS SDK (por exemplo, Java, Python, Golang, JavaScript) e o servidor do Amazon SQS. Uma solicitação HTTP de uma operação de API do Amazon SQS aceita entrada formatada em JSON. A operação do Amazon SQS é executada, e a resposta de execução é enviada de volta ao cliente do SDK no formato JSON. Comparado com a consulta da AWS, o JSON da AWS é mais simples, rápido e eficiente para transportar dados entre o cliente e o servidor.

- O protocolo JSON da AWS atua como um mediador entre o cliente e o servidor do Amazon SQS.
- O servidor não entende a linguagem de programação na qual a operação do Amazon SQS é criada, mas ele entende o protocolo JSON da AWS.
- O protocolo JSON da AWS usa a serialização (conversão do objeto no formato JSON) e a desserialização (conversão do formato JSON em objeto) entre o cliente e o servidor do Amazon SQS.

Para receber mais informações sobre o protocolo JSON da AWS com Amazon SQS, consulte [Protocolo Amazon SQS JSON AWS FAQs](#).

O protocolo JSON da AWS está disponível na [versão do AWS SDK](#) especificada. Para examinar a versão do SDK e as datas de lançamento em todas as variantes de linguagem, consulte a [Matriz de suporte a versões de AWS SDKs e ferramentas](#) no Guia de referência de AWS SDKs e ferramentas.

Fazer solicitações de API de consulta usando o protocolo AWS JSON no Amazon SQS

Este tópico explica como criar um endpoint do Amazon SQS, fazer solicitações POST e interpretar respostas.

Note

AWS O protocolo JSON é compatível com a maioria das variantes de linguagem. Para ver uma lista completa de variantes de linguagem compatíveis, consulte [Quais idiomas são compatíveis com o protocolo AWS JSON usado no Amazon APIs SQS?](#).

Criar um endpoint

Para trabalhar com filas do Amazon SQS, você deve criar um endpoint. Para ter informações sobre endpoints do Amazon SQS, consulte as seguintes páginas na Referência geral da Amazon Web Services:

- [Endpoints regionais](#)
- [Endpoints e cotas do Amazon Simple Queue Service](#)

Cada endpoint do Amazon SQS é totalmente independente. Por exemplo, se duas filas forem nomeadas MyQueue, uma tiver um endpoint `sqs.us-east-2.amazonaws.com` e a outra tiver o endpoint `sqs.eu-west-2.amazonaws.com`, elas não compartilham dados entre si.

Veja a seguir um exemplo de endpoint que faz uma solicitação para criar uma fila.

```
POST / HTTP/1.1
Host: sqs.us-west-2.amazonaws.com
X-Amz-Target: AmazonSQS.CreateQueue
X-Amz-Date: <Date>
Content-Type: application/x-amz-json-1.0
Authorization: <AuthParams>
Content-Length: <PayloadSizeBytes>
Connection: Keep-Alive
{
  "QueueName": "MyQueue",
  "Attributes": {
    "VisibilityTimeout": "40"
  },
  "tags": {
    "QueueType": "Production"
  }
}
```

Note

Os nomes das filas e as URLs das filas diferenciam maiúsculas de minúsculas.

A estrutura de **AUTHPARAMS** depende de como você assina sua solicitação de API. Para obter mais informações, consulte [Assinar solicitações de AWS API](#) na Referência geral da Amazon Web Services.

Como fazer uma solicitação POST

As solicitações POST do Amazon SQS enviam parâmetros de consulta como um formulário no corpo de uma solicitação HTTP.

Veja a seguir um exemplo de cabeçalho HTTP com X-Amz-Target definido como `AmazonSQS.<operationName>` e um cabeçalho HTTP com Content-Type definido como `application/x-amz-json-1.0`.

```
POST / HTTP/1.1
Host: sqs.<region>.<domain>
X-Amz-Target: AmazonSQS.SendMessage
X-Amz-Date: <Date>
Content-Type: application/x-amz-json-1.0
Authorization: <AuthParams>
Content-Length: <PayloadSizeBytes>
Connection: Keep-Alive
{
  "QueueUrl": "https://sqs.<region>.<domain>/<awsAccountId>/<queueName>/",
  "MessageBody": "This is a test message"
}
```

Essa solicitação HTTP POST envia uma mensagem a uma fila do Amazon SQS.

Note

Os dois cabeçalhos HTTP X-Amz-Target e Content-Type são obrigatórios. Seu cliente HTTP pode adicionar outros itens à solicitação HTTP, de acordo com a versão do HTTP do cliente.

Interpretar as respostas da API JSON do Amazon SQS

Quando você envia uma solicitação para o Amazon SQS, ela retorna uma resposta JSON com os resultados. A estrutura de resposta depende da ação da API que você usou.

Para entender os detalhes dessas respostas, consulte:

- A ação específica da API nas [ações da API](#) na Referência de API do Amazon Simple Queue Service
- A [Protocolo Amazon SQS JSON AWS FAQs](#)

Estrutura de resposta JSON bem-sucedida

Se a solicitação for bem-sucedida, o principal elemento de resposta será `x-amzn-RequestId`, que contém o identificador único universal (UUID) da solicitação, bem como outros campos de resposta anexados. Por exemplo, a resposta [CreateQueue](#) a seguir contém o campo `QueueUrl`, que, por sua vez, contém o URL da fila criada.


```
HTTP/1.1 200 OK
x-amzn-RequestId: <requestId>
Content-Length: <PayloadSizeBytes>
Date: <Date>
Content-Type: application/x-amz-json-1.0
{
  "QueueUrl": "https://sqs.us-east-1.amazonaws.com/111122223333/MyQueue"
}
```

Estrutura de resposta de erro JSON

Se uma solicitação não for bem-sucedida, o Amazon SQS retornará a resposta principal, incluindo o cabeçalho HTTP e o corpo.

No cabeçalho HTTP, `x-amzn-RequestId` contém o UUID da solicitação. `x-amzn-query-error` contém duas informações: o tipo de erro e se o erro foi do produtor ou do consumidor.

No corpo de resposta, `"__type"` indica outros detalhes do erro, e `Message` indica a condição de erro em um formato legível.

Veja a seguir um exemplo de resposta de erro no formato JSON:

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: 66916324-67ca-54bb-a410-3f567a7a0571
x-amzn-query-error: AWS.SimpleQueueService.NonExistentQueue;Sender
Content-Length: <PayloadSizeBytes>
Date: <Date>
Content-Type: application/x-amz-json-1.0
{
  "__type": "com.amazonaws.sqs#QueueDoesNotExist",
  "message": "The specified queue does not exist."
}
```

Protocolo Amazon SQS JSON AWS FAQs

Este tópico aborda perguntas frequentes sobre o uso do protocolo AWS JSON com o Amazon SQS.

O que é o protocolo AWS JSON e como ele difere das solicitações e respostas existentes da API Amazon SQS?

JSON é um dos métodos de conexão mais amplamente usados e aceitos para comunicação entre sistemas heterogêneos. O Amazon SQS usa o JSON como meio de comunicação entre um

cliente AWS SDK (por exemplo, Java, Python, Golang) JavaScript e o servidor Amazon SQS. Uma solicitação HTTP de uma operação de API do Amazon SQS aceita entrada formatada em JSON. A operação do Amazon SQS é executada, e a resposta de execução é compartilhada de volta com o cliente do SDK no formato JSON. Comparado com a consulta da AWS, o JSON é mais eficiente para transportar dados entre o cliente e o servidor.

- O protocolo Amazon SQS AWS JSON atua como um mediador entre o cliente e o servidor do Amazon SQS.
- O servidor não entende a linguagem de programação na qual a operação do Amazon SQS é criada, mas entende o protocolo AWS JSON.
- O protocolo Amazon SQS AWS JSON usa a serialização (converter objeto no formato JSON) e a desserialização (converter formato JSON em objeto) entre o cliente e o servidor do Amazon SQS.

Como faço para começar a usar os protocolos AWS JSON para o Amazon SQS?

Para começar com a versão mais recente do AWS SDK e obter mensagens mais rápidas para o Amazon SQS, atualize AWS seu SDK para a versão especificada ou qualquer versão posterior. Para saber mais sobre os clientes do SDK, consulte a coluna “Guia” na tabela abaixo.

A seguir está uma lista das versões do SDK em todas as variantes de linguagem do protocolo AWS JSON para uso com o Amazon SQS: APIs

Linguagem	Repositório do cliente do SDK	Versão obrigatória do cliente do SDK	Guia
C++	aws/ aws-sdk-cpp	1.11.98	AWS SDK para C++
Golang 1.x	aws/ aws-sdk-go	v1.47.7	AWS SDK para Go
Golang 2.x	serra/ 2 aws-sdk-go-v	v1.28.0	AWS SDK para Go V2
Java 1.x	aws/ aws-sdk-java	1.12.585	AWS SDK para Java
Java 2.x		2.21.19	AWS SDK para Java

Linguagem	Repositório do cliente do SDK	Versão obrigatória do cliente do SDK	Guia
	serra/ 2 aws-sdk-java-v		
JavaScript v2.x	aws/ aws-sdk-js	JavaScript em AWS	
JavaScript v3.x	serra/ 3 aws-sdk-js-v	v3.447.0	JavaScript em AWS
.NET	aws/ aws-sdk-net	3.7.681.0	AWS SDK para .NET
PHP	aws/ aws-sdk-php	3.285.2	AWS SDK para PHP
Python-boto3	boto/boto3	1.28.82	AWS SDK para Python (Boto3)
Python-botocore	boto/botocore	1.31.82	AWS SDK para Python (Boto3)
awscli	AWS CLI	1.29.82	AWS Command Line Interface
Ruby	aws/ aws-sdk-ruby	1.67,0	AWS SDK para Ruby

Quais são os riscos de habilitar o protocolo JSON para minhas workloads do Amazon SQS?

Se você estiver usando uma implementação personalizada do AWS SDK ou uma combinação de clientes personalizados e AWS SDK para interagir com o Amazon SQS que AWS gera respostas baseadas em consultas (também conhecidas como baseadas em XML), ela pode ser incompatível com o protocolo JSON. AWS Se você encontrar algum problema, entre em contato com o AWS Support.

E se eu já estiver usando a versão mais recente do AWS SDK, mas minha solução de código aberto não for compatível com JSON?

É necessário alterar a versão do SDK para a versão anterior à que você está usando. Consulte [Como faço para começar a usar os protocolos AWS JSON para o Amazon SQS?](#) para obter mais informações. AWS As versões do SDK listadas em [Como faço para começar a usar os protocolos AWS JSON para o Amazon SQS?](#) usam o protocolo de conexão JSON para o Amazon SQS. APIs Se você alterar seu AWS SDK para a versão anterior, seu Amazon APIs SQS AWS usará a consulta.

Quais idiomas são compatíveis com o protocolo AWS JSON usado no Amazon APIs SQS?

O Amazon SQS oferece suporte a todas as variantes de idioma onde AWS SDKs estão disponíveis ao público em geral (GA). No momento, não há compatibilidade com Kotlin, Rust e Swift. Para saber mais sobre outras variantes de linguagem, consulte [Ferramentas para criar com a AWS](#).

Quais regiões são compatíveis com o protocolo AWS JSON usado no Amazon SQS APIs

O Amazon SQS oferece suporte ao protocolo AWS JSON em todas as [AWS regiões](#) em que o Amazon SQS está disponível.

Quais melhorias de latência posso esperar ao atualizar para as versões especificadas do AWS SDK para o Amazon SQS usando o protocolo JSON?AWS

AWS O protocolo JSON é mais eficiente na serialização e desserialização de solicitações e respostas quando comparado ao protocolo de consulta. AWS Com base em testes de AWS desempenho para uma carga útil de mensagem de 5 KB, o protocolo JSON para Amazon SQS end-to-end reduz a latência do processamento de mensagens em até 23% e reduz o uso da CPU e da memória do lado do cliente do aplicativo.

O protocolo AWS de consulta será descontinuado?

AWS o protocolo de consulta continuará sendo suportado. Você pode continuar usando o protocolo de AWS consulta, desde que sua versão do AWS SDK esteja definida como uma versão anterior diferente da listada em [Como começar a usar protocolos AWS JSON para Amazon SQS](#).

Onde posso receber mais informações sobre o protocolo JSON da AWS?

Você pode receber mais informações sobre o protocolo JSON em [AWS JSON 1.0 protocol](#) na documentação da Smithy. Para saber mais sobre as solicitações de API do Amazon SQS usando o protocolo JSON da AWS, consulte [Fazer solicitações de API de consulta usando o protocolo AWS JSON no Amazon SQS](#).

Fazer solicitações de API de consulta usando o protocolo de consulta da AWS no Amazon SQS

Este tópico explica como criar um endpoint do Amazon SQS, fazer solicitações GET e POST e interpretar respostas.

Criar um endpoint

Para trabalhar com filas do Amazon SQS, você deve criar um endpoint. Para ter informações sobre endpoints do Amazon SQS, consulte as seguintes páginas na Referência geral da Amazon Web Services:

- [Endpoints regionais](#)
- [Endpoints e cotas do Amazon Simple Queue Service](#)

Cada endpoint do Amazon SQS é totalmente independente. Por exemplo, se duas filas forem denominadas MyQueue, uma tiver o endpoint `sqs.us-east-2.amazonaws.com` e a outra tiver o endpoint `sqs.eu-west-2.amazonaws.com`, as duas filas não compartilharão dados entre si.

Veja a seguir um exemplo de um endpoint que faz uma solicitação para criar uma fila.

```
https://sqs.eu-west-2.amazonaws.com/  
?Action=CreateQueue  
&DefaultVisibilityTimeout=40  
&QueueName=MyQueue  
&Version=2012-11-05  
&AUTHPARAMS
```

Note

Os nomes e os URLs de fila diferenciam maiúsculas e minúsculas.

A estrutura de **AUTHPARAMS** depende de como você assina sua solicitação de API. Para obter mais informações, consulte [Assinatura de solicitações da API da AWS](#) na Referência geral da Amazon Web Services.

Como fazer uma solicitação GET

Uma solicitação GET do Amazon SQS é estruturada como um URL que consiste no seguinte:

- Endpoint: o recurso no qual a solicitação está agindo (o [nome da fila e o URL](#)), por exemplo: `https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue`
- Ação: a [ação](#) que você quer executar no endpoint. Um ponto de interrogação (?) separa o endpoint da ação, por exemplo: `?Action=SendMessage&MessageBody=Your%20Message%20Text`
- Parâmetros: os parâmetros da solicitação. Cada parâmetro é separado por um E comercial (&); por exemplo: `&Version=2012-11-05&AUTHPARAMS`

Veja a seguir um exemplo de solicitação GET que envia mensagens a uma fila do Amazon SQS.

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue
?Action=SendMessage&MessageBody=Your%20message%20text
&Version=2012-11-05
&AUTHPARAMS
```

Note

Os nomes e os URLs de fila diferenciam maiúsculas e minúsculas. Como as solicitações GET são URLs, você deve codificar todos os valores de parâmetro no URL. Como não são permitidos espaços nos URLs, cada espaço é codificado no URL como `%20`. O restante do exemplo não foi codificado no URL para facilitar a leitura.

Como fazer uma solicitação POST

As solicitações POST do Amazon SQS enviam parâmetros de consulta como um formulário no corpo de uma solicitação HTTP.

Veja a seguir um exemplo de cabeçalho HTTP com Content-Type definido como `application/x-www-form-urlencoded`.

```
POST /123456789012/MyQueue HTTP/1.1
Host: sqs.us-east-2.amazonaws.com
Content-Type: application/x-www-form-urlencoded
```

O cabeçalho é seguido por uma solicitação GET [form-urlencoded](#) que envia uma mensagem a uma fila do Amazon SQS. Cada parâmetro é separado por um E comercial (&).

```
Action=SendMessage
&MessageBody=Your+Message+Text
&Expires=2020-10-15T12%3A00%3A00Z
&Version=2012-11-05
&AUTHPARAMS
```

Note

Somente o cabeçalho HTTP Content-Type é obrigatório. O **AUTHPARAMS** é o mesmo para a solicitação GET.
Seu cliente HTTP pode adicionar outros itens à solicitação HTTP, de acordo com a versão do HTTP do cliente.

Interpretar as respostas da API XML do Amazon SQS

Quando você envia uma solicitação para o Amazon SQS, ele retorna uma resposta XML que contém os resultados da solicitação. Para entender a estrutura e os detalhes dessas respostas, consulte as [ações de API](#) específicas na Referência de API do Amazon Simple Queue Service.

Estrutura de resposta de XML bem-sucedida

Se a solicitação for bem-sucedida, o elemento de resposta principal receberá o nome da ação, com Response anexada (por exemplo, **ActionNameResponse**).

Esse elemento contém os seguintes elementos filho:

- **ActionNameResult** – contém um elemento específico à ação. Por exemplo, o elemento **CreateQueueResult** contém o elemento **QueueUrl** que, por sua vez, contém o URL da fila criada.
- **ResponseMetadata**: contém o **RequestId**, que, por sua vez, contém o Universal Unique Identifier (UUID) da solicitação.

Veja a seguir um exemplo de uma resposta bem-sucedida no formato XML:

```
<CreateQueueResponse
  xmlns=https://sqs.us-east-2.amazonaws.com/doc/2012-11-05/
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:type=CreateQueueResponse>
  <CreateQueueResult>
    <QueueUrl>https://sqs.us-east-2.amazonaws.com/770098461991/queue2</QueueUrl>
  </CreateQueueResult>
  <ResponseMetadata>
    <RequestId>cb919c0a-9bce-4afe-9b48-9bdf2412bb67</RequestId>
  </ResponseMetadata>
</CreateQueueResponse>
```

Estrutura de resposta de erro de XML

Se uma solicitação não tiver êxito, o Amazon SQS retornará o elemento de resposta principal `ErrorResponse`. Esse elemento contém um elemento `Error` e um elemento `RequestId`.

O elemento `Error` contém os seguintes elementos filhos:

- **Type** – especifica se o erro foi de um produtor ou de um consumidor.
- **Code** – especifica o tipo de erro.
- **Message** – especifica a condição do erro em um formato legível.
- **Detail** – (opcional) especifica detalhes adicionais sobre o erro.

O elemento `RequestId` contém o UUID do pedido.

Veja a seguir um exemplo de uma resposta com erro no formato XML:

```
<ErrorResponse>
  <Error>
    <Type>Sender</Type>
    <Code>InvalidParameterValue</Code>
    <Message>
      Value (quename_nonalpha) for parameter QueueName is invalid.
      Must be an alphanumeric String of 1 to 80 in length.
    </Message>
  </Error>
  <RequestId>42d59b56-7407-4c4a-be0f-4c88daeea257</RequestId>
```



```
</ErrorResponse>
```

Autenticação de solicitações para o Amazon SQS

A autenticação é o processo para identificar e verificar quem envia uma solicitação. Durante a primeira etapa de autenticação, a AWS verifica a identidade do produtor e se ele está [registrado para usar a AWS](#) (para obter mais informações, consulte [Etapa 1: criar um usuário Conta da AWS e IAM](#)). Em seguida, a AWS segue o procedimento a seguir:

1. O produtor (remetente) obtém as credenciais necessárias.
2. O produtor envia uma solicitação e a credencial para o consumidor (destinatário).
3. O consumidor usa a credencial para verificar se o produtor enviou a solicitação.
4. Uma das seguintes situações acontece:
 - Se a autenticação for bem-sucedida, o consumidor processará a solicitação.
 - Se a autenticação falhar, o consumidor rejeitará a solicitação e retornará um erro.

Processo de autenticação básica com HMAC-SHA

Ao acessar o Amazon SQS usando a API de consulta, você deve fornecer os seguintes itens para que a solicitação seja autenticada:

- O ID da chave de acesso da AWS que identifica a Conta da AWS, que a AWS usa para procurar sua chave de acesso secreta.
 - A assinatura da solicitação HMAC-SHA, que é calculada usando sua chave de acesso secreta (um segredo compartilhado do qual somente você e a AWS têm conhecimento. Para obter mais informações, consulte [RFC2104](#)). O [SDK da AWS](#) lida com o processo de assinatura. No entanto, se você enviar uma solicitação de consulta por meio de HTTP ou HTTPS, precisará incluir uma assinatura em cada solicitação de consulta.
1. Derive uma chave de assinatura Signature versão 4. Para obter mais informações, consulte [Derivar a chave de assinatura com Java](#).

Note

O Amazon SQS oferece suporte ao Signature versão 4, que fornece segurança e performance aprimorados com base em SHA256 em relação às versões anteriores.

Ao criar novas aplicações que usem o Amazon SQS, você deverá usar o Signature versão 4.

2. Codifique a assinatura da solicitação usando base64. Este exemplo do código Java faz o seguinte:

```
package amazon.webservices.common;

// Define common routines for encoding data in AWS requests.
public class Encoding {

    /* Perform base64 encoding of input bytes.
     * rawData is the array of bytes to be encoded.
     * return is the base64-encoded string representation of rawData.
     */
    public static String EncodeBase64(byte[] rawData) {
        return Base64.encodeBytes(rawData);
    }
}
```

- O timestamp (ou a expiração) da solicitação. O timestamp que você usa na solicitação deve ser um objeto `dateTime`, com [a data completa, incluindo horas, minutos e segundos](#). Por exemplo: `2007-01-31T23:59:59Z` Embora isso não seja necessário, recomendamos que você informe o objeto usando o fuso horário Tempo Universal Coordenado (Horário de Greenwich).

Note

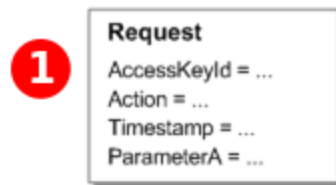
Certifique-se de que a hora do servidor esteja definida corretamente. Se você especificar um timestamp (em vez de uma expiração), a solicitação expirará automaticamente 15 minutos após o horário especificado (a AWS não processa solicitações com timestamps superiores a 15 minutos antes da hora atual nos servidores da AWS).

Se você está usando .NET, não deve enviar timestamps excessivamente específicos (devido a interpretações diferentes em relação a como a precisão de tempo extra deve ser aplicada). Neste caso, você deve criar objetos `dateTime` manualmente com precisão inferior a um milissegundo.

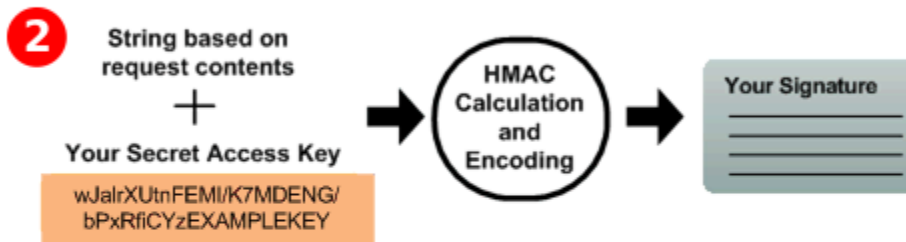
Parte 1: a solicitação do usuário

Veja a seguir o processo que você deve seguir para autenticar solicitações da AWS usando uma assinatura de solicitação HMAC-SHA.

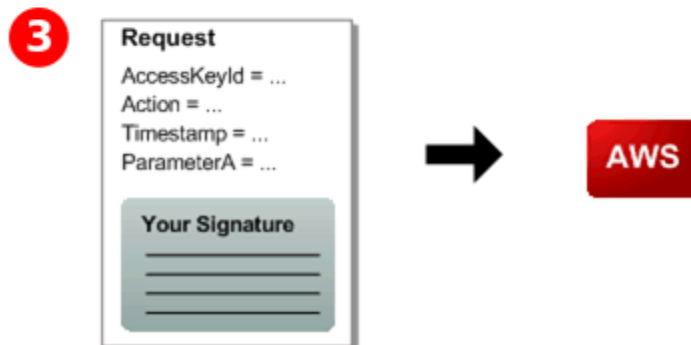
Create a request:



Create an HMAC-SHA signature:



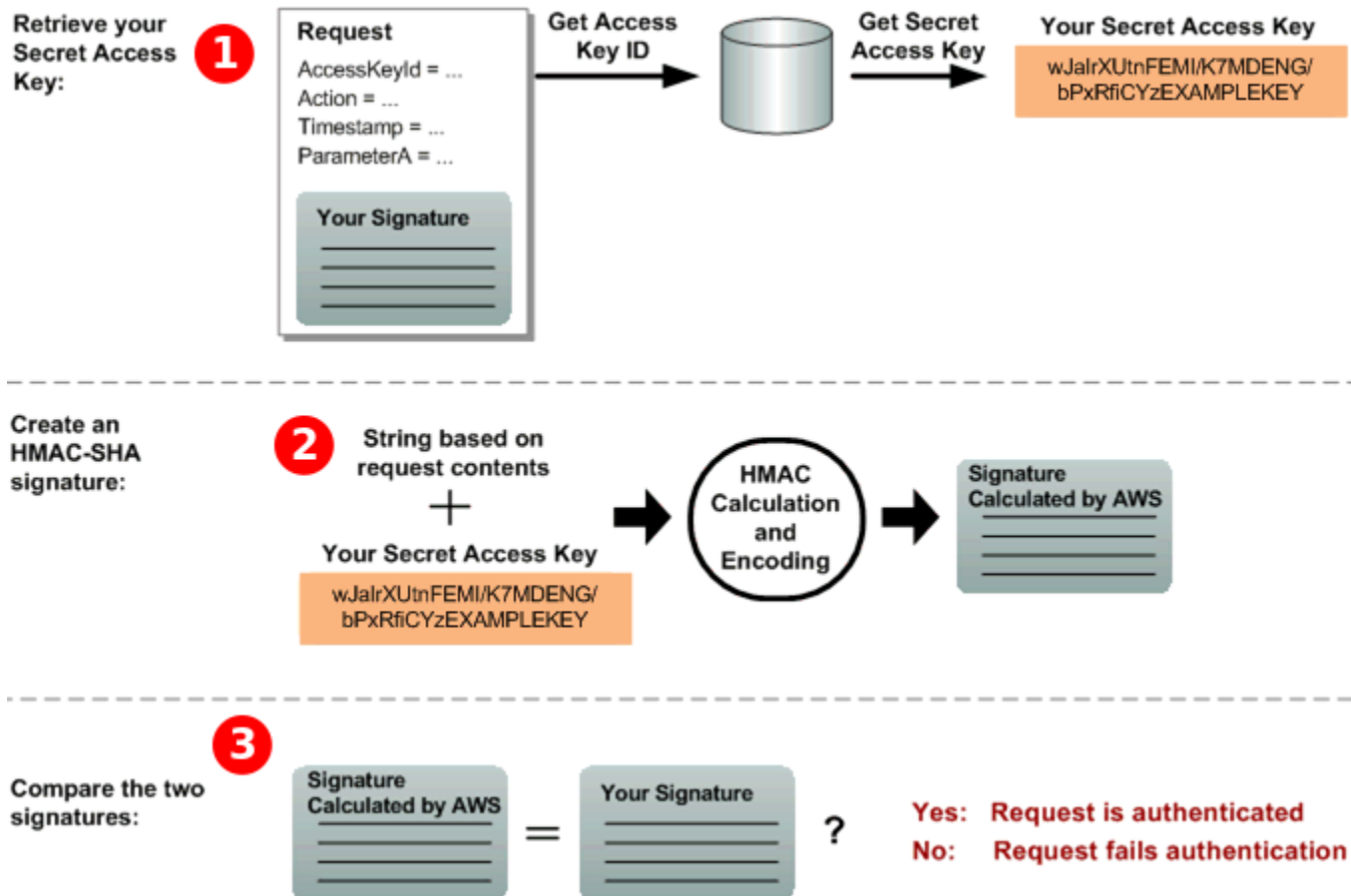
Send the request and signature to AWS:



1. Crie uma solicitação para a AWS.
2. Calcule uma assinatura com código de autenticação de mensagem de hash com chave (HMAC-SHA) usando sua chave de acesso secreta.
3. Inclua a assinatura e o ID da chave de acesso na solicitação e, em seguida, envie a solicitação para a AWS.

Parte 2: a resposta da AWS

AWSA inicia este processo em resposta.



1. AWSA usa o ID da chave de acesso para pesquisar sua chave de acesso secreta.
2. AWSA gera uma assinatura a partir dos dados da solicitação e da chave de acesso secreta usando o mesmo algoritmo usado para calcular a assinatura enviada na solicitação.
3. Uma das seguintes situações acontece:
 - Se a assinatura gerada pela AWS corresponder àquela enviada na solicitação, a AWS considerará a solicitação como autêntica.
 - Se a comparação falhar, a solicitação será descartada e a AWS retornará uma resposta de erro.

Ações em lote do Amazon SQS

O Amazon SQS fornece ações em lote para ajudar você a reduzir custos e manipular até dez mensagens com uma única ação. Essas ações em lote incluem:

- [SendMessageBatch](#)

- [DeleteMessageBatch](#)
- [ChangeMessageVisibilityBatch](#)

Com as ações em lote, é possível realizar várias operações em uma única chamada de API, o que ajuda a otimizar o desempenho e reduzir custos. Aproveite a funcionalidade de lotes usando a API de consulta ou um SDK da AWS compatível com as ações em lote do Amazon SQS.

Detalhes importantes

- Limite de tamanho de mensagem: o tamanho total de todas as mensagens enviadas em uma única chamada de `SendMessageBatch` não pode exceder 1.048.576 bytes (1 MiB)
- Permissões: não é possível definir permissões explicitamente para `SendMessageBatch`, `DeleteMessageBatch` ou `ChangeMessageVisibilityBatch`. Em vez disso, a definição de permissões para `SendMessage`, `DeleteMessage` ou `ChangeMessageVisibility` define permissões para as versões de lote correspondentes dessas ações.
- Suporte do console: o console do Amazon SQS não é compatível com ações em lote. Você deve usar a API de consulta ou um SDK da AWS para realizar operações em lote.

Agrupar ações de mensagem em lotes

Para otimizar ainda mais os custos e a eficiência, considere as seguintes práticas recomendadas para agrupar ações de mensagens em lote:

- Ações de API em lote: use as [ações de API em lote do Amazon SQS](#) para enviar, receber e excluir mensagens, e para alterar o tempo limite de visibilidade de várias mensagens com uma única ação. Isso reduz o número de chamadas de API e os custos associados.
- Buffer do lado do cliente e sondagem longa: combine o buffer do lado do cliente com o agrupamento de solicitações em lote usando a sondagem longa com o [cliente assíncrono em buffer](#) incluído no AWS SDK para Java. Essa abordagem ajuda a minimizar o número de solicitações e otimiza o gerenciamento de grandes volumes de mensagens.

Note

Atualmente, o cliente assíncrono no buffer do Amazon SQS não oferece suporte a filas FIFO.

Habilitar o buffer no lado do cliente e o agrupamento de solicitações em lote com o Amazon SQS

O [AWS SDK para Java](#) inclui o `AmazonSQSBufferedAsyncClient`, que acessa o Amazon SQS. Esse cliente permite um simples agrupamento em lote de solicitações usando o buffer do lado do cliente. Chamadas feitas do cliente são armazenadas primeiramente em buffer, depois enviadas como uma solicitação em lote para o Amazon SQS.

O armazenamento em buffer no lado do cliente permite que até 10 solicitações sejam armazenadas em buffer e enviadas como uma solicitação em lote, diminuindo o custo de uso do Amazon SQS e reduzindo o número de solicitações enviadas. O `AmazonSQSBufferedAsyncClient` armazena tanto as chamadas síncronas quanto as assíncronas em buffer. Solicitações em lote e suporte para [sondagem longa](#) também podem ajudar a aumentar a taxa de transferência. Para obter mais informações, consulte [Aumento do throughput usando escalabilidade horizontal e processamento de ações em lote com o Amazon SQS](#).

como o `AmazonSQSBufferedAsyncClient` implementa a mesma interface que o `AmazonSQSAsyncClient`, migrar de `AmazonSQSAsyncClient` para `AmazonSQSBufferedAsyncClient` normalmente requer apenas pequenas mudanças no seu código existente.

Note

Atualmente, o cliente assíncrono no buffer do Amazon SQS não oferece suporte a filas FIFO.

Usar o `AmazonSQSBufferedAsyncClient`

Antes de começar, conclua as etapas em [Configurar o Amazon SQS](#).

AWS SDK para Java 1.x

Para o AWS SDK para Java 1.x, você pode criar um `AmazonSQSBufferedAsyncClient` novo com base no exemplo a seguir:

```
// Create the basic Amazon SQS async client
final AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient();

// Create the buffered client
```

```
final AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync);
```

Depois de criar o novo `AmazonSQSBufferedAsyncClient`, você pode usá-lo para enviar várias solicitações ao Amazon SQS (da mesma forma que faria com o `AmazonSQSAsyncClient`), por exemplo:

```
final CreateQueueRequest createRequest = new
    CreateQueueRequest().withQueueName("MyQueue");

final CreateQueueResult res = bufferedSqs.createQueue(createRequest);

final SendMessageRequest request = new SendMessageRequest();
final String body = "Your message text" + System.currentTimeMillis();
request.setMessageBody( body );
request.setQueueUrl(res.getQueueUrl());

final Future<SendMessageResult> sendResult = bufferedSqs.sendMessageAsync(request);

final ReceiveMessageRequest receiveRq = new ReceiveMessageRequest()
    .withMaxNumberOfMessages(1)
    .withQueueUrl(queueUrl);
final ReceiveMessageResult rx = bufferedSqs.receiveMessage(receiveRq);
```

Configurar o `AmazonSQSBufferedAsyncClient`

`AmazonSQSBufferedAsyncClient` é pré-configurado com configurações que funcionarão para a maioria dos casos de uso. Você pode configurar ainda mais o `AmazonSQSBufferedAsyncClient`, por exemplo:

1. Crie uma instância da classe `QueueBufferConfig` com os parâmetros de configuração necessários.
2. Informe a instância para o construtor `AmazonSQSBufferedAsyncClient`.

```
// Create the basic Amazon SQS async client
final AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient();

final QueueBufferConfig config = new QueueBufferConfig()
    .withMaxInflightReceiveBatches(5)
    .withMaxDoneReceiveBatches(15);

// Create the buffered client
```

```
final AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync, config);
```

Parâmetros de configuração QueueBufferConfig

Parameter	Valor padrão	Descrição
longPoll	true	Quando longPoll está definido como true, AmazonSQSBufferedAsyncClient tenta usar a sondagem longa ao consumir mensagens.
longPollWaitTimeoutSeconds	20 s	A quantidade máxima de tempo, em segundos, em que uma chamada ReceiveMessage é bloqueada no servidor aguardando as mensagens aparecerem na fila antes de retornar com um resultado de recebimento vazio.  Note Quando a sondagem longa está desativada, essa configuração não tem efeito.
maxBatchOpenMs	200ms	A quantidade máxima de tempo (em milissegundos) que uma chamada de saída aguarda outras chamadas

Parameter	Valor padrão	Descrição
		com as quais ela coloca mensagens do mesmo tipo em lote. Quanto maior for a configuração, menos lotes serão necessários para executar a mesma quantidade de trabalho (no entanto, a primeira chamada em um lote deve passar mais tempo em espera). Quando esse parâmetro é definido como 0, as solicitações enviadas não aguardam outras solicitações, desativando efetivamente o processamento em lotes.

Parameter	Valor padrão	Descrição
maxBatchSize	10 solicitações por lote	O número máximo de mensagens que são armazenadas em lote em uma única solicitação. Quanto maior a configuração, menos lotes serão necessários para executar o mesmo número de solicitações.  Note Dez solicitações por lote é o valor máximo permitido para o Amazon SQS.
maxBatchSizeBytes	1 MiB	O tamanho máximo de um lote de mensagens, em bytes, que o cliente tenta enviar ao Amazon SQS.  Note 1 MiB é o valor máximo permitido para o Amazon SQS.

Parameter	Valor padrão	Descrição
<code>maxDoneReceiveBatches</code>	10 lotes	O número máximo de lotes de recebimento que AmazonSQS <code>BufferedAsyncClient</code> pré-busca e armazena no lado do cliente. Quanto maior for a configuração, mais solicitações de recebimento poderão ser atendidas sem a necessidade de fazer uma chamada ao Amazon SQS (no entanto, quanto mais mensagens forem buscadas previamente, mais tempo elas permanecem no buffer, fazendo com que o tempo limite de visibilidade expire).  Note 0 indica que toda a pré-busca de mensagens está desabilitada e as mensagens são consumidas apenas sob demanda.

Parameter	Valor padrão	Descrição
<code>maxInflightOutboundBatches</code>	5 lotes	O número máximo de lotes de saída ativos que podem ser processados ao mesmo tempo. Quanto maior for a configuração, mais rapidamente os lotes de saída poderão ser enviados (sujeito a outras cotas, como CPU ou largura de banda) e mais threads serão consumidos pelo <code>AmazonSQSBufferedAsyncClient</code>.

Parameter	Valor padrão	Descrição
<code>maxInflightReceiveBatches</code>	10 lotes	O número máximo de lotes de recebimento ativos que podem ser processados ao mesmo tempo. Quanto maior for a configuração, mais mensagens serão recebidas (sujeito a outras cotas, como CPU ou largura de banda) e mais threads serão consumidos pelo <code>AmazonSQSBufferedAsyncClient</code>.  Note 0 indica que toda a pré-busca de mensagens está desabilitada e as mensagens são consumidas apenas sob demanda.

Parameter	Valor padrão	Descrição
<code>visibilityTimeoutSeconds</code>	-1	Quando esse parâmetro é definido como um valor positivo e diferente de zero, o tempo limite de visibilidade definido aqui substitui o tempo limite de visibilidade definido na fila a partir da qual as mensagens são consumidas.  Note -1 indica que a configuração padrão foi selecionada para a fila. Não é possível configurar o tempo limite de visibilidade para 0.

AWS SDK para Java 2.x

Para o AWS SDK para Java 2.x, você pode criar um `SqsAsyncBatchManager` novo com base no exemplo a seguir:

```
// Create the basic Sqs Async Client
SqsAsyncClient sqs = SqsAsyncClient.builder()
    .region(Region.US_EAST_1)
    .build();

// Create the batch manager
SqsAsyncBatchManager sqsAsyncBatchManager = sqs.batchManager();
```

Depois de criar o novo `SqsAsyncBatchManager`, você pode usá-lo para enviar várias solicitações ao Amazon SQS (da mesma forma que faria com o `SqsAsyncClient`), por exemplo:

```
final String queueName = "MyAsyncBufferedQueue" + UUID.randomUUID();
final CreateQueueRequest request =
    CreateQueueRequest.builder().queueName(queueName).build();
final String queueUrl = sqs.createQueue(request).join().queueUrl();
System.out.println("Queue created: " + queueUrl);

// Send messages
CompletableFuture<SendMessageResponse> sendMessageFuture;
for (int i = 0; i < 10; i++) {
    final int index = i;
    sendMessageFuture = sqsAsyncBatchManager.sendMessage(
        r -> r.messageBody("Message " + index).queueUrl(queueUrl));
    SendMessageResponse response = sendMessageFuture.join();
    System.out.println("Message " + response.messageId() + " sent!");
}

// Receive messages with customized configurations
CompletableFuture<ReceiveMessageResponse> receiveResponseFuture =
    customizedBatchManager.receiveMessage(
        r -> r.queueUrl(queueUrl)
            .waitTimeSeconds(10)
            .visibilityTimeout(20)
            .maxNumberOfMessages(10)
    );
System.out.println("You have received " +
    receiveResponseFuture.join().messages().size() + " messages in total.");

// Delete messages
DeleteQueueRequest deleteQueueRequest =
    DeleteQueueRequest.builder().queueUrl(queueUrl).build();
int code = sqs.deleteQueue(deleteQueueRequest).join().sdkHttpResponse().statusCode();
System.out.println("Queue is deleted, with statusCode " + code);
```

Como configurar o `SqsAsyncBatchManager`

`SqsAsyncBatchManager` é pré-configurado com configurações que funcionarão para a maioria dos casos de uso. Você pode configurar ainda mais o `SqsAsyncBatchManager`, por exemplo:

Como criar configurações personalizadas usando o `SqsAsyncBatchManager.Builder`:

```

SqsAsyncBatchManager customizedBatchManager = SqsAsyncBatchManager.builder()
    .client(sqs)
    .scheduledExecutor(Executors.newScheduledThreadPool(5))
    .overrideConfiguration(b -> b
        .maxBatchSize(10)
        .sendRequestFrequency(Duration.ofMillis(200))
        .receiveMessageMinWaitDuration(Duration.ofSeconds(10))
        .receiveMessageVisibilityTimeout(Duration.ofSeconds(20))
        .receiveMessageAttributeNames(Collections.singletonList("*"))

        .receiveMessageSystemAttributeNames(Collections.singletonList(MessageSystemAttributeName.ALL))
    .build();

```

BatchOverrideConfiguration

Parâmetros do

Parameter	Valor padrão	Descrição
maxBatchSize	10 solicitações por lote	O número máximo de mensagens que são armazenadas em lote em uma única solicitação. Quanto maior a configuração, menos lotes serão necessários para executar o mesmo número de solicitações.  Note O valor máximo permitido para o Amazon SQS é de 10 solicitações por lote.
sendRequestFrequency	200ms	A quantidade máxima de tempo (em milissegundos) que uma chamada de saída aguarda outras chamadas com as quais ela coloca

Parameter	Valor padrão	Descrição
		mensagens do mesmo tipo em lote. Quanto maior for a configuração, menos lotes serão necessários para executar a mesma quantidade de trabalho (no entanto, a primeira chamada em um lote deve passar mais tempo em espera). Quando esse parâmetro é definido como 0, as solicitações enviadas não aguardam outras solicitações, desativando efetivamente o processamento em lotes.

Parameter	Valor padrão	Descrição
<code>receiveMessageVisibilityTimeout</code>	-1	Quando esse parâmetro é definido como um valor positivo e diferente de zero, o tempo limite de visibilidade definido aqui substitui o tempo limite de visibilidade definido na fila a partir da qual as mensagens são consumidas.  Note 1 indica que a configuração padrão foi selecionada para a fila. Não é possível configurar o tempo limite de visibilidade para 0.
<code>receiveMessageMinimumWaitDuration</code>	50 ms	O tempo mínimo (em milissegundos) que uma chamada de <code>receiveMessage</code> aguardará para que as mensagens disponíveis sejam buscadas. Quanto maior a configuração, menos lotes serão necessários para executar o mesmo número de solicitações.

Aumento do throughput usando escalabilidade horizontal e processamento de ações em lote com o Amazon SQS

O Amazon SQS é compatível com mensagens de alto throughput. Para obter detalhes sobre os limites de throughput, consulte [Cotas de mensagens do Amazon SQS](#).

Para maximizar o throughput:

- [Escale](#) produtores e consumidores horizontalmente adicionando mais instâncias de cada um.
- Use o [agrupamento de ações em lote](#) para enviar ou receber várias mensagens em uma única solicitação, reduzindo a sobrecarga de chamadas da API.

Escalabilidade horizontal

Como você acessa o Amazon SQS por meio de um protocolo HTTP de solicitação-resposta, a latência da solicitação (o intervalo de tempo entre o início de uma solicitação e o recebimento de uma resposta) limita a taxa de transferência que você pode obter de uma única thread por meio de uma única conexão. Por exemplo, se a latência média de um cliente com base no Amazon EC2 para o Amazon SQS na mesma região for de cerca de 20 ms, a taxa de transferência máxima de uma única thread por meio de uma única conexão será em média 50 TPS.

A escalabilidade horizontal envolve o aumento do número de produtores de mensagem (que fazem a solicitação [SendMessage](#)) e dos consumidores (que fazem solicitações [ReceiveMessage](#) e [DeleteMessage](#)) para aumentar sua taxa de transferência de fila geral. Você pode escalar horizontalmente de três formas:

- Aumentar o número de threads por cliente
- Adicionar mais clientes
- Aumentar o número de threads por cliente e adicionar mais clientes

Ao adicionar mais clientes, você obtém ganhos essencialmente lineares na taxa de transferência da fila. Por exemplo, se você dobrar o número de clientes, terá duas vezes a taxa de transferência.

Processamento de ações em lotes

O processamento em lotes executa mais trabalho durante a ida e a volta do serviço (por exemplo, quando você envia várias mensagens com uma única solicitação [SendMessageBatch](#)). As ações de em lote do Amazon SQS são [SendMessageBatch](#), [DeleteMessageBatch](#) e

[ChangeMessageVisibilityBatch](#). Para aproveitar o processamento em lotes sem alterar os produtores ou consumidores, você pode usar o [cliente assíncrono armazenado em buffer para o Amazon SQS](#).

 Note

Como [ReceiveMessage](#) pode processar 10 mensagens por vez, não há nenhuma ação `ReceiveMessageBatch`.

O processamento em lotes distribui a latência da ação de lote nas várias mensagens de uma solicitação em lote em vez de aceitar toda a latência para uma única mensagem (por exemplo, uma solicitação [SendMessage](#)). Como cada ida e volta carrega mais trabalho, as solicitações de lote tornam mais eficiente o uso de threads e conexões, melhorando, dessa forma, a taxa de transferência.

Você pode combinar processamentos em lote com escalabilidade horizontal para fornecer taxa de transferência com menos threads, conexões e solicitações em comparação com as solicitações de mensagens individuais. Você pode usar ações em lotes do Amazon SQS para enviar, receber ou excluir até 10 mensagens por vez. Como o Amazon SQS cobra por solicitação, o processamento em lotes pode reduzir substancialmente os custos.

O processamento em lotes pode criar certa complexidade para o seu aplicativo (por exemplo, o aplicativo precisa acumular as mensagens antes de enviá-las e, às vezes, precisará esperar mais por uma resposta). No entanto, o processamento em lotes pode ser eficaz nos seguintes casos:

- Seu aplicativo gera muitas mensagens em um curto intervalo de tempo, portanto, o atraso nunca é muito longo.
- Um consumidor de mensagem busca as mensagens de uma fila a seu critério, ao contrário de produtores de mensagem típicos que precisam enviar mensagens em resposta a eventos que eles não controlam.

 Important

Uma solicitação de lote pode ser bem-sucedida, mesmo que ocorra falha nas mensagens individuais no lote. Após uma solicitação de lote, você sempre deve verificar a existência de falhas em mensagens individuais e repetir a ação, se necessário.

Exemplo de Java funcional para operações únicas e solicitações em lote

Pré-requisitos

Adicione os pacotes `aws-java-sdk-sqs.jar`, `aws-java-sdk-ec2.jar` e `commons-logging.jar` ao caminho da classe de compilação do Java. O exemplo a seguir mostra essas dependências em um arquivo `pom.xml` do projeto Maven.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-sqs</artifactId>
    <version>LATEST</version>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-ec2</artifactId>
    <version>LATEST</version>
  </dependency>
  <dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>LATEST</version>
  </dependency>
</dependencies>
```

SimpleProducerConsumer.java

O exemplo de código Java a seguir implementa um padrão simples de produtor-consumidor. O thread principal gera um número de threads de produtor e consumidor que processam mensagens de 1 KB em um determinado momento. Ele inclui produtores e os consumidores que fazem solicitações de operação únicas e outros que fazem solicitações de lote.

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 * https://aws.amazon.com/apache2.0
 *
```

```
* or in the "license" file accompanying this file. This file is distributed
* on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
* express or implied. See the License for the specific language governing
* permissions and limitations under the License.
*
*/

import com.amazonaws.AmazonClientException;
import com.amazonaws.ClientConfiguration;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.*;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Start a specified number of producer and consumer threads, and produce-consume
 * for the least of the specified duration and 1 hour. Some messages can be left
 * in the queue because producers and consumers might not be in exact balance.
 */
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Log log = LogFactory.getLog(SimpleProducerConsumer.class);

    public static void main(String[] args) throws InterruptedException {

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
        final int producerCount = input.nextInt();
```

```
System.out.print("Enter the number of consumers: ");
final int consumerCount = input.nextInt();

System.out.print("Enter the number of messages per batch: ");
final int batchSize = input.nextInt();

System.out.print("Enter the message size in bytes: ");
final int messageSizeByte = input.nextInt();

System.out.print("Enter the run time in minutes: ");
final int runTimeMinutes = input.nextInt();

/*
 * Create a new instance of the builder with all defaults (credentials
 * and region) set automatically. For more information, see Creating
 * Service Clients in the AWS SDK for Java Developer Guide.
 */
final ClientConfiguration clientConfiguration = new ClientConfiguration()
    .withMaxConnections(producerCount + consumerCount);

final AmazonSQS sqsClient = AmazonSQSClientBuilder.standard()
    .withClientConfiguration(clientConfiguration)
    .build();

final String queueUrl = sqsClient
    .getQueueUrl(new GetQueueUrlRequest(queueName)).getQueueUrl();

// The flag used to stop producer, consumer, and monitor threads.
final AtomicBoolean stop = new AtomicBoolean(false);

// Start the producers.
final AtomicInteger producedCount = new AtomicInteger();
final Thread[] producers = new Thread[producerCount];
for (int i = 0; i < producerCount; i++) {
    if (batchSize == 1) {
        producers[i] = new Producer(sqsClient, queueUrl, messageSizeByte,
            producedCount, stop);
    } else {
        producers[i] = new BatchProducer(sqsClient, queueUrl, batchSize,
            messageSizeByte, producedCount,
            stop);
    }
    producers[i].start();
}
```

```
}

// Start the consumers.
final AtomicInteger consumedCount = new AtomicInteger();
final Thread[] consumers = new Thread[consumerCount];
for (int i = 0; i < consumerCount; i++) {
    if (batchSize == 1) {
        consumers[i] = new Consumer(sqsClient, queueUrl, consumedCount,
            stop);
    } else {
        consumers[i] = new BatchConsumer(sqsClient, queueUrl, batchSize,
            consumedCount, stop);
    }
    consumers[i].start();
}

// Start the monitor thread.
final Thread monitor = new Monitor(producedCount, consumedCount, stop);
monitor.start();

// Wait for the specified amount of time then stop.
Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runtimeMinutes,
    MAX_RUNTIME_MINUTES)));
stop.set(true);

// Join all threads.
for (int i = 0; i < producerCount; i++) {
    producers[i].join();
}

for (int i = 0; i < consumerCount; i++) {
    consumers[i].join();
}

monitor.interrupt();
monitor.join();
}

private static String makeRandomString(int sizeByte) {
    final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
    new Random().nextBytes(bs);
    bs[0] = (byte) ((bs[0] | 64) & 127);
    return new BigInteger(bs).toString(32);
}
```



```
/**
 * The producer thread uses {@code SendMessage}
 * to send messages until it is stopped.
 */
private static class Producer extends Thread {
    final AmazonSQS sqsClient;
    final String queueUrl;
    final AtomicInteger producedCount;
    final AtomicBoolean stop;
    final String theMessage;

    Producer(AmazonSQS sqsQueueBuffer, String queueUrl, int messageSizeByte,
            AtomicInteger producedCount, AtomicBoolean stop) {
        this.sqsClient = sqsQueueBuffer;
        this.queueUrl = queueUrl;
        this.producedCount = producedCount;
        this.stop = stop;
        this.theMessage = makeRandomString(messageSizeByte);
    }

    /**
     * The producedCount object tracks the number of messages produced by
     * all producer threads. If there is an error, the program exits the
     * run() method.
     */
    public void run() {
        try {
            while (!stop.get()) {
                sqsClient.sendMessage(new SendMessageRequest(queueUrl,
                    theMessage));
                producedCount.incrementAndGet();
            }
        } catch (AmazonClientException e) {
            /**
             * By default, AmazonSQSClient retries calls 3 times before
             * failing. If this unlikely condition occurs, stop.
             */
            log.error("Producer: " + e.getMessage());
            System.exit(1);
        }
    }
}
```

```
/**
 * The producer thread uses {@code SendMessageBatch}
 * to send messages until it is stopped.
 */
private static class BatchProducer extends Thread {
    final AmazonSQS sqsClient;
    final String queueUrl;
    final int batchSize;
    final AtomicInteger producedCount;
    final AtomicBoolean stop;
    final String theMessage;

    BatchProducer(AmazonSQS sqsQueueBuffer, String queueUrl, int batchSize,
                  int messageSizeByte, AtomicInteger producedCount,
                  AtomicBoolean stop) {
        this.sqsClient = sqsQueueBuffer;
        this.queueUrl = queueUrl;
        this.batchSize = batchSize;
        this.producedCount = producedCount;
        this.stop = stop;
        this.theMessage = makeRandomString(messageSizeByte);
    }

    public void run() {
        try {
            while (!stop.get()) {
                final SendMessageBatchRequest batchRequest =
                    new SendMessageBatchRequest().withQueueUrl(queueUrl);

                final List<SendMessageBatchRequestEntry> entries =
                    new ArrayList<SendMessageBatchRequestEntry>();
                for (int i = 0; i < batchSize; i++)
                    entries.add(new SendMessageBatchRequestEntry()
                        .withId(Integer.toString(i))
                        .withMessageBody(theMessage));
                batchRequest.setEntries(entries);

                final SendMessageBatchResult batchResult =
                    sqsClient.sendMessageBatch(batchRequest);
                producedCount.addAndGet(batchResult.getSuccessful().size());

                /*
                 * Because SendMessageBatch can return successfully, but
                 * individual batch items fail, retry the failed batch items.
                 */
            }
        } catch (Exception e) {
            // Handle exception
        }
    }
}
```

```

        */
        if (!batchResult.getFailed().isEmpty()) {
            log.warn("Producer: retrying sending "
                + batchResult.getFailed().size() + " messages");
            for (int i = 0, n = batchResult.getFailed().size();
                i < n; i++) {
                sqsClient.sendMessage(new
                    SendMessageRequest(queueUrl, theMessage));
                producedCount.incrementAndGet();
            }
        }
    }
} catch (AmazonClientException e) {
    /*
     * By default, AmazonSQSClient retries calls 3 times before
     * failing. If this unlikely condition occurs, stop.
     */
    log.error("BatchProducer: " + e.getMessage());
    System.exit(1);
}
}

/**
 * The consumer thread uses {@code ReceiveMessage} and {@code DeleteMessage}
 * to consume messages until it is stopped.
 */
private static class Consumer extends Thread {
    final AmazonSQS sqsClient;
    final String queueUrl;
    final AtomicInteger consumedCount;
    final AtomicBoolean stop;

    Consumer(AmazonSQS sqsClient, String queueUrl, AtomicInteger consumedCount,
        AtomicBoolean stop) {
        this.sqsClient = sqsClient;
        this.queueUrl = queueUrl;
        this.consumedCount = consumedCount;
        this.stop = stop;
    }

    /*
     * Each consumer thread receives and deletes messages until the main
     * thread stops the consumer thread. The consumedCount object tracks the

```

```

    * number of messages that are consumed by all consumer threads, and the
    * count is logged periodically.
    */
    public void run() {
        try {
            while (!stop.get()) {
                try {
                    final ReceiveMessageResult result = sqsClient
                        .receiveMessage(new
                            ReceiveMessageRequest(queueUrl));

                    if (!result.getMessages().isEmpty()) {
                        final Message m = result.getMessages().get(0);
                        sqsClient.deleteMessage(new
                            DeleteMessageRequest(queueUrl,
                                m.getReceiptHandle()));
                        consumedCount.incrementAndGet();
                    }
                } catch (AmazonClientException e) {
                    log.error(e.getMessage());
                }
            }
        } catch (AmazonClientException e) {
            /*
             * By default, AmazonSQSClient retries calls 3 times before
             * failing. If this unlikely condition occurs, stop.
             */
            log.error("Consumer: " + e.getMessage());
            System.exit(1);
        }
    }
}

/**
 * The consumer thread uses {@code ReceiveMessage} and {@code
 * DeleteMessageBatch} to consume messages until it is stopped.
 */
private static class BatchConsumer extends Thread {
    final AmazonSQS sqsClient;
    final String queueUrl;
    final int batchSize;
    final AtomicInteger consumedCount;
    final AtomicBoolean stop;

```

```
BatchConsumer(AmazonSQS sqsClient, String queueUrl, int batchSize,
               AtomicInteger consumedCount, AtomicBoolean stop) {
    this.sqsClient = sqsClient;
    this.queueUrl = queueUrl;
    this.batchSize = batchSize;
    this.consumedCount = consumedCount;
    this.stop = stop;
}

public void run() {
    try {
        while (!stop.get()) {
            final ReceiveMessageResult result = sqsClient
                .receiveMessage(new ReceiveMessageRequest(queueUrl)
                    .withMaxNumberOfMessages(batchSize));

            if (!result.getMessages().isEmpty()) {
                final List<Message> messages = result.getMessages();
                final DeleteMessageBatchRequest batchRequest =
                    new DeleteMessageBatchRequest()
                        .withQueueUrl(queueUrl);

                final List<DeleteMessageBatchRequestEntry> entries =
                    new ArrayList<DeleteMessageBatchRequestEntry>();
                for (int i = 0, n = messages.size(); i < n; i++)
                    entries.add(new DeleteMessageBatchRequestEntry()
                        .withId(Integer.toString(i))
                        .withReceiptHandle(messages.get(i)
                            .getReceiptHandle()));
                batchRequest.setEntries(entries);

                final DeleteMessageBatchResult batchResult = sqsClient
                    .deleteMessageBatch(batchRequest);
                consumedCount.addAndGet(batchResult.getSuccessful().size());

                /*
                 * Because DeleteMessageBatch can return successfully,
                 * but individual batch items fail, retry the failed
                 * batch items.
                 */
                if (!batchResult.getFailed().isEmpty()) {
                    final int n = batchResult.getFailed().size();
                    log.warn("Producer: retrying deleting " + n
                        + " messages");
                }
            }
        }
    }
}
```

```

        for (BatchResultErrorEntry e : batchResult
            .getFailed()) {

            sqsClient.deleteMessage(
                new DeleteMessageRequest(queueUrl,
                    messages.get(Integer
                        .parseInt(e.getId()))
                        .getReceiptHandle()));

            consumedCount.incrementAndGet();
        }
    }
}
} catch (AmazonClientException e) {
    /**
     * By default, AmazonSQSClient retries calls 3 times before
     * failing. If this unlikely condition occurs, stop.
     */
    log.error("BatchConsumer: " + e.getMessage());
    System.exit(1);
}
}

/**
 * This thread prints every second the number of messages produced and
 * consumed so far.
 */
private static class Monitor extends Thread {
    private final AtomicInteger producedCount;
    private final AtomicInteger consumedCount;
    private final AtomicBoolean stop;

    Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
        AtomicBoolean stop) {
        this.producedCount = producedCount;
        this.consumedCount = consumedCount;
        this.stop = stop;
    }

    public void run() {
        try {
            while (!stop.get()) {

```

```
        Thread.sleep(1000);
        log.info("produced messages = " + producedCount.get()
                + ", consumed messages = " + consumedCount.get());
    }
} catch (InterruptedException e) {
    // Allow the thread to exit.
}
}
}
```

Monitorar métricas de volume da execução de exemplo

O Amazon SQS gera automaticamente métricas de volume para mensagens enviadas, recebidas e excluídas. Você pode acessar essas e outras métricas por meio da guia Monitoring (Monitoramento) de sua fila ou no [console do CloudWatch](#).

Note

As métricas podem levar até 15 minutos após a fila começar para ficar disponíveis.

Usar o Amazon SQS com um SDK da AWS

AWS oferece kits de desenvolvimento de software (SDKs) que estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ Exemplos de código do
AWS CLI	AWS CLI Exemplos de código da
AWS SDK para Go	AWS SDK para Go Exemplos de código da
AWS SDK para Java	AWS SDK para Java Exemplos de código da
AWS SDK para JavaScript	AWS SDK para JavaScript Exemplos de código da

Documentação do SDK	Exemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin Exemplos de código da
AWS SDK para .NET	AWS SDK para .NET Exemplos de código da
AWS SDK para PHP	AWS SDK para PHP Exemplos de código da
Ferramentas da AWS para PowerShell	Ferramentas da AWS para PowerShell Exemplos de código da
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) Exemplos de código da
AWS SDK para Ruby	AWS SDK para Ruby Exemplos de código da
AWS SDK para Rust	AWS SDK para Rust Exemplos de código da
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP Exemplos de código da
AWS SDK para Swift	AWS SDK para Swift Exemplos de código do

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link
Fornecer feedback na parte inferior desta página.

Usar o JMS com o Amazon SQS

A biblioteca de mensagens Java do Amazon SQS é uma interface Java Message Service (JMS) para o Amazon SQS que permite aproveitar o Amazon SQS em aplicações que já utilizam JMS. A interface permite que você use o Amazon SQS como o provedor de JMS com o mínimo de alterações no código. Junto com o AWS SDK para Java, a biblioteca de mensagens Java do Amazon SQS permite criar conexões e sessões JMS, bem como produtores e consumidores que enviam e recebem mensagens de e para filas do Amazon SQS.

A biblioteca oferece suporte ao envio e recebimento de mensagens para uma fila (o modelo ponto a ponto JMS) de acordo com a [especificação JMS 1.1](#). A biblioteca oferece suporte ao envio de mensagens de texto, de byte ou de objeto de forma síncrona para filas do Amazon SQS. A biblioteca também dá suporte ao recebimento de objetos de forma síncrona ou assíncrona.

Para mais informações sobre recursos da biblioteca de mensagens Java do Amazon SQS compatíveis com a especificação JMS 1.1, consulte [Implementações do Amazon SQS compatíveis com o JMS 1.1](#) e as [Perguntas frequentes do Amazon SQS](#).

Pré-requisitos para trabalhar com o JMS e o Amazon SQS

Antes de começar, você deve cumprir os seguintes pré-requisitos:

- SDK para Java

Há duas maneiras de incluir o SDK for Java no seu projeto:

- Faça download e instale o SDK for Java.
- Use o Maven para obter a biblioteca de mensagens Java do Amazon SQS.

Note

O SDK for Java é incluído como uma dependência.

O [SDK for Java](#) e a biblioteca cliente Java estendida para o Amazon SQS exigem o J2SE Development Kit 8.0 ou posterior.

Para obter mais informações sobre como fazer download do SDK for Java, consulte [SDK for Java](#).

- Biblioteca de Mensagens Java do Amazon SQS

Se você não usar o Maven, é necessário adicionar o pacote `amazon-sqs-java-messaging-lib.jar` ao caminho da classe Java. Para obter mais informações sobre como fazer download da biblioteca, consulte [Biblioteca de mensagens Java do Amazon SQS](#).

 Note

A biblioteca de mensagens Java do Amazon SQS inclui suporte para [Maven](#) e [Spring Framework](#).

Para exemplos de código que usam Maven, Spring Framework e a biblioteca de mensagens Java do Amazon SQS, consulte [Exemplos de Java funcionais para usar o JMS com filas padrão do Amazon SQS](#).

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-sqs-java-messaging-lib</artifactId>
  <version>1.0.4</version>
  <type>jar</type>
</dependency>
```

- Fila do Amazon SQS

Crie uma fila usando o Console de gerenciamento da AWS do Amazon SQS, a API `CreateQueue` ou o cliente encapsulado do Amazon SQS incluído na biblioteca de mensagens Java do Amazon SQS.

- Para obter mais informações sobre como criar uma fila com o Amazon SQS usando o Console de gerenciamento da AWS ou a API `CreateQueue`, [consulte Criar uma fila](#).
- Para obter mais informações sobre o uso da biblioteca de mensagens Java do Amazon SQS, consulte [Usar a Biblioteca de Mensagens Java do Amazon SQS](#).

Usar a Biblioteca de Mensagens Java do Amazon SQS

Para começar a usar o Serviço de Mensagens Java (JMS) com o Amazon SQS, use os exemplos de código nesta seção. As seções a seguir mostram como criar uma conexão e uma sessão do JMS, e como enviar e receber uma mensagem.

O objeto do cliente encapsulado do Amazon SQS incluído na biblioteca de mensagens Java do Amazon SQS verifica se uma fila do Amazon SQS existe. Se a fila não existir, o cliente a criará.

Criação de uma conexão do JMS

Antes de começar, consulte os pré-requisitos em [Pré-requisitos para trabalhar com o JMS e o Amazon SQS](#).

1. Crie uma connection factory e chame o método `createConnection` contra a factory.

```
// Create a new connection factory with all defaults (credentials and region) set
// automatically
SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
    new ProviderConfiguration(),
    AmazonSQSClientBuilder.defaultClient()
);

// Create the connection.
SQSConnection connection = connectionFactory.createConnection();
```

A classe `SQSConnection` estende `javax.jms.Connection`. Com os métodos de conexão do JMS padrão, `SQSConnection` oferece métodos adicionais, como `getAmazonSQSClient` e `getWrappedAmazonSQSClient`. Os dois métodos permitem que você execute operações administrativas não incluídas na especificação JMS, como a criação de novas filas. Contudo, o método `getWrappedAmazonSQSClient` também fornece uma versão encapsulada do cliente do Amazon SQS usada pela conexão atual. O wrapper transforma cada exceção de um cliente em um `JMSEException`, permitindo que ele seja mais facilmente usado pelo código existente que espera ocorrências de `JMSEException`.

2. Você pode usar objetos de cliente retornados de `getAmazonSQSClient` e de `getWrappedAmazonSQSClient` para executar operações administrativas não incluídas na especificação do JMS (por exemplo, você pode criar uma fila do Amazon SQS).

Se você tiver um código que espera exceções JMS, deve usar `getWrappedAmazonSQSClient`:

- Se você usar `getWrappedAmazonSQSClient`, o objeto de cliente retornado transformará todas as exceções em exceções JMS.
- Se você usar `getAmazonSQSClient`, todas as exceções serão exceções do Amazon SQS.

Criar uma fila do Amazon SQS

O objeto de cliente encapsulado verifica se uma fila do Amazon SQS existe.

Se a fila não existir, o cliente a criará. Se a fila existir, a função não retornará nada. Para obter mais informações, consulte a seção "Criar uma fila, se necessário" no exemplo [TextMessageSender.java](#).

Como criar uma fila padrão

```
// Get the wrapped client
AmazonSQSMessagingClientWrapper client = connection.getWrappedAmazonSQSClient();

// Create an SQS queue named MyQueue, if it doesn't already exist
if (!client.queueExists("MyQueue")) {
    client.createQueue("MyQueue");
}
```

Para criar uma fila FIFO

```
// Get the wrapped client
AmazonSQSMessagingClientWrapper client = connection.getWrappedAmazonSQSClient();

// Create an Amazon SQS FIFO queue named MyQueue.fifo, if it doesn't already exist
if (!client.queueExists("MyQueue.fifo")) {
    Map<String, String> attributes = new HashMap<String, String>();
    attributes.put("FifoQueue", "true");
    attributes.put("ContentBasedDeduplication", "true");
    client.createQueue(new
    CreateQueueRequest().withQueueName("MyQueue.fifo").withAttributes(attributes));
}
```

Note

O nome de uma fila FIFO deve terminar com o sufixo `.fifo`.

Para obter mais informações sobre o atributo `ContentBasedDeduplication`, consulte [Processamento exatamente uma vez no Amazon SQS](#).

Envio de mensagens de forma síncrona

1. Quando a conexão e a fila do Amazon SQS subjacente estiverem prontas, crie uma sessão JMS sem transação com o modo AUTO_ACKNOWLEDGE.

```
// Create the nontransacted session with AUTO_ACKNOWLEDGE mode
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
```

2. Para enviar uma mensagem de texto para a fila, crie uma identidade de fila JMS e um produtor de mensagem.

```
// Create a queue identity and specify the queue name to the session
Queue queue = session.createQueue("MyQueue");

// Create a producer for the 'MyQueue'
MessageProducer producer = session.createProducer(queue);
```

3. Crie uma mensagem de texto e envie-a para a fila.

- Para enviar uma mensagem para uma fila padrão, você não precisa definir parâmetros adicionais.

```
// Create the text message
TextMessage message = session.createTextMessage("Hello World!");

// Send the message
producer.send(message);
System.out.println("JMS Message " + message.getJMSMessageID());
```

- Para enviar uma mensagem para uma fila FIFO, você deve definir o ID do grupo de mensagens. Você também pode definir um ID de eliminação de duplicação de mensagem. Para obter mais informações, consulte [Termos-chave de fila FIFO do Amazon SQS](#).

```
// Create the text message
TextMessage message = session.createTextMessage("Hello World!");

// Set the message group ID
message.setStringProperty("JMSXGroupID", "Default");

// You can also set a custom message deduplication ID
// message.setStringProperty("JMS_SQS_DeduplicationId", "hello");
```

```
// Here, it's not needed because content-based deduplication is enabled for the
queue

// Send the message
producer.send(message);
System.out.println("JMS Message " + message.getJMSMessageID());
System.out.println("JMS Message Sequence Number " +
    message.getStringProperty("JMS_SQS_SequenceNumber"));
```

Recebimento de mensagens de forma síncrona

1. Para receber mensagens, crie um consumidor para a mesma fila e invoque o método `start`.

Você também pode chamar o método `start` na conexão a qualquer momento. No entanto, o consumidor não começa a receber mensagens até você chamá-lo.

```
// Create a consumer for the 'MyQueue'
MessageConsumer consumer = session.createConsumer(queue);
// Start receiving incoming messages
connection.start();
```

2. Chame o método `receive` no consumidor com um tempo limite definido como 1 segundo e imprima o conteúdo da mensagem recebida.

- Após receber uma mensagem de uma fila padrão, você pode acessar o conteúdo da mensagem.

```
// Receive a message from 'MyQueue' and wait up to 1 second
Message receivedMessage = consumer.receive(1000);

// Cast the received message as TextMessage and display the text
if (receivedMessage != null) {
    System.out.println("Received: " + ((TextMessage) receivedMessage).getText());
}
```

- Após receber uma mensagem de uma fila FIFO, você pode acessar o conteúdo da mensagem e outros atributos de mensagem específicos à FIFO, como o ID do grupo de mensagens, o ID de eliminação de duplicação de mensagens e o número de sequência. Para obter mais informações, consulte [Termos-chave de fila FIFO do Amazon SQS](#).

```
// Receive a message from 'MyQueue' and wait up to 1 second
Message receivedMessage = consumer.receive(1000);

// Cast the received message as TextMessage and display the text
if (receivedMessage != null) {
    System.out.println("Received: " + ((TextMessage) receivedMessage).getText());
    System.out.println("Group id: " +
        receivedMessage.getStringProperty("JMSXGroupID"));
    System.out.println("Message deduplication id: " +
        receivedMessage.getStringProperty("JMS_SQS_DeduplicationId"));
    System.out.println("Message sequence number: " +
        receivedMessage.getStringProperty("JMS_SQS_SequenceNumber"));
}
```

3. Feche a conexão e a sessão.

```
// Close the connection (and the session).
connection.close();
```

A saída será semelhante à seguinte:

```
JMS Message ID:8example-588b-44e5-bbcf-d816example2
Received: Hello World!
```

Note

Você pode usar o Spring Framework para inicializar esses objetos. Para obter mais informações, consulte `SpringExampleConfiguration.xml`, `SpringExample.java` e as outras classes auxiliares em `ExampleConfiguration.java` e `ExampleCommon.java` na seção [Exemplos de Java funcionais para usar o JMS com filas padrão do Amazon SQS](#).

Para exemplos completos de envio e recebimento de objetos, consulte [TextMessageSender.java](#) e [SyncMessageReceiver.java](#).

Recebimento de mensagens de forma assíncrona

No exemplo em [Usar a Biblioteca de Mensagens Java do Amazon SQS](#), uma mensagem é enviada para MyQueue e recebida de forma síncrona.

O exemplo a seguir mostra como receber as mensagens de forma assíncrona por meio de um listener.

1. Implemente a interface `MessageListener`.

```
class MyListener implements MessageListener {

    @Override
    public void onMessage(Message message) {
        try {
            // Cast the received message as TextMessage and print the text to
            screen.
            System.out.println("Received: " + ((TextMessage) message).getText());
        } catch (JMSEException e) {
            e.printStackTrace();
        }
    }
}
```

O método `onMessage` da interface `MessageListener` é chamado quando você recebe uma mensagem. Nesta implementação de listener, o texto armazenado na mensagem é impresso.

2. Em vez de explicitamente chamar o método `receive` no consumidor, defina o listener da mensagem do consumidor como uma instância da implementação `MyListener`. O thread principal aguarda um segundo.

```
// Create a consumer for the 'MyQueue'.
MessageConsumer consumer = session.createConsumer(queue);

// Instantiate and set the message listener for the consumer.
consumer.setMessageListener(new MyListener());

// Start receiving incoming messages.
connection.start();

// Wait for 1 second. The listener onMessage() method is invoked when a message is
received.
```



```
Thread.sleep(1000);
```

As demais etapas são idênticas às do exemplo [Usar a Biblioteca de Mensagens Java do Amazon SQS](#). Para um exemplo completo de um consumidor assíncrono, consulte `AsyncMessageReceiver.java` em [Exemplos de Java funcionais para usar o JMS com filas padrão do Amazon SQS](#).

A saída deste exemplo é similar ao seguinte:

```
JMS Message ID:8example-588b-44e5-bbcf-d816example2
Received: Hello World!
```

Uso do modo de reconhecimento do cliente

O exemplo em [Usar a Biblioteca de Mensagens Java do Amazon SQS](#) usa o modo `AUTO_ACKNOWLEDGE` em que cada mensagem recebida é confirmada automaticamente (e, portanto, excluída da fila do Amazon SQS subjacente).

1. Para explicitamente reconhecer as mensagens depois de processadas, você deve criar a sessão com o modo `CLIENT_ACKNOWLEDGE`.

```
// Create the non-transacted session with CLIENT_ACKNOWLEDGE mode.
Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
```

2. Quando a mensagem é recebida, exiba-a e confirme-a explicitamente.

```
// Cast the received message as TextMessage and print the text to screen. Also
// acknowledge the message.
if (receivedMessage != null) {
    System.out.println("Received: " + ((TextMessage) receivedMessage).getText());
    receivedMessage.acknowledge();
    System.out.println("Acknowledged: " + message.getJMSMessageID());
}
```

Note

Nesse modo, quando uma mensagem é confirmada, todas as mensagens recebidas antes desta mensagem são implicitamente confirmadas. Por exemplo, se 10 mensagens são recebidas e apenas a 10ª mensagem é reconhecida (na ordem em que as

mensagens são recebidas), todas as nove mensagens anteriores também são reconhecidas.

As demais etapas são idênticas às do exemplo [Usar a Biblioteca de Mensagens Java do Amazon SQS](#). Para um exemplo completo de um consumidor síncrono com modo de reconhecimento do cliente, consulte `SyncMessageReceiverClientAcknowledge.java` em [Exemplos de Java funcionais para usar o JMS com filas padrão do Amazon SQS](#).

A saída deste exemplo é similar ao seguinte:

```
JMS Message ID:4example-aa0e-403f-b6df-5e02example5
Received: Hello World!
Acknowledged: ID:4example-aa0e-403f-b6df-5e02example5
```

Uso do modo de reconhecimento não ordenado

Ao usar o modo `CLIENT_ACKNOWLEDGE`, todas as mensagens recebidas antes de uma mensagem explicitamente reconhecida são automaticamente reconhecidas. Para obter mais informações, consulte [Uso do modo de reconhecimento do cliente](#).

A biblioteca de mensagens Java do Amazon SQS fornece outro modo de confirmação. Ao usar o modo `UNORDERED_ACKNOWLEDGE`, todas as mensagens recebidas devem ser individual e explicitamente reconhecidas pelo cliente, independentemente de sua ordem de recebimento. Para fazer isso, cria uma sessão com o modo `UNORDERED_ACKNOWLEDGE`.

```
// Create the non-transacted session with UNORDERED_ACKNOWLEDGE mode.
Session session = connection.createSession(false, SQSSession.UNORDERED_ACKNOWLEDGE);
```

As etapas restantes são idênticas às do exemplo [Uso do modo de reconhecimento do cliente](#). Para um exemplo completo de um consumidor síncrono com o modo `UNORDERED_ACKNOWLEDGE`, consulte `SyncMessageReceiverUnorderedAcknowledge.java`.

Neste exemplo, a saída é similar ao seguinte:

```
JMS Message ID:dexample-73ad-4adb-bc6c-4357example7
Received: Hello World!
Acknowledged: ID:dexample-73ad-4adb-bc6c-4357example7
```

Usar o Serviço de Mensagens Java com outros clientes do Amazon SQS

Usar o cliente Java Message Service (JMS) do Amazon SQS com o AWS SDK limita o tamanho das mensagens do Amazon SQS em 256 KB. No entanto, você pode criar um provedor JMS usando qualquer cliente do Amazon SQS. Por exemplo, você pode usar o cliente JMS com a biblioteca cliente Java estendida para o Amazon SQS para enviar uma mensagem do Amazon SQS que contenha uma referência à carga útil da mensagem (até 2 GB) no Amazon S3. Para obter mais informações, consulte [Gerenciar mensagens grandes do Amazon SQS usando Java e Amazon S3](#).

O exemplo de código Java a seguir cria o provedor do JMS para a biblioteca do cliente em versão ampliada.

Veja os pré-requisitos em [Pré-requisitos para trabalhar com o JMS e o Amazon SQS](#) antes de testar este exemplo.

```
AmazonS3 s3 = new AmazonS3Client(credentials);
Region s3Region = Region.getRegion(Regions.US_WEST_2);
s3.setRegion(s3Region);

// Set the Amazon S3 bucket name, and set a lifecycle rule on the bucket to
// permanently delete objects a certain number of days after each object's creation
// date.
// Next, create the bucket, and enable message objects to be stored in the bucket.
BucketLifecycleConfiguration.Rule expirationRule = new
    BucketLifecycleConfiguration.Rule();
expirationRule.withExpirationInDays(14).withStatus("Enabled");
BucketLifecycleConfiguration lifecycleConfig = new
    BucketLifecycleConfiguration().withRules(expirationRule);

s3.createBucket(s3BucketName);
s3.setBucketLifecycleConfiguration(s3BucketName, lifecycleConfig);
System.out.println("Bucket created and configured.");

// Set the SQS extended client configuration with large payload support enabled.
ExtendedClientConfiguration extendedClientConfig = new ExtendedClientConfiguration()
    .withLargePayloadSupportEnabled(s3, s3BucketName);

AmazonSQS sqsExtended = new AmazonSQSExtendedClient(new AmazonSQSClient(credentials),
    extendedClientConfig);
Region sqsRegion = Region.getRegion(Regions.US_WEST_2);
```

```
sqsExtended.setRegion(sqsRegion);
```

O exemplo de código Java a seguir cria a connection factory:

```
// Create the connection factory using the environment variable credential provider.  
// Pass the configured Amazon SQS Extended Client to the JMS connection factory.  
SQSConnectionFactory connectionFactory = new SQSConnectionFactory(  
    new ProviderConfiguration(),  
    sqsExtended  
);  
  
// Create the connection.  
SQSConnection connection = connectionFactory.createConnection();
```

Exemplos de Java funcionais para usar o JMS com filas padrão do Amazon SQS

Veja a seguir exemplos de código que mostram como usar o JMS (Java Message Service) com filas padrão do Amazon SQS. Para obter mais informações sobre como trabalhar com filas FIFO, consulte [Para criar uma fila FIFO](#), [Envio de mensagens de forma síncrona](#) e [Recebimento de mensagens de forma síncrona](#). (O recebimento de mensagens de forma síncrona é igual para filas padrão e FIFO. No entanto, as mensagens em filas FIFO contêm mais atributos).

Veja os pré-requisitos em [Pré-requisitos para trabalhar com o JMS e o Amazon SQS](#) antes de testar os exemplos a seguir.

ExampleConfiguration.java

O exemplo a seguir de código Java do SDK v 1.x define o nome de fila padrão, a região e as credenciais a serem usadas com outros exemplos de Java.

```
/*  
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License.  
 * A copy of the License is located at  
 *  
 * https://aws.amazon.com/apache2.0  
 */
```

```

* or in the "license" file accompanying this file. This file is distributed
* on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
* express or implied. See the License for the specific language governing
* permissions and limitations under the License.
*
*/

public class ExampleConfiguration {
    public static final String DEFAULT_QUEUE_NAME = "SQSJMSClientExampleQueue";

    public static final Region DEFAULT_REGION = Region.getRegion(Regions.US_EAST_2);

    private static String getParameter( String args[], int i ) {
        if( i + 1 >= args.length ) {
            throw new IllegalArgumentException( "Missing parameter for " + args[i] );
        }
        return args[i+1];
    }

    /**
     * Parse the command line and return the resulting config. If the config parsing
     fails
     * print the error and the usage message and then call System.exit
     *
     * @param app the app to use when printing the usage string
     * @param args the command line arguments
     * @return the parsed config
     */
    public static ExampleConfiguration parseConfig(String app, String args[]) {
        try {
            return new ExampleConfiguration(args);
        } catch (IllegalArgumentException e) {
            System.err.println( "ERROR: " + e.getMessage() );
            System.err.println();
            System.err.println( "Usage: " + app + " [--queue <queue>] [--region
<region>] [--credentials <credentials>] ");
            System.err.println( "    or" );
            System.err.println( "    " + app + " <spring.xml>" );
            System.exit(-1);
            return null;
        }
    }

    private ExampleConfiguration(String args[]) {

```

```

        for( int i = 0; i < args.length; ++i ) {
            String arg = args[i];
            if( arg.equals( "--queue" ) ) {
                setQueueName(getParameter(args, i));
                i++;
            } else if( arg.equals( "--region" ) ) {
                String regionName = getParameter(args, i);
                try {
                    setRegion(Region.getRegion(Regions.fromName(regionName)));
                } catch( IllegalArgumentException e ) {
                    throw new IllegalArgumentException( "Unrecognized region " +
regionName );
                }
                i++;
            } else if( arg.equals( "--credentials" ) ) {
                String credsFile = getParameter(args, i);
                try {
                    setCredentialsProvider( new
PropertiesFileCredentialsProvider(credsFile) );
                } catch (AmazonClientException e) {
                    throw new IllegalArgumentException("Error reading credentials from
" + credsFile, e );
                }
                i++;
            } else {
                throw new IllegalArgumentException("Unrecognized option " + arg);
            }
        }
    }

    private String queueName = DEFAULT_QUEUE_NAME;
    private Region region = DEFAULT_REGION;
    private AWSCredentialsProvider credentialsProvider = new
DefaultAWSCredentialsProviderChain();

    public String getQueueName() {
        return queueName;
    }

    public void setQueueName(String queueName) {
        this.queueName = queueName;
    }

    public Region getRegion() {

```

```
        return region;
    }

    public void setRegion(Region region) {
        this.region = region;
    }

    public AWSCredentialsProvider getCredentialsProvider() {
        return credentialsProvider;
    }

    public void setCredentialsProvider(AWSCredentialsProvider credentialsProvider) {
        // Make sure they're usable first
        credentialsProvider.getCredentials();
        this.credentialsProvider = credentialsProvider;
    }
}
```

TextMessageSender.java

O seguinte exemplo de código Java cria um produtor de mensagem de texto.

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 * https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

public class TextMessageSender {
    public static void main(String args[]) throws JMSEException {
        ExampleConfiguration config =
            ExampleConfiguration.parseConfig("TextMessageSender", args);
    }
}
```

```
ExampleCommon.setupLogging();

// Create the connection factory based on the config
SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
    new ProviderConfiguration(),
    AmazonSQSClientBuilder.standard()
        .withRegion(config.getRegion().getName())
        .withCredentials(config.getCredentialsProvider())
    );

// Create the connection
SQSConnection connection = connectionFactory.createConnection();

// Create the queue if needed
ExampleCommon.ensureQueueExists(connection, config.getQueueName());

// Create the session
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
MessageProducer producer =
session.createProducer( session.createQueue( config.getQueueName() ) );

sendMessages(session, producer);

// Close the connection. This closes the session automatically
connection.close();
System.out.println( "Connection closed" );
}

private static void sendMessages( Session session, MessageProducer producer ) {
    BufferedReader inputReader = new BufferedReader(
        new InputStreamReader( System.in, Charset.defaultCharset() ) );

    try {
        String input;
        while( true ) {
            System.out.print( "Enter message to send (leave empty to exit): " );
            input = inputReader.readLine();
            if( input == null || input.equals("") ) break;

            TextMessage message = session.createTextMessage(input);
            producer.send(message);
            System.out.println( "Send message " + message.getJMSMessageID() );
        }
    } catch (EOFException e) {
```



```

        // Just return on EOF
    } catch (IOException e) {
        System.err.println( "Failed reading input: " + e.getMessage() );
    } catch (JMSEException e) {
        System.err.println( "Failed sending message: " + e.getMessage() );
        e.printStackTrace();
    }
}
}

```

SyncMessageReceiver.java

O seguinte exemplo de código Java cria um consumidor de mensagem síncrona.

```

/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 * https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

public class SyncMessageReceiver {
    public static void main(String args[]) throws JMSEException {
        ExampleConfiguration config =
            ExampleConfiguration.parseConfig("SyncMessageReceiver", args);

        ExampleCommon.setupLogging();

        // Create the connection factory based on the config
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
            new ProviderConfiguration(),
            AmazonSQSClientBuilder.standard()
                .withRegion(config.getRegion().getName())
                .withCredentials(config.getCredentialsProvider())

```

```
        );

        // Create the connection
        SQSConnection connection = connectionFactory.createConnection();

        // Create the queue if needed
        ExampleCommon.ensureQueueExists(connection, config.getQueueName());

        // Create the session
        Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
        MessageConsumer consumer =
session.createConsumer( session.createQueue( config.getQueueName() ) );

        connection.start();

        receiveMessages(session, consumer);

        // Close the connection. This closes the session automatically
        connection.close();
        System.out.println( "Connection closed" );
    }

    private static void receiveMessages( Session session, MessageConsumer consumer ) {
        try {
            while( true ) {
                System.out.println( "Waiting for messages");
                // Wait 1 minute for a message
                Message message = consumer.receive(TimeUnit.MINUTES.toMillis(1));
                if( message == null ) {
                    System.out.println( "Shutting down after 1 minute of silence" );
                    break;
                }
                ExampleCommon.handleMessage(message);
                message.acknowledge();
                System.out.println( "Acknowledged message " + message.getJMSMessageID() );
            }
        } catch (JMSEException e) {
            System.err.println( "Error receiving from SQS: " + e.getMessage() );
            e.printStackTrace();
        }
    }
}
```

AsyncMessageReceiver.java

O seguinte exemplo de código Java cria um consumidor de mensagem assíncrona.

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 * https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

public class AsyncMessageReceiver {
    public static void main(String args[]) throws JMSEException, InterruptedException {
        ExampleConfiguration config =
            ExampleConfiguration.parseConfig("AsyncMessageReceiver", args);

        ExampleCommon.setupLogging();

        // Create the connection factory based on the config
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
            new ProviderConfiguration(),
            AmazonSQSClientBuilder.standard()
                .withRegion(config.getRegion().getName())
                .withCredentials(config.getCredentialsProvider())
        );

        // Create the connection
        SQSConnection connection = connectionFactory.createConnection();

        // Create the queue if needed
        ExampleCommon.ensureQueueExists(connection, config.getQueueName());

        // Create the session
        Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
```

```

        MessageConsumer consumer =
session.createConsumer( session.createQueue( config.getQueueName() ) );

        // No messages are processed until this is called
connection.start();

        ReceiverCallback callback = new ReceiverCallback();
consumer.setMessageListener( callback );

        callback.waitForOneMinuteOfSilence();
System.out.println( "Returning after one minute of silence" );

        // Close the connection. This closes the session automatically
connection.close();
System.out.println( "Connection closed" );
    }

private static class ReceiverCallback implements MessageListener {
    // Used to listen for message silence
    private volatile long timeOfLastMessage = System.nanoTime();

    public void waitForOneMinuteOfSilence() throws InterruptedException {
        for(;;) {
            long timeSinceLastMessage = System.nanoTime() - timeOfLastMessage;
            long remainingTillOneMinuteOfSilence =
                TimeUnit.MINUTES.toNanos(1) - timeSinceLastMessage;
            if( remainingTillOneMinuteOfSilence < 0 ) {
                break;
            }
            TimeUnit.NANOSECONDS.sleep(remainingTillOneMinuteOfSilence);
        }
    }

    @Override
    public void onMessage(Message message) {
        try {
            ExampleCommon.handleMessage(message);
            message.acknowledge();
            System.out.println( "Acknowledged message " +
message.getJMSMessageID() );
            timeOfLastMessage = System.nanoTime();
        } catch (JMSEException e) {

```

```

        System.err.println( "Error processing message: " + e.getMessage() );
        e.printStackTrace();
    }
}
}
}

```

SyncMessageReceiverClientAcknowledge.java

O seguinte exemplo de código Java cria um consumidor síncrono com modo de reconhecimento do cliente.

```

/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 * https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

/**
 * An example class to demonstrate the behavior of CLIENT_ACKNOWLEDGE mode for received
 * messages. This example
 * complements the example given in {@link SyncMessageReceiverUnorderedAcknowledge} for
 * UNORDERED_ACKNOWLEDGE mode.
 *
 * First, a session, a message producer, and a message consumer are created. Then, two
 * messages are sent. Next, two messages
 * are received but only the second one is acknowledged. After waiting for the
 * visibility time out period, an attempt to
 * receive another message is made. It's shown that no message is returned for this
 * attempt since in CLIENT_ACKNOWLEDGE mode,
 * as expected, all the messages prior to the acknowledged messages are also
 * acknowledged.
 */

```

```
* This ISN'T the behavior for UNORDERED_ACKNOWLEDGE mode. Please see {@link
SyncMessageReceiverUnorderedAcknowledge}
* for an example.
*/
public class SyncMessageReceiverClientAcknowledge {

    // Visibility time-out for the queue. It must match to the one set for the queue
    for this example to work.
    private static final long TIME_OUT_SECONDS = 1;

    public static void main(String args[]) throws JMSEException, InterruptedException {
        // Create the configuration for the example
        ExampleConfiguration config =
        ExampleConfiguration.parseConfig("SyncMessageReceiverClientAcknowledge", args);

        // Setup logging for the example
        ExampleCommon.setupLogging();

        // Create the connection factory based on the config
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
            new ProviderConfiguration(),
            AmazonSQSClientBuilder.standard()
                .withRegion(config.getRegion().getName())
                .withCredentials(config.getCredentialsProvider())
        );

        // Create the connection
        SQSConnection connection = connectionFactory.createConnection();

        // Create the queue if needed
        ExampleCommon.ensureQueueExists(connection, config.getQueueName());

        // Create the session with client acknowledge mode
        Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);

        // Create the producer and consume
        MessageProducer producer =
        session.createProducer(session.createQueue(config.getQueueName()));
        MessageConsumer consumer =
        session.createConsumer(session.createQueue(config.getQueueName()));

        // Open the connection
        connection.start();
    }
}
```

```

        // Send two text messages
        sendMessage(producer, session, "Message 1");
        sendMessage(producer, session, "Message 2");

        // Receive a message and don't acknowledge it
        receiveMessage(consumer, false);

        // Receive another message and acknowledge it
        receiveMessage(consumer, true);

        // Wait for the visibility time out, so that unacknowledged messages reappear
in the queue
        System.out.println("Waiting for visibility timeout...");
        Thread.sleep(TimeUnit.SECONDS.toMillis(TIME_OUT_SECONDS));

        // Attempt to receive another message and acknowledge it. This results in
receiving no messages since
        // we have acknowledged the second message. Although we didn't explicitly
acknowledge the first message,
        // in the CLIENT_ACKNOWLEDGE mode, all the messages received prior to the
explicitly acknowledged message
        // are also acknowledged. Therefore, we have implicitly acknowledged the first
message.
        receiveMessage(consumer, true);

        // Close the connection. This closes the session automatically
        connection.close();
        System.out.println("Connection closed.");
    }

    /**
     * Sends a message through the producer.
     *
     * @param producer Message producer
     * @param session Session
     * @param messageText Text for the message to be sent
     * @throws JMSEException
     */
    private static void sendMessage(MessageProducer producer, Session session, String
messageText) throws JMSEException {
        // Create a text message and send it
        producer.send(session.createTextMessage(messageText));
    }

```

```

/**
 * Receives a message through the consumer synchronously with the default timeout
 (TIME_OUT_SECONDS).
 * If a message is received, the message is printed. If no message is received,
 "Queue is empty!" is
 * printed.
 *
 * @param consumer Message consumer
 * @param acknowledge If true and a message is received, the received message is
 acknowledged.
 * @throws JMSEException
 */
private static void receiveMessage(MessageConsumer consumer, boolean acknowledge)
throws JMSEException {
    // Receive a message
    Message message =
consumer.receive(TimeUnit.SECONDS.toMillis(TIME_OUT_SECONDS));

    if (message == null) {
        System.out.println("Queue is empty!");
    } else {
        // Since this queue has only text messages, cast the message object and
print the text
        System.out.println("Received: " + ((TextMessage) message).getText());

        // Acknowledge the message if asked
        if (acknowledge) message.acknowledge();
    }
}
}

```

SyncMessageReceiverUnorderedAcknowledge.java

O seguinte exemplo de código Java cria um consumidor síncrono com modo de reconhecimento não ordenado.

```

/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *

```



```

* https://aws.amazon.com/apache2.0
*
* or in the "license" file accompanying this file. This file is distributed
* on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
* express or implied. See the License for the specific language governing
* permissions and limitations under the License.
*
*/

/**
 * An example class to demonstrate the behavior of UNORDERED_ACKNOWLEDGE mode for
 * received messages. This example
 * complements the example given in {@link SyncMessageReceiverClientAcknowledge} for
 * CLIENT_ACKNOWLEDGE mode.
 *
 * First, a session, a message producer, and a message consumer are created. Then, two
 * messages are sent. Next, two messages
 * are received but only the second one is acknowledged. After waiting for the
 * visibility time out period, an attempt to
 * receive another message is made. It's shown that the first message received in the
 * prior attempt is returned again
 * for the second attempt. In UNORDERED_ACKNOWLEDGE mode, all the messages must be
 * explicitly acknowledged no matter what
 * the order they're received.
 *
 * This ISN'T the behavior for CLIENT_ACKNOWLEDGE mode. Please see {@link
 * SyncMessageReceiverClientAcknowledge}
 * for an example.
 */
public class SyncMessageReceiverUnorderedAcknowledge {

    // Visibility time-out for the queue. It must match to the one set for the queue
    // for this example to work.
    private static final long TIME_OUT_SECONDS = 1;

    public static void main(String args[]) throws JMSEException, InterruptedException {
        // Create the configuration for the example
        ExampleConfiguration config =
        ExampleConfiguration.parseConfig("SyncMessageReceiverUnorderedAcknowledge", args);

        // Setup logging for the example
        ExampleCommon.setupLogging();

        // Create the connection factory based on the config

```

```
    SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
        new ProviderConfiguration(),
        AmazonSQSClientBuilder.standard()
            .withRegion(config.getRegion().getName())
            .withCredentials(config.getCredentialsProvider())
    );

    // Create the connection
    SQSConnection connection = connectionFactory.createConnection();

    // Create the queue if needed
    ExampleCommon.ensureQueueExists(connection, config.getQueueName());

    // Create the session with unordered acknowledge mode
    Session session = connection.createSession(false,
        SQSSession.UNORDERED_ACKNOWLEDGE);

    // Create the producer and consume
    MessageProducer producer =
    session.createProducer(session.createQueue(config.getQueueName()));
    MessageConsumer consumer =
    session.createConsumer(session.createQueue(config.getQueueName()));

    // Open the connection
    connection.start();

    // Send two text messages
    sendMessage(producer, session, "Message 1");
    sendMessage(producer, session, "Message 2");

    // Receive a message and don't acknowledge it
    receiveMessage(consumer, false);

    // Receive another message and acknowledge it
    receiveMessage(consumer, true);

    // Wait for the visibility time out, so that unacknowledged messages reappear
    in the queue
    System.out.println("Waiting for visibility timeout...");
    Thread.sleep(TimeUnit.SECONDS.toMillis(TIME_OUT_SECONDS));

    // Attempt to receive another message and acknowledge it. This results in
    receiving the first message since
```

```
// we have acknowledged only the second message. In the UNORDERED_ACKNOWLEDGE
mode, all the messages must
// be explicitly acknowledged.
receiveMessage(consumer, true);

// Close the connection. This closes the session automatically
connection.close();
System.out.println("Connection closed.");
}

/**
 * Sends a message through the producer.
 *
 * @param producer Message producer
 * @param session Session
 * @param messageText Text for the message to be sent
 * @throws JMSEException
 */
private static void sendMessage(MessageProducer producer, Session session, String
messageText) throws JMSEException {
    // Create a text message and send it
    producer.send(session.createTextMessage(messageText));
}

/**
 * Receives a message through the consumer synchronously with the default timeout
 (TIME_OUT_SECONDS).
 * If a message is received, the message is printed. If no message is received,
 "Queue is empty!" is
 * printed.
 *
 * @param consumer Message consumer
 * @param acknowledge If true and a message is received, the received message is
 acknowledged.
 * @throws JMSEException
 */
private static void receiveMessage(MessageConsumer consumer, boolean acknowledge)
throws JMSEException {
    // Receive a message
    Message message =
consumer.receive(TimeUnit.SECONDS.toMillis(TIME_OUT_SECONDS));

    if (message == null) {
        System.out.println("Queue is empty!");
    }
}
```

```
    } else {  
        // Since this queue has only text messages, cast the message object and  
        print the text  
        System.out.println("Received: " + ((TextMessage) message).getText());  
  
        // Acknowledge the message if asked  
        if (acknowledge) message.acknowledge();  
    }  
}  
}
```

SpringExampleConfiguration.xml

O seguinte exemplo de código XML é um arquivo de configuração bean para [SpringExample.java](#).

```
<!--  
    Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
  
    Licensed under the Apache License, Version 2.0 (the "License").  
    You may not use this file except in compliance with the License.  
    A copy of the License is located at  
  
    https://aws.amazon.com/apache2.0  
  
    or in the "license" file accompanying this file. This file is distributed  
    on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either  
    express or implied. See the License for the specific language governing  
    permissions and limitations under the License.  
-->  
  
<?xml version="1.0" encoding="UTF-8"?>  
<beans  
    xmlns="http://www.springframework.org/schema/beans"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xmlns:util="http://www.springframework.org/schema/util"  
    xmlns:p="http://www.springframework.org/schema/p"  
    xsi:schemaLocation="  
        http://www.springframework.org/schema/beans http://www.springframework.org/  
schema/beans/spring-beans-3.0.xsd  
        http://www.springframework.org/schema/util http://www.springframework.org/  
schema/util/spring-util-3.0.xsd  
    ">
```

```
<bean id="CredentialsProviderBean"
class="com.amazonaws.auth.DefaultAWSCredentialsProviderChain"/>

<bean id="ClientBuilder" class="com.amazonaws.services.sqs.AmazonSQSClientBuilder"
factory-method="standard">
    <property name="region" value="us-east-2"/>
    <property name="credentials" ref="CredentialsProviderBean"/>
</bean>

<bean id="ProviderConfiguration"
class="com.amazon.sqs.javamessaging.ProviderConfiguration">
    <property name="numberOfMessagesToPrefetch" value="5"/>
</bean>

<bean id="ConnectionFactory"
class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
    <constructor-arg ref="ProviderConfiguration" />
    <constructor-arg ref="ClientBuilder" />
</bean>

<bean id="Connection" class="javax.jms.Connection"
    factory-bean="ConnectionFactory"
    factory-method="createConnection"
    init-method="start"
    destroy-method="close" />

<bean id="QueueName" class="java.lang.String">
    <constructor-arg value="SQSJMSClientExampleQueue"/>
</bean>
</beans>
```

SpringExample.java

O seguinte exemplo de código Java usa o arquivo de configuração bean para inicializar seus objetos.

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 * https://aws.amazon.com/apache2.0
```

```
*
* or in the "license" file accompanying this file. This file is distributed
* on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
* express or implied. See the License for the specific language governing
* permissions and limitations under the License.
*
*/

public class SpringExample {
    public static void main(String args[]) throws JMSEException {
        if( args.length != 1 || !args[0].endsWith(".xml")) {
            System.err.println( "Usage: " + SpringExample.class.getName() + " <spring
config.xml>" );
            System.exit(1);
        }

        File springFile = new File( args[0] );
        if( !springFile.exists() || !springFile.canRead() ) {
            System.err.println( "File " + args[0] + " doesn't exist or isn't
readable." );
            System.exit(2);
        }

        ExampleCommon.setupLogging();

        FileSystemXmlApplicationContext context =
            new FileSystemXmlApplicationContext( "file://" +
springFile.getAbsolutePath() );

        Connection connection;
        try {
            connection = context.getBean(Connection.class);
        } catch( NoSuchBeanDefinitionException e ) {
            System.err.println( "Can't find the JMS connection to use: " +
e.getMessage() );
            System.exit(3);
            return;
        }

        String queueName;
        try {
            queueName = context.getBean("QueueName", String.class);
        } catch( NoSuchBeanDefinitionException e ) {
```

```
        System.err.println( "Can't find the name of the queue to use: " +
e.getMessage() );
        System.exit(3);
        return;
    }

    if( connection instanceof SQSConnection ) {
        ExampleCommon.ensureQueueExists( (SQSConnection) connection, queueName );
    }

    // Create the session
    Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
    MessageConsumer consumer =
session.createConsumer( session.createQueue( queueName ) );

    receiveMessages(session, consumer);

    // The context can be setup to close the connection for us
    context.close();
    System.out.println( "Context closed" );
}

private static void receiveMessages( Session session, MessageConsumer consumer ) {
    try {
        while( true ) {
            System.out.println( "Waiting for messages");
            // Wait 1 minute for a message
            Message message = consumer.receive(TimeUnit.MINUTES.toMillis(1));
            if( message == null ) {
                System.out.println( "Shutting down after 1 minute of silence" );
                break;
            }
            ExampleCommon.handleMessage(message);
            message.acknowledge();
            System.out.println( "Acknowledged message" );
        }
    } catch (JMSEException e) {
        System.err.println( "Error receiving from SQS: " + e.getMessage() );
        e.printStackTrace();
    }
}
}
```

ExampleCommon.java

O código de exemplo Java a seguir verifica se existe uma fila do Amazon SQS e, se não existir, cria uma. Ele também inclui código de registro de exemplo.

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 * https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

public class ExampleCommon {
    /**
     * A utility function to check the queue exists and create it if needed. For most
     * use cases this is usually done by an administrator before the application is
     * run.
     */
    public static void ensureQueueExists(SQSConnection connection, String queueName)
        throws JMSEException {
        AmazonSQSMessagingClientWrapper client =
            connection.getWrappedAmazonSQSClient();

        /**
         * In most cases, you can do this with just a createQueue call, but
         * GetQueueUrl
         * (called by queueExists) is a faster operation for the common case where the
         * queue
         * already exists. Also many users and roles have permission to call
         * GetQueueUrl
         * but don't have permission to call CreateQueue.
         */
        if( !client.queueExists(queueName) ) {
            client.createQueue( queueName );
        }
    }
}
```



```

    }
}

public static void setupLogging() {
    // Setup logging
    BasicConfigurator.configure();
    Logger.getRootLogger().setLevel(Level.WARN);
}

public static void handleMessage(Message message) throws JMSEException {
    System.out.println( "Got message " + message.getJMSMessageID() );
    System.out.println( "Content: " );
    if( message instanceof TextMessage ) {
        TextMessage txtMessage = ( TextMessage ) message;
        System.out.println( "\t" + txtMessage.getText() );
    } else if( message instanceof BytesMessage ){
        BytesMessage byteMessage = ( BytesMessage ) message;
        // Assume the length fits in an int - SQS only supports sizes up to 256k so
that
        // should be true
        byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
        byteMessage.readBytes(bytes);
        System.out.println( "\t" + Base64.encodeAsString( bytes ) );
    } else if( message instanceof ObjectMessage ) {
        ObjectMessage objMessage = (ObjectMessage) message;
        System.out.println( "\t" + objMessage.getObject() );
    }
}
}

```

Implementações do Amazon SQS compatíveis com o JMS 1.1

A biblioteca de mensagens Java do Amazon SQS oferece suporte às seguintes [implantações do JMS 1.1](#). Para mais informações sobre os recursos compatíveis e capacidade da biblioteca de mensagens Java do Amazon SQS, consulte as [Perguntas frequentes do Amazon SQS](#).

Interfaces comuns com suporte

- Connection
- ConnectionFactory
- Destination

- Session
- MessageConsumer
- MessageProducer

Tipos de mensagens com suporte

- ByteMessage
- ObjectMessage
- TextMessage

Modos de reconhecimento de mensagens com suporte

- AUTO_ACKNOWLEDGE
- CLIENT_ACKNOWLEDGE
- DUPS_OK_ACKNOWLEDGE
- UNORDERED_ACKNOWLEDGE

Note

O modo UNORDERED_ACKNOWLEDGE não faz parte da especificação JMS 1.1. Esse modo ajuda o Amazon SQS a permitir que um cliente JMS reconheça explicitamente uma mensagem.

Cabeçalhos definidos pelo JMS e propriedades reservadas

Para enviar mensagens

Ao enviar mensagens, você pode definir os seguintes cabeçalhos e propriedades para cada mensagem:

- JMSXGroupID (obrigatório para filas FIFO, não permitido para filas padrão)
- JMS_SQS_DeduplicationId (opcional para filas FIFO, não permitido para filas padrão)

Quando você envia mensagens, o Amazon SQS define os seguintes cabeçalhos e propriedades para cada uma:

- `JMSMessageID`
- `JMS_SQS_SequenceNumber` (somente para filas FIFO)

Para receber mensagens

Quando você recebe mensagens, o Amazon SQS define os seguintes cabeçalhos e propriedades para cada uma:

- `JMSDestination`
- `JMSMessageID`
- `JMSRedelivered`
- `JMSXDeliveryCount`
- `JMSXGroupID` (somente para filas FIFO)
- `JMS_SQS_DeduplicationId` (somente para filas FIFO)
- `JMS_SQS_SequenceNumber` (somente para filas FIFO)

Tutoriais do Amazon SQS

Este tópico fornece tutoriais que ajudam a explorar os recursos e funcionalidades do Amazon SQS.

Tutoriais

- [Criar uma fila do Amazon SQS usando o CloudFormation](#)
- [Tutorial: Envio de uma mensagem a uma fila do Amazon SQS pela Amazon Virtual Private Cloud](#)

Criar uma fila do Amazon SQS usando o CloudFormation

Use o console do CloudFormation e um modelo JSON ou YAML para criar uma fila do Amazon SQS. Para obter mais detalhes, consulte [Trabalhar com modelos do CloudFormation](#) e o [recurso AWS::SQS::Queue](#) no Guia do usuário do AWS CloudFormation.

Para usar o CloudFormation a fim de criar uma fila do Amazon SQS.

1. Copie o seguinte código JSON em um arquivo denominado `MyQueue.json`. Para criar uma fila padrão, omita as propriedades `FifoQueue` e `ContentBasedDeduplication`. Para obter mais informações sobre a eliminação de duplicação baseada em conteúdo, consulte [Processamento exatamente uma vez no Amazon SQS](#).

Note

O nome de uma fila FIFO deve terminar com o sufixo `.fifo`.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "MyQueue": {
      "Properties": {
        "QueueName": "MyQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": true
      },
      "Type": "AWS::SQS::Queue"
    }
  },
}
```

```

"Outputs": {
  "QueueName": {
    "Description": "The name of the queue",
    "Value": {
      "Fn::GetAtt": [
        "MyQueue",
        "QueueName"
      ]
    }
  },
  "QueueURL": {
    "Description": "The URL of the queue",
    "Value": {
      "Ref": "MyQueue"
    }
  },
  "QueueARN": {
    "Description": "The ARN of the queue",
    "Value": {
      "Fn::GetAtt": [
        "MyQueue",
        "Arn"
      ]
    }
  }
}
}

```

2. Faça login no [console do CloudFormation](#) e, em seguida, selecione Create Stack (Criar pilha).
3. No painel Specify Template (Especificar modelo), escolha Upload a template file (Fazer upload de um arquivo de modelo), selecione o arquivo MyQueue.json e escolha Next (Próximo).
4. Na página Specify Details, digite MyQueue em Stack Name e escolha Next.
5. Na página Options (Opções), escolha Next (Avançar).
6. Na página Revisar, escolha Criar.

O CloudFormation começa a criar a pilha MyQueue e exibe o status CREATE_IN_PROGRESS. Quando o processo é concluído, o CloudFormation exibe o status CREATE_COMPLETE.

Filter: Active ▾		By Stack Name		Showing 1 stack	
	Stack Name	Created Time	Status	Description	
☒	MyQueue	2017-02-20 11:39:47 UTC-0800	CREATE_COMPLETE		

7. (Opcional) Para exibir o nome, a URL e o nome de recurso da Amazon (ARN) da fila, escolha o nome da pilha e, em seguida, na próxima página, expanda a seção Outputs.

Tutorial: Envio de uma mensagem a uma fila do Amazon SQS pela Amazon Virtual Private Cloud

Este tutorial mostra como enviar mensagens para uma fila do Amazon SQS por uma rede privada e segura. A rede inclui:

- Uma VPC contendo uma instância do Amazon EC2.
- Uma interface endpoint da VPC, que permite que a instância do Amazon EC2 se conecte ao Amazon SQS sem usar a Internet pública.

Mesmo em uma rede totalmente privada, você pode se conectar à instância do Amazon EC2 e enviar mensagens para a fila do Amazon SQS. Para obter mais informações, consulte [Endpoints da Amazon Virtual Private Cloud para o Amazon SQS](#).

Important

- Você pode usar a Amazon Virtual Private Cloud somente com endpoints HTTPS do Amazon SQS.
- Ao configurar o Amazon SQS para enviar mensagens pela Amazon VPC, habilite o DNS privado e especifique endpoints no formato `sqs.us-east-2.amazonaws.com` ou `sqs.us-east-2.api.aws` para o endpoint de pilha dupla.
- O Amazon SQS também oferece suporte a endpoints FIPS por meio do PrivateLink usando o serviço de endpoint `com.amazonaws.region.sqs-fips`. Você pode se conectar aos endpoints FIPS no formato `sqs-fips.region.amazonaws.com`.
- Ao usar o endpoint de pilha dupla na Amazon Virtual Private Cloud, as solicitações serão enviadas usando IPv4 e IPv6.
- O DNS privado não oferece suporte a endpoints legados, como `queue.amazonaws.com` ou `us-east-2.queue.amazonaws.com`.

Etapa 1: criar um par de chaves do Amazon EC2

Um par de chaves permite que você se conecte a uma instância do Amazon EC2. Esse par consiste em uma chave pública que criptografa suas informações de login e uma chave privada que as descriptografa.

1. Faça login no [console do Amazon EC2](#).
2. No menu de navegação, em Network & Security (Rede e segurança), selecione Key Pairs (Pares de chaves).
3. Escolha Criar par de chaves.
4. Na caixa de diálogo Create Key Pair (Criar par de chaves), para Key pair name (Nome do par de chaves), insira `SQS-VPCE-Tutorial-Key-Pair` e selecione Create (Criar).
5. O navegador faz download do arquivo da chave privada `SQS-VPCE-Tutorial-Key-Pair.pem` automaticamente.

Important

Salve esse arquivo em um lugar seguro. O EC2 não gera um segundo arquivo `.pem` para o mesmo par de chaves.

6. Para permitir que um cliente SSH se conecte à sua instância do EC2, defina as permissões do arquivo da chave privada para que somente o seu usuário possa ter permissões de leitura para ele, por exemplo:

```
chmod 400 SQS-VPCE-Tutorial-Key-Pair.pem
```

Etapa 2: criar recursos da AWS

Para configurar a infraestrutura necessária, é preciso usar um modelo do CloudFormation, que é um esquema para criar uma pilha composta por recursos da AWS, como instâncias do Amazon EC2 e filas do Amazon SQS.

A pilha deste tutorial inclui os seguintes recursos:

- Uma VPC e os recursos de rede associados, incluindo uma sub-rede, um grupo de segurança, um gateway da Internet e uma tabela de rotas.

- Uma instância do Amazon EC2 executada na sub-rede da VPC
- Uma fila do Amazon SQS

1. Faça download do modelo do CloudFormation chamado [SQS-VPCE-Tutorial-CloudFormation.yaml](#) do GitHub.
2. Faça login no [console do CloudFormation](#).
3. Escolha Create Stack (Criar pilha).
4. Na página Select Template (Selecionar modelo), selecione Upload a template to Amazon S3 (Fazer upload de um modelo no Amazon S3), selecione o arquivo SQS-VPCE-SQS-Tutorial-CloudFormation.yaml e, então, selecione Next (Próximo).
5. Na página Especificar detalhes, faça o seguinte:
 - a. Para Stack name (Nome da pilha), insira SQS-VPCE-Tutorial-Stack.
 - b. Para KeyName (Nome da chave), escolha SQS-VPCE-Tutorial-Key-Pair.
 - c. Escolha Próximo.
6. Na página Options (Opções), escolha Next (Avançar).
7. Na página Revisar, na seção Recursos, selecione Reconheço que o AWS CloudFormation pode criar recursos do IAM com nomes personalizados. e escolha Criar.

O CloudFormation começa a criar a pilha e exibe o status CREATE_IN_PROGRESS. Quando o processo é concluído, o CloudFormation exibe o status CREATE_COMPLETE.

Etapa 3: confirmar que sua instância do EC2 não é acessível publicamente

O modelo do CloudFormation executa uma instância do EC2 chamada SQS-VPCE-Tutorial-EC2-Instance na VPC. Essa instância do EC2 não permite tráfego de saída e não é capaz de enviar mensagens para o Amazon SQS. Para verificar isso, você deve se conectar à instância, tentar se conectar a um endpoint público e, então, tentar enviar uma mensagem para o Amazon SQS.

1. Faça login no [console do Amazon EC2](#).
2. No menu de navegação, em Instances (Instâncias), selecione Instances (Instâncias).
3. Selecione SQS-VPCE-Tutorial-EC2Instance.
4. Copie o nome de host em DNS público, por exemplo, ec2-203-0-113-0.us-west-2.compute.amazonaws.com.

5. A partir do diretório que contém [o par de chaves criado anteriormente](#), conecte-se à instância usando o comando a seguir, por exemplo:

```
ssh -i SQS-VPCE-Tutorial-Key-Pair.pem ec2-user@ec2-203-0-113-0.us-east-2.compute.amazonaws.com
```

6. Tente conectar-se a qualquer endpoint público, por exemplo:

```
ping amazon.com
```

A tentativa de conexão falha, conforme esperado.

7. Faça login no [console do Amazon SQS](#).
8. Na lista de filas, selecione a fila criada por seu modelo do CloudFormation, por exemplo, VPCE-SQS-Tutorial-Stack-CFQueue-1ABCDEFGH2IJK.
9. Na tabela Details (Detalhes), copie o URL, por exemplo, `https://sqs.us-east-2.amazonaws.com/123456789012/`.
10. A partir de sua instância do EC2, tente publicar uma mensagem na fila usando o comando a seguir, por exemplo:

```
aws sqs send-message --region us-east-2 --endpoint-url https://sqs.us-east-2.amazonaws.com/ --queue-url https://sqs.us-east-2.amazonaws.com/123456789012/ --message-body "Hello from Amazon SQS."
```

A tentativa de envio falha, conforme esperado.

Important

Posteriormente, ao criar um endpoint da VPC para o Amazon SQS, sua tentativa de envio será bem-sucedida.

Etapa 4: criar um endpoint da Amazon VPC para o Amazon SQS

Para conectar sua VPC ao Amazon SQS, você precisa definir uma interface de endpoint da VPC. Depois de adicionar o endpoint, você poderá usar a API do Amazon SQS a partir da instância do EC2 em sua VPC. Isso permite que você envie mensagens para uma fila dentro da rede da AWS sem precisar passar pela internet pública.

 Note

A instância do EC2 ainda não tem acesso a outros serviços e endpoints da AWS na internet.

1. Faça login no [console da Amazon VPC](#).
2. No menu de navegação, selecione Endpoints.
3. Escolha Criar endpoint.
4. Na página Create Endpoint (Criar endpoint), em Service Name (Nome do serviço), selecione o nome do serviço para o Amazon SQS.

 Note

Os nomes de serviços variam de acordo com a atual região da AWS. Por exemplo, se você estiver no Leste dos EUA (Ohio), o nome do serviço será com `amazonaws.us-east-2.sqs`.

5. Para VPC, selecione SQS-VPCE-Tutorial-VPC.
6. Para Subnets (Sub-redes), selecione a sub-rede cujo Subnet ID (ID da sub-rede) contenha SQS-VPCE-Tutorial-Subnet.
7. Para Security group (Grupo de segurança), selecione Select security groups (Selecionar grupos de segurança) e selecione o grupo de segurança cujo Group Name (Nome do grupo) contenha SQS VPCE Tutorial Security Group.
8. Escolha Criar endpoint.

O VPC endpoint de interface é criado e o ID dele é exibido, por exemplo, `vpce-0ab1cdef2ghi3j456k`.

9. Escolha Fechar.

O console da Amazon VPC abre a página Endpoints.

A Amazon VPC começa a criar o endpoint e exibe o status pending (pendente). Quando o processo é concluído, a Amazon VPC exibe o status available (disponível).

Etapa 5: enviar uma mensagem para sua fila do Amazon SQS

Agora que a VPC inclui um endpoint para o Amazon SQS, você pode se conectar à instância do EC2 e enviar mensagens para a fila.

1. Reconecte-se à instância do EC2, por exemplo:

```
ssh -i SQS-VPCE-Tutorial-Key-Pair.pem ec2-user@ec2-203-0-113-0.us-east-2.compute.amazonaws.com
```

2. Tente publicar uma mensagem na fila novamente usando o comando a seguir, por exemplo:

```
aws sqs send-message --region us-east-2 --endpoint-url https://sqs.us-east-2.amazonaws.com/ --queue-url https://sqs.us-east-2.amazonaws.com/123456789012/ --message-body "Hello from Amazon SQS."
```

A tentativa de envio é bem-sucedida e o resumo MD5 do corpo da mensagem e o ID da mensagem são exibidos, por exemplo:

```
{
  "MD5OfMessageBody": "a1bcd2ef3g45hi678j90klmn12p34qr5",
  "MessageId": "12345a67-8901-2345-bc67-d890123e45fg"
}
```

Para obter informações sobre como receber e excluir a mensagem da fila criada pelo modelo do CloudFormation (por exemplo, VPCE-SQS-Tutorial-Stack-CFQueue-1ABCDEFGH2IJK), consulte [Receber e excluir uma mensagem no Amazon SQS](#).

Para obter informações sobre como excluir seus recursos, consulte o seguinte:

- [Excluir um endpoint da VPC](#) no Guia do usuário da Amazon VPC
- [Excluir uma fila do Amazon SQS](#)
- [Encerramento de instâncias do Amazon EC2](#) no Guia do usuário do Amazon EC2
- [Excluir a VPC](#) no Guia do usuário da Amazon VPC
- [Excluir uma pilha no console do CloudFormation](#) no Guia do usuário do AWS CloudFormation
- [Excluir o par de chaves](#) no Guia do usuário do Amazon EC2.

Exemplos de código para o Amazon SQS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon SQS com um kit de desenvolvimento AWS de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de código

- [Exemplos básicos para o Amazon SQS usando AWS SDKs](#)
 - [Olá, Amazon SQS](#)
 - [Ações para o Amazon SQS usando AWS SDKs](#)
 - [Usar AddPermission com uma CLI](#)
 - [Use ChangeMessageVisibility com um AWS SDK ou CLI](#)
 - [Usar ChangeMessageVisibilityBatch com uma CLI](#)
 - [Use CreateQueue com um AWS SDK ou CLI](#)
 - [Use DeleteMessage com um AWS SDK ou CLI](#)
 - [Use DeleteMessageBatch com um AWS SDK ou CLI](#)
 - [Use DeleteQueue com um AWS SDK ou CLI](#)
 - [Use GetQueueAttributes com um AWS SDK ou CLI](#)
 - [Use GetQueueUrl com um AWS SDK ou CLI](#)
 - [Usar ListDeadLetterSourceQueues com uma CLI](#)
 - [Use ListQueues com um AWS SDK ou CLI](#)
 - [Usar PurgeQueue com uma CLI](#)
 - [Use ReceiveMessage com um AWS SDK ou CLI](#)

- [Usar RemovePermission com uma CLI](#)
- [Use SendMessage com um AWS SDK ou CLI](#)
- [Use SendMessageBatch com um AWS SDK ou CLI](#)
- [Use SetQueueAttributes com um AWS SDK ou CLI](#)
- [Cenários para o Amazon SQS usando AWS SDKs](#)
 - [Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS](#)
 - [Criar uma aplicação de mensageiro com o Step Functions](#)
 - [Criar uma aplicação de exploração do Amazon Textract](#)
 - [Crie e publique em um tópico FIFO do Amazon SNS usando um SDK AWS](#)
 - [Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS](#)
 - [Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um SDK AWS](#)
 - [Receba e processe notificações de eventos do Amazon S3 usando um SDK AWS](#)
 - [Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um SDK AWS](#)
 - [Envie e receba lotes de mensagens com o Amazon SQS usando um SDK AWS](#)
 - [Use o AWS Message Processing Framework para .NET para publicar e receber mensagens do Amazon SQS](#)
 - [Usar a Biblioteca de Mensagens Java do Amazon SQS para trabalhar com a interface do Serviço de Mensagens Java \(JMS\) para o Amazon SQS](#)
 - [Trabalhe com tags de fila e Amazon SQS usando um SDK AWS](#)
- [Exemplos de tecnologia sem servidor para o Amazon SQS](#)
 - [Invocar uma função do Lambda em um trigger do Amazon SQS](#)
 - [Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS](#)

Exemplos básicos para o Amazon SQS usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Simple Queue Service com AWS SDKs

Exemplos

- [Olá, Amazon SQS](#)
- [Ações para o Amazon SQS usando AWS SDKs](#)
 - [Usar AddPermission com uma CLI](#)

- [Use ChangeMessageVisibility com um AWS SDK ou CLI](#)
- [Usar ChangeMessageVisibilityBatch com uma CLI](#)
- [Use CreateQueue com um AWS SDK ou CLI](#)
- [Use DeleteMessage com um AWS SDK ou CLI](#)
- [Use DeleteMessageBatch com um AWS SDK ou CLI](#)
- [Use DeleteQueue com um AWS SDK ou CLI](#)
- [Use GetQueueAttributes com um AWS SDK ou CLI](#)
- [Use GetQueueUrl com um AWS SDK ou CLI](#)
- [Usar ListDeadLetterSourceQueues com uma CLI](#)
- [Use ListQueues com um AWS SDK ou CLI](#)
- [Usar PurgeQueue com uma CLI](#)
- [Use ReceiveMessage com um AWS SDK ou CLI](#)
- [Usar RemovePermission com uma CLI](#)
- [Use SendMessage com um AWS SDK ou CLI](#)
- [Use SendMessageBatch com um AWS SDK ou CLI](#)
- [Use SetQueueAttributes com um AWS SDK ou CLI](#)

Olá, Amazon SQS

Os exemplos de código a seguir mostram como começar a usar o Amazon SQS.

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
using Amazon.SQS;  
using Amazon.SQS.Model;
```

```
namespace SQSActions;

public static class HelloSQS
{
    static async Task Main(string[] args)
    {
        var sqsClient = new AmazonSQSClient();

        Console.WriteLine($"Hello Amazon SQS! Following are some of your queues:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five queues.
        var response = await sqsClient.ListQueuesAsync(
            new ListQueuesRequest()
            {
                MaxResults = 5
            });

        foreach (var queue in response.QueueUrls)
        {
            Console.WriteLine($"\\tQueue Url: {queue}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte [ListQueues](#) a Referência AWS SDK para .NET da API.

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Código para o CMake arquivo CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sqs)

# Set this project's name.
project("hello_sqs")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
    libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
        "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if(WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
    for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
    may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
        ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif()

add_executable(${PROJECT_NAME}
    hello_sqs.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```


Código para o arquivo de origem `hello_sqs.cpp`.

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>

/*
 * A "Hello SQS" starter application that initializes an Amazon Simple Queue
 * Service
 * (Amazon SQS) client and lists the SQS queues in the current account.
 *
 * main function
 *
 * Usage: 'hello_sqs'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SQS::SQSClient sqsClient(clientConfig);

        Aws::Vector<Aws::String> allQueueUrls;
        Aws::String nextToken; // Next token is used to handle a paginated
        response.
        do {
            Aws::SQS::Model::ListQueuesRequest request;

            Aws::SQS::Model::ListQueuesOutcome outcome =
            sqsClient.ListQueues(request);

            if (outcome.IsSuccess()) {
```

```

        const Aws::Vector<Aws::String> &pageOfQueueUrls =
outcome.GetResult().GetQueueUrls();
        if (!pageOfQueueUrls.empty()) {
            allQueueUrls.insert(allQueueUrls.cend(),
pageOfQueueUrls.cbegin(),
                                pageOfQueueUrls.cend());
        }
    }
    else {
        std::cerr << "Error with SQS::ListQueues. "
        << outcome.GetError().GetMessage()
        << std::endl;

        break;
    }
    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

std::cout << "Hello Amazon SQS! You have " << allQueueUrls.size() << "
queue"
        << (allQueueUrls.size() == 1 ? "" : "s") << " in your account."
        << std::endl;

if (!allQueueUrls.empty()) {
    std::cout << "Here are your queue URLs." << std::endl;
    for (const Aws::String &queueUrl: allQueueUrls) {
        std::cout << "  * " << queueUrl << std::endl;
    }
}

}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- Para obter detalhes da API, consulte [ListQueues](#) a Referência AWS SDK para C++ da API.

Go

SDK para Go V2

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    sqsClient := sqs.NewFromConfig(sdkConfig)
    fmt.Println("Let's list the queues for your account.")
    var queueUrls []string
    paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
    for paginator.HasMorePages() {
        output, err := paginator.NextPage(ctx)
        if err != nil {
```

```
    log.Printf("Couldn't get queues. Here's why: %v\n", err)
    break
} else {
    queueUrls = append(queueUrls, output.QueueUrls...)
}
}
if len(queueUrls) == 0 {
    fmt.Println("You don't have any queues!")
} else {
    for _, queueUrl := range queueUrls {
        fmt.Printf("\t%v\n", queueUrl)
    }
}
}
```

- Para obter detalhes da API, consulte [ListQueues](#) a Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
```

```
*/
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                .flatMap(r -> r.queueUrls().stream())
                .forEach(content -> System.out.println(" Queue URL: " +
                    content.toLowerCase()));
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte [ListQueues](#) a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Inicializar um cliente Amazon SQS e listar as filas.

```
import { SQSClient, paginateListQueues } from "@aws-sdk/client-sqs";

export const helloSqs = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SQSClient({});

  // You can also use `ListQueuesCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
  `ListQueuesCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedQueues = paginateListQueues({ client }, {});
  const queues = [];

  for await (const page of paginatedQueues) {
    if (page.QueueUrls?.length) {
      queues.push(...page.QueueUrls);
    }
  }

  const suffix = queues.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SQS! You have ${queues.length} queue${suffix} in your
    account.`
  );
  console.log(queues.map((t) => ` * ${t}`).join("\n"));
};
```

- Para obter detalhes da API, consulte [ListQueues](#) a Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
package com.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- Para obter detalhes da API, consulte a [ListQueues](#) referência da API AWS SDK for Kotlin.

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

O arquivo `Package.swift`.

```
import PackageDescription

let package = Package(
    name: "sqs-basics",
    // Let Xcode know the minimum Apple platforms supported.
```

```

platforms: [
    .macOS(.v13),
    .iOS(.v15)
],
dependencies: [
    // Dependencies declare other packages that this package depends on.
    .package(
        url: "https://github.com/aws-labs/aws-sdk-swift",
        from: "1.0.0"),
    .package(
        url: "https://github.com/apple/swift-argument-parser.git",
        branch: "main"
    )
],
targets: [
    // Targets are the basic building blocks of a package, defining a module
    // or a test suite.
    // Targets can depend on other targets in this package and products
    // from dependencies.
    .executableTarget(
        name: "sqs-basics",
        dependencies: [
            .product(name: "AWSSQS", package: "aws-sdk-swift"),
            .product(name: "ArgumentParser", package: "swift-argument-
parser")
        ],
        path: "Sources")
]
)

```

O código-fonte do Swift, `entry.swift`.

```

import ArgumentParser
import AWSClientRuntime
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"
}

```



```
static var configuration = CommandConfiguration(
    commandName: "sqs-basics",
    abstract: ""
    This example shows how to list all of your available Amazon SQS queues.
    "",
    discussion: ""
    ""
)

/// Called by ``main()`` to run the bulk of the example.
func runAsync() async throws {
    let config = try await SQSClient.SQSClientConfiguration(region: region)
    let sqsClient = SQSClient(config: config)

    var queues: [String] = []
    let outputPages = sqsClient.listQueuesPaginated(
        input: ListQueuesInput()
    )

    // Each time a page of results arrives, process its contents.

    for try await output in outputPages {
        guard let urls = output.queueUrls else {
            print("No queues found.")
            return
        }

        // Iterate over the queue URLs listed on this page, adding them
        // to the `queues` array.

        for queueUrl in urls {
            queues.append(queueUrl)
        }
    }

    print("You have \(queues.count) queues:")
    for queue in queues {
        print("    \(queue)")
    }
}

/// The program's asynchronous entry point.
@main
```

```
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte [ListQueues](#) a referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações para o Amazon SQS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon SQS com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon SQS e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em [Cenários para o Amazon SQS usando AWS SDKs](#).

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na [Referência da API do Amazon Simple Queue Service](#).

Exemplos

- [Usar AddPermission com uma CLI](#)
- [Use ChangeMessageVisibility com um AWS SDK ou CLI](#)
- [Usar ChangeMessageVisibilityBatch com uma CLI](#)
- [Use CreateQueue com um AWS SDK ou CLI](#)
- [Use DeleteMessage com um AWS SDK ou CLI](#)

- [Use DeleteMessageBatch com um AWS SDK ou CLI](#)
- [Use DeleteQueue com um AWS SDK ou CLI](#)
- [Use GetQueueAttributes com um AWS SDK ou CLI](#)
- [Use GetQueueUrl com um AWS SDK ou CLI](#)
- [Usar ListDeadLetterSourceQueues com uma CLI](#)
- [Use ListQueues com um AWS SDK ou CLI](#)
- [Usar PurgeQueue com uma CLI](#)
- [Use ReceiveMessage com um AWS SDK ou CLI](#)
- [Usar RemovePermission com uma CLI](#)
- [Use SendMessage com um AWS SDK ou CLI](#)
- [Use SendMessageBatch com um AWS SDK ou CLI](#)
- [Use SetQueueAttributes com um AWS SDK ou CLI](#)

Usar **AddPermission** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddPermission`.

CLI

AWS CLI

Como adicionar uma permissão a uma fila

Este exemplo permite que a AWS conta especificada envie mensagens para a fila especificada.

Comando:

```
aws sqs add-permission --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --label SendMessageFromMyQueue --aws-account-ids 12345EXAMPLE --actions SendMessage
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte [AddPermission](#) em Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo permite que Conta da AWS o especificado envie mensagens da fila especificada.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE  
-Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-  
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [AddPermission](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo permite que Conta da AWS o especificado envie mensagens da fila especificada.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE  
-Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-  
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [AddPermission](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **ChangeMessageVisibility** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o **ChangeMessageVisibility**.

C++

SDK para C++

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    ///! Changes the visibility timeout of a message in an Amazon Simple Queue Service
    ///! (Amazon SQS) queue.
    /*!
    \param queueUrl: An Amazon SQS queue URL.
    \param messageReceiptHandle: A message receipt handle.
    \param visibilityTimeoutSeconds: Visibility timeout in seconds.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::changeMessageVisibility(
        const Aws::String &queue_url,
        const Aws::String &messageReceiptHandle,
        int visibilityTimeoutSeconds,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::ChangeMessageVisibilityRequest request;
        request.SetQueueUrl(queue_url);
        request.SetReceiptHandle(messageReceiptHandle);
        request.SetVisibilityTimeout(visibilityTimeoutSeconds);

        auto outcome = sqsClient.ChangeMessageVisibility(request);
        if (outcome.IsSuccess()) {
            std::cout << "Successfully changed visibility of message " <<
                messageReceiptHandle << " from queue " << queue_url <<
std::endl;
        }
        else {

```

```
        std::cout << "Error changing visibility of message from queue "
        << queue_url << ": " <<
        outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte [ChangeMessageVisibility](#) a Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como alterar a visibilidade do tempo limite de uma mensagem

Este exemplo altera a visibilidade do tempo limite da mensagem especificada para 10 horas (10 horas * 60 minutos * 60 segundos).

Comando:

```
aws sqs change-message-visibility --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --receipt-handle AQEBTpyI...t6HyQg== --visibility-timeout 36000
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte [ChangeMessageVisibility](#) em Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receba uma mensagem do Amazon SQS e altere sua visibilidade de tempo limite.

```
import {
  ReceiveMessageCommand,
  ChangeMessageVisibilityCommand,
  SQSClient,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 1,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 1,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  const response = await client.send(
    new ChangeMessageVisibilityCommand({
      QueueUrl: queueUrl,
      ReceiptHandle: Messages[0].ReceiptHandle,
      VisibilityTimeout: 20,
    }),
  );
  console.log(response);
}
```

```
    return response;
};
```

- Para obter detalhes da API, consulte [ChangeMessageVisibility](#) a Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receba uma mensagem do Amazon SQS e altere sua visibilidade de tempo limite.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region to us-west-2
AWS.config.update({ region: "us-west-2" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME";

var params = {
    AttributeNames: ["SentTimestamp"],
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ["All"],
    QueueUrl: queueURL,
};

sqs.receiveMessage(params, function (err, data) {
    if (err) {
        console.log("Receive Error", err);
    } else {
        // Make sure we have a message
        if (data.Messages != null) {
            var visibilityParams = {
                QueueUrl: queueURL,
                ReceiptHandle: data.Messages[0].ReceiptHandle,
```



```
    VisibilityTimeout: 20, // 20 second timeout
  };
  sqs.changeMessageVisibility(visibilityParams, function (err, data) {
    if (err) {
      console.log("Delete Error", err);
    } else {
      console.log("Timeout Changed", data);
    }
  });
} else {
  console.log("No messages to change");
}
}
});
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [ChangeMessageVisibility](#) a Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera o tempo limite de visibilidade da mensagem com o identificador de recebimento especificado na fila especificada para 10 horas (1 hora x 60 minutos x 60 segundos = 36.000 segundos).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -
VisibilityTimeout 36000
```

- Para obter detalhes da API, consulte [ChangeMessageVisibility](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o tempo limite de visibilidade da mensagem com o identificador de recebimento especificado na fila especificada para 10 horas (1 hora x 60 minutos x 60 segundos = 36.000 segundos).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- Para obter detalhes da API, consulte [ChangeMessageVisibility](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

begin
  queue_name = 'my-queue'
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Receive up to 10 messages
  receive_message_result_before = sqs.receive_message({
                                                                    queue_url: queue_url,
                                                                    max_number_of_messages:
10
                                                                    })

  puts "Before attempting to change message visibility timeout: received
#{receive_message_result_before.messages.count} message(s)."
```

```
  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
                                    queue_url: queue_url,
                                    receipt_handle: message.receipt_handle,
```

```
visibility_timeout: 30 # This message will
not be visible for 30 seconds after first receipt.
    })

end

# Try to retrieve the original messages after setting their visibility timeout.
receive_message_result_after = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
})

puts "\nAfter attempting to change message visibility timeout: received
#{receive_message_result_after.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
    puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
exist."
end
```

- Para obter detalhes da API, consulte [ChangeMessageVisibility](#) a Referência AWS SDK para Ruby da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ChangeMessageVisibilityBatch** com uma CLI

Os exemplos de código a seguir mostram como usar o **ChangeMessageVisibilityBatch**.

CLI

AWS CLI

Como alterar as visibilidades de tempo limite de várias mensagens como um lote

Este exemplo altera as visibilidades do tempo limite das duas mensagens especificadas para 10 horas (10 horas * 60 minutos * 60 segundos).

Comando:

```
aws sqs change-message-visibility-batch --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://change-message-visibility-batch.json
```

Arquivo de entrada (change-message-visibility-batch.json):

```
[
  {
    "Id": "FirstMessage",
    "ReceiptHandle": "AQEBhz2q...Jf3kaw==",
    "VisibilityTimeout": 36000
  },
  {
    "Id": "SecondMessage",
    "ReceiptHandle": "AQEBkTUH...HifSnw==",
    "VisibilityTimeout": 36000
  }
]
```

Saída:

```
{
  "Successful": [
    {
      "Id": "SecondMessage"
    },
    {
      "Id": "FirstMessage"
    }
  ]
}
```

- Para obter detalhes da API, consulte [ChangeMessageVisibilityBatch](#) em Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera o tempo limite de visibilidade para duas mensagens com os identificadores de recebimento especificados na fila especificada. O tempo limite de

visibilidade da primeira mensagem é alterado para 10 horas (10 horas x 60 minutos x 60 segundos = 36.000 segundos). O tempo limite de visibilidade da segunda mensagem é alterado para 5 horas (5 horas x 60 minutos x 60 segundos = 18.000 segundos).

```
$changeVisibilityRequest1 = New-Object
    Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
    Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
    $changeVisibilityRequest2
```

Saída:

```
Failed    Successful
-----
{}        {Request2, Request1}
```

- Para obter detalhes da API, consulte [ChangeMessageVisibilityBatch](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o tempo limite de visibilidade para duas mensagens com os identificadores de recebimento especificados na fila especificada. O tempo limite de visibilidade da primeira mensagem é alterado para 10 horas (10 horas x 60 minutos x 60 segundos = 36.000 segundos). O tempo limite de visibilidade da segunda mensagem é alterado para 5 horas (5 horas x 60 minutos x 60 segundos = 18.000 segundos).

```
$changeVisibilityRequest1 = New-Object
    Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
```

```
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
    Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
    $changeVisibilityRequest2
```

Saída:

```
Failed      Successful
-----
{}          {Request2, Request1}
```

- Para obter detalhes da API, consulte [ChangeMessageVisibilityBatch](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **CreateQueue** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o **CreateQueue**.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Publicar mensagens em filas](#)
- [Enviar e receber lotes de mensagens](#)
- [Utilizar a biblioteca de mensagens Java do Amazon SQS para trabalhar com a interface do JMS](#)

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Criar uma fila com um nome específico.

```
/// <summary>
/// Create a queue with a specific name.
/// </summary>
/// <param name="queueName">The name for the queue.</param>
/// <param name="useFifoQueue">True to use a FIFO queue.</param>
/// <returns>The url for the queue.</returns>
public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
{
    int maxMessage = 256 * 1024;
    var queueAttributes = new Dictionary<string, string>
    {
        {
            QueueAttributeName.MaximumMessageSize,
            maxMessage.ToString()
        }
    };

    var createQueueRequest = new CreateQueueRequest()
    {
        QueueName = queueName,
        Attributes = queueAttributes
    };

    if (useFifoQueue)
    {
        // Update the name if it is not correct for a FIFO queue.
        if (!queueName.EndsWith(".fifo"))
        {
            createQueueRequest.QueueName = queueName + ".fifo";
        }
    }
}
```

```
        // Add an attribute for a FIFO queue.
        createQueueRequest.Attributes.Add(
            QueueAttributeName.FifoQueue, "true");
    }

    var createResponse = await _amazonSQSClient.CreateQueueAsync(
        new CreateQueueRequest()
        {
            QueueName = queueName
        });
    return createResponse.QueueUrl;
}
```

Crie uma fila do Amazon SQS e envie uma mensagem para ela.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.SQS;
using Amazon.SQS.Model;

public class CreateSendExample
{
    // Specify your AWS Region (an example Region is shown).
    private static readonly string QueueName = "Example_Queue";
    private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
    private static IAmazonSQS client;

    public static async Task Main()
    {
        client = new AmazonSQSClient(ServiceRegion);
        var createQueueResponse = await CreateQueue(client, QueueName);

        string queueUrl = createQueueResponse.QueueUrl;

        Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
        {
```



```

        { "Title",    new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
        { "Author",  new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
        { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } },
    };

    string messageBody = "Information about current NY Times fiction
bestseller for week of 12/11/2016.";

    var sendMsgResponse = await SendMessage(client, queueUrl,
messageBody, messageAttributes);
    }

    /// <summary>
    /// Creates a new Amazon SQS queue using the queue name passed to it
    /// in queueName.
    /// </summary>
    /// <param name="client">An SQS client object used to send the message.</
param>
    /// <param name="queueName">A string representing the name of the queue
    /// to create.</param>
    /// <returns>A CreateQueueResponse that contains information about the
    /// newly created queue.</returns>
    public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS
client, string queueName)
    {
        var request = new CreateQueueRequest
        {
            QueueName = queueName,
            Attributes = new Dictionary<string, string>
            {
                { "DelaySeconds", "60" },
                { "MessageRetentionPeriod", "86400" },
            },
        };

        var response = await client.CreateQueueAsync(request);
        Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

        return response;
    }

```

```

    /// <summary>
    /// Sends a message to an SQS queue.
    /// </summary>
    /// <param name="client">An SQS client object used to send the message.</
param>
    /// <param name="queueUrl">The URL of the queue to which to send the
    /// message.</param>
    /// <param name="messageBody">A string representing the body of the
    /// message to be sent to the queue.</param>
    /// <param name="messageAttributes">Attributes for the message to be
    /// sent to the queue.</param>
    /// <returns>A SendMessageResponse object that contains information
    /// about the message that was sent.</returns>
    public static async Task<SendMessageResponse> SendMessage(
        IAmazonSQS client,
        string queueUrl,
        string messageBody,
        Dictionary<string, MessageAttributeValue> messageAttributes)
    {
        var sendMessageRequest = new SendMessageRequest
        {
            DelaySeconds = 10,
            MessageAttributes = messageAttributes,
            MessageBody = messageBody,
            QueueUrl = queueUrl,
        };

        var response = await client.SendMessageAsync(sendMessageRequest);
        Console.WriteLine($"Sent a message with id : {response.MessageId}");

        return response;
    }
}

```

- Para obter detalhes da API, consulte [CreateQueue](#) a Referência AWS SDK para .NET da API.

C++

SDK para C++

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Create an Amazon Simple Queue Service (Amazon SQS) queue.
    /*!
    \param queueName: An Amazon SQS queue name.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::createQueue(const Aws::String &queueName,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::CreateQueueRequest request;
        request.SetQueueName(queueName);

        const Aws::SQS::Model::CreateQueueOutcome outcome =
            sqsClient.CreateQueue(request);
        if (outcome.IsSuccess()) {
            std::cout << "Successfully created queue " << queueName << " with a queue
            URL "
                        << outcome.GetResult().GetQueueUrl() << "." << std::endl;
        }
        else {
            std::cerr << "Error creating queue " << queueName << ": " <<
                outcome.GetError().GetMessage() << std::endl;
        }

        return outcome.IsSuccess();
    }

```

- Para obter detalhes da API, consulte [CreateQueue](#) na Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar uma fila

Este exemplo cria uma fila com o nome especificado, define o período de retenção da mensagem para 3 dias (3 dias * 24 horas * 60 minutos * 60 segundos) e define a fila de mensagens não entregues da fila especificada com uma contagem máxima de recebimento de 1.000 mensagens.

Comando:

```
aws sqs create-queue --queue-name MyQueue --attributes file://create-queue.json
```

Arquivo de entrada (create-queue.json):

```
{
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":\"1000\"}\",
  \"MessageRetentionPeriod\": \"259200\"
}
```

Saída:

```
{
  \"QueueUrl\": \"https://queue.amazonaws.com/80398EXAMPLE/MyQueue\"
}
```

- Para obter detalhes da API, consulte [CreateQueue](#) em Referência de AWS CLI Comandos.

Go

SDK para Go V2

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// CreateQueue creates an Amazon SQS queue with the specified name. You can
// specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
    isFifoQueue bool) (string, error) {
    var queueUrl string
    queueAttributes := map[string]string{}
    if isFifoQueue {
        queueAttributes["FifoQueue"] = "true"
    }
    queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
        QueueName: aws.String(queueName),
```

```
Attributes: queueAttributes,
}))
if err != nil {
    log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
} else {
    queueUrl = *queue.QueueUrl
}

return queueUrl, err
}
```

- Para obter detalhes da API, consulte [CreateQueue](#) a Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient
                .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
        }
    }
}
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void listQueues(SqsClient sqsClient) {

        System.out.println("\nList Queues");
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
                ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
                sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
        // List queues with filters
        String namePrefix = "queue";
        ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
            .queueNamePrefix(namePrefix)
            .build();

        ListQueuesResponse listQueuesFilteredResponse =
            sqsClient.listQueues(filterListRequest);
        System.out.println("Queue URLs with prefix: " + namePrefix);
        for (String url : listQueuesFilteredResponse.queueUrls()) {
            System.out.println(url);
        }

        System.out.println("\nSend message");
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
```



```
        .queueUrl(queueUrl)
        .messageBody("Hello world!")
        .delaySeconds(10)
        .build());

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nSend multiple messages");
    try {
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
            .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello
from msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)

                .build())
            .build();
        sqsClient.sendMessageBatch(sendMessageBatchRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

    System.out.println("\nReceive messages");
    try {
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
            .queueUrl(queueUrl)
            .maxNumberOfMessages(5)
            .build();
```

```
        return sqsClient.receiveMessage(receiveMessageRequest).messages();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nChange Message Visibility");
    try {

        for (Message message : messages) {
            ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .visibilityTimeout(100)
                .build();
            sqsClient.changeMessageVisibility(req);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    System.out.println("\nDelete Messages");

    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    }
```

```
        } catch (SqsException e) {  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Para obter detalhes da API, consulte [CreateQueue](#) a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Crie uma fila padrão do Amazon SQS.

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";  
  
const client = new SQSClient({});  
const SQS_QUEUE_NAME = "test-queue";  
  
export const main = async (sqsQueueName = SQS_QUEUE_NAME) => {  
    const command = new CreateQueueCommand({  
        QueueName: sqsQueueName,  
        Attributes: {  
            DelaySeconds: "60",  
            MessageRetentionPeriod: "86400",  
        },  
    });  
  
    const response = await client.send(command);  
    console.log(response);  
    return response;  
};
```

Crie uma fila do Amazon SQS com sondagem longa.

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "queue_name";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const response = await client.send(
    new CreateQueueCommand({
      QueueName: queueName,
      Attributes: {
        // When the wait time for the ReceiveMessage API action is greater than
        // 0,
        // long polling is in effect. The maximum long polling wait time is 20
        // seconds. Long polling helps reduce the cost of using Amazon SQS by,
        // eliminating the number of empty responses and false empty responses.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
        // SQSDeveloperGuide/sqs-short-and-long-polling.html
        ReceiveMessageWaitTimeSeconds: "20",
      },
    }),
  );
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [CreateQueue](#) na Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Crie uma fila padrão do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    DelaySeconds: "60",
    MessageRetentionPeriod: "86400",
  },
};

sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

Crie uma fila do Amazon SQS que aguarda a chegada de uma mensagem.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
};
```

```
sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [CreateQueue](#) na Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val getQueueUrlRequest =
            GetQueueUrlRequest {
                queueName = queueNameVal
            }
    }
```

```
        val getUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getUrlResponse.queueUrl.toString()
    }
}
```

- Para obter detalhes da API, consulte a [CreateQueue](#) referência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma fila com o nome especificado.

```
New-SQSQueue -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [CreateQueue](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma fila com o nome especificado.

```
New-SQSQueue -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [CreateQueue](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
def create_queue(name, attributes=None):
    """
    Creates an Amazon SQS queue.

    :param name: The name of the queue. This is part of the URL assigned to the
    queue.
    :param attributes: The attributes of the queue, such as maximum message size
    or
                       whether it's a FIFO queue.
    :return: A Queue object that contains metadata about the queue and that can
    be used
             to perform queue operations like sending and receiving messages.
    """
    if not attributes:
        attributes = {}

    try:
        queue = sqs.create_queue(QueueName=name, Attributes=attributes)
        logger.info("Created queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't create queue named '%s'.", name)
        raise error
    else:
        return queue
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
```



```

    """
    Initialize the SqsWrapper.

    :param sqs_client: A Boto3 Amazon SQS client.
    """
    self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)

    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

```

```

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
            raise

```

- Para obter detalhes da API, consulte a [CreateQueue](#) Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

This code example demonstrates how to create a queue in Amazon Simple Queue Service (Amazon SQS).

```
require 'aws-sdk-sqs'
```

```
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
```

```
# @param queue_name [String] The name of the queue.
```

```
# @return [Boolean] true if the queue was created; otherwise, false.
```

```
# @example
```

```
#   exit 1 unless queue_created?(
```

```
#     Aws::SQS::Client.new(region: 'us-west-2'),
```

```
#     'my-queue'
```

```
#   )
```

```
def queue_created?(sqs_client, queue_name)
```

```
  sqs_client.create_queue(queue_name: queue_name)
```

```
  true
```

```
rescue StandardError => e
```

```
  puts "Error creating queue: #{e.message}"
```

```
  false
```

```

end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts 'Queue created.'
  else
    puts 'Queue not created.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__

```

- Para obter detalhes da API, consulte [CreateQueue](#) na Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Crie uma fila padrão do Amazon SQS.

```

TRY.
    oo_result = lo_sqs->createqueue( iv_queuename = iv_queue_name ).
    oo_result is returned for testing purposes. "
    MESSAGE 'SQS queue created.' TYPE 'I'.

```

```

    CATCH /aws1/cx_sqsqueuedeletedrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
        queue with the same name.' TYPE 'E'.
    CATCH /aws1/cx_sqsqueueexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.

```

Crie uma fila do Amazon SQS que aguarda a chegada de uma mensagem.

```

    TRY.
        DATA lt_attributes TYPE /aws1/cl_sqsqueueattrmap_w=>tt_queueattributemap.
        DATA ls_attribute TYPE /aws1/
        cl_sqsqueueattrmap_w=>ts_queueattributemap_maprow.
        ls_attribute-key = 'ReceiveMessageWaitTimeSeconds'. " Time
        in seconds for long polling, such as how long the call waits for a message to
        arrive in the queue before returning. "
        ls_attribute-value = NEW /aws1/cl_sqsqueueattrmap_w( iv_value =
        iv_wait_time ).
        INSERT ls_attribute INTO TABLE lt_attributes.
        oo_result = lo_sqs->createqueue( " oo_result is returned
        for testing purposes. "
            iv_queue_name = iv_queue_name
            it_attributes = lt_attributes ).
        MESSAGE 'SQS queue created.' TYPE 'I'.
    CATCH /aws1/cx_sqsqueuedeletedrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
        queue with the same name.' TYPE 'E'.
    CATCH /aws1/cx_sqsqueueexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.

```

- Para obter detalhes da API, consulte a [CreateQueue](#) referência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import AWSSQS

let config = try await SQSClient.SQSClientConfiguration(region: region)
let sqsClient = SQSClient(config: config)

let output = try await sqsClient.createQueue(
    input: CreateQueueInput(
        queueName: queueName
    )
)

guard let queueUrl = output.queueUrl else {
    print("No queue URL returned.")
    return
}
```

- Para obter detalhes da API, consulte [CreateQueue](#) a referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **DeleteMessage** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteMessage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Enviar e receber lotes de mensagens](#)

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receba uma mensagem de uma fila do Amazon SQS e, em seguida, exclua a mensagem.

```
public static async Task Main()
{
    // If the AWS Region you want to use is different from
    // the AWS Region defined for the default user, supply
    // the specify your AWS Region to the client constructor.
    var client = new AmazonSQSClient();
    string queueName = "Example_Queue";

    var queueUrl = await GetQueueUrl(client, queueName);
    Console.WriteLine($"The SQS queue's URL is {queueUrl}");

    var response = await ReceiveAndDeleteMessage(client, queueUrl);

    Console.WriteLine($"Message: {response.Messages[0]}");
}

/// <summary>
/// Retrieve the queue URL for the queue named in the queueName
/// property using the client object.
/// </summary>
/// <param name="client">The Amazon SQS client used to retrieve the
/// queue URL.</param>
/// <param name="queueName">A string representing name of the queue
/// for which to retrieve the URL.</param>
/// <returns>The URL of the queue.</returns>
public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
{
    var request = new GetQueueUrlRequest
    {
        QueueName = queueName,
```

```

    };

    GetQueueUrlResponse response = await
client.GetQueueUrlAsync(request);
    return response.QueueUrl;
}

/// <summary>
/// Retrieves the message from the quque at the URL passed in the
/// queueURL parameters using the client.
/// </summary>
/// <param name="client">The SQS client used to retrieve a message.</
param>
/// <param name="queueUrl">The URL of the queue from which to retrieve
/// a message.</param>
/// <returns>The response from the call to ReceiveMessageAsync.</returns>
public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
{
    // Receive a single message from the queue.
    var receiveMessageRequest = new ReceiveMessageRequest
    {
        AttributeNames = { "SentTimestamp" },
        MaxNumberOfMessages = 1,
        MessageAttributeNames = { "All" },
        QueueUrl = queueUrl,
        VisibilityTimeout = 0,
        WaitTimeSeconds = 0,
    };

    var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

    // Delete the received message from the queue.
    var deleteMessageRequest = new DeleteMessageRequest
    {
        QueueUrl = queueUrl,
        ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
    };

    await client.DeleteMessageAsync(deleteMessageRequest);

    return receiveMessageResponse;
}

```

```
}
```

- Para obter detalhes da API, consulte [DeleteMessage](#) na Referência AWS SDK para .NET da API.

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

//! Delete a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
 \param queueUrl: An Amazon SQS queue URL.
 \param messageReceiptHandle: A message receipt handle.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteMessage(const Aws::String &queueUrl,
                                const Aws::String &messageReceiptHandle,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetReceiptHandle(messageReceiptHandle);

    const Aws::SQS::Model::DeleteMessageOutcome outcome =
    sqsClient.DeleteMessage(
        request);
```



```
    if (outcome.IsSuccess()) {  
        std::cout << "Successfully deleted message from queue " << queueUrl  
                    << std::endl;  
    }  
    else {  
        std::cerr << "Error deleting message from queue " << queueUrl << ": " <<  
                    outcome.GetError().GetMessage() << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- Para obter detalhes da API, consulte [DeleteMessage](#) na Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir uma mensagem

Este exemplo exclui a mensagem especificada.

Comando:

```
aws sqs delete-message --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --receipt-handle AQEBRXTo...q2doVA==
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte [DeleteMessage](#) na Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- Para obter detalhes da API, consulte [DeleteMessage](#) a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receber e excluir mensagens do Amazon SQS.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
```

```
        ReceiptHandle: message.ReceiptHandle,
    })),
  }},
);
}
};
```

- Para obter detalhes da API, consulte [DeleteMessage](#) na Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receber e excluir mensagens do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  }
});
```

```
    } else if (data.Messages) {
        var deleteParams = {
            QueueUrl: queueURL,
            ReceiptHandle: data.Messages[0].ReceiptHandle,
        };
        sqs.deleteMessage(deleteParams, function (err, data) {
            if (err) {
                console.log("Delete Error", err);
            } else {
                console.log("Message Deleted", data);
            }
        });
    }
});
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [DeleteMessage](#) Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
    }
}
```

```
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Para obter detalhes da API, consulte a [DeleteMessage](#) referência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a mensagem com os identificadores de recebimento especificados na fila especificada.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue -ReceiptHandle AQEBd329...v6gl8Q==
```

- Para obter detalhes da API, consulte [DeleteMessage](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a mensagem com os identificadores de recebimento especificados na fila especificada.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue -ReceiptHandle AQEBd329...v6gl8Q==
```

- Para obter detalhes da API, consulte [DeleteMessage](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
def delete_message(message):  
    """  
    Delete a message from a queue. Clients must delete messages after they  
    are received and processed to remove them from the queue.  
  
    :param message: The message to delete. The message's queue URL is contained  
in  
                    the message's metadata.  
    :return: None  
    """  
    try:  
        message.delete()  
        logger.info("Deleted message: %s", message.message_id)  
    except ClientError as error:  
        logger.exception("Couldn't delete message: %s", message.message_id)  
        raise error
```

- Para obter detalhes da API, consulte a [DeleteMessage](#) Referência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
TRY.  
    lo_sqs->deletemessage(  
        iv_queueurl = iv_queue_url  
        iv_receipthandle = iv_receipt_handle ).  
    MESSAGE 'Message deleted from SQS queue.' TYPE 'I'.  
CATCH /aws1/cx_sqsinvalididformat.  
    MESSAGE 'The specified receipt handle is not valid.' TYPE 'E'.  
CATCH /aws1/cx_sqsreceipthandleisinv.  
    MESSAGE 'The specified receipt handle is not valid for the current  
version.' TYPE 'E'.  
ENDTRY.
```

- Para obter detalhes da API, consulte a [DeleteMessage](#) referência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **DeleteMessageBatch** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o DeleteMessageBatch.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Processar notificações de eventos do S3](#)
- [Publicar mensagens em filas](#)
- [Enviar e receber lotes de mensagens](#)

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete a batch of messages from a queue by its url.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
List<Message> messages)
{
    var deleteRequest = new DeleteMessageBatchRequest()
    {
        QueueUrl = queueUrl,
        Entries = new List<DeleteMessageBatchRequestEntry>()
    };
    foreach (var message in messages)
    {
        deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
        {
            ReceiptHandle = message.ReceiptHandle,
            Id = message.MessageId
        });
    }

    var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

    return deleteResponse.Failed.Any();
}
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) na Referência AWS SDK para .NET da API.

C++

SDK para C++

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageBatchRequest request;
    request.SetQueueUrl(queueURLS[i]);
    int id = 1; // Ids must be unique within a batch delete request.
    for (const Aws::String &receiptHandle: receiptHandles) {
        Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
        entry.SetId(std::to_string(id));
        ++id;
        entry.SetReceiptHandle(receiptHandle);
        request.AddEntries(entry);
    }

    Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
        sqsClient.DeleteMessageBatch(request);

    if (outcome.IsSuccess()) {
        std::cout << "The batch deletion of messages was successful."
                  << std::endl;
    }
    else {
        std::cerr << "Error with SQS::DeleteMessageBatch. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
```

```
        sqsClient);  
  
        return false;  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) na Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir várias mensagens como um lote

Este exemplo exclui as mensagens especificadas.

Comando:

```
aws sqs delete-message-batch --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://delete-message-batch.json
```

Arquivo de entrada (delete-message-batch.json):

```
[  
  {  
    "Id": "FirstMessage",  
    "ReceiptHandle": "AQEB1mg1...Z4GuLw=="  
  },  
  {  
    "Id": "SecondMessage",  
    "ReceiptHandle": "AQEBLsYM...VQubAA=="  
  }  
]
```

Saída:

```
{  
  "Successful": [  
    {
```

```
    "Id": "FirstMessage"
  },
  {
    "Id": "SecondMessage"
  }
]
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) em Referência de AWS CLI Comandos.

Go

SDK para Go V2

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}
```

```
// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
// messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
    messages []types.Message) error {
    entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
    for msgIndex := range messages {
        entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
        entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
    }
    _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
        Entries: entries,
        QueueUrl: aws.String(queueUrl),
    })
    if err != nil {
        log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
            queueUrl, err)
    }
    return err
}
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) na Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import {
    ReceiveMessageCommand,
    DeleteMessageCommand,
    SQSClient,
    DeleteMessageBatchCommand,
```

```
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) na Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui duas mensagens com os identificadores de recebimento especificados na fila especificada.

```
$deleteMessageRequest1 = New-Object
    Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object
    Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBq0VY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $deleteMessageRequest1,
    $deleteMessageRequest2
```

Saída:

```
Failed      Successful
-----
{}          {Request1, Request2}
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui duas mensagens com os identificadores de recebimento especificados na fila especificada.

```
$deleteMessageRequest1 = New-Object
    Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object
    Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBq0VY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $deleteMessageRequest1,
    $deleteMessageRequest2
```

Saída:

```
Failed    Successful
-----
{}        {Request1, Request2}
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
```



```

        :param messages: The list of messages to delete.
        :return: The response from SQS that contains the list of successful and
        failed
            message deletions.
        """
        try:
            entries = [
                {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
                for ind, msg in enumerate(messages)
            ]
            response = queue.delete_messages(Entries=entries)
            if "Successful" in response:
                for msg_meta in response["Successful"]:
                    logger.info("Deleted %s",
                        messages[int(msg_meta["Id"])]receipt_handle)
            if "Failed" in response:
                for msg_meta in response["Failed"]:
                    logger.warning(
                        "Could not delete %s",
                        messages[int(msg_meta["Id"])]receipt_handle
                    )
        except ClientError:
            logger.exception("Couldn't delete messages from queue %s", queue)
        else:
            return response

```

```

class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """

```

```

        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)

    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
    bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

```

```

        logger.info(f"Deleted {len(messages)} messages from {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error deleting messages: {error_code} - {e}")
        raise

```

- Para obter detalhes da API, consulte a [DeleteMessageBatch](#) Referência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_sqs->deletemessagebatch(      " oo_result is returned for
testing purposes. "
        iv_queueurl = iv_queue_url
        it_entries = it_entries ).
    MESSAGE 'Messages deleted from SQS queue.' TYPE 'I'.
CATCH /aws1/cx_sqsbtcentidsnotdist00.
    MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
CATCH /aws1/cx_sqsemptybatchrequest.
    MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
CATCH /aws1/cx_sqsinvbatchentryid.
    MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE
'E'.
CATCH /aws1/cx_sqstoomanyentriesin00.
    MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
ENDTRY.

```

- Para obter detalhes da API, consulte a [DeleteMessageBatch](#) referência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import AWSSQS

let config = try await SQSClient.SQSClientConfiguration(region: region)
let sqsClient = SQSClient(config: config)

// Create the list of message entries.

var entries: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []
var messageNumber = 1

for handle in handles {
    let entry = SQSClientTypes.DeleteMessageBatchRequestEntry(
        id: "\(messageNumber)",
        receiptHandle: handle
    )
    entries.append(entry)
    messageNumber += 1
}

// Delete the messages.

let output = try await sqsClient.deleteMessageBatch(
    input: DeleteMessageBatchInput(
        entries: entries,
        queueUrl: queue
    )
)
```

```
)

// Get the lists of failed and successful deletions from the output.

guard let failedEntries = output.failed else {
    print("Failed deletion list is missing!")
    return
}
guard let successfulEntries = output.successful else {
    print("Successful deletion list is missing!")
    return
}

// Display a list of the failed deletions along with their
// corresponding explanation messages.

if failedEntries.count != 0 {
    print("Failed deletions:")

    for entry in failedEntries {
        print("Message #\(entry.id ?? "<unknown>") failed:
\(\entry.message ?? "<unknown>")")
    }
} else {
    print("No failed deletions.")
}

// Output a list of the message numbers that were successfully deleted.

if successfulEntries.count != 0 {
    var successes = ""

    for entry in successfulEntries {
        if successes.count == 0 {
            successes = entry.id ?? "<unknown>"
        } else {
            successes = "\(\successes), \(\entry.id ?? "<unknown>")"
        }
    }
    print("Succeeded: ", successes)
} else {
    print("No successful deletions.")
}
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **DeleteQueue** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteQueue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Publicar mensagens em filas](#)
- [Enviar e receber lotes de mensagens](#)
- [Utilizar a biblioteca de mensagens Java do Amazon SQS para trabalhar com a interface do JMS](#)

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Excluir uma fila usando seu URL.

```
/// <summary>
/// Delete a queue by its URL.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteQueueByUrl(string queueUrl)
{
    var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
```

```

        new DeleteQueueRequest()
        {
            QueueUrl = queueUrl
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}

```

- Para obter detalhes da API, consulte [DeleteQueue](#) na Referência AWS SDK para .NET da API.

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    /*! Delete an Amazon Simple Queue Service (Amazon SQS) queue.
    *!
    *! \param queueURL: An Amazon SQS queue URL.
    *! \param clientConfiguration: AWS client configuration.
    *! \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::deleteQueue(const Aws::String &queueURL,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);
        Aws::SQS::Model::DeleteQueueRequest request;
        request.SetQueueUrl(queueURL);

        const Aws::SQS::Model::DeleteQueueOutcome outcome =
            sqsClient.DeleteQueue(request);
        if (outcome.IsSuccess()) {

```

```
        std::cout << "Successfully deleted queue with url " << queueURL <<
            std::endl;
    }
    else {
        std::cerr << "Error deleting queue " << queueURL << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte [DeleteQueue](#) na Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir uma fila

Este exemplo exclui a fila especificada.

Comando:

```
aws sqs delete-queue --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyNewerQueue
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte [DeleteQueue](#) em Referência de AWS CLI Comandos.

Go

SDK para Go V2

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).


```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
    _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
        QueueUrl: aws.String(queueUrl)})
    if err != nil {
        log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
    }
    return err
}
```

- Para obter detalhes da API, consulte [DeleteQueue](#) a Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <queueName>

            Where:
                queueName - The name of the Amazon SQS queue to delete.

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
```

```
SqsClient sqs = SqsClient.builder()
    .region(Region.US_WEST_2)
    .build();

deleteSQSQueue(sqs, queueName);
sqs.close();
}

public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [DeleteQueue](#) a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Excluir uma fila do Amazon SQS.

```
import { DeleteQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "test-queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new DeleteQueueCommand({ QueueUrl: queueUrl });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [DeleteQueue](#) na Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Excluir uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueUrl: "SQS_QUEUE_URL",
```

```
};

sqs.deleteQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [DeleteQueue](#) na Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
```

```
val request =
    DeleteQueueRequest {
        queueUrl = queueUrlVal
    }

SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
    sqsClient.deleteQueue(request)
    println("$queueUrlVal was deleted!")
}
}
```

- Para obter detalhes da API, consulte a [DeleteQueue](#) referência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a fila especificada.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Para obter detalhes da API, consulte [DeleteQueue](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a fila especificada.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Para obter detalhes da API, consulte [DeleteQueue](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
def remove_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
```

```
Create an SqsWrapper instance using a default boto3 client.

:return: An instance of this class.
"""
sqs_client = boto3.client('sqs')
return cls(sqs_client)

def delete_queue(self, queue_url: str) -> bool:
    """
    Delete an SQS queue.

    :param queue_url: The URL of the queue to delete.
    :return: True if successful.
    :raises ClientError: If the queue deletion fails.
    """
    try:
        self.sqs_client.delete_queue(QueueUrl=queue_url)

        logger.info(f"Deleted queue: {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
            logger.warning(f"Queue not found: {queue_url}")
            return True # Already deleted
        else:
            logger.error(f"Error deleting queue: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a [DeleteQueue](#) Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

- Para obter detalhes da API, consulte [DeleteQueue](#) a Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
TRY.
    lo_sqs->deletequeue( iv_queueurl = iv_queue_url ).
    MESSAGE 'SQS queue deleted' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a [DeleteQueue](#) referência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import AWSSQS

let config = try await SQSClient.SQSClientConfiguration(region: region)
let sqsClient = SQSClient(config: config)

do {
    _ = try await sqsClient.deleteQueue(
        input: DeleteQueueInput(
            queueUrl: queueUrl
        )
    )
} catch _ as AWSSQS.QueueDoesNotExist {
    print("Error: The specified queue doesn't exist.")
    return
}
```

- Para obter detalhes da API, consulte [DeleteQueue](#) a referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **GetQueueAttributes** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetQueueAttributes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Processar notificações de eventos do S3](#)

- [Publicar mensagens em filas](#)

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
/// <summary>
/// Get the ARN for a queue from its URL.
/// </summary>
/// <param name="queueUrl">The URL of the queue.</param>
/// <returns>The ARN of the queue.</returns>
public async Task<string> GetQueueArnByUrl(string queueUrl)
{
    var getAttributesRequest = new GetQueueAttributesRequest()
    {
        QueueUrl = queueUrl,
        AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
    };

    var getAttributesResponse = await
        _amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

    return getAttributesResponse.QueueARN;
}
```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) a Referência AWS SDK para .NET da API.

C++

SDK para C++

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::GetQueueAttributesRequest request;
request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

Aws::SQS::Model::GetQueueAttributesOutcome outcome =
    sqsClient.GetQueueAttributes(request);

if (outcome.IsSuccess()) {
    const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
        outcome.GetResult().GetAttributes();
    const auto &iter = attributes.find(
        Aws::SQS::Model::QueueAttributeName::QueueArn);
    if (iter != attributes.end()) {
        queueARN = iter->second;
        std::cout << "The queue ARN '" << queueARN
            << "' has been retrieved."
            << std::endl;
    }
}
else {
    std::cerr << "Error with SQS::GetQueueAttributes. "
        << outcome.GetError().GetMessage()
        << std::endl;
```

```
}
```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) a Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como obter os atributos de uma fila

Este exemplo obtém todos os atributos da fila especificada.

Comando:

```
aws sqs get-queue-attributes --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names All
```

Saída:

```
{
  "Attributes": {
    "ApproximateNumberOfMessagesNotVisible": "0",
    "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":\"1000\"}",
    "MessageRetentionPeriod": "345600",
    "ApproximateNumberOfMessagesDelayed": "0",
    "MaximumMessageSize": "262144",
    "CreatedTimestamp": "1442426968",
    "ApproximateNumberOfMessages": "0",
    "ReceiveMessageWaitTimeSeconds": "0",
    "DelaySeconds": "0",
    "VisibilityTimeout": "30",
    "LastModifiedTimestamp": "1442426968",
    "QueueArn": "arn:aws:sqs:us-east-1:80398EXAMPLE:MyNewQueue"
  }
}
```

Este exemplo obtém somente os atributos especificados de tempo limite de visibilidade e de tamanho máximo da mensagem da fila.

Comando:

```
aws sqs get-queue-attributes --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyNewQueue --attribute-names MaximumMessageSize VisibilityTimeout
```

Saída:

```
{
  "Attributes": {
    "VisibilityTimeout": "30",
    "MaximumMessageSize": "262144"
  }
}
```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) em Referência de AWS CLI Comandos.

Go

SDK para Go V2

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
```

```
"github.com/aws/aws-sdk-go-v2/service/sqs"
"github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
// (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
(string, error) {
    var queueArn string
    arnAttributeName := types.QueueAttributeNameQueueArn
    attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
        &sqs.GetQueueAttributesInput{
            QueueUrl:      aws.String(queueUrl),
            AttributeNames: []types.QueueAttributeName{arnAttributeName},
        })
    if err != nil {
        log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
    } else {
        queueArn = attribute.Attributes[string(arnAttributeName)]
    }
    return queueArn, err
}
```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) a Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import { GetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const getQueueAttributes = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new GetQueueAttributesCommand({
    QueueUrl: queueUrl,
    AttributeNames: ["DelaySeconds"],
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '747a1192-c334-5682-a508-4cd5e8dc4e79',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: { DelaySeconds: '1' }
  // }
  return response;
};
```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) na Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todos os atributos da fila especificada.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout           : 30
DelaySeconds                : 0
MaximumMessageSize         : 262144
MessageRetentionPeriod     : 345600
ApproximateNumberOfMessages : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed : 0
CreatedTimestamp           : 2/11/2015 5:53:35 PM
LastModifiedTimestamp      : 12/29/2015 2:23:17 PM
QueueARN                   : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                     : {"Version":"2012-10-17",
  "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy", "Statement":
  [{"Sid":"Sid14
    495134224EX", "Effect":"Allow", "Principal":
    {"AWS":"*"}, "Action":"SQS:SendMessage", "Resource":"arn:aws:sqs:us-east-1:80
      398EXAMPLE:MyQueue", "Condition":
    {"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
    {"Sid":
      "SendMessageFromMyQueue", "Effect":"Allow", "Principal":
    {"AWS":"80398EXAMPLE"}, "Action":"SQS:SendMessage", "Resource":":
      arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue"}]}]
Attributes                  : {[QueueArn, arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                              [ApproximateNumberOfMessagesNotVisible,
    0], [ApproximateNumberOfMessagesDelayed, 0]...}
```

Exemplo 2: esse exemplo lista separadamente somente os atributos especificados da fila especificada.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -
QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout           : 30
DelaySeconds                : 0
MaximumMessageSize         : 262144
MessageRetentionPeriod     : 345600
ApproximateNumberOfMessages : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed : 0
CreatedTimestamp           : 2/11/2015 5:53:35 PM
LastModifiedTimestamp      : 12/29/2015 2:23:17 PM
QueueARN                   : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                     : {"Version":"2012-10-17",
  "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
  [{"Sid":"Sid14
    495134224EX","Effect":"Allow","Principal":
    {"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
    398EXAMPLE:MyQueue","Condition":
    {"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}}],
    {"Sid":
      "SendMessageFromMyQueue","Effect":"Allow","Principal":
      {"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":":
      arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                  : {[MaximumMessageSize, 262144],
  [VisibilityTimeout, 30]}
```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os atributos da fila especificada.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```

VisibilityTimeout           : 30
DelaySeconds                : 0
MaximumMessageSize         : 262144
MessageRetentionPeriod     : 345600
ApproximateNumberOfMessages : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed : 0
CreatedTimestamp           : 2/11/2015 5:53:35 PM
LastModifiedTimestamp       : 12/29/2015 2:23:17 PM
QueueARN                   : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                     : {"Version":"2012-10-17",
  "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy", "Statement":
  [{"Sid":"Sid14

    495134224EX", "Effect":"Allow", "Principal":
    {"AWS":"*"}, "Action":"SQS:SendMessage", "Resource":"arn:aws:sqs:us-east-1:80
    398EXAMPLE:MyQueue", "Condition":
    {"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}}],
    {"Sid":

      "SendMessageFromMyQueue", "Effect":"Allow", "Principal":
      {"AWS":"80398EXAMPLE"}, "Action":"SQS:SendMessage", "Resource":":
      arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                  : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                              [ApproximateNumberOfMessagesNotVisible,
0], [ApproximateNumberOfMessagesDelayed, 0]...}

```

Exemplo 2: esse exemplo lista separadamente somente os atributos especificados da fila especificada.

```

Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -
QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue

```

Saída:

```

VisibilityTimeout           : 30
DelaySeconds                : 0

```

```

MaximumMessageSize           : 262144
MessageRetentionPeriod       : 345600
ApproximateNumberOfMessages  : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed : 0
CreatedTimestamp             : 2/11/2015 5:53:35 PM
LastModifiedTimestamp        : 12/29/2015 2:23:17 PM
QueueARN                     : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                       : {"Version":"2012-10-17",
  "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy", "Statement":
  [{"Sid":"Sid14

    495134224EX", "Effect":"Allow", "Principal":
    {"AWS":"*"}, "Action":"SQS:SendMessage", "Resource":"arn:aws:sqs:us-east-1:80
    398EXAMPLE:MyQueue", "Condition":
    {"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
    {"Sid":

      "SendMessageFromMyQueue", "Effect":"Allow", "Principal":
      {"AWS":"80398EXAMPLE"}, "Action":"SQS:SendMessage", "Resource":":
      arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                   : {[MaximumMessageSize, 262144],
  [VisibilityTimeout, 30]}

```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

```

```
def __init__(self, sqs_client: Any) -> None:
    """
    Initialize the SqsWrapper.

    :param sqs_client: A Boto3 Amazon SQS client.
    """
    self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)

    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a [GetQueueAttributes](#) Referência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import AWSSQS

let config = try await SQSClient.SQSClientConfiguration(region: region)
let sqsClient = SQSClient(config: config)

let output = try await sqsClient.getQueueAttributes(
    input: GetQueueAttributesInput(
        attributeNames: [
            .approximateNumberOfMessages,
            .maximumMessageSize
        ],
        queueUrl: url
    )
)

guard let attributes = output.attributes else {
    print("No queue attributes returned.")
    return
}

for (attr, value) in attributes {
    switch(attr) {
    case "ApproximateNumberOfMessages":
        print("Approximate message count: \(value)")
    case "MaximumMessageSize":
        print("Maximum message size: \(value)kB")
    }
}
```

```
        default:
            continue
    }
}
```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) a referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **GetQueueUrl** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetQueueUrl`.

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class GetQueueUrl
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// GetQueueUrlAsync method to retrieve the URL of an Amazon SQS
    /// queue.
    /// </summary>
    public static async Task Main()
```

```
{
    // If the Amazon SQS message queue is not in the same AWS Region as
    your
    // default user, you need to provide the AWS Region as a parameter to
    the
    // client constructor.
    var client = new AmazonSQSClient();

    string queueName = "New-Example-Queue";

    try
    {
        var response = await client.GetQueueUrlAsync(queueName);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"The URL for {queueName} is:
{response.QueueUrl}");
        }
    }
    catch (QueueDoesNotExistException ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine($"The queue {queueName} was not found.");
    }
}
```

- Para obter detalhes da API, consulte [GetQueueUrl](#) Referência AWS SDK para .NET da API.

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).


```

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    /** Get the URL for an Amazon Simple Queue Service (Amazon SQS) queue.
    */
    \param queueName: An Amazon SQS queue name.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::SQS::getQueueUrl(const Aws::String &queueName,
                              const Aws::Client::ClientConfiguration
                              &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::GetQueueUrlRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::GetQueueUrlOutcome outcome =
    sqsClient.GetQueueUrl(request);
    if (outcome.IsSuccess()) {
        std::cout << "Queue " << queueName << " has url " <<
            outcome.GetResult().GetQueueUrl() << std::endl;
    }
    else {
        std::cerr << "Error getting url for queue " << queueName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Para obter detalhes da API, consulte [GetQueueUrl](#) Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como obter um URL de fila

Este exemplo obtém o URL da fila especificada.

Comando:

```
aws sqs get-queue-url --queue-name MyQueue
```

Saída:

```
{
  "QueueUrl": "https://queue.amazonaws.com/80398EXAMPLE/MyQueue"
}
```

- Para obter detalhes da API, consulte [GetQueueUrl](#) em Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
GetQueueUrlResponse getQueueUrlResponse = sqsClient
    .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
```

- Para obter detalhes da API, consulte [GetQueueUrl](#) em Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obtenha o URL para uma fila do Amazon SQS.

```
import { GetQueueUrlCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const command = new GetQueueUrlCommand({ QueueName: queueName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [GetQueueUrl](#) Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obtenha o URL para uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
};

sqs.getQueueUrl(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [GetQueueUrl](#) Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista o URL da fila com o nome especificado.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [GetQueueUrl](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista o URL da fila com o nome especificado.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [GetQueueUrl](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
def get_queue(name):  
    """  
    Gets an SQS queue by name.  
  
    :param name: The name that was used to create the queue.  
    :return: A Queue object.  
    """  
    try:  
        queue = sqs.get_queue_by_name(QueueName=name)  
        logger.info("Got queue '%s' with URL=%s", name, queue.url)  
    except ClientError as error:  
        logger.exception("Couldn't get queue named %s.", name)  
        raise error  
    else:  
        return queue
```

- Para obter detalhes da API, consulte a [GetQueueUrl](#) Referência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sqs->getqueueurl( iv_queueename = iv_queue_name ).      "  
oo_result is returned for testing purposes. "  
    MESSAGE 'Queue URL retrieved.' TYPE 'I'.  
CATCH /aws1/cx_sqsqueuedoesnotexist.  
    MESSAGE 'The requested queue does not exist.' TYPE 'E'.  
ENDTRY.
```

- Para obter detalhes da API, consulte a [GetQueueUrl](#) referência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListDeadLetterSourceQueues** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListDeadLetterSourceQueues`.

CLI

AWS CLI

Como listar filas de origem de mensagens não entregues

Este exemplo lista as filas associadas à fila de origem de mensagens não entregues especificada.

Comando:

```
aws sqs list-dead-letter-source-queues --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Saída:

```
{
  "queueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue"
  ]
}
```

- Para obter detalhes da API, consulte [ListDeadLetterSourceQueues](#) em Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista todas URLs as filas que dependem da fila especificada como fila de letras mortas.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Para obter detalhes da API, consulte [ListDeadLetterSourceQueues](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista todas URLs as filas que dependem da fila especificada como fila de letras mortas.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Para obter detalhes da API, consulte [ListDeadLetterSourceQueues](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **ListQueues** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o ListQueues.

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;  
// Optional: Set to the AWS Region (overrides config file).  
// clientConfig.region = "us-east-1";  
  
//! List the Amazon Simple Queue Service (Amazon SQS) queues within an AWS  
account.
```



```

    /*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool
    AwsDoc::SQS::listQueues(const Aws::Client::ClientConfiguration
    &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::ListQueuesRequest listQueuesRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::String> allQueueUrls;

        do {
            if (!nextToken.empty()) {
                listQueuesRequest.SetNextToken(nextToken);
            }
            const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
                listQueuesRequest);
            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::String> &queueUrls =
outcome.GetResult().GetQueueUrls();
                allQueueUrls.insert(allQueueUrls.end(),
                                    queueUrls.begin(),
                                    queueUrls.end());

                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing queues: " <<
                    outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        } while (!nextToken.empty());

        std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." <<
std::endl;
        for (const auto &iter: allQueueUrls) {
            std::cout << " " << iter << std::endl;
        }

        return true;
    }

```

```
}
```

- Para obter detalhes da API, consulte [ListQueues](#) na Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar filas

Este exemplo lista todas as filas.

Comando:

```
aws sqs list-queues
```

Saída:

```
{
  "QueueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/TestQueue1",
    "https://queue.amazonaws.com/80398EXAMPLE/TestQueue2"
  ]
}
```

Este exemplo lista somente as filas que começam com “My”.

Comando:

```
aws sqs list-queues --queue-name-prefix My
```

Saída:

```
{
  "QueueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue"
  ]
}
```

```
]
}
```

- Para obter detalhes da API, consulte [ListQueues](#) em Referência de AWS CLI Comandos.

Go

SDK para Go V2

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    sqsClient := sqs.NewFromConfig(sdkConfig)
    fmt.Println("Let's list the queues for your account.")
}
```

```
var queueUrls []string
paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
for paginator.HasMorePages() {
    output, err := paginator.NextPage(ctx)
    if err != nil {
        log.Printf("Couldn't get queues. Here's why: %v\n", err)
        break
    } else {
        queueUrls = append(queueUrls, output.QueueUrls...)
    }
}
if len(queueUrls) == 0 {
    fmt.Println("You don't have any queues!")
} else {
    for _, queueUrl := range queueUrls {
        fmt.Printf("\t\t%v\n", queueUrl)
    }
}
}
```

- Para obter detalhes da API, consulte [ListQueues](#) na Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
```

```
        System.out.println(url);
    }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [ListQueues](#) a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Listar filas do Amazon SQS.

```
import { paginateListQueues, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});

export const main = async () => {
    const paginatedListQueues = paginateListQueues({ client }, {});

    /** @type {string[]} */
    const urls = [];
    for await (const page of paginatedListQueues) {
        const nextUrls = page.QueueUrls?.filter((qurl) => !!qurl) || [];
        urls.push(...nextUrls);
        for (const url of urls) {
            console.log(url);
        }
    }

    return urls;
}
```

```
};
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [ListQueues](#) a Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Listar filas do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {};

sqs.listQueues(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrls);
  }
});
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [ListQueues](#) a Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
suspend fun listQueues() {  
    println("\nList Queues")  
  
    val prefix = "que"  
    val listQueuesRequest =  
        ListQueuesRequest {  
            queueNamePrefix = prefix  
        }  
  
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->  
        val response = sqsClient.listQueues(listQueuesRequest)  
        response.queueUrls?.forEach { url ->  
            println(url)  
        }  
    }  
}
```

- Para obter detalhes da API, consulte a [ListQueues](#) referência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as filas.

```
Get-SQSQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Exemplo 2: esse exemplo lista qualquer fila que comece com o nome especificado.

```
Get-SQSQueue -QueueNamePrefix My
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Para obter detalhes da API, consulte [ListQueues](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as filas.

```
Get-SQSQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Exemplo 2: esse exemplo lista qualquer fila que comece com o nome especificado.

```
Get-SQSQueue -QueueNamePrefix My
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```



```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Para obter detalhes da API, consulte [ListQueues](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
def get_queues(prefix=None):
    """
    Gets a list of SQS queues. When a prefix is specified, only queues with names
    that start with the prefix are returned.

    :param prefix: The prefix used to restrict the list of returned queues.
    :return: A list of Queue objects.
    """
    if prefix:
        queue_iter = sqs.queues.filter(QueueNamePrefix=prefix)
    else:
        queue_iter = sqs.queues.all()
    queues = list(queue_iter)
    if queues:
        logger.info("Got queues: %s", ", ".join([q.url for q in queues]))
    else:
        logger.warning("No queues found.")
    return queues
```

- Para obter detalhes da API, consulte a [ListQueues](#) Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-west-2'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: ['All']
  )
end
```

```
attributes.attributes.each do |key, value|
  puts "#{key}: #{value}"
end
rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'

  sqs_client = Aws::SQS::Client.new(region: region)

  puts 'Listing available queue URLs...'
  list_queue_urls(sqs_client)

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
  #{sts_client.get_caller_identity.account}/#{queue_name}"

  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end
```

- Para obter detalhes da API, consulte [ListQueues](#) Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Recuperar a primeira fila do Amazon SQS listada na região.

```
async fn find_first_queue(client: &Client) -> Result<String, Error> {
    let queues = client.list_queues().send().await?;
    let queue_urls = queues.queue_urls();
    Ok(queue_urls
        .first()
        .expect("No queues in this account and Region. Create a queue to
proceed.")
        .to_string())
}
```

- Para obter detalhes da API, consulte a [ListQueues](#) referência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sqs->listqueues( ).          " oo_result is returned for
testing purposes. "
    MESSAGE 'Retrieved list of queues.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a [ListQueues](#) referência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import AWSSQS

let config = try await SQSClient.SQSClientConfiguration(region: region)
let sqsClient = SQSClient(config: config)

var queues: [String] = []
let outputPages = sqsClient.listQueuesPaginated(
    input: ListQueuesInput()
)

// Each time a page of results arrives, process its contents.

for try await output in outputPages {
    guard let urls = output.queueUrls else {
        print("No queues found.")
        return
    }

    // Iterate over the queue URLs listed on this page, adding them
    // to the `queues` array.

    for queueUrl in urls {
        queues.append(queueUrl)
    }
}
```

- Para obter detalhes da API, consulte [ListQueues](#) a referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PurgeQueue** com uma CLI

Os exemplos de código a seguir mostram como usar o PurgeQueue.

CLI

AWS CLI

Como limpar uma fila

Este exemplo exclui todas as mensagens na fila especificada.

Comando:

```
aws sqs purge-queue --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyNewQueue
```

Saída:

```
None .
```

- Para obter detalhes da API, consulte [PurgeQueue](#) em Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui todas as mensagens da fila especificada.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [PurgeQueue](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui todas as mensagens da fila especificada.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [PurgeQueue](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **ReceiveMessage** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ReceiveMessage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Gerenciar mensagens grandes usando o S3](#)
- [Processar notificações de eventos do S3](#)
- [Publicar mensagens em filas](#)
- [Enviar e receber lotes de mensagens](#)

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receba mensagens de uma fila usando seu URL.

```
/// <summary>
/// Receive messages from a queue by its URL.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>The list of messages.</returns>
```

```

    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

```

Receba uma mensagem de uma fila do Amazon SQS e, em seguida, exclua a mensagem.

```

public static async Task Main()
{
    // If the AWS Region you want to use is different from
    // the AWS Region defined for the default user, supply
    // the specify your AWS Region to the client constructor.
    var client = new AmazonSQSClient();
    string queueName = "Example_Queue";

    var queueUrl = await GetQueueUrl(client, queueName);
    Console.WriteLine($"The SQS queue's URL is {queueUrl}");

    var response = await ReceiveAndDeleteMessage(client, queueUrl);

    Console.WriteLine($"Message: {response.Messages[0]}");
}

/// <summary>
/// Retrieve the queue URL for the queue named in the queueName
/// property using the client object.
/// </summary>
/// <param name="client">The Amazon SQS client used to retrieve the
/// queue URL.</param>
/// <param name="queueName">A string representing name of the queue

```



```

    /// for which to retrieve the URL.</param>
    /// <returns>The URL of the queue.</returns>
    public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
    {
        var request = new GetQueueUrlRequest
        {
            QueueName = queueName,
        };

        GetQueueUrlResponse response = await
client.GetQueueUrlAsync(request);
        return response.QueueUrl;
    }

    /// <summary>
    /// Retrieves the message from the quque at the URL passed in the
    /// queueURL parameters using the client.
    /// </summary>
    /// <param name="client">The SQS client used to retrieve a message.</
param>
    /// <param name="queueUrl">The URL of the queue from which to retrieve
    /// a message.</param>
    /// <returns>The response from the call to ReceiveMessageAsync.</returns>
    public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
    {
        // Receive a single message from the queue.
        var receiveMessageRequest = new ReceiveMessageRequest
        {
            AttributeNames = { "SentTimestamp" },
            MaxNumberOfMessages = 1,
            MessageAttributeNames = { "All" },
            QueueUrl = queueUrl,
            VisibilityTimeout = 0,
            WaitTimeSeconds = 0,
        };

        var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

        // Delete the received message from the queue.
        var deleteMessageRequest = new DeleteMessageRequest
        {

```

```

        QueueUrl = queueUrl,
        ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
    };

    await client.DeleteMessageAsync(deleteMessageRequest);

    return receiveMessageResponse;
}
}

```

- Para obter detalhes da API, consulte [ReceiveMessage](#) a Referência AWS SDK para .NET da API.

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    /*! Receive a message from an Amazon Simple Queue Service (Amazon SQS) queue.
    */
    \param queueUrl: An Amazon SQS queue URL.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::SQS::receiveMessage(const Aws::String &queueUrl,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ReceiveMessageRequest request;

```

```

    request.SetQueueUrl(queueUrl);
    request.SetMaxNumberOfMessages(1);

    const Aws::SQS::Model::ReceiveMessageOutcome outcome =
sqClient.ReceiveMessage(
    request);
    if (outcome.IsSuccess()) {

        const Aws::Vector<Aws::SQS::Model::Message> &messages =
            outcome.GetResult().GetMessages();
        if (!messages.empty()) {
            const Aws::SQS::Model::Message &message = messages[0];
            std::cout << "Received message:" << std::endl;
            std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
            std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
std::endl;
            std::cout << "  Body: " << message.GetBody() << std::endl <<
std::endl;
        }
        else {
            std::cout << "No messages received from queue " << queueUrl <<
std::endl;
        }
    }
    else {
        std::cerr << "Error receiving message from queue " << queueUrl << ": "
            << outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

```

- Para obter detalhes da API, consulte [ReceiveMessage](#) na Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como receber uma mensagem

Este exemplo recebe até 10 mensagens disponíveis e retorna todos os atributos disponíveis.

Comando:

```
aws sqs receive-message --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names All --message-attribute-names All --max-number-of-messages 10
```

Saída:

```
{
  "Messages": [
    {
      "Body": "My first message.",
      "ReceiptHandle": "AQEBzbVv...fqNzFw==",
      "MD5ofBody": "1000f835...a35411fa",
      "MD5ofMessageAttributes": "9424c491...26bc3ae7",
      "MessageId": "d6790f8d-d575-4f01-bc51-40122EXAMPLE",
      "Attributes": {
        "ApproximateFirstReceiveTimestamp": "1442428276921",
        "SenderId": "AIDAIAZKMSNQ7EXAMPLE",
        "ApproximateReceiveCount": "5",
        "SentTimestamp": "1442428276921"
      },
      "MessageAttributes": {
        "PostalCode": {
          "DataType": "String",
          "StringValue": "ABC123"
        },
        "City": {
          "DataType": "String",
          "StringValue": "Any City"
        }
      }
    }
  ]
}
```

Este exemplo recebe a próxima mensagem disponível, retornando somente os SentTimestamp atributos SenderId e, bem como o atributo da PostalCode mensagem.

Comando:

```
aws sqs receive-message --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names SenderId SentTimestamp --message-attribute-names PostalCode
```

Saída:

```
{
  "Messages": [
    {
      "Body": "My first message.",
      "ReceiptHandle": "AQEB6nR4...HzlvZQ==",
      "MD5OfBody": "1000f835...a35411fa",
      "MD5OfMessageAttributes": "b8e89563...e088e74f",
      "MessageId": "d6790f8d-d575-4f01-bc51-40122EXAMPLE",
      "Attributes": {
        "SenderId": "AIDAIASZKMSNQ7TEXAMPLE",
        "SentTimestamp": "1442428276921"
      },
      "MessageAttributes": {
        "PostalCode": {
          "DataType": "String",
          "StringValue": "ABC123"
        }
      }
    }
  ]
}
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) em Referência de AWS CLI Comandos.

Go

SDK para Go V2

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
// queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
    maxMessages int32, waitTime int32) ([]types.Message, error) {
    var messages []types.Message
    result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
        QueueUrl:      aws.String(queueUrl),
        MaxNumberOfMessages: maxMessages,
        WaitTimeSeconds: waitTime,
    })
    if err != nil {
        log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
            err)
    } else {
        messages = result.Messages
    }
    return messages, err
}
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) a Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .numberOfMessages(5)
        .build();
    return sqsClient.receiveMessage(receiveMessageRequest).messages();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receba uma mensagem de uma fila do Amazon SQS.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
```



```
        ReceiptHandle: message.ReceiptHandle,
      })),
    }},
  );
}
};
```

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
import { ReceiveMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new ReceiveMessageCommand({
    AttributeNames: ["SentTimestamp"],
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ["All"],
    QueueUrl: queueUrl,
    // The duration (in seconds) for which the call waits for a message
    // to arrive in the queue before returning. If a message is available,
    // the call returns sooner than WaitTimeSeconds. If no messages are
    // available and the wait time expires, the call returns successfully
    // with an empty list of messages.
    // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/
    API_ReceiveMessage.html#API_ReceiveMessage_RequestSyntax
    WaitTimeSeconds: 20,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) a Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  WaitTimeSeconds: 20,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [ReceiveMessage](#) na Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- Para obter detalhes da API, consulte a [ReceiveMessage](#) referência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista as informações de até as próximas 10 mensagens a serem recebidas na fila especificada. As informações conterão valores para os atributos de mensagem especificados, se existirem.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
  StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
Attributes          : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
  1451495923744]}
Body                : Information about John Doe's grade.
MD5ofBody           : ea572796e3c231f974fe75d89EXAMPLE
MD5ofMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes    : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
  [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId           : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle       : AQEBpfGp...20Q5cg==
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as informações de até as próximas 10 mensagens a serem recebidas na fila especificada. As informações conterão valores para os atributos de mensagem especificados, se existirem.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
  StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
Attributes          : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
  1451495923744]}
Body                : Information about John Doe's grade.
MD5ofBody           : ea572796e3c231f974fe75d89EXAMPLE
MD5ofMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes    : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
  [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId           : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle       : AQEBpfGp...20Q5cg==
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual
    number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning.
    When
                       this number is greater than zero, long polling is used.
    This
                       can result in reduced costs and fewer false empty
    responses.
    :return: The list of Message objects received. These each contain the body
            of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
```

```
else:
    return messages
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)

    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
    List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
```

```

        QueueUrl=queue_url,
        MaxNumberOfMessages=max_messages,
        WaitTimeSeconds=2, # Short polling
        MessageAttributeNames=['All']
    )

    messages = response.get('Messages', [])
    logger.info(f"Received {len(messages)} messages from {queue_url}")
    return messages

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error receiving messages: {error_code} - {e}")
    raise

```

- Para obter detalhes da API, consulte a [ReceiveMessage](#) Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example

```

```
# receive_messages(
#   Aws::SQS::Client.new(region: 'us-west-2'),
#   'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#   10
# )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)
  if max_number_of_messages > 10
    puts 'Maximum number of messages to receive must be 10 or less. ' \
      'Stopping program.'
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )

  if response.messages.count.zero?
    puts 'No messages to receive, or all messages have already ' \
      'been previously received.'
    return
  end

  response.messages.each do |message|
    puts '-' * 20
    puts "Message body: #{message.body}"
    puts "Message ID:  #{message.message_id}"
  end
rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
```



```

queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

sqs_client = Aws::SQS::Client.new(region: region)

puts "Receiving messages from queue '#{queue_name}'..."

receive_messages(sqs_client, queue_url, max_number_of_messages)
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__

```

- Para obter detalhes da API, consulte [ReceiveMessage](#) a Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

async fn receive(client: &Client, queue_url: &String) -> Result<(), Error> {
    let rcv_message_output =
        client.receive_message().queue_url(queue_url).send().await?;

    println!("Messages from queue with url: {}", queue_url);

    for message in rcv_message_output.messages.unwrap_or_default() {
        println!("Got the message: {:#?}", message);
    }

    Ok(())
}

```

- Para obter detalhes da API, consulte a [ReceiveMessage](#) referência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Receba uma mensagem de uma fila do Amazon SQS.

```
TRY.
    oo_result = lo_sqs->receivemessage( iv_queueurl = iv_queue_url ).      "
oo_result is returned for testing purposes. "
    DATA(lt_messages) = oo_result->get_messages( ).
    MESSAGE 'Message received from SQS queue.' TYPE 'I'.
    CATCH /aws1/cx_sqsoverlimit.
        MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
ENDTRY.
```

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
TRY.
    oo_result = lo_sqs->receivemessage(                                " oo_result is returned for
testing purposes. "
        iv_queueurl = iv_queue_url
        iv_waittimeseconds = iv_wait_time ).      " Time in seconds for
long polling, such as how long the call waits for a message to arrive in the
queue before returning. " ).
    DATA(lt_messages) = oo_result->get_messages( ).
    MESSAGE 'Message received from SQS queue.' TYPE 'I'.
    CATCH /aws1/cx_sqsoverlimit.
        MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a [ReceiveMessage](#) referência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import AWSSQS

let config = try await SQSClient.SQSClientConfiguration(region: region)
let sqsClient = SQSClient(config: config)

let output = try await sqsClient.receiveMessage(
    input: ReceiveMessageInput(
        maxNumberOfMessages: maxMessages,
        queueUrl: url
    )
)

guard let messages = output.messages else {
    print("No messages received.")
    return
}

for message in messages {
    print("Message ID:      \(message.messageId ?? "<unknown>")")
    print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
    print(message.body ?? "<body missing>")
    print("---")
}
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) a referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **RemovePermission** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemovePermission`.

CLI

AWS CLI

Como remover uma permissão

Este exemplo remove a permissão com o rótulo especificado da fila especificada.

Comando:

```
aws sqs remove-permission --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --label SendMessageFromMyQueue
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte [RemovePermission](#) em Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove as configurações permissão com o rótulo especificado da fila especificada.

```
Remove-SQSPermission -Label SendMessageFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [RemovePermission](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove as configurações de permissão com o rótulo especificado da fila especificada.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [RemovePermission](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **SendMessage** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SendMessage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Gerenciar mensagens grandes usando o S3](#)
- [Enviar e receber lotes de mensagens](#)

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Crie uma fila do Amazon SQS e envie uma mensagem para ela.

```
using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;
```

```

using Amazon;
using Amazon.SQS;
using Amazon.SQS.Model;

public class CreateSendExample
{
    // Specify your AWS Region (an example Region is shown).
    private static readonly string QueueName = "Example_Queue";
    private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
    private static IAmazonSQS client;

    public static async Task Main()
    {
        client = new AmazonSQSClient(ServiceRegion);
        var createQueueResponse = await CreateQueue(client, QueueName);

        string queueUrl = createQueueResponse.QueueUrl;

        Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
        {
            { "Title",    new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
            { "Author",  new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
            { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } },
        };

        string messageBody = "Information about current NY Times fiction
bestseller for week of 12/11/2016.";

        var sendMsgResponse = await SendMessage(client, queueUrl,
messageBody, messageAttributes);
    }

    /// <summary>
    /// Creates a new Amazon SQS queue using the queue name passed to it
    /// in queueName.
    /// </summary>
    /// <param name="client">An SQS client object used to send the message.</
param>
    /// <param name="queueName">A string representing the name of the queue

```

```

    /// to create.</param>
    /// <returns>A CreateQueueResponse that contains information about the
    /// newly created queue.</returns>
    public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS
client, string queueName)
    {
        var request = new CreateQueueRequest
        {
            QueueName = queueName,
            Attributes = new Dictionary<string, string>
            {
                { "DelaySeconds", "60" },
                { "MessageRetentionPeriod", "86400" },
            },
        };

        var response = await client.CreateQueueAsync(request);
        Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

        return response;
    }

    /// <summary>
    /// Sends a message to an SQS queue.
    /// </summary>
    /// <param name="client">An SQS client object used to send the message.</
param>
    /// <param name="queueUrl">The URL of the queue to which to send the
    /// message.</param>
    /// <param name="messageBody">A string representing the body of the
    /// message to be sent to the queue.</param>
    /// <param name="messageAttributes">Attributes for the message to be
    /// sent to the queue.</param>
    /// <returns>A SendMessageResponse object that contains information
    /// about the message that was sent.</returns>
    public static async Task<SendMessageResponse> SendMessage(
        IAmazonSQS client,
        string queueUrl,
        string messageBody,
        Dictionary<string, MessageAttributeValue> messageAttributes)
    {
        var sendMessageRequest = new SendMessageRequest
        {
            DelaySeconds = 10,

```

```

        MessageAttributes = messageAttributes,
        MessageBody = messageBody,
        QueueUrl = queueUrl,
    };

    var response = await client.SendMessageAsync(sendMessageRequest);
    Console.WriteLine($"Sent a message with id : {response.MessageId}");

    return response;
}
}

```

- Para obter detalhes da API, consulte [SendMessage](#) Referência AWS SDK para .NET da API.

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Send a message to an Amazon Simple Queue Service (Amazon SQS) queue.
    /*!
    \param queueUrl: An Amazon SQS queue URL.
    \param messageBody: A message body.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::sendMessage(const Aws::String &queueUrl,
                                const Aws::String &messageBody,

```



```

        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SendMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMessageBody(messageBody);

    const Aws::SQS::Model::SendMessageOutcome outcome =
sqsClient.SendMessage(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message to " << queueUrl <<
            std::endl;
    }
    else {
        std::cerr << "Error sending message to " << queueUrl << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Para obter detalhes da API, consulte [SendMessage](#) a Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para enviar uma mensagem

Este exemplo envia uma mensagem com o corpo da mensagem, o período de atraso e os atributos da mensagem especificados para a fila especificada.

Comando:

```

aws sqs send-message --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --message-body "Information about the largest city in Any Region." --delay-seconds 10 --message-attributes file://send-message.json

```

Arquivo de entrada (send-message.json):

```
{
  "City": {
    "DataType": "String",
    "StringValue": "Any City"
  },
  "Greeting": {
    "DataType": "Binary",
    "BinaryValue": "Hello, World!"
  },
  "Population": {
    "DataType": "Number",
    "StringValue": "1250800"
  }
}
```

Saída:

```
{
  "MD50fMessageBody": "51b0a325...39163aa0",
  "MD50fMessageAttributes": "00484c68...59e48f06",
  "MessageId": "da68f62c-0c07-4bee-bf5f-7e856EXAMPLE"
}
```

- Para obter detalhes da API, consulte [SendMessage](#) em Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Seguem dois exemplos da operação SendMessage:

- Enviar uma mensagem com um corpo e um atraso
- Enviar uma mensagem com um corpo e atributos de mensagem

Enviar uma mensagem com um corpo e um atraso.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <queueName> <message>

            Where:
                queueName - The name of the queue.
                message - The message to send.
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }
}
```

```

    public static void sendMessage(SqsClient sqsClient, String queueName, String
message) {
        try {
            CreateQueueRequest request = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();
            sqsClient.createQueue(request);

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody(message)
                .delaySeconds(5)
                .build();

            sqsClient.sendMessage(sendMsgRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Enviar uma mensagem com um corpo e atributos de mensagem.

```

/**
 * <p>This method demonstrates how to add message attributes to a message.
 * Each attribute must specify a name, value, and data type. You use a Java
Map to supply the attributes. The map's
 * key is the attribute name, and you specify the map's entry value using a
builder that includes the attribute
 * value and data type.</p>
 *
 * <p>The data type must start with one of "String", "Number" or "Binary".
You can optionally
 * define a custom extension by using a "." and your extension.</p>
 *

```

```

    * <p>The SQS Developer Guide provides more information on @see <a
    * href="https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-message-metadata.html#sqs-message-attributes">message
    * attributes</a>.</p>
    *
    * @param thumbnailPath Filesystem path of the image.
    * @param queueUrl      URL of the SQS queue.
    */
    static void sendMessageWithAttributes(Path thumbnailPath, String queueUrl) {
        Map<String, MessageAttributeValue> messageAttributeMap;
        try {
            messageAttributeMap = Map.of(
                "Name", MessageAttributeValue.builder()
                    .stringValue("Jane Doe")
                    .dataType("String").build(),
                "Age", MessageAttributeValue.builder()
                    .stringValue("42")
                    .dataType("Number.int").build(),
                "Image", MessageAttributeValue.builder()

.binaryValue(SdkBytes.fromByteArray(Files.readAllBytes(thumbnailPath)))
                    .dataType("Binary.jpg").build()
            );
        } catch (IOException e) {
            LOGGER.error("An I/O exception occurred reading thumbnail image: {}",
e.getMessage(), e);
            throw new RuntimeException(e);
        }

        SendMessageRequest request = SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody("Hello SQS")
            .messageAttributes(messageAttributeMap)
            .build();

        try {
            SendMessageResponse sendMessageResponse =
SQS_CLIENT.sendMessage(request);
            LOGGER.info("Message ID: {}", sendMessageResponse.messageId());
        } catch (SqsException e) {
            LOGGER.error("Exception occurred sending message: {}",
e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
}

```

- Para obter detalhes da API, consulte [SendMessage](#) Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Enviar uma mensagem para uma fila do Amazon SQS.

```
import { SendMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

export const main = async (sqsQueueUrl = SQS_QUEUE_URL) => {
  const command = new SendMessageCommand({
    QueueUrl: sqsQueueUrl,
    DelaySeconds: 10,
    MessageAttributes: {
      Title: {
        DataType: "String",
        StringValue: "The Whistler",
      },
      Author: {
        DataType: "String",
        StringValue: "John Grisham",
      },
      WeeksOn: {
        DataType: "Number",
        StringValue: "6",
      },
    },
  },
  MessageBody:
```

```
    "Information about current NY Times fiction bestseller for week of  
    12/11/2016.",  
    });  
  
    const response = await client.send(command);  
    console.log(response);  
    return response;  
};
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [SendMessage](#) Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Enviar uma mensagem para uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js  
var AWS = require("aws-sdk");  
// Set the region  
AWS.config.update({ region: "REGION" });  
  
// Create an SQS service object  
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });  
  
var params = {  
    // Remove DelaySeconds parameter and value for FIFO queues  
    DelaySeconds: 10,  
    MessageAttributes: {  
        Title: {  
            DataType: "String",  
            StringValue: "The Whistler",  
        },  
        Author: {
```

```
        DataType: "String",
        StringValue: "John Grisham",
    },
    WeeksOn: {
        DataType: "Number",
        StringValue: "6",
    },
},
MessageBody:
    "Information about current NY Times fiction bestseller for week of
12/11/2016.",
// MessageDeduplicationId: "TheWhistler", // Required for FIFO queues
// MessageGroupId: "Group1", // Required for FIFO queues
QueueUrl: "SQS_QUEUE_URL",
};

sqs.sendMessage(params, function (err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.MessageId);
    }
});
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Para obter detalhes da API, consulte [SendMessage](#) na Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
suspend fun sendMessages(
```



```
        queueUrlVal: String,
        message: String,
    ) {
        println("Sending multiple messages")
        println("\nSend message")
        val sendRequest =
            SendMessageRequest {
                queueUrl = queueUrlVal
                messageBody = message
                delaySeconds = 10
            }

        SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
            sqsClient.sendMessage(sendRequest)
            println("A single message was successfully sent.")
        }
    }

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }

    val msg2 =
        SendMessageBatchRequestEntry {
            id = "id2"
            messageBody = "Hello from msg 2"
        }

    val sendMessageBatchRequest =
        SendMessageBatchRequest {
            queueUrl = queueUrlVal
            entries = listOf(msg1, msg2)
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- Para obter detalhes da API, consulte a [SendMessage](#) referência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo envia uma mensagem com os atributos e o corpo da mensagem especificados para a fila especificada com a entrega da mensagem atrasada por 10 segundos.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

MD5ofMessageAttributes	MD5ofMessageBody	MessageId
-----	-----	-----
1d3e51347bc042efbdf6dda31EXAMPLE c739-4d0c-818b-1820eEXAMPLE	51b0a3256d59467f973009b73EXAMPLE	c35fed8f-

- Para obter detalhes da API, consulte [SendMessage](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo envia uma mensagem com os atributos e o corpo da mensagem especificados para a fila especificada com a entrega da mensagem atrasada por 10 segundos.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

MD5ofMessageAttributes	MD5ofMessageBody	MessageId
-----	-----	-----
1d3e51347bc042efbdf6dda31EXAMPLE c739-4d0c-818b-1820eEXAMPLE	51b0a3256d59467f973009b73EXAMPLE	c35fed8f-

- Para obter detalhes da API, consulte [SendMessage](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
def send_message(queue, message_body, message_attributes=None):
    """
    Send a message to an Amazon SQS queue.

    :param queue: The queue that receives the message.
    :param message_body: The body text of the message.
    :param message_attributes: Custom attributes of the message. These are key-
value
                                pairs that can be whatever you want.
    :return: The response from SQS that contains the assigned message ID.
    """
    if not message_attributes:
        message_attributes = {}

    try:
        response = queue.send_message(
            MessageBody=message_body, MessageAttributes=message_attributes
        )
    except ClientError as error:
        logger.exception("Send message failed: %s", message_body)
        raise error
    else:
        return response
```

- Para obter detalhes da API, consulte a [SendMessage](#)Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  message_body = 'This is my message.'
```

```

sts_client = Aws::STS::Client.new(region: region)

# For example:
# 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

sqs_client = Aws::SQS::Client.new(region: region)

puts "Sending a message to the queue named '#{queue_name}'..."

if message_sent?(sqs_client, queue_url, message_body)
  puts 'Message sent.'
else
  puts 'Message not sent.'
end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__

```

- Para obter detalhes da API, consulte [SendMessage](#) a Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

async fn send(client: &Client, queue_url: &String, message: &SQSMessage) ->
Result<(), Error> {
  println!("Sending message to queue with URL: {}", queue_url);

  let rsp = client
    .send_message()

```

```

        .queue_url(queue_url)
        .message_body(&message.body)
        // If the queue is FIFO, you need to set .message_deduplication_id
        // and message_group_id or configure the queue for
        ContentBasedDeduplication.
        .send()
        .await?;

println!("Send message to the queue: {:#?}", rsp);

Ok(())
}

```

- Para obter detalhes da API, consulte a [SendMessage](#) referência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_sqs->sendmessage(                " oo_result is returned for
testing purposes. "
    iv_queueurl = iv_queue_url
    iv_messagebody = iv_message ).
    MESSAGE 'Message sent to SQS queue.' TYPE 'I'.
CATCH /aws1/cx_sqsinvalidmsgconts.
    MESSAGE 'Message contains non-valid characters.' TYPE 'E'.
CATCH /aws1/cx_sqsunsupportedop.
    MESSAGE 'Operation not supported.' TYPE 'E'.
ENDTRY.

```

- Para obter detalhes da API, consulte a [SendMessage](#) referência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **SendMessageBatch** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SendMessageBatch`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Enviar e receber lotes de mensagens](#)

CLI

AWS CLI

Como enviar várias mensagens como um lote

Este exemplo envia duas mensagens com os corpos da mensagem, os períodos de atraso e os atributos de mensagem especificados para a fila especificada.

Comando:

```
aws sqs send-message-batch --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://send-message-batch.json
```

Arquivo de entrada (send-message-batch.json):

```
[
  {
    "Id": "FuelReport-0001-2015-09-16T140731Z",
    "MessageBody": "Fuel report for account 0001 on 2015-09-16 at 02:07:31 PM.",
    "DelaySeconds": 10,
    "MessageAttributes": {
      "SellerName": {
        "DataType": "String",
        "StringValue": "Example Store"
      },
      "City": {
```



```

        "DataType": "String",
        "StringValue": "Any City"
    },
    "Region": {
        "DataType": "String",
        "StringValue": "WA"
    },
    "PostalCode": {
        "DataType": "String",
        "StringValue": "99065"
    },
    "PricePerGallon": {
        "DataType": "Number",
        "StringValue": "1.99"
    }
}
},
{
    "Id": "FuelReport-0002-2015-09-16T140930Z",
    "MessageBody": "Fuel report for account 0002 on 2015-09-16 at 02:09:30
PM.",
    "DelaySeconds": 10,
    "MessageAttributes": {
        "SellerName": {
            "DataType": "String",
            "StringValue": "Example Fuels"
        },
        "City": {
            "DataType": "String",
            "StringValue": "North Town"
        },
        "Region": {
            "DataType": "String",
            "StringValue": "WA"
        },
        "PostalCode": {
            "DataType": "String",
            "StringValue": "99123"
        },
        "PricePerGallon": {
            "DataType": "Number",
            "StringValue": "1.87"
        }
    }
}
}

```

```
}  
]
```

Saída:

```
{  
  "Successful": [  
    {  
      "MD5fMessageBody": "203c4a38...7943237e",  
      "MD5fMessageAttributes": "10809b55...baf283ef",  
      "Id": "FuelReport-0001-2015-09-16T140731Z",  
      "MessageId": "d175070c-d6b8-4101-861d-adeb3EXAMPLE"  
    },  
    {  
      "MD5fMessageBody": "2cf0159a...c1980595",  
      "MD5fMessageAttributes": "55623928...ae354a25",  
      "Id": "FuelReport-0002-2015-09-16T140930Z",  
      "MessageId": "f9b7d55d-0570-413e-b9c5-a9264EXAMPLE"  
    }  
  ]  
}
```

- Para obter detalhes da API, consulte [SendMessageBatch](#) em Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
SendMessageBatchRequest sendMessageBatchRequest =  
SendMessageBatchRequest.builder()  
    .queueUrl(queueUrl)  
  
    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello  
from msg 1").build(),
```

```
SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg  
2").delaySeconds(10)  
                                .build()  
    .build();  
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Para obter detalhes da API, consulte [SendMessageBatch](#) Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo envia duas mensagens com os atributos especificados e corpos de mensagem para a fila especificada. A entrega é adiada por 15 segundos para a primeira mensagem e 10 segundos para a segunda mensagem.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue  
$student1NameAttributeValue.DataType = "String"  
$student1NameAttributeValue.StringValue = "John Doe"  
  
$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue  
$student1GradeAttributeValue.DataType = "Number"  
$student1GradeAttributeValue.StringValue = "89"  
  
$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue  
$student2NameAttributeValue.DataType = "String"  
$student2NameAttributeValue.StringValue = "Jane Doe"  
  
$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue  
$student2GradeAttributeValue.DataType = "Number"  
$student2GradeAttributeValue.StringValue = "93"  
  
$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry  
$message1.DelaySeconds = 15  
$message1.Id = "FirstMessage"  
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)  
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)  
$message1.MessageBody = "Information about John Doe's grade."
```

```
$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

Saída:

```
Failed      Successful
-----
{}          {FirstMessage, SecondMessage}
```

- Para obter detalhes da API, consulte [SendMessageBatch](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo envia duas mensagens com os atributos especificados e corpos de mensagem para a fila especificada. A entrega é adiada por 15 segundos para a primeira mensagem e 10 segundos para a segunda mensagem.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"

$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"
```

```

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2

```

Saída:

```

Failed      Successful
-----
{}          {FirstMessage, SecondMessage}

```

- Para obter detalhes da API, consulte [SendMessageBatch](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.

```

This request may return overall success even when some messages were not sent.

The caller must inspect the Successful and Failed lists in the response and resend any failed messages.

:param queue: The queue to receive the messages.

:param messages: The messages to send to the queue. These are simplified to contain only the message body and attributes.

:return: The response from SQS that contains the list of successful and failed messages.

```

"""
try:
    entries = [
        {
            "Id": str(ind),
            "MessageBody": msg["body"],
            "MessageAttributes": msg["attributes"],
        }
        for ind, msg in enumerate(messages)
    ]
    response = queue.send_messages(Entries=entries)
    if "Successful" in response:
        for msg_meta in response["Successful"]:
            logger.info(
                "Message sent: %s: %s",
                msg_meta["MessageId"],
                messages[int(msg_meta["Id"])]["body"],
            )
    if "Failed" in response:
        for msg_meta in response["Failed"]:
            logger.warning(
                "Failed to send: %s: %s",
                msg_meta["MessageId"],
                messages[int(msg_meta["Id"])]["body"],
            )
except ClientError as error:
    logger.exception("Send messages failed to queue: %s", queue)
    raise error
else:
    return response

```

- Para obter detalhes da API, consulte a [SendMessageBatch](#) Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
```

```
sqs_client.send_message_batch(
  queue_url: queue_url,
  entries: entries
)
true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
  end
end
```


- Para obter detalhes da API, consulte [SendMessageBatch](#) a Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sqs->sendmessagebatch(          " oo_result is returned for
testing purposes. "
        iv_queueurl = iv_queue_url
        it_entries = it_messages ).
    MESSAGE 'Messages sent to SQS queue.' TYPE 'I'.
CATCH /aws1/cx_sqsbtcentidsnotdist00.
    MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
CATCH /aws1/cx_sqsbatchreqtoolong.
    MESSAGE 'The length of all the messages put together is more than the
limit.' TYPE 'E'.
CATCH /aws1/cx_sqsemptybatchrequest.
    MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
CATCH /aws1/cx_sqsinvbatchentryid.
    MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE
'E'.
CATCH /aws1/cx_sqstoomanyentriesin00.
    MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
CATCH /aws1/cx_sqsunsupportedop.
    MESSAGE 'Operation not supported.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a [SendMessageBatch](#) referência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **SetQueueAttributes** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetQueueAttributes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Publicar mensagens em filas](#)

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Definir o atributo de política de uma fila para um tópico.

```
/// <summary>
/// Set the policy attribute of a queue for a topic.
/// </summary>
/// <param name="queueArn">The ARN of the queue.</param>
/// <param name="topicArn">The ARN of the topic.</param>
/// <param name="queueUrl">The url for the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
{
    var queuePolicy = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
            "\"Effect\": \"Allow\", " +
            "\"Principal\": { " +
                "\"Service\": " +
                    "\"sns.amazonaws.com\"" +
```

```

        "}," +
        "\"Action\": \"sqs:SendMessage\"," +
        $"\"Resource\": \"{queueArn}\"," +
        "\"Condition\": {" +
        "    \"ArnEquals\": {" +
        "        \"$\"aws:SourceArn\":
    \"{topicArn}\"" +
        "    }" +
        "}" +
        "}]";
    var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
        new SetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
        });
    return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
}

```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) na Referência AWS SDK para .NET da API.

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

//! Set the value for an attribute in an Amazon Simple Queue Service (Amazon SQS)
queue.

```

```

/ *!
\param queueUrl: An Amazon SQS queue URL.
\param attributeName: An attribute name enum.
\param attribute: The attribute value as a string.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SQS::setQueueAttributes(const Aws::String &queueURL,
                                     Aws::SQS::Model::QueueAttributeName
                                     attributeName,
                                     const Aws::String &attribute,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
        attributeName,
        attribute);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
sqsClient.SetQueueAttributes(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set the attribute " <<

    Aws::SQS::Model::QueueAttributeNameMapper::GetNameForQueueAttributeName(
        attributeName)
        << " with value " << attribute << " in queue " <<
        queueURL << "." << std::endl;
    }
    else {
        std::cout << "Error setting attribute for queue " <<
        queueURL << ": " << outcome.GetError().GetMessage() <<
        std::endl;
    }

    return outcome.IsSuccess();
}

```

Configurar uma dead-letter queue.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Connect an Amazon Simple Queue Service (Amazon SQS) queue to an associated
    //! dead-letter queue.
    /*!
    \param srcQueueUrl: An Amazon SQS queue URL.
    \param deadLetterQueueARN: The Amazon Resource Name (ARN) of an Amazon SQS
    dead-letter queue.
    \param maxReceiveCount: The max receive count of a message before it is sent to
    the dead-letter queue.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::setDeadLetterQueue(const Aws::String &srcQueueUrl,
                                         const Aws::String &deadLetterQueueARN,
                                         int maxReceiveCount,
                                         const Aws::Client::ClientConfiguration
                                         &clientConfiguration) {
        Aws::String redrivePolicy = MakeRedrivePolicy(deadLetterQueueARN,
maxReceiveCount);

        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::SetQueueAttributesRequest request;
        request.SetQueueUrl(srcQueueUrl);
        request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
            redrivePolicy);

        const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);
        if (outcome.IsSuccess()) {
            std::cout << "Successfully set dead letter queue for queue " <<
                srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
        }
        else {
            std::cerr << "Error setting dead letter queue for queue " <<
                srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
    }

```

```

        return outcome.IsSuccess();
    }

    //! Make a redrive policy for a dead-letter queue.
    /*!
    \param queueArn: An Amazon SQS ARN for the dead-letter queue.
    \param maxReceiveCount: The max receive count of a message before it is sent to
    the dead-letter queue.
    \return Aws::String: Policy as JSON string.
    */
    Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
        Aws::Utils::Json::JsonValue redrive_arn_entry;
        redrive_arn_entry.AsString(queueArn);

        Aws::Utils::Json::JsonValue max_msg_entry;
        max_msg_entry.AsInteger(maxReceiveCount);

        Aws::Utils::Json::JsonValue policy_map;
        policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
        policy_map.WithObject("maxReceiveCount", max_msg_entry);

        return policy_map.View().WriteReadable();
    }

```

Configurar uma fila do Amazon SQS para usar sondagem longa.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Set the wait time for an Amazon Simple Queue Service (Amazon SQS) queue poll.
    /*!
    \param queueUrl: An Amazon SQS queue URL.
    \param pollTimeSeconds: The receive message wait time in seconds.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::setQueueLongPollingAttribute(const Aws::String &queueURL,
                                                    const Aws::String
                                                    &pollTimeSeconds,
                                                    const
                                                    Aws::Client::ClientConfiguration &clientConfiguration) {

```

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::SetQueueAttributesRequest request;
request.SetQueueUrl(queueURL);
request.AddAttributes(
    Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
    pollTimeSeconds);

const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
sqsClient.SetQueueAttributes(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully updated long polling time for queue " <<
        queueURL << " to " << pollTimeSeconds << std::endl;
}
else {
    std::cout << "Error updating long polling time for queue " <<
        queueURL << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}

return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) a Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como definir atributos de fila

Este exemplo define a fila especificada com um atraso de entrega de 10 segundos, um tamanho máximo de mensagem de 128 KB (128 KB * 1.024 bytes), um período de retenção de mensagens de 3 dias (3 dias * 24 horas * 60 minutos * 60 segundos), um tempo de espera de recebimento de mensagens de 20 segundos e um tempo limite de visibilidade padrão de 60 segundos. Este exemplo também associa a fila de mensagens não entregues especificada a uma contagem máxima de recebimento de 1.000 mensagens.

Comando:

```
aws sqs set-queue-attributes --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyNewQueue --attributes file://set-queue-attributes.json
```

Arquivo de entrada (set-queue-attributes.json):

```
{
  "DelaySeconds": "10",
  "MaximumMessageSize": "131072",
  "MessageRetentionPeriod": "259200",
  "ReceiveMessageWaitTimeSeconds": "20",
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":\"1000\"}",
  "VisibilityTimeout": "60"
}
```

Saída:

None.

- Para obter detalhes da API, consulte [SetQueueAttributes](#) em Referência de AWS CLI Comandos.

Go

SDK para Go V2

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"
```



```
"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/sqs"
"github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
// to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
// to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
string, queueArn string, topicArn string) error {
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect:    "Allow",
            Action:  "sqs:SendMessage",
            Principal: map[string]string{"Service": "sns.amazonaws.com"},
            Resource: aws.String(queueArn),
            Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
topicArn}},
        }},
    }
    policyBytes, err := json.Marshal(policyDoc)
    if err != nil {
        log.Printf("Couldn't create policy document. Here's why: %v\n", err)
        return err
    }
    _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
        Attributes: map[string]string{
            string(types.QueueAttributeNamePolicy): string(policyBytes),
        },
        QueueUrl: aws.String(queueUrl),
    })
    if err != nil {
```

```
    log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
        queueUrl, err)
}
return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
    Version    string
    Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
    Effect    string
    Action    string
    Principal map[string]string `json:",omitempty"`
    Resource  *string             `json:",omitempty"`
    Condition PolicyCondition     `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string
```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) na Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Configure um Amazon SQS para usar a criptografia do lado do servidor (SSE) usando uma chave do KMS personalizada.

```
public static void addEncryption(String queueName, String kmsMasterKeyAlias)
{
    SqsClient sqsClient = SqsClient.create();

    GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
        .queueName(queueName)
        .build();

    GetQueueUrlResponse getQueueUrlResponse;
    try {
        getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
    } catch (QueueDoesNotExistException e) {
        LOGGER.error(e.getMessage(), e);
        throw new RuntimeException(e);
    }
    String queueUrl = getQueueUrlResponse.queueUrl();

    Map<QueueAttributeName, String> attributes = Map.of(
        QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
        QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" //
    );
    // Set the data key reuse period to 140 seconds.
    // This is how long SQS can reuse the data key before requesting a new one from KMS.

    SetQueueAttributesRequest attRequest =
    SetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)
        .attributes(attributes)
        .build();

    try {
        sqsClient.setQueueAttributes(attRequest);
        LOGGER.info("The attributes have been applied to {}", queueName);
    } catch (InvalidAttributeNameException | InvalidAttributeValueException
e) {
        LOGGER.error(e.getMessage(), e);
        throw new RuntimeException(e);
    } finally {
        sqsClient.close();
    }
}
```

```
    }  
  }  
}
```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";  
  
const client = new SQSClient({});  
const SQS_QUEUE_URL = "queue-url";  
  
export const main = async (queueUrl = SQS_QUEUE_URL) => {  
  const command = new SetQueueAttributesCommand({  
    QueueUrl: queueUrl,  
    Attributes: {  
      DelaySeconds: "1",  
    },  
  });  
  
  const response = await client.send(command);  
  console.log(response);  
  return response;  
};
```

Configurar uma fila do Amazon SQS para usar sondagem longa.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";  
  
const client = new SQSClient({});
```

```
const SQS_QUEUE_URL = "queue_url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    Attributes: {
      ReceiveMessageWaitTimeSeconds: "20",
    },
    QueueUrl: queueUrl,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Configurar uma dead-letter queue.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";
const DEAD_LETTER_QUEUE_ARN = "dead_letter_queue_arn";

export const main = async (
  queueUrl = SQS_QUEUE_URL,
  deadLetterQueueArn = DEAD_LETTER_QUEUE_ARN,
) => {
  const command = new SetQueueAttributesCommand({
    Attributes: {
      RedrivePolicy: JSON.stringify({
        // Amazon SQS supports dead-letter queues (DLQ), which other
        // queues (source queues) can target for messages that can't
        // be processed (consumed) successfully.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
        SQSDeveloperGuide/sqs-dead-letter-queues.html
        deadLetterTargetArn: deadLetterQueueArn,
        maxReceiveCount: "10",
      }),
    },
    QueueUrl: queueUrl,
  });
```

```
const response = await client.send(command);
console.log(response);
return response;
};
```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) a Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra como definir uma política para assinar um tópico do SNS com uma fila. Quando uma mensagem é publicada no tópico, ela é enviada à fila assinada.

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName
"QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
        }
      }
    }
  ]
}
```

```

    }
  }
]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }

```

Exemplo 2: esse exemplo define os atributos especificados da fila especificada.

```

Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
  "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue

```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como definir uma política para assinar um tópico do SNS com uma fila. Quando uma mensagem é publicada no tópico, ela é enviada à fila assinada.

```

# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName
  "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",

```

```

    "Resource": "$qarn",
    "Condition": {
        "ArnEquals": {
            "aws:SourceArn": "$topicarn"
        }
    }
}
]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }

```

Exemplo 2: esse exemplo define os atributos especificados da fila especificada.

```

Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
    "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue

```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Definir o atributo de política de uma fila para um tópico.

```

class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

```



```

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)

    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
        queue_url: str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

```

```

        ]
    }

    self.sqs_client.set_queue_attributes(
        QueueUrl=queue_url,
        Attributes={
            'Policy': json.dumps(policy)
        }
    )

    logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
    return True

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error setting queue policy: {error_code} - {e}")
    raise

```

- Para obter detalhes da API, consulte a [SetQueueAttributes](#) Referência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

import AWSSQS

let config = try await SQSClient.SQSClientConfiguration(region: region)
let sqsClient = SQSClient(config: config)

do {
    _ = try await sqsClient.setQueueAttributes(

```

```
        input: SetQueueAttributesInput(  
            attributes: [  
                "MaximumMessageSize": "\(maxSize)"  
            ],  
            queueUrl: url  
        )  
    )  
} catch _ as AWSSQS.InvalidAttributeValue {  
    print("Invalid maximum message size: \(maxSize) kB.")  
}
```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) a referência da API AWS SDK for Swift.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cenários para o Amazon SQS usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon SQS com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon SQS ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- [Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS](#)
- [Criar uma aplicação de mensageiro com o Step Functions](#)
- [Criar uma aplicação de exploração do Amazon Textract](#)
- [Crie e publique em um tópico FIFO do Amazon SNS usando um SDK AWS](#)
- [Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS](#)
- [Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um SDK AWS](#)
- [Receba e processe notificações de eventos do Amazon S3 usando um SDK AWS](#)

- [Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um SDK AWS](#)
- [Envie e receba lotes de mensagens com o Amazon SQS usando um SDK AWS](#)
- [Use o AWS Message Processing Framework para.NET para publicar e receber mensagens do Amazon SQS](#)
- [Usar a Biblioteca de Mensagens Java do Amazon SQS para trabalhar com a interface do Serviço de Mensagens Java \(JMS\) para o Amazon SQS](#)
- [Trabalhe com tags de fila e Amazon SQS usando um SDK AWS](#)

Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS

Os exemplos de código a seguir mostram como criar uma aplicação de mensagens usando o Amazon SQS.

Java

SDK para Java 2.x

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Kotlin

SDK para Kotlin

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar uma aplicação de mensageiro com o Step Functions

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.
- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Lambda

- Amazon SQS
- Step Functions

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar uma aplicação de exploração do Amazon Textract

Os exemplos de código a seguir mostram como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use o modo assíncrono APIs para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Crie e publique em um tópico FIFO do Amazon SNS usando um SDK AWS

Os exemplos de código a seguir mostram como criar e publicar em um tópico FIFO do Amazon SNS.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O [teste](#) verifica o recebimento da mensagem em cada fila. O [exemplo completo](#) também mostra a adição de políticas de acesso e exclui os recursos no final.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName> <analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will be created for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```



```
final String fifoTopicName = args[0];
final String wholeSaleQueueName = args[1];
final String retailQueueName = args[2];
final String analyticsQueueName = args[3];

// For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
// name and type.
List<QueueData> queues = List.of(
    new QueueData(wholeSaleQueueName, QueueType.FIFO),
    new QueueData(retailQueueName, QueueType.FIFO),
    new QueueData(analyticsQueueName, QueueType.Standard));

// Create queues.
createQueues(queues);

// Create a topic.
String topicARN = createFIFOTopic(fifoTopicName);

// Subscribe each queue to the topic.
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false",
            "FifoThroughputScope", "MessageGroup");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
```

```

        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    String topicArn = response.topicArn();
    System.out.println("The topic ARN is" + topicArn);

    return topicArn;

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
    }

```

```
String attributeName = "business";
String attributeValue = "wholesale";

MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
    .dataType("String")
    .stringValue(attributeValue)
    .build();

Map<String, MessageAttributeValue> attributes = new HashMap<>();
attributes.put(attributeName, msgAttValue);
PublishRequest pubRequest = PublishRequest.builder()
    .topicArn(topicArn)
    .subject(subject)
    .message(payload)
    .messageGroupId(groupId)
    .messageDeduplicationId(dedupId)
    .messageAttributes(attributes)
    .build();

final PublishResponse response = snsClient.publish(pubRequest);
System.out.println(response.messageId());
System.out.println(response.sequenceNumber());
System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Publicar](#)
 - [Assinar](#)

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Crie um tópico FIFO do Amazon SNS, inscreva filas padrão e FIFO do Amazon SQS no tópico e publique uma mensagem no tópico.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
```

```
    QueueName=prefix + "retail.fifo",
    Attributes={
        "MaximumMessageSize": str(4096),
        "ReceiveMessageWaitTimeSeconds": str(10),
        "VisibilityTimeout": str(300),
        "FifoQueue": str(True),
        "ContentBasedDeduplication": str(True),
    },
)
queues.add(retail_queue)
print(f"Created FIFO queue with URL: {retail_queue.url}.")

analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
queues.add(analytics_queue)
print(f"Created standard queue with URL: {analytics_queue.url}.")

topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

for q in queues:
    fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

for q in queues:
    sub = fifo_topic_wrapper.subscribe_queue_to_topic(
        topic, q.attributes["QueueArn"]
    )
    subscriptions.add(sub)

print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

input("Press Enter to publish a message to the topic.")

message_id = fifo_topic_wrapper.publish_price_update(
    topic, '{"product": 214, "price": 79.99}', "Consumables"
)

print(f"Published price update with message ID: {message_id}.")

# Clean up the subscriptions, queues, and topic.
input("Press Enter to clean up resources.")
for s in subscriptions:
```

```

        sns_wrapper.delete_subscription(s)

sns_wrapper.delete_topic(topic)

for q in queues:
    fifo_topic_wrapper.delete_queue(q)

print(f"Deleted subscriptions, queues, and topic.")

print("Thanks for watching!")
print("-" * 88)

class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
            )

```

```

        logger.info("Created FIFO topic with name=%s.", topic_name)
        return topic
    except ClientError as error:
        logger.exception("Couldn't create topic with name=%s!", topic_name)
        raise error

@staticmethod
def add_access_policy(queue, topic_arn):
    """
    Add the necessary access policy to a queue, so
    it can receive messages from a topic.

    :param queue: The queue resource.
    :param topic_arn: The ARN of the topic.
    :return: None.
    """
    try:
        queue.set_attributes(
            Attributes={
                "Policy": json.dumps(
                    {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Sid": "test-sid",
                                "Effect": "Allow",
                                "Principal": {"AWS": "*"},
                                "Action": "SQS:SendMessage",
                                "Resource": queue.attributes["QueueArn"],
                                "Condition": {
                                    "ArnLike": {"aws:SourceArn": topic_arn}
                                }
                            }
                        ],
                    }
                ),
            }
        )
        logger.info("Added trust policy to the queue.")
    except ClientError as error:
        logger.exception("Couldn't add trust policy to the queue!")
        raise error

```

```
@staticmethod
def subscribe_queue_to_topic(topic, queue_arn):
    """
    Subscribe a queue to a topic.

    :param topic: The topic resource.
    :param queue_arn: The ARN of the queue.
    :return: The subscription resource.
    """
    try:
        subscription = topic.subscribe(
            Protocol="sqs",
            Endpoint=queue_arn,
        )
        logger.info("The queue is subscribed to the topic.")
        return subscription
    except ClientError as error:
        logger.exception("Couldn't subscribe queue to topic!")
        raise error


@staticmethod
def publish_price_update(topic, payload, group_id):
    """
    Compose and publish a message that updates the wholesale price.

    :param topic: The topic to publish to.
    :param payload: The message to publish.
    :param group_id: The group ID for the message.
    :return: The ID of the message.
    """
    try:
        att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
        dedup_id = uuid.uuid4()
        response = topic.publish(
            Subject="Price Update",
            Message=payload,
            MessageAttributes=att_dict,
            MessageGroupId=group_id,
            MessageDeduplicationId=str(dedup_id),
        )
        message_id = response["MessageId"]
```



```
        logger.info("Published message to topic %s.", topic.arn)
    except ClientError as error:
        logger.exception("Couldn't publish message to topic %s.", topic.arn)
        raise error
    return message_id

@staticmethod
def delete_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
 - [CreateTopic](#)
 - [Publicar](#)
 - [Assinar](#)

SAP ABAP

SDK para SAP ABAP

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Crie um tópico FIFO, inscreva uma fila FIFO do Amazon SQS no tópico e publique uma mensagem em um tópico do Amazon SNS.

```
" Creates a FIFO topic. "
DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsm_w=>tt_topicattributesmap.
DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsm_w=>ts_topicattributesmap_maprow.
ls_tpc_attributes-key = 'FifoTopic'.
ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsm_w( iv_value =
'true' ).
INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

TRY.
  DATA(lo_create_result) = lo_sns->createtopic(
    iv_name = iv_topic_name
    it_attributes = lt_tpc_attributes ).
  DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
  ov_topic_arn = lv_topic_arn.
  "
  ov_topic_arn is returned for testing purposes. "
  MESSAGE 'FIFO topic created' TYPE 'I'.
  CATCH /aws1/cx_snstopiclimitexcdex.
  MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
ENDTRY.

" Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
SNS) topic. "
" Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
to an SNS FIFO topic. "
TRY.
```

```

        DATA(lo_subscribe_result) = lo_sns->subscribe(
            iv_topicarn = lv_topic_arn
            iv_protocol = 'sqs'
            iv_endpoint = iv_queue_arn ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.
        "
        ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    CATCH /aws1/cx_snssubscriptionlmt00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
        number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
        cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
        cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
        'String'

        iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
            iv_topicarn = lv_topic_arn
            iv_message = 'The price of your mobile plan has been increased from
            $19 to $23'
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes ).
        ov_message_id = lo_result->get_messageid( ).
        "
        ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.

```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para SAP ABAP.
 - [CreateTopic](#)
 - [Publicar](#)
 - [Assinar](#)

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um SDK AWS

O exemplo de código a seguir mostra como usar a biblioteca do cliente em versão ampliada do Amazon SQS para trabalhar com mensagens grandes do SQS.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
```

```
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
        LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
            DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
            ExtendedClientConfiguration()
```

```
        .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody(largeMessage)
                .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
```



```
        if (messages.isEmpty()) {
            throw new RuntimeException("No messages received");
        }

        Message message = messages.getFirst();
        logger.info("\nMessage received.");
        logger.info("  ID: {}", message.messageId());
        logger.info("  Receipt handle: {}", message.receiptHandle());
        logger.info("  Message body size: {}", message.body().length());
        logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

        return message;
    } catch (RuntimeException e) {
        logger.error("Error during message processing: {}", e.getMessage(),
e);
        throw e;
    }
}
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Java 2.x](#).
- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Receba e processe notificações de eventos do Amazon S3 usando um SDK AWS

O exemplo de código a seguir mostra como trabalhar com notificações de eventos do S3 de uma forma orientada a objetos.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Esse exemplo mostra como processar o evento de notificação do S3 usando o Amazon SQS.

```
/**
 * This method receives S3 event notifications by using an SqsAsyncClient.
 * After the client receives the messages it deserializes the JSON payload
and logs them. It uses
 * the S3EventNotification class (part of the S3 event notification API for
Java) to deserialize
 * the JSON payload and access the messages in an object-oriented way.
 *
 * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
notifications.
 * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
API</a>.
 * <p>
 * To use S3 event notification serialization/deserialization to objects, add
the following
 * dependency to your Maven pom.xml file.
 * <dependency>
 * <groupId>software.amazon.awssdk</groupId>
 * <artifactId>s3-event-notifications</artifactId>
 * <version><LATEST></version>
 * </dependency>
 * <p>
 * The S3 event notification API became available with version 2.25.11 of the
Java SDK.
 * <p>
 * This example shows the use of the API with AWS SQS, but it can be used to
process S3 event notifications
 * in AWS SNS or AWS Lambda as well.
 * <p>
```

```

    * Note: The S3EventNotification class does not work with messages routed
    through AWS EventBridge.
    */
    static void processS3Events(String bucketName, String queueUrl, String
queueArn) {
        try {
            // Configure the bucket to send Object Created and Object Tagging
            notifications to an existing SQS queue.
            s3Client.putBucketNotificationConfiguration(b -> b
                .notificationConfiguration(ncb -> ncb
                    .queueConfigurations(qcb -> qcb
                        .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                            .queueArn(queueArn)))
                    .bucket(bucketName)
                ).join();

            triggerS3EventNotifications(bucketName);
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {
                // Display the number of messages that are available in the
                queue.
                sqsClient.getQueueAttributes(b -> b
                    .queueUrl(queueUrl)

                    .attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                        .thenAccept(attributeResponse ->
                            logger.info("Approximate number of messages in
the queue: {}",
attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                    ).join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                    .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();
            }
        }
    }

```

```

        // Create a collection to hold the received message for deletion
        // after we log the messages.
        HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
        // Process each message.
        response.messages().forEach(message -> {
            logger.info("Message id: {}", message.messageId());
            // Deserialize JSON message body to a S3EventNotification
object
            // to access messages in an object-oriented way.
            S3EventNotification event =
S3EventNotification.fromJson(message.body());

            // Log the S3 event notification record details.
            if (event.getRecords() != null) {
                event.getRecords().forEach(record -> {
                    String eventName = record.getEventName();
                    String key = record.getS3().getObject().getKey();
                    logger.info(record.toString());
                    logger.info("Event name is {} and key is {}",
eventName, key);
                });
            }
            // Add logged messages to collection for batch deletion.
            messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                .id(message.messageId())
                .receiptHandle(message.receiptHandle())
                .build());
        });
        // Delete messages.
        if (!messagesToDelete.isEmpty()) {

sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
            .queueUrl(queueUrl)
            .entries(messagesToDelete)
            .build()
        ).join();
        }
    } // End of while block.
} catch (InterruptedException | ExecutionException e) {
    throw new RuntimeException(e);
}
}

```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um SDK AWS

Os exemplos de código a seguir mostram como:

- Crie um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Execute um cenário interativo em um prompt de comando.

```
/// <summary>  
/// Console application to run a feature scenario for topics and queues.
```

```

/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
"sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>

```

```
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
    SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
}

/// <summary>
/// Run the scenario for working with topics and queues.
/// </summary>
/// <returns>True if successful.</returns>
public static async Task<bool> RunScenario()
{
    try
    {
        await SetupTopic();

        await SetupQueues();

        await PublishMessages();

        foreach (var queueUrl in _queueUrls)
        {
            var messages = await PollForMessages(queueUrl);
            if (messages.Any())
            {
                await DeleteMessages(queueUrl, messages);
            }
        }
        await CleanupResources();

        Console.WriteLine("Messaging with topics and queues scenario is
complete.");
        return true;
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
        await CleanupResources();
        Console.WriteLine(new string('-', 80));
        return false;
    }
}
```

```

    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
            $"{r\nYou can select from several options for
configuring the topic and the subscriptions for the 2 queues." +
            $"{r\nYou can then post to the topic and see the
results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
            $"{r\nFIFO topics deliver messages in order and support
deduplication and message filtering." +
            $"{r\nYou can then post to the topic and see the
results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
"example-topic");

```



```

        Console.WriteLine(
            "Because you have selected a FIFO topic, '.fifo' must be appended
to the topic name.\r\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Because you have chosen a FIFO topic,
deduplication is supported." +
            $"\r\nDeduplication IDs are either set in the
message or automatically generated " +
            $"\r\nfrom content using a hash function.\r\n" +
            $"\r\nIf a message is successfully published to an
SNS FIFO topic, any message " +
            $"\r\npublished and determined to have the same
deduplication ID, " +
            $"\r\nwithin the five-minute deduplication
interval, is accepted but not delivered.\r\n" +
            $"\r\nFor more information about deduplication, " +
            $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

        _useContentBasedDeduplication = GetYesNoResponse("Use content-based
deduplication instead of entering a deduplication ID?");
        Console.WriteLine(new string('-', 80));
    }

    _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
_useFifoTopic, _useContentBasedDeduplication);

    Console.WriteLine($"Your new topic with the name {_topicName}" +
        $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
        $"\r\nhas been created.\r\n");

    Console.WriteLine(new string('-', 80));
    return _topicArn;
}

/// <summary>
/// Set up the queues.
/// </summary>
/// <returns>Async task.</returns>
private static async Task SetupQueues()
{
    Console.WriteLine(new string('-', 80));

```

```
    Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

    // Repeat this section for each queue.
    for (int i = 0; i < _queueCount; i++)
    {
        var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
        if (_useFifoTopic)
        {
            // Only explain this once.
            if (i == 0)
            {
                Console.WriteLine(
                    "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
            }

            var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

            _queueUrls[i] = queueUrl;

            Console.WriteLine($"Your new queue with the name {queueName}" +
                $"{r\n}and queue URL {queueUrl}" +
                $"{r\n}has been created.{r\n}");

            if (i == 0)
            {
                Console.WriteLine(
                    $"The queue URL is used to retrieve the queue ARN,{r\n}" +
                    $"which is used to create a subscription.");
                Console.WriteLine(new string('-', 80));
            }

            var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

            if (i == 0)
            {
                Console.WriteLine(
                    $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive{r\n}" +
                    $"messages from an SNS topic");
            }
        }
    }
}
```

```
        await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

        await SetupFilters(i, queueArn, queueName);
    }
}

Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Set up filters with user options for a queue.
/// </summary>
/// <param name="queueCount">The number of this queue.</param>
/// <param name="queueArn">The ARN of the queue.</param>
/// <param name="queueName">The name of the queue.</param>
/// <returns>Async Task.</returns>
public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
{
    if (_useFifoTopic)
    {
        Console.WriteLine(new string('-', 80));
        // Only explain this once.
        if (queueCount == 0)
        {
            Console.WriteLine(
                "Subscriptions to a FIFO topic can have filters." +
                "If you add a filter to this subscription, then only the
filtered messages " +
                "will be received in the queue.");

            Console.WriteLine(
                "For information about message filtering, " +
                "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

            Console.WriteLine(
                "For this example, you can filter messages by a " +
                "TONE attribute.");
        }
    }
}
```

```

        var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
subscription to the topic?");

        string? filterPolicy = null;
        if (useFilter)
        {
            filterPolicy = CreateFilterPolicy();
        }
        var subscriptionArn = await
SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
            queueArn);
        _subscriptionArns[queueCount] = subscriptionArn;

        Console.WriteLine(
            $"The queue {queueName} has been subscribed to the topic
{_topicName} " +
            $"with the subscription ARN {subscriptionArn}");
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Use user input to create a filter policy for a subscription.
/// </summary>
/// <returns>The serialized filter policy.</returns>
public static string CreateFilterPolicy()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine(
        $"You can filter messages by one or more of the following" +
        $"TONE attributes.");

    List<string> filterSelections = new List<string>();

    var selectionNumber = 0;
    do
    {
        Console.WriteLine(
            $"Enter a number to add a TONE filter, or enter 0 to stop adding
filters.");
        for (int i = 0; i < _tones.Length; i++)
        {
            Console.WriteLine($"\\t{i + 1}. {_tones[i]}");
        }
    }
}

```

```

        var selection = GetUserResponse("", filterSelections.Any() ? "0" :
"1");
        int.TryParse(selection, out selectionNumber);
        if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
        {
            filterSelections.Add(_tones[selectionNumber - 1]);
        }
    } while (selectionNumber != 0);

    var filters = new Dictionary<string, List<string>>
    {
        { "tone", filterSelections }
    };
    string filterPolicy = JsonSerializer.Serialize(filters);
    return filterPolicy;
}

/// <summary>
/// Publish messages using user settings.
/// </summary>
/// <returns>Async task.</returns>
public static async Task PublishMessages()
{
    Console.WriteLine("Now we can publish messages.");

    var keepSendingMessages = true;
    string? deduplicationId = null;
    string? toneAttribute = null;
    while (keepSendingMessages)
    {
        Console.WriteLine();
        var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

        if (_useFifoTopic)
        {
            Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
"\r\nAll messages within the same group will be
received in the order " +
"they were published.");

```

```
        Console.WriteLine();
        var messageId = GetUserResponse("Enter a message group ID
for this message:", "1");

        if (!_useContentBasedDeduplication)
        {
            Console.WriteLine("Because you are not using content-based
deduplication, " +
                               "you must enter a deduplication ID.");

            Console.WriteLine("Enter a deduplication ID for this
message.");
            deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
        }

        if (GetYesNoResponse("Add an attribute to this message?"))
        {
            Console.WriteLine("Enter a number for an attribute.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", "1");
            int.TryParse(selection, out var selectionNumber);

            if (selectionNumber > 0 && selectionNumber < _tones.Length)
            {
                toneAttribute = _tones[selectionNumber - 1];
            }
        }

        var messageId = await SnsWrapper.PublishToTopicWithAttribute(
            _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

        Console.WriteLine($"Message published with id {messageID}.");
    }

    keepSendingMessages = GetYesNoResponse("Send another message?",
false);
}
}
```

```
/// <summary>
/// Poll for the published messages to see the results of the user's choices.
/// </summary>
/// <returns>Async task.</returns>
public static async Task<List<Message>> PollForMessages(string queueUrl)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
        "\r\nPress any key to continue.");
    if (UseConsole)
    {
        Console.ReadLine();
    }

    var moreMessages = true;
    var messages = new List<Message>();
    while (moreMessages)
    {
        var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

        moreMessages = newMessages.Any();
        if (moreMessages)
        {
            messages.AddRange(newMessages);
        }
    }

    Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

    foreach (var message in messages)
    {
        Console.WriteLine("\tMessage:" +
            $"{message.Body}");
    }

    Console.WriteLine(new string('-', 80));
    return messages;
}

/// <summary>
```

```
/// Delete the message using handles in a batch.
/// </summary>
/// <returns>Async task.</returns>
public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
    await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <returns>Async task.</returns>
private static async Task CleanupResources()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"Clean up resources.");

    try
    {
        foreach (var queueUrl in _queueUrls)
        {
            if (!string.IsNullOrEmpty(queueUrl))
            {
                var deleteQueue =
                    GetYesNoResponse($"Delete queue with url {queueUrl}?");
                if (deleteQueue)
                {
                    await SqsWrapper.DeleteQueueByUrl(queueUrl);
                }
            }
        }

        foreach (var subscriptionArn in _subscriptionArns)
        {
            if (!string.IsNullOrEmpty(subscriptionArn))
            {
                await SnsWrapper.UnsubscribeByArn(subscriptionArn);
            }
        }
    }
}
```



```

        var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
        if (deleteTopic)
        {
            await SnsWrapper.DeleteTopicByArn(_topicArn);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Unable to clean up resources. Here's why:
{ex.Message}.");
    }

    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <param name="defaultAnswer">Optional default answer to use.</param>
/// <returns>True if the user responds with a yes.</returns>
private static bool GetYesNoResponse(string question, bool defaultAnswer =
true)
{
    if (UseConsole)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
            ynResponse.Equals("y",
                StringComparison.InvariantCultureIgnoreCase);

        return response;
    }
    // If not using the console, use the default.
    return defaultAnswer;
}

/// <summary>
/// Helper method to get a string response from the user through the console.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>

```

```

    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}

```

Crie uma classe que envolva operações do Amazon SQS.

```

    /// <summary>
    /// Wrapper for Amazon Simple Queue Service (SQS) operations.
    /// </summary>
    public class SQSWrapper
    {
        private readonly IAmazonSQS _amazonSQSClient;

        /// <summary>
        /// Constructor for the Amazon SQS wrapper.
        /// </summary>
        /// <param name="amazonSQS">The injected Amazon SQS client.</param>
        public SQSWrapper(IAmazonSQS amazonSQS)
        {
            _amazonSQSClient = amazonSQS;
        }

        /// <summary>
        /// Create a queue with a specific name.
        /// </summary>
        /// <param name="queueName">The name for the queue.</param>

```

```
/// <param name="useFifoQueue">True to use a FIFO queue.</param>
/// <returns>The url for the queue.</returns>
public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
{
    int maxMessage = 256 * 1024;
    var queueAttributes = new Dictionary<string, string>
    {
        {
            QueueAttributeName.MaximumMessageSize,
            maxMessage.ToString()
        }
    };

    var createQueueRequest = new CreateQueueRequest()
    {
        QueueName = queueName,
        Attributes = queueAttributes
    };

    if (useFifoQueue)
    {
        // Update the name if it is not correct for a FIFO queue.
        if (!queueName.EndsWith(".fifo"))
        {
            createQueueRequest.QueueName = queueName + ".fifo";
        }

        // Add an attribute for a FIFO queue.
        createQueueRequest.Attributes.Add(
            QueueAttributeName.FifoQueue, "true");
    }

    var createResponse = await _amazonSQSClient.CreateQueueAsync(
        new CreateQueueRequest()
        {
            QueueName = queueName
        });
    return createResponse.QueueUrl;
}

/// <summary>
/// Get the ARN for a queue from its URL.
/// </summary>
```

```

    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
            _amazonSQSClient.GetQueueAttributesAsync(
                getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
        topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                    "\"Service\": " +
                        "\"sns.amazonaws.com\"" +
                    "}," +
                "\"Action\": \"sqs:SendMessage\"," +
                "\"Resource\": \"{queueArn}\"," +
                "\"Condition\": {" +
                    "\"ArnEquals\": {" +
                        "\"aws:SourceArn\":
\\\"{topicArn}\"" +
                    "}" +
                "}" +
            "}]}" +

```

```

        "});
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,

```

```

        Entries = new List<DeleteMessageBatchRequestEntry>()
    };
    foreach (var message in messages)
    {
        deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
        {
            ReceiptHandle = message.ReceiptHandle,
            Id = message.MessageId
        });
    }

    var deleteResponse = await
        _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

    return deleteResponse.Failed.Any();
}

/// <summary>
/// Delete a queue by its URL.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteQueueByUrl(string queueUrl)
{
    var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
        new DeleteQueueRequest()
        {
            QueueUrl = queueUrl
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
}

```

Crie uma classe que envolva operações do Amazon SNS.

```

/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

```

```
/// <summary>
/// Constructor for the Amazon SNS wrapper.
/// </summary>
/// <param name="amazonSQS">The injected Amazon SNS client.</param>
public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
{
    _amazonSNSClient = amazonSNS;
}

/// <summary>
/// Create a new topic with a name and specific FIFO and de-duplication
attributes.
/// </summary>
/// <param name="topicName">The name for the topic.</param>
/// <param name="useFifoTopic">True to use a FIFO topic.</param>
/// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
/// <returns>The ARN of the new topic.</returns>
public async Task<string> CreateTopicWithName(string topicName, bool
useFifoTopic, bool useContentBasedDeduplication)
{
    var createTopicRequest = new CreateTopicRequest()
    {
        Name = topicName,
    };

    if (useFifoTopic)
    {
        // Update the name if it is not correct for a FIFO topic.
        if (!topicName.EndsWith(".fifo"))
        {
            createTopicRequest.Name = topicName + ".fifo";
        }

        // Add the attributes from the method parameters.
        createTopicRequest.Attributes = new Dictionary<string, string>
        {
            { "FifoTopic", "true" }
        };
        if (useContentBasedDeduplication)
        {
            createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
        }
    }
}
```

```

        }
    }

    var createResponse = await
    _amazonSNSClient.CreateTopicAsync(createTopicRequest);
    return createResponse.TopicArn;
}

/// <summary>
/// Subscribe a queue to a topic with optional filters.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
/// <param name="queueArn">The ARN of the queue.</param>
/// <returns>The ARN of the new subscription.</returns>
public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
{
    var subscribeRequest = new SubscribeRequest()
    {
        TopicArn = topicArn,
        Protocol = "sqs",
        Endpoint = queueArn
    };

    if (!string.IsNullOrEmpty(filterPolicy))
    {
        subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
    }

    var subscribeResponse = await
    _amazonSNSClient.SubscribeAsync(subscribeRequest);
    return subscribeResponse.SubscriptionArn;
}

/// <summary>
/// Publish a message to a topic with an attribute and optional deduplication
and group IDs.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <param name="message">The message to publish.</param>

```



```
/// <param name="attributeName">The optional attribute for the message.</param>
/// <param name="attributeValue">The optional attribute value for the message.</param>
/// <param name="deduplicationId">The optional deduplication ID for the message.</param>
/// <param name="groupId">The optional group ID for the message.</param>
/// <returns>The ID of the message published.</returns>
public async Task<string> PublishToTopicWithAttribute(
    string topicArn,
    string message,
    string? attributeName = null,
    string? attributeValue = null,
    string? deduplicationId = null,
    string? groupId = null)
{
    var publishRequest = new PublishRequest()
    {
        TopicArn = topicArn,
        Message = message,
        MessageDeduplicationId = deduplicationId,
        MessageGroupId = groupId
    };

    if (attributeValue != null)
    {
        // Add the string attribute if it exists.
        publishRequest.MessageAttributes =
            new Dictionary<string, MessageAttributeValue>
            {
                { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String" } }
            };
    }

    var publishResponse = await
_amazonSNSClient.PublishAsync(publishRequest);
    return publishResponse.MessageId;
}

/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
```

```
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicar](#)
 - [ReceiveMessage](#)

- [SetQueueAttributes](#)
- [Assinar](#)
- [Cancelar assinatura](#)

C++

SDK para C++

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
SQS.
/*!
\param clientConfig Aws client configuration.
\return bool: Successful completion.
*/
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
        << NUMBER_OF_QUEUES <<
        " SQS queues to the topic." << std::endl;
    std::cout
        << "You can select from several options for configuring the topic and
the subscriptions for the "
        << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
queues."
        << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);
```

```
printAsterisksLine();

std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
          << std::endl;
std::cout
    << "FIFO topics deliver messages in order and support deduplication
and message filtering."
    << std::endl;
bool isFifoTopic = askYesNoQuestion(
    "Would you like to work with FIFO topics? (y/n) ");

bool contentBasedDeduplication = false;
Aws::String topicName;
if (isFifoTopic) {
    printAsterisksLine();
    std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
              << std::endl;
    std::cout
        << "Deduplication IDs are either set in the message or
automatically generated "
        << "from content using a hash function." << std::endl;
    std::cout
        << "If a message is successfully published to an SNS FIFO topic,
any message "
        << "published and determined to have the same deduplication ID, "
        << std::endl;
    std::cout
        << "within the five-minute deduplication interval, is accepted
but not delivered."
        << std::endl;
    std::cout
        << "For more information about deduplication, "
        << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html."
        << std::endl;
    contentBasedDeduplication = askYesNoQuestion(
        "Use content-based deduplication instead of entering a
deduplication ID? (y/n) ");
}

printAsterisksLine();

Aws::SQS::SQSClient sqsClient(clientConfiguration);
```

```
Aws::Vector<Aws::String> queueURLS;
Aws::Vector<Aws::String> subscriptionARNS;

Aws::String topicARN;
{
    topicName = askQuestion("Enter a name for your SNS topic. ");

    // 1. Create an Amazon SNS topic, either FIFO or non-FIFO.
    Aws::SNS::Model::CreateTopicRequest request;

    if (isFifoTopic) {
        request.AddAttributes("FifoTopic", "true");
        if (contentBasedDeduplication) {
            request.AddAttributes("ContentBasedDeduplication", "true");
        }
        topicName = topicName + FIFO_SUFFIX;

        std::cout
            << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
            << std::endl;
    }

    request.SetName(topicName);

    Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARN = outcome.GetResult().GetTopicArn();
        std::cout << "Your new topic with the name '" << topicName
            << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
        std::cout << "'" << topicARN << "' has been created." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::CreateTopic. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
```

```

        snsClient,
        sqsClient);

    return false;
}
}

printAsterisksLine();

std::cout << "Now you will create " << NUMBER_OF_QUEUES
           << " SQS queues to subscribe to the topic." << std::endl;
Aws::Vector<Aws::String> queueNames;
bool filteringMessages = false;
bool first = true;
for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
    Aws::String queueURL;
    Aws::String queueName;
    {
        printAsterisksLine();
        std::ostringstream ostream;
        ostream << "Enter a name for " << (first ? "an" : "the next")
                << " SQS queue. ";
        queueName = askQuestion(ostream.str());

        // 2. Create an SQS queue.
        Aws::SQS::Model::CreateQueueRequest request;
        if (isFifoTopic) {
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                "true");
            queueName = queueName + FIFO_SUFFIX;

            if (first) // Only explain this once.
            {
                std::cout
                    << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                    << "be appended to the queue name." << std::endl;
            }
        }

        request.SetQueueName(queueName);
        queueNames.push_back(queueName);
    }
}

```

```

        Aws::SQS::Model::CreateQueueOutcome outcome =
            sqsClient.CreateQueue(request);

        if (outcome.IsSuccess()) {
            queueURL = outcome.GetResult().GetQueueUrl();
            std::cout << "Your new SQS queue with the name '" << queueName
                        << "' and the queue URL " << std::endl;
            std::cout << "'" << queueURL << "' has been created." <<
std::endl;
        }
        else {
            std::cerr << "Error with SQS::CreateQueue. "
                        << outcome.GetError().GetMessage()
                        << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }
    queueURLS.push_back(queueURL);

    if (first) // Only explain this once.
    {
        std::cout
            << "The queue URL is used to retrieve the queue ARN, which is
"
            << "used to create a subscription." << std::endl;
    }

    Aws::String queueARN;
    {
        // 3. Get the SQS queue ARN attribute.
        Aws::SQS::Model::GetQueueAttributesRequest request;
        request.SetQueueUrl(queueURL);

        request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

        Aws::SQS::Model::GetQueueAttributesOutcome outcome =
            sqsClient.GetQueueAttributes(request);
    }

```

```

        if (outcome.IsSuccess()) {
            const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                outcome.GetResult().GetAttributes();
            const auto &iter = attributes.find(
                Aws::SQS::Model::QueueAttributeName::QueueArn);
            if (iter != attributes.end()) {
                queueARN = iter->second;
                std::cout << "The queue ARN '" << queueARN
                    << "' has been retrieved."
                    << std::endl;
            }
            else {
                std::cerr
                    << "Error ARN attribute not returned by
GetQueueAttribute."
                    << std::endl;

                cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

                return false;
            }
        }
        else {
            std::cerr << "Error with SQS::GetQueueAttributes. "
                << outcome.GetError().GetMessage()
                << std::endl;

            cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

            return false;
        }
    }

    if (first) {

```



```
        std::cout
            << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
            "messages from an SNS topic." << std::endl;
    }

    {
        // 4. Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
        Aws::SQS::Model::SetQueueAttributesRequest request;
        request.SetQueueUrl(queueURL);
        Aws::String policy = createPolicyForQueue(queueARN, topicARN);
        request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                             policy);

        Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);

        if (outcome.IsSuccess()) {
            std::cout << "The attributes for the queue '" << queueName
                << "' were successfully updated." << std::endl;
        }
        else {
            std::cerr << "Error with SQS::SetQueueAttributes. "
                << outcome.GetError().GetMessage()
                << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    {
        // 5. Subscribe the SQS queue to the SNS topic.
        Aws::SNS::Model::SubscribeRequest request;
        request.SetTopicArn(topicARN);
        request.SetProtocol("sqs");
```

```

        request.SetEndpoint(queueARN);
        if (isFifoTopic) {
            if (first) {
                std::cout << "Subscriptions to a FIFO topic can have
filters."
                        << std::endl;
                std::cout
                    << "If you add a filter to this subscription, then
only the filtered messages "
                    << "will be received in the queue." << std::endl;
                std::cout << "For information about message filtering, "
                    << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                    << std::endl;
                std::cout << "For this example, you can filter messages by a
\""
                        << TONE_ATTRIBUTE << "\" attribute." << std::endl;
            }

            std::ostringstream ostream;
            ostream << "Filter messages for \"" << queueName
                << "\"'s subscription to the topic \""
                << topicName << "\"? (y/n)";

            // Add filter if user answers yes.
            if (askYesNoQuestion(ostream.str())) {
                Aws::String jsonPolicy = getFilterPolicyFromUser();
                if (!jsonPolicy.empty()) {
                    filteringMessages = true;

                    std::cout << "This is the filter policy for this
subscription."
                            << std::endl;
                    std::cout << jsonPolicy << std::endl;

                    request.AddAttributes("FilterPolicy", jsonPolicy);
                }
                else {
                    std::cout
                        << "Because you did not select any attributes, no
filter "
                        << "will be added to this subscription." <<
std::endl;
                }
            }
        }
    }
}

```

```

        }
    } // if (isFifoTopic)
    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
                    << "' has been subscribed to the topic '"
                    << "'" << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN <<
". "

                    << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                    << outcome.GetError().GetMessage()
                    << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

first = false;
}

first = true;
do {
    printAsterisksLine();

    // 6. Publish a message to the SNS topic.
    Aws::SNS::Model::PublishRequest request;
    request.SetTopicArn(topicARN);
    Aws::String message = askQuestion("Enter a message text to publish. ");
    request.SetMessage(message);
    if (isFifoTopic) {

```

```

        if (first) {
            std::cout
                << "Because you are using a FIFO topic, you must set a
message group ID."
                << std::endl;
            std::cout
                << "All messages within the same group will be received
in the "
                << "order they were published." << std::endl;
        }
        Aws::String messageGroupId = askQuestion(
            "Enter a message group ID for this message. ");
        request.SetMessageGroupId(messageGroupId);
        if (!contentBasedDeduplication) {
            if (first) {
                std::cout
                    << "Because you are not using content-based
deduplication, "
                    << "you must enter a deduplication ID." << std::endl;
            }
            Aws::String deduplicationID = askQuestion(
                "Enter a deduplication ID for this message. ");
            request.SetMessageDeduplicationId(deduplicationID);
        }
    }

    if (filteringMessages && askYesNoQuestion(
        "Add an attribute to this message? (y/n) ")) {
        for (size_t i = 0; i < TONES.size(); ++i) {
            std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
        }
        int selection = askQuestionForIntRange(
            "Enter a number for an attribute. ",
            1, static_cast<int>(TONES.size()));
        Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
        messageAttributeValue.SetDataType("String");
        messageAttributeValue.SetStringValue(TONES[selection - 1]);
        request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
    }

    Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Your message was successfully published." << std::endl;
    }

```

```

    }
    else {
        std::cerr << "Error with TopicsAndQueues::Publish. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }

    first = false;
} while (askYesNoQuestion("Post another message? (y/n) "));

printAsterisksLine();

std::cout << "Now the SQS queue will be polled to retrieve the messages."
          << std::endl;
askQuestion("Press any key to continue...", alwaysTrueTest);

for (size_t i = 0; i < queueURLS.size(); ++i) {
    // 7. Poll an SQS queue for its messages.
    std::vector<Aws::String> messages;
    std::vector<Aws::String> receiptHandles;
    while (true) {
        Aws::SQS::Model::ReceiveMessageRequest request;
        request.SetMaxNumberOfMessages(10);
        request.SetQueueUrl(queueURLS[i]);

        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        request.SetWaitTimeSeconds(1);
        Aws::SQS::Model::ReceiveMessageOutcome outcome =
            sqsClient.ReceiveMessage(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();

```

```

        if (newMessages.empty()) {
            break;
        }
        else {
            for (const Aws::SQS::Model::Message &message: newMessages) {
                messages.push_back(message.GetBody());
                receiptHandles.push_back(message.GetReceiptHandle());
            }
        }
    }
    else {
        std::cerr << "Error with SQS::ReceiveMessage. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

printAsterisksLine();

if (messages.empty()) {
    std::cout << "No messages were ";
}
else if (messages.size() == 1) {
    std::cout << "One message was ";
}
else {
    std::cout << messages.size() << " messages were ";
}
std::cout << "received by the queue '" << queueNames[i]
          << "'." << std::endl;
for (const Aws::String &message: messages) {
    std::cout << "  Message : '" << message << "'."
              << std::endl;
}

// 8. Delete a batch of messages from an SQS queue.

```

```

        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                    << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
                cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNs,
                    snsClient,
                    sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
        queueURLS,
        subscriptionARNs,
        snsClient,
        sqsClient,
        true); // askUser
}

bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,

```

```

const Aws::Vector<Aws::String> &queueURLS,
const Aws::Vector<Aws::String>
&subscriptionARNs,

const Aws::SNS::SNSClient &snsClient,
const Aws::SQS::SQSClient &sqsClient,
bool askUser) {

    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9. Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                    << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNs) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                    << "' was successful." << std::endl;
            }
        }
    }
}

```



```

        else {
            std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                << outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }
}

printAsterisksLine();
if (!topicARN.empty() && askUser &&
    askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

    // 11. Delete an SNS topic.
    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "The topic with ARN '" << topicARN
            << "' was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
from an SNS topic.
/*!
\sa createPolicyForQueue()
\param queueARN: The SQS queue Amazon Resource Name (ARN).
\param topicARN: The SNS topic ARN.
\return Aws::String: The policy as JSON.
*/

```

```

Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
&queueARN,
                                                    const Aws::String
&topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                "Resource": ")" << queueARN << R"(",
                "Condition": {
                    "ArnEquals": {
                        "aws:SourceArn": ")" << topicARN << R"("
                    }
                }
            }
        ]
    })";

    return policyStream.str();
}

```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicar](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Assinar](#)

- [Cancelar assinatura](#)

Go

SDK para Go V2

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Execute um cenário interativo em um prompt de comando.

```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"
    "strings"
    "topics_and_queues/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
    Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
so that
```

```
// they are simpler to read and understand.
type ScenarioRunner struct {
    questioner demotools.IQuestioner
    snsActor    *actions.SnsActions
    sqsActor    *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string,
    bool, bool) {
    log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
    standard.\n" +
        "FIFO topics deliver messages in order and support deduplication and message
    filtering.")
    isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
    topics? (y/n) ", "y")

    contentBasedDeduplication := false
    if isFifoTopic {
        log.Println(strings.Repeat("-", 88))
        log.Println("Because you have chosen a FIFO topic, deduplication is supported.
    \n" +
            "Deduplication IDs are either set in the message or are automatically
    generated\n" +
            "from content using a hash function. If a message is successfully published to
    \n" +
            "an SNS FIFO topic, any message published and determined to have the same\n" +
            "deduplication ID, within the five-minute deduplication interval, is accepted
    \n" +
            "but not delivered. For more information about deduplication, see:\n" +
            "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
        contentBasedDeduplication = runner.questioner.AskBool(
            "\nDo you want to use content-based deduplication instead of entering a
    deduplication ID? (y/n) ", "y")
    }
    log.Println(strings.Repeat("-", 88))

    topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
    if isFifoTopic {
        topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
        log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
    \n"+
            "the topic name.", FIFO_SUFFIX)
    }
}
```

```

    topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
        contentBasedDeduplication)
    if err != nil {
        panic(err)
    }
    log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
    \n"+
        "'%v' has been created.", topicName, topicArn)

    return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
    isFifoTopic bool) (string, string) {
    queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
    queue. ", ordinal))
    if isFifoTopic {
        queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
        if ordinal == "first" {
            log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
                "be appended to the queue name.\n", FIFO_SUFFIX)
        }
    }
    queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
    if err != nil {
        panic(err)
    }
    log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
        "'%v' has been created.", queueName, queueUrl)

    return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
    ctx context.Context, queueName string, queueUrl string, topicName string,
    topicArn string, ordinal string,
    isFifoTopic bool) (string, bool) {

    queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
    if err != nil {
        panic(err)
    }
    log.Printf("The ARN of your queue is: %v.\n", queueArn)

```

```

err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
if err != nil {
    panic(err)
}
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
    "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
    if ordinal == "first" {
        log.Println("Subscriptions to a FIFO topic can have filters.\n" +
            "If you add a filter to this subscription, then only the filtered messages\n"
+
            "will be received in the queue.\n" +
            "For information about message filtering, see\n" +
            "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
            "For this example, you can filter messages by a \"tone\" attribute.")
    }

    wantFiltering := runner.questioner.AskBool(
        fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
            "from the %v topic? (y/n) ", queueName, topicName), "y")
    if wantFiltering {
        log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

        var toneSelections []string
        askAboutTones := true
        for askAboutTones {
            toneIndex := runner.questioner.AskChoice(
                "Enter the number of the tone you want to filter by:\n", ToneChoices)
            toneSelections = append(toneSelections, ToneChoices[toneIndex])
            askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
        }
        log.Printf("Your subscription will be filtered to only pass the following
tones: %v\n", toneSelections)
        filterPolicy = map[string][]string{TONE_KEY: toneSelections}
    }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)

```

```

    if err != nil {
        panic(err)
    }
    log.Printf("The queue %v is now subscribed to the topic %v with the subscription
    ARN %v.\n",
        queueName, topicName, subscriptionArn)

    return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn
    string, isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
    var message string
    var groupId string
    var dedupId string
    var toneSelection string
    publishMore := true
    for publishMore {
        groupId = ""
        dedupId = ""
        toneSelection = ""
        message = runner.questioner.Ask("Enter a message to publish: ")
        if isFifoTopic {
            log.Println("Because you are using a FIFO topic, you must set a message group
            ID.\n" +
                "All messages within the same group will be received in the order they were
            published.")
            groupId = runner.questioner.Ask("Enter a message group ID: ")
            if !contentBasedDeduplication {
                log.Println("Because you are not using content-based deduplication,\n" +
                    "you must enter a deduplication ID.")
                dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
            }
        }
    }
    if usingFilters {
        if runner.questioner.AskBool("Add a tone attribute so this message can be
        filtered? (y/n) ", "y") {
            toneIndex := runner.questioner.AskChoice(
                "Enter the number of the tone you want to filter by:\n", ToneChoices)
            toneSelection = ToneChoices[toneIndex]
        }
    }
}

```

```

    err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId,
TONE_KEY, toneSelection)
    if err != nil {
        panic(err)
    }
    log.Println(("Your message was published.))

    publishMore = runner.questioner.AskBool("Do you want to publish another
messsage? (y/n) ", "y")
}
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
[]string) {
    log.Println("Polling queues for messages...")
    for _, queueUrl := range queueUrls {
        var messages []types.Message
        for {
            currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
            if err != nil {
                panic(err)
            }
            if len(currentMsgs) == 0 {
                break
            }
            messages = append(messages, currentMsgs...)
        }
        if len(messages) == 0 {
            log.Printf("No messages were received by queue %v.\n", queueUrl)
        } else if len(messages) == 1 {
            log.Printf("One message was received by queue %v:\n", queueUrl)

        } else {
            log.Printf("%v messages were received by queue %v:\n", len(messages),
queueUrl)
        }
        for msgIndex, message := range messages {
            messageBody := MessageBody{}
            err := json.Unmarshal([]byte(*message.Body), &messageBody)
            if err != nil {
                panic(err)
            }
            log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
        }
    }
}

```



```

    if len(messages) > 0 {
        log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
        err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
        if err != nil {
            panic(err)
        }
    }
}
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
// the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
func RunTopicsAndQueuesScenario(
    ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
    resources := Resources{}
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.\n" +
                "Cleaning up any resources that were created...")
            resources.Cleanup(ctx)
        }
    }()
    queueCount := 2

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome to messaging with topics and queues.\n\n"+
        "In this scenario, you will create an SNS topic and subscribe %v SQS queues to\n"+

```

```

"topic. You can select from several options for configuring the topic and the
\n"+
"subscriptions for the queues. You can then post to the topic and see the
results\n"+
"in the queues.\n", queueCount)

log.Println(strings.Repeat("-", 88))

runner := ScenarioRunner{
    questioner: questioner,
    snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
    sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
}
resources.snsActor = runner.snsActor
resources.sqsActor = runner.sqsActor

topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
runner.CreateTopic(ctx)
resources.topicArn = topicArn
log.Println(strings.Repeat("-", 88))

log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
ordinals := []string{"first", "next"}
usingFilters := false
for _, ordinal := range ordinals {
    queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
    resources.queueUrls = append(resources.queueUrls, queueUrl)

    _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
topicName, topicArn, ordinal, isFifoTopic)
    usingFilters = usingFilters || filtering
}

log.Println(strings.Repeat("-", 88))
runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
usingFilters)
log.Println(strings.Repeat("-", 88))
runner.PollForMessages(ctx, resources.queueUrls)

log.Println(strings.Repeat("-", 88))

wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
created for this scenario? (y/n) ", "y")

```

```
if wantCleanup {
    log.Println("Cleaning up resources...")
    resources.Cleanup(ctx)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Defina um struct que envolva as ações do Amazon SNS usadas neste exemplo.

```
import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type SnsActions struct {
    SnsClient *sns.Client
}

// CreateTopic creates an Amazon SNS topic with the specified name. You can
// optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
// based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
    isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
    var topicArn string
    topicAttributes := map[string]string{}
```

```
    if isFifoTopic {
        topicAttributes["FifoTopic"] = "true"
    }
    if contentBasedDeduplication {
        topicAttributes["ContentBasedDeduplication"] = "true"
    }
    topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
        Name:      aws.String(topicName),
        Attributes: topicAttributes,
    })
    if err != nil {
        log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
    } else {
        topicArn = *topic.TopicArn
    }

    return topicArn, err
}

// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
    _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
        TopicArn: aws.String(topicArn)})
    if err != nil {
        log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
    }
    return err
}

// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
// an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
// policy
// so that messages are only sent to the queue when the message has the specified
// attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
    queueArn string, filterMap map[string][]string) (string, error) {
    var subscriptionArn string
    var attributes map[string]string
    if filterMap != nil {
```

```

    filterBytes, err := json.Marshal(filterMap)
    if err != nil {
        log.Printf("Couldn't create filter policy, here's why: %v\n", err)
        return "", err
    }
    attributes = map[string]string{"FilterPolicy": string(filterBytes)}
}
output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
    Protocol:          aws.String("sqs"),
    TopicArn:          aws.String(topicArn),
    Attributes:        attributes,
    Endpoint:          aws.String(queueArn),
    ReturnSubscription: true,
})
if err != nil {
    log.Printf("Couldn't subscribe queue %v to topic %v. Here's why: %v\n",
        queueArn, topicArn, err)
} else {
    subscriptionArn = *output.SubscriptionArn
}

return subscriptionArn, err
}

// Publish publishes a message to an Amazon SNS topic. The message is then sent
// to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
// group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
// value
// filter attribute can be specified so that the message can be filtered
// according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
    string, groupId string, dedupId string, filterKey string, filterValue string)
    error {
    publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
    aws.String(message)}
    if groupId != "" {
        publishInput.MessageGroupId = aws.String(groupId)
    }
    if dedupId != "" {

```

```

    publishInput.MessageDeduplicationId = aws.String(dedupId)
}
if filterKey != "" && filterValue != "" {
    publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
        filterKey: {DataType: aws.String("String"), StringValue:
aws.String(filterValue)},
    }
}
_, err := actor.SnsClient.Publish(ctx, &publishInput)
if err != nil {
    log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
err)
}
return err
}

```

Defina um struct que envolva as ações do Amazon SQS usadas neste exemplo.

```

import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// CreateQueue creates an Amazon SQS queue with the specified name. You can
// specify
// whether the queue is created as a FIFO queue.

```

```
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
    isFifoQueue bool) (string, error) {
    var queueUrl string
    queueAttributes := map[string]string{}
    if isFifoQueue {
        queueAttributes["FifoQueue"] = "true"
    }
    queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
        QueueName:  aws.String(queueName),
        Attributes: queueAttributes,
    })
    if err != nil {
        log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
    } else {
        queueUrl = *queue.QueueUrl
    }

    return queueUrl, err
}

// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
// (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
    (string, error) {
    var queueArn string
    arnAttributeName := types.QueueAttributeNameQueueArn
    attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
        &sqs.GetQueueAttributesInput{
            QueueUrl:      aws.String(queueUrl),
            AttributeNames: []types.QueueAttributeName{arnAttributeName},
        })
    if err != nil {
        log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
    } else {
        queueArn = attribute.Attributes[string(arnAttributeName)]
    }
    return queueArn, err
}
```

```
// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
// to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
// to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
string, queueArn string, topicArn string) error {
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect:    "Allow",
            Action:   "sqs:SendMessage",
            Principal: map[string]string{"Service": "sns.amazonaws.com"},
            Resource:  aws.String(queueArn),
            Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
topicArn}},
        }},
    }
    policyBytes, err := json.Marshal(policyDoc)
    if err != nil {
        log.Printf("Couldn't create policy document. Here's why: %v\n", err)
        return err
    }
    _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
        Attributes: map[string]string{
            string(types.QueueAttributeNamePolicy): string(policyBytes),
        },
        QueueUrl: aws.String(queueUrl),
    })
    if err != nil {
        log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
queueUrl, err)
    }
    return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
    Version    string
    Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
```



```
type PolicyStatement struct {
    Effect    string
    Action    string
    Principal map[string]string `json:",omitempty"`
    Resource  *string             `json:",omitempty"`
    Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string

// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
// queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
    maxMessages int32, waitTime int32) ([]types.Message, error) {
    var messages []types.Message
    result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
        QueueUrl:      aws.String(queueUrl),
        MaxNumberOfMessages: maxMessages,
        WaitTimeSeconds: waitTime,
    })
    if err != nil {
        log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
            err)
    } else {
        messages = result.Messages
    }
    return messages, err
}

// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
// messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
    messages []types.Message) error {
    entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
    for msgIndex := range messages {
        entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
        entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
    }
}
```

```

_, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
    Entries:  entries,
    QueueUrl: aws.String(queueUrl),
})
if err != nil {
    log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
        queueUrl, err)
}
return err
}

// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
    _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
        QueueUrl: aws.String(queueUrl)})
    if err != nil {
        log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
    }
    return err
}

```

Limpe recursos.

```

import (
    "context"
    "fmt"
    "log"
    "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
    topicArn  string
    queueUrls []string
    snsActor  *actions.SnsActions
    sqsActor  *actions.SqsActions
}

```

```
// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Println("Something went wrong during cleanup. Use the AWS Management
            Console\n" +
                "to remove any remaining resources that were created for this scenario.")
        }
    }()

    var err error
    if resources.topicArn != "" {
        log.Printf("Deleting topic %v.\n", resources.topicArn)
        err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
        if err != nil {
            panic(err)
        }
    }

    for _, queueUrl := range resources.queueUrls {
        log.Printf("Deleting queue %v.\n", queueUrl)
        err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
        if err != nil {
            panic(err)
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicar](#)

- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Assinar](#)
- [Cancelar assinatura](#)

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
package com.example.sns;

import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
    software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
```

```
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
```

```
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
```

```
String msgAttValue = "";

System.out.println(DASHES);
System.out.println("Welcome to messaging with topics and queues.");
System.out.println("In this scenario, you will create an SNS topic and
subscribe an SQS queue to the topic.\n" +
    "You can select from several options for configuring the topic and
the subscriptions for the queue.\n" +
    "You can then post to the topic and see the results in the queue.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
    "FIFO topics deliver messages in order and support deduplication and
message filtering.\n" +
    "Would you like to work with FIFO topics? (y/n)");
useFIFO = in.nextLine();
if (useFIFO.compareTo("y") == 0) {
    selectFIFO = true;
    System.out.println("You have selected FIFO");
    System.out.println(" Because you have chosen a FIFO topic,
deduplication is supported.\n" +
        "          Deduplication IDs are either set in the message or
automatically generated from content using a hash function.\n"
        +
        "          If a message is successfully published to an SNS FIFO
topic, any message published and determined to have the same deduplication ID,
\n"
        +
        "          within the five-minute deduplication interval, is
accepted but not delivered.\n" +
        "          For more information about deduplication, see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html");

    System.out.println(
        "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
    duplication = in.nextLine();
    if (duplication.compareTo("y") == 0) {
        System.out.println("Please enter a group id value");
        groupId = in.nextLine();
    } else {
        System.out.println("Please enter deduplication Id value");
```

```
        deduplicationID = in.nextLine();
        System.out.println("Please enter a group id value");
        groupId = in.nextLine();
    }
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a topic.");
System.out.println("Enter a name for your SNS topic.");
topicName = in.nextLine();
if (selectFIFO) {
    System.out.println("Because you have selected a FIFO topic, '.fifo'
must be appended to the topic name.");
    topicName = topicName + ".fifo";
    System.out.println("The name of the topic is " + topicName);
    topicArn = createFIFO(snsClient, topicName, duplication);
    System.out.println("The ARN of the FIFO topic is " + topicArn);

} else {
    System.out.println("The name of the topic is " + topicName);
    topicArn = createSNSTopic(snsClient, topicName);
    System.out.println("The ARN of the non-FIFO topic is " + topicArn);

}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create an SQS queue.");
System.out.println("Enter a name for your SQS queue.");
sqsQueueName = in.nextLine();
if (selectFIFO) {
    sqsQueueName = sqsQueueName + ".fifo";
}
sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
System.out.println("The queue URL is " + sqsQueueUrl);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the SQS queue ARN attribute.");
sqsQueueArn = getSqsQueueAttrs(sqsClient, sqsQueueUrl);
System.out.println("The ARN of the new queue is " + sqsQueueArn);
System.out.println(DASHES);
```



```

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = ""
        {
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "sns.amazonaws.com"
                    },
                    "Action": "sqs:SendMessage",
                    "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                    "Condition": {
                        "ArnEquals": {
                            "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                        }
                    }
                }
            ]
        }
        """".formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
messages will be received in the queue.\n"
                +
                "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                +
                "For this example, you can filter messages by a \"tone\"
attribute.");
            System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic ")

```

```

        + topicName + "? (y/n)");
String filterAns = in.nextLine();
if (filterAns.compareTo("y") == 0) {
    boolean moreAns = false;
    System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
    System.out.println("1. cheerful");
    System.out.println("2. funny");
    System.out.println("3. serious");
    System.out.println("4. sincere");
    while (!moreAns) {
        System.out.println("Select a number or choose 0 to end.");
        String ans = in.nextLine();
        switch (ans) {
            case "1":
                filterList.add("cheerful");
                break;
            case "2":
                filterList.add("funny");
                break;
            case "3":
                filterList.add("serious");
                break;
            case "4":
                filterList.add("sincere");
                break;
            default:
                moreAns = true;
                break;
        }
    }
}

subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Publish a message to the topic.");
if (selectFIFO) {
    System.out.println("Would you like to add an attribute to this
message? (y/n)");
    String msgAns = in.nextLine();
    if (msgAns.compareTo("y") == 0) {

```

```
        System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
        System.out.println("1. cheerful");
        System.out.println("2. funny");
        System.out.println("3. serious");
        System.out.println("4. sincere");
        System.out.println("Select a number or choose 0 to end.");
        String ans = in.nextLine();
        switch (ans) {
            case "1":
                msgAttValue = "cheerful";
                break;
            case "2":
                msgAttValue = "funny";
                break;
            case "3":
                msgAttValue = "serious";
                break;
            default:
                msgAttValue = "sincere";
                break;
        }

        System.out.println("Selected value is " + msgAttValue);
    }
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

} else {
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessage(snsClient, message, topicArn);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Display the message. Press any key to continue.");
in.nextLine();
messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
for (Message mes : messageList) {
    System.out.println("Message Id: " + mes.messageId());
    System.out.println("Full Message: " + mes.body());
}
```

```

    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("9. Delete the received message. Press any key to
continue.");
    in.nextLine();
    deleteMessages(sqsClient, sqsQueueUrl, messageList);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
    in.nextLine();
    unSub(snsClient, subscriptionArn);
    deleteSQSQueue(sqsClient, sqsQueueName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("11. Delete the topic. Press any key to continue.");
    in.nextLine();
    deleteSNSTopic(snsClient, topicArn);

    System.out.println(DASHES);
    System.out.println("The SNS/SQS workflow has completed successfully.");
    System.out.println(DASHES);
}

public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);
        System.out.println(queueName + " was successfully deleted.");

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
            + "\nSubscription was removed for " + request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    try {
        List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
        for (Message msg : messages) {
            DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
```

```

        .id(msg.messageId())
        .build();

        entries.add(entry);
    }

    DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
        .queueUrl(queueUrl)
        .entries(entries)
        .build();

    sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
    System.out.println("The batch delete of messages was successful");

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
    try {
        if (msgAttValue.isEmpty()) {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .numberOfMessages(5)
                .build();
            return
sqsClient.receiveMessage(receiveMessageRequest).messages();
        } else {
            // We know there are filters on the message.
            ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                .numberOfMessages(5)
                .build();

            return sqsClient.receiveMessage(receiveRequest).messages();
        }
    }
}

```

```
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void pubMessageFIFO(SnsClient snsClient,
                                String message,
                                String topicArn,
                                String msgAttValue,
                                String duplication,
                                String groupId,
                                String deduplicationID) {

    try {
        PublishRequest request;
        // Means the user did not choose to use a message attribute.
        if (msgAttValue.isEmpty()) {
            if (duplication.compareTo("y") == 0) {
                request = PublishRequest.builder()
                    .message(message)
                    .messageGroupId(groupId)
                    .topicArn(topicArn)
```

```
        .build();
    } else {
        request = PublishRequest.builder()
            .message(message)
            .messageDeduplicationId(deduplicationID)
            .messageGroupId(groupId)
            .topicArn(topicArn)
            .build();
    }

    } else {
        Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
        messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
            .dataType("String")
            .stringValue("true")
            .build());

        if (duplication.compareTo("y") == 0) {
            request = PublishRequest.builder()
                .message(message)
                .messageGroupId(groupId)
                .topicArn(topicArn)
                .build();
        } else {
            // Create a publish request with the message and attributes.
            request = PublishRequest.builder()
                .topicArn(topicArn)
                .message(message)
                .messageDeduplicationId(deduplicationID)
                .messageGroupId(groupId)
                .messageAttributes(messageAttributes)
                .build();
        }
    }

    // Publish the message to the topic.
    PublishResponse result = snsClient.publish(request);
    System.out
        .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

} catch (SnsException e) {
```



```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Subscribe to the SQS queue.
public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
    try {
        SubscribeRequest request;
        if (filterList.isEmpty()) {
            // No filter subscription is added.
            request = SubscribeRequest.builder()
                .protocol("sqs")
                .endpoint(queueArn)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                "with the subscription ARN " + result.subscriptionArn());
            return result.subscriptionArn();
        } else {
            request = SubscribeRequest.builder()
                .protocol("sqs")
                .endpoint(queueArn)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                "with the subscription ARN " + result.subscriptionArn());

            String attributeName = "FilterPolicy";
            Gson gson = new Gson();
            String jsonString = "{\"tone\": []}";
            JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
            JsonArray toneArray = jsonObject.getAsJsonArray("tone");
            for (String value : filterList) {

```

```
        toneArray.add(new JsonPrimitive(value));
    }

    String updatedJsonString = gson.toJson(jsonObject);
    System.out.println(updatedJsonString);
    SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
        .subscriptionArn(result.subscriptionArn())
        .attributeName(attributeName)
        .attributeValue(updatedJsonString)
        .build();

    snsClient.setSubscriptionAttributes(attRequest);
    return result.subscriptionArn();
}

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

// Attach a policy to the queue.
public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
    try {
        Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
        attrMap.put(QueueAttributeName.POLICY, policy);

        SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributes(attrMap)
            .build();

        sqsClient.setQueueAttributes(attributesRequest);
        System.out.println("The policy has been successfully attached.");

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}

public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
    // Specify the attributes to retrieve.
    List<QueueAttributeName> atts = new ArrayList<>();
    atts.add(QueueAttributeName.QUEUE_ARN);

    GetQueueAttributesRequest attributesRequest =
    GetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)
        .attributeNames(atts)
        .build();

    GetQueueAttributesResponse response =
    sqsClient.getQueueAttributes(attributesRequest);
    Map<String, String> queueAttrs = response.attributesAsStrings();
    for (Map.Entry<String, String> queueAtt : queueAttrs.entrySet())
        return queueAtt.getValue();

    return "";
}

public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
    try {
        System.out.println("\nCreate Queue");
        if (selectFIFO) {
            Map<QueueAttributeName, String> attrs = new HashMap<>();
            attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
            CreateQueueRequest createQueueRequest =
            CreateQueueRequest.builder()
                .queueName(queueName)
                .attributes(attrs)
                .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

            .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
        } else {
            CreateQueueRequest createQueueRequest =
            CreateQueueRequest.builder()
```

```

        .queueName(queueName)
        .build();

        sqsClient.createQueue(createQueueRequest);
        System.out.println("\nGet queue url");
        GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();
    }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = new HashMap<>();
        if (duplication.compareTo("n") == 0) {
            topicAttributes.put("FifoTopic", "true");
            topicAttributes.put("ContentBasedDeduplication", "false");
        } else {

```

```
        topicAttributes.put("FifoTopic", "true");
        topicAttributes.put("ContentBasedDeduplication", "true");
    }

    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    return response.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicar](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Assinar](#)
 - [Cancelar assinatura](#)

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Esse é o ponto de entrada para esse cenário.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

O código anterior fornece as dependências necessárias e inicia o cenário. A próxima seção contém a maior parte do exemplo.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];
```

```
export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../libs/prompter.js').Prompter} prompter
   * @param {import('../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
    this.sqsClient = sqsClient;
    this.prompter = prompter;
    this.logger = logger;
  }

  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
    }
  }
}
```

```
    await this.logger.log(MESSAGES.deduplicationNotice);
    await this.logger.log(MESSAGES.deduplicationDescription);
    this.autoDedup = await this.prompter.confirm({
      message: MESSAGES.deduplicationPrompt,
    });
  }
}

async createTopic() {
  await this.logger.log(MESSAGES.creatingTopics);
  this.topicName = await this.prompter.input({
    message: MESSAGES.topicNamePrompt,
  });
  if (this.isFifo) {
    this.topicName += ".fifo";
    this.logger.logSeparator(MESSAGES.headerFifoNaming);
    await this.logger.log(MESSAGES.appendFifoNotice);
  }

  const response = await this.snsClient.send(
    new CreateTopicCommand({
      Name: this.topicName,
      Attributes: {
        FifoTopic: this.isFifo ? "true" : "false",
        ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
      },
    }),
  );

  this.topicArn = response.TopicArn;

  await this.logger.log(
    MESSAGES.topicCreatedNotice
      .replace("${TOPIC_NAME}", this.topicName)
      .replace("${TOPIC_ARN}", this.topicArn),
  );
}

async createQueues() {
  await this.logger.log(MESSAGES.createQueuesNotice);
  // Increase this number to add more queues.
  const maxQueues = 2;

  for (let i = 0; i < maxQueues; i++) {
```



```
    await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
    let queueName = await this.prompter.input({
      message: MESSAGES.queueNamePrompt.replace(
        "${EXAMPLE_NAME}",
        i === 0 ? "good-news" : "bad-news",
      ),
    });

    if (this.isFifo) {
      queueName += ".fifo";
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.sqsClient.send(
      new CreateQueueCommand({
        QueueName: queueName,
        Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
      }),
    );

    const { Attributes } = await this.sqsClient.send(
      new GetQueueAttributesCommand({
        QueueUrl: response.QueueUrl,
        AttributeNames: ["QueueArn"],
      }),
    );

    this.queues.push({
      queueName,
      queueArn: Attributes.QueueArn,
      queueUrl: response.QueueUrl,
    });

    await this.logger.log(
      MESSAGES.queueCreatedNotice
        .replace("${QUEUE_NAME}", queueName)
        .replace("${QUEUE_URL}", response.QueueUrl)
        .replace("${QUEUE_ARN}", Attributes.QueueArn),
    );
  }
}

async attachQueueIamPolicies() {
  for (const [index, queue] of this.queues.entries()) {
```

```
const policy = JSON.stringify(
  {
    Statement: [
      {
        Effect: "Allow",
        Principal: {
          Service: "sns.amazonaws.com",
        },
        Action: "sqs:SendMessage",
        Resource: queue.queueArn,
        Condition: {
          ArnEquals: {
            "aws:SourceArn": this.topicArn,
          },
        },
      },
    ],
  },
  null,
  2,
);

if (index !== 0) {
  this.logger.logSeparator();
}

await this.logger.log(MESSAGES.attachPolicyNotice);
console.log(policy);
const addPolicy = await this.prompter.confirm({
  message: MESSAGES.addPolicyConfirmation.replace(
    "${QUEUE_NAME}",
    queue.queueName,
  ),
});

if (addPolicy) {
  await this.sqsClient.send(
    new SetQueueAttributesCommand({
      QueueUrl: queue.queueUrl,
      Attributes: {
        Policy: policy,
      },
    }),
  );
};
```

```

        queue.policy = policy;
    } else {
        await this.logger.log(
            MESSAGES.policyNotAttachedNotice.replace(
                "${QUEUE_NAME}",
                queue.queueName,
            ),
        );
    }
}
}

async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
        /**
         * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
         */
        const subscribeParams = {
            TopicArn: this.topicArn,
            Protocol: "sqs",
            Endpoint: queue.queueArn,
        };
        let tones = [];

        if (this.isFifo) {
            if (index === 0) {
                await this.logger.log(MESSAGES.fifoFilterNotice);
            }
            tones = await this.prompter.checkbox({
                message: MESSAGES.fifoFilterSelect.replace(
                    "${QUEUE_NAME}",
                    queue.queueName,
                ),
                choices: toneChoices,
            });

            if (tones.length) {
                subscribeParams.Attributes = {
                    FilterPolicyScope: "MessageAttributes",
                    FilterPolicy: JSON.stringify({
                        tone: tones,
                    }),
                };
            }
        }
    }
}

```

```
    }

    const { SubscriptionArn } = await this.snsClient.send(
      new SubscribeCommand(subscribeParams),
    );

    this.subscriptionArns.push(SubscriptionArn);

    await this.logger.log(
      MESSAGES.queueSubscribedNotice
        .replace("${QUEUE_NAME}", queue.queueName)
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
    );
  }
}

async publishMessages() {
  const message = await this.prompter.input({
    message: MESSAGES.publishMessagePrompt,
  });

  let groupId;
  let deduplicationId;
  let choices;

  if (this.isFifo) {
    await this.logger.log(MESSAGES.groupIdNotice);
    groupId = await this.prompter.input({
      message: MESSAGES.groupIdPrompt,
    });

    if (this.autoDedup === false) {
      await this.logger.log(MESSAGES.deduplicationIdNotice);
      deduplicationId = await this.prompter.input({
        message: MESSAGES.deduplicationIdPrompt,
      });
    }

    choices = await this.prompter.checkbox({
      message: MESSAGES.messageAttributesPrompt,
      choices: toneChoices,
    });
  }
}
```

```
await this.snsClient.send(
  new PublishCommand({
    TopicArn: this.topicArn,
    Message: message,
    ...(groupId
      ? {
          MessageGroupId: groupId,
        }
      : {}),
    ...(deduplicationId
      ? {
          MessageDeduplicationId: deduplicationId,
        }
      : {}),
    ...(choices
      ? {
          MessageAttributes: {
            tone: {
              DataType: "String.Array",
              StringValue: JSON.stringify(choices),
            },
          },
        }
      : {}),
  )),
);

const publishAnother = await this.prompter.confirm({
  message: MESSAGES.publishAnother,
});

if (publishAnother) {
  await this.publishMessages();
}

}

async receiveAndDeleteMessages() {
  for (const queue of this.queues) {
    const { Messages } = await this.sqsClient.send(
      new ReceiveMessageCommand({
        QueueUrl: queue.queueUrl,
      }),
    );
  }
}
```

```
    if (Messages) {
      await this.logger.log(
        MESSAGES.messagesReceivedNotice.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      );
      console.log(Messages);

      await this.sqsClient.send(
        new DeleteMessageBatchCommand({
          QueueUrl: queue.queueUrl,
          Entries: Messages.map((message) => ({
            Id: message.MessageId,
            ReceiptHandle: message.ReceiptHandle,
          })),
        }),
      );
    } else {
      await this.logger.log(
        MESSAGES.noMessagesReceivedNotice.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      );
    }
  }

  const deleteAndPoll = await this.prompter.confirm({
    message: MESSAGES.deleteAndPollConfirmation,
  });

  if (deleteAndPoll) {
    await this.receiveAndDeleteMessages();
  }
}

async destroyResources() {
  for (const subscriptionArn of this.subscriptionArns) {
    await this.snsClient.send(
      new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
    );
  }
}
```

```
    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicar](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Assinar](#)
 - [Cancelar assinatura](#)

Kotlin

SDK para Kotlin

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
```



```
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner
```

```
/**
```

Before running this Kotlin code example, set up your development environment, including your AWS credentials.

For more information, see the following documentation topic:

<https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html>

This Kotlin example performs the following tasks:

1. Gives the user three options to choose from.
2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
6. Subscribes to the SQS queue.
7. Publishes a message to the topic.
8. Displays the messages.
9. Deletes the received message.
10. Unsubscribes from the topic.
11. Deletes the SNS topic.

```
*/
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
```

```

val sqsQueueUrl: String?
val sqsQueueArn: String
val subscriptionArn: String?
var selectFIFO = false
val message: String
val messageList: List<Message?>?
val filterList = ArrayList<String>()
var msgAttValue = ""

println(DASHES)
println("Welcome to the AWS SDK for Kotlin messaging with topics and
queues.")
println(
    """
        In this scenario, you will create an SNS topic and subscribe an
        SQS queue to the topic.
        You can select from several options for configuring the topic and
        the subscriptions for the queue.
        You can then post to the topic and see the results in the queue.
    """.trimIndent(),
)
println(DASHES)

println(DASHES)
println(
    """
        SNS topics can be configured as FIFO (First-In-First-Out).
        FIFO topics deliver messages in order and support deduplication
        and message filtering.
        Would you like to work with FIFO topics? (y/n)
    """.trimIndent(),
)
useFIFO = input.nextLine()
if (useFIFO.compareTo("y") == 0) {
    selectFIFO = true
    println("You have selected FIFO")
    println(
        """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
        generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
        published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
        delivered.
    """
    )
}

```

```

        For more information about deduplication, see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
    )

    println("Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)")
    duplication = input.nextLine()
    if (duplication.compareTo("y") == 0) {
        println("Enter a group id value")
        groupId = input.nextLine()
    } else {
        println("Enter deduplication Id value")
        deduplicationID = input.nextLine()
        println("Enter a group id value")
        groupId = input.nextLine()
    }
}
println(DASHES)

println(DASHES)
println("2. Create a topic.")
println("Enter a name for your SNS topic.")
topicName = input.nextLine()
if (selectFIFO) {
    println("Because you have selected a FIFO topic, '.fifo' must be appended
to the topic name.")
    topicName = "$topicName.fifo"
    println("The name of the topic is $topicName")
    topicArn = createFIFO(topicName, duplication)
    println("The ARN of the FIFO topic is $topicArn")
} else {
    println("The name of the topic is $topicName")
    topicArn = createSNSTopic(topicName)
    println("The ARN of the non-FIFO topic is $topicArn")
}
println(DASHES)

println(DASHES)
println("3. Create an SQS queue.")
println("Enter a name for your SQS queue.")
sqsQueueName = input.nextLine()
if (selectFIFO) {
    sqsQueueName = "$sqsQueueName.fifo"
}

```

```
sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
println("The queue URL is $sqsQueueUrl")
println(DASHES)

println(DASHES)
println("4. Get the SQS queue ARN attribute.")
sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
println("The ARN of the new queue is $sqsQueueArn")
println(DASHES)

println(DASHES)
println("5. Attach an IAM policy to the queue.")
// Define the policy to use.
val policy = """{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "$sqsQueueArn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicArn"
        }
      }
    }
  ]
}"""
setQueueAttr(sqsQueueUrl, policy)
println(DASHES)

println(DASHES)
println("6. Subscribe to the SQS queue.")
if (selectFIFO) {
  println(
    """If you add a filter to this subscription, then only the filtered
    messages will be received in the queue.
    For information about message filtering, see https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html
    For this example, you can filter messages by a "tone" attribute."""
  )
}
```

```

        println("Would you like to filter messages for $sqsQueueName's
subscription to the topic $topicName? (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the
following \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
}

subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
println(DASHES)

println(DASHES)
println("7. Publish a message to the topic.")
if (selectFIFO) {
    println("Would you like to add an attribute to this message? (y/n)")
    val msgAns: String = input.nextLine()
    if (msgAns.compareTo("y") == 0) {
        println("You can filter messages by one or more of the following
\"tone\" attributes.")
        println("1. cheerful")
        println("2. funny")
        println("3. serious")
        println("4. sincere")
        println("Select a number or choose 0 to end.")
        val ans: String = input.nextLine()
        msgAttValue = when (ans) {
            "1" -> "cheerful"
            "2" -> "funny"

```

```
        "3" -> "serious"
        else -> "sincere"
    }
    println("Selected value is $msgAttValue")
}
println("Enter a message.")
message = input.nextLine()
pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
} else {
    println("Enter a message.")
    message = input.nextLine()
    pubMessage(message, topicArn)
}
println(DASHES)

println(DASHES)
println("8. Display the message. Press any key to continue.")
input.nextLine()
messageList = receiveMessages(sqsQueueUrl, msgAttValue)
if (messageList != null) {
    for (mes in messageList) {
        println("Message Id: ${mes.messageId}")
        println("Full Message: ${mes.body}")
    }
}
println(DASHES)

println(DASHES)
println("9. Delete the received message. Press any key to continue.")
input.nextLine()
if (messageList != null) {
    deleteMessages(sqsQueueUrl, messageList)
}
println(DASHES)

println(DASHES)
println("10. Unsubscribe from the topic and delete the queue. Press any key
to continue.")
input.nextLine()
unSub(subscriptionArn)
deleteSQSQueue(sqsQueueName)
println(DASHES)
```

```
println(DASHES)
println("11. Delete the topic. Press any key to continue.")
input.nextLine()
deleteSNSTopic(topicArn)
println(DASHES)

println(DASHES)
println("The SNS/SQS workflow has completed successfully.")
println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
    }
}
```

```
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}
```



```
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
```

```

        val result = snsClient.publish(request)
        println(result.messageId.toString() + " Message sent.")
    }
}
} else {
    val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
        dataType = "String"
        stringValue = "true"
    }

    val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
        mapOf(msgAttValue to messAttr)
    if (duplication.compareTo("y") == 0) {
        val request = PublishRequest {
            message = messageVal
            messageGroupId = groupIdVal
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.publish(request)
            println(result.messageId.toString() + " Message sent.")
        }
    } else {
        // Create a publish request with the message and attributes.
        val request = PublishRequest {
            topicArn = topicArnVal
            message = messageVal
            messageDeduplicationId = deduplicationID
            messageGroupId = groupIdVal
            messageAttributes = mapAtt
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.publish(request)
            println(result.messageId.toString() + " Message sent.")
        }
    }
}
}

// Subscribe to the SQS queue.

```

```

suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
+ topicArnVal + "\n" +
                "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
$topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val json0bject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = json0bject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(json0bject)

```

```

        println(updatedJsonString)
        val attRequest = SetSubscriptionAttributesRequest {
            subscriptionArn = result.subscriptionArn
            attributeName = attributeNameVal
            attributeValue = updatedJsonString
        }

        snsClient.setSubscriptionAttributes(attRequest)
        return result.subscriptionArn
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
}

```

```
    }  
  }  
  return ""  
}  
  
suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {  
  println("\nCreate Queue")  
  if (selectFIFO) {  
    val attrs = mutableMapOf<String, String>()  
    attrs[QueueAttributeName.FifoQueue.toString()] = "true"  
  
    val createQueueRequest = CreateQueueRequest {  
      queueName = queueNameVal  
      attributes = attrs  
    }  
  
    SqsClient { region = "us-east-1" }.use { sqsClient ->  
      sqsClient.createQueue(createQueueRequest)  
      println("\nGet queue url")  
  
      val urlRequest = GetQueueUrlRequest {  
        queueName = queueNameVal  
      }  
  
      val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)  
      return getQueueUrlResponse.queueUrl  
    }  
  } else {  
    val createQueueRequest = CreateQueueRequest {  
      queueName = queueNameVal  
    }  
  
    SqsClient { region = "us-east-1" }.use { sqsClient ->  
      sqsClient.createQueue(createQueueRequest)  
      println("Get queue url")  
  
      val urlRequest = GetQueueUrlRequest {  
        queueName = queueNameVal  
      }  
  
      val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)  
      return getQueueUrlResponse.queueUrl  
    }  
  }  
}
```

```
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)

- [GetQueueAttributes](#)
- [Publicar](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Assinar](#)
- [Cancelar assinatura](#)

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Execute um cenário interativo em um prompt de comando.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
```

```
self.queue_count = 2
self.queue_urls = []
self.subscription_arns = []
self.tones = ["cheerful", "funny", "serious", "sincere"]

def run_scenario(self) -> None:
    """Run the Topics and Queues feature scenario."""
    print(self.DASHES)
    print("Welcome to messaging with topics and queues.")
    print(self.DASHES)
    print(f"""
    In this scenario, you will create an SNS topic and subscribe
    {self.queue_count} SQS queues to the topic.
    You can select from several options for configuring the topic and the
    subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
    """)

    try:
        # Setup Phase
        print(self.DASHES)
        self._setup_topic()
        print(self.DASHES)

        self._setup_queues()
        print(self.DASHES)

        # Demonstration Phase
        self._publish_messages()
        print(self.DASHES)

        # Examination Phase
        self._poll_queues_for_messages()
        print(self.DASHES)

        # Cleanup Phase
        self._cleanup_resources()
        print(self.DASHES)

    except Exception as e:
        logger.error(f"Scenario failed: {e}")
        print(f"There was a problem with the scenario: {e}")
        print("\nInitiating cleanup...")
        try:
```



```

        self._cleanup_resources()
    except Exception as cleanup_error:
        logger.error(f"Error during cleanup: {cleanup_error}")

    print("Messaging with topics and queues scenario is complete.")
    print(self.DASHES)

def _setup_topic(self) -> None:
    """Set up the SNS topic to be used with the queues."""
    print("SNS topics can be configured as FIFO (First-In-First-Out).")
    print("FIFO topics deliver messages in order and support deduplication
and message filtering.")
    print()

    self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/
n): ", q.is_yesno)

    if self.use_fifo_topic:
        print(self.DASHES)
        self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)
        print("Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name.")
        print()

        print(self.DASHES)
        print("""
Because you have chosen a FIFO topic, deduplication is supported.
Deduplication IDs are either set in the message or automatically generated
from content using a hash function.

If a message is successfully published to an SNS FIFO topic, any message
published and determined to have the same deduplication ID,
within the five-minute deduplication interval, is accepted but not delivered.

For more information about deduplication,
see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
""")

        self.use_content_based_deduplication = q.ask(
            "Use content-based deduplication instead of entering a
deduplication ID? (y/n): ",
            q.is_yesno
        )

```

```
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)

            print(self.DASHES)

            # Create the topic
            self.topic_arn = self.sns_wrapper.create_topic(
                self.topic_name,
                self.use_fifo_topic,
                self.use_content_based_deduplication
            )

            print(f"Your new topic with the name {self.topic_name}")
            print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
            print(f"  has been created.")
            print()

        def _setup_queues(self) -> None:
            """Set up the SQS queues and subscribe them to the topic."""
            print(f"Now you will create {self.queue_count} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.")

            for i in range(self.queue_count):
                queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ",
q.non_empty)

                if self.use_fifo_topic and i == 0:
                    print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

                # Create the queue
                queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
                self.queue_urls.append(queue_url)

                print(f"Your new queue with the name {queue_name}")
                print(f"  and queue URL {queue_url}")
                print(f"  has been created.")
                print()

                if i == 0:
                    print("The queue URL is used to retrieve the queue ARN,")
                    print("which is used to create a subscription.")
```

```
        print(self.DASHES)

        # Get queue ARN
        queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

        if i == 0:
            print("An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue,")
            print("enabling it to receive messages from an SNS topic.")

        # Set queue policy to allow SNS to send messages
        self.sqs_wrapper.set_queue_policy_for_topic(queue_arn,
self.topic_arn, queue_url)

        # Set up message filtering if using FIFO
        subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
queue_name)
        self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
                print()
                print("For this example, you can filter messages by a TONE
attribute.")

            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription
to the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()
```

```

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
{self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

def _create_filter_policy(self) -> str:
    """Create a message filter policy based on user selections."""
    print(self.DASHES)
    print("You can filter messages by one or more of the following TONE
attributes.")

    filter_selections = []
    selection_number = 0

    while True:
        print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
        for i, tone in enumerate(self.tones, 1):
            print(f"  {i}. {tone}")

        selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

        if selection == 0:
            break
        elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
            filter_selections.append(self.tones[selection - 1])
            print(f"Added '{self.tones[selection - 1]}' to filter list.")

    if filter_selections:
        filters = {"tone": filter_selections}
        return json.dumps(filters)
    return None

def _publish_messages(self) -> None:
    """Publish messages to the topic with various options."""
    print("Now we can publish messages.")

```

```
keep_sending = True
while keep_sending:
    print()
    message = q.ask("Enter a message to publish: ", q.non_empty)

    message_group_id = None
    deduplication_id = None
    tone_attribute = None

    if self.use_fifo_topic:
        print("Because you are using a FIFO topic, you must set a message
group ID.")
        print("All messages within the same group will be received in the
order they were published.")
        print()
        message_group_id = q.ask("Enter a message group ID for this
message: ", q.non_empty)

        if not self.use_content_based_deduplication:
            print("Because you are not using content-based
deduplication,")
            print("you must enter a deduplication ID.")
            deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

        # Ask about tone attribute
        add_attribute = q.ask("Add an attribute to this message? (y/n):
", q.is_yn)
        if add_attribute:
            print("Enter a number for an attribute:")
            for i, tone in enumerate(self.tones, 1):
                print(f" {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
len(self.tones)))
            if 1 <= selection <= len(self.tones):
                tone_attribute = self.tones[selection - 1]

    # Publish the message
    message_id = self.sns_wrapper.publish_message(
        self.topic_arn,
        message,
        tone_attribute,
```

```

        deduplication_id,
        message_group_id
    )

    print(f"Message published with ID: {message_id}")

    keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

def _poll_queues_for_messages(self) -> None:
    """Poll all queues for messages and display results."""
    for i, queue_url in enumerate(self.queue_urls):
        print(f"Polling queue #{i+1} at {queue_url} for messages...")

        q.ask("Press Enter to continue...")

        messages = self._poll_queue_for_messages(queue_url)

        if messages:
            print(f"{len(messages)} message(s) were received by queue #{i+1}")

            for j, message in enumerate(messages, 1):
                print(f"  Message {j}:")
                # Parse the SNS message body to get the actual message
                try:
                    sns_message = json.loads(message['Body'])
                    actual_message = sns_message.get('Message',
message['Body'])

                    print(f"    {actual_message}")
                except (json.JSONDecodeError, KeyError):
                    print(f"    {message['Body']}")

            # Delete the messages
            self.sqs_wrapper.delete_messages(queue_url, messages)
            print(f"Messages deleted from queue #{i+1}")
        else:
            print(f"No messages received by queue #{i+1}")

    print(self.DASHES)

def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
    """Poll a single queue for messages."""
    all_messages = []
    max_polls = 3 # Limit polling to avoid infinite loops

```

```

    for poll_count in range(max_polls):
        messages = self.sqs_wrapper.receive_messages(queue_url, 10)

        if messages:
            all_messages.extend(messages)
            print(f" Received {len(messages)} messages in poll {poll_count +
1}")

            # Small delay between polls
            time.sleep(1)
        else:
            print(f" No messages in poll {poll_count + 1}")
            break

    return all_messages

def _cleanup_resources(self) -> None:
    """Clean up all resources created during the scenario."""
    print("Cleaning up resources...")

    # Delete queues
    for i, queue_url in enumerate(self.queue_urls):
        if queue_url:
            delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}?
(y/n): ", q.is_yesno)
            if delete_queue:
                try:
                    self.sqs_wrapper.delete_queue(queue_url)
                    print(f"Deleted queue #{i+1}")
                except Exception as e:
                    print(f"Error deleting queue #{i+1}: {e}")

    # Unsubscribe from topic
    for i, subscription_arn in enumerate(self.subscription_arns):
        if subscription_arn:
            try:
                self.sns_wrapper.unsubscribe(subscription_arn)
                print(f"Unsubscribed subscription #{i+1}")
            except Exception as e:
                print(f"Error unsubscribing #{i+1}: {e}")

    # Delete topic
    if self.topic_arn:
        delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
q.is_yesno)

```

```

        if delete_topic:
            try:
                self.sns_wrapper.delete_topic(self.topic_arn)
                print(f"Deleted topic {self.topic_name}")
            except Exception as e:
                print(f"Error deleting topic: {e}")

    print("Resource cleanup complete.")

```

Crie classes que envolvam as operações do Amazon SNS e do Amazon SQS para uso no cenário.

```

class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """

```



```

        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
    try:
        # Add .fifo suffix for FIFO topics
        if is_fifo and not topic_name.endswith('.fifo'):
            topic_name += '.fifo'

        attributes = {}
        if is_fifo:
            attributes['FifoTopic'] = 'true'
            if content_based_deduplication:
                attributes['ContentBasedDeduplication'] = 'true'

        response = self.sns_client.create_topic(
            Name=topic_name,
            Attributes=attributes
        )

        topic_arn = response['TopicArn']
        logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
        return topic_arn

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error creating topic {topic_name}: {error_code} -
{e}")
        raise

    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

```

```

:param topic_arn: The ARN of the SNS topic.
:param queue_arn: The ARN of the SQS queue.
:param filter_policy: Optional JSON filter policy for message filtering.
:return: The ARN of the subscription.
:raises ClientError: If the subscription fails.
"""
try:
    attributes = {}
    if filter_policy:
        attributes['FilterPolicy'] = filter_policy

    response = self.sns_client.subscribe(
        TopicArn=topic_arn,
        Protocol='sqs',
        Endpoint=queue_arn,
        Attributes=attributes
    )

    subscription_arn = response['SubscriptionArn']
    logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
    return subscription_arn

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
    raise

def publish_message(
    self,
    topic_arn: str,
    message: str,
    tone_attribute: Optional[str] = None,
    deduplication_id: Optional[str] = None,
    message_group_id: Optional[str] = None
) -> str:
    """
    Publish a message to an SNS topic.

    :param topic_arn: The ARN of the SNS topic.
    :param message: The message content to publish.
    :param tone_attribute: Optional tone attribute for message filtering.
    :param deduplication_id: Optional deduplication ID for FIFO topics.

```

```

:param message_group_id: Optional message group ID for FIFO topics.
:return: The message ID of the published message.
:raises ClientError: If the message publication fails.
"""
try:
    publish_args = {
        'TopicArn': topic_arn,
        'Message': message
    }

    # Add message attributes if tone is specified
    if tone_attribute:
        publish_args['MessageAttributes'] = {
            'tone': {
                'DataType': 'String',
                'StringValue': tone_attribute
            }
        }

    # Add FIFO-specific parameters
    if message_group_id:
        publish_args['MessageGroupId'] = message_group_id

    if deduplication_id:
        publish_args['MessageDeduplicationId'] = deduplication_id

    response = self.sns_client.publish(**publish_args)

    message_id = response['MessageId']
    logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
    return message_id

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error publishing message to topic: {error_code} -
{e}")
    raise

def unsubscribe(self, subscription_arn: str) -> bool:
    """
    Unsubscribe from an SNS topic.

```

```
    :param subscription_arn: The ARN of the subscription to remove.
    :return: True if successful.
    :raises ClientError: If the unsubscribe operation fails.
    """
    try:
        self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

        logger.info(f"Unsubscribed: {subscription_arn}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'NotFound':
            logger.warning(f"Subscription not found: {subscription_arn}")
            return True  # Already unsubscribed
        else:
            logger.error(f"Error unsubscribing: {error_code} - {e}")
            raise

def delete_topic(self, topic_arn: str) -> bool:
    """
    Delete an SNS topic.

    :param topic_arn: The ARN of the topic to delete.
    :return: True if successful.
    :raises ClientError: If the topic deletion fails.
    """
    try:
        self.sns_client.delete_topic(TopicArn=topic_arn)

        logger.info(f"Deleted topic: {topic_arn}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'NotFound':
            logger.warning(f"Topic not found: {topic_arn}")
            return True  # Already deleted
        else:
            logger.error(f"Error deleting topic: {error_code} - {e}")
            raise
```

```

def list_topics(self) -> list:
    """
    List all SNS topics in the account using pagination.

    :return: List of topic ARNs.
    :raises ClientError: If listing topics fails.
    """
    try:
        topics = []
        paginator = self.sns_client.get_paginator('list_topics')

        for page in paginator.paginate():
            topics.extend([topic['TopicArn'] for topic in page.get('Topics',
[])])

        logger.info(f"Found {len(topics)} topics")
        return topics

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        if error_code == 'AuthorizationError':
            logger.error("Authorization error listing topics - check IAM
permissions")
        else:
            logger.error(f"Error listing topics: {error_code} - {e}")
        raise

class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """

```

```

        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)

def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
    """
    Create an SQS queue.

    :param queue_name: The name of the queue to create.
    :param is_fifo: Whether to create a FIFO queue.
    :return: The URL of the created queue.
    :raises ClientError: If the queue creation fails.
    """
    try:
        # Add .fifo suffix for FIFO queues
        if is_fifo and not queue_name.endswith('.fifo'):
            queue_name += '.fifo'

        attributes = {}
        if is_fifo:
            attributes['FifoQueue'] = 'true'

        response = self.sqs_client.create_queue(
            QueueName=queue_name,
            Attributes=attributes
        )

        queue_url = response['QueueUrl']
        logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
        return queue_url

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
        raise

def get_queue_arn(self, queue_url: str) -> str:
    """

```

```

    Get the ARN of an SQS queue.

    :param queue_url: The URL of the queue.
    :return: The ARN of the queue.
    :raises ClientError: If getting queue attributes fails.
    """
    try:
        response = self.sqs_client.get_queue_attributes(
            QueueUrl=queue_url,
            AttributeNames=['QueueArn']
        )

        queue_arn = response['Attributes']['QueueArn']
        logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
        return queue_arn

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error getting queue ARN: {error_code} - {e}")
        raise

def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
queue_url: str) -> bool:
    """
    Set the queue policy to allow SNS to send messages to the queue.

    :param queue_arn: The ARN of the SQS queue.
    :param topic_arn: The ARN of the SNS topic.
    :param queue_url: The URL of the SQS queue.
    :return: True if successful.
    :raises ClientError: If setting the queue policy fails.
    """
    try:
        # Create policy that allows SNS to send messages to the queue
        policy = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "sns.amazonaws.com"
                    },
                    "Action": "sqs:SendMessage",

```

```

        "Resource": queue_arn,
        "Condition": {
            "ArnEquals": {
                "aws:SourceArn": topic_arn
            }
        }
    ]
}

self.sqs_client.set_queue_attributes(
    QueueUrl=queue_url,
    Attributes={
        'Policy': json.dumps(policy)
    }
)

logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
return True

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error setting queue policy: {error_code} - {e}")
    raise

def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
    """
    Receive messages from an SQS queue.

    :param queue_url: The URL of the queue to receive messages from.
    :param max_messages: Maximum number of messages to receive (1-10).
    :return: List of received messages.
    :raises ClientError: If receiving messages fails.
    """
    try:
        # Ensure max_messages is within valid range
        max_messages = max(1, min(10, max_messages))

        response = self.sqs_client.receive_message(
            QueueUrl=queue_url,
            MaxNumberOfMessages=max_messages,

```



```

        WaitTimeSeconds=2, # Short polling
        MessageAttributeNames=['All']
    )

    messages = response.get('Messages', [])
    logger.info(f"Received {len(messages)} messages from {queue_url}")
    return messages

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error receiving messages: {error_code} - {e}")
    raise

def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
    """
    Delete messages from an SQS queue in batches.

    :param queue_url: The URL of the queue.
    :param messages: List of messages to delete.
    :return: True if successful.
    :raises ClientError: If deleting messages fails.
    """
    try:
        if not messages:
            return True

        # Build delete entries for batch delete
        delete_entries = []
        for i, message in enumerate(messages):
            delete_entries.append({
                'Id': str(i),
                'ReceiptHandle': message['ReceiptHandle']
            })

        # Delete messages in batches of 10 (SQS limit)
        batch_size = 10
        for i in range(0, len(delete_entries), batch_size):
            batch = delete_entries[i:i + batch_size]

            response = self.sqs_client.delete_message_batch(
                QueueUrl=queue_url,
                Entries=batch
            )

```

```

        )

        # Check for failures
        if 'Failed' in response and response['Failed']:
            for failed in response['Failed']:
                logger.warning(f"Failed to delete message: {failed}")

        logger.info(f"Deleted {len(messages)} messages from {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error deleting messages: {error_code} - {e}")
        raise

def delete_queue(self, queue_url: str) -> bool:
    """
    Delete an SQS queue.

    :param queue_url: The URL of the queue to delete.
    :return: True if successful.
    :raises ClientError: If the queue deletion fails.
    """
    try:
        self.sqs_client.delete_queue(QueueUrl=queue_url)

        logger.info(f"Deleted queue: {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
            logger.warning(f"Queue not found: {queue_url}")
            return True # Already deleted
        else:
            logger.error(f"Error deleting queue: {error_code} - {e}")
            raise

def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
    """
    List all SQS queues in the account using pagination.

```

```

:param queue_name_prefix: Optional prefix to filter queue names.
:return: List of queue URLs.
:raises ClientError: If listing queues fails.
"""
try:
    queue_urls = []
    paginator = self.sqs_client.get_paginator('list_queues')

    page_params = {}
    if queue_name_prefix:
        page_params['QueueNamePrefix'] = queue_name_prefix

    for page in paginator.paginate(**page_params):
        queue_urls.extend(page.get('QueueUrls', []))

    logger.info(f"Found {len(queue_urls)} queues")
    return queue_urls

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    if error_code == 'AccessDenied':
        logger.error("Access denied listing queues - check IAM
permissions")
    else:
        logger.error(f"Error listing queues: {error_code} - {e}")
    raise

def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
    """
    Send a message to an SQS queue.

    :param queue_url: The URL of the queue.
    :param message_body: The message content.
    :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
    :return: The message ID.
    :raises ClientError: If sending the message fails.
    """
    try:
        send_params = {
            'QueueUrl': queue_url,
            'MessageBody': message_body,
            **kwargs

```

```
}

response = self.sqs_client.send_message(**send_params)

message_id = response['MessageId']
logger.info(f"Sent message to {queue_url} with ID: {message_id}")
return message_id

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error sending message: {error_code} - {e}")
    raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicar](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Assinar](#)
 - [Cancelar assinatura](#)

Swift

SDK para Swift

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: ""
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        "",
        discussion: ""
        Supports filtering using a "tone" attribute.
        ""
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
```

```
        str = readLine()

        if str != nil && str?.count == 0 {
            str = nil
        }
    }

    return str!
}

/// Ask a yes/no question.
///
/// - Parameter prompt: A prompt string to print.
///
/// - Returns: `true` if the user answered "Y", otherwise `false`.
func yesNoRequest(prompt: String) -> Bool {
    while true {
        let answer = stringRequest(prompt: prompt).lowercased()
        if answer == "y" || answer == "n" {
            return answer == "y"
        }
    }
}

/// Display a menu of options then request a selection.
///
/// - Parameters:
///   - prompt: A prompt string to display before the menu.
///   - options: An array of strings giving the menu options.
///
/// - Returns: The index number of the selected option or 0 if no item was
///   selected.
func menuRequest(prompt: String, options: [String]) -> Int {
    let numOptions = options.count

    if numOptions == 0 {
        return 0
    }

    print(prompt)

    for (index, value) in options.enumerated() {
        print("\(index) \(value)")
    }
}
```

```
repeat {
    print("Enter your selection (0 - \$(numOptions-1)): ", terminator: "")
    if let answer = readLine() {
        guard let answer = Int(answer) else {
            print("Please enter the number matching your selection.")
            continue
        }

        if answer >= 0 && answer < numOptions {
            return answer
        } else {
            print("Please enter the number matching your selection.")
        }
    }
} while true
}

/// Ask the user too press RETURN. Accepts any input but ignores it.
///
/// - Parameter prompt: The text prompt to display.
func returnRequest(prompt: String) {
    print(prompt, terminator: "")
    _ = readLine()
}

var attrValues = [
    "<none>",
    "cheerful",
    "funny",
    "serious",
    "sincere"
]

/// Ask the user to choose one of the attribute values to use as a filter.
///
/// - Parameters:
///   - message: A message to display before the menu of values.
///   - attrValues: An array of strings giving the values to choose from.
///
/// - Returns: The string corresponding to the selected option.
func askForFilter(message: String, attrValues: [String]) -> String? {
    print(message)
    for (index, value) in attrValues.enumerated() {
```

```

        print("  [\\(index)] \\(value)")
    }

    var answer: Int?
    repeat {
        answer = Int(stringRequest(prompt: "Select an value for the 'tone'
attribute or 0 to end: "))
    } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

    if answer == 0 {
        return nil
    }
    return attrValues[answer!]
}

/// Prompts the user for filter terms and constructs the attribute
/// record that specifies them.
///
/// - Returns: A mapping of "FilterPolicy" to a JSON string representing
///   the user-defined filter.
func buildFilterAttributes() -> [String:String] {
    var attr: [String:String] = [:]
    var filterString = ""

    var first = true

    while let ans = askForFilter(message: "Choose a value to apply to the
'tone' attribute.",
                                attrValues: attrValues) {
        if !first {
            filterString += ","
        }
        first = false

        filterString += "\\\"\\(ans)\\\""
    }

    let filterJSON = "{ \"tone\": [\\(filterString)]}"
    attr["FilterPolicy"] = filterJSON

    return attr
}
/// Create a queue, returning its URL string.
///

```



```

    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
    throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]

            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that
name was recently deleted.")
                continue
            }
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws ->
    String? {

```

```

    let output = try await sqsClient.getQueueAttributes(
        input: GetQueueAttributesInput(
            attributeNames: [.queueArn],
            queueUrl: queueUrl
        )
    )

    guard let attributes = output.attributes else {
        return nil
    }

    return attributes["QueueArn"]
}

/// Applies the needed policy to the specified queue.
///
/// - Parameters:
///   - sqsClient: The Amazon SQS client to use.
///   - queueUrl: The queue to apply the policy to.
///   - queueArn: The ARN of the queue to apply the policy to.
///   - topicArn: The topic that should have access via the policy.
///
/// - Throws: Errors from the SQS `SetQueueAttributes` action.
func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                    queueArn: String, topicArn: String) async throws {
    _ = try await sqsClient.setQueueAttributes(
        input: SetQueueAttributesInput(
            attributes: [
                "Policy":
                    """
                    {
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "sns.amazonaws.com"
                                },
                                "Action": "sqs:SendMessage",
                                "Resource": "\(queueArn)",
                                "Condition": {
                                    "ArnEquals": {
                                        "aws:SourceArn": "\(topicArn)"
                                    }
                                }
                            }
                        ]
                    }
                    """
            ]
        )
    )
}

```

```

        }
    ]
}
"""

    ],
    queueUrl: queueUrl
)
)
}

/// Receive the available messages on a queue, outputting them to the
/// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
/// all the received messages.
///
/// - Parameters:
///   - sqsClient: The Amazon SQS client to use.
///   - queueUrl: The SQS queue on which to receive messages.
///
/// - Throws: Errors from `SQSClient.receiveMessage()`
///
/// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
///   items, each describing one received message in the format needed to
///   delete it.
func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async
throws
    ->
[SQSClientTypes.DeleteMessageBatchRequestEntry] {
    let output = try await sqsClient.receiveMessage(
        input: ReceiveMessageInput(
            maxNumberOfMessages: 10,
            queueUrl: queueUrl
        )
    )

    guard let messages = output.messages else {
        print("No messages received.")
        return []
    }

    var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

    // Print out all the messages that were received, including their
    // attributes, if any.

```

```

    for message in messages {
        print("Message ID:      \(message.messageId ?? "<unknown>")")
        print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
        print("Message JSON:    \(message.body ?? "<body missing>")")

        if message.receiptHandle != nil {
            deleteList.append(
                SQSClientTypes.DeleteMessageBatchRequestEntry(
                    id: message.messageId,
                    receiptHandle: message.receiptHandle
                )
            )
        }
    }

    return deleteList
}

/// Delete all the messages in the specified list.
///
/// - Parameters:
///   - sqsClient: The Amazon SQS client to use.
///   - queueUrl: The SQS queue to delete messages from.
///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
///     describing the messages to delete.
///
/// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                      deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
    let output = try await sqsClient.deleteMessageBatch(
        input: DeleteMessageBatchInput(entries: deleteList, queueUrl:
queueUrl)
    )

    if let failed = output.failed {
        print("\(failed.count) errors occurred deleting messages from the
queue.")
        for message in failed {
            print("---> Failed to delete message \(message.id ?? "<unknown
ID>") with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
        }
    }
}

```

```

    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
            \(\rowOfStars)
            Welcome to the cross-service messaging with topics and queues
example.
            In this workflow, you'll create an SNS topic, then create two SQS
            queues which will be subscribed to that topic.

            You can specify several options for configuring the topic, as well
as
            the queue subscriptions. You can then post messages to the topic
and
            receive the results on the queues.
            \(\rowOfStars)\n
            """)
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region:
region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region:
region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //     to use content-based deduplication instead of requiring a
        //     deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/
N)? ")
        var isContentBasedDeduplication = false

        if isFIFO {
            print("""
                \(\rowOfStars)
                Because you've chosen to create a FIFO topic, deduplication is

```

```

supported.

Deduplication IDs are either set in the message or are
automatically
generated from the content using a hash function.

If a message is successfully published to an SNS FIFO topic,
any
message published and found to have the same deduplication ID
(within a five-minute deduplication interval), is accepted but
not delivered.

For more information about deduplication, see:
https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html.
"""
    )

    isContentBasedDeduplication = yesNoRequest(
        prompt: "Use content-based deduplication instead of entering a
deduplication ID (Y/N)? ")
    print(rowOfStars)
}

var topicName = stringRequest(prompt: "Enter the name of the topic to
create: ")

// 2. Create the topic. Append ".fifo" to the name if FIFO was
// requested, and set the "FifoTopic" attribute to "true" if so as
// well. Set the "ContentBasedDeduplication" attribute to "true" if
// content-based deduplication was requested.

if isFIFO {
    topicName += ".fifo"
}

print("Topic name: \(topicName)")

var attributes = [
    "FifoTopic": (isFIFO ? "true" : "false")
]

// If it's a FIFO topic with content-based deduplication, set the
// "ContentBasedDeduplication" attribute.

```

```
    if isContentBasedDeduplication {
        attributes["ContentBasedDeduplication"] = "true"
    }

    // Create the topic and retrieve the ARN.

    let output = try await snsClient.createTopic(
        input: CreateTopicInput(
            attributes: attributes,
            name: topicName
        )
    )

    guard let topicArn = output.topicArn else {
        print("No topic ARN returned!")
        return
    }

    print("""
        Topic '\(topicName)' has been created with the
        topic ARN '\(topicArn)'.
    """)

    )

    print(rowOfStars)

    // 3. Create an SQS queue. Append ".fifo" to the name if one of the
    //     FIFO topic configurations was chosen, and set "FifoQueue" to
    //     "true" if the topic is FIFO.

    print("""
        Next, you will create two SQS queues that will be subscribed
        to the topic you just created.\n
    """)

    )

    let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                     sqsClient: sqsClient, isFIFO: isFIFO)

    guard let q1Url else {
        print("Unable to create queue 1!")
        return
    }
}
```

```
// 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

guard let q1Arn else {
    print("Unable to get ARN of queue 1!")
    return
}
print("Got queue 1 ARN: \(q1Arn)")

// 5. Attach an AWS IAM policy to the queue using
//     `SetQueueAttributes`.

try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                        queueArn: q1Arn, topicArn: topicArn)

// 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
//     the request. Set the protocol to "sqs". Set the queue ARN to the
//     ARN just received in step 5. For FIFO topics, give the option to
//     apply a filter. A filter allows only matching messages to enter
//     the queue.

var q1Attributes: [String:String]? = nil

if isFIFO {
    print(
        """
        If you add a filter to this subscription, then only the filtered
messages will
        be received in the queue. For information about message
filtering, see
https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html
        For this example, you can filter messages by a 'tone' attribute.
        """
    )

    let subPrompt = """
        Would you like to filter messages for the first queue's
subscription to the
        topic \(topicName) (Y/N)?
    """
}
```



```

        """"
        if (yesNoRequest(prompt: subPrompt)) {
            q1Attributes = buildFilterAttributes()
        }
    }

    let sub1Output = try await snsClient.subscribe(
        input: SubscribeInput(
            attributes: q1Attributes,
            endpoint: q1Arn,
            protocol: "sqs",
            topicArn: topicArn
        )
    )

    guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
        print("Invalid subscription ARN returned for queue 1!")
        return
    }

    // 7. Repeat steps 3-6 for the second queue.

    let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                     sqsClient: sqsClient, isFIFO: isFIFO)

    guard let q2Url else {
        print("Unable to create queue 2!")
        return
    }

    let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

    guard let q2Arn else {
        print("Unable to get ARN of queue 2!")
        return
    }
    print("Got queue 2 ARN: \(q2Arn)")

    try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                             queueArn: q2Arn, topicArn: topicArn)

    var q2Attributes: [String:String]? = nil

```

```

        if isFIFO {
            let subPrompt = ""
            Would you like to filter messages for the second queue's
subscription to the
            topic \((topicName) (Y/N)?
            ""
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //     body for each message. Handle the types of topic correctly (SEE
        //     MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\((rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.

```

```

        if isFIFO {
            if first {
                print("""
                    Because you're using a FIFO topic, you must set a message
                    group ID. All messages within the same group will be
                    received in the same order in which they were published.

\n
                    """)
            }
            publishInput.messageGroupId = stringRequest(prompt: "Enter a
message group ID for this message: ")

            if !isContentBasedDeduplication {
                if first {
                    print("""
                        Because you're not using content-based
deduplication, you
                        must enter a deduplication ID. If other messages
with the
                        same deduplication ID are published within the same
deduplication interval, they will not be delivered.
                    """)
                }
            }
            publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
        }
    }

    // Allow the user to add a value for the "tone" attribute if they
    // wish to do so.

    var messageAttributes: [String: SNSClientTypes.MessageAttributeValue]
= [:]

    let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

    if attrValSelection != 0 {
        let val = SNSClientTypes.MessageAttributeValue(dataType:
"String", stringValue: attrValues[attrValSelection])
        messageAttributes["tone"] = val
    }

```

```
publishInput.messageAttributes = messageAttributes

// Publish the message and display its ID.

let publishOutput = try await snsClient.publish(input: publishInput)

guard let messageId = publishOutput.messageId else {
    print("Unable to get the published message's ID!")
    return
}

print("Message published with ID \(messageId).")
first = false

// 9. Repeat step 8 until the user says they don't want to post
//    another.

} while (yesNoRequest(prompt: "Post another message (Y/N)? "))

// 10. Display a list of the messages in each queue by using
//     `ReceiveMessage`. Show at least the body and the attributes.

print(rowOfStars)
print("Contents of queue 1:")
let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
print("\n\nContents of queue 2:")
let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
print(rowOfStars)

returnRequest(prompt: "\nPress return to clean up: ")

// 11. Delete the received messages using `DeleteMessageBatch`.

print("Deleting the messages from queue 1...")
try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
print("\nDeleting the messages from queue 2...")
try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

// 12. Unsubscribe and delete both queues.
```

```
        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publicar](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Assinar](#)
 - [Cancelar assinatura](#)

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Envie e receba lotes de mensagens com o Amazon SQS usando um SDK AWS

Os exemplos de código a seguir mostram como:

- Criar uma fila do Amazon SQS.
- Enviar lotes de mensagens para a fila.
- Receber lotes de mensagens de uma fila.
- Excluir lotes de mensagens de uma fila.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Conforme mostrado nos exemplos a seguir, você pode gerenciar operações de mensagens em lote com o Amazon SQS usando duas abordagens diferentes com o AWS SDK for Java 2.x:

`SendRecvBatch.java` usa operações em lote explícitas. Você cria manualmente lotes de mensagens e chama `sendMessageBatch()` e `deleteMessageBatch()` diretamente. Você também gerencia respostas em lote, incluindo mensagens com falha. Essa abordagem oferece controle total sobre o dimensionamento do lote e o gerenciamento de erros. No entanto, ela requer mais código para gerenciar a lógica de agrupamento em lote.

`SimpleProducerConsumer.java` usa a `SqsAsyncBatchManager` biblioteca de alto nível para agrupamento automático de solicitações. Você faz chamadas individuais para `sendMessage()` e `deleteMessage()` com as mesmas assinaturas de método do cliente padrão. O SDK armazena automaticamente essas chamadas em buffer e as envia como operações em lote. Essa abordagem requer alterações mínimas no código e, ao mesmo tempo, fornece benefícios de desempenho em lote.

Use lotes explícitos quando precisar de um controle refinado sobre a composição do lote e o gerenciamento de erros. Use o agrupamento automático em lote quando quiser otimizar o desempenho com o mínimo de alterações no código.

`SendRecvBatch.java` - Usa operações em lote explícitas com mensagens.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.BatchResultErrorEntry;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResponse;
```

```
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.MessageAttributeValue;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

/**
 * This code demonstrates basic message operations in Amazon Simple Queue Service
 * (Amazon SQS).
 */

public class SendRecvBatch {
    private static final Logger LOGGER =
        LoggerFactory.getLogger(SendRecvBatch.class);
    private static final SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {
        usageDemo();
    }
}
```



```

    }
    /**
     * Send a batch of messages in a single request to an SQS queue.
     * This request may return overall success even when some messages were not
    sent.
     * The caller must inspect the Successful and Failed lists in the response
    and
     * resend any failed messages.
     *
     * @param queueUrl The URL of the queue to receive the messages.
     * @param messages The messages to send to the queue. Each message contains
    a body and attributes.
     * @return The response from SQS that contains the list of successful and
    failed messages.
     */
    public static SendMessageBatchResponse sendMessages(
        String queueUrl, List<MessageEntry> messages) {

        try {
            List<SendMessageBatchRequestEntry> entries = new ArrayList<>();

            for (int i = 0; i < messages.size(); i++) {
                MessageEntry message = messages.get(i);
                entries.add(SendMessageBatchRequestEntry.builder()
                    .id(String.valueOf(i))
                    .messageBody(message.getBody())
                    .messageAttributes(message.getAttributes())
                    .build());
            }

            SendMessageBatchRequest sendBatchRequest =
                SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            SendMessageBatchResponse response =
                sqsClient.sendMessageBatch(sendBatchRequest);

            if (!response.successful().isEmpty()) {
                for (SendMessageBatchResultEntry resultEntry :
                    response.successful()) {
                    LOGGER.info("Message sent: {}: {}", resultEntry.messageId(),

```

```

        messages.get(Integer.parseInt(resultEntry.id())).getBody());
    }
}

    if (!response.failed().isEmpty()) {
        for (BatchResultErrorEntry errorEntry : response.failed()) {
            LOGGER.warn("Failed to send: {}: {}", errorEntry.id(),
messages.get(Integer.parseInt(errorEntry.id())).getBody());
        }
    }

    return response;

} catch (SqsException e) {
    LOGGER.error("Send messages failed to queue: {}", queueUrl, e);
    throw e;
}
}

/**
 * Receive a batch of messages in a single request from an SQS queue.
 *
 * @param queueUrl The URL of the queue from which to receive messages.
 * @param maxNumber The maximum number of messages to receive (capped at 10
by SQS).
 *
 * The actual number of messages received might be less.
 * @param waitTime The maximum time to wait (in seconds) before returning.
When
 *
 * this number is greater than zero, long polling is used.
This
 *
 * can result in reduced costs and fewer false empty
responses.
 * @return The list of Message objects received. These each contain the body
 * of the message and metadata and custom attributes.
 */
public static List<Message> receiveMessages(String queueUrl, int maxNumber,
int waitTime) {
    try {
        ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
            .queueUrl(queueUrl)
            .numberOfMessages(maxNumber)

```

```

        .waitTimeSeconds(waitTime)
        .messageAttributeNames("All")
        .build();

    List<Message> messages =
sqsClient.receiveMessage(receiveRequest).messages();

    for (Message message : messages) {
        LOGGER.info("Received message: {}: {}", message.messageId(),
message.body());
    }

    return messages;

} catch (SqsException e) {
    LOGGER.error("Couldn't receive messages from queue: {}", queueUrl,
e);
    throw e;
}

}

/**
 * Delete a batch of messages from a queue in a single request.
 *
 * @param queueUrl The URL of the queue from which to delete the messages.
 * @param messages The list of messages to delete.
 * @return The response from SQS that contains the list of successful and
failed
 *         message deletions.
 */
public static DeleteMessageBatchResponse deleteMessages(String queueUrl,
List<Message> messages) {
    try {
        List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();

        for (int i = 0; i < messages.size(); i++) {
            entries.add(DeleteMessageBatchRequestEntry.builder()
                .id(String.valueOf(i))
                .receiptHandle(messages.get(i).receiptHandle())
                .build());
        }

        DeleteMessageBatchRequest deleteRequest =
DeleteMessageBatchRequest.builder()

```

```

        .queueUrl(queueUrl)
        .entries(entries)
        .build();

        DeleteMessageBatchResponse response =
sqsClient.deleteMessageBatch(deleteRequest);

        if (!response.successful().isEmpty()) {
            for (DeleteMessageBatchResultEntry resultEntry :
response.successful()) {
                LOGGER.info("Deleted {}",
messages.get(Integer.parseInt(resultEntry.id())).receiptHandle());
            }
        }

        if (!response.failed().isEmpty()) {
            for (BatchResultErrorEntry errorEntry : response.failed()) {
                LOGGER.warn("Could not delete {}",
messages.get(Integer.parseInt(errorEntry.id())).receiptHandle());
            }
        }

        return response;

    } catch (SqsException e) {
        LOGGER.error("Couldn't delete messages from queue {}", queueUrl, e);
        throw e;
    }
}

/**
 * Helper class to represent a message with body and attributes.
 */
public static class MessageEntry {
    private final String body;
    private final Map<String, MessageAttributeValue> attributes;

    public MessageEntry(String body, Map<String, MessageAttributeValue>
attributes) {
        this.body = body;
        this.attributes = attributes != null ? attributes : new HashMap<>();
    }

    public String getBody() {

```

```
        return body;
    }

    public Map<String, MessageAttributeValue> getAttributes() {
        return attributes;
    }
}

/**
 * Shows how to:
 * * Read the lines from a file and send the lines in
 *   batches of 10 as messages to a queue.
 * * Receive the messages in batches until the queue is empty.
 * * Reassemble the lines of the file and verify they match the original
 *   file.
 */
public static void usageDemo() {
    LOGGER.info("-".repeat(88));
    LOGGER.info("Welcome to the Amazon Simple Queue Service (Amazon SQS)
demo!");
    LOGGER.info("-".repeat(88));

    String queueUrl = null;
    try {
        // Create a queue for the demo.
        String queueName = "sqs-usage-demo-message-wrapper-" +
System.currentTimeMillis();
        CreateQueueRequest createRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();
        queueUrl = sqsClient.createQueue(createRequest).queueUrl();
        LOGGER.info("Created queue: {}", queueUrl);

        try (InputStream inputStream =
SendRecvBatch.class.getResourceAsStream("/log4j2.xml");
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream))) {

            List<String> lines = reader.lines().toList();

            // Send file lines in batches.
            int batchSize = 10;
            LOGGER.info("Sending file lines in batches of {} as messages.",
batchSize);
```

```

        for (int i = 0; i < lines.size(); i += batchSize) {
            List<MessageEntry> messageBatch = new ArrayList<>();

            for (int j = i; j < Math.min(i + batchSize, lines.size()); j++) {
                String line = lines.get(j);
                if (line == null || line.trim().isEmpty()) {
                    continue; // Skip empty lines.
                }

                Map<String, MessageAttributeValue> attributes = new
HashMap<>();

                attributes.put("line", MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(String.valueOf(j))
                    .build());

                messageBatch.add(new MessageEntry(lines.get(j),
attributes));
            }

            sendMessages(queueUrl, messageBatch);
            System.out.print(".");
            System.out.flush();
        }

        LOGGER.info("\nDone. Sent {} messages.", lines.size());

        // Receive and process messages.
        LOGGER.info("Receiving, handling, and deleting messages in
batches of {}. ", batchSize);
        String[] receivedLines = new String[lines.size()];
        boolean moreMessages = true;

        while (moreMessages) {
            List<Message> receivedMessages = receiveMessages(queueUrl,
batchSize, 5);

            for (Message message : receivedMessages) {
                int lineNumber =
Integer.parseInt(message.messageAttributes().get("line").stringValue());
                receivedLines[lineNumber] = message.body();
            }
        }
    }

```

```

        if (!receivedMessages.isEmpty()) {
            deleteMessages(queueUrl, receivedMessages);
        } else {
            moreMessages = false;
        }
    }

    LOGGER.info("\nDone.");

    // Verify that all lines were received correctly.
    boolean allLinesMatch = true;
    for (int i = 0; i < lines.size(); i++) {
        String originalLine = lines.get(i);
        String receivedLine = receivedLines[i] == null ? "" :
receivedLines[i];

        if (!originalLine.equals(receivedLine)) {
            allLinesMatch = false;
            break;
        }
    }

    if (allLinesMatch) {
        LOGGER.info("Successfully reassembled all file lines!");
    } else {
        LOGGER.info("Uh oh, some lines were missed!");
    }
}
} catch (SqsException e) {
    LOGGER.error("SQS operation failed", e);
} catch (RuntimeException | IOException e) {
    LOGGER.error("Unexpected runtime error during demo", e);
} finally {
    // Clean up by deleting the queue if it was created.
    if (queueUrl != null) {
        try {
            DeleteQueueRequest deleteQueueRequest =
DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();
            sqsClient.deleteQueue(deleteQueueRequest);
            LOGGER.info("Deleted queue: {}", queueUrl);
        } catch (SqsException e) {

```

```

        LOGGER.error("Failed to delete queue: {}", queueUrl, e);
    }
}

LOGGER.info("Thanks for watching!");
LOGGER.info("-".repeat(88));
}
}

```

SimpleProducerConsumer.java - Usa o agrupamento automático de mensagens.

```

package com.example.sqs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsAsyncClient;
import software.amazon.awssdk.services.sqs.batchmanager.SqsAsyncBatchManager;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageResponse;
import software.amazon.awssdk.core.exception.SdkException;

import java.math.BigInteger;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Demonstrates the AWS SDK for Java 2.x Automatic Request Batching API for
 * Amazon SQS.
 */

```


- * This example showcases the high-level `SqsAsyncBatchManager` library that provides
 - * efficient batching and buffering for SQS operations. The batch manager offers
 - * methods that directly mirror `SqsAsyncClient` methods—`sendMessage`, `changeMessageVisibility`,
 - * `deleteMessage`, and `receiveMessage`—making it a drop-in replacement with minimal code changes.
- *
- * Key features of the `SqsAsyncBatchManager`:
 - * - Automatic batching: The SDK automatically buffers individual requests and sends them
 - * as batches when `maxBatchSize` (default: 10) or `sendRequestFrequency` (default: 200ms)
 - * thresholds are reached
 - * - Familiar API: Method signatures match `SqsAsyncClient` exactly, requiring no learning curve
 - * - Background optimization: The batch manager maintains internal buffers and handles
 - * batching logic transparently
 - * - Asynchronous operations: All methods return `CompletableFuture` for non-blocking execution
- *
- * Performance benefits demonstrated:
 - * - Reduced API calls: Multiple individual requests are consolidated into single batch operations
 - * - Lower costs: Fewer API calls result in reduced SQS charges
 - * - Higher throughput: Batch operations process more messages per second
 - * - Efficient resource utilization: Fewer network round trips and better connection reuse
- *
- * This example compares:
 - * 1. Single-message operations using `SqsAsyncClient` directly
 - * 2. Batch operations using `SqsAsyncBatchManager` with identical method calls
- *
- * Usage patterns:
 - * - Set batch size to 1 to use `SqsAsyncClient` for baseline performance measurement
 - * - Set batch size > 1 to use `SqsAsyncBatchManager` for optimized batch processing
 - * - Monitor real-time throughput metrics to observe performance improvements
- *
- * Prerequisites:
 - * - AWS SDK for Java 2.x version 2.28.0 or later
 - * - An existing SQS queue

```
* - Valid AWS credentials configured
*
* The program displays real-time metrics showing the dramatic performance
difference
* between individual operations and automatic batching.
*/
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Logger log =
LoggerFactory.getLogger(SimpleProducerConsumer.class);

    /**
     * Runs the SQS batching demonstration with user-configured parameters.
     *
     * Prompts for queue name, thread counts, batch size, message size, and
runtime.
     * Creates producer and consumer threads to demonstrate batching performance.
     *
     * @param args command line arguments (not used)
     * @throws InterruptedException if thread operations are interrupted
     */
    public static void main(String[] args) throws InterruptedException {

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
        final int producerCount = input.nextInt();

        System.out.print("Enter the number of consumers: ");
        final int consumerCount = input.nextInt();

        System.out.print("Enter the number of messages per batch: ");
        final int batchSize = input.nextInt();

        System.out.print("Enter the message size in bytes: ");
        final int messageSizeByte = input.nextInt();

        System.out.print("Enter the run time in minutes: ");
        final int runTimeMinutes = input.nextInt();
```

```
// Create SQS async client and batch manager for all operations.
// The SqsAsyncBatchManager is created from the SqsAsyncClient using the
// batchManager() factory method, which provides default batching
configuration.
// This high-level library automatically handles request buffering and
batching
// while maintaining the same method signatures as SqsAsyncClient.
final SqsAsyncClient sqsAsyncClient = SqsAsyncClient.create();
final SqsAsyncBatchManager batchManager = sqsAsyncClient.batchManager();

final String queueUrl =
sqsAsyncClient.getQueueUrl(GetQueueUrlRequest.builder()
    .queueName(queueName)
    .build()).join().queueUrl();

// The flag used to stop producer, consumer, and monitor threads.
final AtomicBoolean stop = new AtomicBoolean(false);

// Start the producers.
final AtomicInteger producedCount = new AtomicInteger();
final Thread[] producers = new Thread[producerCount];
for (int i = 0; i < producerCount; i++) {
    if (batchSize == 1) {
        producers[i] = new Producer(sqsAsyncClient, queueUrl,
messageSizeByte,
            producedCount, stop);
    } else {
        producers[i] = new BatchProducer(batchManager, queueUrl,
batchSize,
            messageSizeByte, producedCount, stop);
    }
    producers[i].start();
}

// Start the consumers.
final AtomicInteger consumedCount = new AtomicInteger();
final Thread[] consumers = new Thread[consumerCount];
for (int i = 0; i < consumerCount; i++) {
    if (batchSize == 1) {
        consumers[i] = new Consumer(sqsAsyncClient, queueUrl,
consumedCount, stop);
    } else {
```

```

        consumers[i] = new BatchConsumer(batchManager, queueUrl,
batchSize,
        consumedCount, stop);
    }
    consumers[i].start();
}

// Start the monitor thread.
final Thread monitor = new Monitor(producedCount, consumedCount, stop);
monitor.start();

// Wait for the specified amount of time then stop.
Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runTimeMinutes,
    MAX_RUNTIME_MINUTES)));
stop.set(true);

// Join all threads.
for (int i = 0; i < producerCount; i++) {
    producers[i].join();
}

for (int i = 0; i < consumerCount; i++) {
    consumers[i].join();
}

monitor.interrupt();
monitor.join();

// Close resources
batchManager.close();
sqsAsyncClient.close();
}

/**
 * Creates a random string of approximately the specified size in bytes.
 *
 * @param sizeByte the target size in bytes for the generated string
 * @return a random string encoded in base-32
 */
private static String makeRandomString(int sizeByte) {
    final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
    new Random().nextBytes(bs);
    bs[0] = (byte) ((bs[0] | 64) & 127);
    return new BigInteger(bs).toString(32);
}

```

```

    }

    /**
     * Sends messages individually using SqsAsyncClient for baseline performance
     measurement.
     *
     * This producer demonstrates traditional single-message operations without
     batching.
     * Each sendMessage() call results in a separate API request to SQS,
     providing
     * a performance baseline for comparison with the batch operations.
     *
     * The sendMessage() method signature is identical to
     SqsAsyncBatchManager.sendMessage(),
     * showing how the high-level batching library maintains API compatibility
     while
     * adding automatic optimization behind the scenes.
     */
    private static class Producer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for sending messages
         * @param queueUrl the URL of the target queue
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        Producer(SqsAsyncClient sqsAsyncClient, String queueUrl, int
        messageSizeByte,
            AtomicInteger producedCount, AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }
    }

```

```

    /**
     * Continuously sends messages until the stop flag is set.
     *
     * Uses SqsAsyncClient.sendMessage() directly, resulting in one API call
    per message.
     * This approach provides baseline performance metrics for comparison
    with batching.
     * Each call blocks until the individual message is sent, demonstrating
    traditional
     * one-request-per-operation behavior.
     */
    public void run() {
        try {
            while (!stop.get()) {
                sqsAsyncClient.sendMessage(SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(theMessage)
                    .build()).join();
                producedCount.incrementAndGet();
            }
        } catch (SdkException | java.util.concurrent.CompletionException e) {
            // Handle both SdkException and CompletionException from async
operations.

            // If this unlikely condition occurs, stop.
            log.error("Producer: " + e.getMessage());
            System.exit(1);
        }
    }
}

/**
 * Sends messages using SqsAsyncBatchManager for automatic request batching
and optimization.
 *
 * This producer demonstrates the AWS SDK for Java 2.x high-level batching
library.
 * The SqsAsyncBatchManager automatically buffers individual sendMessage()
calls and
 * sends them as batches when thresholds are reached:
 * - batchSize: Maximum 10 messages per batch (default)
 * - sendRequestFrequency: 200ms timeout before sending partial batches
(default)
 *
 * Key advantages of the batching approach:

```

```

    * - Identical API: batchManager.sendMessage() has the same signature as
    sqsAsyncClient.sendMessage()
    * - Automatic optimization: No code changes needed to benefit from batching
    * - Transparent buffering: The SDK handles batching logic internally
    * - Reduced API calls: Multiple messages sent in single batch requests
    * - Lower costs: Fewer API calls result in reduced SQS charges
    * - Higher throughput: Batch operations process significantly more messages
    per second
    */
    private static class BatchProducer extends Thread {
        final SqsAsyncBatchManager batchManager;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for batch operations.
         *
         * @param batchManager the batch manager for efficient message sending
         * @param queueUrl the URL of the target queue
         * @param batchSize the number of messages to send per batch
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        BatchProducer(SqsAsyncBatchManager batchManager, String queueUrl, int
        batchSize,
                        int messageSizeByte, AtomicInteger producedCount,
                        AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends batches of messages using the high-level batching
         library.
         *

```

```

        * Notice how batchManager.sendMessage() uses the exact same method
signature
        * and request builder pattern as SqsAsyncClient.sendMessage(). This
demonstrates
        * the drop-in replacement capability of the SqsAsyncBatchManager.
        *
        * The SDK automatically:
        * - Buffers individual sendMessage() calls internally
        * - Groups them into batch requests when thresholds are met
        * - Sends SendMessageBatchRequest operations to SQS
        * - Returns individual CompletableFuture responses for each message
        *
        * This transparent batching provides significant performance
improvements
        * without requiring changes to application logic or error handling
patterns.
    */
    public void run() {
        try {
            while (!stop.get()) {
                // Send multiple messages using the high-level batch manager.
                // Each batchManager.sendMessage() call uses identical syntax
to
                // sqsAsyncClient.sendMessage(), demonstrating API
compatibility.
                // The SDK automatically buffers these calls and sends them
as
                // batch operations when maxBatchSize (10) or
sendRequestFrequency (200ms)
                // thresholds are reached, significantly improving
throughput.
                for (int i = 0; i < batchSize; i++) {
                    CompletableFuture<SendMessageResponse> future =
batchManager.sendMessage(
                        SendMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .messageBody(theMessage)
                            .build());

                    // Handle the response asynchronously
                    future.whenComplete((response, throwable) -> {
                        if (throwable == null) {
                            producedCount.incrementAndGet();

```



```

        } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                    !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
            log.error("BatchProducer: Failed to send
message", throwable);
        }
        // Ignore CancellationException and executor shutdown
errors - expected during shutdown
    });
}

    // Small delay to allow batching to occur
    Thread.sleep(10);
}
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    log.error("BatchProducer interrupted: " + e.getMessage());
} catch (SdkException | java.util.concurrent.CompletionException e) {
    log.error("BatchProducer: " + e.getMessage());
    System.exit(1);
}
}
}

/**
 * Receives and deletes messages individually using SqsAsyncClient for
baseline measurement.
 *
 * This consumer demonstrates traditional single-message operations without
batching.
 * Each receiveMessage() and deleteMessage() call results in separate API
requests,
 * providing a performance baseline for comparison with batch operations.
 *
 * The method signatures are identical to SqsAsyncBatchManager methods:
 * - receiveMessage() matches batchManager.receiveMessage()
 * - deleteMessage() matches batchManager.deleteMessage()
 *
 * This API consistency allows easy migration to the high-level batching
library.
 */
private static class Consumer extends Thread {
    final SqsAsyncClient sqsAsyncClient;

```

```
final String queueUrl;
final AtomicInteger consumedCount;
final AtomicBoolean stop;

/**
 * Creates a consumer thread for single-message operations.
 *
 * @param sqsAsyncClient the SQS client for receiving messages
 * @param queueUrl the URL of the source queue
 * @param consumedCount shared counter for tracking processed messages
 * @param stop shared flag to signal thread termination
 */
Consumer(SqsAsyncClient sqsAsyncClient, String queueUrl, AtomicInteger
consumedCount,
        AtomicBoolean stop) {
    this.sqsAsyncClient = sqsAsyncClient;
    this.queueUrl = queueUrl;
    this.consumedCount = consumedCount;
    this.stop = stop;
}

/**
 * Continuously receives and deletes messages using traditional single-
request operations.
 *
 * Uses SqsAsyncClient methods directly:
 * - receiveMessage(): One API call per receive operation
 * - deleteMessage(): One API call per delete operation
 *
 * This approach demonstrates the baseline performance without batching
optimization.
 * Compare these method calls with the identical signatures used in
BatchConsumer
 * to see how the high-level batching library maintains API
compatibility.
 */
public void run() {
    try {
        while (!stop.get()) {
            try {
                final ReceiveMessageResponse result =
sqsAsyncClient.receiveMessage(
                    ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
```

```

        .build()).join();

        if (!result.messages().isEmpty()) {
            final Message m = result.messages().get(0);
            // Note: deleteMessage() signature identical to
batchManager.deleteMessage()

sqsAsyncClient.deleteMessage(DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .receiptHandle(m.receiptHandle())
                    .build()).join();
            consumedCount.incrementAndGet();
        }
    } catch (SdkException |
java.util.concurrent.CompletionException e) {
        log.error(e.getMessage());
    }
}
} catch (SdkException | java.util.concurrent.CompletionException e) {
    // Handle both SdkException and CompletionException from async
operations.
    // If this unlikely condition occurs, stop.
    log.error("Consumer: " + e.getMessage());
    System.exit(1);
}
}
}

/**
 * Receives and deletes messages using SqsAsyncBatchManager for automatic
optimization.
 *
 * This consumer demonstrates the AWS SDK for Java 2.x high-level batching
library
 * for message consumption. The SqsAsyncBatchManager provides two key
optimizations:
 *
 * 1. Receive optimization: Maintains an internal buffer of messages fetched
in the
 * background, so receiveMessage() calls return immediately from the
buffer
 * 2. Delete batching: Automatically buffers deleteMessage() calls and sends
them
 * as DeleteMessageBatchRequest operations when thresholds are reached

```

```

*
* Key features:
* - Identical API: receiveMessage() and deleteMessage() have the same
signatures
*   as SqsAsyncClient methods, making this a true drop-in replacement
* - Background fetching: The batch manager continuously fetches messages to
keep
*   the internal buffer populated, reducing receive latency
* - Automatic delete batching: Individual deleteMessage() calls are buffered
and
*   sent as batch operations (up to 10 per batch, 200ms frequency)
* - Transparent optimization: No application logic changes needed to benefit
*
* Performance benefits:
* - Reduced API calls through automatic batching of delete operations
* - Lower latency for receives due to background message buffering
* - Higher overall throughput with fewer network round trips
*/
private static class BatchConsumer extends Thread {
    final SqsAsyncBatchManager batchManager;
    final String queueUrl;
    final int batchSize;
    final AtomicInteger consumedCount;
    final AtomicBoolean stop;

    /**
     * Creates a consumer thread for batch operations.
     *
     * @param batchManager the batch manager for efficient message processing
     * @param queueUrl the URL of the source queue
     * @param batchSize the maximum number of messages to receive per batch
     * @param consumedCount shared counter for tracking processed messages
     * @param stop shared flag to signal thread termination
     */
    BatchConsumer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                  AtomicInteger consumedCount, AtomicBoolean stop) {
        this.batchManager = batchManager;
        this.queueUrl = queueUrl;
        this.batchSize = batchSize;
        this.consumedCount = consumedCount;
        this.stop = stop;
    }
}

```

```

/**
 * Continuously receives and deletes messages using the high-level
 batching library.
 *
 * Demonstrates the key advantage of SqsAsyncBatchManager: identical
 method signatures
 * with automatic optimization. Notice how:
 *
 * - batchManager.receiveMessage() uses the same syntax as
 sqsAsyncClient.receiveMessage()
 * - batchManager.deleteMessage() uses the same syntax as
 sqsAsyncClient.deleteMessage()
 *
 * Behind the scenes, the batch manager:
 * 1. Maintains an internal message buffer populated by background
 fetching
 * 2. Returns messages immediately from the buffer (reduced latency)
 * 3. Automatically batches deleteMessage() calls into
 DeleteMessageBatchRequest operations
 * 4. Sends batch deletes when batchSize (10) or sendRequestFrequency
 (200ms) is reached
 *
 * This provides significant performance improvements with zero code
 changes
 * compared to traditional SqsAsyncClient usage patterns.
 */
public void run() {
    try {
        while (!stop.get()) {
            // Receive messages using the high-level batch manager.
            // This call uses identical syntax to
 sqsAsyncClient.receiveMessage()
            // but benefits from internal message buffering for improved
 performance.

            final ReceiveMessageResponse result =
 batchManager.receiveMessage(
                ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .numberOfMessages(Math.min(batchSize, 10))
                    .build()).join();

            if (!result.messages().isEmpty()) {
                final List<Message> messages = result.messages();

```

```

        // Delete messages using the batch manager.
        // Each deleteMessage() call uses identical syntax to

SqsAsyncClient

        // but the SDK automatically buffers these calls and
sends them

        // as DeleteMessageBatchRequest operations for optimal
performance.

        for (Message message : messages) {
            CompletableFuture<DeleteMessageResponse> future =
batchManager.deleteMessage(
                DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)

                .receiptHandle(message.receiptHandle())

                    .build());

            future.whenComplete((response, throwable) -> {
                if (throwable == null) {
                    consumedCount.incrementAndGet();
                } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                    !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                    log.error("BatchConsumer: Failed to delete
message", throwable);
                }
                // Ignore CancellationException and executor
shutdown errors - expected during shutdown
            });
        }
    }

    // Small delay to prevent tight polling
    Thread.sleep(10);
}
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    log.error("BatchConsumer interrupted: " + e.getMessage());
} catch (SdkException | java.util.concurrent.CompletionException e) {
    // Handle both SdkException and CompletionException from async
operations.

    // If this unlikely condition occurs, stop.
    log.error("BatchConsumer: " + e.getMessage());
    System.exit(1);
}

```

```

    }
}

/**
 * Displays real-time throughput statistics every second.
 *
 * This thread logs the current count of produced and consumed messages
 * to help you monitor the performance comparison.
 */
private static class Monitor extends Thread {
    private final AtomicInteger producedCount;
    private final AtomicInteger consumedCount;
    private final AtomicBoolean stop;

    /**
     * Creates a monitoring thread that displays throughput statistics.
     *
     * @param producedCount shared counter for messages sent
     * @param consumedCount shared counter for messages processed
     * @param stop shared flag to signal thread termination
     */
    Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
            AtomicBoolean stop) {
        this.producedCount = producedCount;
        this.consumedCount = consumedCount;
        this.stop = stop;
    }

    /**
     * Logs throughput statistics every second until stopped.
     *
     * Displays the current count of produced and consumed messages
     * to help monitor the performance comparison between batching
     strategies.
     */
    public void run() {
        try {
            while (!stop.get()) {
                Thread.sleep(1000);
                log.info("produced messages = " + producedCount.get()
                        + ", consumed messages = " + consumedCount.get());
            }
        } catch (InterruptedException e) {

```

```
        // Allow the thread to exit.  
    }  
}  
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
 - [CreateQueue](#)
 - [DeleteMessage](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)
 - [SendMessageBatch](#)

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Criar funções para encapsular funções de mensagem do Amazon SQS.

```
import logging  
import sys  
  
import boto3  
from botocore.exceptions import ClientError  
  
import queue_wrapper  
  
logger = logging.getLogger(__name__)  
sqs = boto3.resource("sqs")
```



```
def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not
    sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and
    failed
           messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
            for ind, msg in enumerate(messages)
        ]
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
```

```

        return response

def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual
    number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning.
    When
                       this number is greater than zero, long polling is used.
    This
                       can result in reduced costs and fewer false empty
    responses.
    :return: The list of Message objects received. These each contain the body
            of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return messages

def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.

```

```

: return: The response from SQS that contains the list of successful and
failed
        message deletions.
"""
try:
    entries = [
        {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
        for ind, msg in enumerate(messages)
    ]
    response = queue.delete_messages(Entries=entries)
    if "Successful" in response:
        for msg_meta in response["Successful"]:
            logger.info("Deleted %s",
messages[int(msg_meta["Id"])]receipt_handle)
    if "Failed" in response:
        for msg_meta in response["Failed"]:
            logger.warning(
                "Could not delete %s",
messages[int(msg_meta["Id"])]receipt_handle
            )
except ClientError:
    logger.exception("Couldn't delete messages from queue %s", queue)
else:
    return response

```

Use as funções de wrapper para enviar e receber mensagens em lotes.

```

def usage_demo():
    """
    Shows how to:
    * Read the lines from this Python file and send the lines in
      batches of 10 as messages to a queue.
    * Receive the messages in batches until the queue is empty.
    * Reassemble the lines of the file and verify they match the original file.
    """

    def pack_message(msg_path, msg_body, msg_line):
        return {
            "body": msg_body,
            "attributes": {

```

```
        "path": {"StringValue": msg_path, "DataType": "String"},
        "line": {"StringValue": str(msg_line), "DataType": "String"},
    },
}

def unpack_message(msg):
    return (
        msg.message_attributes["path"]["StringValue"],
        msg.body,
        int(msg.message_attributes["line"]["StringValue"]),
    )

print("-" * 88)
print("Welcome to the Amazon Simple Queue Service (Amazon SQS) demo!")
print("-" * 88)

queue = queue_wrapper.create_queue("sqs-usage-demo-message-wrapper")

with open(__file__) as file:
    lines = file.readlines()

line = 0
batch_size = 10
received_lines = [None] * len(lines)
print(f"Sending file lines in batches of {batch_size} as messages.")
while line < len(lines):
    messages = [
        pack_message(__file__, lines[index], index)
        for index in range(line, min(line + batch_size, len(lines)))
    ]
    line = line + batch_size
    send_messages(queue, messages)
    print(".", end="")
    sys.stdout.flush()
print(f"Done. Sent {len(lines) - 1} messages.")

print(f"Receiving, handling, and deleting messages in batches of
{batch_size}.")
more_messages = True
while more_messages:
    received_messages = receive_messages(queue, batch_size, 2)
    print(".", end="")
    sys.stdout.flush()
    for message in received_messages:
```

```
        path, body, line = unpack_message(message)
        received_lines[line] = body
    if received_messages:
        delete_messages(queue, received_messages)
    else:
        more_messages = False
    print("Done.")

    if all([lines[index] == received_lines[index] for index in
            range(len(lines))]):
        print(f"Successfully reassembled all file lines!")
    else:
        print(f"Uh oh, some lines were missed!")

    queue.delete()

    print("Thanks for watching!")
    print("-" * 88)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
 - [CreateQueue](#)
 - [DeleteMessage](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)
 - [SendMessageBatch](#)

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use o AWS Message Processing Framework para.NET para publicar e receber mensagens do Amazon SQS

O exemplo de código a seguir mostra como criar aplicativos que publicam e recebem mensagens do Amazon SQS usando o AWS Message Processing Framework para.NET.

.NET

SDK para .NET

Fornece um tutorial para o AWS Message Processing Framework para.NET. O tutorial cria uma aplicação web que permite ao usuário publicar uma mensagem do Amazon SQS e uma aplicação de linha de comando que recebe a mensagem.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o [tutorial completo](#) no Guia do AWS SDK para .NET desenvolvedor e o exemplo em [GitHub](#).

Serviços usados neste exemplo

- Amazon SQS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar a Biblioteca de Mensagens Java do Amazon SQS para trabalhar com a interface do Serviço de Mensagens Java (JMS) para o Amazon SQS

O exemplo de código a seguir mostra como usar a Biblioteca de Mensagens Java do Amazon SQS para trabalhar com a interface do JMS.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Os exemplos a seguir funcionam com filas padrão do Amazon SQS e incluem:

- Envio de uma mensagem de texto.
- Recebimento de mensagens de forma síncrona.
- Recebimento de mensagens de forma assíncrona.
- Recebimento de mensagens usando o modo `CLIENT_ACKNOWLEDGE`.
- Recebimento de mensagens usando o modo `UNORDERED_ACKNOWLEDGE`.
- Uso do Spring para injetar dependências.
- Uma classe de utilitário que fornece métodos comuns usados pelos outros exemplos.

Consulte mais informações sobre como usar o JMS com o Amazon SQS no [Guia do desenvolvedor do Amazon SQS](#).

Envio de uma mensagem de texto.

```
/**
 * This method establishes a connection to a standard Amazon SQS queue using
the Amazon SQS
 * Java Messaging Library and sends text messages to it. It uses JMS (Java
Message Service) API
 * with automatic acknowledgment mode to ensure reliable message delivery,
and automatically
 * manages all messaging resources.
 *
 * @throws JMSEException If there is a problem connecting to or sending
messages to the queue
 */
public static void doSendTextMessage() throws JMSEException {
    // Create a connection factory.
    SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
        new ProviderConfiguration(),
        SqsClient.create()
    );

    // Create the connection in a try-with-resources statement so that it's
closed automatically.
    try (SQSConnection connection = connectionFactory.createConnection()) {

        // Create the queue if needed.
        SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
    }
```

```

        // Create a session that uses the JMS auto-acknowledge mode.
        Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
        MessageProducer producer =
session.createProducer(session.createQueue(Queue.NAME));

        createAndSendMessage(session, producer);
    } // The connection closes automatically. This also closes the session.
    LOG.info("Connection closed");
}

/**
 * This method reads text input from the keyboard and sends each line as a
 * separate message
 * to a standard Amazon SQS queue using the Amazon SQS Java Messaging
 * Library. It continues
 * to accept input until the user enters an empty line, using JMS (Java
 * Message Service) API to
 * handle the message delivery.
 *
 * @param session The JMS session used to create messages
 * @param producer The JMS message producer used to send messages to the
 * queue
 */
private static void createAndSendMessage(Session session, MessageProducer
producer) {
    BufferedReader inputReader = new BufferedReader(
        new InputStreamReader(System.in, Charset.defaultCharset()));

    try {
        String input;
        while (true) {
            LOG.info("Enter message to send (leave empty to exit): ");
            input = inputReader.readLine();
            if (input == null || input.isEmpty()) break;

            TextMessage message = session.createTextMessage(input);
            producer.send(message);
            LOG.info("Send message {}", message.getMessageID());
        }
    } catch (EOFException e) {
        // Just return on EOF
    } catch (IOException e) {
        LOG.error("Failed reading input: {}", e.getMessage(), e);
    }
}

```



```
    } catch (JMSEException e) {  
        LOGGER.error("Failed sending message: {}", e.getMessage(), e);  
    }  
}
```

Recebimento de mensagens de forma síncrona.

```
/**  
 * This method receives messages from a standard Amazon SQS queue using the  
Amazon SQS Java  
 * Messaging Library. It creates a connection to the queue using JMS (Java  
Message Service),  
 * waits for messages to arrive, and processes them one at a time. The method  
handles all  
 * necessary setup and cleanup of messaging resources.  
 *  
 * @throws JMSEException If there is a problem connecting to or receiving  
messages from the queue  
 */  
public static void doReceiveMessageSync() throws JMSEException {  
    // Create a connection factory.  
    SQSConnectionFactory connectionFactory = new SQSConnectionFactory(  
        new ProviderConfiguration(),  
        SqsClient.create()  
    );  
  
    // Create a connection.  
    try (SQSConnection connection = connectionFactory.createConnection() ) {  
  
        // Create the queue if needed.  
        SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,  
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);  
  
        // Create a session.  
        Session session = connection.createSession(false,  
Session.CLIENT_ACKNOWLEDGE);  
        MessageConsumer consumer =  
session.createConsumer(session.createQueue(QUEUE_NAME));  
  
        connection.start();  
  
        receiveMessages(consumer);  
    }  
}
```

```

    } // The connection closes automatically. This also closes the session.
    LOGGER.info("Connection closed");
}

/**
 * This method continuously checks for new messages from a standard Amazon
 * SQS queue using
 * the Amazon SQS Java Messaging Library. It waits up to 20 seconds for each
 * message, processes
 * it using JMS (Java Message Service), and confirms receipt. The method
 * stops checking for
 * messages after 20 seconds of no activity.
 *
 * @param consumer The JMS message consumer that receives messages from the
 * queue
 */
private static void receiveMessages(MessageConsumer consumer) {
    try {
        while (true) {
            LOGGER.info("Waiting for messages...");
            // Wait 1 minute for a message
            Message message =
consumer.receive(Duration.ofSeconds(20).toMillis());
            if (message == null) {
                LOGGER.info("Shutting down after 20 seconds of silence.");
                break;
            }
            SqsJmsExampleUtils.handleMessage(message);
            message.acknowledge();
            LOGGER.info("Acknowledged message {}",
message.getJMSMessageID());
        }
    } catch (JMSEException e) {
        LOGGER.error("Error receiving from SQS: {}", e.getMessage(), e);
    }
}

```

Recebimento de mensagens de forma assíncrona.

```

/**
 * This method sets up automatic message handling for a standard Amazon SQS
 * queue using the

```

```

    * Amazon SQS Java Messaging Library. It creates a listener that processes
    messages as soon
    * as they arrive using JMS (Java Message Service), runs for 5 seconds, then
    cleans up all
    * messaging resources.
    *
    * @throws JMSEException If there is a problem connecting to or receiving
    messages from the queue
    */
    public static void doReceiveMessageAsync() throws JMSEException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
            new ProviderConfiguration(),
            SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
            SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session.
            Session session = connection.createSession(false,
            Session.CLIENT_ACKNOWLEDGE);

            try {
                // Create a consumer for the queue.
                MessageConsumer consumer =
                session.createConsumer(session.createQueue(QUEUE_NAME));
                // Provide an implementation of the MessageListener interface,
                which has a single 'onMessage' method.
                // We use a lambda expression for the implementation.
                consumer.setMessageListener(message -> {
                    try {
                        SqsJmsExampleUtils.handleMessage(message);
                        message.acknowledge();
                    } catch (JMSEException e) {
                        LOGGER.error("Error processing message: {}",
                        e.getMessage());
                    }
                });
            }
        }
        // Start receiving incoming messages.
    }

```

```

        connection.start();
        LOGGER.info("Waiting for messages...");
    } catch (JMSEException e) {
        throw new RuntimeException(e);
    }
    try {
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
} // The connection closes automatically. This also closes the session.
LOGGER.info( "Connection closed" );
}

```

Recebimento de mensagens usando o modo CLIENT_ACKNOWLEDGE.

```

/**
 * This method demonstrates how message acknowledgment affects message
 * processing in a standard
 * Amazon SQS queue using the Amazon SQS Java Messaging Library. It sends
 * messages to the queue,
 * then shows how JMS (Java Message Service) client acknowledgment mode
 * handles both explicit
 * and implicit message confirmations, including how acknowledging one
 * message can automatically
 * acknowledge previous messages.
 *
 * @throws JMSEException If there is a problem with the messaging operations
 */
public static void doReceiveMessagesSyncClientAcknowledge() throws
JMSEException {
    // Create a connection factory.
    SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
        new ProviderConfiguration(),
        SqsClient.create()
    );

    // Create the connection in a try-with-resources statement so that it's
    // closed automatically.
    try (SQSConnection connection = connectionFactory.createConnection() ) {

        // Create the queue if needed.
    }
}

```

```
SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

// Create a session with client acknowledge mode.
Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

// Create a producer and consumer.
MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

// Open the connection.
connection.start();

// Send two text messages.
sendMessage(producer, session, "Message 1");
sendMessage(producer, session, "Message 2");

// Receive a message and don't acknowledge it.
receiveMessage(consumer, false);

// Receive another message and acknowledge it.
receiveMessage(consumer, true);

// Wait for the visibility time out, so that unacknowledged messages
reappear in the queue,
LOGGER.info("Waiting for visibility timeout...");
try {
    Thread.sleep(TIME_OUT_MILLIS);
} catch (InterruptedException e) {
    LOGGER.error("Interrupted while waiting for visibility timeout",
e);

    Thread.currentThread().interrupt();
    throw new RuntimeException("Processing interrupted", e);
}

/* We will attempt to receive another message, but none will be
available. This is because in
    CLIENT_ACKNOWLEDGE mode, when we acknowledged the second message,
all previous messages were
    automatically acknowledged as well. Therefore, although we never
directly acknowledged the first
```

```

        message, it was implicitly acknowledged when we confirmed the
second one. */
        receiveMessage(consumer, true);
    } // The connection closes automatically. This also closes the session.
    LOGGER.info("Connection closed.");
}

/**
 * Sends a text message using the specified JMS MessageProducer and Session.
 *
 * @param producer    The JMS MessageProducer used to send the message
 * @param session      The JMS Session used to create the text message
 * @param messageText  The text content to be sent in the message
 * @throws JMSEException If there is an error creating or sending the message
 */
private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSEException {
    // Create a text message and send it.
    producer.send(session.createTextMessage(messageText));
}

/**
 * Receives and processes a message from a JMS queue using the specified
consumer.
 * The method waits for a message until the configured timeout period is
reached.
 * If a message is received, it is logged and optionally acknowledged based
on the
 * acknowledge parameter.
 *
 * @param consumer    The JMS MessageConsumer used to receive messages from
the queue
 * @param acknowledge Boolean flag indicating whether to acknowledge the
message.
 *
 *                If true, the message will be acknowledged after
processing
 * @throws JMSEException If there is an error receiving, processing, or
acknowledging the message
 */
private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSEException {
    // Receive a message.

```

```

        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object
            and print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
            message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
}

```

Recebimento de mensagens usando o modo UNORDERED_ACKNOWLEDGE.

```

/**
 * Demonstrates message acknowledgment behavior in UNORDERED_ACKNOWLEDGE mode
 * with Amazon SQS JMS.
 * In this mode, each message must be explicitly acknowledged regardless of
 * receive order.
 * Unacknowledged messages return to the queue after the visibility timeout
 * expires,
 * unlike CLIENT_ACKNOWLEDGE mode where acknowledging one message
 * acknowledges all previous messages.
 *
 * @throws JMSEException      If a JMS-related error occurs during message
 * operations
 */
public static void doReceiveMessagesUnorderedAcknowledge() throws
JMSEException {
    // Create a connection factory.
    SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
        new ProviderConfiguration(),
        SqsClient.create()
    );

    // Create the connection in a try-with-resources statement so that it's
    closed automatically.
    try( SQSConnection connection = connectionFactory.createConnection() ) {

```

```
// Create the queue if needed.
SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

// Create a session with unordered acknowledge mode.
Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

// Create the producer and consumer.
MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

// Open a connection.
connection.start();

// Send two text messages.
sendMessage(producer, session, "Message 1");
sendMessage(producer, session, "Message 2");

// Receive a message and don't acknowledge it.
receiveMessage(consumer, false);

// Receive another message and acknowledge it.
receiveMessage(consumer, true);

// Wait for the visibility time out, so that unacknowledged messages
reappear in the queue.
LOGGER.info("Waiting for visibility timeout...");
try {
    Thread.sleep(TIME_OUT_MILLIS);
} catch (InterruptedException e) {
    LOGGER.error("Interrupted while waiting for visibility timeout",
e);

    Thread.currentThread().interrupt();
    throw new RuntimeException("Processing interrupted", e);
}

/* We will attempt to receive another message, and we'll get the
first message again. This occurs
    because in UNORDERED_ACKNOWLEDGE mode, each message requires its
own separate acknowledgment.
```



```

        Since we only acknowledged the second message, the first message
        remains in the queue for
            redelivery. */
        receiveMessage(consumer, true);

        LOGGER.info("Connection closed.");
    } // The connection closes automatically. This also closes the session.
}

/**
 * Sends a text message to an Amazon SQS queue using JMS.
 *
 * @param producer    The JMS MessageProducer for the queue
 * @param session      The JMS Session for message creation
 * @param messageText  The message content
 * @throws JMSEException If message creation or sending fails
 */
private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSEException {
    // Create a text message and send it.
    producer.send(session.createTextMessage(messageText));
}

/**
 * Synchronously receives a message from an Amazon SQS queue using the JMS
API
 * with an acknowledgment parameter.
 *
 * @param consumer    The JMS MessageConsumer for the queue
 * @param acknowledge If true, acknowledges the message after receipt
 * @throws JMSEException If message reception or acknowledgment fails
 */
private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSEException {
    // Receive a message.
    Message message = consumer.receive(TIME_OUT_MILLIS);

    if (message == null) {
        LOGGER.info("Queue is empty!");
    } else {
        // Since this queue has only text messages, cast the message object
        and print the text.
        LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);
    }
}

```

```
        // Acknowledge the message if asked.  
        if (acknowledge) message.acknowledge();  
    }  
}
```

Uso do Spring para injetar dependências.

```
package com.example.sqs.jms.spring;  
  
import com.amazon.sqs.javamessaging.SQSConnection;  
import com.example.sqs.jms.SqsJmsExampleUtils;  
import jakarta.jms.*;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.NoSuchBeanDefinitionException;  
import org.springframework.context.support.FileSystemXmlApplicationContext;  
  
import java.io.File;  
import java.net.URL;  
import java.util.concurrent.TimeUnit;  
  
/**  
 * Demonstrates how to send and receive messages using the Amazon SQS Java  
 * Messaging Library  
 * with Spring Framework integration. This example connects to a standard Amazon  
 * SQS message  
 * queue using Spring's dependency injection to configure the connection and  
 * messaging components.  
 * The application uses the JMS (Java Message Service) API to handle message  
 * operations.  
 */  
public class SpringExample {  
    private static final Integer POLLING_SECONDS = 15;  
    private static final String SPRING_XML_CONFIG_FILE =  
    "SpringExampleConfiguration.xml.txt";  
    private static final Logger LOGGER =  
    LoggerFactory.getLogger(SpringExample.class);  
  
    /**  
     * Demonstrates sending and receiving messages through a standard Amazon SQS  
     * message queue
```

```
* using Spring Framework configuration. This method loads connection
settings from an XML file,
* establishes a messaging session using the Amazon SQS Java Messaging
Library, and processes
* messages using JMS (Java Message Service) operations. If the queue doesn't
exist, it will
* be created automatically.
*
* @param args Command line arguments (not used)
*/
public static void main(String[] args) {

    URL resource =
SpringExample.class.getClassLoader().getResource(STRING_XML_CONFIG_FILE);
    File springFile = new File(resource.getFile());
    if (!springFile.exists() || !springFile.canRead()) {
        LOGGER.error("File " + STRING_XML_CONFIG_FILE + " doesn't exist or
isn't readable.");
        System.exit(1);
    }

    try (FileSystemXmlApplicationContext context =
        new FileSystemXmlApplicationContext("file://" +
springFile.getAbsolutePath())) {

        Connection connection;
        try {
            connection = context.getBean(Connection.class);
        } catch (NoSuchBeanDefinitionException e) {
            LOGGER.error("Can't find the JMS connection to use: " +
e.getMessage(), e);
            System.exit(2);
            return;
        }

        String queueName;
        try {
            queueName = context.getBean("queueName", String.class);
        } catch (NoSuchBeanDefinitionException e) {
            LOGGER.error("Can't find the name of the queue to use: " +
e.getMessage(), e);
            System.exit(3);
            return;
        }
    }
```

```

        try {
            if (connection instanceof SQSConnection) {
                SqsJmsExampleUtils.ensureQueueExists((SQSConnection)
connection, queueName, SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
            }
            // Create the JMS session.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

            SqsJmsExampleUtils.sendTextMessage(session, queueName);
            MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));

            receiveMessages(consumer);
        } catch (JMSEException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        }
    } // Spring context autocloses. Managed Spring beans that implement
AutoClosable, such as the
    // 'connection' bean, are also closed.
    LOGGER.info("Context closed");
}

/**
 * Continuously checks for and processes messages from a standard Amazon SQS
message queue
 * using the Amazon SQS Java Messaging Library underlying the JMS API. This
method waits for incoming messages,
 * processes them when they arrive, and acknowledges their receipt using JMS
(Java Message
 * Service) operations. The method will stop checking for messages after 15
seconds of
 * inactivity.
 *
 * @param consumer The JMS message consumer used to receive messages from the
queue
 */
private static void receiveMessages(MessageConsumer consumer) {
    try {
        while (true) {
            LOGGER.info("Waiting for messages...");
            // Wait 15 seconds for a message.

```

```

        Message message =
consumer.receive(TimeUnit.SECONDS.toMillis(POLLING_SECONDS));
        if (message == null) {
            LOGGER.info("Shutting down after {} seconds of silence.",
POLLING_SECONDS);
            break;
        }
        SqsJmsExampleUtils.handleMessage(message);
        message.acknowledge();
        LOGGER.info("Message acknowledged.");
    }
} catch (JMSEException e) {
    LOGGER.error("Error receiving from SQS.", e);
}
}
}

```

Definições de bean do Spring.

```

<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/
schema/beans/spring-beans-3.0.xsd
">
    <!-- Define the AWS Region -->
    <bean id="region" class="software.amazon.awssdk.regions.Region" factory-
method="of">
        <constructor-arg value="us-east-1"/>
    </bean>

    <bean id="credentialsProviderBean"
class="software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider"
        factory-method="create"/>

    <bean id="clientBuilder"
class="software.amazon.awssdk.services.sqs.SqsClient" factory-method="builder"/>

    <bean id="regionSetClientBuilder" factory-bean="clientBuilder" factory-
method="region">

```

```
        <constructor-arg ref="region"/>
    </bean>

    <!-- Configure the Builder with Credentials Provider -->
    <bean id="sqsClient" factory-bean="regionSetClientBuilder" factory-
method="credentialsProvider">
        <constructor-arg ref="credentialsProviderBean"/>
    </bean>

    <bean id="providerConfiguration"
class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="connectionFactory"
class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="providerConfiguration"/>
        <constructor-arg ref="clientBuilder"/>
    </bean>

    <bean id="connection"
        factory-bean="connectionFactory"
        factory-method="createConnection"
        init-method="start"
        destroy-method="close"/>

    <bean id="queueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
    </bean>
</beans>
```

Uma classe de utilitário que fornece métodos comuns usados pelos outros exemplos.

```
package com.example.sqs.jms;

import com.amazon.sqs.javamessaging.AmazonSQSMessagingClientWrapper;
import com.amazon.sqs.javamessaging.ProviderConfiguration;
import com.amazon.sqs.javamessaging.SQSConnection;
import com.amazon.sqs.javamessaging.SQSConnectionFactory;
import jakarta.jms.*;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;

import java.time.Duration;
import java.util.Base64;
import java.util.Map;

/**
 * This utility class provides helper methods for working with Amazon Simple
 * Queue Service (Amazon SQS)
 * through the Java Message Service (JMS) interface. It contains common
 * operations for managing message
 * queues and handling message delivery.
 */
public class SqsJmsExampleUtils {
    private static final Logger LOGGER =
        LoggerFactory.getLogger(SqsJmsExampleUtils.class);
    public static final Long QUEUE_VISIBILITY_TIMEOUT = 5L;

    /**
     * This method verifies that a message queue exists and creates it if
     * necessary. The method checks for
     * an existing queue first to optimize performance.
     *
     * @param connection The active connection to the messaging service
     * @param queueName The name of the queue to verify or create
     * @param visibilityTimeout The duration in seconds that messages will be
     * hidden after being received
     * @throws JMSEException If there is an error accessing or creating the queue
     */
    public static void ensureQueueExists(SQSConnection connection, String
        queueName, Long visibilityTimeout) throws JMSEException {
        AmazonSQSMessagingClientWrapper client =
            connection.getWrappedAmazonSQSClient();

        /* In most cases, you can do this with just a 'createQueue' call, but
        'getQueueUrl'
        (called by 'queueExists') is a faster operation for the common case where
        the queue
        already exists. Also, many users and roles have permission to call
        'getQueueUrl'
    
```

```

        but don't have permission to call 'createQueue'.
    */
    if( !client.queueExists(queueName) ) {
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .attributes(Map.of(QueueAttributeName.VISIBILITY_TIMEOUT,
String.valueOf(visibilityTimeout)))
            .build();
        client.createQueue( createQueueRequest );
    }
}

/**
 * This method sends a simple text message to a specified message queue. It
handles all necessary
 * setup for the message delivery process.
 *
 * @param session The active messaging session used to create and send the
message
 * @param queueName The name of the queue where the message will be sent
 */
public static void sendTextMessage(Session session, String queueName) {
    // Rest of implementation...

    try {
        MessageProducer producer =
session.createProducer( session.createQueue( queueName) );
        Message message = session.createTextMessage("Hello world!");
        producer.send(message);
    } catch (JMSEException e) {
        LOGGER.error( "Error receiving from SQS", e );
    }
}

/**
 * This method processes incoming messages and logs their content based on
the message type.
 * It supports text messages, binary data, and Java objects.
 *
 * @param message The message to be processed and logged
 * @throws JMSEException If there is an error reading the message content
 */
public static void handleMessage(Message message) throws JMSEException {
    // Rest of implementation...

```



```

        LOGGER.info( "Got message {}", message.getJMSMessageID() );
        LOGGER.info( "Content: ");
        if(message instanceof TextMessage txtMessage) {
            LOGGER.info( "\t{}", txtMessage.getText() );
        } else if(message instanceof BytesMessage byteMessage){
            // Assume the length fits in an int - SQS only supports sizes up to
256k so that
            // should be true
            byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
            byteMessage.readBytes(bytes);
            LOGGER.info( "\t{}", Base64.getEncoder().encodeToString( bytes ) );
        } else if( message instanceof ObjectMessage ) {
            ObjectMessage objMessage = (ObjectMessage) message;
            LOGGER.info( "\t{}", objMessage.getObject() );
        }
    }

    /**
     * This method sets up automatic message processing for a specified queue. It
creates a listener
     * that will receive and handle incoming messages without blocking the main
program.
     *
     * @param session The active messaging session
     * @param queueName The name of the queue to monitor
     * @param connection The active connection to the messaging service
     */
    public static void receiveMessagesAsync(Session session, String queueName,
Connection connection) {
        // Rest of implementation...
        try {
            // Create a consumer for the queue.
            MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));
            // Provide an implementation of the MessageListener interface, which
has a single 'onMessage' method.
            // We use a lambda expression for the implementation.
            consumer.setMessageListener(message -> {
                try {
                    SqsJmsExampleUtils.handleMessage(message);
                    message.acknowledge();
                } catch (JMSEException e) {
                    LOGGER.error("Error processing message: {}", e.getMessage());
                }
            });
        }
    }

```

```

        });
        // Start receiving incoming messages.
        connection.start();
    } catch (JMSEException e) {
        throw new RuntimeException(e);
    }
    try {
        Thread.sleep(2000);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
}

/**
 * This method performs cleanup operations after message processing is
 * complete. It receives
 *   * any messages in the specified queue, removes the message queue and closes
 *   all
 *   * active connections to prevent resource leaks.
 *
 *   * @param queueName The name of the queue to be removed
 *   * @param visibilityTimeout The duration in seconds that messages are hidden
 *   after being received
 *   * @throws JMSEException If there is an error during the cleanup process
 */
public static void cleanUpExample(String queueName, Long visibilityTimeout)
throws JMSEException {
    LOGGER.info("Performing cleanup.");

    SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
        new ProviderConfiguration(),
        SqsClient.create()
    );

    try (SQSConnection connection = connectionFactory.createConnection() ) {
        ensureQueueExists(connection, queueName, visibilityTimeout);
        Session session = connection.createSession(false,
            Session.AUTO_ACKNOWLEDGE);

        receiveMessagesAsync(session, queueName, connection);

        SqsClient sqsClient =
            connection.getWrappedAmazonSQSClient().getAmazonSQSClient();
    }
}

```

```

        try {
            String queueUrl = sqsClient.getQueueUrl(b ->
b.queueName(queueName)).queueUrl();
            sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
            LOGGER.info("Queue deleted: {}", queueUrl);
        } catch (SdkException e) {
            LOGGER.error("Error during SQS operations: ", e);
        }
    }
    LOGGER.info("Clean up: Connection closed");
}

/**
 * This method creates a background task that sends multiple messages to a
specified queue
 * after waiting for a set time period. The task operates independently to
ensure efficient
 * message processing without interrupting other operations.
 *
 * @param queueName The name of the queue where messages will be sent
 * @param secondsToWait The number of seconds to wait before sending messages
 * @param numMessages The number of messages to send
 * @param visibilityTimeout The duration in seconds that messages remain
hidden after being received
 * @return A task that can be executed to send the messages
 */
public static Runnable sendAMessageAsync(String queueName, Long
secondsToWait, Integer numMessages, Long visibilityTimeout) {
    return () -> {
        try {
            Thread.sleep(Duration.ofSeconds(secondsToWait).toMillis());
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            throw new RuntimeException(e);
        }
        try {
            SQSConnectionFactory connectionFactory = new
SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
            );
            try (SQSConnection connection =
connectionFactory.createConnection()) {
                ensureQueueExists(connection, queueName, visibilityTimeout);
            }
        }
    }
}

```

```
        Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
        for (int i = 1; i <= numMessages; i++) {
            MessageProducer producer =
session.createProducer(session.createQueue(queueName));
            producer.send(session.createTextMessage("Hello World " +
i + " !"));
        }
    } catch (JMSException e) {
        LOGGER.error(e.getMessage(), e);
        throw new RuntimeException(e);
    }
};
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
 - [CreateQueue](#)
 - [DeleteQueue](#)

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Trabalhe com tags de fila e Amazon SQS usando um SDK AWS

O exemplo de código a seguir mostra como realizar uma operação de marcação com o Amazon SQS.

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

O exemplo a seguir cria tags para uma fila, lista tags e remove uma tag.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ListQueueTagsResponse;
import software.amazon.awssdk.services.sqs.model.QueueDoesNotExistException;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.Map;
import java.util.UUID;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials. For more
 * information, see the <a href="https://docs.aws.amazon.com/sdk-for-java/latest/
 * developer-guide/get-started.html">AWS
 * SDK for Java Developer Guide</a>.
 */
public class TagExamples {
    static final SqsClient sqsClient = SqsClient.create();
    static final String queueName = "TagExamples-queue-" +
        UUID.randomUUID().toString().replace("-", "").substring(0, 20);
    private static final Logger LOGGER =
        LoggerFactory.getLogger(TagExamples.class);

    public static void main(String[] args) {
        final String queueUrl;
        try {
            queueUrl = sqsClient.createQueue(b ->
b.queueName(queueName)).queueUrl();
            LOGGER.info("Queue created. The URL is: {}", queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because queue was not created.");
            throw new RuntimeException(e);
        }
        try {
            addTags(queueUrl);
            listTags(queueUrl);
            removeTags(queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because of an error in a method.");
        }
    }
}
```

```

        } finally {
            try {
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue successfully deleted. Program ending.");
                sqsClient.close();
            } catch (RuntimeException e) {
                LOGGER.error("Program ending.");
            } finally {
                sqsClient.close();
            }
        }
    }

    /** This method demonstrates how to use a Java Map to tag a queue.
     * @param queueUrl The URL of the queue to tag.
     */
    public static void addTags(String queueUrl) {
        // Build a map of the tags.
        final Map<String, String> tagsToAdd = Map.of(
            "Team", "Development",
            "Priority", "Beta",
            "Accounting ID", "456def");

        try {
            // Add tags to the queue using a Consumer<TagQueueRequest.Builder>
            parameter.
                sqsClient.tagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tags(tagsToAdd)
                );
        } catch (QueueDoesNotExistException e) {
            LOGGER.error("Queue does not exist: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }

    /** This method demonstrates how to view the tags for a queue.
     * @param queueUrl The URL of the queue whose tags you want to list.
     */
    public static void listTags(String queueUrl) {
        ListQueueTagsResponse response;
        try {

```

```

        // Call the listQueueTags method with a
        Consumer<ListQueueTagsRequest.Builder> parameter that creates a
        ListQueueTagsRequest.
        response = sqsClient.listQueueTags(b -> b
            .queueUrl(queueUrl));
    } catch (SqsException e) {
        LOGGER.error("Exception thrown: {}", e.getMessage(), e);
        throw new RuntimeException(e);
    }

    // Log the tags.
    response.tags()
        .forEach((k, v) ->
            LOGGER.info("Key: {} -> Value: {}", k, v));
}

/**
 * This method demonstrates how to remove tags from a queue.
 * @param queueUrl The URL of the queue whose tags you want to remove.
 */
public static void removeTags(String queueUrl) {
    try {
        // Call the untagQueue method with a
        Consumer<UntagQueueRequest.Builder> parameter.
        sqsClient.untagQueue(b -> b
            .queueUrl(queueUrl)
            .tagKeys("Accounting ID") // Remove a single tag.
        );
    } catch (SqsException e) {
        LOGGER.error("Exception thrown: {}", e.getMessage(), e);
        throw new RuntimeException(e);
    }
}
}

```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
 - [ListQueueTags](#)
 - [TagQueue](#)
 - [UntagQueue](#)

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de tecnologia sem servidor para o Amazon SQS

Os exemplos de código a seguir mostram como usar o Amazon SQS com AWS SDKs

Exemplos

- [Invocar uma função do Lambda em um trigger do Amazon SQS](#)
- [Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS](#)

Invocar uma função do Lambda em um trigger do Amazon SQS

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do SQS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
into a .NET class.
```



```
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }

        context.Logger.LogInformation("done");
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
        ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
            Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK para Go V2

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
    "fmt"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
    for _, record := range event.Records {
        err := processMessage(record)
        if err != nil {
            return err
        }
    }
    fmt.Println("done")
    return nil
}

func processMessage(record events.SQSMessage) error {
    fmt.Printf("Processed message %s\n", record.Body)
    // TODO: Do interesting work based on the new message
    return nil
}

func main() {
    lambda.Start(handler)
}
```

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

```
}  
}
```

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumindo um evento SQS com o JavaScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: Apache-2.0  
exports.handler = async (event, context) => {  
  for (const message of event.Records) {  
    await processMessageAsync(message);  
  }  
  console.info("done");  
};  
  
async function processMessageAsync(message) {  
  try {  
    console.log(`Processed message ${message.body}`);  
    // TODO: Do interesting work based on the new message  
    await Promise.resolve(1); //Placeholder for actual async work  
  } catch (err) {  
    console.error("An error occurred");  
    throw err;  
  }  
}
```

Consumindo um evento SQS com o TypeScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: Apache-2.0  
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";
```

```

export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}

```

PHP

SDK para PHP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do SQS com o Lambda usando PHP.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;

```

```
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
    event['Records'].each do |message|
        process_message(message)
    end
    puts "done"
end

def process_message(message)
    begin
        puts "Processed message #{message['body']}"
        # TODO: Do interesting work based on the new message
    end
```

```
rescue StandardError => err
  puts "An error occurred"
  raise err
end
end
```

Rust

SDK para Rust

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consoma um evento do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
            record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
        time.
        .without_time()
}
```



```
.init();

run(service_fn(function_handler)).await
}
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

Os exemplos de código a seguir mostram como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

.NET

SDK para .NET

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
// into a .NET class.
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer),
    namespace sqsSample);

public class Function
```

```
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
        ILambdaContext context)
    {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
        List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
                SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
        ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
        context.Logger.LogInformation($"Processed message {message.Body}");
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

Go

SDK para Go V2

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "fmt"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent)
    (map[string]interface{}, error) {
    batchItemFailures := []map[string]interface{}{}

    for _, message := range sqsEvent.Records {
        if len(message.Body) > 0 {
            // Your message processing condition here
            fmt.Printf("Successfully processed message: %s\n", message.Body)
        } else {
            // Message processing failed
            fmt.Printf("Failed to process message %s\n", message.MessageId)
            batchItemFailures = append(batchItemFailures, map[string]interface{}{
                "itemIdentifier": message.MessageId})
        }
    }

    sqsBatchResponse := map[string]interface{}{
        "batchItemFailures": batchItemFailures,
    }
    return sqsBatchResponse, nil
}
```

```
}

func main() {
    lambda.Start(handler)
}
```

Java

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
    SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<SQSBatchResponse.BatchItemFailure>();

        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
            }
        }
    }
}
```

```

        batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(message.getMessageId()));
    }
}
return new SQSBatchResponse(batchItemFailures);
}
}

```

JavaScript

SDK para JavaScript (v3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatando falhas de itens em lote do SQS com o uso do JavaScript Lambda.

```

// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
    console.log(`Processed message: ${record.body}`);
}

```

Relatando falhas de itens em lote do SQS com o uso do TypeScript Lambda.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord }
  from 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
  Promise<SQSBatchResponse> => {
  const batchItemFailures: SQSBatchItemFailure[] = [];

  for (const record of event.Records) {
    try {
      await processMessageAsync(record);
    } catch (error) {
      batchItemFailures.push({ itemIdentifier: record.messageId });
    }
  }

  return {batchItemFailures: batchItemFailures};
};

async function processMessageAsync(record: SQSRecord): Promise<void> {
  if (record.body && record.body.includes("error")) {
    throw new Error('There is an error in the SQS Message.');
```

PHP

SDK para PHP

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}
```

```
$logger = new StderrLogger();  
return new Handler($logger);
```

Python

SDK para Python (Boto3)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
def lambda_handler(event, context):  
    if event:  
        batch_item_failures = []  
        sqs_batch_response = {}  
  
        for record in event["Records"]:  
            try:  
                print(f"Processed message: {record['body']}")  
            except Exception as e:  
                batch_item_failures.append({"itemIdentifier":  
record['messageId']})  
  
        sqs_batch_response["batchItemFailures"] = batch_item_failures  
        return sqs_batch_response
```


Ruby

SDK para Ruby

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
        rescue StandardError => e
          batch_item_failures << {"itemIdIdentifier" => record['messageId']}
        end
      end

      sqs_batch_response["batchItemFailures"] = batch_item_failures
      return sqs_batch_response
    end
  end
end
```

Rust

SDK para Rust

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) ->
    Result<SqsBatchResponse, Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
        }
    }

    Ok(SqsBatchResponse {
        batch_item_failures,
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
```

```
run(service_fn(function_handler)).await  
}
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Usar o Amazon SQS com um SDK da AWS](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Solução de problemas no Amazon SQS

Este tópico fornece orientações para a solução de erros e problemas comuns que você pode encontrar ao usar o console do Amazon SQS, a API do Amazon SQS ou outras ferramentas com o Amazon SQS. Se encontrar um problema que não esteja listado aqui, você poderá usar o botão Feedback desta página para relatá-lo.

Para obter mais orientações sobre solução de problemas e respostas a perguntas comuns de suporte, acesse a [Central de Conhecimento da AWS](#).

Tópicos

- [Solução de problemas de um erro de acesso negado no Amazon SQS](#)
- [Solucionar problemas de erros de API do Amazon SQS](#)
- [Solucionar problemas de fila de mensagens não entregues do Amazon SQS e de redirecionamento de DLQ](#)
- [Solução de problemas de controle de utilização de FIFO no Amazon SQS](#)
- [Solucionar problemas de mensagens não retornadas para uma chamada de API ReceiveMessage do Amazon SQS](#)
- [Solucionar problemas de erros de rede do Amazon SQS](#)
- [Solução de problemas de filas do Amazon Simple Queue Service usando o AWS X-Ray](#)

Solução de problemas de um erro de acesso negado no Amazon SQS

Os tópicos a seguir abordam as causas mais comuns de erros AccessDenied ou AccessDeniedException nas chamadas de API do Amazon SQS. Consulte mais informações sobre como solucionar esses erros em [Como soluciono erros de “AccessDenied” ou “AccessDeniedException” em chamadas de API do Amazon SQS?](#) no Guia do Centro de Conhecimento da AWS.

Exemplos de mensagens de erro:

An error occurred (AccessDenied) when calling the SendMessage operation: Access to the resource https://sqs.us-east-1.amazonaws.com/ is denied.

- OU -

```
An error occurred (KMS.AccessDeniedException) when calling the SendMessage
operation: User: arn:aws:iam::xxxxx:user/xxxx is not authorized to perform:
kms:GenerateDataKey on resource: arn:aws:kms:us-east-1:xxxx:key/xxxx with an
explicit
deny.
```

Política de filas do Amazon SQS e política do IAM

Para verificar se o solicitante tem as permissões adequadas para realizar uma operação do Amazon SQS, faça o seguinte:

- Identifique a entidade principal do IAM que está fazendo a chamada de API do Amazon SQS. Se a entidade principal do IAM for da mesma conta, nem a política de filas do Amazon SQS nem a política do AWS Identity and Access Management (IAM) devem incluir permissões para autorizar explicitamente o acesso à ação.
- Se o a entidade principal for uma entidade do IAM:
 - É possível identificar o perfil ou usuário do IAM no canto superior direito do Console de gerenciamento da AWS ou usando o comando [aws sts get-caller-identity](#).
 - Verifique as políticas do IAM relacionadas ao perfil ou usuário do IAM. É possível usar um dos seguintes métodos:
 - Teste as políticas do IAM com o [simulador de políticas do IAM](#).
 - Analise os diferentes [tipos de política do IAM](#).
 - Se necessário, [edite a política de usuário do IAM](#).
 - Verifique a política de filas e [edite](#), se necessário.
- Se a entidade principal for um serviço da AWS, a política de filas do Amazon SQS deve permitir explicitamente o acesso.
- Se a entidade principal for entre contas, tanto a política de filas do Amazon SQS quanto a política do IAM devem permitir explicitamente o acesso.
- Se a política usar um elemento condicional, verifique se a condição restringe o acesso.

⚠ Important

Uma negação explícita em qualquer política substitui uma permissão explícita. Aqui estão alguns exemplos básicos de [políticas do Amazon SQS](#).

AWS Key Management Service Permissões

Se a fila do Amazon SQS tiver a [criptografia do lado do servidor \(SSE\)](#) ativada com uma AWS KMS key gerenciada pelo cliente, as permissões deverão ser concedidas tanto aos produtores quanto aos consumidores. Para confirmar se uma fila está criptografada, é possível usar o atributo `KmsMasterKeyId` atributo da API [GetQueueAttributes](#) ou do console da fila em Criptografia.

- [Permissões para produtores](#) exigidas:

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "<Key ARN>"
}
```

- [Permissões para consumidores](#) exigidas:

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt"
  ],
  "Resource": "<Key ARN>"
}
```

- Permissões para [acesso entre contas](#) exigidas:

```
{
  "Effect": "Allow",
  "Action": [
    "kms:DescribeKey",
    "kms:Decrypt",

```

```
"kms:ReEncrypt",
"kms:GenerateDataKey"
],
"Resource": "<Key ARN>"
}
```

Escolha uma das seguintes opções para habilitar a criptografia de uma fila do Amazon SQS:

- [SSE-Amazon SQS](#) (chave de criptografia criada e gerenciada pelo serviço Amazon SQS.)
- [Chave padrão gerenciada pela AWS](#) (alias/aws/sqs)
- [Chave gerenciada pelo cliente](#)

No entanto, se você estiver usando uma [chave do KMS](#) gerenciada pela AWS, não poderá modificar a política de chave padrão. Portanto, para conceder acesso a outros serviços e contas cruzadas, use a chave gerenciada pelo cliente. Isso permite que você edite a política de chave.

Política de endpoint da VPC

Se você acessar o [Amazon SQS por meio de um endpoint da Amazon Virtual Private Cloud \(Amazon VPC\)](#), a política de endpoint da VPC do Amazon SQS deverá permitir acesso. É possível criar uma política para endpoints da Amazon VPC para o Amazon SQS, em que você pode especificar o seguinte:

1. A entidade principal que pode executar ações.
2. As ações que podem ser executadas.
3. Os recursos sobre os quais as ações podem ser realizadas.

No exemplo a seguir, a política de endpoint da VPC especifica que o usuário do IAM *MyUser* tem permissão para enviar mensagens para a fila *MyQueue* do Amazon SQS. Outras ações, usuários do IAM e recursos do Amazon SQS têm acesso negado por meio do endpoint da VPC.

```
{
  "Statement": [{
    "Action": ["sqs:SendMessage"],
    "Effect": "Allow",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
    "Principal": {
```

```
    "AWS": "arn:aws:iam:123456789012:user/MyUser"
  }
}]
}
```

Política de controle de serviço da organização

Se a Conta da AWS pertence a uma organização, as políticas do AWS Organizations podem impedir que você acesse as filas do Amazon SQS. Por padrão, as políticas do AWS Organizations não bloqueiam nenhuma solicitação ao Amazon SQS. No entanto, verifique se as políticas do AWS Organizations não foram configuradas para bloquear o acesso às filas do Amazon SQS. Para obter instruções sobre como verificar as políticas do AWS Organizations, consulte [Listar todas as políticas](#) no Guia do usuário do AWS Organizations.

Solucionar problemas de erros de API do Amazon SQS

Os tópicos a seguir abordam os erros mais comuns retornados ao fazer chamadas de API do Amazon SQS e como solucioná-los.

Erro QueueDoesNotExist

Esse erro será retornado quando o serviço Amazon SQS não conseguir encontrar a fila mencionada para a ação do Amazon SQS.

Possíveis causas e mitigações:

- **Região incorreta:** revise a configuração do cliente do Amazon SQS para confirmar se você configurou a região correta no cliente. Quando você não configura uma região no cliente, o SDK ou a AWS CLI escolhe a região no [arquivo de configuração](#) ou na variável de ambiente. Se o SDK não encontrar uma região no arquivo de configuração, ele definirá a região como us-east-1 por padrão.
- **A fila pode ter sido excluída recentemente:** se a fila foi excluída antes da chamada de API ser feita, a chamada da API retornará esse erro. Verifique se há alguma operação de [DeleteQueue](#) no CloudTrail antes do momento do erro.
- **Problemas de permissão:** se o perfil ou usuário solicitante do AWS Identity and Access Management (IAM) não tiver as permissões necessárias, você poderá receber o seguinte erro:

```
The specified queue does not exist or you do not have access to it.
```


Verifique as permissões e faça a chamada de API com as permissões corretas.

Consulte mais detalhes sobre a solução de problemas do erro `QueueDoesNotExist` em [Como soluciono o erro `QueueDoesNotExist` ao fazer chamadas de API para minha fila do Amazon SQS?](#) no Guia do Centro de Conhecimento da AWS.

Erro `InvalidAttributeValue`

Esse erro será retornado ao atualizar a política de recursos de fila do Amazon SQS ou propriedades com uma política ou uma entidade principal incorretas.

Possíveis causas e mitigações:

- Política de recursos inválida: verifique se a política de recursos tem todos os campos obrigatórios. Consulte mais informações [Referência de elemento de política JSON do IAM](#) e [Validação de política do IAM](#). Você também pode usar o [gerador de políticas do IAM](#) para criar e testar uma política de recursos do Amazon SQS. Verifique se a política está no formato JSON.
- Entidade principal inválida: verifique se o elemento `Principal` existe na política de recursos e se o valor é válido. Se o elemento `Principal` da política de recursos do Amazon SQS incluir uma entidade do IAM, certifique-se de que a entidade exista antes de usar a política. O Amazon SQS valida a política de recursos e verifica a entidade do IAM. Se a entidade do IAM não existir, você receberá um erro. Para confirmar as entidades do IAM, use as APIs [GetUser](#) e [GetRole](#).

Consulte mais informações sobre a solução de problemas do erro `InvalidAttributeValue` em [Como soluciono o erro `QueueDoesNotExist` ao fazer chamadas de API para minha fila do Amazon SQS?](#) no Guia do Centro de Conhecimento da AWS.

Erro `ReceiptHandle`

Ao fazer uma chamada de API [DeleteMessage](#), o erro `ReceiptHandleIsInvalid` ou `InvalidParameterValue` poderá ser retornado se o identificador do recibo estiver incorreto ou expirado.

- Erro `ReceiptHandleIsInvalid`: se o identificador do recibo estiver incorreto, você receberá um erro semelhante a este exemplo:

An error occurred (ReceiptHandleIsValid) when calling the DeleteMessage operation:
The input receipt handle <YOUR RECEIPT HANDLE> is not a valid receipt handle.

- Erro InvalidParameterValue: se o identificador do recibo estiver expirado, você receberá um erro semelhante a este exemplo:

An error occurred (InvalidParameterValue) when calling the DeleteMessage operation:
Value <YOUR RECEIPT HANDLE> for parameter ReceiptHandle is invalid. Reason: The receipt handle has expired.

Possíveis causas e mitigações:

O identificador do recibo é criado para cada mensagem recebida e só é válido para o período de tempo limite de visibilidade. Quando o período de tempo limite de visibilidade expira, a mensagem fica visível na fila para os consumidores. Ao receber a mensagem novamente do consumidor, você recebe um novo identificador de recibo. Para evitar erros incorretos ou expirados no identificador de recibo, use o identificador de recibo correto para excluir a mensagem dentro do período de tempo limite de visibilidade da fila do Amazon SQS.

Consulte mais informações sobre como solucionar um erro ReceiptHandle em [Como solucionar os erros “ReceiptHandleIsValid” e “InvalidParameterValue” ao usar a chamada de API DeleteMessage do Amazon SQS?](#) no Guia do Centro de Conhecimento da AWS.

Solucionar problemas de fila de mensagens não entregues do Amazon SQS e de redirecionamento de DLQ

Os tópicos a seguir abordam as causas mais comuns dos problemas de DLQ e redirecionamento de DLQ do Amazon SQS e como solucioná-los. Consulte mais informações em [How do I troubleshoot Amazon SQS DLQ redrive issues?](#) no Guia do Centro de Conhecimento da AWS.

Problemas de DLQ

Saiba mais sobre problemas comuns de DLQ e como resolvê-los.

Visualizar mensagens usando o console pode fazer com que elas sejam movidas para uma fila de mensagens mortas

O Amazon SQS considera a visualização de uma mensagem no console para a política de redirecionamento da fila correspondente. Dessa forma, se você visualizar uma mensagem no console pelo número de vezes especificado na política de redirecionamento da fila correspondente, a mensagem será movida para a fila de mensagens não entregues da fila correspondente.

Para ajustar esse comportamento, você pode executar uma das seguintes ações:

- Aumentar a definição de Maximum Receives da política de redirecionamento da fila correspondente.
- Evitar a visualização de mensagens da fila correspondente no console.

O **NumberOfMessagesSent** e o **NumberOfMessagesReceived** de uma fila de mensagens não entregues não correspondem

Se você enviar uma mensagem para uma fila de mensagens não entregues manualmente, ela será capturada pela métrica [NumberOfMessagesSent](#). No entanto, se uma mensagem for enviada para uma dead-letter queue como resultado de uma falha na tentativa de processamento, ela não será capturada por essa métrica. Assim, é possível que os valores de **NumberOfMessagesSent** e [NumberOfMessagesReceived](#) sejam diferentes.

Criar e configurar redirecionamento de fila de mensagens não entregues

O redirecionamento da fila de mensagens não entregues exige que você defina as [permissões](#) apropriadas para que o Amazon SQS receba mensagens da fila de mensagens não entregues e envie mensagens para a fila de destino. Se você não tiver as permissões corretas, a tarefa de redirecionamento da fila de mensagens não entregues poderá falhar. Você pode exibir o status da tarefa de redirecionamento de mensagens para corrigir os problemas e tentar novamente.

Gerenciamento de falhas de mensagens de fila padrão e FIFO

As [filas padrão](#) continuam a processar mensagens até a expiração do [período de retenção](#). Esse processamento contínuo minimiza as chances de a fila ser bloqueada por mensagens não consumidas. Ter um número grande de mensagens que o consumir falha repetidamente em excluir pode aumentar os custos e colocar uma carga extra no hardware. Para manter os custos baixos, mova mensagens com falha para a fila de mensagens não entregues.

As filas padrão também permitem grande número de mensagens em trânsito. Se a maioria de suas mensagens não puder ser consumida e não for enviada para uma fila de mensagens não entregues, sua taxa de processamento de mensagens válidas poderá diminuir. Para manter a eficiência da fila, garanta que sua aplicação lide com o processamento de mensagens corretamente.

As [filas FIFO](#) fornecem o processamento exatamente uma vez, consumindo mensagens de um grupo de mensagens em sequência. Assim, embora o consumidor possa continuar a recuperar mensagens ordenadas de outro grupo de mensagens, o primeiro grupo permanece indisponível até que a mensagem que está bloqueando a fila seja processada com sucesso ou movida para uma fila de mensagens não entregues.

Além disso, as filas FIFO permitem um número menor de mensagens em trânsito. Para evitar que a fila FIFO seja bloqueada por uma mensagem, garanta que sua aplicação lide com o processamento de mensagens corretamente.

Para obter mais informações, consulte [Cotas de mensagens do Amazon SQS](#) e [Práticas recomendadas do Amazon SQS](#).

Problemas de redirecionamento de DLQ

Saiba mais sobre problemas comuns de redirecionamento de DLQ e como resolvê-los.

Problema de permissão AccessDenied

O erro AccessDenied ocorre quando o redirecionamento de DLQ falha porque a entidade do AWS Identity and Access Management (IAM) não tem as permissões necessárias.

Exemplo de mensagem de erro:

```
Failed to create redrive task. Error code: AccessDenied - Queue Permissions to Redrive.
```

As seguintes permissões de API são necessárias para fazer solicitações de redirecionamento de DLQ:

Como iniciar um redirecionamento de mensagens:

- Permissões da fila de mensagens não entregues:
 - `sqs:StartMessageMoveTask`
 - `sqs:ReceiveMessage`

- `sqs:DeleteMessage`
- `sqs:GetQueueAttributes`
- `kms:Decrypt`: quando a fila de mensagens não entregues ou a fila de origem original são criptografadas.
- Permissões da fila de destino:
 - `sqs:SendMessage`
 - `kms:GenerateDataKey`: quando a fila de destino é criptografada.
 - `kms:Decrypt`: quando a fila de destino é criptografada.

Como cancelar um redirecionamento de mensagem em andamento:

- Permissões da fila de mensagens não entregues:
 - `sqs:CancelMessageMoveTask`
 - `sqs:ReceiveMessage`
 - `sqs:DeleteMessage`
 - `sqs:GetQueueAttributes`
 - `kms:Decrypt`: quando a fila de mensagens não entregues ou a fila de origem original são criptografadas.

Como exibir o status de movimentação de uma mensagem:

- Permissões da fila de mensagens não entregues:
 - `sqs:ListMessageMoveTasks`
 - `sqs:GetQueueAttributes`

Erro **NonExistentQueue**

O erro `NonExistentQueue` ocorre quando a fila de origem do Amazon SQS não existe ou foi excluída. Verifique e redirecione para uma fila do Amazon SQS que está presente.

Exemplo de mensagem de erro:

```
Failed: AWS.SimpleQueueService.NonExistentQueue
```

Erro CouldNotDetermineMessageSource

O erro `CouldNotDetermineMessageSource` ocorre quando você tenta iniciar um redirecionamento de DLQ com os seguintes cenários:

- Uma mensagem do Amazon SQS enviada diretamente para a DLQ com API [SendMessage](#).
- Uma mensagem do tópico do Amazon Simple Notification Service (Amazon SNS) ou da função do AWS Lambda com a DLQ configurada.

Para resolver esse erro, escolha Redirecionar para um destino personalizado ao iniciar o redirecionamento. Depois, insira o ARN da fila do Amazon SQS para mover todas as mensagens da DLQ para a fila de destino.

Exemplo de mensagem de erro:

```
Failed: CouldNotDetermineMessageSource
```

Solução de problemas de controle de utilização de FIFO no Amazon SQS

Por padrão, as filas FIFO comportam 300 transações por segundo, por ação de API para [SendMessage](#), [ReceiveMessage](#) e [DeleteMessage](#). Solicitações acima de 300 TPS receberão o erro `ThrottlingException` mesmo se as mensagens na fila estiverem disponíveis. Para mitigar isso, você pode usar os seguintes métodos:

- [Habilitar throughput alto para filas FIFO no Amazon SQS](#).
- Use as ações em lote `SendMessageBatch`, `DeleteMessageBatch` e `ChangeMessageVisibilityBatch` da API do Amazon SQS para aumentar o limite de TPS de até 3 mil mensagens por segundo por ação da API e reduzir custos. Para a API `ReceiveMessage`, defina o parâmetro `MaxNumberOfMessages` para receber até dez mensagens por transação. Para obter mais informações, consulte [Ações em lote do Amazon SQS](#).
- Para filas FIFO com throughput alto, siga as recomendações para [otimizar a utilização da partição](#). Envie mensagens com os mesmos IDs de grupo de mensagens em lotes. Exclua mensagens ou altere os valores de tempo limite de visibilidade da mensagem em lotes com identificadores de recebimento das mesmas solicitações de API `ReceiveMessage`.

- Aumente o número de valores [MessageGroupId](#) exclusivos. Isso permite uma distribuição uniforme entre partições de fila FIFO. Consulte mais informações em “Using the Amazon SQS message group ID”.

Consulte mais informações em [Por que minha fila FIFO do Amazon SQS não retorna todas as mensagens ou mensagens de outros grupos de mensagens?](#) no Guia do Centro de Conhecimento da AWS.

Solucionar problemas de mensagens não retornadas para uma chamada de API ReceiveMessage do Amazon SQS

Os tópicos a seguir abordam as causas mais comuns pelas quais uma mensagem do Amazon SQS pode não ser retornada aos consumidores e como solucioná-las. Consulte mais informações em [Why can't I receive messages from my Amazon SQS queue?](#) no Guia do Centro de Conhecimento da AWS.

Fila vazia

Para determinar se uma fila está vazia, use a sondagem longa para chamar a API [ReceiveMessage](#). Você também pode usar as métricas `ApproximateNumberOfMessagesVisible`, `ApproximateNumberOfMessagesNotVisible` e `ApproximateNumberOfMessagesDelayed` do CloudWatch. Se todos os valores de métricas são definidos como 0 por vários minutos, a fila é considerada vazia.

Limite de trânsito atingido

Se você usar a [sondagem longa](#) e se o limite de trânsito da fila (por padrão, 120 mil) for atingido, o Amazon SQS não retornará mensagens de erro que [excedam os limites da cota](#).

Atraso de mensagens

Se a fila do Amazon SQS estiver configurada como uma [fila de atraso](#) ou se as mensagens tiverem sido enviadas com [temporizadores de mensagens](#), as mensagens não ficarão visíveis até que o tempo de atraso termine. Para verificar se uma fila está configurada como fila de atraso, use o atributo `DelaySeconds` da API [GetQueueAttributes](#) ou confira no console de fila em Atraso de entrega. Verifique a métrica [ApproximateNumberOfMessagesDelayed](#) do CloudWatch para entender se alguma mensagem está atrasada.

A mensagem está em trânsito

Se um consumidor diferente tiver sondado a mensagem, ela ficará em trânsito ou invisível durante o período de [tempo limite de visibilidade](#). As sondagens adicionais podem retornar um recibo vazio. Verifique a métrica [ApproximateNumberOfMessagesVisible](#) do CloudWatch para entender o número de mensagens que estão disponíveis para serem recebidas. No caso de filas FIFO, se uma mensagem com o ID do grupo de mensagens estiver em trânsito, nenhuma outra mensagem será retornada, a menos que você exclua a mensagem ou que ela se torne visível. Isso ocorre porque a [ordem das mensagens](#) é mantida no nível do grupo de mensagens em uma fila FIFO.

Método de sondagem

Se você estiver usando a [sondagem curta](#), ([WaitTimeSeconds](#) é 0), o Amazon SQS obtém amostras de um subconjunto de seus servidores e retorna mensagens apenas desses servidores. Portanto, você pode não receber as mensagens, mesmo que elas estejam disponíveis para serem recebidas. As solicitações de sondagem subsequentes retornarão as mensagens.

Se você estiver usando uma [sondagem longa](#), o Amazon SQS sondará todos os servidores e enviará uma resposta após coletar pelo menos uma mensagem disponível e até o número máximo especificado. Se o valor de [WaitTimeSeconds](#) de `ReceiveMessage` for muito baixo, talvez você não receba todas as mensagens disponíveis.

Solucionar problemas de erros de rede do Amazon SQS

Os tópicos a seguir abordam as causas mais comuns de problemas de rede no Amazon SQS e como solucioná-los.

ETIMOUT error

O erro ETIMOUT ocorre quando o cliente não consegue estabelecer uma conexão TCP com um endpoint do Amazon SQS.

Solução de problemas:

- Verifique a conexão de rede

Teste sua conexão de rede com o Amazon SQS executando comandos como `telnet`.

Example: `telnet sqs.us-east-1.amazonaws.com 443`

- Verifique as configurações de rede

- Certifique-se de que as regras de firewall, rotas e listas de controle de acesso (ACLs) locais permitam tráfego na porta que você usa.
- As regras de saída do grupo de segurança devem permitir o tráfego para a porta 80 ou 443.
- As regras de saída da ACL de rede devem permitir o tráfego para a porta TCP 80 ou 443.
- As regras de entrada da ACL de rede devem permitir o tráfego nas portas TCP de 1024 a 65535.
- As instâncias do Amazon Elastic Compute Cloud (Amazon EC2) que se conectam à internet pública devem ter [conectividade com a internet](#).
- Endpoints da Amazon Virtual Private Cloud (Amazon VPC)

Se você acessar o Amazon SQS por meio de um endpoint da Amazon VPC, o grupo de segurança dos endpoints deverá permitir tráfego de entrada para o grupo de segurança do cliente na porta 443. A ACL de rede associada à sub-rede do endpoint da VPC deve ter esta configuração:

- As regras de saída da ACL de rede devem permitir o tráfego nas portas TCP de 1024 a 65535 (portas efêmeras).
- As regras de entrada da ACL de rede devem permitir o tráfego na porta 443.

Além disso, a política do AWS Identity and Access Management (IAM) de endpoint da VPC do Amazon SQS deve permitir o acesso. No exemplo a seguir, a política de endpoint da VPC especifica que o usuário do IAM *MyUser* tem permissão para enviar mensagens à fila *MyQueue* do Amazon SQS. Outras ações, usuários do IAM e recursos do Amazon SQS têm acesso negado por meio do endpoint da VPC.

```
{
  "Statement": [{
    "Action": ["sqs:SendMessage"],
    "Effect": "Allow",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
  }]
}
```

UnknownHostException error

O erro `UnknownHostException` ocorre quando não é possível determinar o endereço IP do host.

Solução de problemas:

Use o utilitário nslookup para retornar o endereço IP associado ao nome do host:

- Windows and Linux OS

```
nslookup sqs.<region>.amazonaws.com
```

- AWS CLI ou SDK para endpoints legados do Python:

```
nslookup <region>.queue.amazonaws.com
```

Se você recebeu uma saída malsucedida, siga as instruções em [Como o DNS funciona e como solucionar falhas parciais ou intermitentes de DNS?](#) no Guia do Centro de Conhecimento da AWS.

Se você recebeu uma saída válida, é provável que seja um problema no nível da aplicação. Para solucionar problemas no nível da aplicação, tente os seguintes métodos:

- Reinicie a aplicação.
- Confirme se a aplicação Java não tem um cache DNS inválido. Se possível, configure a aplicação para aderir ao TTL do DNS. Consulte mais informações em [Setting the JVM TTL for DNS name lookups](#).

Consulte mais informações sobre como solucionar erros de rede em [How do I troubleshoot Amazon SQS “ETIMEOUT” and “UnknownHostException” connection errors?](#) no Guia do Centro de Conhecimento da AWS.

Solução de problemas de filas do Amazon Simple Queue Service usando o AWS X-Ray

AWS X-Ray coleta dados sobre solicitações que seu aplicativo atende e permite visualizar e filtrar dados para identificar possíveis problemas e oportunidades de otimização. Para qualquer solicitação rastreada para o aplicativo, é possível ver informações detalhadas sobre a solicitação, a resposta e as chamadas que o aplicativo faz para recursos, microsserviços, bancos de dados da AWS de downstream e APIs da web HTTP.

Para enviar cabeçalhos de rastreamento do AWS X-Ray por meio do Amazon SQS, é possível executar uma das seguintes ações:

- Usar o [cabeçalho de rastreamento](#) X-Amzn-Trace-Id.
- Usar o [atributo do sistema de mensagens](#) AWSTraceHeader

Para coletar dados sobre erros e latência, é necessário instrumentar o cliente do [AmazonSQS](#) usando o [SDK do AWS X-Ray](#).

É possível usar o console do AWS X-Ray para visualizar o mapa de conexões entre o Amazon SQS e outros serviços usados pela aplicação. Também é possível usar o console para visualizar métricas como a latência média e as taxas de falha. Para obter mais informações, consulte [Amazon SQS e AWS X-Ray](#) no Guia do desenvolvedor do AWS X-Ray.

Segurança no Amazon SQS

Esta seção fornece informações sobre a segurança, a autenticação e o controle de acesso do Amazon SQS, bem como a Linguagem de políticas de acesso do Amazon SQS.

Tópicos

- [Proteção de dados no Amazon SQS](#)
- [Gerenciamento de identidade e acesso no Amazon SQS](#)
- [Registrar em log e monitorar no Amazon SQS](#)
- [Validação de compatibilidade para o Amazon SQS](#)
- [Resiliência no Amazon SQS](#)
- [Segurança da infraestrutura no Amazon SQS](#)
- [Práticas recomendadas de segurança para o Amazon SQS](#)

Proteção de dados no Amazon SQS

O [modelo de responsabilidade AWS compartilhada](#) se aplica à proteção de dados no Amazon Simple Queue Service. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para saber mais sobre a privacidade de dados, consulte as [Data Privacy FAQ](#). Para saber mais sobre a proteção de dados na Europa, consulte a postagem do blog [AWS Shared Responsibility Model and RGPD](#) no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.

- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como [trabalhar com CloudTrail trilhas](#) no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte [Federal Information Processing Standard \(FIPS\) 140-3](#).

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Amazon SQS ou outro Serviços da AWS usando o console, a API ou. AWS CLI AWS SDKs Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

As seções a seguir fornecem informações sobre a proteção de dados no Amazon SQS.

Criptografia de dados no Amazon SQS

Proteção de dados refere-se a proteger os dados em trânsito (à medida que são transferidos para e do Amazon SQS) e em repouso (enquanto estão armazenados em discos em datacenters do Amazon SQS). Você pode proteger os dados em trânsito usando Secure Sockets Layer (SSL) ou criptografia no lado do cliente. Por padrão, o Amazon SQS armazena mensagens e arquivos usando criptografia de disco. Para proteger dados em repouso, solicite que o Amazon SQS criptografe as mensagens antes de salvá-las nos sistemas de arquivo criptografados em seus data centers. O Amazon SQS recomenda o uso de SSE para otimizar a criptografia de dados.

Tópicos

- [Criptografia em repouso no Amazon SQS](#)
- [Gerenciamento de chaves do Amazon SQS](#)

Criptografia em repouso no Amazon SQS

A criptografia no lado do servidor (SSE) permite que você transmita dados sigilosos em filas criptografadas. O SSE protege o conteúdo das mensagens em filas usando chaves de criptografia gerenciadas pelo SQS (SSE-SQS) ou chaves gerenciadas no (SSE-KMS). AWS Key Management Service Para obter informações sobre como gerenciar o SSE usando o Console de gerenciamento da AWS, consulte o seguinte:

- [Configurar o SSE-SQS para uma fila \(console\)](#)
- [Configurar o SSE-KMS para uma fila \(console\)](#)

Para obter informações sobre como gerenciar o SSE usando as AWS SDK para Java (e [GetQueueAttributes](#) as ações [CreateQueueSetQueueAttributes](#), e), consulte os exemplos a seguir:

- [Usar criptografia do lado do servidor com filas do Amazon SQS](#)
- [Configurando permissões do KMS para Serviços da AWS](#)

A SSE criptografa mensagens assim que o Amazon SQS as recebe. As mensagens são armazenadas no formato criptografado e o Amazon SQS as descriptografa apenas quando elas são enviadas a um consumidor autorizado.

Important

Todas as solicitações para filas com SSE habilitada devem usar HTTPS e o [Signature versão 4](#).

Uma [fila criptografada](#) que usa a chave padrão (chave KMS AWS gerenciada para Amazon SQS) não pode invocar uma função Lambda em outra. Conta da AWS

Alguns recursos dos AWS serviços que podem enviar notificações para o Amazon SQS usando a AWS Security Token Service [AssumeRole](#) ação são compatíveis com o SSE, mas funcionam somente com filas padrão:

- [Ganchos do ciclo de vida do Auto Scaling](#)
- [AWS Lambda Dead Letter Queues](#)

Para obter informações sobre a compatibilidade de outros produtos com filas criptografadas, consulte [Configurar permissões do KMS para serviços AWS](#) e a documentação do seu produto.

AWS KMS combina hardware e software seguros e de alta disponibilidade para fornecer um sistema de gerenciamento de chaves dimensionado para a nuvem. Quando você usa o Amazon SQS com AWS KMS, as [chaves de dados](#) que criptografam os dados da sua mensagem também são criptografadas e armazenadas com os dados que elas protegem.

Os benefícios de usar o AWS KMS são os seguintes:

- É possível criar e gerenciar [AWS KMS keys](#) por conta própria:
- Você também pode usar a chave KMS AWS gerenciada para o Amazon SQS, que é exclusiva para cada conta e região.
- Os padrões AWS KMS de segurança podem ajudá-lo a atender aos requisitos de conformidade relacionados à criptografia.

Para obter mais informações, consulte [O que é o AWS Key Management Service?](#) no Guia do desenvolvedor do AWS Key Management Service .

Escopo de criptografia

A SSE criptografa o corpo de uma mensagem em uma fila do Amazon SQS.

A SSE não criptografa o seguinte:

- Metadados de fila (nome e atributos da fila)
- Metadados de mensagens (ID de mensagem, carimbo de data/hora e atributos)
- Métricas por fila

A criptografia de uma mensagem torna indisponível seu conteúdo para usuários não autorizados ou anônimos. Com a SSE habilitada, as solicitações anônimas `SendMessage` e `ReceiveMessage` à fila criptografada serão rejeitadas. As práticas recomendadas de segurança do Amazon SQS não aconselham o uso de solicitações anônimas. Se você quiser enviar solicitações anônimas a uma fila do Amazon SQS, desabilite o SSE. Isso não afeta o funcionamento normal do Amazon SQS:

- Uma mensagem só será criptografada se for enviada após a habilitação da criptografia de uma fila. O Amazon SQS não criptografa mensagens com lista de pendências.
- Qualquer mensagem criptografada permanecerá dessa forma mesmo se a criptografia de sua fila for desabilitada.

A transferência de uma mensagem para uma [dead letter queue](#) não afeta sua criptografia:

- Quando o Amazon SQS move uma mensagem de uma fila de origem criptografada para uma fila de mensagens não entregues não criptografada, a mensagem permanece criptografada.
- Quando o Amazon SQS move uma mensagem de uma fila de origem não criptografada para uma fila de mensagens não entregues criptografada, a mensagem permanece descriptografada.

Principais termos

Os seguintes termos-chave podem ajudar você a entender melhor a funcionalidade da SSE. Para obter descrições detalhadas, consulte a [Referência da API do Amazon Simple Queue Service](#).

Chave de dados

A chave (DEK) responsável por criptografar o conteúdo de mensagens do Amazon SQS.

Para obter mais informações, consulte [Chaves de dados](#) no Guia do desenvolvedor do AWS Key Management Service no Guia do desenvolvedor do AWS Encryption SDK .

Período de reutilização de chaves de dados

O período de tempo, em segundos, durante o qual o Amazon SQS pode reutilizar uma chave de dados para criptografar ou descriptografar mensagens antes de ligar novamente. AWS KMS Um número inteiro que representa segundos, entre 60 segundos (1 minuto) e 86.400 segundos (24 horas). O padrão é 300 (5 minutos). Para obter mais informações, consulte [Entender o período de reutilização de chaves de dados](#).

Note

No caso improvável de não conseguir acessar AWS KMS, o Amazon SQS continua usando a chave de dados em cache até que a conexão seja restabelecida.

ID da chave do KMS

O alias, o ARN do alias, o ID da chave ou o ARN da chave de uma chave KMS AWS gerenciada ou de uma chave KMS personalizada — na sua conta ou em outra conta. Embora o alias da chave KMS AWS gerenciada para o Amazon SQS seja `alias/aws/sqs` sempre, o alias de uma chave KMS personalizada pode, por exemplo, ser `alias/MyAlias`. Você pode usar essas chaves do KMS para proteger as mensagens em filas do Amazon SQS.

Note

Lembre-se do seguinte:

- Se você não especificar uma chave KMS personalizada, o Amazon SQS usa a chave KMS gerenciada para AWS o Amazon SQS.
- A primeira vez que você usa o Console de gerenciamento da AWS para especificar a chave KMS AWS gerenciada para o Amazon SQS para uma fila AWS KMS , cria a chave KMS gerenciada para AWS o Amazon SQS.
- Como alternativa, na primeira vez em que você usa a `SendMessageBatch` ação `SendMessage` or em uma fila com o SSE ativado, AWS KMS cria a chave KMS AWS gerenciada para o Amazon SQS.

Você pode criar chaves KMS, definir as políticas que controlam como as chaves KMS podem ser usadas e auditar o uso da chave KMS usando a seção Chaves gerenciadas pelo cliente do AWS KMS console ou da ação. [CreateKey](#) AWS KMS Para obter mais informações, consulte [Chaves do KMS](#) e [Como criar chaves do KMS](#) no Guia do desenvolvedor da AWS Key Management Service . Para obter mais exemplos de identificadores de chave KMS, consulte a [KeyId](#)Referência da AWS Key Management Service API. Para obter informações sobre como encontrar identificadores de chave do KMS, consulte [Encontrar o ID da chave e o ARN](#) no Guia do desenvolvedor do AWS Key Management Service .

Important

Há taxas adicionais pelo uso AWS KMS. Para saber mais, consulte [Estimando custos AWS KMS](#) e [Definição de preço do AWS Key Management Service](#).

Criptografia de envelope

A segurança dos dados criptografados depende em parte da proteção da chave de dados que pode descriptografá-los. O Amazon SQS usa a chave do KMS para criptografar a chave de dados. Em seguida, a chave de dados criptografada é armazenada com a mensagem criptografada. Essa prática de uso de uma chave do KMS para criptografar chaves de dados é conhecida como criptografia de envelope.

Para obter mais informações, consulte [Criptografia de envelope](#) no Guia do desenvolvedor do AWS Encryption SDK .

Gerenciamento de chaves do Amazon SQS

O Amazon SQS se integra ao AWS Key Management Service (KMS) para gerenciar [chaves KMS para criptografia do lado](#) do servidor (SSE). Consulte [Criptografia em repouso no Amazon SQS](#) para obter informações sobre SSE e definições de gerenciamento de chaves. O Amazon SQS usa chaves do KMS para validar e proteger as chaves de dados que criptografam e descriptografam as mensagens. As seções a seguir fornecem informações sobre como trabalhar com chaves do KMS e chaves de dados no produto Amazon SQS.

Configurar permissões do AWS KMS

Cada chave do KMS deve ter uma política de chaves. Observe que você não pode modificar a política de chaves de uma chave KMS AWS gerenciada para o Amazon SQS. A política dessa chave do KMS inclui permissões de uso das filas criptografadas para todos os principais na conta (que estão autorizados a usar o Amazon SQS).

O Amazon SQS distingue os chamadores com base em suas AWS credenciais, independentemente de estarem usando AWS contas, usuários do IAM ou funções do IAM diferentes. Além disso, o mesmo perfil do IAM com políticas de escopo diferentes será tratada como solicitantes distintos. No entanto, ao usar sessões de perfil do IAM, variar apenas RoleSessionName e manter o mesmo perfil do IAM e políticas de escopo não criará solicitantes distintos. Portanto, ao especificar os AWS KMS principais princípios de política, evite confiar apenas nas RoleSessionName diferenças, pois essas sessões serão tratadas como o mesmo solicitante.

Como alternativa, é possível especificar as permissões necessárias em uma política do IAM atribuída aos principais que produzem e consomem mensagens criptografadas. Para obter mais informações, consulte [Usar políticas do IAM com AWS KMS](#) no Guia do desenvolvedor do AWS Key Management Service .

 Note

Embora você possa configurar permissões globais para enviar e receber do Amazon SQS, é AWS KMS necessário nomear explicitamente o ARN completo das chaves KMS em regiões específicas na seção de uma política do IAM. Resource

Configurar permissões do KMS para serviços AWS

Vários AWS serviços atuam como fontes de eventos que podem enviar eventos para as filas do Amazon SQS. Para permitir que essas fontes de eventos funcionem com filas criptografadas, você deve criar uma chave KMS gerenciada pelo cliente e adicionar permissões na política de chaves para que o serviço use os métodos de AWS KMS API necessários. Execute as etapas a seguir para configurar as permissões.

 Warning

Ao alterar a chave do KMS para criptografar suas mensagens do Amazon SQS, esteja ciente de que as mensagens existentes criptografadas com a chave do KMS antiga permanecerão criptografadas com essa chave. Para descriptografar essas mensagens, você deve reter a chave do KMS antiga e garantir que sua política de chaves conceda ao Amazon SQS as permissões para `kms:Decrypt` e `kms:GenerateDataKey`. Depois de atualizar para uma nova chave do KMS para criptografar novas mensagens, certifique-se de que todas as mensagens existentes criptografadas com a chave do KMS antiga sejam processadas e removidas da fila antes de excluir ou desabilitar a chave do KMS antiga.

1. Para criar uma chave do KMS gerenciada pelo cliente. Para obter mais informações, consulte [Criação de chaves](#) no Guia do desenvolvedor AWS Key Management Service .
2. Para permitir que a fonte do evento de AWS serviço use os métodos `kms:Decrypt` e `kms:GenerateDataKey` da API, adicione a seguinte declaração à política de chaves do KMS.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
```

```

    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
    ],
    "Resource": "*"
  }
}

```

Substitua “service” no exemplo acima pelo nome do serviço da origem de evento. As origens de evento incluem os serviços a seguir.

Origem do evento.	Nome do serviço
CloudWatch Eventos da Amazon	events.amazonaws.com
Notificações de eventos do Amazon S3	s3.amazonaws.com
Assinaturas de tópicos do Amazon SNS	sns.amazonaws.com

3. [Configure uma fila com SSE existente](#) usando o ARN de sua chave do KMS.
4. Forneça o ARN da fila criptografada para a fonte do evento.

Configurar AWS KMS permissões para produtores

Quando o [período de reutilização da chave de dados](#) expirar, a próxima chamada do produtor para `SendMessage` ou `SendMessageBatch` também acionará chamadas para `kms:Decrypt` e `kms:GenerateDataKey`. A chamada para `kms:Decrypt` tem o intuito de verificar a integridade da nova chave de dados antes de usá-la. Portanto, o produtor deve ter as permissões `kms:Decrypt` e `kms:GenerateDataKey` para a chave do KMS.

Adicione a declaração a seguir à política do IAM do produtor. Lembre-se de usar os valores de ARN corretos para o recurso da chave e o recurso da fila.

JSON

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
    ],
    "Resource": "arn:aws:kms:us-east-2:123456789012:key/111112222233333"
  },
  {
    "Effect": "Allow",
    "Action": [
      "sqs:SendMessage"
    ],
    "Resource": "arn:aws:sqs:*:123456789012:MyQueue"
  }
]
```

Configurar AWS KMS permissões para consumidores

Quando o período de reutilização da chave de dados expirar, a próxima chamada do consumidor `ReceiveMessage` também acionará uma chamada para `kms:Decrypt`, a fim de verificar a integridade da nova chave de dados antes de usá-la. Portanto, o consumidor deve ter a permissão `kms:Decrypt` para qualquer chave do KMS que é usada para criptografar as mensagens na fila específica. Se a fila agir como uma [dead letter queue](#), o consumidor também deverá ter a permissão `kms:Decrypt` para qualquer chave do KMS que for usada para criptografar as mensagens na fila de origem. Adicione a declaração a seguir à política do IAM do consumidor. Lembre-se de usar os valores de ARN corretos para o recurso da chave e o recurso da fila.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "arn:aws:kms:us-east-2:123456789012:key/111122223333",
  },
  {
    "Effect": "Allow",
    "Action": [
      "sqs:ReceiveMessage"
    ],
    "Resource": "arn:aws:sqs:*:123456789012:MyQueue"
  }
]
}

```

Configurar permissões do AWS KMS com proteção contra representante confuso

Quando a entidade principal em uma declaração de política é uma [AWS entidade principal do serviço da](#), você pode usar as chaves de condição global [aws:SourceArn](#) ou [aws:SourceAccount](#) para proteção contra o [cenário de representante confuso](#). Para usar essas chaves de condição, defina o valor como o nome do recurso da Amazon (ARN) do recurso que está sendo criptografado. Se você não conhece o ARN do recurso, use `aws:SourceAccount` em vez disso.

Nesta política de chaves do KMS, um recurso específico do serviço que é de propriedade da conta 111122223333 tem permissão para chamar o KMS para ações Decrypt e GenerateDataKey, que ocorrem durante o uso do SSE do Amazon SQS.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sqs.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {

```

```
    "ArnEquals": {  
      "aws:SourceArn": [  
        "arn:aws:sqs:us-west-1:111122223333:resource"  
      ]  
    }  
  }  
}
```

Ao usar filas do Amazon SQS habilitadas para SSE, os seguintes serviços são compatíveis com `aws:SourceArn`:

- Amazon SNS
- Amazon S3
- CloudWatch Eventos
- AWS Lambda
- CodeBuild
- Amazon Connect Customer Profiles
- AWS Auto Scaling
- Amazon Chime

Entender o período de reutilização de chaves de dados

O [período de reutilização da chave de dados](#) define a duração máxima de reutilização da mesma chave de dados pelo Amazon SQS. Quando o período de reutilização da chave de dados terminar, o Amazon SQS gerará uma nova chave de dados. Observe as diretrizes a seguir sobre o período de reutilização.

- Um período de reutilização mais curto oferece melhor segurança, mas resulta em mais chamadas para AWS KMS, o que pode gerar cobranças além do nível gratuito.
- Embora a chave de dados seja armazenada em cache separadamente para a criptografia e a descriptografia, o período de reutilização se aplica a ambas as cópias da chave de dados.
- Quando o período de reutilização da chave de dados termina, a próxima chamada `SendMessage` ou `SendMessageBatch` normalmente aciona uma chamada para o AWS KMS `GenerateDataKey` método para obter uma nova chave de dados. Além disso, as próximas

chamadas para `SendMessage` e cada uma `ReceiveMessage` acionará uma chamada AWS KMS `Decrypt` para verificar a integridade da chave de dados antes de usá-la.

- [Diretores](#) (Contas da AWS ou usuários) não compartilham chaves de dados (mensagens enviadas por diretores exclusivos sempre recebem chaves de dados exclusivas). Portanto, o volume de chamadas para AWS KMS é um múltiplo do número de principais exclusivos em uso durante o período de reutilização da chave de dados.

Estimando custos AWS KMS

Para prever custos e entender melhor sua AWS fatura, talvez você queira saber com que frequência o Amazon SQS usa sua chave KMS.

Note

Embora a fórmula a seguir possa dar a você uma boa ideia sobre os custos esperados, os custos reais poderão ser mais altos por conta da natureza distribuída do Amazon SQS.

Para calcular o número de solicitações de APIs (R) por fila, use a seguinte fórmula:

$$R = (B / D) * (2 * P + C)$$

B é o período de faturamento (em segundos).

D é o [período de reutilização da chave de dados](#) (em segundos).

P é o número de [entidades](#) de produção que enviam para a fila do Amazon SQS.

C é o número de entidades de consumo que recebem da fila do Amazon SQS.

Important

De modo geral, os principais de produção geram o dobro do custo das entidades principais de consumo. Para obter mais informações, consulte [Entender o período de reutilização de chaves de dados](#).

Se o produtor e o consumidor tiverem usuários diferentes do , o custo aumentará.

Estes são cálculos de exemplo. Para obter informações sobre a definição de preços, consulte [Definição de preços do AWS Key Management Service](#).

Exemplo 1: cálculo do número de chamadas de AWS KMS API para 2 principais e 1 fila

Este exemplo supõe o seguinte:

- O período de faturamento é de 1 a 31 de janeiro (2.678.400 segundos).
- O período de reutilização de chave de dados é definido como 5 minutos (300 segundos).
- Há 1 fila.
- Há 1 entidade principal de produção e 1 entidade principal de consumo.

$$(2,678,400 / 300) * (2 * 1 + 1) = 26,784$$

Exemplo 2: cálculo do número de chamadas de AWS KMS API para vários produtores e consumidores e duas filas

Este exemplo supõe o seguinte:

- O período de faturamento é de 1 a 28 de fevereiro (2.419.200 segundos).
- O período de reutilização de chave de dados é definido como 24 horas (86.400 segundos).
- Há duas filas.
- A primeira fila tem 3 entidades principais de produção e 1 entidade principal de consumo.
- A segunda fila tem 5 entidades principais de produção e 2 entidades principais de consumo.

$$(2,419,200 / 86,400 * (2 * 3 + 1)) + (2,419,200 / 86,400 * (2 * 5 + 2)) = 532$$

AWS KMS erros

Quando você trabalha com o Amazon SQS e AWS KMS, você pode encontrar erros. As referências a seguir descrevem os erros e possíveis soluções de problemas.

- [Erros comuns do AWS KMS](#)
- [Erros decriptografia do AWS KMS](#)
- [AWS KMS GenerateDataKey erros](#)

Privacidade do tráfego entre redes no Amazon SQS

Um endpoint da Amazon Virtual Private Cloud (Amazon VPC) para Amazon SQS é uma entidade lógica dentro de uma VPC que permite conectividade apenas com o Amazon SQS. A VPC roteia as solicitações para o Amazon SQS e as respostas de volta para a VPC. As seções a seguir contêm informações sobre como trabalhar com VPC endpoints e criar políticas de VPC endpoint.

Endpoints da Amazon Virtual Private Cloud para o Amazon SQS

Se você usa o Amazon VPC para hospedar seus AWS recursos, você pode estabelecer uma conexão entre seu VPC e o Amazon SQS. Você pode usar essa conexão para enviar mensagens às suas filas do Amazon SQS sem precisar passar pela Internet pública.

A Amazon VPC permite que você lance AWS recursos em uma rede virtual personalizada. Você pode usar uma VPC para controlar as configurações de rede, como o intervalo de endereços IP, sub-redes, tabelas de rotas e gateways de rede. Para obter mais informações sobre VPCs, consulte o Guia do [usuário da Amazon VPC](#).

Para conectar a VPC ao Amazon SQS, primeiro você deve definir um endpoint da VPC de interface, que permite conectar a VPC a outros produtos da AWS. O endpoint fornece uma conectividade confiável e escalável ao Amazon SQS sem a necessidade de um gateway da Internet, de uma instância de conversão de endereço de rede (NAT) ou de uma conexão VPN. Para obter mais informações, consulte [Tutorial: Envio de uma mensagem a uma fila do Amazon SQS pela Amazon Virtual Private Cloud](#) e [Exemplo 5: negar o acesso se não vier de um VPC endpoint](#) neste guia e [Endpoints da VPC de interface \(AWS PrivateLink\)](#) no Guia do usuário da Amazon VPC.

Important

- Você pode usar a Amazon Virtual Private Cloud somente com endpoints HTTPS do Amazon SQS.
- Ao configurar o Amazon SQS para enviar mensagens pela Amazon VPC, habilite o DNS privado e especifique endpoints no formato `sqs.us-east-2.amazonaws.com` ou `sqs.us-east-2.api.aws` para o endpoint de pilha dupla.
- O Amazon SQS também oferece suporte a endpoints FIPS por meio do PrivateLink uso do serviço de endpoint. `com.amazonaws.region.sqs-fips` Você pode se conectar aos endpoints FIPS no formato `sqs-fips.region.amazonaws.com`.
- Ao usar o endpoint de pilha dupla na Amazon Virtual Private Cloud, as solicitações serão enviadas usando e. IPv4 IPv6

- O DNS privado não oferece suporte a endpoints legados, como `queue.amazonaws.com` ou `us-east-2.queue.amazonaws.com`.

Criar uma política de endpoint da Amazon VPC para o Amazon SQS

É possível criar uma política para endpoints da Amazon VPC para o Amazon SQS na qual se especifica o seguinte:

- A entidade principal que pode realizar ações.
- As ações que podem ser realizadas.
- Os recursos aos quais as ações podem ser aplicadas.

Para obter mais informações, consulte [Controlar o acesso a produtos com endpoints da VPC](#) no Guia do usuário da Amazon VPC

O exemplo de política de endpoint da VPC a seguir especifica que o usuário `MyUser` tem permissão para enviar mensagens à fila `MyQueue` do Amazon SQS.

```
{
  "Statement": [{
    "Action": ["sqs:SendMessage"],
    "Effect": "Allow",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
  }]
}
```

O seguinte é negado:

- Outras ações de API do Amazon SQS, como `sqs:CreateQueue` e `sqs>DeleteQueue`.
- Outros usuários e regras do que tentam usar esse VPC endpoint.
- Envio de mensagens de `MyUser` para outra fila do Amazon SQS.

 Note

O usuário ainda pode usar outras ações de API do Amazon SQS de fora da VPC. Para obter mais informações, consulte [Exemplo 5: negar o acesso se não vier de um VPC endpoint](#).

Conecte-se ao Amazon SQS usando endpoints de pilha dupla (e) IPv4 IPv6

Os endpoints de pilha dupla oferecem suporte tanto ao tráfego quanto ao tráfego. IPv4 IPv6 Quando você faz uma solicitação para um endpoint de pilha dupla, o URL do endpoint é resolvido para um endereço ou um. IPv4 IPv6 Para obter mais informações sobre endpoints de pilha dupla e FIPS, consulte o [Guia de referência de SDK](#).

O Amazon SQS oferece suporte a endpoints regionais de pilha dupla, o que significa que você deve especificar a AWS região como parte do nome do endpoint. Os nomes de endpoint de pilha dupla usam a seguinte convenção de nomenclatura: `sqs.Region.amazonaws.com` Por exemplo, o nome do endpoint de pilha dupla para a região eu-west-1 é `sqs.eu-west-1.amazonaws.com`.

Para obter a lista completa dos endpoints do Amazon SQS, consulte a [Referência geral da AWS](#).

Gerenciamento de identidade e acesso no Amazon SQS

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para utilizar os recursos do Amazon SQS. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte [Resolução de problemas de identidade e acesso do Amazon Simple Queue Service](#)).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte [Como o Amazon Simple Queue Service funciona com o IAM](#))

- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte [Práticas recomendadas de política](#))

Autenticação com identidades

A autenticação é como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte [Como fazer login em sua Conta da AWS](#) no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte [AWS Signature Version 4 para solicitações de API](#) no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em [Tarefas que exigem credenciais de usuário-raiz](#) no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários humanos usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório corporativo, provedor de identidade da web ou Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem funções que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos Centro de Identidade do AWS IAM. Para saber mais, consulte [O que é o IAM Identity Center?](#) no Guia do usuário do Centro de Identidade do AWS IAM .

Usuários e grupos do IAM

Um [usuário do IAM](#) é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte [Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias](#) no Guia do usuário do IAM.

Um [grupo do IAM](#) especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte [Casos de uso de usuários do IAM](#) no Guia do usuário do IAM.

Perfis do IAM

Uma [perfil do IAM](#) é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função [mudando de um usuário para uma função do IAM \(console\)](#) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte [Métodos para assumir um perfil](#) no Manual do usuário do IAM.

As funções do IAM são úteis para acesso de usuários federados, permissões temporárias de usuários do IAM, acesso entre contas, acesso entre serviços e aplicativos executados na Amazon. EC2 Consulte mais informações em [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte [Visão geral das políticas JSON](#) no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você anexa a uma identidade (usuário, grupo ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte [Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente](#) no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte [Escolher entre políticas gerenciadas e políticas em linha](#) no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário [especificar uma entidade principal](#) em uma política baseada em recursos.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte [Limites de permissões para identidades do IAM](#) no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — Especifique as permissões máximas para uma organização ou unidade organizacional em AWS Organizations. Para saber mais, consulte [Políticas de controle de serviço](#) no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs) — Defina o máximo de permissões disponíveis para recursos em suas contas. Para obter mais informações, consulte [Políticas de controle de recursos \(RCPs\)](#) no Guia AWS Organizations do usuário.

- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte [Políticas de sessão](#) no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte [Lógica de avaliação de políticas](#) no Guia do usuário do IAM.

Visão geral do gerenciamento de acesso no Amazon SQS

Cada AWS recurso é de propriedade de um Conta da AWS, e as permissões para criar ou acessar um recurso são regidas por políticas de permissões. Um administrador da conta pode anexar políticas de permissões a identidades do IAM (usuários, grupos e funções) e alguns produtos (como o Amazon SQS) também oferecem suporte à anexação de políticas de permissões aos recursos.

Note

O administrador da conta (ou usuário administrador) é um usuário com privilégios administrativos. Para obter mais informações, consulte [Práticas recomendadas de segurança no IAM](#) no Guia do usuário do IAM.

Ao conceder permissões, você especifica os usuários que recebem permissões, o recurso para o qual as permissões são concedidas e as ações específicas que você deseja permitir no recurso.

Recursos e operações do Amazon Simple Queue Service

No Amazon SQS, o único recurso é a fila. Em uma política, use um nome de recurso da Amazon (ARN) para identificar o recurso ao qual a política se aplica. O seguinte recurso tem um ARN exclusivo associado a ele:

Tipo de atributo	Formato ARN
Fila	<code>arn:aws:sqs: <i>region</i>:<i>account_id</i>:<i>queue_name</i></code>

Veja a seguir exemplos do formato do ARN para filas:

- Um ARN para uma fila nomeada `my_queue` na região Leste dos EUA (Ohio), pertencente à AWS Conta 123456789012:

```
arn:aws:sqs:us-east-2:123456789012:my_queue
```

- Um ARN para uma fila chamada `my_queue` em cada uma das diferentes regiões compatíveis com o Amazon SQS:

```
arn:aws:sqs:*:123456789012:my_queue
```

- Um ARN que usa `*` ou `?` como um curinga para o nome da fila. No exemplo a seguir, o ARN corresponde a todas as filas prefixadas com `my_prefix_`:

```
arn:aws:sqs:*:123456789012:my_prefix_*
```

Você pode obter o valor do ARN para uma fila existente chamando a ação [GetQueueAttributes](#). O valor do atributo `QueueArn` é o ARN da fila. Para obter mais informações sobre ARNs, consulte [IAM ARNs](#) no Guia do usuário do IAM.

O Amazon SQS fornece um conjunto de ações que funcionam com o recurso de fila. Para obter mais informações, consulte [Permissões da API do Amazon SQS: referência de ações e recurso](#).

Informações sobre propriedade de recursos

Ele Conta da AWS possui os recursos criados na conta, independentemente de quem criou os recursos. Mais especificamente, o proprietário do recurso é a Conta da AWS da entidade principal (ou seja, a conta raiz, um usuário ou um perfil do IAM) que autentica a solicitação de criação de recursos. Os seguintes exemplos mostram como isso funciona:

- Se você usar as credenciais da sua conta raiz Conta da AWS para criar uma fila do Amazon SQS, Conta da AWS você é o proprietário do recurso (no Amazon SQS, o recurso é a fila do Amazon SQS).
- Se você criar um usuário no seu Conta da AWS e conceder permissões para criar uma fila para o usuário, o usuário poderá criar a fila. No entanto, a Conta da AWS (à qual o usuário pertence) é a proprietária do recurso de fila.

- Se você criar uma função do IAM Conta da AWS com permissões para criar uma fila do Amazon SQS, qualquer pessoa que possa assumir a função poderá criar uma fila. Seu Conta da AWS (ao qual a função pertence) é proprietário do recurso de fila.

Gerenciar acesso aos recursos da

Uma política de permissões descreve as permissões concedidas às contas. A seção a seguir explica as opções disponíveis para a criação das políticas de permissões.

Note

Esta seção discute o uso do IAM no contexto do Amazon SQS. Não são fornecidas informações detalhadas sobre o serviço IAM. Para ver a documentação completa do IAM, consulte [What is IAM?](#) no IAM User Guide. Para obter mais informações sobre a sintaxe e as descrições da política do IAM, consulte a [Referência de política do AWS IAM](#) no Guia do usuário do IAM.

As políticas anexadas a uma identidade do IAM são conhecidas como políticas baseadas em identidade (políticas do IAM;) e as políticas anexadas a um recurso são conhecidas como políticas baseadas em recurso.

Políticas baseadas em identidade

Há duas maneiras de conceder aos usuários permissões às suas filas do Amazon SQS: usando os sistemas de políticas do Amazon SQS e do IAM. Você pode usar um dos sistemas, ou ambos, para anexar políticas a usuários ou funções. Na maioria dos casos, você pode atingir o mesmo resultado usando um dos sistemas. Por exemplo, você pode fazer o seguinte:

- Anexar uma política de permissões a um usuário ou grupo na conta: para conceder a um usuário permissões para criar uma fila do Amazon SQS, anexe uma política de permissões a um usuário ou grupo a que o usuário pertença.
- Anexar uma política de permissão a um usuário em outro Conta da AWS — Você pode anexar uma política de permissões a um usuário em outro Conta da AWS para permitir que ele interaja com uma fila do Amazon SQS. No entanto, as permissões entre contas não se aplicam às seguintes ações:

As permissões entre contas não se aplicam às seguintes ações:

- [AddPermission](#)
- [CancelMessageMoveTask](#)
- [CreateQueue](#)
- [DeleteQueue](#)
- [ListMessageMoveTask](#)
- [ListQueues](#)
- [ListQueueTags](#)
- [RemovePermission](#)
- [SetQueueAttributes](#)
- [StartMessageMoveTask](#)
- [TagQueue](#)
- [UntagQueue](#)

Para conceder acesso a essas ações, o usuário deve pertencer à mesma Conta da AWS que detém a fila do Amazon SQS.

- Anexar uma política de permissões a um perfil (conceder permissões entre contas): para conceder permissões entre contas a uma fila do SQS, você deve combinar políticas do IAM e baseadas em recursos:

1. Na Conta A (que é proprietária da fila):

- Anexe uma política baseada em recursos à fila SQS. Essa política deve conceder explicitamente as permissões necessárias (por exemplo, [SendMessage](#), [ReceiveMessage](#)) à entidade principal na Conta B (como um perfil do IAM).

2. Na Conta A, crie um perfil do IAM.

- Uma política de confiança que permita que a Conta B ou um AWS service (Serviço da AWS) assuma o perfil.

 Note

Se você quiser que um AWS service (Serviço da AWS) (como Lambda ou EventBridge) assuma a função, especifique o principal do serviço (por exemplo, `lambda.amazonaws.com`) na política de confiança.

- Uma política baseada em identidade que concede ao perfil assumido permissões para interagir com a fila.

3. Na Conta B, conceda permissão para assumir o perfil na Conta A.

Você deve configurar a política de acesso da fila para permitir a entidade principal entre contas. As políticas baseadas em identidade do IAM, por si só, não são suficientes para o acesso entre contas às filas do SQS.

Para obter mais informações sobre o uso do IAM para delegar permissões, consulte [Gerenciamento de acesso](#) no Guia do usuário do IAM.

Embora o Amazon SQS funcione com políticas do IAM, ele tem sua própria infraestrutura de políticas. Você pode usar uma política do Amazon SQS com uma fila para especificar quais AWS contas têm acesso à fila. Você pode especificar o tipo de acesso e condições (por exemplo, uma condição que conceda permissões para usar `SendMessage`, `ReceiveMessage` se a solicitação for feita antes de 31 de dezembro de 2010). As ações específicas para as quais você pode conceder permissões são um subconjunto de toda a lista de ações do Amazon SQS. Quando você grava uma política do Amazon SQS e especifica `*` para “permitir todas as ações do Amazon SQS”, significa que um usuário pode realizar todas as ações nesse subconjunto.

O diagrama a seguir ilustra o conceito de uma dessas políticas básicas do Amazon SQS que abrange o subconjunto de ações. A política é para `queue_xyz` e concede à AWS Conta 1 e à AWS Conta 2 permissões para usar qualquer uma das ações permitidas com a fila especificada.

Note

O recurso na política é especificado como `123456789012/queue_xyz`, onde `123456789012` está o AWS ID da conta que possui a fila.

SQS Policy on queue_xyz

Allow who:

AWS account 1

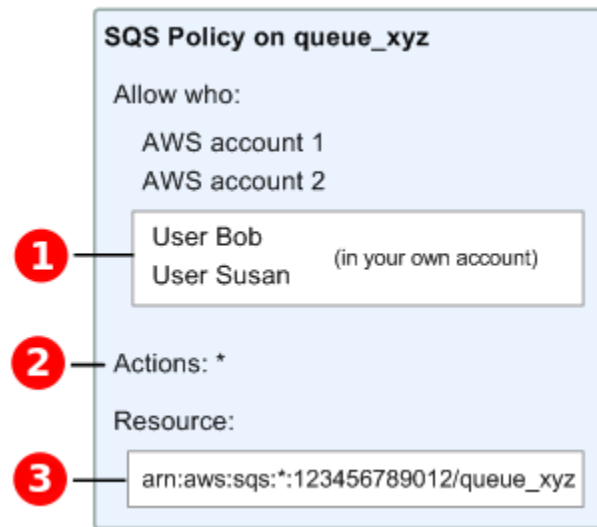
AWS account 2

Actions: *

Resource:

123456789012/queue_xyz

Com a introdução do IAM e os conceitos de Users e Amazon Resource Names (ARNs), algumas coisas mudaram nas políticas do SQS. O diagrama e a tabela a seguir descrevem as alterações.



1 Para obter informações sobre como conceder permissões a usuários em contas diferentes, consulte [Tutorial: Delegar acesso entre AWS contas usando funções do IAM](#) no Guia do usuário do IAM.

2 O subconjunto de ações incluídas em * foi expandido. Para obter uma lista de ações permitidas, consulte [Permissões da API do Amazon SQS: referência de ações e recurso](#).

3 Você pode especificar o recurso usando o nome do recurso da Amazon (ARN), a forma padrão de especificar recursos nas políticas do IAM. Para obter informações sobre o formato ARN para filas do Amazon SQS, consulte [Recursos e operações do Amazon Simple Queue Service](#).

Por exemplo, de acordo com a política do Amazon SQS no diagrama anterior, qualquer pessoa que possua as credenciais de segurança da AWS Conta 1 ou AWS da Conta 2 pode acessar. queue_xyz Além disso, os usuários Bob e Susan em sua própria conta da AWS (com o ID 123456789012) podem acessar a fila.

Antes da introdução do IAM, o Amazon SQS concedia automaticamente ao criador de uma fila o controle total sobre ela (ou seja, o acesso a todas as ações possíveis do Amazon SQS nessa fila). Isso não é mais verdadeiro, a menos que o criador use credenciais de segurança da AWS. Qualquer usuário que tenha permissões para criar uma fila também deve ter permissões para usar outras ações do Amazon SQS, para fazer qualquer coisa com as filas criadas.

Veja a seguir uma política de exemplo que permite que um usuário use todas as ações do Amazon SQS, mas apenas com as filas cujos nomes estejam prefixados com a string literal `bob_queue_`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sqs:*",
    "Resource": "arn:aws:sqs:*:123456789012:bob_queue_*"
  }]
}
```

Para obter mais informações, consulte [Usar políticas com o Amazon SQS](#) e [Identities \(usuários, grupos e funções\)](#) no Guia do usuário do IAM.

Especificar elementos da política: ações, efeitos, recursos e entidades principais

Para cada [recurso do Amazon Simple Queue Service](#), o produto define um conjunto de [ações](#). Para conceder permissões a essas ações, o Amazon SQS define um conjunto de ações que podem ser especificadas em uma política.

Note

A execução de uma ação pode exigir permissões para mais de uma ação. Ao conceder permissões para ações específicas, você também identifica o recurso para o qual as ações são permitidas ou recusadas.

Estes são os elementos de política mais básicos:

- **Recurso:** em uma política, você usa um Amazon Resource Name (ARN – Nome do recurso da Amazon) para identificar o recurso a que a política se aplica.
- **Ação:** você usa palavras-chave de ação para identificar as ações de recurso que deseja permitir ou negar. Por exemplo, a permissão `sqs:CreateQueue` permite que o usuário execute a ação `CreateQueue` do Amazon Simple Queue Service.

- **Efeito:** você especifica o efeito quando o usuário solicita a ação específica, que pode ser permitir ou negar. Se você não conceder explicitamente acesso a um recurso, o acesso estará implicitamente negado. Você também pode negar explicitamente o acesso a um recurso a fim de ter certeza de que um usuário não conseguirá acessá-lo, mesmo que uma política diferente conceda acesso.
- **Principal:** em políticas baseadas em identidade (políticas do IAM), o usuário ao qual a política é anexada é implicitamente o principal. Para as políticas baseadas em recursos, você especifica quais usuários, contas, serviços ou outras entidades deseja que recebam permissões (isso se aplica somente a políticas baseadas em recursos).

Para saber mais sobre a sintaxe e as descrições da política do Amazon SQS, consulte a [Referência de política do IAM da AWS](#) no Guia do usuário do IAM.

Para ver uma tabela com todas as ações do Amazon Simple Queue Service e os recursos a que elas se aplicam, consulte [Permissões da API do Amazon SQS: referência de ações e recurso](#).

Como o Amazon Simple Queue Service funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon SQS, entenda que recursos do IAM estão disponíveis para uso com o Amazon SQS.

Recursos do IAM que você pode usar com o Amazon Simple Queue Service

Recurso do IAM	Compatibilidade com o Amazon SQS
Políticas baseadas em identidade	Sim
Políticas baseadas em atributos	Sim
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
ABAC (tags em políticas)	Parcial

Recurso do IAM	Compatibilidade com o Amazon SQS
Credenciais temporárias	Sim
Sessões de acesso direto (FAS)	Sim
Perfis de serviço	Sim
Perfis vinculados a serviço	Não

Para obter uma visão de alto nível de como o Amazon SQS e AWS outros serviços funcionam com a maioria dos recursos do IAM, [AWS consulte os serviços que funcionam com o IAM no Guia](#) do usuário do IAM.

Controle de acesso

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a [visão geral da lista de controle de acesso \(ACL\)](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

Note

É importante entender que todos Contas da AWS podem delegar suas permissões aos usuários em suas contas. O acesso entre contas permite que você compartilhe o acesso aos seus AWS recursos sem precisar gerenciar usuários adicionais. Para obter informações sobre como usar o acesso entre contas, consulte [Habilitar o acesso entre contas](#) no Guia do usuário do IAM.

Consulte [Limitações das políticas personalizadas do Amazon SQS](#) para obter mais detalhes sobre permissões de conteúdo cruzado e chaves de condição nas políticas personalizadas do Amazon SQS.

Políticas baseadas em identidade do Amazon SQS

Compatível com políticas baseadas em identidade: sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que podem ser anexados a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte [Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente](#) no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte [Referência de elemento de política JSON do IAM](#) no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para o Amazon SQS

Para ver exemplos de políticas baseadas em identidade do Amazon SQS, consulte [Práticas recomendadas de política](#).

Políticas baseadas em recursos no Amazon SQS

Compatível com políticas baseadas em recursos: sim

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. É necessário [especificar uma entidade principal](#) em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Consulte mais informações em [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

Ações de políticas para o Amazon SQS

Compatível com ações de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista das ações do Amazon SQS, consulte [Tipos de recursos definidos pelo Amazon SQS](#) na Referência de autorização do serviço.

As ações de política no Amazon SQS usam o seguinte prefixo antes da ação:

```
sqs
```

Para especificar várias ações em uma única declaração, separe-as com vírgulas.

```
"Action": [  
    "sqs:action1",  
    "sqs:action2"  
]
```

Para ver exemplos de políticas baseadas em identidade do Amazon SQS, consulte [Práticas recomendadas de política](#).

Recursos de políticas para o Amazon SQS

Compatível com recursos de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu [nome do recurso da Amazon \(ARN\)](#). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Amazon SQS e seus ARNs, consulte [Ações definidas pelo Amazon Simple Queue Service](#) na Referência de autorização de serviço. Para saber com quais

ações você pode especificar o ARN de cada recurso, consulte [Tipos de recursos definidos pelo Amazon SQS](#).

Para ver exemplos de políticas baseadas em identidade do Amazon SQS, consulte [Práticas recomendadas de política](#).

Chaves de condição de política para o Amazon SQS

Compatível com chaves de condição de política específicas de serviço: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem [agentes de condição](#), como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as [chaves de contexto de condição AWS global](#) no Guia do usuário do IAM.

Para ver uma lista de chaves de condição do Amazon SQS, consulte [Ações, recursos e chaves de condição do AWS Key Management Service](#) na Referência de autorização do serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte [Tipos de recursos definidos pelo Amazon SQS](#).

Para ver exemplos de políticas baseadas em identidade do Amazon SQS, consulte [Práticas recomendadas de política](#).

ACLs no Amazon SQS

Suportes ACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com o Amazon SQS

Compatível com ABAC (tags em políticas): parcial

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos chamados de tags. Você pode anexar tags a entidades e AWS recursos

do IAM e, em seguida, criar políticas ABAC para permitir operações quando a tag do diretor corresponder à tag no recurso.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no [elemento de condição](#) de uma política usando as `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para saber mais sobre o ABAC, consulte [Definir permissões com autorização do ABAC](#) no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte [Usar controle de acesso por atributo \(ABAC\)](#) no Guia do usuário do IAM.

Usar credenciais temporárias com o Amazon SQS

Compatível com credenciais temporárias: sim

As credenciais temporárias fornecem acesso de curto prazo aos AWS recursos e são criadas automaticamente quando você usa a federação ou troca de funções. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para ter mais informações, consulte [Credenciais de segurança temporárias no IAM](#) e [Serviços da Serviços da AWS que funcionam com o IAM](#) no Guia do usuário do IAM.

Sessões de acesso direto para o Amazon SQS

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

As sessões de acesso direto (FAS) usam as permissões do principal chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) de fazer solicitações aos serviços posteriores. Para obter detalhes da política ao fazer solicitações de FAS, consulte [Sessões de acesso direto](#).

Perfis de serviço para o Amazon SQS

Compatível com perfis de serviço: sim

O perfil de serviço é um [perfil do IAM](#) que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber

mais, consulte [Criar um perfil para delegar permissões a um AWS service \(Serviço da AWS\)](#) no Guia do Usuário do IAM.

 Warning

A alteração das permissões de um perfil de serviço pode interromper a funcionalidade do Amazon SQS. Edite perfis de serviço somente quando o Amazon SQS fornecer orientação para isso.

Perfis vinculados ao serviço para o Amazon SQS

Compatível com perfis vinculados ao serviço: Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir o perfil de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte [Serviços da AWS que funcionam com o IAM](#). Encontre um serviço na tabela que inclua um Yes na coluna Perfil vinculado ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Atualizações do Amazon SQS para AWS políticas gerenciadas

Para adicionar permissões a usuários, grupos e perfis, é mais fácil usar políticas gerenciadas pela AWS do que gravar políticas por conta própria. É necessário tempo e experiência para [criar políticas gerenciadas pelo cliente do IAM](#) que fornecem à sua equipe apenas as permissões de que precisam. Para começar rapidamente, é possível usar nossas políticas gerenciadas pela AWS . Essas políticas abrangem casos de uso comuns e estão disponíveis na sua conta da AWS . Para obter mais informações sobre políticas AWS gerenciadas, consulte [políticas AWS gerenciadas](#) no Guia do usuário do IAM.

AWS os serviços mantêm e atualizam as políticas AWS gerenciadas. Você não pode alterar as permissões nas políticas AWS gerenciadas. Ocasionalmente, os serviços adicionam permissões

adicionais a uma política AWS gerenciada para oferecer suporte a novos recursos. Esse tipo de atualização afeta todas as identidades (usuários, grupos e funções) em que a política está anexada. É mais provável que os serviços atualizem uma política AWS gerenciada quando um novo recurso é lançado ou quando novas operações são disponibilizadas. Os serviços não removem as permissões de uma política AWS gerenciada, portanto, as atualizações de políticas não violarão suas permissões existentes.

Além disso, AWS oferece suporte a políticas gerenciadas para funções de trabalho que abrangem vários serviços. Por exemplo, a política `ReadOnlyAccess` AWS gerenciada fornece acesso somente de leitura a todos os AWS serviços e recursos. Quando um serviço lança um novo recurso, AWS adiciona permissões somente de leitura para novas operações e recursos. Para obter uma lista e descrições das políticas de perfis de trabalho, consulte [Políticas gerenciadas pela AWS para perfis de trabalho](#) no Guia do usuário do IAM.

AWS política gerenciada: Amazon SQSFull Access

É possível anexar a política `AmazonSQSFullAccess` às suas identidades do Amazon SQS. Essa política concede permissões de acesso total ao Amazon SQS.

Para ver as permissões dessa política, consulte [Amazon SQSFull Access](#) na Referência de políticas AWS gerenciadas.

AWS política gerenciada: Amazon SQSRead OnlyAccess

É possível anexar a política `AmazonSQSReadOnlyAccess` às suas identidades do Amazon SQS. Essa política concede permissões de acesso somente leitura ao Amazon SQS.

Para ver as permissões dessa política, consulte [Amazon SQSRead OnlyAccess](#) na Referência de políticas AWS gerenciadas.

AWS política gerenciada: SQSUnlock QueuePolicy

Se você configurou incorretamente sua política de fila para uma conta membro para negar a todos os usuários o acesso à sua fila do Amazon SQS, você pode usar a política gerenciada para `SQSUnlockQueuePolicy` AWS desbloquear a fila.

Para obter mais informações sobre como remover uma política de fila mal configurada que impede que todos os principais acessem uma fila do Amazon SQS, consulte [Executar uma tarefa privilegiada em uma conta membro no](#) Guia do usuário do IAM. AWS Organizations

Atualizações do Amazon SQS para AWS políticas gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Amazon SQS desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre mudanças nesta página, assine o feed RSS na página [Histórico de documentos](#) do Amazon SQS.

Alteração	Descrição	Data
SQSUnlockQueuePolicy	O Amazon SQS adicionou uma nova política AWS gerenciada chamada <code>SQSUnlockQueuePolicy</code> para desbloquear uma fila e remover uma política de fila mal configurada que impede que todos os principais acessem uma fila do Amazon SQS.	15 de novembro de 2024
AmazonSQSReadOnlyAccess	O Amazon SQS adicionou a ação ListQueueTags , que recupera todas as tags associadas a uma fila especificada do Amazon SQS. Ele permite visualizar os pares de chave-valor que foram atribuídos à fila para fins organizacionais ou de metadados. Essa ação está associada à operação de API <code>ListQueueTags</code> .	20 de junho de 2024
AmazonSQSReadOnlyAccess	O Amazon SQS adicionou uma nova ação que permite listar as tarefas mais recentes de movimentação de	9 de junho de 2023

Alteração	Descrição	Data
	mensagens (até dez) em uma fila de origem específica. Essa ação está associada à operação de API ListMessages geMoveTasks .	

Resolução de problemas de identidade e acesso do Amazon Simple Queue Service

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Amazon SQS e o IAM.

Não tenho autorização para executar uma ação no Amazon SQS

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *my-example-widget* fictício, mas não tem as permissões fictícias do `sqs:GetWidget`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
sqs:GetWidget on resource: my-example-widget
```

Nesse caso, a política de Mateo deve ser atualizada para permitir que ele tenha acesso ao recurso *my-example-widget* usando a ação `sqs:GetWidget`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Caso receba uma mensagem de erro informando que você não tem autorização para executar a ação `iam:PassRole`, as políticas deverão ser atualizadas para permitir a transmissão de um perfil ao Amazon SQS.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no Amazon SQS. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas de fora da minha Conta da AWS acessem meus recursos do Amazon SQS

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon SQS é compatível com esses recursos, consulte [Como o Amazon Simple Queue Service funciona com o IAM](#).
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte [Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui](#) no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte [Como fornecer acesso Contas da AWS a terceiros](#) no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte [Conceder acesso a usuários autenticados externamente \(federação de identidades\)](#) no Guia do usuário do IAM.

- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

Quero desbloquear minha fila

Se você Conta da AWS pertence a uma organização, AWS Organizations as políticas podem impedir que você acesse os recursos do Amazon SQS. Por padrão, AWS Organizations as políticas não bloqueiam nenhuma solicitação para o Amazon SQS. No entanto, certifique-se de que suas AWS Organizations políticas não tenham sido configuradas para bloquear o acesso às filas do Amazon SQS. Para obter instruções sobre como verificar suas AWS Organizations políticas, consulte [Listar todas as políticas](#) no Guia AWS Organizations do usuário.

Além disso, se você configurou incorretamente a política da fila para uma conta de membro a fim de negar a todos os usuários o acesso à fila do Amazon SQS, é possível desbloquear a fila iniciando uma sessão privilegiada para a conta de membro no IAM. Depois de iniciar uma sessão privilegiada, é possível excluir a política de fila configurada incorretamente para recuperar o acesso à fila. Para obter mais informações, consulte [Executar uma tarefa privilegiada em uma conta de AWS Organizations membro](#) no Guia do usuário do IAM.

Usar políticas com o Amazon SQS

Este tópico fornece exemplos de políticas baseadas em identidade em que um administrador de conta pode anexar políticas de permissões a identidades do IAM (usuários, grupos e funções).

Important

Recomendamos analisar primeiro os tópicos introdutórios que explicam os conceitos básicos e as opções disponíveis para gerenciar o acesso aos recursos do Amazon Simple Queue Service. Para obter mais informações, consulte [Visão geral do gerenciamento de acesso no Amazon SQS](#).

Com exceção de `ListQueues`, todas as ações do Amazon SQS oferecem suporte a permissões no nível do recurso. Para obter mais informações, consulte [Permissões da API do Amazon SQS: referência de ações e recurso](#).

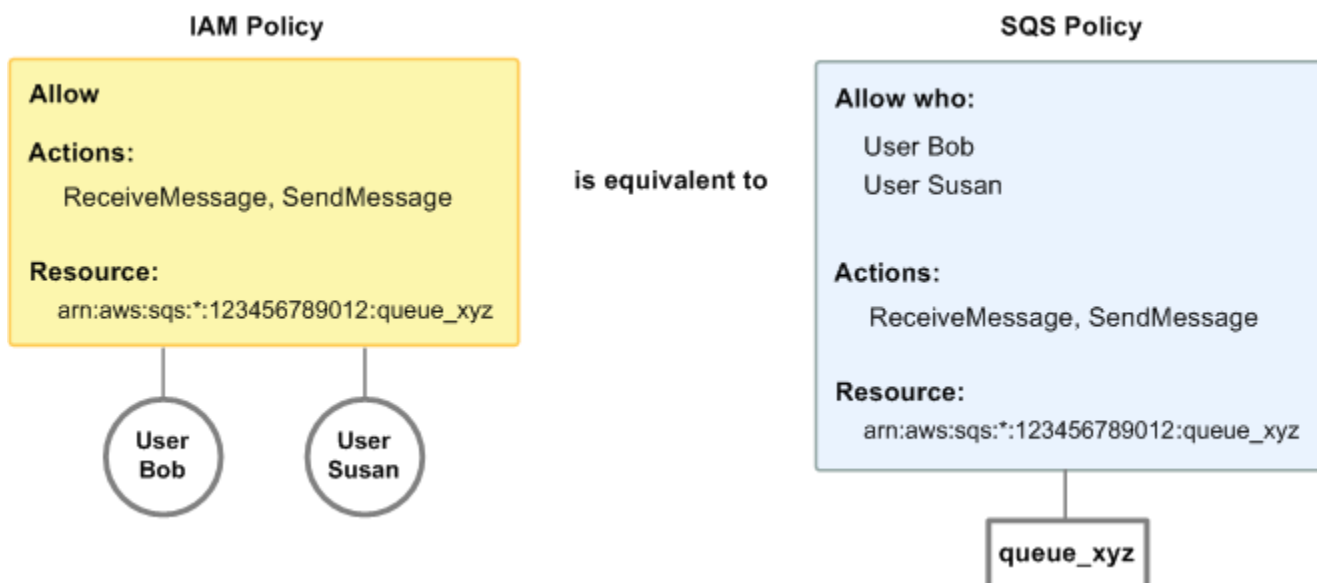
Usar políticas do Amazon SQS e do IAM

Há duas maneiras de conceder aos usuários permissões aos recursos do Amazon SQS: usando o sistema de políticas do Amazon SQS (políticas baseadas em recursos) e usando o sistema de política do IAM (políticas baseadas em identidade). É possível usar um ou os dois métodos, com exceção da ação `ListQueues`, que é uma permissão regional que só pode ser definida em uma política do IAM.

Por exemplo, o diagrama a seguir mostra uma política do IAM e uma política equivalente do Amazon SQS. A política do IAM concede os direitos ao Amazon SQS `ReceiveMessage` e às `SendMessage` ações da fila chamada `queue_xyz` em sua AWS conta, e a política é anexada aos usuários chamados Bob e Susan (Bob e Susan têm as permissões declaradas na política). Essa política do Amazon SQS também oferece a Bob e Susan direitos às ações `ReceiveMessage` e `SendMessage` para a mesma fila.

Note

O exemplo a seguir mostra políticas simples sem condições. Você pode especificar uma determinada condição na política e obter o mesmo resultado.



Há uma grande diferença entre as políticas do IAM e do Amazon SQS: o sistema de políticas do Amazon SQS permite que você conceda permissão para AWS outras contas, enquanto o IAM não.

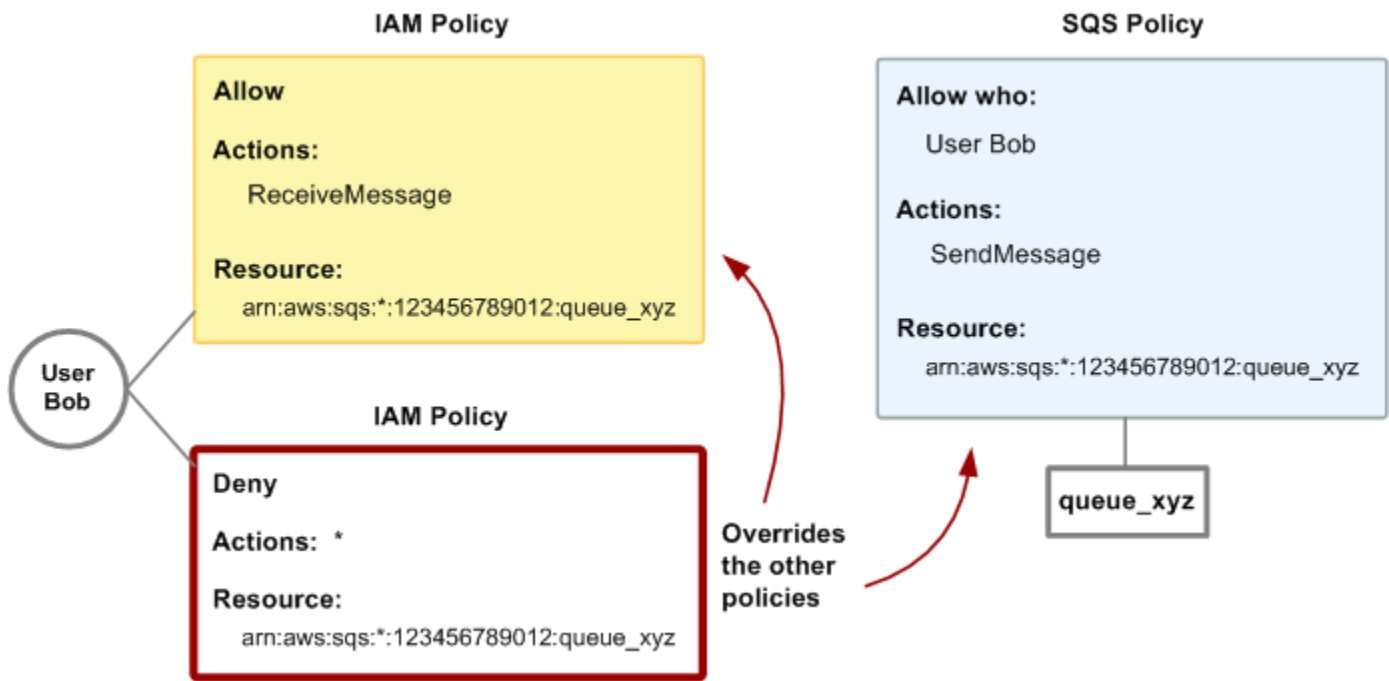
Você é quem decide como usar os dois sistemas para gerenciar suas permissões. Os exemplos a seguir mostram como os dois sistemas de política funcionam em conjunto.

- No primeiro exemplo, Bob tem uma política do IAM e uma do Amazon SQS que se aplicam à sua conta. A política do IAM concede à sua conta permissão para a ação `ReceiveMessage` em `queue_xyz`, enquanto a política do Amazon SQS fornece à sua conta permissão para a ação `SendMessage` na mesma fila. O seguinte diagrama ilustra o conceito.



Se Bob enviar uma solicitação `ReceiveMessage` a `queue_xyz`, a política do IAM permitirá a ação. Se Bob enviar uma solicitação `SendMessage` a `queue_xyz`, a política do Amazon SQS permitirá a ação.

- No segundo exemplo, Bob abusa de seu acesso a `queue_xyz`, para que seja necessário remover todo o seu acesso à fila. O mais fácil a fazer é adicionar uma política que negue a ele acesso a todas as ações para a fila. Essa política substitui as outras duas, pois uma `deny` explícita sempre substitui uma `allow`. Para obter mais informações sobre a lógica de avaliação da política, consulte [Usar políticas personalizadas com linguagem de políticas de acesso do Amazon SQS](#). O seguinte diagrama ilustra o conceito.



Você também pode adicionar outra instrução à política do Amazon SQS que nega a Bob qualquer tipo de acesso à fila. Ela tem o mesmo efeito que a adição de uma política do IAM que nega o acesso de Bob à fila. Para obter exemplos de políticas que abrangem ações e recursos do Amazon SQS, consulte [Exemplos básicos de políticas do Amazon SQS](#). Para obter mais informações sobre como elaborar políticas do Amazon SQS, consulte [Usar políticas personalizadas com linguagem de políticas de acesso do Amazon SQS](#).

Permissões necessárias para usar o console do Amazon SQS

Um usuário que queira trabalhar com o console do Amazon SQS deve ter o conjunto mínimo de permissões para trabalhar com as filas do Amazon SQS na Conta da AWS do usuário. Por exemplo, o usuário deve ter a permissão para chamar a ação `ListQueues` a fim de poder listar as filas, ou a permissão para chamar a ação `CreateQueue` para poder criar filas. Além de permissões do Amazon SQS, para inscrever uma fila do Amazon SQS em um tópico do Amazon SNS, o console também exige permissões para ações do Amazon SNS.

Se você criar uma política do IAM que seja mais restritiva que as permissões mínimas necessárias, o console poderá não funcionar como pretendido para os usuários com essa política do IAM.

Você não precisa permitir permissões mínimas do console para usuários que fazem chamadas somente para as ações AWS CLI ou para as ações do Amazon SQS.

Exemplos de políticas baseadas em identidade para o Amazon SQS

Por padrão, os usuários e perfis não têm permissão para criar ou modificar recursos do Amazon SQS. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte [Criar políticas do IAM \(console\)](#) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Amazon SQS, incluindo o formato de cada um dos tipos de recursos, consulte [Ações, recursos e chaves de condição ARNs para o Amazon Simple Queue Service](#) na Referência de autorização de serviço.

Note

Ao configurar ganchos de ciclo de vida para o Amazon EC2 Auto Scaling, você não precisa escrever uma política para enviar mensagens para uma fila do Amazon SQS. Para obter mais informações, consulte [Amazon EC2 Auto Scaling Lifecycle Hooks](#) no Guia do usuário da Amazon. EC2

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon SQS em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e passe para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte [Políticas gerenciadas pela AWS](#) ou [Políticas gerenciadas pela AWS para funções de trabalho](#) no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte [Políticas e permissões no IAM](#) no Guia do usuário do IAM.

- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte [Elementos da política JSON do IAM: condição](#) no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte [Validação de políticas do IAM Access Analyzer](#) no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte [Configuração de acesso à API protegido por MFA](#) no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte [Práticas recomendadas de segurança no IAM](#) no Guia do usuário do IAM.

Usar o console do Amazon SQS

Para acessar o console do Amazon Simple Queue Service, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Amazon SQS em seu. Conta da AWS Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam à operação de API que estiverem tentando executar.

Para garantir que usuários e funções ainda possam usar o console do Amazon SQS, anexe também a política gerenciada do Amazon `AmazonSQSReadOnlyAccess` AWS SQS às entidades. Para obter informações, consulte [Adicionar permissões a um usuário](#) no Guia do usuário do IAM.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou AWS .

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```


Permitir que um usuário crie filas

No exemplo a seguir, criamos uma política para Bob que permite acessar todas as ações do Amazon SQS, mas apenas com as filas cujos nomes sejam prefixados com a string literal `alice_queue_`.

O Amazon SQS não concede automaticamente ao criador de uma fila permissões para usá-la. Portanto, é preciso conceder explicitamente permissões a Bob para usar todas as ações do Amazon SQS, além da ação `CreateQueue` na política do IAM.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sqs:*",
    "Resource": "arn:aws:sqs:*:123456789012:alice_queue_*"
  }]
}
```

Permitir que os desenvolvedores gravem mensagens em uma fila compartilhada

No exemplo a seguir, criamos um grupo para desenvolvedores e anexamos uma política que permite que o grupo use a `SendMessage` ação do Amazon SQS, mas somente com a fila que pertence ao especificado Conta da AWS e é nomeada. `MyCompanyQueue`

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sqs:SendMessage",
    "Resource": "arn:aws:sqs:*:123456789012:MyCompanyQueue"
  }]
}
```

Você pode usar `*` em vez de `SendMessage` para conceder as seguintes ações a um principal em uma fila compartilhada: `ChangeMessageVisibility`, `DeleteMessage`, `GetQueueAttributes`, `GetQueueUrl`, `ReceiveMessage` e `SendMessage`.

Note

Embora `*` inclua o acesso fornecido por outros tipos de permissão, o Amazon SQS considera as permissões separadamente. Por exemplo, é possível conceder permissões `*` e `SendMessage` a um usuário, embora `*` inclua o acesso fornecido pelo `SendMessage`. Esse conceito também se aplica quando você remove uma permissão. Se uma entidade principal tiver apenas uma permissão `*`, a solicitação de remoção de uma permissão `SendMessage` não deixará a entidade principal com uma permissão do tipo tudo, exceto. Em vez disso, a solicitação não fará nada, pois a entidade principal não tinha uma permissão `SendMessage` explícita. Para deixar a entidade principal apenas com a permissão `ReceiveMessage`, primeiro adicione a permissão `ReceiveMessage` e remova a permissão `*`.

Permitir que gerentes obtenham o tamanho geral de filas

No exemplo a seguir, criamos um grupo para gerentes e anexamos uma política que permite que o grupo use a `GetQueueAttributes` ação do Amazon SQS com todas as filas que pertencem à conta especificada. AWS

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sqs:GetQueueAttributes",
    "Resource": "*"
  }]
}
```

Permitir que um parceiro envie mensagens a uma fila específica

Você pode realizar essa tarefa usando uma política do Amazon SQS ou do IAM. Se seu parceiro tiver uma Conta da AWS, talvez seja mais fácil usar uma política do Amazon SQS. No entanto, qualquer usuário da empresa do parceiro que possua as credenciais AWS de segurança pode enviar mensagens para a fila. Para limitar o acesso a um determinado usuário ou aplicação, você deve tratar o parceiro como um usuário em sua própria empresa e usar uma política do IAM em vez de uma política do Amazon SQS.

Esse exemplo executa as seguintes ações:

1. Crie um grupo chamado WidgetCo para representar a empresa parceira.
2. Criar um usuário para o usuário ou aplicativo específico na empresa do parceiro que precisa de acesso.
3. Adicione o usuário ao grupo .
4. Associe uma política que ofereça ao grupo acesso apenas à ação SendMessage somente para a fila denominada WidgetPartnerQueue.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sqs:SendMessage",
    "Resource": "arn:aws:sqs:*:123456789012:WidgetPartnerQueue"
  }]
}
```

Exemplos básicos de políticas do Amazon SQS

Esta seção mostra exemplos de políticas para casos de uso comuns do Amazon SQS.

Você pode usar o console para verificar os efeitos de cada política à medida que anexa a política ao usuário. Inicialmente, o usuário não tem permissões e, portanto, não poderá fazer nada no console. À medida que você anexar políticas ao usuário, poderá verificar se o usuário pode executar várias ações no console.

 Note

Recomendamos que você use duas janelas do navegador: uma para conceder permissões e outra para fazer login Console de gerenciamento da AWS usando as credenciais do usuário para verificar as permissões à medida que você as concede ao usuário.

Exemplo 1: Conceder uma permissão a um Conta da AWS

O exemplo de política a seguir Conta da AWS concede 111122223333 ao número a SendMessage permissão para a fila nomeada 444455556666/queue1 na região Leste dos EUA (Ohio).

JSON

```
{
  "Version": "2012-10-17",
  "Id": "Queue1_Policy_UUID",
  "Statement": [{
    "Sid": "Queue1_SendMessage",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "111122223333"
      ]
    },
    "Action": "sqs:SendMessage",
    "Resource": "arn:aws:sqs:us-east-2:444455556666:queue1"
  }]
}
```

Exemplo 2: conceder duas permissões a uma Conta da AWS

O exemplo de política a seguir Conta da AWS concede o número 111122223333 SendMessage e a ReceiveMessage permissão para a fila nomeada 444455556666/queue1.

JSON

```
{
  "Version": "2012-10-17",
```

```

    "Id": "Queue1_Policy_UUID",
    "Statement": [{
      "Sid": "Queue1_Send_Receive",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "111122223333"
        ]
      },
      "Action": [
        "sqs:SendMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": "arn:aws:sqs:*:444455556666:queue1"
    }]
  }

```

Exemplo 3: Conceder todas as permissões a dois Contas da AWS

O exemplo de política a seguir concede dois Contas da AWS números diferentes (111122223333e444455556666) permissão para usar todas as ações às quais o Amazon SQS permite acesso compartilhado para a fila nomeada 123456789012/queue1 na região Leste dos EUA (Ohio).

JSON

```

{
  "Version": "2012-10-17",
  "Id": "Queue1_Policy_UUID",
  "Statement": [{
    "Sid": "Queue1_AllActions",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "111122223333",
        "444455556666"
      ]
    },
    "Action": "sqs:*",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:queue1"
  }]
}

```

```
}
```

Exemplo 4: conceder permissões entre contas a um perfil e um nome de usuário

O exemplo de política a seguir concede `role1` e `username1` sob Conta da AWS número, permissão `111122223333` entre contas para usar todas as ações às quais o Amazon SQS permite acesso compartilhado à fila `123456789012/queue1` nomeada na região Leste dos EUA (Ohio).

As permissões entre contas não se aplicam às seguintes ações:

- [AddPermission](#)
- [CancelMessageMoveTask](#)
- [CreateQueue](#)
- [DeleteQueue](#)
- [ListMessageMoveTask](#)
- [ListQueues](#)
- [ListQueueTags](#)
- [RemovePermission](#)
- [SetQueueAttributes](#)
- [StartMessageMoveTask](#)
- [TagQueue](#)
- [UntagQueue](#)

JSON

```
{
  "Version": "2012-10-17",
  "Id": "Queue1_Policy_UUID",
  "Statement": [{
    "Sid": "Queue1_AllActions",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "arn:aws:iam::111122223333:role/role1",
        "arn:aws:iam::111122223333:user/username1"
      ]
    }
  ]
}
```

```
    },  
    "Action": "sqs:*",  
    "Resource": "arn:aws:sqs:us-east-2:123456789012:queue1"  
  }  
}
```

Exemplo 5: conceder uma permissão a todos os usuários

O exemplo de política a seguir concede a todos os usuários (anônimos) a permissão `ReceiveMessage` para a fila denominada `111122223333/queue1`.

JSON

```
{  
  "Version": "2012-10-17",  
  "Id": "Queue1_Policy_UUID",  
  "Statement": [{  
    "Sid": "Queue1_AnonymousAccess_ReceiveMessage",  
    "Effect": "Allow",  
    "Principal": "*",  
    "Action": "sqs:ReceiveMessage",  
    "Resource": "arn:aws:sqs:*:111122223333:queue1"  
  }]  
}
```

Exemplo 6: conceder uma permissão de tempo limitado a todos os usuários

O exemplo de política a seguir concede a todos os usuários (anônimos) a permissão `ReceiveMessage` para a fila denominada `111122223333/queue1`, mas apenas das 12h (meio dia) às 15h em 31 de janeiro de 2009.

JSON

```
{  
  "Version": "2012-10-17",  
  "Id": "Queue1_Policy_UUID",  
  "Statement": [{  
    "Sid": "Queue1_AnonymousAccess_ReceiveMessage_TimeLimit",  
    "Effect": "Allow",  
    "Action": "sqs:ReceiveMessage",  
    "Resource": "arn:aws:sqs:*:111122223333:queue1",  
    "Condition": {  
      "DateEquals": {  
        "sqs:ReceiveMessage": "2009-01-31T12:00:00Z",  
        "sqs:ReceiveMessage": "2009-01-31T15:00:00Z"  
      }  
    }  
  }]  
}
```

```

    "Principal": "*",
    "Action": "sqs:ReceiveMessage",
    "Resource": "arn:aws:sqs:*:111122223333:queue1",
    "Condition" : {
        "DateGreaterThan" : {
            "aws:CurrentTime": "2009-01-31T12:00Z"
        },
        "DateLessThan" : {
            "aws:CurrentTime": "2009-01-31T15:00Z"
        }
    }
}
}]
}

```

Exemplo 7: conceder todas as permissões a todos os usuários em um intervalo CIDR

A política de exemplo a seguir concede a todos os usuários (anônimos) permissão para usar todas as ações possíveis do Amazon SQS que podem ser compartilhadas para a fila denominada 111122223333/queue1, mas somente se a solicitação vier do intervalo CIDR 192.0.2.0/24.

JSON

```

{
    "Version": "2012-10-17",
    "Id": "Queue1_Policy_UUID",
    "Statement": [{
        "Sid": "Queue1_AnonymousAccess_AllActions_AllowlistIP",
        "Effect": "Allow",
        "Principal": "*",
        "Action": "sqs:*",
        "Resource": "arn:aws:sqs:*:111122223333:queue1",
        "Condition" : {
            "IpAddress" : {
                "aws:SourceIp": "192.0.2.0/24"
            }
        }
    }]
}

```


Exemplo 8: lista de permissões e lista de bloqueios para usuários em diferentes intervalos CIDR

O exemplo de política a seguir contém duas instruções:

- A primeira instrução concede a todos os usuários (anônimos) no intervalo 192.0.2.0/24 CIDR (exceto para 192.0.2.188) permissão para usar a ação `SendMessage` para a fila denominada 111122223333/queue1.
- A segunda instrução impede que todos os usuários (anônimos) no intervalo CIDR 12.148.72.0/23 usem a fila.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "Queue1_Policy_UUID",
  "Statement": [{
    "Sid": "Queue1_AnonymousAccess_SendMessage_IPLimit",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "sqs:SendMessage",
    "Resource": "arn:aws:sqs:*:111122223333:queue1",
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": "192.0.2.0/24"
      },
      "NotIpAddress": {
        "aws:SourceIp": "192.0.2.188/32"
      }
    }
  }, {
    "Sid": "Queue1_AnonymousAccess_AllActions_IPLimit_Deny",
    "Effect": "Deny",
    "Principal": "*",
    "Action": "sqs:*",
    "Resource": "arn:aws:sqs:*:111122223333:queue1",
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": "12.148.72.0/23"
      }
    }
  }
}]
```

```
}
```

Usar políticas personalizadas com linguagem de políticas de acesso do Amazon SQS

Para conceder permissões básicas (como [SendMessage](#) ou [ReceiveMessage](#)) com base somente em uma Conta da AWS ID, você não precisa criar uma política personalizada. Em vez disso, use a ação [AddPermission](#) do Amazon SQS.

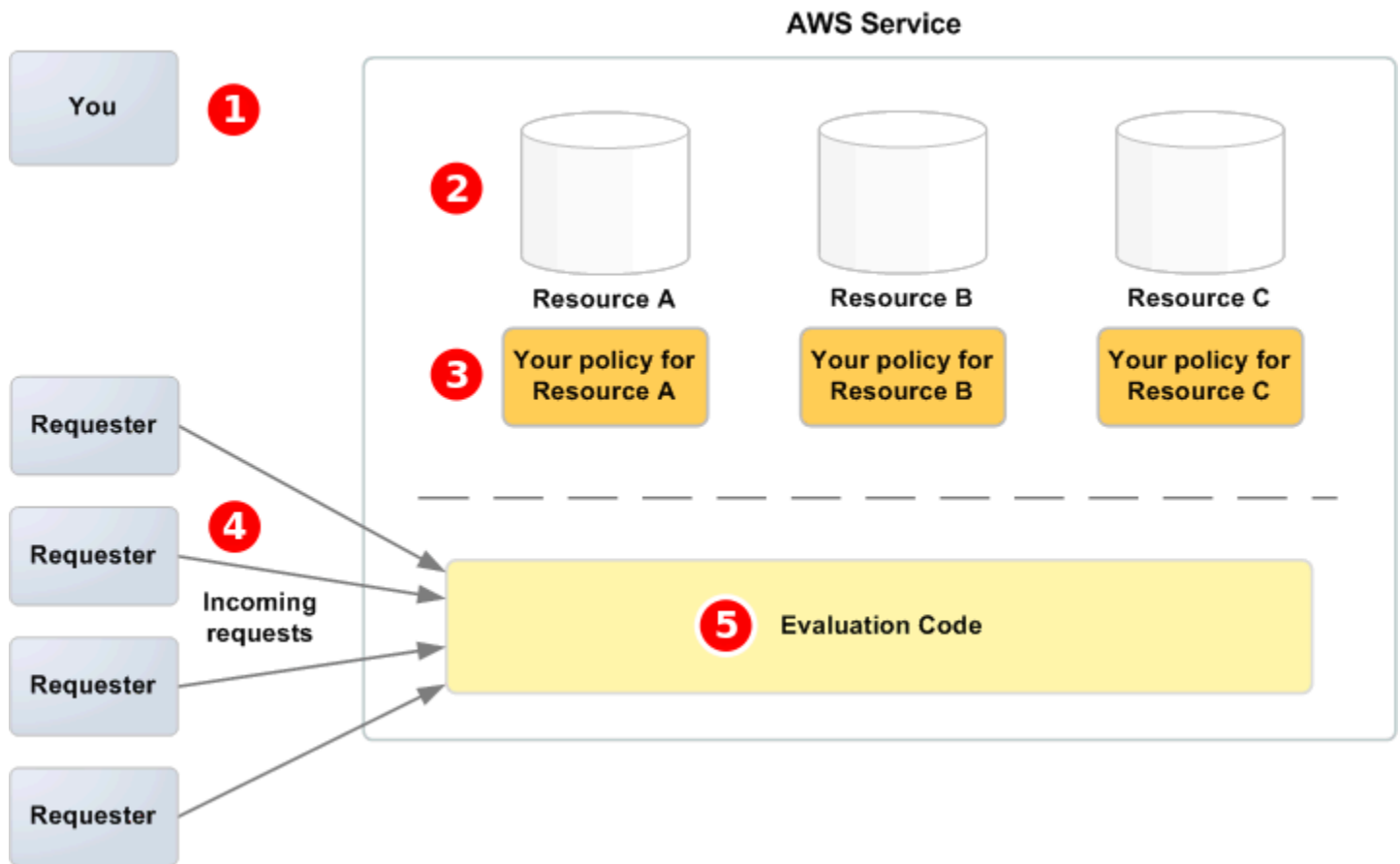
Para permitir ou negar o acesso com base em condições específicas, como o horário da solicitação ou o endereço IP do solicitante, você deve criar uma política personalizada do Amazon SQS e carregá-la usando [SetQueueAttributes](#) ação.

Tópicos

- [Arquitetura do controle de acesso do Amazon SQS](#)
- [Fluxo de trabalho do processo de controle de acesso do Amazon SQS](#)
- [Conceitos-chave da linguagem de políticas de acesso do Amazon SQS](#)
- [Lógica de avaliação da linguagem de políticas de acesso do Amazon SQS](#)
- [Relações entre negações explícitas e padrão na linguagem de políticas de acesso do Amazon SQS](#)
- [Limitações das políticas personalizadas do Amazon SQS](#)
- [Exemplos de linguagem de políticas de acesso do Amazon SQS personalizadas](#)

Arquitetura do controle de acesso do Amazon SQS

O diagrama a seguir descreve o controle de acesso para seus recursos do Amazon SQS.

**1**

Você, o proprietário do recurso.

2

recursos contidos no AWS serviço (por exemplo, filas do Amazon SQS).

Seus

3

Suas políticas. É uma boa prática ter uma política por recurso. O AWS serviço fornece uma API que você usa para carregar e gerenciar suas políticas.

4

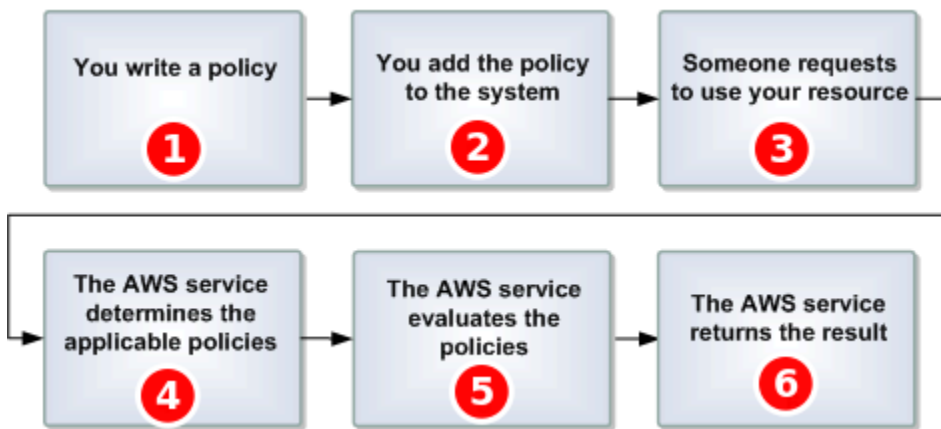
Os solicitantes e suas solicitações de entrada para o serviço da AWS .

5

O código de avaliação da linguagem de políticas de acesso. Esse é o conjunto de códigos dentro do AWS serviço que avalia as solicitações recebidas em relação às políticas aplicáveis e determina se o solicitante tem permissão para acessar o recurso.

Fluxo de trabalho do processo de controle de acesso do Amazon SQS

O diagrama a seguir descreve o fluxo de trabalho geral do controle de acesso com a linguagem de políticas de acesso do Amazon SQS.



1

Você cria uma política do Amazon SQS para a fila.

2

Você carrega sua política para AWS. O AWS serviço fornece uma API que você usa para carregar suas políticas. Por exemplo, você pode usar a ação `SetQueueAttributes` do Amazon SQS com a finalidade de fazer upload de uma política para uma determinada fila do Amazon SQS.

3

Alguém envia uma solicitação para usar sua fila do Amazon SQS.

4

O Amazon SQS examina todas as políticas do Amazon SQS disponíveis e determina quais são aplicáveis.

5

O Amazon SQS avalia as políticas e determina se o solicitante tem permissão para usar sua fila.

6

Com base no resultado da avaliação da política, o Amazon SQS retorna um erro `Access denied` para o solicitante ou continua a processar a solicitação.

Conceitos-chave da linguagem de políticas de acesso do Amazon SQS

Para criar suas próprias políticas, você deve estar familiarizado com [JSON](#) e um número de conceitos-chave.

Permitir

O resultado de uma [Instrução](#) que tenha [Efeito](#) definido como allow.

Ação

A atividade que o [Principal](#) tem permissão para executar, normalmente uma solicitação para a AWS.

Negação padrão

O resultado de uma [Instrução](#) que não tem as configurações [Permitir](#) e [Negação explícita](#).

Condição

Qualquer restrição ou detalhe sobre uma [Permissão](#). Condições comuns são relacionadas a data e hora e a endereços IP.

Efeito

O resultado que você deseja que a [Instrução](#) de uma [Política](#) retorne no momento da avaliação. Você especifica o valor deny ou allow ao gravar a declaração de política. Há três resultados possíveis no momento da avaliação da política: [Negação padrão](#), [Permitir](#) e [Negação explícita](#).

Negação explícita

O resultado de uma [Instrução](#) que tenha [Efeito](#) definido como deny.

Avaliação

O processo que o Amazon SQS usa para determinar se uma solicitação recebida deve ser negada ou permitida com base em uma [Política](#).

Emissor

O usuário que grava uma [Política](#) para conceder permissões a um recurso. O emissor, por definição, é sempre o proprietário do recurso. AWS não permite que os usuários do Amazon SQS criem políticas para recursos que não possuem.

Chave

A característica específica que é a base para a restrição de acesso.

Permissão

O conceito de permissão ou não de acesso a um recurso usando uma [Condição](#) e uma [Chave](#).

Política

O documento que atua como um contêiner para uma ou mais [instruções](#).



O Amazon SQS usa a política para determinar se deverá conceder acesso a um usuário para um recurso.

Principal

O usuário que recebe [Permissão](#) na [Política](#).

Recurso

O objeto ao qual a [Principal](#) solicita acesso.

Instrução

A descrição formal de uma única permissão, escrita na linguagem de políticas de acesso como parte de um documento de [Política](#) mais amplo.

Solicitante

O usuário que envia uma solicitação de acesso a um [Recurso](#).

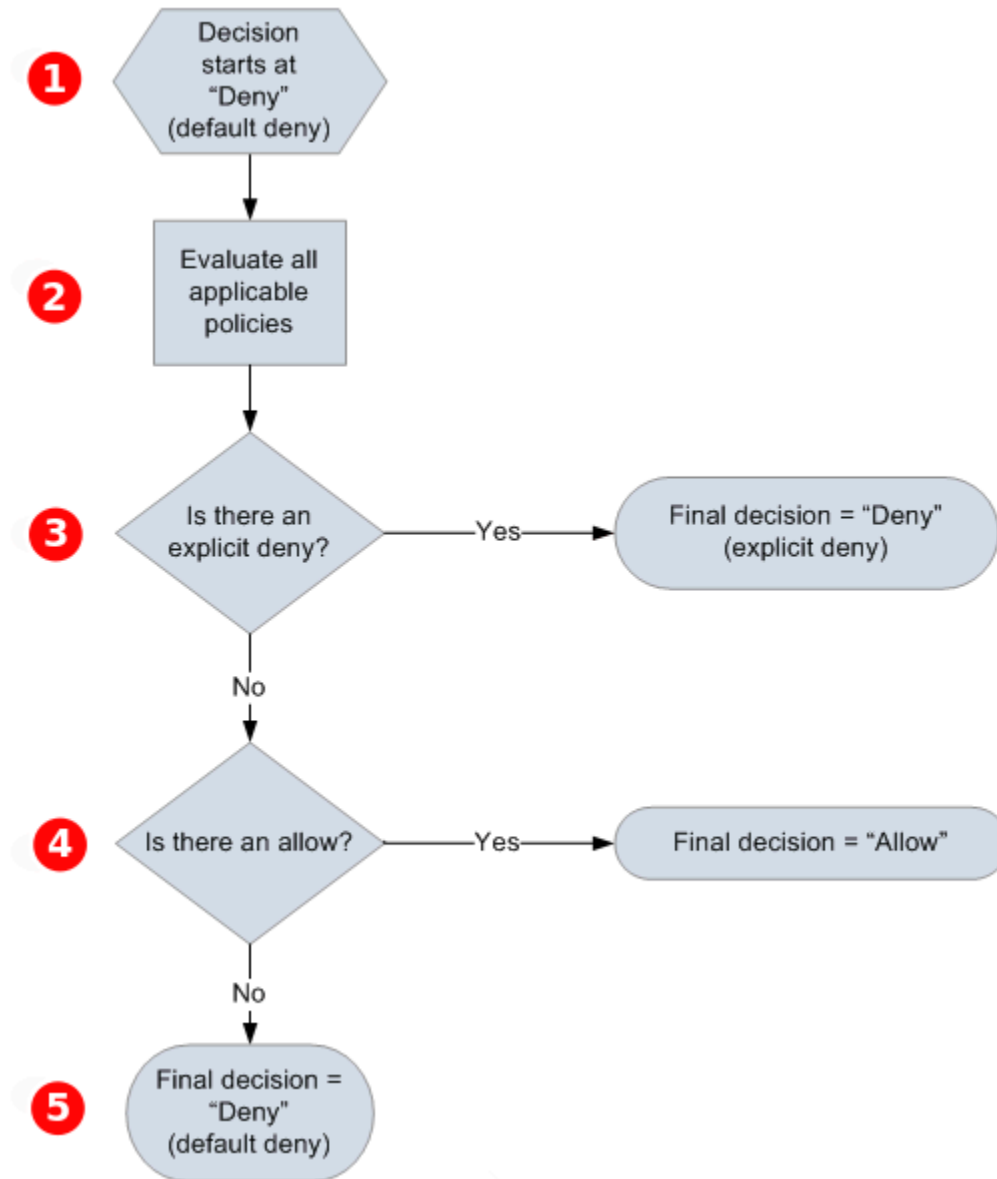
Lógica de avaliação da linguagem de políticas de acesso do Amazon SQS

No momento da avaliação, o Amazon SQS determina se uma solicitação de alguém que não seja o proprietário do recurso deve ser permitida ou negada. A lógica de avaliação segue várias regras básicas:

- Por padrão, todas as solicitações para usar o recurso que venham de outras pessoas, e não de você, serão negadas.

- Um [Permitir](#) substitui qualquer [Negação padrão](#).
- Uma [Negação explícita](#) substitui qualquer permissão.
- A ordem em que as políticas são avaliadas não é importante.

O diagrama a seguir descreve em detalhes como o Amazon SQS avalia as decisões sobre permissões de acesso.



- 1 A decisão começa com uma negação padrão.

2

O código de aplicação avalia todas as políticas que são aplicáveis à solicitação (com base no recurso, na entidade principal, na ação e nas condições). A ordem em que o código de aplicação avalia as políticas não é importante.

3

O código de imposição procura uma instrução de negação explícita que possa ser aplicada à solicitação. Se encontrar um código, o código de aplicação retornará uma decisão de negação e o processo será concluído.

4

Se nenhuma instrução de negação explícita for encontrada, o código de imposição procurará qualquer instrução de permissão que possa ser aplicada à solicitação. Se encontrar uma instrução "permitir", o código de aplicação retornará uma decisão de permitir, e o processo será concluído (o serviço continua a processar a solicitação).

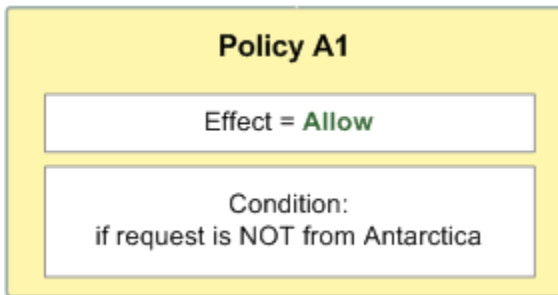
5

Se nenhuma instrução de permissão for encontrada, a decisão final será negar (como não há nenhuma negação explícita ou permissão, isso é considerado uma negação padrão).

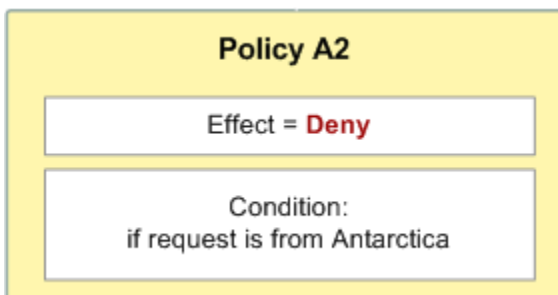
Relações entre negações explícitas e padrão na linguagem de políticas de acesso do Amazon SQS

Se uma política do Amazon SQS não se aplicar diretamente a uma solicitação, esta resultará em uma [Negação padrão](#). Por exemplo, se um usuário solicitar permissão para usar o Amazon SQS, mas a única política que se aplica a ele puder usar o DynamoDB, as solicitações resultarão em uma negação padrão.

Se uma condição em uma instrução não for atendida, a solicitação resultará em uma negação padrão. Se todas as condições em uma instrução forem atendidas, a solicitação resultará em [Permitir](#) ou [Negação explícita](#), com base no valor do elemento [Efeito](#) da política. As políticas não especificam o que fazer se uma condição não for atendida, portanto, o resultado padrão nesse caso é uma negação padrão. Por exemplo, digamos que você deseje evitar solicitações da Antártica. Você elabora uma Política A1 que permite uma solicitação apenas se não vier da Antártica. O seguinte diagrama ilustra a política do Amazon SQS.

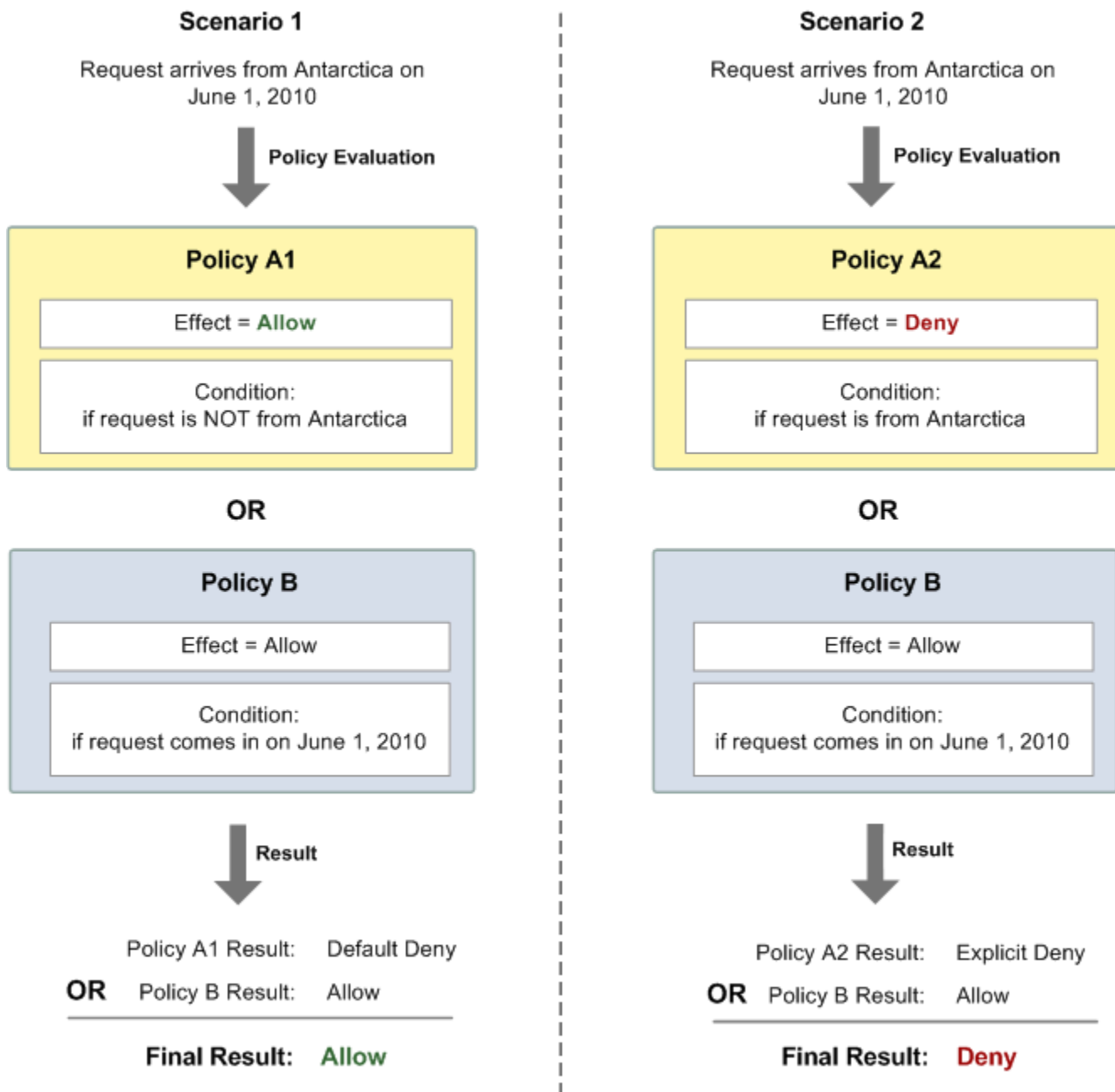


Se um usuário enviar uma solicitação dos EUA, a condição será atendida (a solicitação não será da Antártica) e a solicitação resultará em uma permissão. No entanto, se um usuário enviar uma solicitação da Antártica, a condição não será atendida, e a solicitação será padronizada para uma solicitação padrão. Você pode alterar o resultado para uma negação explícita criando a Política A2 que nega explicitamente uma solicitação quando ela for recebida da Antártica. O seguinte diagrama ilustra a política.



Se um usuário enviar uma solicitação da Antártica, a condição será atendida, e a solicitação resultará em uma negação explícita.

A distinção entre uma negação padrão e uma negação explícita é importante porque uma permissão pode substituir a anterior, mas não a última. Por exemplo, a Política B permite solicitações que cheguem em 1º de junho de 2010. O diagrama a seguir compara a combinação dessa política com Política A1 e Política A2.



No Cenário 1, a Política A1 resulta em uma negação padrão e a Política B resulta em uma permissão porque a política permite solicitações recebidas em 1º de junho de 2010. A permissão da Política B substitui a negação padrão da Política A1, e a solicitação é permitida.

No Cenário 2, a Política B2 resulta em uma negação explícita, e a Política B resulta em uma permissão. A negação explícita da Política A2 substitui a permissão da Política B, e a solicitação é negada.

Limitações das políticas personalizadas do Amazon SQS

Acesso entre contas

As permissões entre contas não se aplicam às seguintes ações:

- [AddPermission](#)
- [CancelMessageMoveTask](#)
- [CreateQueue](#)
- [DeleteQueue](#)
- [ListMessageMoveTask](#)
- [ListQueues](#)
- [ListQueueTags](#)
- [RemovePermission](#)
- [SetQueueAttributes](#)
- [StartMessageMoveTask](#)
- [TagQueue](#)
- [UntagQueue](#)

Chaves de condição

Atualmente, o Amazon SQS suporta apenas um subconjunto limitado de [chaves de condições disponíveis no IAM](#). Para obter mais informações, consulte [Permissões da API do Amazon SQS: referência de ações e recurso](#).

Exemplos de linguagem de políticas de acesso do Amazon SQS personalizadas

Os seguintes são exemplos de políticas de acesso típicas do Amazon SQS.

Exemplo 1: conceder permissão a uma conta

O exemplo de política do Amazon SQS a seguir concede à Conta da AWS 111122223333 permissão para enviar e receber da queue2, de propriedade da Conta da AWS 444455556666.

JSON

```
{
```

```
"Version": "2012-10-17",
"Id": "UseCase1",
"Statement" : [{
  "Sid": "1",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "111122223333"
    ]
  },
  "Action": [
    "sqs:SendMessage",
    "sqs:ReceiveMessage"
  ],
  "Resource": "arn:aws:sqs:us-east-2:444455556666:queue2"
}]
}
```

Exemplo 2: conceder permissão a uma ou mais contas

O exemplo a seguir da política do Amazon SQS dá a um ou mais Contas da AWS acesso às filas pertencentes à sua conta por um período de tempo específico. É necessário criar essa política e fazer upload no Amazon SQS usando a ação [SetQueueAttributes](#), porque a ação [AddPermission](#) não permite especificar uma restrição de tempo ao conceder acesso a uma fila.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "UseCase2",
  "Statement" : [{
    "Sid": "1",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "111122223333",
        "444455556666"
      ]
    },
    "Action": [
      "sqs:SendMessage",
```

```

    "sqs:ReceiveMessage"
  ],
  "Resource": "arn:aws:sqs:us-east-2:444455556666:queue2",
  "Condition": {
    "DateLessThan": {
      "AWS:CurrentTime": "2009-06-30T12:00Z"
    }
  }
}
}

```

Exemplo 3: dê permissão para solicitações de EC2 instâncias da Amazon

O exemplo a seguir da política do Amazon SQS dá acesso às solicitações provenientes de instâncias da Amazon EC2 . Esse exemplo se baseia no exemplo "[Exemplo 2: conceder permissão a uma ou mais contas](#)": ele restringe o acesso a antes de 30 de junho de 2009 ao meio-dia (UTC), que restringe o acesso ao intervalo de IP 203.0.113.0/24. É necessário criar essa política e fazer upload no Amazon SQS usando a ação [SetQueueAttributes](#), porque a ação [AddPermission](#) não permite especificar uma restrição de endereço IP ao conceder acesso a uma fila.

JSON

```

{
  "Version": "2012-10-17",
  "Id": "UseCase3",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "111122223333"
        ]
      },
      "Action": [
        "sqs:SendMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": "arn:aws:sqs:us-east-2:444455556666:queue2",
      "Condition": {
        "DateLessThan": {
          "AWS:CurrentTime": "2009-06-30T12:00Z"
        }
      }
    }
  ]
}

```

```

    },
    "IpAddress": {
      "AWS:SourceIp": "203.0.113.0/24"
    }
  }
}

```

Exemplo 4: negar acesso a uma conta específica

O exemplo a seguir da política do Amazon SQS nega um Conta da AWS acesso específico à sua fila. Este exemplo se baseia no exemplo [Exemplo 1: conceder permissão a uma conta](#) """: ele nega acesso ao especificado. Conta da AWSÉ necessário criar essa política e fazer upload no Amazon SQS usando a ação [SetQueueAttributes](#), porque a ação [AddPermission](#) não permite negar acesso a uma fila (ela permite apenas conceder acesso a uma fila).

JSON

```

{
  "Version": "2012-10-17",
  "Id": "UseCase4",
  "Statement": [{
    "Sid": "1",
    "Effect": "Deny",
    "Principal": {
      "AWS": [
        "111122223333"
      ]
    },
    "Action": [
      "sqs:SendMessage",
      "sqs:ReceiveMessage"
    ],
    "Resource": "arn:aws:sqs:us-east-2:444455556666:queue2"
  }]
}

```

Exemplo 5: negar o acesso se não vier de um VPC endpoint

O exemplo de política do Amazon SQS a seguir restringe o acesso à queue1: 111122223333 pode realizar as ações [SendMessage](#) e [ReceiveMessage](#) somente do ID de endpoint da VPC vpce-1a2b3c4d (especificado usando a condição `aws:sourceVpce`). Para obter mais informações, consulte [Endpoints da Amazon Virtual Private Cloud para o Amazon SQS](#).

Note

- A condição `aws:sourceVpce` não requer um ARN para o recurso do VPC endpoint, somente o ID do VPC endpoint.
- Você pode modificar o exemplo a seguir para restringir todas as ações para um endpoint da VPC negando todas as ações do Amazon SQS (`sqs:*`) na segunda instrução. No entanto, essa instrução de política determinará que todas as ações (incluindo ações administrativas necessárias para modificar as permissões da fila) deverão ser feitas por meio do VPC endpoint específico definido na política, potencialmente impedindo que o usuário modifique as permissões da fila no futuro.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "UseCase5",
  "Statement": [{
    "Sid": "1",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "111122223333"
      ]
    },
    "Action": [
      "sqs:SendMessage",
      "sqs:ReceiveMessage"
    ],
    "Resource": "arn:aws:sqs:us-east-2:111122223333:queue1"
  ],
  {
```

```
    "Sid": "2",
    "Effect": "Deny",
    "Principal": "*",
    "Action": [
        "sqs:SendMessage",
        "sqs:ReceiveMessage"
    ],
    "Resource": "arn:aws:sqs:us-east-2:111122223333:queue1",
    "Condition": {
        "StringNotEquals": {
            "aws:sourceVpce": "vpce-1a2b3c4d"
        }
    }
}
```

Usar credenciais de segurança temporárias com o Amazon SQS

Além de criar usuários com suas próprias credenciais de segurança, o IAM também permite que você conceda credenciais de segurança temporárias a qualquer usuário, permitindo que ele acesse seus AWS serviços e recursos. Você pode gerenciar os usuários que têm Contas da AWS. Você também pode gerenciar usuários do seu sistema que não têm Contas da AWS (usuários federados). Além disso, os aplicativos que você cria para acessar seus AWS recursos também podem ser considerados “usuários”.

Use essas credenciais de segurança temporárias para fazer solicitações ao Amazon SQS. As bibliotecas de API calculam o valor de assinatura necessário usando essas credenciais para autenticar sua solicitação. Se você enviar solicitações usando credenciais vencidas, o Amazon SQS negará a solicitação.

Note

Você não pode definir uma política com base em credenciais temporárias.

Pré-requisitos

1. Use o IAM para criar credenciais de segurança temporárias:

- Token de segurança

- Access Key ID
 - Secret Access Key
2. Prepare sua string para assinar com o ID da chave de acesso temporária e o token de segurança.
 3. Use a chave de acesso secreta temporária em vez de sua própria chave de acesso secreta para assinar a solicitação de API de consulta.

 Note

Quando você enviar a solicitação de API de consulta assinada, use o ID da chave de acesso temporária em vez de seu próprio ID da chave de acesso e para incluir o token de segurança. Para obter mais informações sobre o suporte do IAM para credenciais de segurança temporárias, consulte Como [conceder acesso temporário aos seus AWS recursos no Guia](#) do usuário do IAM.

Para chamar uma ação de API de consulta do Amazon SQS usando credenciais de segurança temporárias

1. Solicite um token de segurança temporário usando AWS Identity and Access Management o. Para obter mais informações, consulte [Criar credencias de segurança temporárias para habilitar o acesso para usuários do IAM](#) no Guia do usuário do IAM.

O IAM retorna um token de segurança, um ID da chave de acesso e uma chave de acesso secreta.

2. Prepare sua consulta usando o ID da chave de acesso temporária em vez de seu próprio ID da chave de acesso e para incluir o token de segurança. Assine sua solicitação usando a chave de acesso secreta temporária em vez de sua própria.
3. Envie sua string de consulta assinada com o ID da chave de acesso temporária e o token de segurança.

O exemplo a seguir demonstra como usar credenciais de segurança temporárias para autenticar uma solicitação do Amazon SQS. A estrutura de **AUTHPARAMS** depende de como você assina sua solicitação de API. Para obter mais informações, consulte [Assinar solicitações de AWS API](#) na Referência geral da Amazon Web Services.

```
https://sqs.us-east-2.amazonaws.com/  
?Action=CreateQueue  
&DefaultVisibilityTimeout=40  
&QueueName=MyQueue  
&Attribute.1.Name=VisibilityTimeout  
&Attribute.1.Value=40  
&Expires=2020-12-18T22%3A52%3A43PST  
&SecurityToken=wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY  
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE  
&Version=2012-11-05  
&AUTHPARAMS
```

O exemplo a seguir usa credenciais de segurança temporárias para enviar duas mensagens usando a ação `SendMessageBatch`.

```
https://sqs.us-east-2.amazonaws.com/  
?Action=SendMessageBatch  
&SendMessageBatchRequestEntry.1.Id=test_msg_001  
&SendMessageBatchRequestEntry.1.MessageBody=test%20message%20body%201  
&SendMessageBatchRequestEntry.2.Id=test_msg_002  
&SendMessageBatchRequestEntry.2.MessageBody=test%20message%20body%202  
&SendMessageBatchRequestEntry.2.DelaySeconds=60  
&Expires=2020-12-18T22%3A52%3A43PST  
&SecurityToken=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY  
&AWSAccessKeyId=AKIAI44QH8DHBEXAMPLE  
&Version=2012-11-05  
&AUTHPARAMS
```

Gerenciamento de acesso para filas criptografadas do Amazon SQS com políticas de privilégio mínimo

É possível usar o Amazon SQS para trocar dados sigilosos entre aplicações usando a criptografia do lado do servidor (SSE) integrada ao [AWS Key Management Service \(KMS\)](#). Com a integração do Amazon SQS e AWS KMS, você pode gerenciar centralmente as chaves que protegem o Amazon SQS, bem como as chaves que protegem seus outros recursos. AWS

Vários AWS serviços podem atuar como fontes de eventos que enviam eventos para o Amazon SQS. [Para permitir que uma fonte de eventos acesse a fila criptografada do Amazon SQS, você precisa configurar a fila com uma chave gerenciada pelo cliente.](#) AWS KMS Em seguida, use a

política de chaves para permitir que o serviço use os métodos de AWS KMS API necessários. O serviço também exige permissões para autenticar o acesso e permitir que a fila envie eventos. É possível fazer isso usando uma política do Amazon SQS, que é uma política baseada em recursos que você pode usar para controlar o acesso à fila do Amazon SQS e os respectivos dados.

As seções a seguir fornecem informações sobre como controlar o acesso à sua fila criptografada do Amazon SQS por meio da política do Amazon SQS e da política de chaves. AWS KMS As políticas deste guia ajudarão você a alcançar o [privilegio mínimo](#).

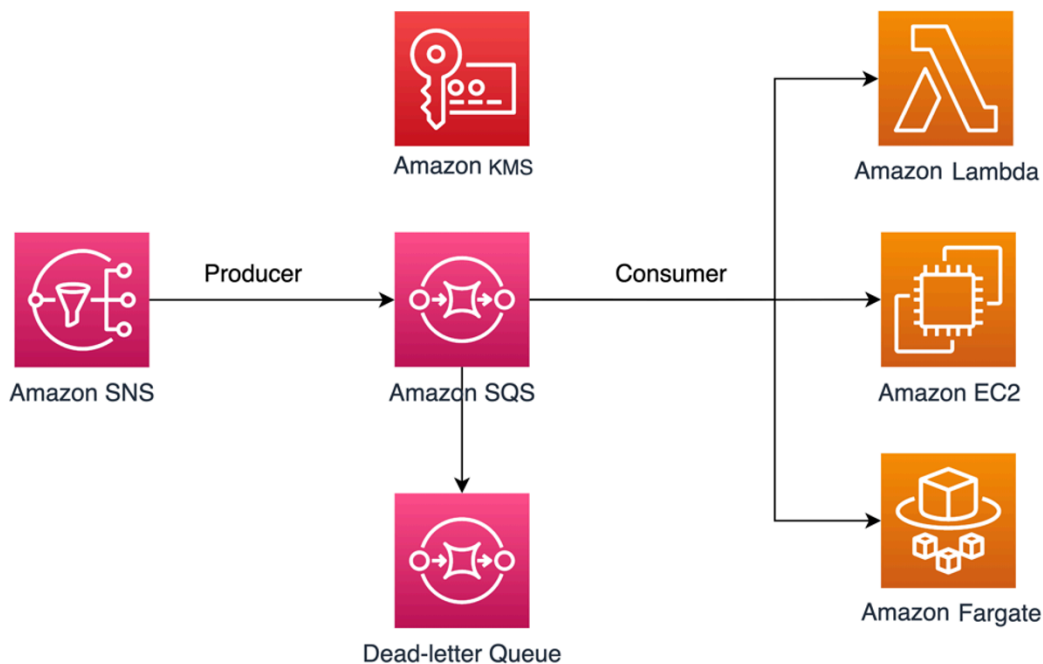
Este guia também descreve como as políticas baseadas em recursos resolvem o [problema de confused-deputy](#) usando as chaves de contexto de condição globais do IAM [aws:SourceArn](#), [aws:SourceAccount](#) e [aws:PrincipalOrgID](#).

Tópicos

- [Visão geral do](#)
- [Política de chaves de privilégio mínimo para o Amazon SQS](#)
- [Declarações de política do Amazon SQS para a fila de mensagens não entregues](#)
- [Prevenção do problema de adjunto confuso entre serviços](#)
- [Usar o IAM Access Analyzer para analisar o acesso entre contas](#)

Visão geral do

Neste tópico, vamos mostrar um caso de uso comum para ilustrar como você pode criar a política de chaves e a política de filas do Amazon SQS. Esse caso de uso é mostrado na imagem a seguir.



Neste exemplo, o produtor de mensagens é um tópico do [Amazon Simple Notification Service \(SNS\)](#), que está configurado para fazer fanout de mensagens para a fila criptografada do Amazon SQS. O consumidor da mensagem é um serviço computacional, como uma [AWS Lambda](#) função, uma instância do [Amazon Elastic Compute Cloud \(EC2\)](#) ou um [AWS Fargate](#) contêiner. Sua fila do Amazon SQS é então configurada para enviar mensagens de falha Amazon a uma [fila de mensagens não entregues \(DLQ\)](#). Isso é útil para depurar seu aplicativo ou sistema de mensagens, pois DLQs permite isolar mensagens não consumidas para determinar por que seu processamento não foi bem-sucedido. Na solução definida neste tópico, um serviço de computação, como uma função do Lambda, é usado para processar mensagens armazenadas na fila do Amazon SQS. Se o consumidor da mensagem estiver localizado em uma nuvem privada virtual (VPC), a declaração de política [DenyReceivingIfNotThroughVPCE](#) incluída neste guia permite restringir o recebimento de mensagens a essa VPC específica.

Note

Este guia contém somente as permissões do IAM necessárias na forma de declarações de política. Para criar a política, você precisa adicionar as declarações à sua política do Amazon SQS ou à sua política de AWS KMS chaves. Este guia não fornece instruções sobre como criar a fila ou a chave do Amazon SQS. AWS KMS Para obter instruções sobre a criação desses recursos, consulte [“Creating an Amazon SQS queue”](#) (Criar uma fila do Amazon SQS) e [Creating keys](#) (Criar chaves).

A política do Amazon SQS definida neste guia não é compatível com o redirecionamento de mensagens diretamente para a mesma fila ou uma fila diferente do Amazon SQS.

Política de chaves de privilégio mínimo para o Amazon SQS

Nesta seção, descrevemos as permissões de privilégio mínimo necessárias AWS KMS para a chave gerenciada pelo cliente que você usa para criptografar sua fila do Amazon SQS. Com essas permissões, você pode limitar o acesso somente às entidades pretendidas e, ao mesmo tempo, implementar o privilégio mínimo. A política principal deve consistir nas seguintes declarações de política, que descrevemos em detalhes abaixo:

- [Conceda permissões de administrador à AWS KMS chave](#)
- [Conceder acesso somente leitura aos metadados de chave](#)
- [Conceder permissões de KMS do Amazon SNS para que este publique mensagens na fila](#)
- [Permitir que os consumidores decodifiquem mensagens da fila](#)

Conceda permissões de administrador à AWS KMS chave

Para criar uma AWS KMS chave, você precisa fornecer permissões de AWS KMS administrador para a função do IAM que você usa para implantar a AWS KMS chave. Essas permissões de administrador são definidas na declaração de política do AllowKeyAdminPermissions a seguir. Ao adicionar essa declaração à sua política de AWS KMS chaves, certifique-se de *<admin-role ARN>* substituí-la pelo Amazon Resource Name (ARN) da função do IAM usada para implantar a AWS KMS chave, gerenciar a AWS KMS chave ou ambas. Essa pode ser o perfil do IAM do pipeline de implantação ou a [função de administrador da sua organização](#) no [AWS Organizations](#).

```
{
  "Sid": "AllowKeyAdminPermissions",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "<admin-role ARN>"
    ]
  },
  "Action": [
    "kms:Create*",
    "kms:Describe*",
```

```

    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:TagResource",
    "kms:UntagResource",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

Note

Em uma política AWS KMS chave, o valor do Resource elemento precisa ser*, o que significa “essa AWS KMS chave”. O asterisco (*) identifica a AWS KMS chave à qual a política de chaves está anexada.

Conceder acesso somente leitura aos metadados de chave

Para conceder a outros perfis do IAM acesso somente leitura aos metadados de chave, adicione a declaração AllowReadAccessToKeyMetaData à política de chaves. Por exemplo, a declaração a seguir permite que você liste todas as AWS KMS chaves em sua conta para fins de auditoria. Essa declaração concede ao usuário AWS raiz acesso somente de leitura aos metadados da chave. Portanto, qualquer entidade principal do IAM na conta pode ter acesso aos metadados de chave quando as respectivas políticas baseadas em identidade tiverem as permissões listadas na seguinte declaração: kms:Describe*, kms:Get* e kms:List*. Certifique-se de *<account-ID>* substituir por suas próprias informações.

```

{
  "Sid": "AllowReadAccesssToKeyMetaData",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::<accountID>:root"
    ]
  }
}
```

```
},
"Action": [
  "kms:Describe*",
  "kms:Get*",
  "kms:List*"
],
"Resource": "*"
}
```

Conceder permissões de KMS do Amazon SNS para que este publique mensagens na fila

Para permitir que o tópico do Amazon SNS publique mensagens na fila criptografada do Amazon SQS, adicione a declaração de política AllowSNSToSendToSQS à sua política de chaves. Essa declaração concede ao Amazon SNS permissões para usar a AWS KMS chave para publicar na sua fila do Amazon SQS. Certifique-se de *<account-ID>* substituir por suas próprias informações.

 Note

A Condition declaração limita o acesso somente ao serviço Amazon SNS na mesma AWS conta.

```
{
  "Sid": "AllowSNSToSendToSQS",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "sns.amazonaws.com"
    ]
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "<account-id>"
    }
  }
}
```

Permitir que os consumidores decodifiquem mensagens da fila

A declaração `AllowConsumersToReceiveFromTheQueue` a seguir concede ao consumidor de mensagens do Amazon SQS as permissões necessárias para descriptografar as mensagens recebidas da fila criptografada do Amazon SQS. Ao anexar a declaração de política, *<consumer's runtime role ARN>* substitua pelo ARN da função de tempo de execução do IAM do consumidor da mensagem.

```
{
  "Sid": "AllowConsumersToReceiveFromTheQueue",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "<consumer's execution role ARN>"
    ]
  },
  "Action": [
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

Política de privilégio mínimo do Amazon SQS

Esta seção mostra as políticas de fila do Amazon SQS de privilégio mínimo para o caso de uso coberto por este guia (por exemplo, do Amazon SNS para o Amazon SQS). A política definida foi projetada para impedir o acesso não intencional usando uma combinação das declarações Deny e Allow. As declarações Allow concedem acesso à entidade ou às entidades pretendidas. As declarações Deny impedem que outras entidades não intencionais acessem a fila do Amazon SQS e exclui a entidade pretendida da condição da política.

A política do Amazon SQS inclui as seguintes declarações, que descrevemos em detalhes abaixo:

- [Restringir as permissões de gerenciamento do Amazon SQS](#)
- [Restringir as ações de fila do Amazon SQS da organização especificada](#)
- [Conceder permissões do Amazon SQS aos consumidores](#)
- [Aplicar a criptografia em trânsito](#)
- [Restringir a transmissão de mensagens para um tópico específico do Amazon SNS](#)
- [\(Opcional\) Restringir o recebimento de mensagens a um endpoint da VPC específico](#)

Restringir as permissões de gerenciamento do Amazon SQS

A declaração de política `RestrictAdminQueueActions` a seguir restringe as permissões de gerenciamento do Amazon SQS somente ao perfil ou aos perfis do IAM que você usa para implantar a fila, gerenciar a fila ou realizar ambas as atividades. Substitua *<placeholder values>* por suas próprias informações. Especifique o ARN da função do IAM usada para implantar a fila do Amazon SQS, bem como o de qualquer função de administrador que ARNs deva ter permissões de gerenciamento do Amazon SQS.

```
{
  "Sid": "RestrictAdminQueueActions",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:AddPermission",
    "sqs:DeleteQueue",
    "sqs:RemovePermission",
    "sqs:SetQueueAttributes"
  ],
  "Resource": "<SQS Queue ARN>",
  "Condition": {
    "StringNotLike": {
      "aws:PrincipalARN": [
        "arn:aws:iam::<account-id>:role/<deployment-role-name>",
        "<admin-role ARN>"
      ]
    }
  }
}
```

Restringir as ações de fila do Amazon SQS da organização especificada

Para ajudar a proteger seus recursos do Amazon SQS contra acesso externo (acesso por uma entidade fora da sua [organização da AWS](#)), use a instrução a seguir. Essa declaração limita o acesso à fila do Amazon SQS à organização que você especifica na `Condition`. Certifique-se de *<SQS queue ARN>* substituir pelo ARN da função do IAM usada para implantar a fila do Amazon SQS; e *<org-id>* pelo ID da sua organização.

```
{
```

```

{
  "Sid": "DenyQueueActionsOutsideOrg",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:AddPermission",
    "sqs:ChangeMessageVisibility",
    "sqs:DeleteQueue",
    "sqs:RemovePermission",
    "sqs:SetQueueAttributes",
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "StringNotEquals": {
      "aws:PrincipalOrgID": [
        "<org-id>"
      ]
    }
  }
}

```

Conceder permissões do Amazon SQS aos consumidores

Para receber mensagens da fila do Amazon SQS, você precisa fornecer ao consumidor da mensagem as permissões necessárias. A declaração de política a seguir concede ao consumidor, especificado por você, as permissões necessárias para consumir mensagens da fila do Amazon SQS. Ao adicionar a declaração à sua política do Amazon SQS, certifique-se de *<consumer's IAM runtime role ARN>* substituí-la pelo ARN da função de tempo de execução do IAM usada pelo consumidor e *<SQS queue ARN>* pelo ARN da função do IAM usada para implantar a fila do Amazon SQS.

```

{
  "Sid": "AllowConsumersToReceiveFromTheQueue",
  "Effect": "Allow",
  "Principal": {
    "AWS": "<consumer's IAM execution role ARN>"
  },
  "Action": [
    "sqs:ChangeMessageVisibility",
    "sqs:DeleteMessage",

```

```
    "sqs:GetQueueAttributes",
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>"
}
```

Para evitar que outras entidades recebam mensagens da fila do Amazon SQS, adicione a declaração `DenyOtherConsumersFromReceiving` à política de filas do Amazon SQS. Essa declaração restringe o consumo de mensagens ao consumidor que você especificar, permitindo que nenhum outro consumidor tenha acesso, mesmo quando suas permissões de identidade concederem acesso. Certifique-se de substituir *<SQS queue ARN>* e *<consumer's runtime role ARN>* com suas próprias informações.

```
{
  "Sid": "DenyOtherConsumersFromReceiving",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:ChangeMessageVisibility",
    "sqs:DeleteMessage",
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "StringNotLike": {
      "aws:PrincipalARN": "<consumer's execution role ARN>"
    }
  }
}
```

Aplicar a criptografia em trânsito

A declaração de política do `DenyUnsecureTransport` a seguir obriga os consumidores e produtores a usarem canais seguros (conexões TLS) para enviar e receber mensagens da fila do Amazon SQS. Certifique-se de *<SQS queue ARN>* substituir pelo ARN da função do IAM usada para implantar a fila do Amazon SQS.

```
{
  "Sid": "DenyUnsecureTransport",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:ReceiveMessage",
    "sqs:SendMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "Bool": {
      "aws:SecureTransport": "false"
    }
  }
}
```

Restringir a transmissão de mensagens para um tópico específico do Amazon SNS

Veja a seguir um exemplo de declaração de política que permite ao tópico do Amazon SNS enviar mensagem à fila do Amazon SQS. Certifique-se de *<SQS queue ARN>* substituir pelo ARN da função do IAM usada para implantar a fila do Amazon SQS *<SNS topic ARN>* e pelo ARN do tópico do Amazon SNS.

```
{
  "Sid": "AllowSNSToSendToTheQueue",
  "Effect": "Allow",
  "Principal": {
    "Service": "sns.amazonaws.com"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn": "<SNS topic ARN>"
    }
  }
}
```

A declaração de política `DenyAllProducersExceptSNSFromSending` a seguir impede que outros produtores enviem mensagens à fila. Substitua *<SQS queue ARN>* e *<SNS topic ARN>* pelas próprias informações.

```
{
  "Sid": "DenyAllProducersExceptSNSFromSending",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "ArnNotLike": {
      "aws:SourceArn": "<SNS topic ARN>"
    }
  }
}
```

(Opcional) Restringir o recebimento de mensagens a um endpoint da VPC específico

Para restringir o recebimento de mensagens apenas a determinado [endpoint da VPC](#), adicione a declaração de política do Amazon SQS à política de fila do Amazon SQS. Essa declaração impede que um consumidor de mensagens receba mensagens da fila, a menos que as mensagens sejam do endpoint da VPC desejado. *<SQS queue ARN>* Substitua pelo ARN da função do IAM usada para implantar a fila do Amazon SQS *<vpce_id>* e pelo ID do VPC endpoint.

```
{
  "Sid": "DenyReceivingIfNotThroughVPCE",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "StringNotEquals": {
      "aws:sourceVpce": "<vpce id>"
    }
  }
}
```

```
}
```

Declarações de política do Amazon SQS para a fila de mensagens não entregues

Adicione as seguintes declarações de política, identificadas pelo ID da declaração, à política de acesso da fila de mensagens não entregues (DLQ):

- `RestrictAdminQueueActions`
- `DenyQueueActionsOutsideOrg`
- `AllowConsumersToReceiveFromTheQueue`
- `DenyOtherConsumersFromReceiving`
- `DenyUnsecureTransport`

Além de adicionar as declarações de política anteriores à política de acesso da DLQ, você também deve adicionar uma declaração para restringir a transmissão de mensagens às filas do Amazon SQS, conforme descrito na seção a seguir.

Restringir a transmissão de mensagens para filas do Amazon SQS

Para restringir o acesso somente às filas do Amazon SQS da mesma conta, adicione a declaração de política `DenyAnyProducersExceptSQS` a seguir à política de DLQs. Essa declaração não limita a transmissão de mensagens para uma fila específica porque é necessário implantar a DLQ antes de criar a fila principal. Por isso, você não saberá o ARN do Amazon SQS ao criar a DLQ. Se você precisar limitar o acesso a apenas uma fila do Amazon SQS, modifique `aws:SourceArn` na `Condition` com o ARN da sua fila de origem do Amazon SQS quando souber.

```
{
  "Sid": "DenyAnyProducersExceptSQS",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS DLQ ARN>",
  "Condition": {
    "ArnNotLike": {
      "aws:SourceArn": "arn:aws:sqs:<region>:<account-id>:*"
    }
  }
}
```

```
}
```

Important

As políticas de fila do Amazon SQS definidas neste guia não restringem à ação `sqs:PurgeQueue` para determinado perfil ou perfis do IAM. A ação `sqs:PurgeQueue` permite que você exclua todas as mensagens na fila do Amazon SQS. Também é possível usar essa ação para fazer alterações no formato da mensagem sem substituir a fila do Amazon SQS. Ao depurar uma aplicação, você pode limpar a fila do Amazon SQS para remover mensagens possivelmente errôneas. Ao testar a aplicação, você pode direcionar um alto volume de mensagens pela fila do Amazon SQS e, depois, limpar a fila para começar do zero antes de entrar em produção. O motivo para não restringir essa ação a determinada função é que essa função pode não ser conhecida ao implantar a fila do Amazon SQS. Você precisará adicionar essa permissão à política baseada em identidades da função para poder limpar a fila.

Prevenção do problema de adjunto confuso entre serviços

O [problema de adjunto confuso](#) é um problema de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir outra entidade mais privilegiada a executá-la. Para evitar isso, AWS fornece ferramentas que ajudam você a proteger sua conta se você fornecer acesso a terceiros (conhecido como contas cruzadas) ou outros AWS serviços (conhecido como serviços cruzados) aos recursos em sua conta. As declarações de política nesta seção podem ajudar a evitar o problema de adjunto confuso entre serviços.

A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado para que ele use as respectivas permissões com o objetivo de acessar os recursos de outro cliente de uma forma que, normalmente, ele não deveria ter permissão. Para ajudar a combater esse problema, as políticas baseadas em recursos definidas nesta publicação usam as chaves de contexto de condição globais do IAM [aws:SourceArn](#), [aws:SourceAccount](#) e [aws:PrincipalOrgID](#). Isso limita as permissões que um serviço tem para um recurso específico, uma conta específica ou uma organização específica em AWS Organizations.

Usar o IAM Access Analyzer para analisar o acesso entre contas

Você pode usar o [AWS IAM Access Analyzer](#) para revisar suas políticas de filas AWS KMS e políticas de chaves do Amazon SQS e alertá-lo quando uma fila do Amazon SQS ou AWS KMS

uma chave concede acesso a uma entidade externa. O IAM Access Analyzer ajuda a identificar os [recursos](#) da sua organização e as contas que são compartilhadas com uma entidade fora da zona de confiança. Essa zona de confiança pode ser uma AWS conta ou a organização dentro de AWS Organizations que você especifica ao habilitar o IAM Access Analyzer.

O IAM Access Analyzer identifica recursos compartilhados com diretores externos usando o raciocínio baseado em lógica para analisar as políticas baseadas em recursos em seu ambiente. Para cada instância de um recurso compartilhado fora de sua zona de confiança, o Access Analyzer gera uma descoberta. As [descobertas](#) incluem informações sobre o acesso e a entidade principal externa a que é concedido. Analise as descobertas para determinar se o acesso é pretendido e seguro ou se não é intencional e representa um risco à segurança. Para qualquer acesso não intencional, avalie a política afetada e corrija-a. Consulte esta [postagem do blog](#) para obter mais informações sobre como o AWS IAM Access Analyzer identifica o acesso não intencional aos seus recursos. AWS

Para obter mais informações sobre o AWS IAM Access Analyzer, consulte a documentação do [AWS IAM Access Analyzer](#).

Permissões da API do Amazon SQS: referência de ações e recurso

Ao configurar o [Controle de acesso](#) e criar políticas de permissões que você possa anexar a uma identidade do IAM, é possível usar a tabela a seguir como referência. A inclui cada ação do Amazon Simple Queue Service, as ações correspondentes para as quais você pode conceder permissões para realizar a ação e o AWS recurso para o qual você pode conceder as permissões.

Especifique as ações no campo Action da política e o valor do recurso no campo Resource da política. Para especificar uma ação, use o prefixo sqs : seguido do nome da ação (por exemplo, sqs :CreateQueue).

No momento, o Amazon SQS é compatível apenas com [chaves de contexto de condições globais disponíveis no IAM](#).

API do Amazon Simple Queue Service e permissões necessárias para ações

[AddPermission](#)

Ação/Ações: sqs:AddPermission

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[ChangeMessageVisibility](#)

Ação/Ações: sqs:ChangeMessageVisibility

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[ChangeMessageVisibilityBatch](#)

Ação/Ações: sqs:ChangeMessageVisibilityBatch

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[CreateQueue](#)

Ação/Ações: sqs:CreateQueue

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[DeleteMessage](#)

Ação/Ações: sqs:DeleteMessage

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[DeleteMessageBatch](#)

Ação/Ações: sqs:DeleteMessageBatch

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[DeleteQueue](#)

Ação/Ações: sqs:DeleteQueue

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[GetQueueAttributes](#)

Ação/Ações: sqs:GetQueueAttributes

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[GetQueueUrl](#)

Ação/Ações: sqs:GetQueueUrl

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[ListDeadLetterSourceQueues](#)

Ação/Ações: sqs:ListDeadLetterSourceQueues

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[ListQueues](#)

Ação/Ações: sqs:ListQueues

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[ListQueueTags](#)

Ação/Ações: sqs:ListQueueTags

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[PurgeQueue](#)

Ação/Ações: sqs:PurgeQueue

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[ReceiveMessage](#)

Ação/Ações: sqs:ReceiveMessage

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[RemovePermission](#)

Ação/Ações: sqs:RemovePermission

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[SendMessage](#) e [SendMessageBatch](#)

Ação/Ações: sqs:SendMessage

Recurso: arn:aws:sqs:*region*:*account_id*:*queue_name*

[SetQueueAttributes](#)

Ação/Ações: sqs:SetQueueAttributes

Recurso: `arn:aws:sqs:region:account_id:queue_name`

TagQueue

Ação/Ações: `sqs:TagQueue`

Recurso: `arn:aws:sqs:region:account_id:queue_name`

UntagQueue

Ação/Ações: `sqs:UntagQueue`

Recurso: `arn:aws:sqs:region:account_id:queue_name`

Registrar em log e monitorar no Amazon SQS

O Amazon Simple Queue Service é integrado ao [AWS CloudTrail](#), um serviço que fornece um registro das ações realizadas por um usuário, perfil ou AWS service (Serviço da AWS). O CloudTrail captura todas as chamadas de API para o Amazon SQS como eventos. As chamadas capturadas incluem chamadas do console do Amazon SQS e chamadas de código para as operações da API do Amazon SQS. Ao fazer uso das informações coletadas pelo CloudTrail, é possível determinar a solicitação feita ao Amazon SQS, o endereço IP que a realizou, quando ela foi feita e detalhes adicionais.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário raiz ou credenciais de usuário.
- Se a solicitação foi feita em nome de um usuário do Centro de Identidade do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

Quando a Conta da AWS é criada, o CloudTrail torna-se ativo nela e você tem acesso automático ao Histórico de eventos do CloudTrail. O Histórico de eventos do CloudTrail fornece um registro visualizável, pesquisável, baixável e imutável dos últimos 90 dias de eventos de gerenciamento gravados em uma Região da AWS. Para obter mais informações, consulte [Trabalhar com histórico de eventos do CloudTrail](#) no Guia do usuário do AWS CloudTrail. Não há cobranças do CloudTrail pela visualização do Histórico de eventos.

Para obter um registro contínuo de eventos em sua Conta da AWS nos últimos 90 dias, crie uma trilha ou um armazenamento de dados de eventos do [CloudTrail Lake](#).

Alarmes do Amazon CloudWatch

Monitore uma única métrica ao longo de um período que você especificar e realize uma ou mais ações com base no valor da métrica em relação a um determinado limite ao longo de vários períodos. Por exemplo, você pode configurar um alarme do CloudWatch para enviar uma notificação para um tópico do Amazon SNS ou acionar uma ação para enviar uma mensagem para uma fila do Amazon SQS. Os alarmes do CloudWatch não realizam ações simplesmente por estarem em um estado específico. O estado deve ser alterado e permanecer assim por um determinado número de períodos.

Para obter mais informações, consulte [Criação de alarmes do CloudWatch para métricas do Amazon SQS](#) e [Como criar alarmes para filas de mensagens não entregues usando o Amazon CloudWatch](#).

Amazon CloudWatch Logs

Monitore, armazene e acesse arquivos de log relacionados ao Amazon SQS configurando suas aplicações ou funções do Lambda que processam mensagens para enviar logs para o CloudWatch Logs. Você pode usar esses logs para analisar o processamento de mensagens, depurar problemas e monitorar o desempenho dos fluxos de trabalho do Amazon SQS.

Para obter mais informações, consulte [Como registrar em log chamadas de API do Amazon Simple Queue Service usando o AWS CloudTrail](#).

Amazon CloudWatch Events

Use o Amazon CloudWatch Events para detectar alterações ou eventos específicos em seu ambiente AWS e encaminhá-los para uma fila do Amazon SQS. Isso permite capturar dados de eventos, acionar fluxos de trabalho ou armazenar eventos para processamento posterior.

Para obter mais informações, consulte [Automação de notificações de produtos da AWS para o Amazon SQS usando o Amazon EventBridge](#) neste guia e [O EventBridge é a evolução do Amazon CloudWatch Events](#) no Guia do usuário Amazon EventBridge.

AWS CloudTrailLogs do

O CloudTrail captura um registro detalhado das ações realizadas no Amazon SQS por usuários, perfis ou Serviços da AWS. Esses logs permitem rastrear chamadas de API, como

[SendMessage](#), [ReceiveMessage](#) ou [DeleteQueue](#), e fornecer detalhes essenciais, como quem fez a solicitação, quando ela ocorreu e o endereço IP de origem.

Para obter mais informações, consulte [Como registrar em log chamadas de API do Amazon Simple Queue Service usando o AWS CloudTrail](#).

AWS Trusted Advisor

Trusted Advisor usa as práticas recomendadas desenvolvidas a partir do atendimento a clientes da AWS para ajudar a otimizar o uso do Amazon SQS. Ele analisa suas filas do Amazon SQS e fornece recomendações práticas para aumentar a segurança, melhorar a confiabilidade do processamento de mensagens e reduzir custos. Por exemplo, ele pode sugerir a ativação de filas de mensagens não entregues ou melhorar as políticas de acesso à fila para garantir operações seguras.

Consulte mais informações em [AWS Trusted Advisor](#) no Guia de Usuário Suporte.

Trilhas do CloudTrail

Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. As trilhas criadas usando o Console de gerenciamento da AWS são de várias regiões. Só é possível criar uma trilha de região única ou de várias regiões usando a AWS CLI. Criar uma trilha de várias regiões é uma prática recomendada, pois você captura atividades em todas as Regiões da AWS da conta. Ao criar uma trilha de região única, é possível visualizar somente os eventos registrados na Região da AWS da trilha. Para obter mais informações sobre trilhas, consulte [Criar uma trilha para a Conta da AWS](#) e [Criar uma trilha para uma organização](#) no Guia do usuário do AWS CloudTrail.

Uma cópia dos eventos de gerenciamento em andamento pode ser entregue no bucket do Amazon S3 sem nenhum custo via CloudTrail com a criação de uma trilha; no entanto, há cobranças de armazenamento do Amazon S3. Para obter mais informações sobre os preços do CloudTrail, consulte [Preços do AWS CloudTrail](#). Para receber informações sobre a definição de preços do Amazon S3, consulte [Definição de preços do Amazon S3](#).

Armazenamentos de dados de eventos do CloudTrail Lake

O CloudTrail Lake permite executar consultas baseadas em SQL nos eventos. O CloudTrail Lake converte eventos existentes em formato JSON baseado em linhas para o formato [Apache ORC](#). O ORC é um formato colunar de armazenamento otimizado para recuperação rápida de dados. Os eventos são agregados em armazenamentos de dados de eventos, que são coleções imutáveis de eventos baseados nos critérios selecionados com a aplicação de [seletores de eventos avançados](#). Os seletores que aplicados a um armazenamento de dados de

eventos controlam quais eventos persistem e estão disponíveis para consulta. Para obter mais informações sobre o CloudTrail Lake, consulte [Trabalhar com o AWS CloudTrail Lake](#), no Guia do usuário do AWS CloudTrail.

Os armazenamentos de dados de eventos e consultas do CloudTrail Lake incorrem em custos. Ao criar um armazenamento de dados de eventos, você escolhe a [opção de preço](#) que deseja usar para ele. A opção de preço determina o custo para a ingestão e para o armazenamento de eventos, e o período de retenção padrão e máximo para o armazenamento de dados de eventos. Para obter mais informações sobre os preços do CloudTrail, consulte [Preços do AWS CloudTrail](#).

Como registrar em log chamadas de API do Amazon Simple Queue Service usando o AWS CloudTrail

O CloudTrail permite que você registre em log e monitore as operações do Amazon SQS usando dois tipos de eventos: eventos de dados e eventos de gerenciamento. Isso facilita o rastreamento e a auditoria da atividade do Amazon SQS em sua conta.

Eventos de dados do Amazon SQS no CloudTrail

Os [Eventos de dados](#) fornecem informações sobre as operações de recursos realizadas em um recurso (por exemplo, envio de mensagens para um objeto do Amazon SQS). Também são conhecidas como operações de plano de dados. Eventos de dados geralmente são atividades de alto volume. Por padrão, o CloudTrail não registra eventos de dados em log. O Histórico de eventos do CloudTrail não registra eventos de dados.

Há cobranças adicionais para eventos de dados. Para obter mais informações sobre os preços do CloudTrail, consulte [Preços do AWS CloudTrail](#).

É possível registrar em log eventos de dados para os tipos de recurso do Amazon SQS usando o console do CloudTrail, a AWS CLI ou as operações de API do CloudTrail. Para obter mais informações sobre como registrar eventos de dados em log, consulte [Registrar eventos de dados com o Console de gerenciamento da AWS](#) e [Registrar eventos de dados com a AWS Command Line Interface](#) no Guia do usuário do AWS CloudTrail.

Para registrar em log eventos de dados do Amazon SQS com o CloudTrail, você deve usar seletores de eventos avançados para configurar os recursos ou ações específicos do Amazon SQS que deseja registrar. Inclua o tipo de recurso [AWS::SQS::Queue](#) para capturar ações relacionadas à fila. Você pode refinar ainda mais suas preferências de registro em log usando filtros como eventName (tais

como eventos [SendMessage](#)). Para obter mais informações, consulte [AdvancedEventSelector](#) na Referência de API do CloudTrail.

Tipo de evento de dados (console)	valor resources.type	APIs de dados registradas no CloudTrail
Fila do Amazon SQS	AWS::SQS::Queue	• ChangeMessageVisibility • ChangeMessageVisibilityBatch • DeleteMessage • DeleteMessageBatch • GetQueueAttributes • GetQueueUrl • ListDeadLetterSourceQueues • ListQueues • ListQueueTags • ReceiveMessage • SendMessage • SendMessageBatch

Use seletores de eventos avançados para filtrar campos e registrar em log somente eventos importantes. Para obter mais informações sobre esses campos, consulte [AdvancedFieldSelector](#) na Referência de API do AWS CloudTrail.

Eventos de gerenciamento do Amazon SQS no CloudTrail

Os [eventos de gerenciamento](#) fornecem informações sobre operações de gerenciamento executadas em recursos na Conta da AWS. Elas também são conhecidas como operações de ambiente de gerenciamento. Por padrão, o CloudTrail registra eventos de gerenciamento em logs.

O Amazon SQS registra em log as operações a seguir do ambiente de gerenciamento no CloudTrail como eventos de gerenciamento.

- [AddPermission](#)

- [CancelMessageMoveTask](#)
- [CreateQueue](#)
- [DeleteQueue](#)
- [ListMessageMoveTasks](#)
- [PurgeQueue](#)
- [RemovePermission](#)
- [SetQueueAttributes](#)
- [StartMessageMoveTask](#)
- [TagQueue](#)
- [UntagQueue](#)

Exemplos de eventos do Amazon SQS

Um evento representa uma única solicitação de qualquer origem e inclui informações sobre a operação solicitada, a data e a hora da operação da API, os parâmetros de solicitação etc. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API públicas, portanto não são exibidos em uma ordem específica.

O exemplo a seguir mostra um evento do CloudTrail que demonstra a operação `SendMessage`.

```
{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/SessionName",
    "accountId": "123456789012",
    "accessKeyId": "ACCESS_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed",
        "accountId": "123456789012",
        "userName": "RoleToBeAssumed"
      }
    },
    "attributes": {
      "creationDate": "2023-11-07T22:13:06Z",
```



```

    "mfaAuthenticated": "false"
  }
},
"eventTime": "2023-11-07T23:59:11Z",
"eventSource": "sqs.amazonaws.com",
"eventName": "SendMessage",
"awsRegion": "ap-southeast-4",
"sourceIPAddress": "10.0.118.80",
"userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
"requestParameters": {
  "queueUrl": "https://sqs.ap-southeast-4.amazonaws.com/123456789012/MyQueue",
  "messageBody": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "messageDeduplicationId": "MsgDedupIdSdk1ae1958f2-bbe8-4442-83e7-4916e3b035aa",
  "messageGroupId": "MsgGroupIdSdk16"
},
"responseElements": {
  "mD50fMessageBody": "9a4e3f7a614d9dd9f8722092dbda17a2",
  "mD50fMessageSystemAttributes": "f88f0587f951b7f5551f18ae699c3a9d",
  "messageId": "93bb6e2d-1090-416c-81b0-31eb1faa8cd8",
  "sequenceNumber": "18881790870905840128"
},
"requestID": "c4584600-fe8a-5aa3-a5ba-1bc42f055fae",
"eventID": "98c735d8-70e0-4644-9432-b6ced4d791b1",
"readOnly": false,
"resources": [
  {
    "accountId": "123456789012",
    "type": "AWS::SQS::Queue",
    "ARN": "arn:aws:sqs:ap-southeast-4:123456789012:MyQueue"
  }
],
"eventType": "AwsApiCall",
"managementEvent": false,
"recipientAccountId": "123456789012",
"eventCategory": "Data",
"tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sqs.ap-southeast-4.amazonaws.com"
}

```

 Note

A operação `ListQueues` é um caso único, pois não atua em um recurso específico. Como resultado, o campo ARN não inclui um nome de fila e usa um caractere curinga (*) em vez disso.

Para obter informações sobre o conteúdo dos registros do CloudTrail, consulte [Conteúdo dos registros do CloudTrail](#) no Guia do usuário do AWS CloudTrail.

Monitoramento de filas do Amazon SQS usando o CloudWatch

O Amazon SQS e o Amazon CloudWatch são integrados para que você possa usar o CloudWatch para visualizar e analisar métricas para suas filas do Amazon SQS. Você pode visualizar e analisar as métricas das filas no [console do Amazon SQS](#), no [console do CloudWatch](#), usando a opção [AWS CLI](#) ou usando a opção a [API do CloudWatch](#). Você também pode [definir alarmes do CloudWatch](#) para métricas do Amazon SQS.

As métricas do CloudWatch para as filas do Amazon SQS são coletadas e enviadas automaticamente ao CloudWatch em intervalos de um minuto. Essas métricas são coletadas em todas as filas que estejam de acordo com as diretrizes do CloudWatch por estarem ativas. O CloudWatch considerará uma fila ativa por até seis horas, se ela contiver mensagens ou se qualquer ação acessá-la.

Quando uma fila do Amazon SQS fica inativa por mais de seis horas, o serviço Amazon SQS é considerado inativo e para de fornecer métricas ao serviço CloudWatch. Dados ausentes, ou dados que representam zero, não podem ser visualizados nas métricas do CloudWatch para o Amazon SQS durante o período em que a fila do Amazon SQS ficou inativa.

 Note

- Uma fila do Amazon SQS pode ser ativada quando o usuário que está chamando uma API na fila não está autorizado e a solicitação falha.
- O console do Amazon SQS executa uma chamada de API [GetQueueAttributes](#) quando a página da fila é aberta. A solicitação da API `GetQueueAttributes` ativa a fila.
- Um atraso de até 15 minutos ocorre nas métricas do CloudWatch quando uma fila é ativada de um estado inativo.

- Não há cobrança para as métricas do Amazon SQS relatadas por meio do CloudWatch. Elas são fornecidas como parte do serviço Amazon SQS.
- As métricas do CloudWatch são compatíveis com filas padrão e FIFO.

Acessar as métricas do CloudWatch para o Amazon SQS

O Amazon SQS e o Amazon CloudWatch são integrados para que você possa usar o CloudWatch para visualizar e analisar métricas para suas filas do Amazon SQS. Você pode visualizar e analisar as métricas das filas no [console do Amazon SQS](#), no [console do CloudWatch](#), usando a opção [AWS CLI](#) ou usando a opção a [API do CloudWatch](#). Você também pode [definir alarmes do CloudWatch](#) para métricas do Amazon SQS.

Usar o console do Amazon SQS

Use o console do Amazon SQS para acessar e analisar métricas de até 10 filas do Amazon SQS.

1. Faça login no [console do Amazon SQS](#).
2. Na lista de filas, escolha (selecione) as caixas das filas cujas métricas você deseja acessar. Você pode exibir métricas para até 10 filas.
3. Escolha a guia Monitoring (Monitoramento).

Vários gráficos são exibidos na seção SQS metrics.

4. Para entender o que um determinado gráfico representa, passe o mouse sobre  ao lado do gráfico desejado, ou consulte [Métricas disponíveis do CloudWatch para o Amazon SQS](#).
5. Para alterar o intervalo de tempo para todos os gráficos ao mesmo tempo, em Time Range, escolha o intervalo de tempo desejado (por exemplo, Last Hour).
6. Para exibir estatísticas adicionais para um gráfico individual, selecione o gráfico.
7. Na caixa de diálogo CloudWatch Monitoring Details (Detalhes de monitoramento do CloudWatch), selecione uma Statistic (Estatística), por exemplo, Sum (Soma). Para obter uma lista de estatísticas suportadas, consulte [Métricas disponíveis do CloudWatch para o Amazon SQS](#).
8. Para alterar o intervalo de tempo que um gráfico individual exibe (por exemplo, para mostrar um intervalo de tempo das últimas 24 horas em vez dos últimos 5 minutos, ou mostrar um período

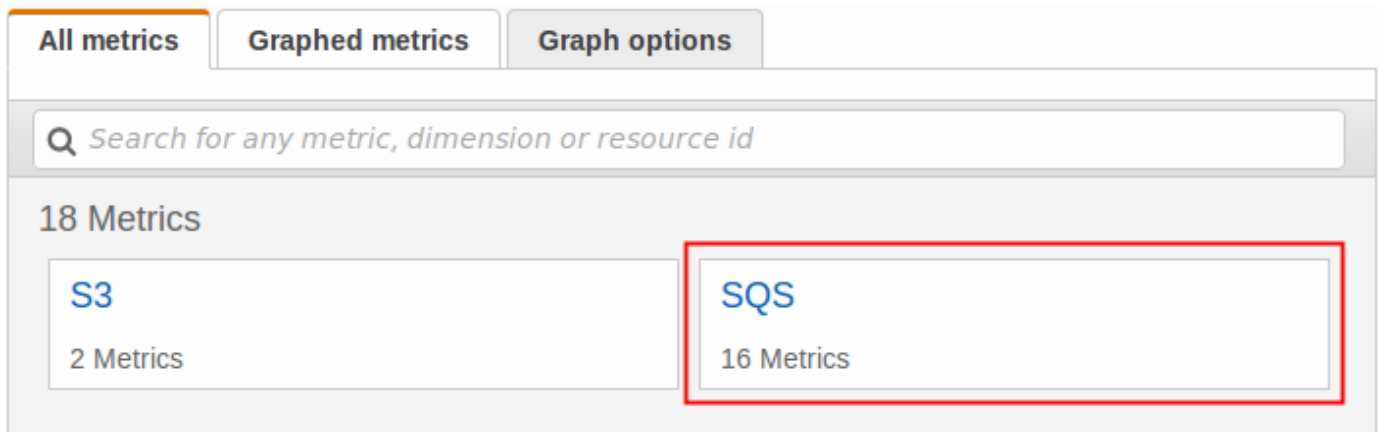
de hora em vez de um período de 5 minutos), com a caixa de diálogo do gráfico ainda exibida, em Time Range, escolha o intervalo de tempo desejado (por exemplo, Last 24 Hours). Em Period, escolha o período desejado no intervalo de tempo especificado (por exemplo, 1 Hour). Ao terminar de visualizar o gráfico, clique em Close.

9. (Opcional) Para trabalhar com outros recursos do CloudWatch, na guia Monitoring (Monitoramento), escolha View all CloudWatch metrics (Visualizar todas as métricas do CloudWatch) e, em seguida, siga as instruções no procedimento [Como usar o console do Amazon CloudWatch](#).

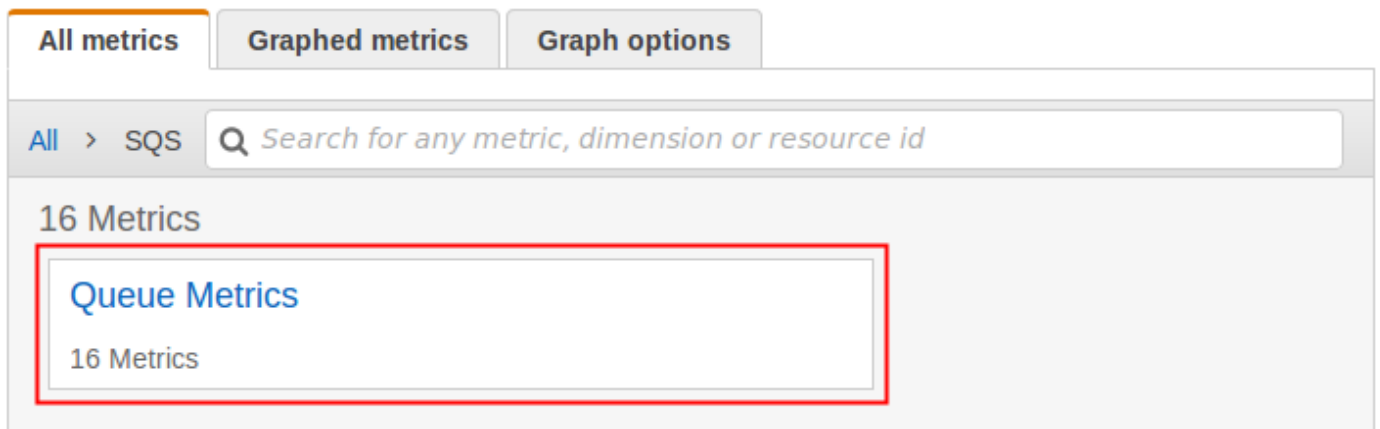
Como usar o console do Amazon CloudWatch

Use o console do CloudWatch para acessar e analisar métricas do Amazon SQS.

1. Faça login no [console do CloudWatch](#).
2. No painel de navegação, selecione Métricas.
3. Selecione o namespace de métrica do SQS.



4. Selecione a dimensão de métrica Queue Metrics.



5. Agora você pode examinar as métricas do Amazon SQS:

- Para classificar a métrica, use o cabeçalho da coluna.
- Para criar um gráfico de uma métrica, marque a caixa de seleção ao lado da métrica.
- Para filtrar por métrica, selecione o nome da métrica e, em seguida, escolha Adicionar à pesquisa.

All metrics			Graphed metrics	Graph options
All	> SQS	> Queue Metrics	Search for any metric, dimension or resource id	
☐	QueueName (16)		Metric Name	
☐	MyQueue		ApproximateAgeOfOldestMessage	
☐	MyQueue		MessagesDelayed	
☐	MyQueue		MessagesNotVisible	
☐	MyQueue		MessagesVisible	
☐	MyQueue			
☐	MyQueue			
☐	MyQueue			
☐	MyQueue			
☐	MyQueue		NumberOfMessagesSent	

Para obter mais informações e opções, consulte [Métricas do gráfico](#) e [Usar painéis do CloudWatch](#) no Guia do usuário do Amazon CloudWatch.

Usar o AWS Command Line Interface

Para acessar métricas do Amazon SQS usando a AWS CLI, execute o comando [get-metric-statistics](#).

Para obter mais informações, consulte [Obter as estatísticas de uma métrica](#) no Guia do usuário do Amazon CloudWatch.

Usando a API do CloudWatch

Para acessar métricas do Amazon SQS usando a API do CloudWatch, use a ação [GetMetricStatistics](#).

Para obter mais informações, consulte [Obter as estatísticas de uma métrica](#) no Guia do usuário do Amazon CloudWatch.

Criação de alarmes do CloudWatch para métricas do Amazon SQS

O CloudWatch permite que você acione alarmes quando uma métrica atinge um limite especificado. Por exemplo, você pode criar um alarme para a métrica `NumberOfMessagesSent`. Por exemplo, se mais de 100 mensagens são enviadas à fila `MyQueue` em 1 hora, uma notificação por e-mail é enviada. Para obter mais informações, consulte [Criar alarmes do Amazon CloudWatch](#) no Guia do usuário do Amazon CloudWatch.

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Alarmes e, em seguida, Criar alarme.
3. Na seção Select Metric (Selecionar métrica) da caixa de diálogo Create Alarm (Criar alerta), selecione Browse Metrics (Procurar métricas), SQS.
4. Em SQS > Queue Metrics (SQS > Métricas de fila), selecione QueueName (Nome da fila) e Metric Name (Nome da métrica) para o qual definir um alerta e, em seguida, selecione Next (Avançar). Para ver uma lista das métricas disponíveis, consulte [Métricas disponíveis do CloudWatch para o Amazon SQS](#).

No exemplo a seguir, a seleção é de um alarme para a métrica `NumberOfMessagesSent` para a fila `MyQueue`. O alarme é acionado quando o número de mensagens enviadas excede 100.

5. Na seção Define Alarm (Definir alerta) da caixa de diálogo Create Alarm (Criar alerta), faça o seguinte:
 - a. Em Alarm Threshold (Limite de alerta), digite Name (Nome) e Description (Descrição) para o alerta.
 - b. Defina is como > 100.
 - c. Defina for (para) como 1 out of 1 datapoints (1 de 1 ponto de dados).
 - d. Em Alarm preview (Visualização do alerta), defina Period (Período) como 1 Hour (1 hora).
 - e. Defina Statistic (Estatística) como Standard (Padrão), Sum (Soma).

- f. Em Actions (Ações), defina Whenever this alarm (Sempre que esse alerta) como State is ALARM (Estado é ALERTA).

Se desejar que o CloudWatch envie uma notificação quando o alerta for acionado, selecione um tópico existente do Amazon SNS ou selecione New list (Nova lista) e insira os endereços de e-mail separados por vírgulas.

 Note

Se você criar um novo tópico do Amazon SNS, os endereços de e-mail deverão ser verificados para que recebam notificações. Se o estado de alarme for alterado antes que os endereços de e-mail sejam verificados, as notificações não serão recebidas.

6. Escolha Create Alarm.

O alarme é criado.

Métricas disponíveis do CloudWatch para o Amazon SQS

O Amazon SQS envia as seguintes métricas para o CloudWatch.

 Note

Para algumas métricas, o resultado é aproximado por causa da arquitetura distribuída do Amazon SQS. Na maioria dos casos, a contagem deve ser próxima da quantidade real de mensagens na fila.

Métricas do Amazon SQS

O Amazon SQS publica automaticamente métricas operacionais no [Amazon CloudWatch](#) sob o namespace AWS/SQS. Essas métricas ajudam a monitorar a integridade e o desempenho da fila. Devido à natureza distribuída do SQS, muitos valores são aproximados, mas precisos o suficiente para a maioria das decisões operacionais.

 Note

- Todas as métricas emitem valores não negativos somente quando a fila está ativa.

- Algumas métricas (como `SentMessageSize`) não são emitidas até que pelo menos uma mensagem seja enviada.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
<code>ApproximateAgeOfOldestMessage</code>	A idade da mensagem não processada mais antiga na fila.	Segundos	Relatado se a fila contém pelo menos uma mensagem ativa.	• Para filas padrão, se uma mensagem for recebida três vezes ou mais e não for excluída, o SQS a moverá para o final da fila. A métrica, então, refletirá a idade da próxima mensagem que não excedeu o limite de recebimento. Essa reordenação ocorre mesmo quando uma política de redirecionamento está em vigor. • As mensagens com pílulas venenosas (aquelas recebidas repetidamente, mas nunca excluídas) são excluídas dessa métrica até serem processadas com sucesso. • Quando uma mensagem é movida para uma DLQ após ultrapassar o valor de <code>maxReceiveCount</code>, a idade é redefinida. Nesse caso, a métrica da DLQ reflete a hora em que a mensagem foi movida,

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
				não quando foi enviada originalmente. • As filas FIFO não reordenam as mensagens para preservar a ordem. Uma mensagem com falha bloqueia seu grupo de mensagens até que seja excluída ou expire. Se uma DLQ estiver configurada, a mensagem será enviada para lá depois que o limite de recebimento for atingido.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
ApproximateNumberOfGroupsWithInflightMessages	Somente para FIFO. O número de grupos de mensagens com uma ou mais mensagens em trânsito.	Contagem	Relatado se a fila FIFO está ativa.	- Uma mensagem é considerada em trânsito depois de ser recebida da fila por um consumidor, mas ainda não foi excluída ou expirou. - Essa métrica ajuda a solucionar problemas e otimizar o throughput da fila FIFO. Valores altos geralmente indicam uma forte simultaneidade. - Se a fila tiver grande acúmulo e esse valor permanecer baixo, considere escalar os consumidores ou aumentar o número de grupos de mensagens ativos. - Para limites de throughput e mensagens em trânsito, consulte Cotas do Amazon SQS.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
<code>ApproximateNumberOfMessagesDelayed</code>	O número de mensagens na fila que estão atrasadas e não disponíveis imediatamente para recuperação.	Contagem	Relatado se existem mensagens atrasadas na fila.	• Aplica-se às filas configuradas com um atraso padrão e às mensagens individuais enviadas com um parâmetro <code>DelaySeconds</code> . • As mensagens atrasadas permanecem ocultas dos consumidores até que o período de atraso expire, o que pode afetar a percepção do acúmulo ou do throughput da fila.
<code>ApproximateNumberOfMessagesNotVisible</code>	O número de mensagens em trânsito que foram recebidas, mas ainda não excluídas ou expiradas.	Contagem	Relatado se existirem mensagens em trânsito.	• As mensagens entram no estado de trânsito após serem enviadas a um consumidor por meio da API ReceiveMessage . • Essas mensagens são temporariamente ocultas de outros consumidores durante a janela de tempo limite de visibilidade. • Use essa métrica para rastrear atrasos no processamento de mensagens ou consumidores presos.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
ApproximateNumberOfMessagesVisible	O número de mensagens atualmente disponíveis para recuperação e processamento.	Contagem	Relatado se a fila está ativa.	- Reflete acúmulo sendo atualmente processado na fila. - Não há limite rígido de quantas mensagens podem ser acumuladas, mas elas estão sujeitas ao período de retenção configurado pela fila. - Um valor consistentemente alto pode indicar consumidores subprovisionados ou lógica de processamento paralisada.
NumberOfEmptyReceives ¹	O número de chamadas de API ReceiveMessage que não retornaram mensagens.	Contagem	Relatado durante as operações de recebimento.	- Essa métrica pode ajudar a identificar ineficiências no comportamento da pesquisa ou instâncias de consumidores subutilizados. - Valores altos podem ocorrer quando a fila está vazia, o consumidor usa uma sondagem curta ou as mensagens estão sendo processadas de forma mais rápida do que são produzidas. - Esse não é um indicador preciso do estado da fila. Ela reflete o comportamento do lado do serviço e pode incluir novas tentativas.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
NumberOfDuplicatedSentMessages	Somente para FIFO. O número de mensagens enviadas que foram desduplicadas e não adicionadas à fila.	Contagem	Relatado se valores ou conteúdo MessageDuplicationId duplicados forem detectados.	• O SQS desduplica mensagens com base no hash MessageDuplicationId ou baseado em conteúdo (se ativado). • Um valor alto pode indicar que um produtor está enviando repetidamente a mesma mensagem dentro da janela de desduplicação de 5 minutos. • Use essa métrica para solucionar problemas de lógica redundante do produtor ou confirmar se a desduplicação está funcionando conforme o esperado.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
NumberOfMessagesDeleted ¹	O número de mensagens excluídas com êxito da fila.	Contagem	Relatado para cada solicitação de exclusão com um identificador de recibo válido.	- Essa métrica conta todas as operações de exclusão bem-sucedidas, mesmo que a mesma mensagem seja excluída mais de uma vez. - Os motivos comuns para valores acima do esperado incluem: - Várias exclusões da mesma mensagem usando identificadores de recebimento diferentes, depois que o tempo limite de visibilidade expira e a mensagem é recebida novamente. - Exclusões duplicadas usando o mesmo identificador de recebimento, que ainda retornam um status de sucesso e incrementam a métrica. - Use essa métrica para monitorar o sucesso do processamento de mensagens, mas não a trate como uma contagem exata de mensagens exclusivas excluídas.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
NumberOfMessagesReceived ¹	O número de mensagens retornadas pela API ReceiveMessage .	Contagem	Relatado durante as operações de recebimento.	• Isso inclui todas as mensagens retornadas aos consumidores, incluindo aquelas que são posteriormente retornadas à fila devido à expiração do tempo limite de visibilidade. • Uma única mensagem pode ser recebida várias vezes se não for excluída, o que pode fazer com que essa métrica exceda o número de mensagens enviadas. • Use isso para rastrear a atividade do consumidor, mas não a trate como uma contagem de mensagens exclusivas processadas.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
<code>NumberOfMessagesSent</code> ¹	O número de mensagens adicionadas com êxito a uma fila.	Contagem	Relatado para cada envio manual bem-sucedido.	• Chamadas manuais para <code>SendMessage</code> ou <code>SendMessageBatch</code> são contabilizadas, incluindo aquelas direcionadas a uma DLQ. • As mensagens que são movidas automaticamente para uma DLQ após excederem a <code>maxReceiveCount</code> não são incluídas nessa métrica. • Como resultado, o <code>NumberOfMessagesSent</code> pode ser menor do que o <code>NumberOfMessagesReceived</code>, principalmente se as políticas de redirecionamento estiverem transferindo muitas mensagens para DLQs no segundo plano.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
SentMessageSize ¹	O tamanho das mensagens enviadas com sucesso para a fila.	Bytes	Não emitido até que pelo menos uma mensagem seja enviada.	- Essa métrica não aparecerá no console do CloudWatch até que a fila receba sua primeira mensagem. - Use essa métrica para rastrear o tamanho de cada mensagem em bytes. Isso é útil para analisar tendências de carga útil ou estimar o custo de throughput. - O tamanho máximo da mensagem para o SQS é de 1 MiB.
ApproximateNumberOfNoisyGroups	O número de grupos de mensagens que são considerados barulhentos em uma fila justa. Um grupo de mensagens barulhentas representa um locatário barulhento de uma fila multilocatário.	Contagem	Um valor não negativo é relatado se a fila estiver ativa .	- Ajuda a identificar possíveis problemas com vizinhos barulhentos em ambientes multilocatário, rastreando o grupos de mensagens que consomem recursos desproporcionais. - Use essa métrica para definir alarmes que são acionados quando o número de grupos barulhentos exceder o limite aceitável, indicando possíveis problemas de equidade na fila.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
<code>ApproximateNumberOfMessagesVisibleInQuietGroups</code>	O número de mensagens visíveis, excluindo mensagens de grupos de mensagens barulhentos.	Contagem	Um valor não negativo é relatado se a fila estiver ativa .	- Fornece visibilidade do acúmulo de filas para grupos de mensagens de taxa padrão, excluindo mensagens de vizinhos barulhentos. - Ajuda a identificar o verdadeiro acúmulo de processamento para grupos de mensagens típicos ao filtrar o impacto de vizinhos barulhentos.
<code>ApproximateNumberOfMessagesNotVisibleInQuietGroups</code>	O número de mensagens em trânsito, excluindo mensagens de grupos de mensagens barulhentos.	Contagem	Um valor não negativo é relatado se a fila estiver ativa .	- Rastreia mensagens em andamento (sendo processadas, mas ainda não excluídas) de grupos de mensagens bem comportados. - Use essa métrica para monitorar o throughput de grupos de mensagens normais e detectar gargalos de processamento que não sejam causados por vizinhos barulhentos.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
ApproximateNumberOfMessagesDelayedInQuietGroups	O número de mensagens, excluindo mensagens de grupos barulhentos que estão atrasadas e indisponíveis para leitura imediata. Isso pode acontecer quando a fila tem a configuração de fila com atraso ou quando uma mensagem foi enviada com um parâmetro de atraso.	Contagem	Um valor não negativo é relatado se a fila estiver ativa .	- Ajuda a monitorar o acúmulo de mensagens atrasadas de grupos de mensagens com padrões de throughput normais ou esperados (em oposição a grupos de alto volume ou barulhentos) - Útil para entender os requisitos futuros de processamento e o planejamento de capacidade para workloads típicas.

Métrica	Descrição	Unidades	Relatório resultante	Notas principais
<code>ApproximateAgeOfOldestMessageInQuietGroups</code>	A idade da mensagem mais antiga não excluída na fila, excluindo mensagens de grupos de mensagens barulhentas.	Segundos	Um valor não negativo é relatado se a fila estiver ativa .	• Usado para monitorar a conformidade do SLA e detectar gargalos de processamento em grupos de mensagens com padrões de throughput normais ou esperados (em oposição a grupos de mensagens de alto volume ou barulhentas que, de outra forma, poderiam distorcer a métrica). • Use essa métrica para definir alarmes para tempos limite de processamento de mensagens que ignoram mensagens transformadas em antigas artificialmente de vizinhos barulhentos.

¹ Essas métricas refletem a atividade no nível do sistema e podem incluir novas tentativas, duplicatas ou mensagens atrasadas. Não use contagens brutas para estimar o estado da fila em tempo real sem considerar o comportamento do ciclo de vida das mensagens.

Filas de mensagens não entregues (DLQs) e métricas do CloudWatch

Ao trabalhar com DLQs, é importante entender como as métricas do Amazon SQS se comportam:

- **NumberOfMessagesSent**: essa métrica se comporta de forma diferente para DLQs:
 - Envio manual: as mensagens enviadas manualmente para uma DLQ são capturadas por essa métrica.
 - Redirecionamento automático: as mensagens movidas automaticamente para uma DLQ devido a falhas de processamento não são capturadas por essa métrica. Como resultado, as métricas

NumberOfMessagesSent e NumberOfMessagesReceived podem mostrar discrepâncias nas DLQs.

- Métrica recomendada para DLQs: para monitorar o estado de uma DLQ, use a métrica `ApproximateNumberOfMessagesVisible`. Essa métrica indica o número de mensagens atualmente disponíveis para processamento na DLQ.

Filas justas e métricas do CloudWatch

Quando você usa [filas justas](#), o Amazon SQS emite as seguintes métricas adicionais:

- `ApproximateNumberOfNoisyGroups`
- `ApproximateNumberOfMessagesVisibleInQuietGroups`
- `ApproximateNumberOfMessagesNotVisibleInQuietGroups`
- `ApproximateNumberOfMessagesDelayedInQuietGroups`
- `ApproximateAgeOfOldestMessageInQuietGroups`

Note

Cada métrica `QuietGroup` é um subconjunto da métrica padrão equivalente em nível de fila `Approximate`, mas exclui mensagens de grupos vizinhos barulhentos.

Grupos barulhentos

Um grupo de mensagens barulhentas representa um locatário barulhento de uma fila multilocatário.

Grupos silenciosos

Grupos de mensagens que excluem grupos barulhentos.

Observação do comportamento das filas justas do SQS

Para monitorar o efeito das filas justas do Amazon SQS, você pode comparar as métricas `Approximate...InQuietGroups` com métricas padrão em nível de fila. Durante picos de tráfego para um locatário específico, as métricas gerais do nível da fila podem revelar o aumento dos acúmulos ou a idade das mensagens mais antigas. No entanto, ao analisar os grupos silenciosos

isoladamente, você pode identificar que a maioria dos grupos de mensagens ou locatários barulhentos não são afetados, e fornece uma estimativa do número total de grupos de mensagens impactados.

Embora essas novas métricas forneçam uma boa visão geral do comportamento das filas justas do Amazon SQS, pode ser benéfico entender qual locatário específico está causando a carga. O [Amazon CloudWatch Contributor Insights](#) permite ver métricas relacionadas aos principais colaboradores, o número total de colaboradores exclusivos e o uso deles. Isso é especialmente útil em cenários em que você está lidando com milhares de locatários que, de outra forma, levariam a dados (e custos) de alta cardinalidade ao emitir métricas tradicionais.

Para ver um exemplo de configuração de monitoramento para filas justas, consulte a amostra no [GitHub](#).

Dimensões para métricas do Amazon SQS

As métricas do Amazon SQS no CloudWatch usam uma única dimensão: **QueueName**. Todos os dados métricos são agrupados e filtrados pelo nome da fila.

Dicas de monitoramento

Monitore o SQS de forma eficaz usando métricas importantes e alarmes do CloudWatch para detectar acúmulos nas filas, otimizar o desempenho e permanecer dentro dos limites do serviço.

- Defina [alarmes do CloudWatch](#) com base em `ApproximateNumberOfMessagesVisible` para capturar o crescimento do acúmulo.
- Monitore `NumberOfEmptyReceives` para ajustar a frequência da pesquisa e reduzir o custo da API.
- Use `ApproximateNumberOfGroupsWithInflightMessages` em filas FIFO para diagnosticar limites de throughput.
- Consulte [Cotas do SQS](#) para entender os limites métricos e de serviço.

Validação de compatibilidade para o Amazon SQS

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte [Serviços da AWS Escopo por Programa de Conformidade](#) [Serviços da AWS](#) e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de [AWS conformidade Programas AWS](#) de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte [Baixar relatórios em AWS Artifact](#).

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. Para obter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte a [documentação AWS de segurança](#).

Resiliência no Amazon SQS

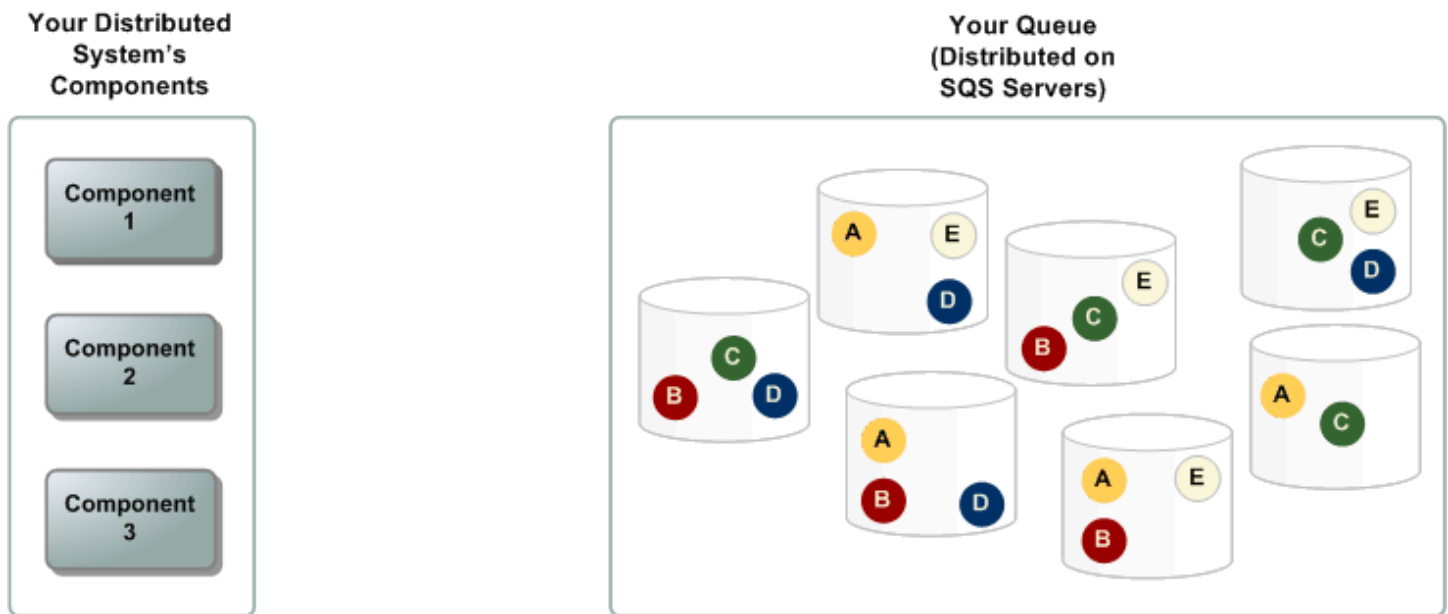
A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas com redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais. Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte [Infraestrutura AWS global](#).

Além da infraestrutura AWS global, o Amazon SQS oferece filas distribuídas.

Filas distribuídas

Há três partes principais em um sistema de mensagens distribuído: os componentes do sistema distribuído, a fila (distribuída em servidores do Amazon SQS) e as mensagens na fila.

No cenário a seguir, o sistema tem vários produtores (componentes que enviam mensagens para a fila) e consumidores (componentes que recebem mensagens da fila). A fila (que contém as mensagens A a E) armazena as mensagens de forma redundante em vários servidores do Amazon SQS.



Segurança da infraestrutura no Amazon SQS

Como um serviço gerenciado, o Amazon SQS é protegido pelos procedimentos de segurança da rede da global da AWS descritos no whitepaper [Amazon Web Services: visão geral dos processos de segurança](#).

Você usa ações de API publicadas pela AWS para acessar o Amazon SQS pela rede. Os clientes devem oferecer suporte a Transport Layer Security (TLS) 1.2 ou posterior. Os clientes também devem ter suporte a pacotes de criptografia com sigilo de encaminhamento perfeito (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE).

Você deve assinar as solicitações usando um ID da chave de acesso e uma chave de acesso secreta associados a um principal do IAM. Como alternativa, você pode usar o [AWS Security Token Service](#) (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Você pode chamar essas ações de API de qualquer local da rede, mas o Amazon SQS oferece suporte a políticas de acesso com base em recursos, que podem incluir restrições com base no endereço IP de origem. Também é possível usar políticas do Amazon SQS para controlar o acesso de endpoints da Amazon VPC ou de VPCs específicas. Isso isola efetivamente o acesso à rede de determinada fila do Amazon SQS apenas a partir da VPC específica dentro da rede da AWS. Para obter mais informações, consulte [Exemplo 5: negar o acesso se não vier de um VPC endpoint](#).

Práticas recomendadas de segurança para o Amazon SQS

A AWS fornece muitos recursos de segurança para o Amazon SQS. Revise-os no contexto da sua própria política de segurança. Veja a seguir as práticas recomendadas de segurança preventiva para o Amazon SQS.

Note

As orientações específicas de implementação fornecidas são para casos de uso e implementações comuns. Sugerimos que você examine as melhores práticas no contexto do seu caso de uso, arquitetura e modelo de ameaças específicos.

Garantir que as filas não sejam acessíveis ao público

A menos que seja nitidamente necessário que qualquer pessoa na Internet possa ler ou gravar na sua fila do Amazon SQS, você deve garantir que a fila não seja acessível ao público (acessível por todo mundo ou por qualquer usuário autenticado da AWS).

- Evite criar políticas com o `Principal` definido como `"`.
- Evite usar um curinga (*). Em vez disso, nomeie um usuário ou usuários específicos.

Implemente o privilégio de acesso mínimo

Quando você concede permissões, você decide quem as recebe, para quais filas as permissões se aplicam, e as ações específicas de API que você deseja permitir para essas filas. A implementação de privilégios mínimos é importante para reduzir os riscos de segurança e o efeito de erros ou intenção maliciosa.

Siga o aviso de segurança padrão de concessão de privilégios mínimos. Ou seja, conceda apenas as permissões necessárias para executar uma tarefa específica. Você pode fazer essa implementação usando uma combinação de políticas de segurança.

O Amazon SQS utiliza o modelo produtor-consumidor, exigindo três tipos de acesso à conta de usuário:

- Administradores: acesso para criar, modificar e excluir filas. Os administradores também controlam as políticas de fila.

- Produtores: acesso para enviar mensagens às filas.
- Consumidores: acesso para receber e excluir mensagens nas filas.

Para obter mais informações, consulte as seções a seguir:

- [Gerenciamento de identidade e acesso no Amazon SQS](#)
- [Permissões da API do Amazon SQS: referência de ações e recurso](#)
- [Usar políticas personalizadas com linguagem de políticas de acesso do Amazon SQS](#)

Usar funções do IAM para aplicações e produtos da AWS que exigem acesso ao Amazon SQS

Para aplicações ou produtos da AWS, como o Amazon EC2 para acessar filas do Amazon SQS, elas devem usar credenciais da AWS em suas solicitações de API da AWS. Como essas credenciais não são alteradas automaticamente, você não deve armazenar credenciais da AWS diretamente no aplicativo ou na instância do EC2.

Você deve usar uma função do IAM para gerenciar credenciais temporárias para aplicações ou produtos que precisem acessar o Amazon SQS. Quando você usa um perfil, não é necessário distribuir credenciais de longo prazo (como um nome de usuário, uma senha e chaves de acesso) para uma instância do EC2 ou serviço da AWS, como o AWS Lambda. Em vez disso, a função fornece permissões temporárias que os aplicativos podem usar ao fazer chamadas para outros recursos da AWS.

Para obter mais informações, consulte [IAM Roles](#) (Funções do IAM) e [Common Scenarios for Roles: Users, Applications, and Services](#) (Cenários comuns para funções: usuários, aplicações e produtos) no Guia do usuário do IAM.

Implemente a criptografia no lado do servidor

Para atenuar problemas de vazamento de dados, utilize a criptografia em repouso para criptografar as mensagens usando uma chave armazenada em um local diferente do local de armazenamento das suas mensagens. A criptografia no lado do servidor (SSE) fornece criptografia dos dados em repouso. O Amazon SQS criptografa os dados no nível da mensagem ao armazená-los e descriptografa as mensagens para você, quando você as acessa. O SSE usa chaves gerenciadas pelo AWS Key Management Service. Se você autenticar sua solicitação e tiver as permissões de acesso, não haverá diferença de acesso entre as filas criptografadas e não criptografadas.

Para obter mais informações, consulte [Criptografia em repouso no Amazon SQS](#) e [Gerenciamento de chaves do Amazon SQS](#).

Aplique a criptografia de dados em trânsito

Sem HTTPS (TLS), um invasor que venha pela rede pode bisbilhotar o tráfego de rede ou manipulá-lo usando um ataque, como man-in-the-middle. Permita somente conexões criptografadas por HTTPS (TLS), usando a condição [aws:SecureTransport](#) na política de fila para forçar que as solicitações usem SSL.

Considere usar endpoints da VPC para acessar o Amazon SQS

Se você tiver filas com as quais você deve poder interagir, mas que não devem de forma alguma ficar expostas à Internet, use VPC endpoints para enfileirar o acesso apenas aos hosts dentro de uma VPC específica. Você pode usar políticas de fila para controlar o acesso a filas de endpoints da Amazon VPC específicos ou de VPCs específicas.

Os endpoints da VPC do Amazon SQS fornecem duas maneiras de controlar o acesso às suas mensagens:

- É possível controlar as solicitações, os usuários ou os grupos permitidos por um VPC endpoint específico.
- Você pode controlar quais VPCs ou VPC endpoints terão acesso à fila usando uma política de fila.

Para obter mais informações, consulte [Endpoints da Amazon Virtual Private Cloud para o Amazon SQS](#) e [Criar uma política de endpoint da Amazon VPC para o Amazon SQS](#).

Recursos do Amazon SQS relacionados

A tabela a seguir lista os recursos relacionados que serão úteis à medida que você utilizar este serviço.

Recurso	Descrição
Referência da API do Amazon Simple Queue Service	Descrições de ações de , parâmetros e tipos de dados, além de uma lista de erros que o serviço retorna.
Amazon SQS na Referência de comandos da AWS CLI	Descrições dos comandos da AWS CLI que você pode usar para trabalhar com filas.
Regiões e endpoints da	Informações sobre regiões e endpoints do Amazon SQS
Páginas do produtos	A principal página da Web para obter informações sobre o Amazon SQS.
Fórum de discussão	Um fórum comunitário para que os desenvolvedores discutam questões técnicas relacionadas ao Amazon SQS.
AWS Informações sobre o Premium Support	A principal página da Web para obter informações sobre o AWS Premium Support, um canal de suporte de resposta rápida e com atendimento individual, para ajudar você a desenvolver e executar aplicações nos serviços de infraestrutura da AWS.

Histórico de documentação

A tabela a seguir descreve as alterações importantes feitas no Guia do desenvolvedor do Amazon Simple Queue Service desde janeiro de 2019. Para receber notificações sobre atualizações dessa documentação, inscreva-se no [feed RSS](#).

Os recursos do serviço às vezes são implementados de forma incremental nas regiões da AWS onde um serviço está disponível. Atualizamos esta documentação apenas para a primeira versão. Não fornecemos informações sobre a disponibilidade da região nem anunciamos lançamentos subsequentes da região. Para obter informações sobre a disponibilidade de recursos de serviços em regiões e se inscrever para receber notificações sobre atualizações, consulte [Quais as novidades da AWS?](#).

Alteração	Descrição	Data
Compatibilidade com filas justas	Compatibilidade com filas justas adicionada para filas padrão.	21 de julho de 2025
Adição de suporte para endpoints de pilha dupla (IPv4 e IPv6)	O Amazon SQS agora é compatível com endpoints de pilha dupla (IPv4 e IPv6), permitindo que as filas sejam acessadas por meio de ambos os protocolos IP.	17 de abril de 2025
Integração com CloudTrail para todas as APIs do Amazon SQS	Integração com CloudTrail adicionada para todas as APIs do Amazon SQS.	10 de janeiro de 2025
SQSUnlockQueuePolicy	O Amazon SQS adicionou uma nova política gerenciada pela AWS chamada SQSUnlockQueuePolicy para desbloquear uma fila e remover uma política de fila configurada incorretamente	15 de novembro de 2024

que nega a todas as entidades principais acesso a uma fila do Amazon SQS.

[AWSkms:Decrypt](#)

O Amazon SQS não exige mais a permissão `kms:Decrypt` para a API `SendMessage`. Agora, os clientes só precisam da permissão `kms:GenerateDataKey` na chave do KMS usada para criptografar a fila, mas ainda precisam da permissão `kms:Decrypt` para chamar `ReceiveMessage`.

24 de julho de 2024

[Atualização de métricas de FIFO](#)

Suporte para `NumberOfDuplicatedSentMessages` e `ApproximateNumberOfGroupsWithInflightMessages` adicionado às métricas de FIFO do Amazon SQS.

3 de julho de 2024

[ListQueueTags](#) A ação permitida na política gerenciada [a AmazonSQSReadOnlyAccess](#)

A política gerenciada `AmazonSQSReadOnlyAccess` permite `ListQueueTags` para recuperar todas as tags associadas a uma fila específica do Amazon SQS.

2 de maio de 2024

[AWSProtocolo JSON da](#)

Faça solicitações de API usando o protocolo JSON da AWS.

27 de julho de 2023

[Nova seção que descreve as políticas gerenciadas pela AWS para o Amazon SQS e as atualizações dessas políticas](#)

O Amazon SQS adicionou uma nova ação que permite listar as tarefas mais recentes de movimentação de mensagens (até dez) em uma fila de origem específica. Essa ação está associada à operação de API `ListMessageMoveTasks`.

7 de junho de 2023

[Redirecionamento da fila de mensagens não entregues usando APIs](#)

Configure o redirecionamento de filas de mensagens não entregues usando as APIs do Amazon SQS.

7 de junho de 2023

[ABAC para o Amazon SQS](#)

Controle de acesso por atributo (ABAC) usando tags de fila para permissões de acesso flexíveis e escaláveis.

10 de novembro de 2022

[Aumento do limite de alto throughput de FIFO](#)

Aumento das cotas padrão para o modo de alto throughput de FIFO em regiões comerciais, além da otimização de documentos de alto throughput de FIFO.

20 de outubro de 2022

[A criptografia do lado do servidor \(SSE\) está disponível](#)

Criptografia do lado do servidor (SSE) usando a criptografia do SQS (SSE-SQS) por padrão.

26 de setembro de 2022

[A compatibilidade com a proteção confusa para adjunto do Amazon SQS está disponível](#)

A proteção confusa para adjunto permite que você especifique novos cabeçalhos nas respectivas solicitações, que são verificados em relação às condições da política do KMS ao usar a SSE gerenciada pelo Amazon SQS.

29 de dezembro de 2021

[A SSE gerenciada está disponível](#)

A SSE gerenciada pelo Amazon SQS (SSE-SQS) é uma criptografia gerenciada do lado do servidor que usa chaves de criptografia de propriedade do SQS para proteger dados sigilosos enviados por filas de mensagens.

23 de novembro de 2021

[O redirecionamento da fila de mensagens não entregues está disponível](#)

O Amazon SQS é compatível com o [redirecionamento da fila de mensagens não entregues](#) para filas padrão.

10 de novembro de 2021

[Disponibilidade de throughput alto para mensagens nas filas FIFO](#)

O alto throughput para filas FIFO do Amazon SQS fornece um maior número de transações por segundo (TPS) para mensagens em filas FIFO. Para obter informações sobre cotas de taxa de transferência, consulte [Cotas relacionadas a mensagens](#).

27 de maio de 2021

[Disponibilidade de throughput alto para mensagens nas filas FIFO na versão de visualização](#)

O alto throughput para filas do Amazon SQS FIFO está na versão de pré-visualização e está sujeita a alterações. Esse recurso proporciona um maior número de transações por segundo (TPS) para mensagens em filas FIFO. Para obter informações sobre cotas de taxa de transferência, consulte [Cotas relacionadas a mensagens](#).

17 de dezembro de 2020

[Novo design do console do Amazon SQS](#)

Para simplificar fluxos de trabalho de desenvolvimento e produção, o console do Amazon SQS apresenta uma [nova experiência do usuário](#).

8 de julho de 2020

[Amazon SQS é compatível com a paginação para listQueues e listDeadLetterSourceQueues](#)

É possível especificar o número máximo de resultados a serem retornados de uma solicitação [listQueues](#) ou [listDeadLetterSourceQueues](#).

22 de junho de 2020

[Amazon SQS é compatível com métricas de um minuto do Amazon CloudWatch em todas as regiões da AWS, exceto as regiões AWS GovCloud \(EUA\)](#)

A métrica de um minuto do CloudWatch para o Amazon SQS está disponível em todas as regiões, exceto as regiões AWS GovCloud (US).

9 de janeiro de 2020

[Amazon SQS é compatível com métricas de um minuto do CloudWatch](#)

Atualmente, a métrica do CloudWatch de um minuto para o Amazon SQS está disponível apenas nas regiões a seguir: Leste dos EUA (Ohio), Europa (Irlanda), Europa (Estocolmo) e Ásia-Pacífico (Tóquio).

25 de novembro de 2019

[AWS Lambda Acionadores do para filas FIFO do Amazon SQS estão disponíveis](#)

Você pode configurar as mensagens que chegam a uma fila FIFO como acionadores de função Lambda.

25 de novembro de 2019

[A criptografia do lado do servidor \(SSE\) para o Amazon SQS está disponível nas regiões da China](#)

O SSE para Amazon SQS está disponível nas regiões da China.

13 de novembro de 2019

[As filas FIFO estão disponíveis na região do Oriente Médio \(Bahrein\)](#)

As filas FIFO estão disponíveis na região do Oriente Médio (Bahrein).

10 de outubro de 2019

[Os endpoints da Amazon Virtual Private Cloud \(Amazon VPC\) para o Amazon SQS estão disponíveis nas regiões AWS GovCloud \(Leste dos EUA\) e AWS GovCloud \(Oeste dos EUA\)](#)

Você pode enviar mensagens para suas filas do Amazon SQS pela Amazon VPC nas regiões AWS GovCloud (EUA-Leste) e AWS GovCloud (EUA-Oeste).

5 de setembro de 2019

[Amazon SQS permite a resolução de problemas de filas usando o AWS X-Ray com atributos do sistema de mensagens](#)

É possível solucionar problemas de mensagens transmitidas por filas do Amazon SQS usando o X-Ray. Esta versão adiciona o parâmetro de solicitação `MessageSystemAttribute` (que permite enviar cabeçalhos de rastreamento do X-Ray pelo Amazon SQS) às operações de API `SendMessage` e `SendMessageBatch`, o atributo `AWSTraceHeader` à operação de API [ReceiveMessage](#) e o tipo de dado `MessageSystemAttributeValue`.

28 de agosto de 2019

[É possível etiquetar filas do Amazon SQS após a criação](#)

É possível usar uma única chamada de API do Amazon SQS, função do SDK da AWS ou comando do AWS Command Line Interface (AWS CLI) para criar simultaneamente uma fila e especificar as tags dela. Além disso, o Amazon SQS oferece suporte às chaves `aws:TagKeys` e `aws:RequestTag` do AWS Identity and Access Management (IAM).

22 de agosto de 2019

[O cliente de fila temporária para Amazon SQS já está disponível](#)

Filas temporárias ajudam você a economizar tempo de desenvolvimento e custos de implantação ao usar padrões comuns de mensagens, como solicitação-resposta. Você pode usar o [Temporary Queue Client](#) para criar filas temporárias de alta taxa de transferência, econômicas e gerenciadas por aplicações.

25 de julho de 2019

[O SSE para Amazon SQS está disponível na região AWS GovCloud \(Leste dos EUA\)](#)

A criptografia no lado do servidor (SSE) para o Amazon SQS está disponível na região AWS GovCloud (EUA-Leste).

20 de junho de 2019

[As filas FIFO estão disponíveis nas regiões Ásia-Pacífico \(Hong Kong\), China \(Pequim\), AWS GovCloud \(Leste dos EUA\) e AWS GovCloud \(Oeste dos EUA\)](#)

As filas FIFO estão disponíveis nas regiões Ásia-Pacífico (Hong Kong), China (Pequim), AWS GovCloud (EUA-Leste) e AWS GovCloud (EUA-Oeste).

15 de maio de 2019

[As políticas de endpoint da Amazon VPC estão disponíveis para o Amazon SQS](#)

Você pode criar políticas de endpoint da Amazon VPC para o Amazon SQS

4 de abril de 2019

[As filas FIFO estão disponíveis nas regiões Europa \(Estocolmo\) e China \(Ningxia\)](#)

As filas FIFO estão disponíveis nas regiões Europa (Estocolmo) e China (Ningxia).

14 de março de 2019

[As filas FIFO estão disponíveis em todas as regiões em que o Amazon SQS está disponível](#)

As filas FIFO estão disponíveis nas regiões Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Norte da Califórnia), Oeste dos EUA (Oregon), Ásia-Pacífico (Mumbai), Ásia-Pacífico (Seul), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda), Europa (Londres), Europa (Paris) e América do Sul (São Paulo).

7 de fevereiro de 2019

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.