aws

Entwicklerhandbuch

Amazon Simple Queue Service

Amazon Simple Queue Service: Entwicklerhandbuch

Table of Contents

Was ist Amazon SQS? ... 1
Vorteile der Verwendung von Amazon SQS ... 1
Grundlegende Architektur ... 2
Verteilte Warteschlangen ... 2
Lebenszyklus einer Nachricht ... 2
Unterschiede zwischen Amazon SQS, Amazon MQ und Amazon SNS ... 4
Einrichtung ... 6
Schritt 1: Erstellen Sie einen AWS-Konto und IAM-Benutzer ... 6
Melden Sie sich an für ein AWS-Konto ... 6
Erstellen Sie einen Benutzer mit Administratorzugriff ... 7
Schritt 2: Erteilen programmgesteuerten Zugriffs ... 8
Schritt 3: Vorbereiten der Verwendung des Beispiel-Codes ... 10
Nächste Schritte ... 11
Erste Schritte ... 12
Voraussetzungen ... 12
Die Amazon-SQS-Konsole verstehen ... 12
Warteschlangentypen ... 14
Erstellen einer Standard-Warteschlange ... 15
Erstellen einer Warteschlange ... 15
Senden einer Nachricht ... 18
Erstellen einer FIFO-Warteschlange ... 18
Erstellen einer Warteschlange ... 18
Senden einer Nachricht ... 21
Verwalten einer Warteschlange ... 23
Voraussetzungen ... 12
Die Amazon-SQS-Konsole verstehen ... 12
Eine Warteschlange bearbeiten ... 24
Empfangen und Löschen einer Nachricht ... 25
Feststellen, ob eine Warteschlange leer ist ... 26
Löschen einer Warteschlange ... 28
Bereinigen einer Warteschlange ... 29
Allgemeine Aufgaben ... 29
Standard-Warteschlangen ... 31
Nachrichtenreihenfolge ... 32

Eine t-least-once Lieferung ... 32
Warteschlangen- und Nachrichten-IDs ... 32
IDs für Standard-Warteschlangen ... 32
Kontingente ... 34
FIFO-Warteschlangen ... 37
FIFO-Bereitstellungslogik ... 38
Reihenfolge von FIFO-Warteschlangennachrichten ... 40
Garantiert einmalige Verarbeitung ... 40
Wechseln von einer Standard- zu einer FIFO-Warteschlange ... 41
Hoher Durchsatz für FIFO-Warteschlangen ... 42
Anwendungsfälle ... 43
Partitionen und Datenverteilung ... 44
Aktivieren des hohen Durchsatzes für FIFO-Warteschlangen ... 47
Wichtige Begriffe ... 48
Kompatibilität ... 49
Warteschlangen- und Nachrichten-IDs ... 49
IDs für FIFO-Warteschlangen ... 32
Zusätzliche Kennungen für FIFO-Warteschlangen ... 51
Kontingente ... 52
FIFO-Warteschlangenkontingente ... 52
Amazon-SQS-Kontingente ... 52
Nachrichtenkontingente ... 54
Richtlinienkontingente ... 58
Funktionen und Funktionen ... 60
Warteschlangen für unzustellbare Nachrichten ... 60
Verwenden von Richtlinien für Warteschlangen mit unzustellbaren Briefen ... 61
Grundlegendes zu den Aufbewahrungsfristen für Nachrichten in Warteschlangen mit unerlaubten Briefen ... 62
Konfigurieren einer Queue für unzustellbare Nachrichten ... 62
Konfigurieren eines Redrives einer Warteschlange für unzustellbare Nachrichten ... 63
CloudTrail Aktualisierungs- und Genehmigungsanforderungen ... 71
Mit Amazon Alarme für Warteschlangen mit unzustellbaren Briefen erstellen CloudWatch ... 75
Nachrichten-Metadaten für Amazon SQS ... 76
Nachrichtenattribute ... 76
Nachrichtensystemattribute ... 80
Für die Nachrichtenverarbeitung erforderliche Ressourcen ... 81

Auflistung der Warteschlangenpaginierung ... 82
Kostenzuordnungs-Tags ... 82
Kurz- und Langabfragen ... 83
Abrufen von Nachrichten durch Kurzabfragen ... 84
Konsumieren von Nachrichten mithilfe von Langabfragen ... 85
Unterschiede zwischen Lang- und Kurzabfragen ... 86
Zeitbeschränkung für die Sichtbarkeit ... 86
In-Flight-Nachrichten ... 88
Einrichten der Zeitbeschränkung für die Sichtbarkeit ... 89
Ändern der Zeitbeschränkung für die Sichtbarkeit für eine Nachricht ... 90
Beenden der Zeitbeschränkung für die Sichtbarkeit für eine Nachricht ... 91
Verzögerungswarteschlangen ... 91
Temporäre Warteschlangen ... 92
Virtuelle Warteschlangen ... 93
Request-Response-Messaging-Muster (virtuelle Warteschlangen) ... 95
Beispielszenario: Verarbeiten einer Anmeldeanforderung ... 95
Bereinigen von Warteschlangen ... 97
Nachrichten-Timer ... 98
Zugreifen auf EventBridge Pipes ... 99
Verwalten großer Nachrichten ... 100
Verwendung der Extended Client Library für Java ... 101
Verwenden der Extended Client Library für Python ... 111
Konfiguration von Amazon SQS ... 115
ABAC für Amazon SQS ... 115
Was ist ABAC? ... 115
Weshalb sollte ich ABAC in Amazon SQS verwenden? ... 116
ABAC-Bedingungsschlüssel ... 117
Markierungen für die Zugriffssteuerung ... 118
Erstellen von IAM-Benutzern und Amazon-SQS-Warteschlangen ... 119
Testen der attributbasierten Zugriffssteuerung ... 122
Konfiguration von Warteschlangenparametern ... 124
Konfigurieren von Zugriffsrichtlinien ... 126
Konfigurieren von SSE-SQS für eine Warteschlange ... 127
Konfigurieren von SSE-KMS für eine Warteschlange ... 128
Konfigurieren von Tags für eine Warteschlange ... 130
Abonnieren einer Warteschlange für ein Thema ... 131

Konfigurieren eines Lambda-Auslösers ... 132
Voraussetzungen ... 133
Automatisieren von Benachrichtigungen mit EventBridge ... 134
Nachrichtenattribute ... 135
Bewährte Methoden ... 137
Empfehlungen für Standard- und FIFO-Warteschlangen ... 137
Arbeiten mit Nachrichten ... 137
Reduzieren von Kosten ... 142
Wechseln von einer Standard- zu einer FIFO-Warteschlange ... 143
Zusätzliche Empfehlungen für FIFO-Warteschlangen ... 143
Verwenden der Nachrichtendeduplizierungs-ID ... 143
Verwenden der Nachrichtengruppen-ID ... 145
Verwenden der Empfangsanforderungsversuch-ID ... 147
Java-SDK-Beispiele ... 148
Verwenden der serverseitigen Verschlüsselung ... 148
Hinzufügen von SSE zu einer vorhandenen Warteschlange ... 148
Deaktivieren von SSE für eine Warteschlange ... 149
Erstellen einer Warteschlange mit SSE ... 150
Abrufen der SSE-Attribute ... 150
Konfigurieren von Tags ... 151
Auflisten von Tags ... 151
Hinzufügen oder Aktualisieren von Tags ... 151
Entfernen von Tags ... 152
Senden von Nachrichtenattributen ... 153
Definieren von Attributen ... 153
Senden einer Nachricht mit Attributen ... 155
Arbeiten mit APIs ... 156
Abfrage-API-Anfragen mithilfe des AWS JSON-Protokolls stellen ... 157
Erstellen eines Endpunkts ... 158
Durchführen einer POST-Anforderung ... 159
Interpretieren von Amazon-SQS-JSON-API-Antworten ... 160
Häufig gestellte Fragen zum Amazon SQS AWS SQS-JSON-Protokoll ... 161
Abfrage-API-Anfragen mithilfe des AWS Abfrageprotokolls stellen ... 164
Erstellen eines Endpunkts ... 165
Durchführen einer GET-Anforderung ... 166
Durchführen einer POST-Anforderung ... 159

Interpretieren von Amazon-SQS-XML-API-Antworten ... 167
Authentifizieren von Anforderungen ... 169
Grundlegender Authentifizierungsprozess mit HMAC-SHA ... 170
Teil 1: Die Anforderung von dem Benutzer ... 171
Teil 2: Die Antwort von AWS ... 172
Stapelaktionen ... 173
Aktivierung der clientseitigen Pufferung und der Batchverarbeitung von Anfragen mit Amazon SQS ... 174
Steigerung des Durchsatzes durch horizontale Skalierung und Action-Batching mit Amazon SQS ... 183
Arbeiten mit JMS ... 197
Voraussetzungen ... 197
Erste Schritte mit der Java Messaging Library ... 199
Erstellen einer JMS-Verbindung ... 199
Erstellen einer Amazon-SQS-Warteschlange ... 200
Synchrones Senden von Nachrichten ... 201
Synchrones Empfangen von Nachrichten ... 202
Asynchrones Empfangen von Nachrichten ... 204
Verwenden des Client-Bestätigungsmodus ... 205
Verwenden des ungeordneten Bestätigungsmodus ... 206
Verwenden des JMS-Clients mit anderen Amazon-SQS-Clients ... 207
Funktionierende Java-Beispiele für die Verwendung von JMS mit Standardwarteschlangen ... 208
ExampleConfiguration.java ... 209
TextMessageSender.java ... 211
SyncMessageReceiver.java ... 213
AsyncMessageReceiver.java ... 215
SyncMessageReceiverClientAcknowledge.java ... 217
SyncMessageReceiverUnorderedAcknowledge.java ... 221
SpringExampleConfiguration.xml ... 224
SpringExample.java ... 226
ExampleCommon.java ... 228
Unterstützte JMS 1.1-Implementierungen ... 230
Unterstützte gängige Schnittstellen ... 230
Unterstützte Nachrichtentypen ... 230
Unterstützte Nachrichtenbestätigungsmodi ... 230
JMS-definierte Kopfzeilen und reservierte Eigenschaften ... 231

Tutorials ... 232
Erstellen einer Amazon SQS SQS-Warteschlange mit AWS CloudFormation ... 232
Senden einer Nachricht von einer VPC ... 234
Schritt 1: Erstellen eines Amazon EC2-Schlüsselpaares ... 235
Schritt 2: Ressourcen erstellen AWS ... 236
Schritt 3: Bestätigen, dass Ihre EC2-Instance nicht öffentlich zugänglich ist ... 237
Schritt 4: Erstellen eines Amazon-VPC-Endpunkts für Amazon SQS ... 238
Schritt 5: Senden einer Nachricht an Ihre Amazon-SQS-Warteschlange ... 239
Fehlerbehebung ... 241
Fehler „Zugriff verweigert“ ... 241
Amazon SQS SQS-Warteschlangenrichtlinie und IAM-Richtlinie ... 242
AWS Key Management Service (AWS KMS) Berechtigungen ... 243
VPC-Endpunktrichtlinie ... 244
Richtlinie zur Dienstkontrolle der Organisation ... 245
API-Fehler ... 245
QueueDoesNotExist Fehler ... 246
InvalidAttributeValue Fehler ... 246
ReceiptHandle Fehler ... 247
Probleme mit DLQ und DLQ Redrive ... 248
DLQ-Probleme ... 248
Probleme mit DLQ-Redrive ... 250
Probleme mit der FIFO-Drosselung ... 252
Nachrichten, die bei einem ReceiveMessage API-Aufruf nicht zurückgegeben wurden ... 253
Leere Warteschlange ... 254
Limit während des Fluges erreicht ... 254
Verzögerung der Nachricht ... 254
Die Nachricht ist im Flug ... 254
Methode der Umfrage ... 255
Netzwerkfehler ... 255
ETIMEOUT error ... 255
UnknownHostException error ... 257
Fehlerbehebung bei Warteschlangen mit X-Ray ... 258
Sicherheit ... 259
Datenschutz ... 259
Datenverschlüsselung ... 260
Richtlinie für den Datenverkehr zwischen Netzwerken ... 273

Identity and Access Management 275
Zielgruppe 276
Authentifizierung mit Identitäten 276
Verwalten des Zugriffs mit Richtlinien 280
Übersicht 283
So funktioniert Amazon Simple Notification Service mit IAM 291
AWS verwaltete Richtlinien 299
Fehlerbehebung 301
Verwenden von -Richtlinien 303
Protokollierung und Überwachung 352
Protokollieren von API-Aufrufen mit CloudTrail 352
Überwachen von Warteschlangen mit CloudWatch 366
Compliance-Validierung 381
Ausfallsicherheit 382
Verteilte Warteschlangen 382
Sicherheit der Infrastruktur 383
Bewährte Methoden 384
Sicherstellen, dass Warteschlangen nicht öffentlich zugänglich sind 384
Implementieren der geringstmöglichen Zugriffsrechte 385
Verwenden Sie IAM-Rollen für Anwendungen und AWS Services, für die Amazon SQS SQS-Zugriff erforderlich ist 385
Implementieren serverseitiger Verschlüsselung 386
Erzwingen der Verschlüsselung von Daten während der Übertragung 386
Erwägen der Verwendung von VPC-Endpunkten für den Zugriff auf Amazon SQS 386
Zugehörige Ressourcen 388
Dokumentationsverlauf 389
..... cccxcvi

Was ist Amazon Simple Queue Service

Amazon Simple Queue Service (Amazon SQS) bietet eine sichere, dauerhafte und verfügbare gehostete Warteschlange, die es Ihnen ermöglicht, verteilte Softwaresysteme und -komponenten zu integrieren und zu entkoppeln. Amazon SQS bietet gängige Konstrukte, wie z. B. Warteschlangen für unzustellbare Nachrichten und Kostenzuordnungs-Tags. Es bietet eine generische Webservice-API, auf die Sie mit jeder vom AWS SDK unterstützten Programmiersprache zugreifen können.

Themen

- Vorteile der Verwendung von Amazon SQS
- Grundlegende Amazon-SQS-Architektur
- Unterschiede zwischen Amazon SQS, Amazon MQ und Amazon SNS

Vorteile der Verwendung von Amazon SQS

- Sicherheit – Sie steuern, wer Mitteilungen an eine Warteschlange senden und Mitteilungen von einer Amazon-SQS-Warteschlange empfangen darf. Sie können wählen, ob Sie vertrauliche Daten übertragen möchten, indem Sie den Inhalt von Nachrichten in Warteschlangen schützen, indem Sie die standardmäßige serverseitige Verschlüsselung (SSE) von Amazon SQS oder benutzerdefinierte SSE-Schlüssel verwenden, die in AWS Key Management Service (AWS KMS) verwaltet werden.
- Dauerhaftigkeit – Um die Sicherheit Ihrer Nachrichten zu gewährleisten, speichert Amazon SQS sie auf mehreren Servern. Standardwarteschlangen unterstützen die at-least-once Nachrichtenzustellung, und FIFO-Warteschlangen unterstützen die Verarbeitung von Nachrichten genau einmal und den Modus mit hohem Durchsatz.
- Verfügbarkeit – Amazon SQS verwendet eine redundante Infrastruktur für den simultanen Zugriff auf Nachrichten und hohe Verfügbarkeit zum Erstellen und Verwenden von Nachrichten.
- Skalierbarkeit – Amazon SQS kann jede gepufferte Anfrage unabhängig verarbeiten und transparent skalieren, um sämtliche zunehmenden Lasten oder Spitzen ohne Bereitstellungsanweisungen zu verarbeiten.
- Zuverlässigkeit – Amazon SQS sperrt Ihre Nachrichten während der Verarbeitung, damit mehrere Produzenten und mehrere Konsumenten Nachrichten gleichzeitig senden und empfangen können.
- Anpassung – Ihre Warteschlangen müssen nicht genau übereinstimmen, sie können z. B. Standardverzögerung für eine Warteschlange festlegen. Sie können den Inhalt der Nachrichten

mit mehr als 256 KB mithilfe von Amazon Simple Storage Service (Amazon S3) oder Amazon DynamoDB speichern, wobei Amazon SQS einen Verweis auf das Amazon-S3-Objekt enthält. Sie können große Nachrichten auch in kleinere Nachrichten aufteilen.

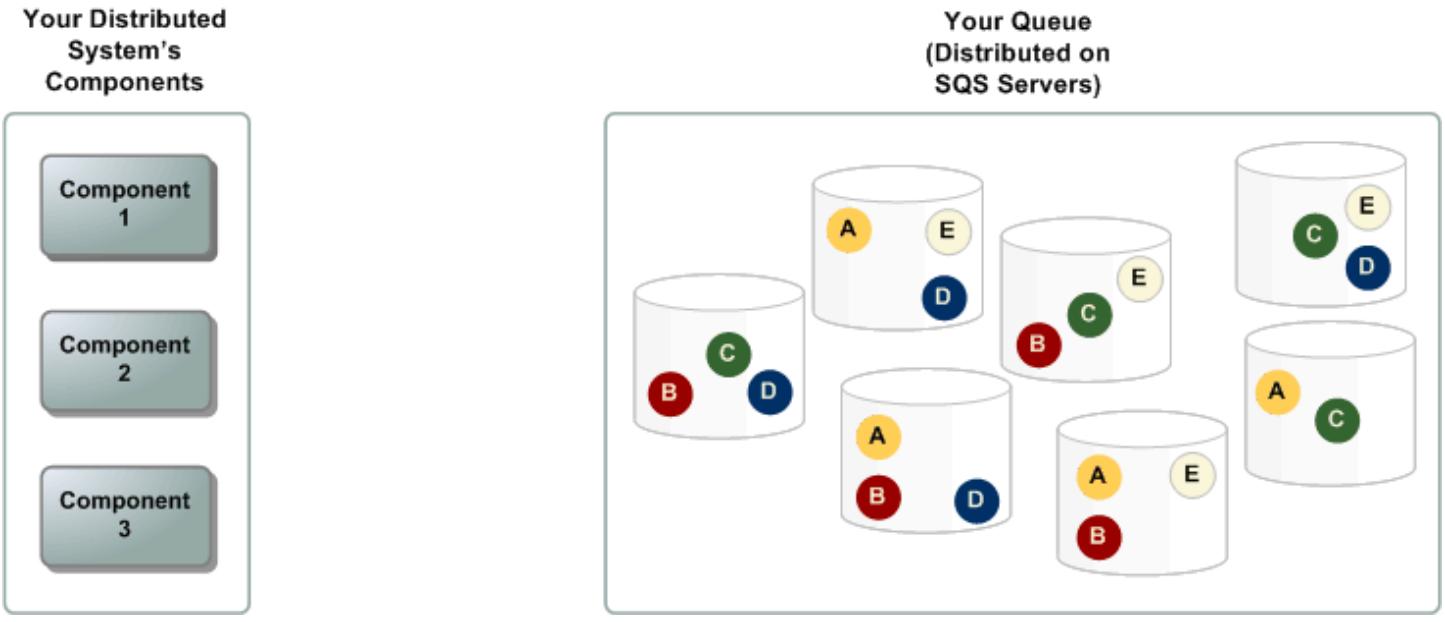
Grundlegende Amazon-SQS-Architektur

In diesem Abschnitt werden die Teile eines verteilten Messaging-Systems aufgeführt. Zudem wird der Lebenszyklus einer Amazon-SQS-Nachricht erläutert.

Verteilte Warteschlangen

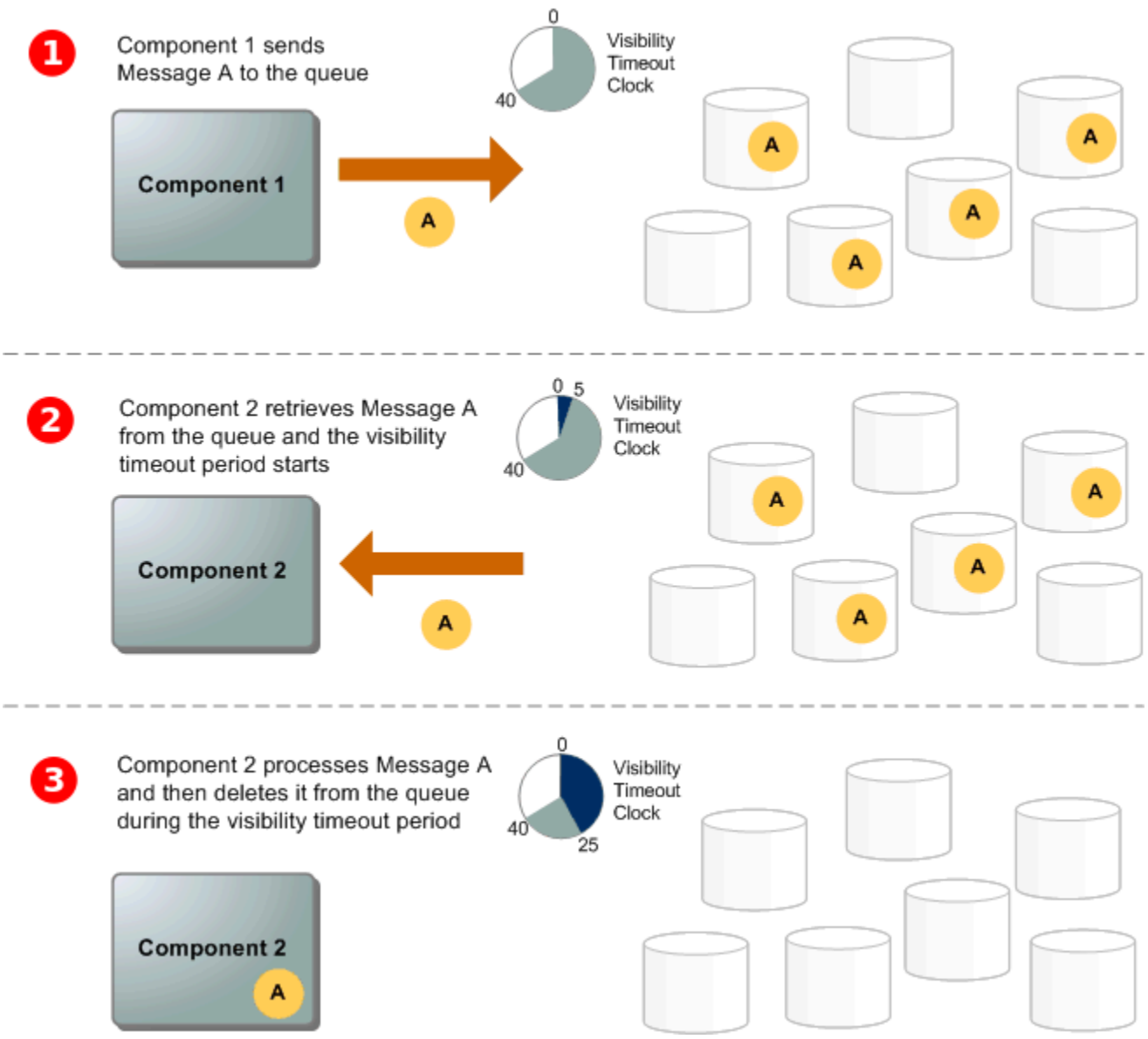
Es gibt drei Hauptkomponenten in einem verteilten Messaging-System: die Komponenten Ihres verteilten Systems, Ihre Warteschlange (auf Amazon-SQS-Server verteilt) und die Nachrichten in der Warteschlange.

Im folgenden Szenario hat Ihr System mehrere Produzenten (Komponenten, die Nachrichten an die Warteschlange senden) und Konsumenten (Komponenten, die Nachrichten aus der Warteschlange empfangen). Die Warteschlange (die die Nachrichten A bis E enthält) speichert die Nachrichten redundant auf mehreren Amazon-SQS-Servern.



Lebenszyklus einer Nachricht

Das folgende Szenario stellt den Lebenszyklus einer Amazon-SQS-Nachricht in einer Warteschlange von der Erstellung zur Löschung dar.



1

Ein Produzent (Komponente 1) sendet Nachricht A an eine Warteschlange und die Nachricht wird redundant über die Amazon-SQS-Server verteilt.

2

Wenn ein Konsument (Komponente 2) bereit ist, Nachrichten zu verarbeiten, werden Nachrichten aus der Warteschlange konsumiert und Nachricht A wird zurückgegeben. Während Nachricht A verarbeitet wird, verbleibt sie in der Warteschlange und wird während der Zeitbeschränkung für die Sichtbarkeit nicht an nachfolgende Empfangsanforderungen zurückgegeben.

3

Der Konsument (Komponente 2) löscht Nachricht A aus der Warteschlange, um zu verhindern, dass die Nachricht nach Ablauf der Zeitbeschränkung für die Sichtbarkeit erneut empfangen und verarbeitet wird.

> **Note**
>
> Amazon SQS löscht Nachrichten automatisch, die sich länger als den maximalen Aufbewahrungszeitraum für Nachrichten in einer Warteschlange befunden haben. Der Standardaufbewahrungszeitraum für Nachrichten beträgt 4 Tage. Sie können jedoch den Aufbewahrungszeitraum für Nachrichten mit der Aktion `SetQueueAttributes` von 60 Sekunden auf 1.209.600 Sekunden (14 Tage) festlegen.

Unterschiede zwischen Amazon SQS, Amazon MQ und Amazon SNS

Amazon SQS, Amazon SNS und Amazon MQ bieten hoch skalierbare und easy-to-use verwaltete Messaging-Dienste, die jeweils für bestimmte Rollen innerhalb verteilter Systeme konzipiert sind. Hier finden Sie einen erweiterten Überblick über die Unterschiede zwischen diesen Diensten:

Amazon SQS entkoppelt und skaliert verteilte Softwaresysteme und Komponenten als Warteschlangenservice. Es verarbeitet Nachrichten in der Regel über einen einzigen Abonnenten. Dies ist ideal für Workflows, bei denen Ordnung und Verlustprävention von entscheidender Bedeutung sind. Für eine breitere Verbreitung ermöglicht die Integration von Amazon SQS mit Amazon SNS ein Fanout-Messaging-Muster, das Nachrichten effektiv an mehrere Abonnenten gleichzeitig weiterleitet.

Amazon SNS ermöglicht es Verlagen, Nachrichten über Themen, die als Kommunikationskanäle dienen, an mehrere Abonnenten zu senden. Abonnenten erhalten veröffentlichte Nachrichten über einen unterstützten Endpunkttyp wie Amazon SQS Amazon Data Firehose, Lambda, HTTP, E-Mail, mobile Push-Benachrichtigungen und mobile Textnachrichten (SMS). Dieser Service ist ideal für Szenarien, in denen sofortige Benachrichtigungen erforderlich sind, z. B. für Benutzerinteraktionen in Echtzeit oder Alarmsysteme. Um Nachrichtenverlust zu verhindern, wenn Abonnenten offline sind, sorgt die Integration von Amazon SNS mit Amazon SQS SQS-Warteschlangennachrichten für eine konsistente Zustellung.

Amazon MQ eignet sich am besten für Unternehmen, die von herkömmlichen Message-Brokern migrieren möchten. Es unterstützt Standard-Messaging-Protokolle wie AMQP und MQTT sowie Apache ActiveMQ und RabbitMQ. Es bietet Kompatibilität mit älteren Systemen, die ein stabiles, zuverlässiges Messaging benötigen, ohne dass eine umfangreiche Neukonfiguration erforderlich ist.

Die folgende Tabelle bietet einen Überblick über die Ressourcentypen der einzelnen Dienste:

Ressourcentyp	Amazon SNS	Amazon SQS	Amazon MQ
Synchron	Nein	Nein	Ja
Asynchron	Ja	Ja	Ja
Warteschlangen	Nein	Ja	Ja
Messaging zwischen Publisher und Subscriber	Ja	Nein	Ja
Message Broker	Nein	Nein	Ja

Wir empfehlen Amazon SQS und Amazon SNS für neue Anwendungen, die von praktisch unbegrenzter Skalierbarkeit und einfacher APIs profitieren können. Sie bieten aufgrund ihrer pay-as-you-go Preisgestaltung im Allgemeinen kostengünstigere Lösungen für großvolumige Anwendungen. Wir empfehlen Amazon MQ für die Migration von Anwendungen von bestehenden Message Brokern, die auf Kompatibilität mit APIs wie JMS oder Protokollen wie Advanced Message Queuing Protocol (AMQP), MQTT und Simple Text Oriented Message Protocol (OpenWireSTOMP) angewiesen sind.

Einrichten von Amazon SQS

Bevor Sie Amazon SQS zum ersten Mal verwenden können, müssen Sie die folgenden Schritte abschließen.

Themen

- Schritt 1: Erstellen Sie einen AWS-Konto und IAM-Benutzer
- Schritt 2: Erteilen programmgesteuerten Zugriffs
- Schritt 3: Vorbereiten der Verwendung des Beispiel-Codes
- Nächste Schritte

Schritt 1: Erstellen Sie einen AWS-Konto und IAM-Benutzer

Um auf einen AWS Service zugreifen zu können, müssen Sie zunächst ein Amazon.com-Konto erstellen AWS-Konto, mit dem Produkte verwendet AWS werden können. Sie können Ihren verwenden AWS-Konto , um Ihre Aktivitäts- und Nutzungsberichte einzusehen und die Authentifizierung und den Zugriff zu verwalten.

Um zu vermeiden, dass Ihr AWS-Konto Root-Benutzer für Amazon SQS-Aktionen verwendet wird, empfiehlt es sich, für jede Person, die Administratorzugriff auf Amazon SQS benötigt, einen IAM-Benutzer zu erstellen.

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus

Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Melden Sie sich als Benutzer mit Administratorzugriff an

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

Schritt 2: Erteilen programmgesteuerten Zugriffs

Um Amazon SQS SQS-Aktionen zu verwenden (z. B. mit Java oder über AWS Command Line Interface), benötigen Sie eine Zugriffsschlüssel-ID und einen geheimen Zugriffsschlüssel.

> **Note**
>
> Die Zugriffsschlüssel-ID und der geheime Zugriffsschlüssel sind spezifisch für. AWS Identity and Access Management Verwechseln Sie sie nicht mit Anmeldeinformationen für andere AWS Dienste, wie z. B. Amazon EC2 EC2-Schlüsselpaaren.

Benutzer benötigen programmgesteuerten Zugriff, wenn sie mit AWS außerhalb des interagieren möchten. AWS Management Console Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt von der Art des Benutzers ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zu AWS IAM Identity Center verwendenden im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs, Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch für AWS SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldedaten im Benutzerhandbuch. AWS

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		CLIAWS Command Line Interface • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools. • Informationen zu AWS APIs finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Schritt 3: Vorbereiten der Verwendung des Beispiel-Codes

Dieses Handbuch enthält Beispiele, die das AWS SDK for Java verwenden. Um den Beispiel-Code auszuführen, folgen Sie den Anweisungen zur Einrichtung unter Erste Schritte mit AWS SDK für Java 2.0.

Sie können AWS Anwendungen in anderen Programmiersprachen wie GoJavaScript, Python und Ruby entwickeln. Weitere Informationen finden Sie unter Tools, auf denen Sie aufbauen können AWS.

> **Note**
>
> Mit Tools wie AWS Command Line Interface (AWS CLI) oder Windows PowerShell können Sie Amazon SQS erkunden, ohne Code schreiben zu müssen. AWS CLI Beispiele finden Sie im Abschnitt Amazon SQS der AWS CLI Befehlsreferenz. PowerShell Windows-Beispiele finden Sie im Abschnitt Amazon Simple Queue Service der AWS Tools for PowerShell Cmdlet-Referenz.

Nächste Schritte

Sie sind jetzt bereit für Erste Schritte mit der Verwaltung von Amazon-SQS-Warteschlangen und -Nachrichten mithilfe der AWS Management Console.

Erste Schritte mit Amazon SQS

In diesem Abschnitt erfahren Sie, wie Sie mit der Amazon SQS SQS-Konsole Standard- oder FIFO-Warteschlangen erstellen.

Themen

- Voraussetzungen
- Die Amazon-SQS-Konsole verstehen
- Amazon-SQS-Warteschlangentypen
- Erstellen einer Amazon-SQS-Standard-Warteschlange und Senden einer Nachricht
- Erstellen einer Amazon-SQS-FIFO-Warteschlange und Senden einer Nachricht

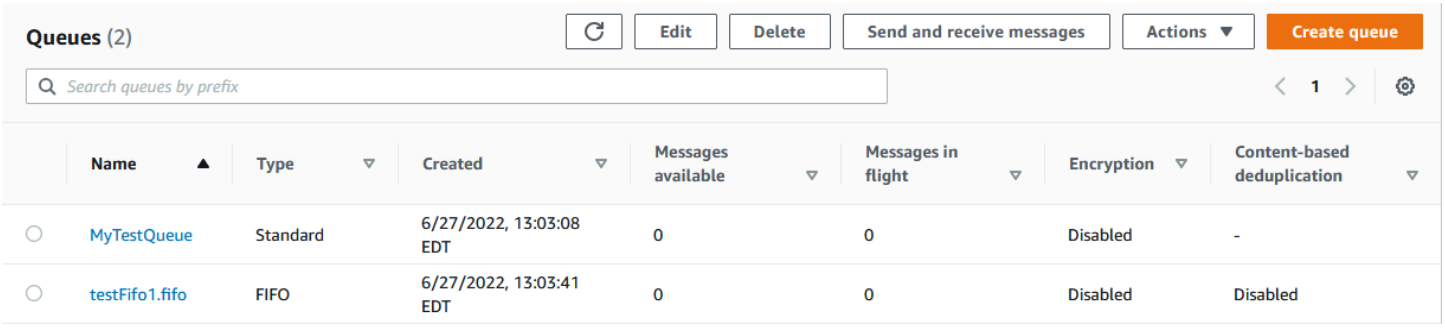
Voraussetzungen

Bevor Sie beginnen, führen Sie die Schritte in Einrichten von Amazon SQS aus.

Die Amazon-SQS-Konsole verstehen

Wenn Sie die Amazon SQS SQS-Konsole öffnen, wählen Sie im Navigationsbereich Warteschlangen aus. Die Seite Warteschlangen enthält Informationen zu allen Ihren Warteschlangen in der aktiven Region.

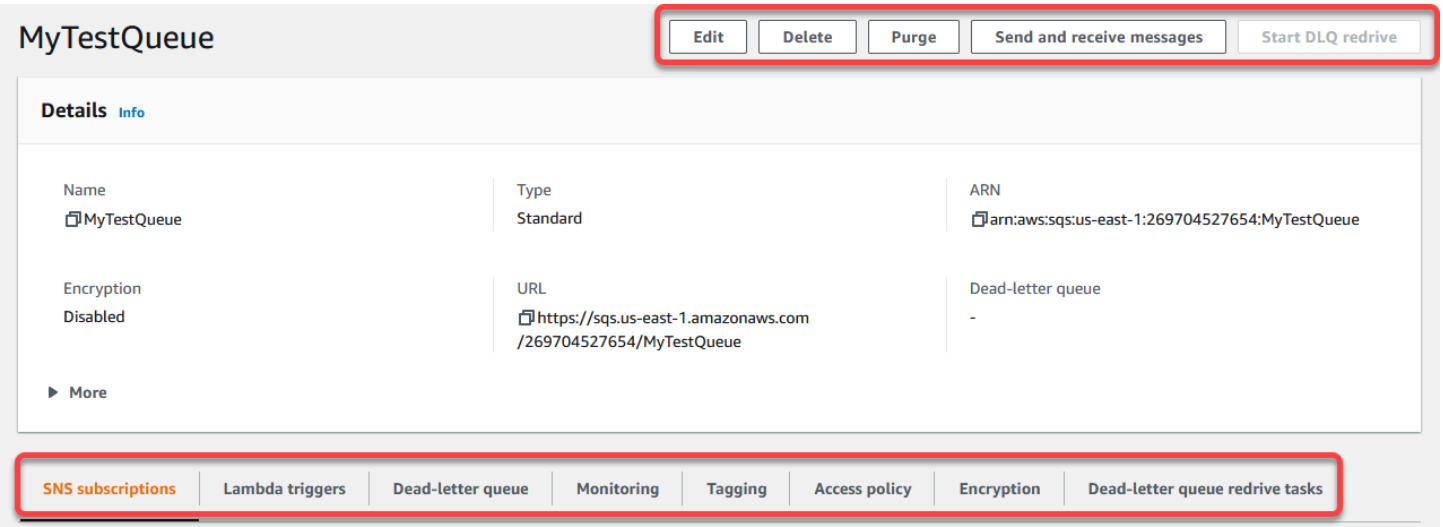
Jeder Warteschlangeneintrag enthält wichtige Informationen über die Warteschlange, einschließlich ihres Typs und ihrer wichtigsten Attribute. Standardwarteschlangen, die für maximalen Durchsatz und optimale Nachrichtenreihenfolge optimiert sind, unterscheiden sich von First-In-First-Out (FIFO) - Warteschlangen, bei denen die Reihenfolge und Eindeutigkeit von Nachrichten für Anwendungen, die eine strikte Nachrichtensequenzierung erfordern, Priorität eingeräumt wird.



Interaktive Elemente und Aktionen

Auf der Seite Warteschlangen haben Sie mehrere Möglichkeiten, Ihre Warteschlangen zu verwalten:

1. Schnellaktionen — Neben jedem Warteschlangennamen bietet ein Dropdownmenü schnellen Zugriff auf häufig verwendete Aktionen wie das Senden von Nachrichten, das Anzeigen oder Löschen von Nachrichten, das Konfigurieren von Triggern und das Löschen der Warteschlange selbst.
2. Detaillierte Ansicht und Konfiguration — Wenn Sie auf einen Warteschlangennamen klicken, wird die zugehörige Detailseite geöffnet, auf der Sie sich eingehender mit den Einstellungen und Konfigurationen der Warteschlange befassen können. Hier können Sie Parameter wie die Aufbewahrungsdauer von Nachrichten, das Zeitlimit für die Sichtbarkeit und die maximale Nachrichtengröße anpassen, um die Warteschlange an die Anforderungen Ihrer Anwendung anzupassen.



Regionsauswahl und Ressourcen-Tags

Stellen Sie sicher, dass Sie richtig sind AWS-Region , um effektiv auf Ihre Warteschlangen zuzugreifen und sie zu verwalten. Erwägen Sie außerdem, Ressourcen-Tags zu verwenden, um Ihre Warteschlangen zu organisieren und zu kategorisieren, um so ein besseres Ressourcenmanagement, eine bessere Kostenzuweisung und Zugriffskontrolle in Ihrer AWS gemeinsamen Umgebung zu ermöglichen.

Durch die Nutzung der in der Amazon SQS SQS-Konsole angebotenen Funktionen können Sie Ihre Messaging-Infrastruktur effizient verwalten, die Warteschlangenleistung optimieren und eine zuverlässige Nachrichtenzustellung für Ihre Anwendungen sicherstellen.

Amazon-SQS-Warteschlangentypen

Amazon SQS unterstützt zwei Arten von Warteschlangen: Standard-Warteschlangen und FIFO-Warteschlangen. Verwenden Sie die Informationen aus der folgenden Tabelle, um die richtige Warteschlange für Ihre Situation auszuwählen. Weitere Informationen zu Amazon-SQS-Warteschlangen finden Sie unter Erste Schritte mit Amazon-SQS-Standard-Warteschlangen und Erste Schritte mit FIFO-Warteschlangen in Amazon SQS.

Standard-Warteschlangen	FIFO-Warteschlangen
Unbegrenzter Durchsatz – Standard-Warteschlangen unterstützen eine nahezu unbegrenzte Anzahl von API-Aufrufen pro Sekunde, pro API-Aktion (`SendMessage` , `ReceiveMessage` oder `DeleteMessage`). Mindestens einmalige Zustellung – Eine Nachricht wird mindestens einmal übermittelt; gelegentlich wird eine Nachricht mehr als einmal gesendet. Bestmögliche Einhaltung von Reihenfolgen – Mitunter werden Nachrichten in einer anderen Reihenfolge zugestellt als der, in der sie gesendet wurden.	Hoher Durchsatz – Wenn Sie die Stapelverarbeitung verwenden, unterstützen FIFO-Warteschlangen bis zu 3 000 Nachrichten pro Sekunde pro API-Methode (`SendMessageBatch` , `ReceiveMessage` oder `DeleteMessageBatch`). Die 3 000 Transaktionen pro Sekunde repräsentieren 300 API-Aufrufe mit jeweils einem Stapel von 10 Nachrichten. Um eine Kontingenterhöhung anzufordern, übermitteln Sie eine Support-Anforderung. Ohne Batching unterstützen FIFO-Warteschlangen bis zu 300 API-Aufrufe pro Sekunde pro API-Methode (`SendMessage` , `ReceiveMessage` , oder `DeleteMessage`). Genau einmalige Verarbeitung – Eine Nachricht wird einmal gesendet und bleibt so lange verfügbar, bis ein Konsument sie gelesen und gelöscht hat. Duplikate werden nicht in die Warteschlange aufgenommen. First-in-First-out-Übermittlung – Die Reihenfolge, in der Nachrichten gesendet und empfangen werden, wird strikt beibehalten.

Standard-Warteschlangen	FIFO-Warteschlangen
	
Übertragen Sie Daten zwischen Anwendungen, wenn der Durchsatz wichtig ist, beispielsweise: • Entkoppeln Sie Anfragen von Live-Benutzern von intensiver Hintergrundarbeit: Benutzer können Medien hochladen, während Sie die Größe der Medien anpassen oder sie kodieren. • Weisen Sie Aufgaben mehreren Worker-Kn oten zu: Bearbeiten Sie eine große Menge an Kreditkarten-Validierungsanfragen. • Bündeln Sie Nachrichten für künftige Verarbeitung: Setzen Sie Zeitpunkte fest, zu denen Einträge in Datenbanken hinzugefügt werden.	Übertragen Sie Daten zwischen Anwendungen, wenn die Reihenfolge der Ereignisse wichtig ist, zum Beispiel: • Stellen Sie sicher, dass vom Benutzer eingegebene Befehle in der richtigen Reihenfolge ausgeführt werden. • Anzeigen des richtigen Produktpreises, indem Preisänderungen in der richtigen Reihenfolge gesendet werden • Verhindern, dass sich ein Student vor dem Erstellen eines Benutzerkontos in einen Kurs einschreibt

Erstellen einer Amazon-SQS-Standard-Warteschlange und Senden einer Nachricht

So erstellen Sie eine Standard-Warteschlange für Amazon SQS:

Erstellen Sie eine Warteschlange mit der Amazon SQS SQS-Konsole

Sie können mit der Amazon-SQS-Konsole Standard-Warteschlangen erstellen. Die Konsole bietet Standardwerte für alle Einstellungen mit Ausnahme des Warteschlangennamens.

> **Important**
>
> Am 17. August 2022 wurde die serverseitige Verschlüsselung (SSE) standardmäßig auf alle Amazon-SQS-Warteschlangen angewendet.

Fügen Sie keine persönlich identifizierbare Informationen (PII) oder andere vertrauliche oder sensible Informationen in Warteschlangennamen hinzu. Warteschlangennamen sind für viele Amazon Web Services zugänglich, einschließlich Abrechnung und CloudWatch Protokollen. Warteschlangennamen sind nicht für private oder sensible Daten gedacht.

So erstellen Sie eine Amazon-SQS-Standard-Warteschlange

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie Create queue (Warteschlange erstellen) aus.
3. Für Typ ist standardmäßig der Standard-Warteschlangentyp festgelegt.

   > **Note**
   >
   > Sie können den Warteschlangentyp nicht ändern, nachdem Sie die Warteschlange erstellt haben.

4. Geben Sie einen Namen für die Warteschlange ein.
5. (Optional) Die Konsole legt Standardwerte für die Konfigurationsparameter der Warteschlange fest. Unter Konfiguration können Sie neue Werte für die folgenden Parameter festlegen:

   a. Geben Sie für Sichtbarkeitszeitbeschränkung die Dauer und die Einheiten ein. Der Bereich liegt zwischen 0 und 12 Stunden. Der Standardwert ist 30 Sekunden.

   b. Geben Sie unter Aufbewahrungszeitraum für Nachrichten die Dauer und die Einheiten ein. Der Bereich liegt zwischen 1 Minute und 14 Tagen. Der Standardwert ist 4 Tage.

   c. Geben Sie für Zustellungsverzögerung die Dauer und die Einheiten ein. Der Bereich liegt zwischen 0 Sekunden und 15 Minuten. Der Standardwert ist 0 Sekunden.

   d. Geben Sie für Maximale Nachrichtengröße einen Wert ein. Der Bereich reicht von 1 KB bis 256 KB. Der Standardwert ist 256 KB.

   e. Geben Sie für Wartezeit für den Empfang von Nachrichten einen Wert ein. Der Bereich liegt zwischen 0 und 20 Sekunden. Der Standardwert ist 0 Sekunden, der kurze Abfragen festlegt. Jeder Wert ungleich Null führt zu einer langen Abfrage.

6. (Optional) Definieren Sie eine Zugriffsrichtlinie. Die Zugriffsrichtlinie definiert die Konten, Benutzer und Rollen, die auf die Warteschlange zugreifen können. Die Zugriffsrichtlinie definiert auch die Aktionen (wie `SendMessage`, `ReceiveMessage` oder `DeleteMessage`),

auf die die Benutzer zugreifen können. Die Standardrichtlinie erlaubt nur dem Eigentümer der Warteschlange, Nachrichten zu senden und zu empfangen.

Führen Sie zum Definieren der Zugriffsrichtlinie einen der folgenden Schritte aus:

- Wählen Sie Einfach, um zu konfigurieren, wer Nachrichten an die Warteschlange senden und wer Nachrichten aus der Warteschlange empfangen kann. Die Konsole erstellt die Richtlinie auf der Grundlage Ihrer Auswahl und zeigt die resultierende Zugriffsrichtlinie im schreibgeschützten JSON-Bereich an.
- Wählen Sie Erweitert, um die JSON-Zugriffsrichtlinie direkt zu ändern. Auf diese Weise können Sie einen benutzerdefinierten Satz von Aktionen angeben, die jeder Prinzipal (Konto, Benutzer oder Rolle) ausführen kann.

7. Wählen Sie für die Redrive-Zulassungsrichtlinie die Option Aktiviert aus. Wählen Sie eine der folgenden Optionen aus: Alle zulassen, Nach Warteschlange oder Alle verweigern. Wenn Sie Nach Warteschlange wählen, geben Sie eine Liste mit bis zu 10 Quellwarteschlangen nach dem Amazon-Ressourcennamen (ARN) an.
8. Amazon SQS bietet standardmäßig verwaltete serverseitige Verschlüsselung. Um einen Verschlüsselungsschlüsseltyp auszuwählen oder die von Amazon SQS verwaltete serverseitige Verschlüsselung zu deaktivieren, erweitern Sie Verschlüsselung. Weitere Informationen zu Verschlüsselungsschlüsseltypen finden Sie unter Konfiguration der serverseitigen Verschlüsselung für eine Warteschlange mithilfe von SQS-verwalteten Verschlüsselungsschlüsseln und Konfiguration der serverseitigen Verschlüsselung für eine Warteschlange mithilfe der Amazon SQS SQS-Konsole.

   > **Note**
   >
   > Wenn SSE aktiviert ist, werden anonyme `SendMessage`- und `ReceiveMessage`-Anfragen an die verschlüsselte Warteschlange abgewiesen. Die bewährten Sicherheitsmethoden von Amazon SQS raten davon ab, anonyme Anfragen zu verwenden. Wenn Sie anonyme Anfragen an eine Amazon-SQS-Warteschlange senden möchten, stellen Sie sicher, dass SSE deaktiviert ist.

9. (Optional) Um eine Warteschlange für unzustellbare Nachrichten für den Empfang von unzustellbaren Nachrichten zu konfigurieren, erweitern Sie Warteschlange für unzustellbare Nachrichten.
10. (Optional) Erweitern Sie Tags, um der Warteschlange Tags hinzuzufügen.

11. Wählen Sie Create queue (Warteschlange erstellen) aus. Amazon SQS erstellt die Warteschlange und zeigt die Seite Details der Warteschlange an.

Amazon SQS verbreitet Informationen über die neue Warteschlange im gesamten System. Da es sich bei Amazon SQS um ein verteiltes System handelt, kann es zu einer leichten Verzögerung kommen, bevor die Konsole die Warteschlange auf der Warteschlangenseite anzeigt.

Senden einer Nachricht

Nachdem Sie Ihre Warteschlange erstellt haben, können Sie eine Nachricht an sie senden.

1. Wählen Sie im linken Navigationsbereich Wartschlangen aus. Wählen Sie in der Warteschlangenliste die Warteschlange aus, die Sie erstellt haben.
2. Wählen Sie unter Aktionen die Option Nachrichten senden und empfangen.

   In der Konsole wird die Seite Nachrichten senden und empfangen angezeigt.
3. Geben Sie unter Nachricht den Nachrichtentext ein.
4. Für eine Standard-Warteschlange können Sie einen Wert für die Lieferverzögerung eingeben und die Einheiten auswählen. Geben Sie beispielsweise `60` ein und wählen Sie Sekunden. Weitere Informationen finden Sie unter Amazon-SQS-Nachrichten-Timer.
5. Klicken Sie auf Send Message (Nachricht senden).

   Wenn Ihre Nachricht gesendet wurde, zeigt die Konsole eine Erfolgsmeldung an. Wählen Sie Details anzeigen, um Informationen zur gesendeten Nachricht anzuzeigen.

Erstellen einer Amazon-SQS-FIFO-Warteschlange und Senden einer Nachricht

So erstellen Sie eine FIFO-Warteschlange für Amazon SQS:

Erstellen einer Warteschlange

Sie können mit der Amazon-SQS-Konsole FIFO-Warteschlangen erstellen. Die Konsole bietet Standardwerte für alle Einstellungen mit Ausnahme des Warteschlangennamens.

> **Important**
>
> Am 17. August 2022 wurde die serverseitige Verschlüsselung (SSE) standardmäßig auf alle Amazon-SQS-Warteschlangen angewendet.
> Fügen Sie keine persönlich identifizierbare Informationen (PII) oder andere vertrauliche oder sensible Informationen in Warteschlangennamen hinzu. Warteschlangennamen sind für viele Amazon Web Services zugänglich, einschließlich Abrechnung und CloudWatch Protokollen. Warteschlangennamen sind nicht für private oder sensible Daten gedacht.

So erstellen Sie eine Amazon-SQS-FIFO-Warteschlange

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie Create queue (Warteschlange erstellen) aus.
3. Für Typ ist standardmäßig der Standard-Warteschlangentyp festgelegt. Um eine FIFO-Warteschlange zu erstellen, wählen Sie FIFO.

   > **Note**
   >
   > Sie können den Warteschlangentyp nicht ändern, nachdem Sie die Warteschlange erstellt haben.

4. Geben Sie einen Namen für die Warteschlange ein.

   Der Name einer FIFO-Warteschlange muss mit dem Suffix `.fifo` enden. Das Suffix wird auf das Kontingent für Warteschlangennamen mit 80 Zeichen angerechnet. Um festzustellen, ob es sich bei einer Warteschlange um eine FIFO-Warteschlange handelt, können Sie überprüfen, ob der Warteschlangenname mit dem Suffix endet.

5. (Optional) Die Konsole legt Standardwerte für die Konfigurationsparameter der Warteschlange fest. Unter Konfiguration können Sie neue Werte für die folgenden Parameter festlegen:

   a. Geben Sie für Sichtbarkeitszeitbeschränkung die Dauer und die Einheiten ein. Der Bereich liegt zwischen 0 und 12 Stunden. Der Standardwert ist 30 Sekunden.

   b. Geben Sie unter Aufbewahrungszeitraum für Nachrichten die Dauer und die Einheiten ein. Der Bereich liegt zwischen 1 Minute und 14 Tagen. Der Standardwert ist 4 Tage.

   c. Geben Sie für Zustellungsverzögerung die Dauer und die Einheiten ein. Der Bereich liegt zwischen 0 Sekunden und 15 Minuten. Der Standardwert ist 0 Sekunden.

d. Geben Sie für Maximale Nachrichtengröße einen Wert ein. Der Bereich reicht von 1 KB bis 256 KB. Der Standardwert ist 256 KB.

e. Geben Sie für Wartezeit für den Empfang von Nachrichten einen Wert ein. Der Bereich liegt zwischen 0 und 20 Sekunden. Der Standardwert ist 0 Sekunden, der kurze Abfragen festlegt. Jeder Wert ungleich Null führt zu einer langen Abfrage.

f. Wählen Sie für eine FIFO-Warteschlange Inhaltsbasierte Deduplizierung, um die inhaltsbasierte Deduplizierung zu aktivieren. Sie Standardeinstellung ist deaktiviert.

g. (Optional) Damit eine FIFO-Warteschlange einen höheren Durchsatz für das Senden und Empfangen von Nachrichten in der Warteschlange ermöglicht, wählen Sie FIFO mit hohem Durchsatz aktivieren.

   Wenn Sie diese Option wählen, werden die zugehörigen Optionen (Deduplizierungsbereich und FIFO-Durchsatz-Limit) auf die erforderlichen Einstellungen geändert, um einen hohen Durchsatz für FIFO-Warteschlangen zu aktivieren. Wenn Sie Einstellungen ändern, die für die Verwendung von FIFO mit hohem Durchsatz erforderlich sind, ist der normale Durchsatz für die Warteschlange wirksam und die Deduplizierung erfolgt wie angegeben. Weitere Informationen finden Sie unter Hoher Durchsatz für FIFO-Warteschlangen in Amazon SQS und Amazon SQS SQS-Nachrichtenkontingente.

6. (Optional) Definieren Sie eine Zugriffsrichtlinie. Die Zugriffsrichtlinie definiert die Konten, Benutzer und Rollen, die auf die Warteschlange zugreifen können. Die Zugriffsrichtlinie definiert auch die Aktionen (wie `SendMessage`, `ReceiveMessage` oder `DeleteMessage`), auf die die Benutzer zugreifen können. Die Standardrichtlinie erlaubt nur dem Eigentümer der Warteschlange, Nachrichten zu senden und zu empfangen.

   Führen Sie zum Definieren der Zugriffsrichtlinie einen der folgenden Schritte aus:

   - Wählen Sie Einfach, um zu konfigurieren, wer Nachrichten an die Warteschlange senden und wer Nachrichten aus der Warteschlange empfangen kann. Die Konsole erstellt die Richtlinie auf der Grundlage Ihrer Auswahl und zeigt die resultierende Zugriffsrichtlinie im schreibgeschützten JSON-Bereich an.
   - Wählen Sie Erweitert, um die JSON-Zugriffsrichtlinie direkt zu ändern. Auf diese Weise können Sie einen benutzerdefinierten Satz von Aktionen angeben, die jeder Prinzipal (Konto, Benutzer oder Rolle) ausführen kann.

7. Wählen Sie für die Redrive-Zulassungsrichtlinie die Option Aktiviert aus. Wählen Sie eine der folgenden Optionen aus: Alle zulassen, Nach Warteschlange oder Alle verweigern. Wenn Sie

Nach Warteschlange wählen, geben Sie eine Liste mit bis zu 10 Quellwarteschlangen nach dem Amazon-Ressourcennamen (ARN) an.

8. Amazon SQS bietet standardmäßig verwaltete serverseitige Verschlüsselung. Um einen Verschlüsselungsschlüsseltyp auszuwählen oder die von Amazon SQS verwaltete serverseitige Verschlüsselung zu deaktivieren, erweitern Sie Verschlüsselung. Weitere Informationen zu Verschlüsselungsschlüsseltypen finden Sie unter Konfiguration der serverseitigen Verschlüsselung für eine Warteschlange mithilfe von SQS-verwalteten Verschlüsselungsschlüsseln und Konfiguration der serverseitigen Verschlüsselung für eine Warteschlange mithilfe der Amazon SQS SQS-Konsole.

   > **Note**
   >
   > Wenn SSE aktiviert ist, werden anonyme `SendMessage`- und `ReceiveMessage`-Anfragen an die verschlüsselte Warteschlange abgewiesen. Die bewährten Sicherheitsmethoden von Amazon SQS raten davon ab, anonyme Anfragen zu verwenden. Wenn Sie anonyme Anfragen an eine Amazon-SQS-Warteschlange senden möchten, stellen Sie sicher, dass SSE deaktiviert ist.

9. (Optional) Um eine Warteschlange für unzustellbare Nachrichten für den Empfang von unzustellbaren Nachrichten zu konfigurieren, erweitern Sie Warteschlange für unzustellbare Nachrichten.
10. (Optional) Erweitern Sie Tags, um der Warteschlange Tags hinzuzufügen.
11. Wählen Sie Create queue (Warteschlange erstellen) aus. Amazon SQS erstellt die Warteschlange und zeigt die Seite Details der Warteschlange an.

Amazon SQS verbreitet Informationen über die neue Warteschlange im gesamten System. Da es sich bei Amazon SQS um ein verteiltes System handelt, kann es zu einer leichten Verzögerung kommen, bevor die Konsole die Warteschlange auf der Warteschlangenseite anzeigt.

Nachdem Sie eine Warteschlange erstellt haben, können Sie Nachrichten an sie senden und Nachrichten empfangen und löschen. Sie können auch alle Einstellungen der Warteschlangenkonfiguration bearbeiten, mit Ausnahme des Warteschlangentyps.

Senden einer Nachricht

Nachdem Sie Ihre Warteschlange erstellt haben, können Sie eine Nachricht an sie senden.

1. Wählen Sie im linken Navigationsbereich Wartschlangen aus. Wählen Sie in der Warteschlangenliste die Warteschlange aus, die Sie erstellt haben.
2. Wählen Sie unter Aktionen die Option Nachrichten senden und empfangen.

   In der Konsole wird die Seite Nachrichten senden und empfangen angezeigt.
3. Geben Sie unter Nachricht den Nachrichtentext ein.
4. Geben Sie für eine First-In-First-Out (FIFO)-Warteschlange eine Nachrichtengruppen-ID ein. Weitere Informationen finden Sie unter Logik für die FIFO-Warteschlangenzustellung in Amazon SQS.
5. (Optional) Für eine FIFO-Warteschlange können Sie eine Nachrichtendeduplizierungs-ID eingeben. Wenn Sie die inhaltsbasierte Deduplizierung für die Warteschlange aktiviert haben, ist die Nachrichtendeduplizierungs-ID nicht erforderlich. Weitere Informationen finden Sie unter Logik für die FIFO-Warteschlangenzustellung in Amazon SQS.
6. FIFO-Warteschlangen unterstützen keine Timer für einzelne Nachrichten. Weitere Informationen finden Sie unter Amazon-SQS-Nachrichten-Timer.
7. Klicken Sie auf Send Message (Nachricht senden).

   Wenn Ihre Nachricht gesendet wurde, zeigt die Konsole eine Erfolgsmeldung an. Wählen Sie Details anzeigen, um Informationen zur gesendeten Nachricht anzuzeigen.

Verwalten einer Amazon-SQS-Warteschlange

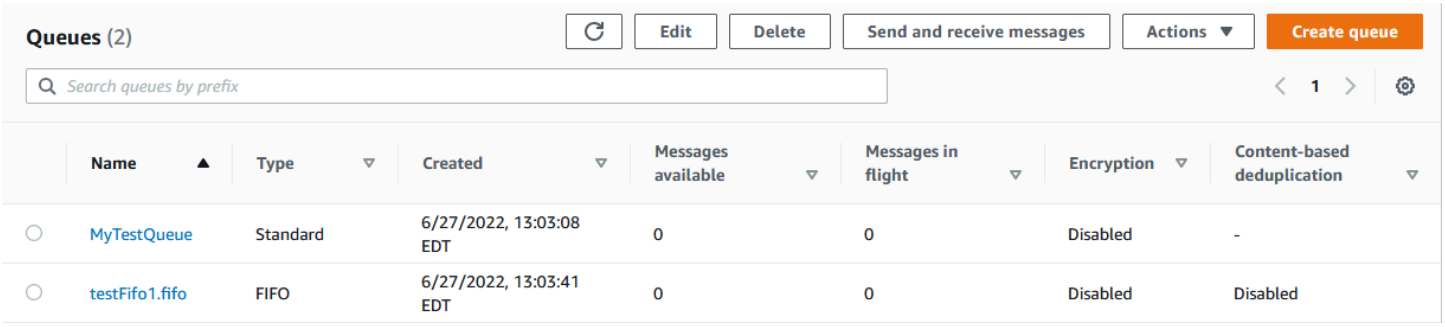
Dieser Abschnitt trägt dazu bei, dass Sie mit Amazon SQS noch vertrauter werden, indem gezeigt wird, wie Sie mithilfe der Amazon-SQS-Konsole Warteschlangen und Nachrichten verwalten.

Voraussetzungen

Bevor Sie beginnen, führen Sie die Schritte in Einrichten von Amazon SQS aus.

Die Amazon-SQS-Konsole verstehen

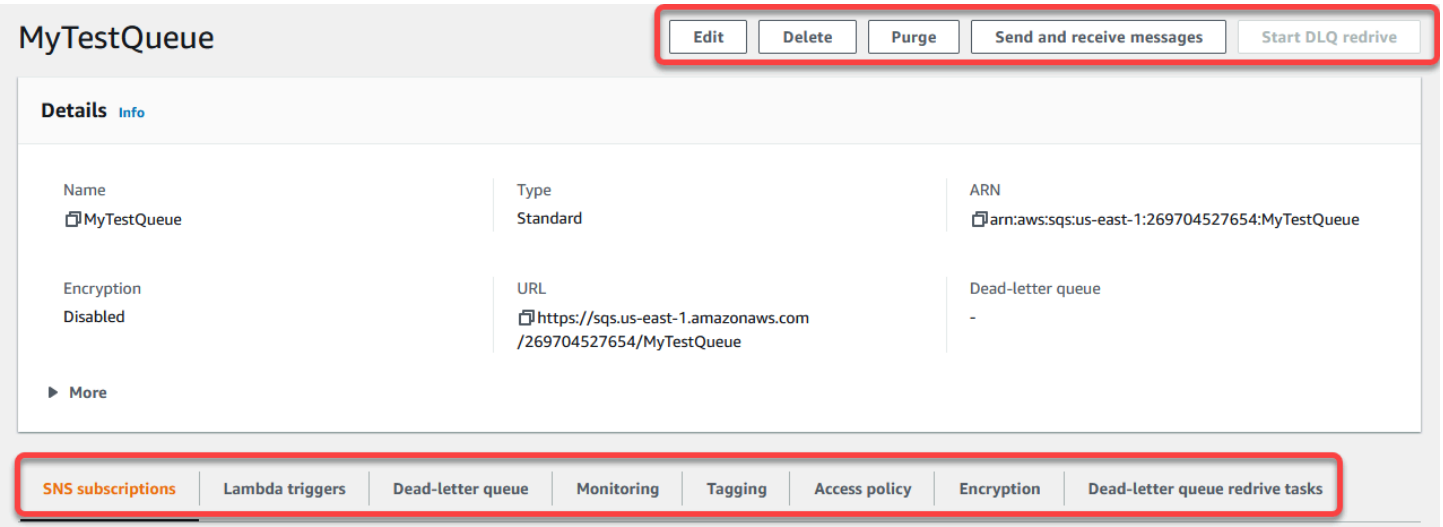
Wenn Sie die Konsole öffnen, wählen Sie im Navigationsbereich Warteschlangen aus, um die Seite Warteschlangen aufzurufen. Die Seite Warteschlangen enthält Informationen zu allen Ihren Warteschlangen in der aktiven Region.



Der Eintrag für jede Warteschlange zeigt den Warteschlangentyp und weitere Informationen zu der Warteschlange. Mithilfe der Spalte Typ können Sie auf einen Blick zwischen Standard-Warteschlangen und First-In-First-Out-Warteschlangen (FIFO) unterscheiden.

Auf der Seite Warteschlangen gibt es zwei Möglichkeiten, Aktionen für eine Warteschlange auszuführen. Sie können die Option neben dem Namen der Warteschlange auswählen und dann die Aktion auswählen, die Sie für die Warteschlange ausführen möchten.

Sie können auch den Namen der Warteschlange wählen, wodurch die Seite Details für die Warteschlange geöffnet wird. Die Seite Details enthält dieselben Aktionen wie die Seite Warteschlangen. Darüber hinaus können Sie eine der Registerkarten unter dem Abschnitt Details auswählen, um weitere Konfigurationsdetails und Aktionen anzuzeigen.



Bearbeiten einer Amazon SQS SQS-Warteschlange mit der Konsole

Sie können die Amazon-SQS-Konsole verwenden, um alle Warteschlangenkonfigurationsparameter (mit Ausnahme des Warteschlangentyps) zu bearbeiten und Warteschlangen-Features hinzuzufügen oder zu entfernen.

So bearbeiten Sie eine Amazon-SQS-Warteschlange (Konsole)

1. Öffnen Sie die Seite Warteschlangen der Amazon-SQS-Konsole.
2. Wählen Sie eine Warteschlange und dann Bearbeiten aus.
3. (Optional) Aktualisieren Sie unter Konfiguration die Konfigurationsparameter der Warteschlange.
4. (Optional) Um die Zugriffsrichtlinie zu aktualisieren, ändern Sie unter Zugriffsrichtlinie die JSON-Richtlinie.
5. (Optional) Erweitern Sie zur Aktualisierung einer Redrive-Zulassungsrichtlinie für eine Warteschlange für unzustellbare Nachrichten die Option Redrive-Zulassungsrichtlinie.
6. (Optional) Erweitern Sie zum Aktualisieren oder Entfernen der Verschlüsselung die Option Verschlüsselung.
7. (Optional) Um eine Warteschlange für unzustellbare Nachrichten hinzuzufügen, zu aktualisieren oder zu entfernen (wodurch Sie unzustellbare Nachrichten empfangen können), erweitern Sie die Option Warteschlange für unzustellbare Nachrichten.
8. (Optional) Um die Tags für die Warteschlange hinzuzufügen, zu aktualisieren oder zu entfernen, erweitern Sie Tags.

9. Wählen Sie Speichern.

   In der Konsole wird die Seite Details für die Warteschlange angezeigt.

Empfangen und Löschen einer Nachricht in Amazon SQS

Nachdem Sie Nachrichten an eine Amazon SQS SQS-Warteschlange gesendet haben, haben Sie die Möglichkeit, sie zu empfangen und zu löschen. Wenn Sie Nachrichten aus einer Warteschlange anfordern, können Sie keine einzelnen Nachrichten angeben. Stattdessen legen Sie die maximale Anzahl von Nachrichten fest, die Sie abrufen möchten, bis zu einer Obergrenze von 10.

Amazon SQS arbeitet als verteiltes System, was gelegentlich zu einer leeren Antwort führen kann, wenn Nachrichten aus einer Warteschlange mit wenigen Nachrichten abgerufen werden. In diesem Fall führen Sie Ihre Anfrage einfach erneut aus. Um den Nachrichtenabruf zu optimieren und leere Antworten zu minimieren, sollten Sie die Verwendung langer Abfragen in Betracht ziehen. Bei langen Abfragen wird die Antwort verzögert, bis eine Nachricht verfügbar ist oder die Umfrage eine Zeitüberschreitung hat, wodurch unnötige Abfragekosten reduziert und die Effizienz verbessert wird.

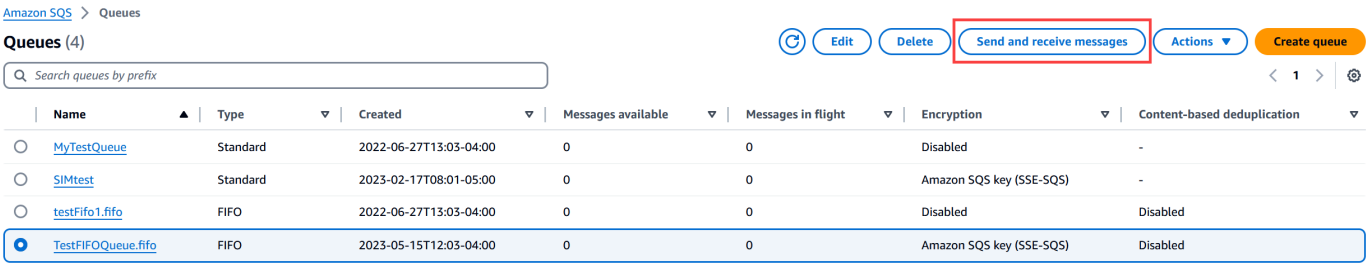
Nachrichten werden nach dem Abrufen nicht automatisch gelöscht, da Amazon SQS sicherstellt, dass Sie aufgrund von Verarbeitungsfehlern, wie z. B. Problemen mit Ihrer Anwendung oder Netzwerkunterbrechungen, nicht den Zugriff auf eine Nachricht verlieren. Um eine Nachricht dauerhaft aus der Warteschlange zu entfernen, müssen Sie nach der Verarbeitung der Nachricht explizit eine Löschanfrage senden, um den erfolgreichen Empfang und die erfolgreiche Bearbeitung zu bestätigen.

Wenn Nachrichten über die Amazon SQS SQS-Konsole abgerufen werden, werden sie sofort wieder sichtbar gemacht, sodass sie erneut abgerufen werden können. Dieses Standardverhalten stellt sicher, dass Nachrichten bei manuellen Vorgängen nicht versehentlich verloren gehen, sondern zu wiederholter Verarbeitung führen können. Passen Sie in automatisierten Umgebungen die Einstellung für das Sichtbarkeits-Timeout an, um zu steuern, wie lange eine Nachricht nach dem Abrufen für andere Benutzer unsichtbar bleibt. Diese Einstellung ist entscheidend, um die Nachrichtenverarbeitung für mehrere Verbraucher zu koordinieren und sicherzustellen, dass Nachrichten nur einmal verarbeitet werden.

Ausführlichere Informationen zum Empfangen und Löschen von Nachrichten finden Sie im Amazon SQS API-Referenzhandbuch. Dieses Handbuch bietet umfassende Informationen zu API-Endpunkten, einschließlich Parametern, mit denen komplexe Nachrichtenverarbeitungsszenarien effektiv verwaltet werden können.

Um eine Nachricht über die Konsole zu empfangen und zu löschen

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie auf der Seite Warteschlangen eine Warteschlange aus und klicken Sie dann auf Nachrichten senden und empfangen.



4. Wählen Sie auf der Seite Nachrichten senden und empfangen die Option Umfrage für Nachrichten aus.

   Amazon SQS beginnt mit der Abfrage von Nachrichten in der Warteschlange. Die Fortschrittsleiste auf der rechten Seite des Abschnitts Nachrichten empfangen zeigt die Dauer der Abfrage an.

   Im Bereich Nachrichten wird eine Liste der empfangenen Nachrichten angezeigt. Für jede Nachricht werden in der Liste die Nachrichten-ID, das Sendedatum, die Größe und die Anzahl der empfangenen Nachrichten angezeigt.
5. Um Nachrichten zu löschen, wählen Sie die Nachrichten aus, die Sie löschen möchten, und wählen Sie dann Löschen.
6. Wählen Sie im Dialogfeld „Nachrichten löschen“ die Option „Löschen“.

Bestätigen, dass eine Amazon SQS SQS-Warteschlange leer ist

In den meisten Fällen können Sie lange Abfragen verwenden, um festzustellen, ob eine Warteschlange leer ist. In seltenen Fällen erhalten Sie möglicherweise leere Antworten, auch wenn eine Warteschlange noch Nachrichten enthält. Dies gilt insbesondere, wenn Sie bei der Erstellung der Warteschlange einen niedrigen Wert für die Wartezeit für den Empfang von Nachrichten angegeben haben. In diesem Abschnitt wird beschrieben, wie Sie prüfen können, ob eine Warteschlange leer ist.

So stellen Sie fest, ob eine Warteschlange leer ist (Konsole)

1. Hindern Sie alle Produzenten daran, Nachrichten zu senden.

2. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
3. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
4. Wählen Sie auf der Seite Warteschlangen eine Warteschlange aus.
5. Wählen Sie die Registerkarte Überwachung.
6. Wählen Sie oben rechts in den Monitoring-Dashboards den Abwärtspfeil neben dem Aktualisierungssymbol aus. Wählen Sie im Dropdown-Menü Automatische Aktualisierung aus. Belassen Sie das Aktualisierungsintervall bei 1 Minute.
7. Beachten Sie die folgenden Dashboards:

   - Ungefähre Anzahl verzögerter Nachrichten
   - Ungefähre Anzahl nicht sichtbarer Nachrichten
   - Ungefähre Anzahl sichtbarer Nachrichten

   Wenn alle für mehrere Minuten `0`-Werte anzeigen, ist die Warteschlange leer.

Um zu bestätigen, dass eine Warteschlange leer ist (AWS CLI, AWS API)

1. Hindern Sie alle Produzenten daran, Nachrichten zu senden.
2. Führen Sie wiederholt einen der folgenden Befehle aus:

   - AWS CLI: `get-queue-attributes`
   - AWS API: `GetQueueAttributes`
3. Beachten Sie die Metriken für die folgenden Attribute:

   - `ApproximateNumberOfMessagesDelayed`
   - `ApproximateNumberOfMessagesNotVisible`
   - `ApproximateNumberOfMessagesVisible`

   Wenn alle für mehrere Minuten `0` anzeigen, ist die Warteschlange leer.

Wenn Sie sich auf CloudWatch Amazon-Metriken verlassen, stellen Sie sicher, dass Sie mehrere aufeinanderfolgende Nulldatenpunkte sehen, bevor Sie die Warteschlange als leer betrachten. Weitere Informationen zu CloudWatch Metriken finden Sie unterVerfügbare CloudWatch Metriken für Amazon SQS.

Löschen einer Amazon SQS SQS-Warteschlange

Wenn Sie eine Amazon-SQS-Warteschlange nicht mehr verwenden und nicht damit rechnen, sie in naher Zukunft zu verwenden, empfehlen wir, sie zu löschen.

> **Tip**
>
> Vergewissern Sie sich vor dem Löschen einer Warteschlange, dass sie leer ist, siehe hierzu Bestätigen, dass eine Amazon SQS SQS-Warteschlange leer ist.

Sie können eine Warteschlange auch dann löschen, wenn sie nicht leer ist. Wenn Sie Nachrichten in einer Warteschlange löschen möchten, aber nicht die Warteschlange selbst, können Sie die Warteschlange leeren.

So löschen Sie eine Warteschlange (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie auf der Seite Warteschlange die zu löschende Warteschlange aus.
4. Wählen Sie Löschen aus.
5. Bestätigen Sie im Dialogfeld Warteschlange löschen durch Eingabe von **`delete`**, dass die Warteschlange gelöscht werden soll.
6. Wählen Sie Löschen aus.

Um eine Warteschlange (AWS CLI und eine API) zu löschen

Sie können einen der folgenden Befehle verwenden, um eine Warteschlange zu löschen:

- AWS CLI: `aws sqs delete-queue`
- AWS API: `DeleteQueue`

Löschen von Nachrichten aus einer Warteschlange mithilfe der Amazon SQS SQS-Konsole

Wenn Sie eine Amazon-SQS-Warteschlange nicht löschen möchten, aber alle darin enthaltenen Nachrichten löschen müssen, können Sie die Warteschlange bereinigen. Das Löschen von Nachrichten dauert bis zu 60 Sekunden. Wir empfehlen, unabhängig von der Warteschlangengröße 60 Sekunden zu warten.

> **Important**
>
> Wenn Sie eine Warteschlange bereinigen, können die daraus gelöschten Nachrichten nicht mehr abgerufen werden.

So bereinigen Sie eine Warteschlange (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie auf der Seite Warteschlange, die zu bereinigende Warteschlange aus.
4. Wählen Sie unter Aktionen die Option Bereinigen aus.
5. Bestätigen Sie im Dialogfeld Warteschlange bereinigen die Bereinigung, indem Sie **`purge`** eingeben und Bereinigen auswählen.

   Alle Nachrichten werden aus der Warteschlange entfernt. Die Konsole zeigt ein Bestätigungsbanner an.

Allgemeine Aufgaben für die ersten Schritte mit Amazon SQS

Nachdem Sie eine Warteschlange erstellt und erfahren haben, wie Sie Nachrichten senden, empfangen und löschen, führen Sie z. B. die folgenden Schritte aus:

- Informationen zum Auslösen einer Lambda-Funktion finden Sie unter Konfiguration einer Amazon SQS SQS-Warteschlange zum Auslösen einer Funktion AWS Lambda.
- Erfahren Sie, wie Sie Warteschlangen konfigurieren, einschließlich SSE und anderer Features.
- Erfahren Sie, wie Sie eine Nachricht mit Attributen senden.
- Erfahren Sie, wie Sie eine Nachricht von einer VPC aus senden.

- Weitere Informationen über Funktionalität und Architektur von Amazon SQS finden Sie unter Amazon-SQS-Warteschlangentypen und Grundlegende Amazon-SQS-Architektur.
- Weitere Informationen über Richtlinien und Einschränkungen, die Ihnen dabei helfen, den größten Nutzen aus Amazon SQS zu ziehen, finden Sie unter Bewährte Methoden für Amazon SQS.
- Sehen Sie sich die Amazon SQS SQS-Beispiele für eines der AWS SDKs an, z. B. den AWS SDK for Java 2.x Developer Guide.
- Weitere Informationen zu Amazon SQS AWS CLI SQS-Befehlen finden Sie in der AWS CLI Befehlsreferenz.
- Weitere Informationen zu Amazon-SQS-Aktionen finden Sie in der Amazon-Simple-Queue-Service-API-Referenz.
- Erfahren Sie, wie Sie programmgesteuert mit Amazon SQS interagieren können: Lesen Sie Working with APIs und erkunden Sie das AWS Development Center:
  - Java
  - JavaScript
  - PHP
  - Python
  - Ruby
  - Windows und .NET

- Erfahren Sie mehr über Kosten- und Ressourcenkontrolle im Abschnitt Behebung von Problemen in Amazon SQS.
- Informationen zum Schutz Ihrer Daten und zum Zugriff auf diese finden Sie im Abschnitt Sicherheit.
- Weitere Informationen zum Amazon SQS SQS-Workflow finden Sie im Abschnitt Amazon SQS SQS-Zugriffskontrollprozess.

Erste Schritte mit Amazon-SQS-Standard-Warteschlangen

Amazon SQS stellt Standard als grundlegenden Warteschlangentyp zur Verfügung. Standardwarteschlangen unterstützen eine nahezu unbegrenzte Anzahl von API-Aufrufen pro Sekunde, pro API-Aktion (`SendMessage`, `ReceiveMessage`, oder `DeleteMessage`). Standardwarteschlangen unterstützen die at-least-once Nachrichtenzustellung. Gelegentlich wird jedoch (aufgrund der hochgradig verteilten Architektur, die den nahezu unbegrenzten Durchsatz ermöglicht) mehr als eine Nachrichtenkopie in der falschen Reihenfolge gesendet. Standard-Warteschlangen bieten eine Sortierung mit bester Leistung, die sicherstellt, dass Nachrichten prinzipiell in derselben Reihenfolge zugestellt werden, in der sie gesendet wurden.

Amazon SQS speichert eine Nachricht redundant in mehr als einer Availability Zone (AZ), bevor eine `SendMessage` bestätigt wird. Da Nachrichtenkopien in mehreren AZs gespeichert werden, kann kein einzelner Computer-, Netzwerk- oder AZ-Fehler dazu führen, dass Nachrichten unzugänglich sind.

Weitere Informationen zum Erstellen und Konfigurieren von Warteschlangen mithilfe der Amazon-SQS-Konsole finden Sie unter Erstellen Sie eine Warteschlange mit der Amazon SQS SQS-Konsole. Java-Beispiele finden Sie unter Beispiele für Amazon SQS Java SDK.

Sie können Standard-Warteschlangen für Nachrichten in vielen Bereichen einsetzen, sofern Ihre Anwendung Nachrichten verarbeiten kann, die mehr als einmal und nicht der Reihenfolge nach eingehen, z. B.:

- Entkoppeln von Echtzeit-Benutzeranfragen von intensiven Hintergrundaufgaben – Benutzer können Medien hochladen, während diese skaliert oder verschlüsselt werden.
- Zuweisen von Aufgaben zu mehreren Worker-Knoten – Verarbeitung einer großen Anzahl von Kreditkarten-Validierungsanfragen.
- Zusammenfassung von Nachrichten für zukünftige Verarbeitung – Mehrere Einträge können für das Hinzufügen zu einer Datenbank geplant werden.

Zu Kontingenten für Standard-Warteschlangen siehe Kontingente.

Informationen zu bewährten Methoden für die Arbeit mit Standard-Warteschlangen finden Sie unter Empfehlungen für Amazon-SQS-Standard- und FIFO-Warteschlangen.

Nachrichtenreihenfolge

In einer Standard-Warteschlange wird so weit wie möglich versucht, die Reihenfolge der Nachrichten zu bewahren. Wenn jedoch mehrere Kopien einer Nachricht gesendet werden, kann die Reihenfolge nicht garantiert werden. Wenn Ihr System eine Beibehaltung der Reihenfolge erfordert, wird empfohlen, eine FIFO (First-In-First-Out)-Warteschlange zu verwenden oder in jede Nachricht Informationen für die Reihenfolge einzufügen, so dass Sie die Nachrichten nach dem Empfang neu anordnen können.

Eine t-least-once Lieferung

Amazon SQS speichert aus Gründen der Redundanz und Hochverfügbarkeit Kopien der Nachrichten auf mehreren Servern. In seltenen Fällen kann es vorkommen, dass einer der Server, auf dem eine Nachrichtenkopie gespeichert ist, nicht verfügbar ist, wenn Sie eine Nachricht erhalten oder löschen.

In diesem Fall wird die Kopie der Nachricht auf dem Server, der nicht verfügbar ist, nicht gelöscht, und Sie erhalten diese Nachrichtenkopie möglicherweise erneut, wenn Sie Nachrichten erhalten. Konzipieren Sie Ihre Anwendungen idempotent (d. h., die mehrmalige Verarbeitung derselben Nachricht darf die Anwendung nicht nachteilig beeinflussen).

IDs von Amazon-SQS-Warteschlangen und -Nachrichten

In diesem Abschnitt werden die IDs von Standard- und FIFO-Warteschlangen beschrieben. Diese IDs helfen Ihnen beim Suchen und Bearbeiten spezifischer Warteschlangen und Nachrichten.

IDs für Amazon-SQS-Standard-Warteschlangen

Weitere Informationen zu den folgenden IDs finden Sie in der Amazon-Simple-Queue-Service-API-Referenz.

Warteschlangenname und URL

Wenn Sie eine neue Warteschlange erstellen, müssen Sie einen für Ihr AWS -Konto und Ihre Region eindeutigen Warteschlangennamen angeben. Amazon SQS weist jeder erstellten Warteschlange eine Kennung, eine sogenannte Warteschlangen-URL, zu, die den Warteschlangennamen und andere Amazon-SQS-Komponenten enthält. Wenn Sie eine Aktion für die Warteschlange ausführen möchten, müssen Sie deren Warteschlangen-URL angeben.

Nachfolgend finden Sie die Warteschlangen-URL für eine Warteschlange namens `MyQueue`, die einem Benutzer mit der AWS-Kontonummer `123456789012` gehört:

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

Sie können die URL einer Warteschlange programmgesteuert abrufen, indem Sie Ihre Warteschlangen auflisten und die Zeichenfolge analysieren, die der Kontonummer folgt. Weitere Informationen finden Sie unter `ListQueues`.

Nachrichten-ID

Jeder Nachricht wird vom System eine Nachrichten-ID zugewiesen, die Amazon SQS in der `SendMessage`-Antwort zurückgibt. Diese ID dient der Identifizierung von Nachrichten. Die maximale Länge einer Nachrichten-ID beträgt 100 Zeichen.

Empfangs-Mitteilung

Beim Empfang einer Nachricht aus einer Warteschlange erhalten Sie jedes Mal eine Empfangs-Mitteilung für diese Nachricht. Diese Mitteilung ist der Aktion des Nachrichtenempfangs zugeordnet und nicht der Nachricht selbst. Um eine Nachricht zu löschen oder ihre Sichtbarkeit zu ändern, müssen Sie die Empfangs-Mitteilung (und nicht die Nachrichten-ID) angeben. Daher müssen Sie eine Nachricht immer zunächst empfangen, bevor Sie sie löschen können. Es ist nicht möglich, eine Nachricht in die Warteschlange zu setzen und dann zurückzurufen. Die maximale Länge einer Empfangs-Mitteilung beträgt 1.024 Zeichen.

> **Important**
>
> Wenn Sie eine Nachricht mehrmals empfangen, erhalten Sie jedes Mal eine unterschiedliche Empfangs-Mitteilung. Wenn Sie die Nachricht löschen möchten, müssen Sie die zuletzt empfangene Empfangs-Mitteilung angeben, da die Nachricht sonst möglicherweise nicht gelöscht wird.

Es folgt ein Beispiel für eine Empfangs-Mitteilung (unterteilt in drei Zeilen).

```
MbZj6wDWli+JvwwJaBV+3dcjk2YW2vA3+STFFljTM8tJJg6HRG6PYSasuWXPJB+Cw
Lj1FjgXUv1uSj1gUPAWV66FU/WeR4mq2OKpEGYWbnLmpRCJVAyeMjeU5ZBdtcQ+QE
auMZc8ZRv37sIW2iJKq3M9MFx1YvV11A2x/KSbkJ0=
```

Kontingente

In der folgenden Tabelle werden die Kontingente im Zusammenhang mit Standard-Warteschlangen aufgeführt.

Kontingent	Beschreibung
Verzögerungswarteschlange	Die Standardverzögerung (Mindestverzögerung) für eine Warteschlange beträgt 0 Sekunden. Der Maximalwert beträgt 15 Minuten.
Aufgelistete Warteschlangen	1.000 Warteschlangen pro `ListQueues` -Anforderung.
Lange Wartezeit für Abfragen	Die maximale Wartezeit für lange Abfragen beträgt 20 Sekunden.
Nachrichten pro Warteschlange (Rückstand)	Die Anzahl der Nachrichten, die eine Amazon-SQS-Warteschlange speichern kann, ist unbegrenzt.
Nachrichten pro Warteschlange (während der Übertragung)	Bei den meisten Standard-Warteschlangen (abhängig vom Warteschlangenverkehr und dem Nachricht enrückstand) kann es maximal etwa 120 000 In-Flight -Nachrichten geben (die von einem Verbraucher aus einer Warteschlange empfangen, aber noch nicht aus der Warteschlange gelöscht wurden). Wenn Sie dieses Kontingent während der Verwendung von Kurzabfragen erreichen, gibt Amazon SQS die Fehlermel dung `OverLimit` zurück. Wenn Sie Langabfragen verwenden, gibt Amazon SQS keine Fehlermeldungen zurück. Um zu vermeiden, dass dieses Kontingent erreicht wird, sollten Sie Nachrichten aus der Warteschl ange löschen, nachdem sie verarbeitet wurden. Sie können auch die Anzahl der Warteschlangen erhöhen, die Sie zur Verarbeitung Ihrer Nachrichten verwenden. Um eine Kontingenterhöhung anzufordern, übermitteln Sie eine Support-Anforderung.

Kontingent	Beschreibung
Queue name (Name der Warteschlange)	Ein Warteschlangenname kann eine Länge von bis zu 80 Zeichen umfassen. Folgende Zeichen sind zulässig: alphanumerische Zeichen, Bindestriche (`-`) und Unterstriche (`_`). **Note** Bei Warteschlangennamen ist die Groß- und Kleinschreibung zu beachten (`Test-queue` und `test-queue` sind z. B. zwei verschiedene Warteschlangen).
Warteschlangen-Tag	Es wird nicht empfohlen, einer Warteschlange mehr als 50 Tags hinzuzufügen. Tagging unterstützt Unicode-Zeichen in UTF-8.
	Das Tag `Key` ist erforderlich, aber das Tag `Value` ist optional.
	Das Tag `Key` und das Tag `Value` unterscheiden zwischen Groß- und Kleinschreibung.
	Das Tag `Key` und das Tag `Value` können alphanumerischen Unicode-Zeichen in UTF-8 und Leerzeichen enthalten. Die folgenden Sonderzeichen sind zulässig: `_ . : / = + - @`
	Die Tags `Key` oder `Value` dürfen nicht das reservierte Präfix `aws:` enthalten (Sie können keine Tagschlüssel oder -werte mit diesem Präfix löschen).
	Die maximale Länge des Tags `Key` beträgt 128 Unicode-Zeichen in UTF-8. Das Tag `Key` darf nicht null oder leer sein.

Kontingent	Beschreibung
	Die maximale Länge des Tags `Value` beträgt 256 Unicode-Zeichen in UTF-8. Das Tag `Value` darf null oder leer sein.
	Tagging-Aktionen sind auf 30 TPS pro Tag begrenzt. AWS-Konto Wenn Ihre Anwendung einen höheren Durchsatz erfordert, reichen Sie eine Anfrage ein.

Erste Schritte mit FIFO-Warteschlangen in Amazon SQS

FIFO (First-In-First-Out)-Warteschlangen verfügen neben der vollständigen Funktionalität von Standard-Warteschlangen über eine erweiterte Nachrichtenübermittlung zwischen Anwendungen, die relevant ist, wenn die Reihenfolge von Vorgängen und Ereignissen entscheidend ist oder keine Duplikate vorkommen dürfen.

In folgenden Situationen können beispielsweise FIFO-Warteschlangen verwendet werden:

1. E-Commerce- oder Managementsysteme, bei denen die Reihenfolge entscheidend ist
2. Integration mit Systemen von Drittanbietern, bei denen Ereignisse der Reihe nach verarbeitet werden müssen
3. Verarbeitung von Benutzereingaben in der angegebenen Reihenfolge
4. Kommunikation und Vernetzung – Senden und Empfangen von Daten und Informationen in derselben Reihenfolge
5. Computersysteme – Sicherstellen, dass vom Benutzer eingegebene Befehle in der richtigen Reihenfolge ausgeführt werden
6. Bildungseinrichtungen – Verhindern, dass sich Studierende vor dem Erstellen eines Benutzerkontos in einen Kurs einschreiben
7. Online-Ticketsysteme – Wo Tickets nach dem Prinzip „Wer zuerst kommt, mahlt zuerst“ verteilt werden

> **Note**
>
> Darüber hinaus stellen FIFO-Warteschlangen auch eine garantierte einmalige Verarbeitung bereit, sind jedoch auf eine begrenzte Anzahl an Transaktionen pro Sekunde (TPS) beschränkt. Sie können den Amazon-SQS-Hochdurchsatzmodus mit Ihrer FIFO-Warteschlange verwenden, um Ihr Transaktions-Limit zu erhöhen. Einzelheiten zur Verwendung des Hochdurchsatzmodus finden Sie unter Hoher Durchsatz für FIFO-Warteschlangen in Amazon SQS. Weitere Informationen zu Durchsatzkontingenten finden Sie unter the section called “Nachrichtenkontingente”.

Amazon-SQS-FIFO-Warteschlangen sind in allen Regionen verfügbar, in denen Amazon SQS verfügbar ist.

Weitere Informationen zur Verwendung von FIFO-Warteschlangen bei komplexen Anordnungen finden Sie unter Lösung komplexer Sortieraufgaben mit Amazon-SQS-FIFO-Warteschlangen.

Weitere Informationen zum Erstellen und Konfigurieren von Warteschlangen mithilfe der Amazon-SQS-Konsole finden Sie unter Erstellen Sie eine Warteschlange mit der Amazon SQS SQS-Konsole. Java-Beispiele finden Sie unter Beispiele für Amazon SQS Java SDK.

Informationen zu bewährten Methoden für die Arbeit mit FIFO-Warteschlangen finden Sie unter Zusätzliche Empfehlungen für Amazon-SQS-FIFO-Warteschlangen und Empfehlungen für Amazon-SQS-Standard- und FIFO-Warteschlangen.

Logik für die FIFO-Warteschlangenzustellung in Amazon SQS

Die folgenden Konzepte können Ihnen dabei helfen, das Senden von Nachrichten an und das Empfangen von Nachrichten von FIFO besser zu verstehen.

Senden von Nachrichten

Wenn nacheinander mehrere Nachrichten mit je einer eindeutigen Nachrichtendeduplizierungs-ID an eine FIFO-Warteschlange gesendet werden, speichert Amazon SQS die Nachrichten und bestätigt die Übertragung. Jede Nachricht kann dann in genau der Reihenfolge empfangen und verarbeitet werden, in der sie übertragen wurde.

Nachrichten werden in FIFO-Warteschlangen basierend auf einer Nachrichtengruppen-ID sortiert. Wenn mehrere Hosts (oder verschiedene Threads auf dem gleichen Host) Nachrichten mit derselben Nachrichtengruppen-ID an eine FIFO-Warteschlange senden, speichert Amazon SQS die Nachrichten in der Reihenfolge, in der sie für die Verarbeitung eingehen. Um sicherzustellen, dass Amazon SQS die Reihenfolge beim Senden und Empfangen von Nachrichten bewahrt, müssen Sie dafür sorgen, dass alle Nachrichten mit einer eindeutigen Nachrichtengruppen-ID gesendet werden.

Die FIFO-Warteschlangenlogik gilt nur pro Nachrichtengruppen-ID. Jede Nachrichtengruppen-ID stellt eine eindeutig geordnete Nachrichtengruppe innerhalb einer Amazon-SQS-Warteschlange dar. Für jede Nachrichtengruppen-ID werden alle Nachrichten in einer strikten Reihenfolge gesendet und empfangen. Nachrichten mit anderen Werten der Nachrichtengruppen-ID werden jedoch möglicherweise nicht der Reihenfolge nach gesendet und empfangen. Sie müssen die Nachrichtengruppen-ID einer Nachricht zuordnen. Wenn Sie keine Nachrichtengruppen-ID angeben, schlägt die Aktion fehl. Wenn Sie eine einzelne Gruppe von sortierten Nachrichten

benötigen, stellen Sie dieselbe Nachrichtengruppen-ID für Nachrichten bereit, die an die FIFO-Warteschlange gesendet werden.

Empfangen von Nachrichten

Sie können keine Anforderung für den Empfang von Nachrichten mit einer bestimmten Nachrichtengruppen-ID senden.

Beim Empfangen von Nachrichten aus einer FIFO-Warteschlange mit mehreren Nachrichtengruppen-IDs versucht Amazon SQS zunächst, so viele Nachrichten mit derselben Nachrichtengruppen-ID wie möglich zurückzugeben. So können andere Konsumenten Nachrichten mit einer anderen Nachrichtengruppen-ID verarbeiten. Wenn Sie eine Nachricht mit einer Nachrichtengruppen-ID erhalten, werden keine weiteren Nachrichten für dieselbe Nachrichtengruppen-ID zurückgegeben, es sei denn, Sie löschen die Nachricht oder sie wird sichtbar.

> **Note**
>
> Mit dem Anforderungsparameter `MaxNumberOfMessages` der API-Aktion `ReceiveMessage` ist es möglich, bis zu 10 Nachrichten in einem einzelnen Aufruf zu empfangen. Diese Nachrichten behalten ihre FIFO-Reihenfolge bei und können dieselbe Nachrichtengruppen-ID haben. Wenn weniger als 10 Nachrichten mit derselben Nachrichtengruppen-ID verfügbar sind, erhalten Sie daher möglicherweise Nachrichten von einer anderen Nachrichtengruppen-ID im gleichen Stapel wie die 10 Nachrichten, aber nach wie vor in FIFO-Reihenfolge.

Mehrere Versuche

FIFO-Warteschlangen ermöglichen es dem Produzenten oder Verbraucher, mehrere Wiederholungen zu versuchen:

- Wenn der Produzent eine fehlgeschlagene `SendMessage`-Aktion feststellt, kann er das Senden so oft wie nötig wiederholen und dabei dieselbe Nachrichten-Deduplizierungs-ID verwenden. Unter der Annahme, dass der Produzent vor Ablauf des Deduplizierungsintervalls mindestens eine Bestätigung erhält, wirken sich mehrere Wiederholungsversuche weder auf die Reihenfolge der Nachrichten aus, noch führen sie zu Duplikaten.
- Wenn der Verbraucher eine fehlgeschlagene `ReceiveMessage`-Aktion feststellt, kann er sie so oft wie nötig wiederholen und dabei dieselbe ID für den Versuch verwenden, die Anfrage zu

empfangen. Unter der Annahme, dass der Verbraucher mindestens eine Bestätigung erhält, bevor die Sichtbarkeitszeitbeschränkung abläuft, wirken sich mehrere Wiederholungsversuche nicht auf die Reihenfolge der Nachrichten aus.

- Wenn Sie eine Nachricht mit einer Nachrichtengruppen-ID erhalten, werden keine weiteren Nachrichten für dieselbe Nachrichtengruppen-ID zurückgegeben, es sei denn, Sie löschen die Nachricht oder sie wird sichtbar.

Bestellung von FIFO-Warteschlangennachrichten in Amazon SQS

Die FIFO-Warteschlange ist eine Verbesserung und Ergänzung der Standard-Warteschlange. Die wichtigsten Features dieses Warteschlangentyps sind FIFO (First-In-First-Out)-Bereitstellung und genau einmalige Verarbeitung:

- Die Reihenfolge, in der Nachrichten gesendet und empfangen werden, wird strikt beibehalten. Eine Nachricht wird einmal zugestellt und ist erst dann verfügbar, wenn ein Verbraucher sie verarbeitet und löscht.
- Duplikate werden nicht in die Warteschlange aufgenommen.

FIFO-Warteschlangen unterstützen zudem Nachrichtengruppen, die mehrere geordnete Nachrichtengruppen innerhalb einer einzigen Warteschlange ermöglichen. Es gibt kein Kontingent für die Anzahl der Nachrichtengruppen in einer FIFO-Warteschlange.

Exactly-Once-Verarbeitung in Amazon SQS

Anders als bei Standard-Warteschlangen werden in FIFO-Warteschlangen keine Duplikate aufgenommen. Mit FIFO-Warteschlangen können Sie das Senden von Duplikaten an eine Warteschlange verhindern. Wenn Sie die Aktion `SendMessage` innerhalb des 5-minütigen Deduplizierungsintervalls erneut ausführen, stellt Amazon SQS keine Duplikate in die Warteschlange.

Um die Deduplizierung zu konfigurieren, müssen Sie eine der folgenden Aktionen ausführen:

- Aktivieren Sie inhaltsbasierte Deduplizierung. So wird Amazon SQS angewiesen, einen SHA-256-Hash zum Generieren der Nachrichtendeduplizierungs-ID zu verwenden. Dabei wird der Inhalt der Nachricht, nicht jedoch die Attribute der Nachricht verwendet. Weitere Informationen finden Sie in der Dokumentation zu den Aktionen `CreateQueue`, `GetQueueAttributes` und `SetQueueAttributes` in der Amazon-Simple-Queue-Service-API-Referenz.

- Stellen Sie die Nachrichtendeduplizierungs-ID für die Nachricht explizit bereit (oder rufen Sie die Sequenznummer ab). Weitere Informationen finden Sie in der Dokumentation zu den Aktionen `SendMessage`, `SendMessageBatch` und `ReceiveMessage` in der Amazon-Simple-Queue-Service-API-Referenz.

Von einer Standardwarteschlange zu einer FIFO-Warteschlange in Amazon SQS wechseln

Wenn Sie in einer vorhandenen Anwendung bereits Standard-Warteschlangen verwenden und von den Sortier-Features oder der garantierten einmaligen Verarbeitung von FIFO-Warteschlangen profitieren möchten, müssen Sie die Warteschlange und Ihre Anwendung korrekt konfigurieren.

> **Note**
>
> Sie können eine vorhandene Standard-Warteschlange nicht in eine FIFO-Warteschlange umwandeln. Sie müssen entweder eine neue FIFO-Warteschlange für Ihre Anwendung erstellen oder Ihre vorhandene Standard-Warteschlange löschen und als FIFO-Warteschlange neu anlegen.

Verwenden Sie die folgende Checkliste, um sicherzustellen, dass Ihre Anwendung mit einer FIFO-Warteschlange korrekt funktioniert.

- Verwenden Sie den empfohlenen Modus mit hohem Durchsatz für FIFO, um einen höheren Durchsatz zu erzielen. Weitere Informationen zu Nachrichtenkontingenten finden Sie unter Amazon SQS SQS-Nachrichtenkontingente.
- FIFO-Warteschlangen unterstützen keine Verzögerung pro Nachricht, sondern nur Verzögerungen pro Warteschlange. Wenn Ihre Anwendung für jede Nachricht denselben Wert für den Parameter `DelaySeconds` festlegt, müssen Sie die Anwendung anpassen und die Verzögerung pro Nachricht entfernen und den Parameter `DelaySeconds` stattdessen für die gesamte Warteschlange konfigurieren.
- Die Nachrichtengruppe ist ein einzigartiges FIFO-Feature, mit dem Kunden Nachrichten parallel verarbeiten und gleichzeitig deren jeweilige Anordnung beibehalten können. Kunden organisieren Nachrichten in Nachrichtengruppen, indem sie eine Nachrichtengruppen-ID angeben. Nachrichtengruppen basieren häufig auf einer Geschäftsdimension für einen bestimmten Workload. Verwenden Sie für eine bessere Skalierung mit FIFO-Warteschlangen eine detailliertere

Geschäftsdimension für die Nachrichten-ID. Je mehr Nachrichtengruppen-IDs Sie haben, an die Sie Nachrichten verteilen, umso mehr Nachrichten stellt FIFO zur Nutzung zur Verfügung.

- Überprüfen Sie vor dem Senden von Nachrichten an eine FIFO-Warteschlange Folgendes:
  - Wenn Ihre Anwendung Nachrichten mit identischem Nachrichtentext senden kann, können Sie die Anwendung dahingehend modifizieren, dass jede gesendete Nachricht eine eindeutige Nachrichtendeduplizierungs-ID erhält.
  - Wenn Ihre Anwendung Nachrichten mit eindeutigem Nachrichtentext sendet, können Sie eine inhaltsbasierte Deduplizierung einrichten.
- Sie müssen keine Codeänderungen am Konsumenten vornehmen. Wenn die Verarbeitung von Nachrichten jedoch viel Zeit in Anspruch nimmt und Sie einen hohen Wert für die Zeitbeschränkung für die Sichtbarkeit konfiguriert haben, sollten Sie zu jeder `ReceiveMessage`-Aktion eine Empfangsanforderungsversuch-ID hinzufügen. So können Sie im Fall von Netzwerkausfällen erneut eine Empfangsanforderung senden und verhindern, dass Warteschlangen aufgrund von fehlgeschlagenen Empfangsversuchen pausiert werden.

Weitere Informationen finden Sie in der API-Referenz zu Amazon Simple Queue Service.

Hoher Durchsatz für FIFO-Warteschlangen in Amazon SQS

FIFO-Warteschlangen mit hohem Durchsatz in Amazon SQS verwalten effizient einen hohen Nachrichtendurchsatz bei gleichzeitiger Einhaltung einer strikten Nachrichtenreihenfolge und gewährleisten so Zuverlässigkeit und Skalierbarkeit für Anwendungen, die zahlreiche Nachrichten verarbeiten. Diese Lösung ist ideal für Szenarien, die sowohl einen hohen Durchsatz als auch eine geordnete Nachrichtenzustellung erfordern.

FIFO-Warteschlangen mit hohem Durchsatz in Amazon SQS sind in Szenarien nicht erforderlich, in denen eine strikte Reihenfolge der Nachrichten nicht entscheidend ist und in denen das Volumen eingehender Nachrichten relativ gering oder sporadisch ist. Wenn Sie beispielsweise über eine kleine Anwendung verfügen, die seltene oder nicht sequenzielle Nachrichten verarbeitet, sind die zusätzliche Komplexität und die Kosten, die mit FIFO-Warteschlangen mit hohem Durchsatz verbunden sind, möglicherweise nicht gerechtfertigt. Wenn Ihre Anwendung die verbesserten Durchsatzfunktionen von FIFO-Warteschlangen mit hohem Durchsatz nicht benötigt, ist die Entscheidung für eine standardmäßige Amazon SQS SQS-Warteschlange möglicherweise kostengünstiger und einfacher zu verwalten.

Um die Anforderungskapazität in FIFO-Warteschlangen mit hohem Durchsatz zu erhöhen, wird empfohlen, die Anzahl der Nachrichtengruppen zu erhöhen. Weitere Informationen zu Nachrichtenkontingenten mit hohem Durchsatz finden Sie unter Amazon-SQS-Service-Quotas im Allgemeine Amazon Web Services-Referenz.

Informationen zu Quotas pro Warteschlange und Strategien zur Datenverteilung finden Sie unter und. Amazon SQS SQS-Nachrichtenkontingente Partitionen und Datenverteilung für hohen Durchsatz für SQS-FIFO-Warteschlangen

Themen

- Anwendungsfälle für hohen Durchsatz für Amazon SQS FIFO-Warteschlangen
- Partitionen und Datenverteilung für hohen Durchsatz für SQS-FIFO-Warteschlangen
- Aktivieren Sie einen hohen Durchsatz für FIFO-Warteschlangen in Amazon SQS

Anwendungsfälle für hohen Durchsatz für Amazon SQS FIFO-Warteschlangen

Die folgenden Anwendungsfälle beleuchten die vielfältigen Einsatzmöglichkeiten von FIFO-Warteschlangen mit hohem Durchsatz und verdeutlichen deren Wirksamkeit in verschiedenen Branchen und Szenarien:

1. Datenverarbeitung in Echtzeit: Anwendungen, die mit Echtzeit-Datenströmen zu tun haben, wie z. B. die Ereignisverarbeitung oder die Erfassung von Telemetriedaten, können von FIFO-Warteschlangen mit hohem Durchsatz profitieren, um den kontinuierlichen Zustrom von Nachrichten zu bewältigen und gleichzeitig deren Reihenfolge für eine genaue Analyse beizubehalten.
2. E-Commerce-Auftragsabwicklung: Auf E-Commerce-Plattformen, auf denen es entscheidend ist, die Reihenfolge der Kundentransaktionen aufrechtzuerhalten, stellen FIFO-Warteschlangen mit hohem Durchsatz sicher, dass Bestellungen auch während der Haupteinkaufssaison sequentiell und ohne Verzögerungen bearbeitet werden.
3. Finanzdienstleistungen: Finanzinstitute, die hochfrequente Handels- oder Transaktionsdaten verarbeiten, verlassen sich auf FIFO-Warteschlangen mit hohem Durchsatz, um Marktdaten und Transaktionen mit minimaler Latenz zu verarbeiten und gleichzeitig die strengen regulatorischen Anforderungen für die Bestellung von Nachrichten einzuhalten.
4. Medienstreaming: Streaming-Plattformen und Medienvertriebsdienste nutzen FIFO-Warteschlangen mit hohem Durchsatz, um die Bereitstellung von Mediendateien und Streaming-

Inhalten zu verwalten und so ein reibungsloses Wiedergabeerlebnis für Benutzer zu gewährleisten und gleichzeitig die richtige Reihenfolge der Inhaltszustellung beizubehalten.

Partitionen und Datenverteilung für hohen Durchsatz für SQS-FIFO-Warteschlangen

Amazon SQS speichert FIFO-Warteschlangendaten in Partitionen. Eine Partition ist eine Speicherzuweisung für eine Warteschlange, die automatisch über mehrere Availability Zones innerhalb einer Region repliziert wird. AWS Sie verwalten keine Partitionen. Stattdessen kümmert sich Amazon SQS um die Partitionsverwaltung.

Für FIFO-Warteschlangen ändert Amazon SQS die Anzahl der Partitionen in einer Warteschlange in den folgenden Situationen:

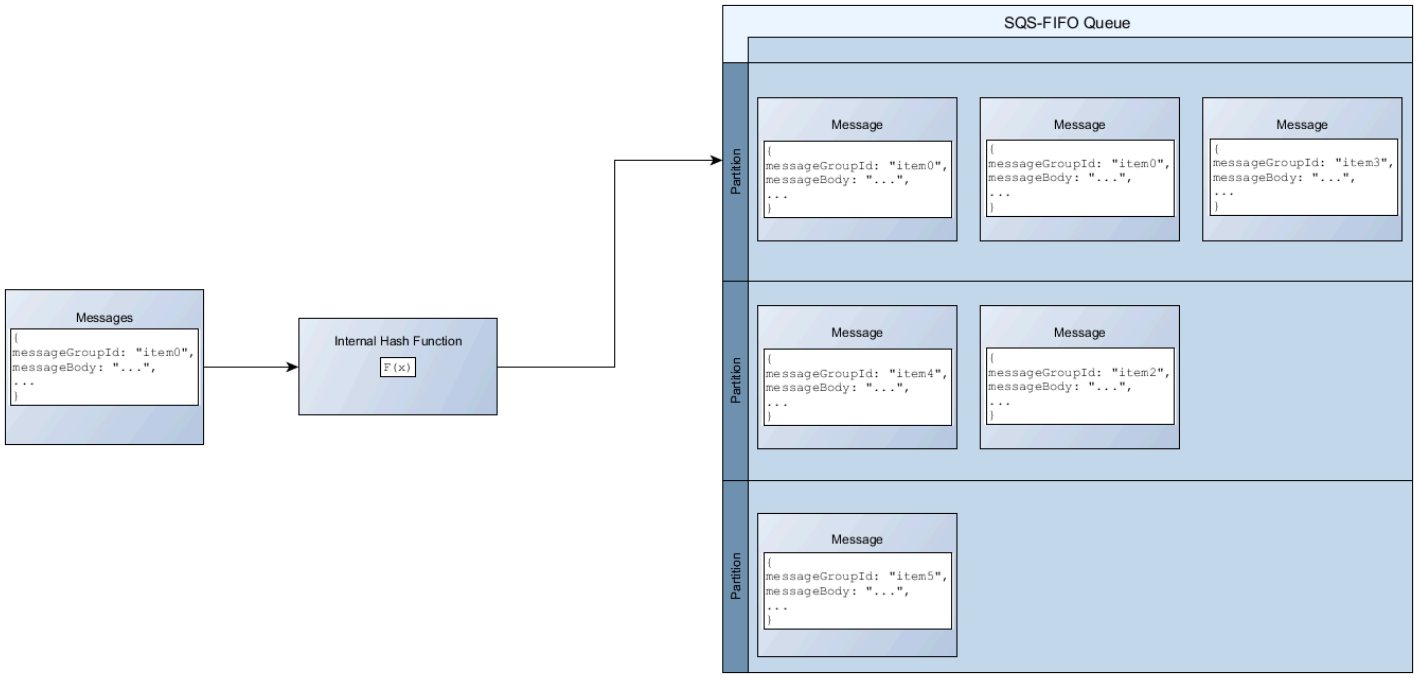
- Wenn sich die aktuelle Anforderungsrate dem, was die vorhandenen Partitionen unterstützen können, nähert oder dies übersteigt, werden zusätzliche Partitionen zugewiesen, bis die Warteschlange das regionale Kontingent erreicht hat. Informationen zu Kontingenten finden Sie unter Amazon SQS SQS-Nachrichtenkontingente.
- Wenn die aktuellen Partitionen wenig ausgelastet sind, kann die Anzahl der Partitionen reduziert werden.

Die Partitionsverwaltung wird automatisch im Hintergrund ausgeführt und ist für Ihre Anwendungen transparent. Ihre Warteschlange und Ihre Nachrichten sind jederzeit verfügbar.

Verteilen von Daten nach Nachrichtengruppen-IDs

Um eine Nachricht zu einer FIFO-Warteschlange hinzuzufügen, verwendet Amazon SQS den Wert der Nachrichtengruppen-ID jeder Nachricht als Eingabe für eine interne Hash-Funktion. Der Ausgabewert der Hash-Funktion bestimmt die Partition, in der die Nachricht gespeichert wird.

Das folgende Diagramm zeigt eine Warteschlange, die sich über mehrere Partitionen erstreckt. Die Nachrichtengruppen-ID der Warteschlange basiert auf der Elementnummer. Amazon SQS verwendet eine Hash-Funktion, um zu ermitteln, wo ein neues Element gespeichert werden soll, und zwar in diesem Fall basierend auf dem Hash-Wert der Zeichenfolge `item0`. Beachten Sie, dass die Elemente in derselben Reihenfolge gespeichert werden, in der sie der Warteschlange hinzugefügt werden. Die Position jedes einzelnen Elements wird durch den Hash-Wert seiner Nachrichtengruppen-ID bestimmt.



> **Note**
>
> Amazon SQS ist für die gleichmäßige Verteilung von Elementen auf die Partitionen einer FIFO-Warteschlange optimiert, unabhängig von der Anzahl der Partitionen. AWS empfiehlt, Nachrichtengruppen-IDs zu verwenden, die eine große Anzahl unterschiedlicher Werte haben können.

Optimierung der Partitionsnutzung

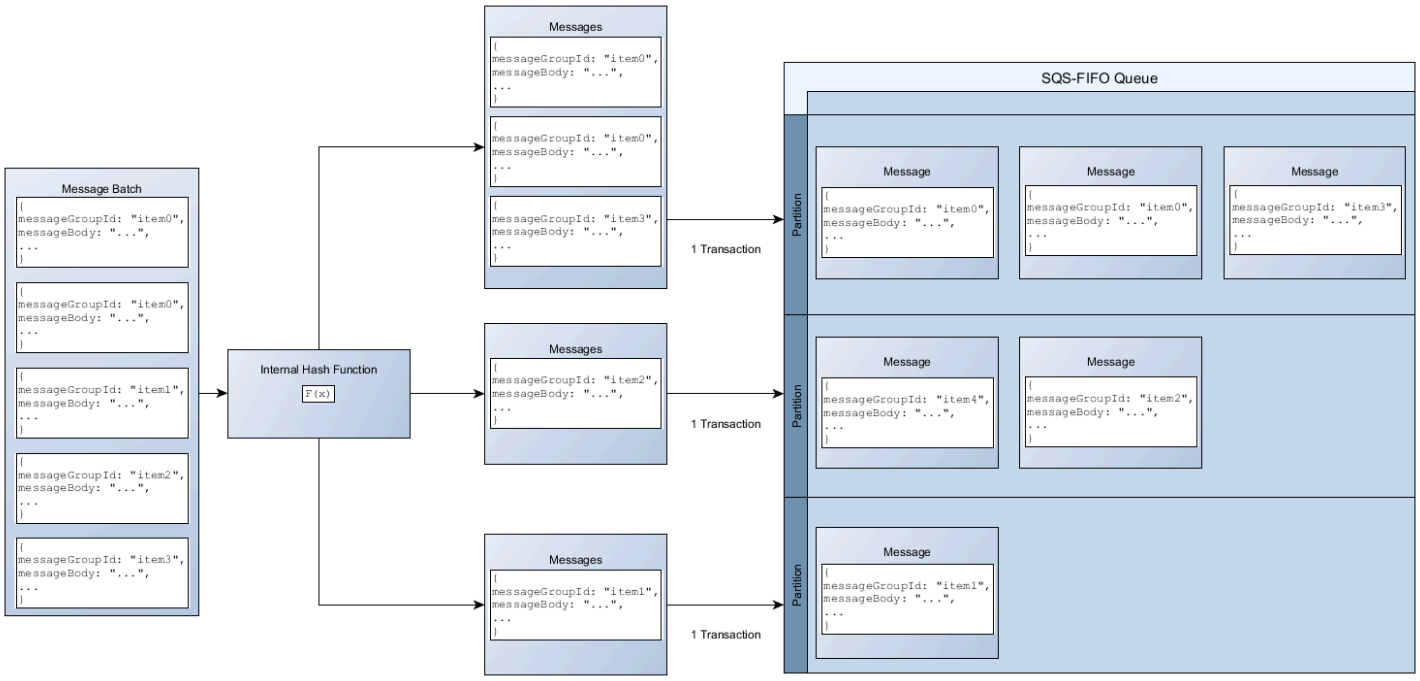
Jede Partition unterstützt bis zu 3 000 Nachrichten pro Sekunde bei Stapelverarbeitung oder bis zu 300 Nachrichten pro Sekunde bei Sende-, Empfangs- und Löschvorgängen in unterstützten Regionen. Weitere Informationen zu Nachrichtenkontingenten mit hohem Durchsatz finden Sie unter Amazon-SQS-Service-Quotas im Allgemeine Amazon Web Services-Referenz.

Bei der Verwendung von Stapel-APIs wird jede Nachricht auf der Grundlage des unter Verteilen von Daten nach Nachrichtengruppen-IDs beschriebenen Prozesses weitergeleitet. Nachrichten, die an dieselbe Partition weitergeleitet werden, werden in einer einzigen Transaktion gruppiert und verarbeitet.

Um die Partitionsnutzung für die `SendMessageBatch` API zu optimieren, AWS empfiehlt, Nachrichten nach Möglichkeit mit denselben Nachrichtengruppen-IDs zu stapeln.

Um die Partitionsnutzung für die `ChangeMessageVisibilityBatch` APIs `DeleteMessageBatch` und zu optimieren, AWS empfiehlt, `ReceiveMessage` Anfragen zu verwenden, bei denen der `MaxNumberOfMessages` Parameter auf 10 gesetzt ist, und die von einer einzelnen Anfrage zurückgegebenen Empfangs-Handles zu stapeln. `ReceiveMessage`

Im folgenden Beispiel wird ein Stapel von Nachrichten mit unterschiedlichen Nachrichtengruppen-IDs gesendet. Der Stapel ist in drei Gruppen aufgeteilt, von denen jede auf das Kontingent für die Partition angerechnet wird.



> **Note**
>
> Amazon SQS garantiert nur, dass Nachrichten mit der internen Hash-Funktion derselben Nachrichtengruppen-ID innerhalb einer Stapelanfrage gruppiert werden. Abhängig von der Ausgabe der internen Hash-Funktion und der Anzahl der Partitionen können Nachrichten mit unterschiedlichen Nachrichtengruppen-IDs gruppiert werden. Da sich die Hash-Funktion oder die Anzahl der Partitionen jederzeit ändern kann, können Nachrichten, die an einem Punkt gruppiert sind, später möglicherweise nicht gruppiert werden.

Aktivieren Sie einen hohen Durchsatz für FIFO-Warteschlangen in Amazon SQS

Sie können den hohen Durchsatz für jede neue oder vorhandene FIFO-Warteschlange aktivieren. Das Feature umfasst drei neue Optionen beim Erstellen und Bearbeiten von FIFO-Warteschlangen:

- FIFO mit hohem Durchsatz aktivieren – Macht hohen Durchsatz für Nachrichten in der Warteschlange verfügbar.
- Deduplizierungsbereich – Gibt an, ob die Deduplizierung auf Warteschlangen- oder Nachrichtengruppenebene erfolgt.
- FIFO-Durchsatz-Limit – Gibt an, ob das Durchsatzkontingent für Nachrichten in der FIFO-Warteschlange auf Warteschlangen- oder Nachrichtengruppenebene festgelegt wird.

So aktivieren Sie den hohen Durchsatz für eine FIFO-Warteschlange (Konsole)

1. Beginnen Sie mit der Erstellung oder Bearbeitung einer FIFO-Warteschlange.
2. Wenn Sie Optionen für die Warteschlange angeben, wählen Sie FIFO mit hohem Durchsatz aktivieren.

   Wenn Sie hohen Durchsatz für FIFO-Warteschlangen aktivieren, werden die entsprechenden Optionen wie folgt festgelegt:

   - Der Deduplizierungsbereich ist auf Nachrichtengruppe festgelegt. Dies ist die erforderliche Einstellung für die Verwendung eines hohen Durchsatzes für FIFO-Warteschlangen.
   - Das FIFO-Durchsatz-Limit ist auf Pro Nachrichtengruppen-ID festgelegt. Dies ist die erforderliche Einstellung für die Verwendung eines hohen Durchsatzes für FIFO-Warteschlangen.

   Wenn Sie eine der Einstellungen ändern, die für die Verwendung eines hohen Durchsatzes für FIFO-Warteschlangen erforderlich sind, ist der normale Durchsatz für die Warteschlange wirksam und die Deduplizierung erfolgt wie angegeben.
3. Geben Sie weiter alle Optionen für die Warteschlange an. Wenn Sie damit fertig sind, wählen Sie Warteschlange erstellen oder Speichern.

Nachdem Sie die FIFO-Warteschlange erstellt oder bearbeitet haben, können Sie Nachrichten an diese senden sowie Nachrichten empfangen und löschen, und dies alles mit einem höheren TPS-

Wert. Informationen zu Kontingenten mit hohem Durchsatz finden Sie unter „Nachrichtendurchsatz“ in Amazon SQS SQS-Nachrichtenkontingente.

Die wichtigsten Begriffe von Amazon SQS

Die folgenden wichtigen Begriffe vermitteln Ihnen ein besseres Verständnis der Funktionalität von FIFO-Warteschlangen. Weitere Informationen finden Sie in der API-Referenz zu Amazon Simple Queue Service.

Nachrichtendeduplizierungs-ID

Das Token, das für die Deduplizierung gesendeter Nachrichten verwendet wird. Wenn eine Nachricht mit einer bestimmten Nachrichtendeduplizierungs-ID erfolgreich gesendet wurde, werden alle Nachrichten, die mit derselben Nachrichtendeduplizierungs-ID gesendet wurden, erfolgreich akzeptiert, aber während des 5-minütigen Deduplizierungsintervalls nicht zugestellt.

> **Note**
>
> Amazon SQS verfolgt weiterhin die Nachrichtendeduplizierungs-ID, auch nachdem die Nachricht empfangen und gelöscht wurde.

Nachrichtengruppen-ID

Der Tag, der angibt, dass eine Nachricht zu einer bestimmten Nachrichtengruppe gehört. Nachrichten, die derselben Nachrichtengruppe angehören, werden immer nacheinander in einer strengen Reihenfolge in Bezug auf die Nachrichtengruppe verarbeitet (Nachrichten, die verschiedenen Nachrichtengruppen angehören, können jedoch in einer anderen Reihenfolge verarbeitet werden).

Empfangsanforderungsversuch-ID

Das Token, das für die Deduplizierung von `ReceiveMessage`-Aufrufen verwendet wird.

Sequenznummer

Die große, nicht fortlaufende Zahl, die Amazon SQS jeder Nachricht zuweist.

FIFO-Kompatibilität in Amazon SQS

Clients

Der Amazon SQS Buffered Asynchronous Client unterstützt derzeit keine FIFO-Warteschlangen.

Services

Wenn Ihre Anwendung mehrere AWS Dienste oder eine Mischung aus AWS externen Diensten verwendet, ist es wichtig zu verstehen, welche Dienstfunktionen FIFO-Warteschlangen nicht unterstützen.

Einige AWS oder externe Dienste, die Benachrichtigungen an Amazon SQS senden, sind möglicherweise nicht mit FIFO-Warteschlangen kompatibel, obwohl Sie eine FIFO-Warteschlange als Ziel festlegen können.

Die folgenden Funktionen von AWS Diensten sind derzeit nicht mit FIFO-Warteschlangen kompatibel:

- Amazon-S3-Ereignis-Benachrichtigungen
- Auto Scaling Lifecycle Hooks
- AWS IoT Regelaktionen
- AWS Lambda Warteschlangen für unzustellbare Nachrichten

Weitere Informationen zur Kompatibilität anderer Services mit FIFO-Warteschlangen finden Sie in Ihrer Service-Dokumentation.

FIFO-Warteschlangen- und Nachrichtenkennungen in Amazon SQS

In diesem Abschnitt werden die IDs von FIFO-Warteschlangen beschrieben. Diese IDs helfen Ihnen beim Suchen und Bearbeiten spezifischer Warteschlangen und Nachrichten.

Themen

- Identifikatoren für FIFO-Warteschlangen in Amazon SQS
- Zusätzliche Kennungen für Amazon-SQS-FIFO-Warteschlangen

Identifikatoren für FIFO-Warteschlangen in Amazon SQS

Weitere Informationen zu den folgenden IDs finden Sie in der Amazon-Simple-Queue-Service-API-Referenz.

Warteschlangenname und URL

Wenn Sie eine neue Warteschlange erstellen, müssen Sie einen für Ihr AWS -Konto und Ihre Region eindeutigen Warteschlangennamen angeben. Amazon SQS weist jeder erstellten Warteschlange eine Kennung, eine sogenannte Warteschlangen-URL, zu, die den Warteschlangennamen und andere Amazon-SQS-Komponenten enthält. Wenn Sie eine Aktion für die Warteschlange ausführen möchten, müssen Sie deren Warteschlangen-URL angeben.

Der Name einer FIFO-Warteschlange muss mit dem Suffix `.fifo` enden. Das Suffix wird auf das Kontingent für Warteschlangennamen mit 80 Zeichen angerechnet. Um festzustellen, ob es sich bei einer Warteschlange um eine FIFO-Warteschlange handelt, können Sie überprüfen, ob der Warteschlangenname mit dem Suffix endet.

Im Folgenden finden Sie die Warteschlangen-URL für eine FIFO-Warteschlange, die einem Benutzer mit der AWS-Kontonummer `MyQueue` `123456789012` gehört.

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

Sie können die URL einer Warteschlange programmgesteuert abrufen, indem Sie Ihre Warteschlangen auflisten und die Zeichenfolge analysieren, die der Kontonummer folgt. Weitere Informationen finden Sie unter `ListQueues`.

Nachrichten-ID

Jeder Nachricht wird vom System eine Nachrichten-ID zugewiesen, die Amazon SQS in der `SendMessage`-Antwort zurückgibt. Diese ID dient der Identifizierung von Nachrichten. Die maximale Länge einer Nachrichten-ID beträgt 100 Zeichen.

Empfangs-Mitteilung

Beim Empfang einer Nachricht aus einer Warteschlange erhalten Sie jedes Mal eine Empfangs-Mitteilung für diese Nachricht. Diese Mitteilung ist der Aktion des Nachrichtenempfangs zugeordnet und nicht der Nachricht selbst. Um eine Nachricht zu löschen oder ihre Sichtbarkeit zu ändern, müssen Sie die Empfangs-Mitteilung (und nicht die Nachrichten-ID) angeben. Daher müssen Sie eine Nachricht immer zunächst empfangen, bevor Sie sie löschen können. Es ist nicht möglich,

eine Nachricht in die Warteschlange zu setzen und dann zurückzurufen. Die maximale Länge einer Empfangs-Mitteilung beträgt 1.024 Zeichen.

> **Important**
>
> Wenn Sie eine Nachricht mehrmals empfangen, erhalten Sie jedes Mal eine unterschiedliche Empfangs-Mitteilung. Wenn Sie die Nachricht löschen möchten, müssen Sie die zuletzt empfangene Empfangs-Mitteilung angeben, da die Nachricht sonst möglicherweise nicht gelöscht wird.

Es folgt ein Beispiel für eine Empfangs-Mitteilung (unterteilt in drei Zeilen).

```
MbZj6wDWli+JvwwJaBV+3dcjk2YW2vA3+STFFljTM8tJJg6HRG6PYSasuWXPJB+Cw
Lj1FjgXUv1uSj1gUPAWV66FU/WeR4mq2OKpEGYWbnLmpRCJVAyeMjeU5ZBdtcQ+QE
auMZc8ZRv37sIW2iJKq3M9MFx1YvV11A2x/KSbkJ0=
```

Zusätzliche Kennungen für Amazon-SQS-FIFO-Warteschlangen

Weitere Informationen zu den folgenden IDs finden Sie unter Exactly-Once-Verarbeitung in Amazon SQS und in der Amazon-Simple-Queue-Service-API-Referenz.

Nachrichtendeduplizierungs-ID

Das Token, das für die Deduplizierung gesendeter Nachrichten verwendet wird. Wenn eine Nachricht mit einer bestimmten Nachrichtendeduplizierungs-ID erfolgreich gesendet wurde, werden alle Nachrichten, die mit derselben Nachrichtendeduplizierungs-ID gesendet wurden, erfolgreich akzeptiert, aber während des 5-minütigen Deduplizierungsintervalls nicht zugestellt.

Nachrichtengruppen-ID

Der Tag, der angibt, dass eine Nachricht zu einer bestimmten Nachrichtengruppe gehört. Nachrichten, die derselben Nachrichtengruppe angehören, werden immer nacheinander in einer strengen Reihenfolge in Bezug auf die Nachrichtengruppe verarbeitet (Nachrichten, die verschiedenen Nachrichtengruppen angehören, können jedoch in einer anderen Reihenfolge verarbeitet werden).

Sequenznummer

Die große, nicht fortlaufende Zahl, die Amazon SQS jeder Nachricht zuweist.

Amazon-SQS-Kontingente

In diesem Thema werden Kontingente innerhalb von Amazon Simple Queue Service (Amazon SQS) aufgeführt.

Themen

- Amazon SQS FIFO-Warteschlangenkontingente
- Amazon SQS SQS-Nachrichtenkontingente
- Amazon SQS SQS-Richtlinienkontingente

Amazon SQS FIFO-Warteschlangenkontingente

Amazon-SQS-Kontingente

In der folgenden Tabelle werden die Kontingente im Zusammenhang mit FIFO-Warteschlangen aufgeführt.

Kontingent	Beschreibung
Verzögerungswarteschlange	Die Standardverzögerung (Mindestverzögerung) für eine Warteschlange beträgt 0 Sekunden. Der Maximalwert beträgt 15 Minuten.
Aufgelistete Warteschlangen	1.000 Warteschlangen pro `ListQueues` -Anforderung.
Lange Wartezeit für Abfragen	Die maximale Wartezeit für lange Abfragen beträgt 20 Sekunden.
Mitteilungsgruppen	Es gibt kein Kontingent für die Anzahl der Nachricht engruppen in einer FIFO-Warteschlange.
Nachrichten pro Warteschlange (Rückstand)	Die Anzahl der Nachrichten, die eine Amazon-SQS-Warteschlange speichern kann, ist unbegrenzt.
Nachrichten pro Warteschlange (während der Übertragung)	Bei FIFO-Warteschlangen können maximal 20 000 In-Flight-Nachrichten enthalten sein (die von einem Verbraucher aus einer Warteschlange empfangen, aber

Kontingent	Beschreibung
	noch nicht aus der Warteschlange gelöscht wurden). Wenn Sie dieses Kontingent erreichen, gibt Amazon SQS keine Fehlermeldungen zurück.
Queue name (Name der Warteschlange)	Der Name einer FIFO-Warteschlange muss mit dem Suffix `.fifo` enden. Das Suffix wird auf das Kontingent für Warteschlangennamen mit 80 Zeichen angerechnet. Um festzustellen, ob es sich bei einer Warteschlange um eine FIFO-Warteschlange handelt, können Sie überprüfen, ob der Warteschlangenname mit dem Suffix endet.
Warteschlangen-Tag	Es wird nicht empfohlen, einer Warteschlange mehr als 50 Tags hinzuzufügen. Tagging unterstützt Unicode-Zeichen in UTF-8.
	Das Tag `Key` ist erforderlich, aber das Tag `Value` ist optional.
	Das Tag `Key` und das Tag `Value` unterscheiden zwischen Groß- und Kleinschreibung.
	Das Tag `Key` und das Tag `Value` können alphanumerischen Unicode-Zeichen in UTF-8 und Leerzeichen enthalten. Die folgenden Sonderzeichen sind zulässig: `_ . : / = + - @`
	Die Tags `Key` oder `Value` dürfen nicht das reservierte Präfix `aws:` enthalten (Sie können keine Tagschlüssel oder -werte mit diesem Präfix löschen).
	Die maximale Länge des Tags `Key` beträgt 128 Unicode-Zeichen in UTF-8. Das Tag `Key` darf nicht null oder leer sein.
	Die maximale Länge des Tags `Value` beträgt 256 Unicode-Zeichen in UTF-8. Das Tag `Value` darf null oder leer sein.

Kontingent	Beschreibung
	Tagging-Aktionen sind auf 30 TPS pro Tag begrenzt. AWS-Konto Wenn Ihre Anwendung einen höheren Durchsatz erfordert, reichen Sie eine Anfrage ein.

Amazon SQS SQS-Nachrichtenkontingente

In der folgenden Tabelle werden die Kontingente im Zusammenhang mit Nachrichten aufgeführt.

Kontingent	Beschreibung
Mitteilungs-ID im Stapel	Eine Stapel-Nachrichten-ID kann bis zu 80 Zeichen lang sein. Folgende Zeichen sind zulässig: alphanumerische Zeichen, Bindestriche (-) und Unterstriche (_).
Nachrichtenattribute	Eine Nachricht kann bis zu 10 Metadatenattribute enthalten.
Nachrichtenstapel	Eine einzelne Nachrichtstapelanforderung kann maximal 10 Nachrichten umfassen. Weitere Informationen finden Sie unter Konfiguration des AmazonSQS-Clients BufferedAsync im Abschnitt Amazon-SQS-Stapelaktionen.
Nachrichteninhalt	Eine Nachricht kann nur XML, JSON und unformatierten Text enthalten. Die folgenden Unicode-Zeichen sind zulässig: `#x9` | `#xA` | `#xD` | `#x20` bis `#xD7FF` | `#xE000` bis `#xFFFD` | `#x10000` to `#x10FFFF` Alle Zeichen, die nicht in diese Liste enthalten sind, werden abgelehnt. Weitere Informationen finden Sie in der W3C-Spezifikation für Zeichen.
Nachrichtengruppen-ID	Verwenden Sie Nachrichten aus dem Rückstau, um das Entstehen eines großen Rückstaus an Nachrichten mit derselben Nachrichtengruppen-ID zu vermeiden.

Kontingent	Beschreibung
	`MessageGroupId` ist für FIFO-Warteschlangen erforderlich. Sie können dies nicht für Standard-Warteschlangen verwenden. Sie müssen einer Nachricht eine `MessageGroupId` zuordnen, die nicht leer ist. Wenn Sie keine `MessageGroupId` angeben, schlägt die Aktion fehl. Die maximale Länge der `MessageGroupId` ist 128 Zeichen. Gültige Werte: alphanumerische Zeichen und Satzzeichen `(!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~)` .
Nachrichtenspeicherung	Standardmäßig wird eine Nachricht 4 Tage aufbewahrt. Die Mindestdauer 60 Sekunden (1 Minute). Die Höchstdauer ist 1 209 600 Sekunden (14 Tage).
Nachrichtendurchsatz	Standardwarteschlangen unterstützen eine nahezu unbegrenzte Anzahl von API-Aufrufen pro Sekunde, pro API-Aktion (`SendMessage` , `ReceiveMessage` , oder `DeleteMessage`).
	FIFO-Warteschlangen • FIFO-Warteschlangen unterstützen ein Kontingent von 300 Transaktionen pro Sekunde und API-Aktion (`SendMessage` , `ReceiveMessage` und `DeleteMessage`). • Wenn Sie die Stapelverarbeitung verwenden, unterstützen FIFO-Warteschlangen bis zu 3 000 Transaktionen pro Sekunde, pro API-Methode (`SendMessage` , `ReceiveMessage` und `DeleteMessage`). Die 3 000 Transaktionen pro Sekunde repräsentieren 300 API-Aufrufe mit jeweils einem Stapel von 10 Nachrichten.

Kontingent	Beschreibung
	[Hoher Durchsatz für FIFO-Warteschlangen] • Ohne Stapelverarbeitung (`SendMessage` , `ReceiveMessage` und `DeleteMessage`) können FIFO-Warteschlangen mit einem hohen Durchsatz bis zu 70 000 Transaktionen pro Sekunde pro API-Aktion in den Regionen USA Ost (Nord-Virginia), USA Ost (Ohio), USA West (Oregon), Europa (Frankfurt) und Europa (Irland) verarbeiten. • Für die Regionen USA Ost (Ohio) und Europa (Frankfurt) beträgt der Standarddurchsatz 18 000 Transaktionen pro Sekunde pro API-Aktion. • Für Asien-Pazifik (Mumbai), Asien-Pazifik (Singapur), Asien-Pazifik (Sydney) und Asien-Pazifik (Tokio) beträgt der Standarddurchsatz 9 000 Transaktionen pro Sekunde pro API-Aktion. • Für Europa (London) und Südamerika (São Paulo) beträgt der Standarddurchsatz 4 500 Transaktionen pro Sekunde pro API-Aktion. • Erhöhen Sie für einen maximalen Durchsatz die Anzahl der Nachrichtengruppen-IDs, die Sie für Nachrichten verwenden, die ohne Stapelverarbeitung gesendet werden. • Sie können den Durchsatz auf bis zu 700 000 Nachrichten pro Sekunde erhöhen, indem Sie in den Regionen USA Ost (Nord-Virginia), USA West (Oregon) und Europa (Irland) Stapelverarbeitungs-APIs (`SendMessageBatch` und `DeleteMessageBatch`) verwenden. Die 700 000 Nachrichten pro Sekunde entsprechen 70 000 Transaktionen pro Sekunde mit jeweils einem Stapel von 10 Nachrichten. In den Regionen Europa (Frankfurt) und USA Ost (Ohio) können Sie mithilfe von Batching-APIs bis zu

Kontingent	Beschreibung
	180 000 Nachrichten pro Sekunde erreichen. Die 180 000 Nachrichten pro Sekunde entsprechen 18 000 Transaktionen pro Sekunde mit jeweils einem Stapel von 10 Nachrichten. Für Asien-Pazifik (Mumbai), Asien-Pazifik (Singapur), Asien-Pazifik (Sydney) und Asien-Pazifik (Tokio) können Sie mit der Stapelverarbeitung bis zu 90 000 Nachrichten pro Sekunde bewältigen. Um den maximalen Durchsatz bei der Verwendung von `SendMessageBatch` und `DeleteMessageBatch` zu erreichen, müssen alle Nachrichten in einer Stapelanforderung dieselbe Nachrichtengruppen-ID verwenden. • In den Regionen Europa (London) und Südamerika (São Paulo) können Sie mit der Stapelverarbeitung bis zu 45 000 Nachrichten pro Sekunde schaffen. Um den maximalen Durchsatz bei der Verwendung von `SendMessageBatch` und `DeleteMessageBatch` zu erreichen, müssen alle Nachrichten in einer Stapelanforderung dieselbe Nachrichtengruppen-ID verwenden. • In allen anderen AWS Regionen beträgt der maximale Durchsatz 2.400 (ohne Batching) oder 24.000 (mithilfe von Batching) Nachrichten pro Sekunde und API-Aktion. • Um eine Erhöhung des Kontingents über das Regionslimit hinaus zu beantragen, reichen Sie eine Support-Anfrage ein. • Weitere Informationen finden Sie unter Partitionen und Datenverteilung für hohen Durchsatz für SQS-FIFO-Warteschlangen.

Kontingent	Beschreibung
Nachrichten-Timer	Die Standardverzögerung (Mindestverzögerung) für eine Nachricht beträgt 0 Sekunden. Der Maximalwert beträgt 15 Minuten.
Nachrichtengröße	Die Mindestnachrichtengröße ist 1 Byte (1 Zeichen). Die maximale Größe beträgt 262 144 Byte (256 KiB). Um Nachrichten mit einer Größe von mehr als 256 KiB zu senden, können Sie die Amazon SQS Extended Client Library für Java und die Amazon SQS Extended Client Library für Python verwenden. Diese Bibliothek erlaubt das Senden einer Amazon-SQS-Nachricht, die auf eine Nachrichtennutzlast in Amazon S3 verweist. Die maximale Nutzlastgröße beträgt 2 GB. **Note** Diese erweiterte Bibliothek funktioniert nur für synchrone Clients.
Zeitbeschränkung für die Sichtbarkeit von Nachrichten	Die Standardzeitbeschränkung für die Sichtbarkeit einer Nachricht ist 30 Sekunden. Der Mindestwert beträgt 0 Sekunden. Der Höchstwert beträgt 12 Stunden.
Richtlinieninformationen	Das Höchstkontingent beträgt 8 192 Byte, 20 Anweisungen, 50 Prinzipale oder 10 Bedingungen. Weitere Informationen finden Sie unter Amazon SQS SQS-Richtlinienkontingente.

Amazon SQS SQS-Richtlinienkontingente

In der folgenden Tabelle werden die Kontingente im Zusammenhang mit Richtlinien aufgeführt.

Name	Maximum
Bytes	8,192
Bedingungen	10
Prinzipale	50
Anweisungen	20
Aktionen pro Anweisung	7

Features und Fähigkeiten von Amazon SQS

Amazon SQS bietet die folgenden grundlegenden Features und Funktionen:

Themen

- Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS
- Nachrichten-Metadaten für Amazon SQS
- Für die Verarbeitung von Amazon-SQS-Nachrichten erforderliche Ressourcen
- Auflistung der Warteschlangenpaginierung
- Amazon-SQS-Kostenzuordnungs-Tags
- Kurz- und Langabfragen in Amazon SQS
- Amazon-SQS-Zeitbeschränkung für die Sichtbarkeit
- Amazon-SQS-Verzögerungswarteschlangen
- Temporäre Amazon-SQS-Warteschlangen
- Amazon-SQS-Nachrichten-Timer
- Zugriff auf Amazon EventBridge Pipes über die Amazon SQS SQS-Konsole
- Verwalten großer Amazon SQS-Nachrichten mit der Extended Client Library und Amazon Simple Storage Service

Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS

Amazon SQS unterstützt Dead-Letter-Warteschlangen (DLQs), auf die Quellwarteschlangen für Nachrichten abzielen können, die nicht erfolgreich verarbeitet wurden. DLQs sind nützlich für das Debuggen Ihrer Anwendung, da Sie nicht verbrauchte Nachrichten isolieren können, um festzustellen, warum die Verarbeitung nicht erfolgreich war. Um eine optimale Leistung zu erzielen, empfiehlt es sich, die Quellwarteschlange und die DLQ innerhalb derselben Region zu halten. AWS-Konto Sobald sich Nachrichten in einer Warteschlange für unzustellbare Nachrichten befinden, können Sie:

- Prüfen Sie die Protokolle auf Ausnahmen, die dazu geführt haben könnten, dass Nachrichten an eine Warteschlange für unzustellbare Nachrichten gesendet wurden.

- Analysieren Sie den Inhalt von Nachrichten, die in die Warteschlange für unzustellbare Nachrichten verschoben wurden, um Anwendungsprobleme zu diagnostizieren.
- Bestimmen, ob der Konsument ausreichend Zeit zum Verarbeiten der Nachrichten hatte
- Verschieben Sie Nachrichten mithilfe von Redrive für unzustellbare Nachrichten aus der Warteschlange für unzustellbare Nachrichten.

Sie müssen zuerst eine neue Warteschlange erstellen, bevor Sie sie als Warteschlange für unzustellbare Briefe konfigurieren können. Informationen zur Konfiguration einer Warteschlange für unzustellbare Nachrichten mithilfe der Amazon-SQS-Konsole finden Sie unter Erfahren Sie, wie Sie mit der Amazon SQS SQS-Konsole eine Warteschlange für unzustellbare Briefe konfigurieren. Hilfe zu Warteschlangen für unzustellbare Nachrichten, z. B. zur Konfiguration eines Alarms für Nachrichten, die in eine Warteschleife mit unzustellbaren Nachrichten verschoben werden, finden Sie unter. Mit Amazon Alarme für Warteschlangen mit unzustellbaren Briefen erstellen CloudWatch

Verwenden von Richtlinien für Warteschlangen mit unzustellbaren Briefen

Verwenden Sie eine Redrive-Richtlinie, um die anzugeben. `maxReceiveCount` Dies gibt `maxReceiveCount` an, wie oft ein Verbraucher eine Nachricht aus einer Quellwarteschlange empfangen kann, bevor sie in eine Warteschlange für unzustellbare Nachrichten verschoben wird. Wenn für beispielsweise ein niedriger Wert wie 1 festgelegt `maxReceiveCount` ist, würde ein Fehler beim Empfang einer Nachricht dazu führen, dass die Nachricht in die Warteschlange für unzustellbare Briefe verschoben wird. Um sicherzustellen, dass Ihr System gegen Fehler resistent ist, legen Sie den Wert für `maxReceiveCount` hoch genug fest, um ausreichend Wiederholungsversuche zu ermöglichen.

Die Redrive-Zulassungsrichtlinie legt fest, welche Quellwarteschlange auf die Warteschlange für unzustellbare Nachrichten zugreifen kann. Sie können wählen, ob Sie alle Quellwarteschlangen, bestimmte Quellwarteschlangen zulassen oder allen Quellwarteschlangen die Verwendung der Warteschlange für unzustellbare Briefe verweigern möchten. Standardmäßig können alle Quellwarteschlangen die Warteschlange für unzustellbare Briefe verwenden. Wenn Sie mit dieser `byQueue` Option bestimmte Warteschlangen zulassen möchten, können Sie mit dem Amazon Resource Name (ARN) der Quellwarteschlange bis zu 10 Quellwarteschlangen angeben. Wenn Sie `denyAll` angeben, kann die Warteschlange nicht als Warteschlange für unzustellbare Nachrichten verwendet werden.

Grundlegendes zu den Aufbewahrungsfristen für Nachrichten in Warteschlangen mit unerlaubten Briefen

Bei Standard-Warteschlangen basiert der Ablauf einer Nachricht immer auf dem ursprünglichen Enqueue-Zeitstempel. Wenn eine Nachricht in eine Warteschlange für unzustellbare Nachrichten verschoben wird, bleibt der Enqueue-Zeitstempel unverändert. Die `ApproximateAgeOfOldestMessage` Metrik gibt an, wann die Nachricht in die Warteschlange für unzustellbare Briefe verschoben wurde, und nicht, wann die Nachricht ursprünglich gesendet wurde. Nehmen wir beispielsweise an, dass sich eine Nachricht einen Tag in der ursprünglichen Warteschlange befindet, bevor sie in eine Warteschlange für unzustellbare Nachrichten verschoben wird. Wenn die Aufbewahrungsfrist der Warteschlange für unzustellbare Nachrichten 4 Tage beträgt, wird die Nachricht nach 3 Tagen aus der Warteschlange für unzustellbare Nachrichten gelöscht und das `ApproximateAgeOfOldestMessage` ist 3 Tage. Es hat sich daher bewährt, die Aufbewahrungsdauer einer Warteschlange für unzustellbare Nachrichten immer so festzulegen, dass sie länger ist als die Aufbewahrungsdauer der ursprünglichen Warteschlange.

Bei FIFO-Warteschlangen wird der Enqueue-Zeitstempel zurückgesetzt, wenn die Nachricht in eine Warteschlange für unzustellbare Nachrichten verschoben wird. Die `ApproximateAgeOfOldestMessage`-Metrik gibt an, wann die Nachricht in die Warteschlange für unzustellbare Nachrichten verschoben wird. Im obigen Beispiel wird die Nachricht nach 4 Tagen aus der Warteschlange für unzustellbare Nachrichten gelöscht und das `ApproximateAgeOfOldestMessage` ist 4 Tage.

Erfahren Sie, wie Sie mit der Amazon SQS SQS-Konsole eine Warteschlange für unzustellbare Briefe konfigurieren

Bei einer Warteschlange mit unerlaubten Briefen handelt es sich um eine Warteschlange, auf die Quellwarteschlangen für Nachrichten abzielen können, die nicht erfolgreich verarbeitet wurden. Weitere Informationen finden Sie unter Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS .

Amazon SQS erstellt die Warteschlange für unzustellbare Nachrichten nicht automatisch. Sie müssen zuerst die Warteschlange erstellen, bevor Sie sie als Warteschlange für unzustellbare Nachrichten festlegen. Anweisungen zum Erstellen einer Warteschlange, die als Warteschlange für unzustellbare Nachrichten verwendet werden kann, finden Sie unter. Erstellen Sie eine Warteschlange mit der Amazon SQS SQS-Konsole

Die Dead-Letter-Warteschlange einer FIFO-Warteschlange muss ebenfalls eine FIFO-Warteschlange sein. Ebenso muss die Dead-Letter-Warteschlange einer Standardwarteschlange eine Standardwarteschlange sein.

Wenn Sie eine Warteschlange erstellen oder bearbeiten, können Sie eine Warteschlange für unzustellbare Nachrichten konfigurieren.

Konfigurieren einer Warteschlange für unzustellbare Nachrichten für eine vorhandene Warteschlange (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie eine Warteschlange aus und klicken Sie auf Bearbeiten.
4. Scrollen Sie zum Abschnitt Warteschlange für unzustellbare Nachrichten und wählen Sie Aktiviert aus.
5. Wählen Sie den Amazon-Ressourcennamen (ARN) einer vorhandenen Warteschlange für unzustellbare Nachrichten aus, die Sie dieser Quellwarteschlange zuordnen möchten.
6. Zum Konfigurieren, wie oft eine Nachricht empfangen werden kann, bevor sie an eine Warteschlange für unzustellbare Nachrichten gesendet wird, legen Sie Maximale Empfangsvorgänge auf einen Wert zwischen 1 und 1 000 fest.
7. Wenn Sie mit der Konfiguration der Warteschlange für unzustellbare Nachrichten fertig sind, wählen Sie Speichern.

   Nachdem Sie die Warteschlange gespeichert haben, zeigt die Konsole die Seite mit den Details für Ihre Warteschlange an. Auf der Seite Details werden auf der Registerkarte Warteschlange für unzustellbare Nachrichten die maximale Anzahl an Empfangsvorgängen und der ARN für die Warteschlange für unzustellbare Nachrichten in der Warteschlange für unzustellbare Nachrichten angezeigt.

Erfahren Sie, wie Sie in Amazon SQS ein Redrive für Warteschlangen mit unerlaubten Briefen konfigurieren

Sie können die Warteschleife für unzustellbare Nachrichten erneut aufrufen, um Nachrichten aus einer bestehenden Warteschlange für unzustellbare Nachrichten zu verschieben. Standardmäßig verschiebt das Redrive für Warteschlangen für unzustellbare Nachrichten Nachrichten aus einer Warteschlange für unzustellbare Nachrichten in eine Quellwarteschlange. Sie können jedoch auch

jede andere Warteschlange als Redrive-Ziel konfigurieren, wenn beide Warteschlangenarten vom gleichen Typ sind. Wenn es sich bei der Warteschlange für unzustellbare Nachrichten beispielsweise um eine FIFO-Warteschlange handelt, muss es sich bei der Redrive-Zielwarteschlange ebenfalls um eine FIFO-Warteschlange handeln. Darüber hinaus können Sie die Redrive-Geschwindigkeit konfigurieren, um die Geschwindigkeit festzulegen, mit der Amazon SQS Nachrichten verschiebt.

> **Note**
>
> Wenn eine Nachricht aus einer FIFO-Warteschlange in eine FIFO-DLQ verschoben wird, wird die Deduplizierungs-ID der ursprünglichen Nachricht durch die ID der ursprünglichen Nachricht ersetzt. Dadurch soll sichergestellt werden, dass die DLQ-Deduplizierung das Speichern von zwei unabhängigen Nachrichten, die zufällig eine gemeinsame Deduplizierungs-ID haben, nicht verhindert.

In Warteschlangen mit unzustellbaren Nachrichten werden Nachrichten in der Reihenfolge ihres Eingangs erneut angezeigt, wobei mit der ältesten Nachricht begonnen wird. Die Zielwarteschlange nimmt jedoch die Nachrichten, die erneut gesendet wurden, sowie Nachrichten von anderen Produzenten auf, je nachdem, wie in welcher Reihenfolge sie empfangen wurden. Wenn ein Producer beispielsweise Nachrichten an eine FIFO-Quell-Warteschlange sendet und gleichzeitig Nachrichten aus einer Warteschlange für unzustellbare Nachrichten empfängt, werden die weitergeleiteten Nachrichten mit den neuen Nachrichten des Produzenten verwoben.

> **Note**
>
> Die Redrive-Aufgabe setzt die Aufbewahrungsfrist zurück. Alle neu zugestellten Nachrichten werden als neue Nachrichten betrachtet `messageID` und `enqueueTime` neu weitergeleiteten Nachrichten zugewiesen.

Themen

- Konfiguration einer Redrive für Warteschlangen mit unerlaubten Briefen für eine bestehende Standardwarteschlange mithilfe der Amazon SQS SQS-API
- Mit der Amazon SQS SQS-Konsole ein Redrive für eine bestehende Standardwarteschlange konfigurieren
- Konfigurieren von Warteschlangenberechtigungen für Redrives von Warteschlangen für unzustellbare Nachrichten

Konfiguration einer Redrive für Warteschlangen mit unerlaubten Briefen für eine bestehende Standardwarteschlange mithilfe der Amazon SQS SQS-API

Mit den API-Aktionen, und können Sie eine Warteschleife für unzustellbare Nachrichten `SendMessageBatch` erneut einrichten`ReceiveMessage`: `DeleteMessageBatch`

API-Aktion	Beschreibung
`StartMessageMoveTask`	Startet eine asynchrone Aufgabe, um Nachrichten von einer angegebenen Quellwarteschlange in eine angegebene Zielwarteschlange zu verschieben.
`ListMessageMoveTasks`	Ruft die neuesten Aufgaben zum Verschieben von Nachrichten (bis zu 10) in einer bestimmten Quellwarteschlange ab.
`CancelMessageMoveTask`	Bricht eine angegebene Aufgabe zum Verschieben von Nachrichten ab. Eine Nachrichtenbewegung kann nur abgebrochen werden, wenn der aktuelle Status RUNNING ist.

Mit der Amazon SQS SQS-Konsole ein Redrive für eine bestehende Standardwarteschlange konfigurieren

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie den Namen der Warteschlange, die Sie als Warteschlange für unzustellbare Nachrichten konfiguriert haben.
4. Wählen Sie DLQ-Redrive starten.
5. Führen Sie unter Redrive-Konfiguration für Nachrichtenziel einen der folgenden Schritte aus:
   - Um Nachrichten erneut in ihre Quellwarteschlange weiterzuleiten, wählen Sie Redrive zu Quell-Warteschlange(n).

- Um Nachrichten erneut in eine andere Warteschlange weiterzuleiten, wählen Sie Redrive zu benutzerdefiniertem Ziel. Geben Sie dann den Amazon-Ressourcennamen (ARN) einer vorhandenen Zielwarteschlange an.

6. Wählen Sie unter Einstellungen für die Geschwindigkeitssteuerung eine der folgenden Optionen aus:

   - Systemoptimiert – Redrive von Nachrichten aus der Warteschlange für unzustellbare Nachrichten mit der maximalen Anzahl an Nachrichten pro Sekunde.

   - Benutzerdefinierte maximale Geschwindigkeit – Redrive von Nachrichten aus der Warteschlange für unzustellbare Nachrichten mit einer benutzerdefinierten Höchstrate an Nachrichten pro Sekunde. Die maximal zulässige Rate beträgt 500 Nachrichten pro Sekunde.
     - Es wird empfohlen, mit einem kleinen Wert für die benutzerdefinierte maximale Geschwindigkeit zu beginnen und sicherzustellen, dass die Quellwarteschlange nicht mit Nachrichten überfüllt wird. Erhöhen Sie von dort aus schrittweise den Wert für die benutzerdefinierte maximale Geschwindigkeit und überwachen Sie weiterhin den Status der Quellwarteschlange.

7. Wenn Sie mit der Konfiguration des Redrives für die Warteschlange für unzustellbare Nachrichten fertig sind, wählen Sie Nachrichten-Redrive aus.

   > **Important**
   >
   > Amazon SQS unterstützt das Filtern und Ändern von Nachrichten nicht, während sie aus der Warteschlange für unzustellbare Nachrichten zurückgeleitet werden.
   > Eine Redrive-Aufgabe für eine Warteschlange für unzustellbare Nachrichten kann maximal 36 Stunden lang ausgeführt werden. Amazon SQS unterstützt maximal 100 aktive Redrive-Aufgaben pro Konto.

8. Wenn Sie die Nachrichten-Redrive-Aufgabe abbrechen möchten, wählen Sie auf der Seite Details für Ihre Warteschlange die Option DLQ-Redrive abbrechen aus. Wenn Sie einen in Bearbeitung befindlichen Nachrichten-Redrive stornieren, verbleiben alle Nachrichten, die bereits erfolgreich in die Warteschlange für die Übertragung von Nachrichten verschoben wurden, in der Zielwarteschlange.

Konfigurieren von Warteschlangenberechtigungen für Redrives von Warteschlangen für unzustellbare Nachrichten

Sie können Benutzern Zugriff auf bestimmte Aktionen in der Warteschlange für unzustellbare Nachrichten gewähren, indem Sie Ihrer Richtlinie entsprechende Berechtigungen hinzufügen. Die Mindestberechtigungen für das Redrive einer Warteschlange für unzustellbare Nachrichten lauten wie folgt:

Mindestberechtigungen	Erforderliche API-Methoden
So starten Sie ein Nachrichten-Redrive	• Fügen Sie `sqs:StartMessageMoveTask` , `sqs:ReceiveMessage` , `sqs:DeleteMessage` und `sqs:GetQueueAttributes` der Warteschlange für unzustellbare Nachrichten hinzu. Wenn entweder die Warteschlange für unzustellbare Nachrichten oder die ursprüngliche Quellwarteschlange verschlüsselt sind (auch als SSE-Warteschlange bezeichnet), ist `kms:Decrypt` für jeden KMS-Schlüssel, der zum Verschlüsseln der Nachrichten verwendet wurde, ebenfalls erforderlich. • Fügen Sie die `sqs:SendMessage` der Zielwarteschlange hinzu. Wenn die Zielwarteschlange verschlüsselt ist, sind `kms:GenerateDataKey` und `kms:Decrypt` ebenfalls erforderlich.
So stornieren Sie einen in Bearbeitung befindlichen Nachrichten-Redrive	• Fügen Sie `sqs:CancelMessageMoveTask` , `sqs:ReceiveMessage` , `sqs:DeleteMessage` und `sqs:GetQueueAttributes` der Warteschlange für unzustellbare Nachrichten hinzu. Wenn die Warteschlange für unzustellbare Nachrichten verschlüsselt ist (auch als SSE-Warteschlange bezeichnet), ist `kms:Decrypt` ebenfalls erforderlich.
So zeigen Sie den Verschiebungsstatus einer Nachricht an	• Fügen Sie `sqs:ListMessageMoveTasks` und `sqs:GetQueueAttributes` der Warteschlange für unzustellbare Nachrichten hinzu.

So konfigurieren Sie Berechtigungen für ein unverschlüsseltes Warteschlangenpaar (eine Quellwarteschlange mit einer Warteschlange für unzustellbare Nachrichten)

Gehen Sie wie folgt vor, um Mindestberechtigungen für ein Redrive einer Warteschlange für unzustellbare Nachrichten zu konfigurieren:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Richtlinien.
3. Erstellen Sie eine Richtlinie mit den folgenden Berechtigungen und fügen Sie sie Ihrem Login-IAM-Benutzer oder der Rolle hinzu.

   - `sqs:StartMessageMoveTask`
   - `sqs:CancelMessageMoveTask`
   - `sqs:ListMessageMoveTasks`
   - `sqs:ListDeadLetterSourceQueues`
   - `sqs:ReceiveMessage`
   - `sqs:DeleteMessage`
   - `sqs:GetQueueAttributes`
   - Der `Resource`-ARN der Warteschlange für unzustellbare Nachrichten (zum Beispiel „arn:aws:sqs:*<DLQ_region>*:*<DLQ_accountId>*:*<DLQ_name>*“).
   - `sqs:SendMessage`
   - Der `Resource` ARN der Zielwarteschlange (zum Beispiel „arn:aws:sqs: *< DestQueue _region>: < _accountId>: < _name> „) DestQueue*. DestQueue
   - `kms:Decrypt` – Ermöglicht eine Entschlüsselungsaktion.
   - `kms:GenerateDataKey`
   - Der/die `Resource`-ARN(s) eines beliebigen KMS-Verschlüsselungsschlüssels, der zum Verschlüsseln der Nachrichten in der ursprünglichen Quellwarteschlange verwendet wurde (zum Beispiel „arn:aws:kms:*<region>*:*<accountId>*:key/*<keyId_used to encrypt the message body>*“).
   - Der Ressourcen-ARN des KMS-Verschlüsselungsschlüssels, der für die Redrive-Zielwarteschlange verwendet wird (zum Beispiel „arn:aws:kms:*<region>*:*<accountId>*:key/*keyId_used for the destination queue>*“).

Ihre Zugriffsrichtlinie sollte wie folgt aussehen:

```
{
  "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:<DLQ_region>:<DLQ_accountId>:<DLQ_name>"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource":
 "arn:aws:sqs:<DestQueue_region>:<DestQueue_accountId>:<DestQueue_name>"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "arn:aws:kms:<region>:<accountId>:key/<keyId>"
    }
  ]
}
```

So konfigurieren Sie Berechtigungen mithilfe eines unverschlüsselten Warteschlangenpaars (einer Quellwarteschlange mit einer Warteschlange für unzustellbare Nachrichten)

Gehen Sie wie folgt vor, um Mindestberechtigungen für eine standardmäßige unverschlüsselte Warteschlange für unzustellbare Nachrichten zu konfigurieren. Erforderliche Mindestberechtigungen

sind diejenigen zum Empfangen, Löschen und Abrufen von Attributen aus der Warteschlange für unzustellbare Nachrichten sowie für das Senden von Attributen an die Quellwarteschlange.

1. Melden Sie sich bei der an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/. AWS Management Console
2. Wählen Sie im Navigationsbereich Richtlinien.
3. Erstellen Sie eine Richtlinie mit den folgenden Berechtigungen und fügen Sie sie Ihrem Login-IAM-Benutzer oder der Rolle hinzu.

   - `sqs:StartMessageMoveTask`
   - `sqs:CancelMessageMoveTask`
   - `sqs:ListMessageMoveTasks`
   - `sqs:ListDeadLetterSourceQueues`
   - `sqs:ReceiveMessage`
   - `sqs:DeleteMessage`
   - `sqs:GetQueueAttributes`
   - Der `Resource`-ARN der Warteschlange für unzustellbare Nachrichten (zum Beispiel „arn:aws:sqs:*<DLQ_region>*:*<DLQ_accountId>*:*<DLQ_name>*“).
   - `sqs:SendMessage`
   - *Der Resource ARN der Zielwarteschlange (zum Beispiel „arn:aws:sqs: < DestQueue _region>: < _accountId>: < _name> „) DestQueue. DestQueue*

   Ihre Zugriffsrichtlinie sollte wie folgt aussehen:

```
{
"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:<DLQ_region>:<DLQ_accountId>:<DLQ_name>"
    },
    {
"Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource":
 "arn:aws:sqs:<DestQueue_region>:<DestQueue_accountId>:<DestQueue_name>"
    }
  ]
}
```

CloudTrail Aktualisierungs- und Genehmigungsanforderungen für Amazon SQS Dead-Letter Queue Redrive

Am 8. Juni 2023 führte Amazon SQS Dead-Letter Queue (DLQ) Redrive für AWS SDK und (CLI) ein. AWS Command Line Interface Diese Funktion ist eine Ergänzung zum bereits unterstützten DLQ-Redrive für die Konsole. AWS Wenn Sie die AWS Konsole bereits verwendet haben, um Nachrichten aus der Warteschleife unzustellbarer Nachrichten erneut zu versenden, sind Sie möglicherweise von den folgenden Änderungen betroffen:

- CloudTrail Umbenennung des Ereignisses für das Redrive der Warteschlange mit unerlaubten Briefen
- Aktualisierte Berechtigungen für Redrive der Warteschlange für unzustellbare Nachrichten

CloudTrail Umbenennen des Ereignisses

Am 15. Oktober 2023 ändern sich die Namen der CloudTrail Ereignisse für das Redrive von Dead-Letter-Warteschlangen auf der Amazon SQS SQS-Konsole. Wenn Sie Alarme für diese CloudTrail Ereignisse eingerichtet haben, müssen Sie sie jetzt aktualisieren. Im Folgenden sind die neuen CloudTrail Ereignisnamen für DLQ Redrive aufgeführt:

Früherer Ereignis-Name	Neuer Ereignis-Name
`CreateMoveTask`	`StartMessageMoveTask`
`CancelMoveTask`	`CancelMessageMoveTask`

Aktualisierte Berechtigungen

Amazon SQS ist in der SDK- und CLI-Version enthalten und hat außerdem die Warteschlangenberechtigungen für DLQ Redrive aktualisiert, um den bewährten Sicherheitsmethoden zu entsprechen. Verwenden Sie die folgenden Warteschlangenberechtigungstypen, um Nachrichten aus Ihren DLQs erneut weiterzuleiten.

1. Aktionsbasierte Berechtigungen (Update für die DLQ-API-Aktionen)
2. Verwaltete Amazon-SQS-Richtlinienberechtigungen
3. Berechtigungsrichtlinie, die den sqs:*-Platzhalter verwendet

> **Important**
>
> Um DLQ Redrive für SDK oder CLI verwenden zu können, benötigen Sie eine DLQ-Redrive-Berechtigungsrichtlinie, die einer der oben genannten Optionen entspricht.

Wenn Ihre Warteschlangenberechtigungen für DLQ Redrive keiner der oben genannten Optionen entsprechen, müssen Sie Ihre Berechtigungen bis zum 31. August 2023 aktualisieren. Bis zum 31. August 2023 kann Ihr Konto Nachrichten mit den Berechtigungen, die Sie in der AWS -Konsole konfiguriert haben, nur in den Regionen erneut senden, in denen Sie DLQ Redrive zuvor verwendet haben. Nehmen wir zum Beispiel an, Sie hatten „Konto A“ sowohl in us-east-1 als auch eu-west-1. „Account A“ wurde vor dem 8. Juni 2023 verwendet, um Nachrichten auf der AWS Konsole in us-east-1 erneut zu senden, aber nicht in eu-west-1. Wenn die Richtlinienberechtigungen von „Konto A“ zwischen dem 8. Juni 2023 und dem 31. August 2023 keiner der oben genannten Optionen entsprechen, können sie nur verwendet werden, um Nachrichten auf der AWS Konsole in us-east-1 und nicht in eu-west-1 erneut zuzustellen.

> **Important**
>
> Wenn Ihre DLQ-Redrive-Berechtigungen nach dem 31. August 2023 mit keiner dieser Optionen übereinstimmen, kann Ihr Konto DLQ-Nachrichten nicht mehr über die AWS -Konsole erneut senden.
>
> Wenn Sie jedoch im August 2023 die DLQ-Redrive-Funktion auf der AWS Konsole verwendet haben, haben Sie eine Verlängerung bis zum 15. Oktober 2023, um die neuen Berechtigungen gemäß einer dieser Optionen zu übernehmen.

Weitere Informationen finden Sie unter the section called “Identifizierung der betroffenen Richtlinien”.

Im Folgenden finden Sie Beispiele für Warteschlangenberechtigungen für jede DLQ-Redrive-Option. Bei der Verwendung serverseitiger verschlüsselter Warteschlangen (SSE) ist die entsprechende AWS KMS Schlüsselberechtigung erforderlich.

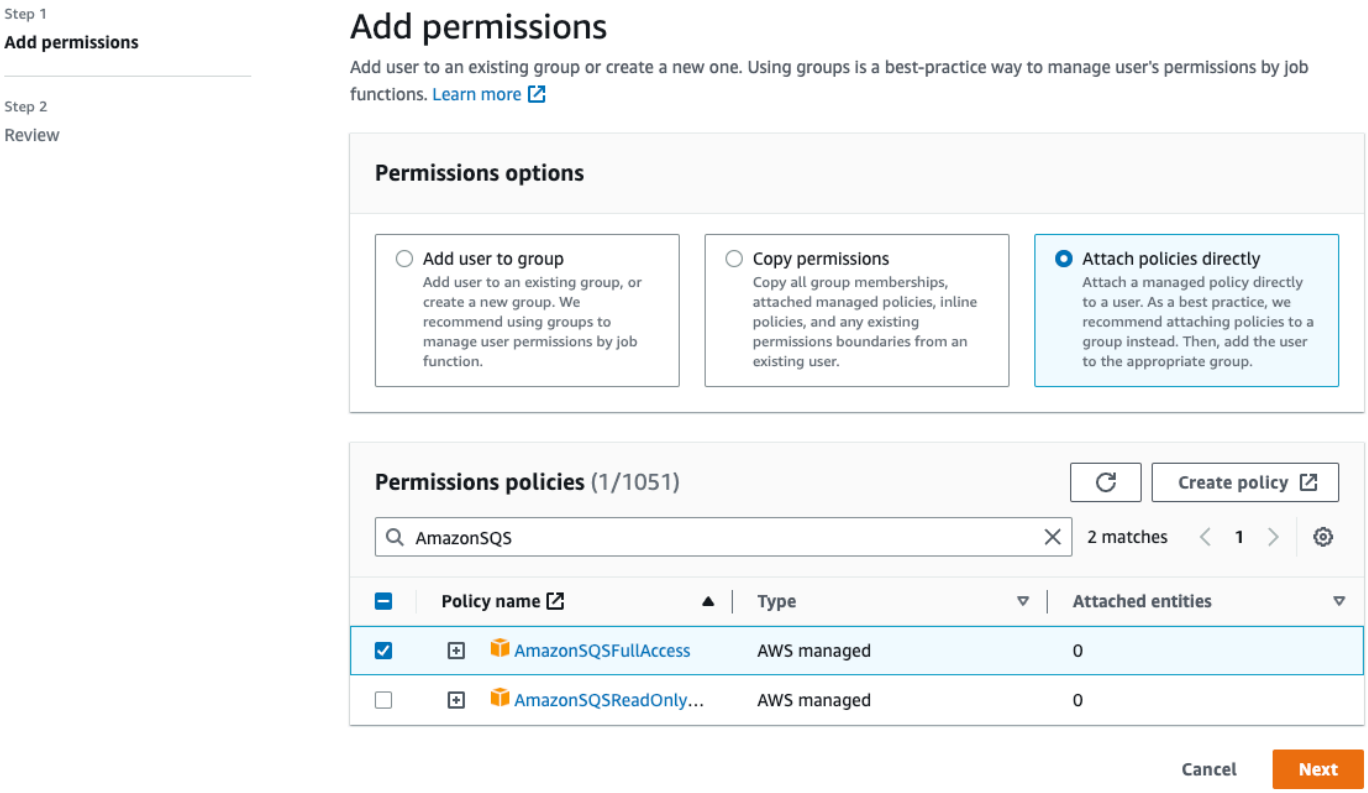
Aktionsbasiert

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:StartMessageMoveTask",
        "sqs:ListMessageMoveTasks",
        "sqs:CancelMessageMoveTask"
      ],
      "Resource": "arn:aws:sqs:<DLQ_region>:<DLQ_accountId>:<DLQ_name>"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource":
 "arn:aws:sqs:<DestQueue_region>:<DestQueue_accountId>:<DestQueue_name>"
    }
  ]
}
```

Verwaltete Richtlinie

Die folgenden verwalteten Richtlinie enthalten die erforderlichen aktualisierten Berechtigungen:

- AmazonSQS FullAccess — Beinhaltet die folgenden Aufgaben zur Wiederherstellung der Warteschleife mit unerlaubten Nachrichten: Starten, Stornieren und Auflisten.
- AmazonSQS ReadOnly Access — Bietet Nur-Lese-Zugriff und beinhaltet die Aufgabe, die Warteschlange erneut aufzulisten.



Berechtigungsrichtlinie, die den sqs*-Platzhalter verwendet

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

Identifizierung der betroffenen Richtlinien

Wenn Sie vom Kunden verwaltete Richtlinien (CMPs) verwenden, können Sie AWS CloudTrail und IAM verwenden, um die Richtlinien zu identifizieren, die von der Aktualisierung der Warteschlangenberechtigungen betroffen sind.

> **Note**
>
> Wenn Sie `AmazonSQSFullAccess` und verwenden`AmazonSQSReadOnlyAccess`, sind keine weiteren Maßnahmen erforderlich.

1. Melden Sie sich bei der Konsole an. AWS CloudTrail
2. Wählen Sie auf der Seite mit dem Ereignisverlauf unter Attribute nachschlagen im Dropdownmenü die Option Ereignisname aus. Suchen Sie dann nach `CreateMoveTask`.
3. Wählen Sie ein Ereignis, um die Seite Details zu öffnen. Rufen Sie im Abschnitt Ereignisdatensätze das `UserName` oder `RoleName` aus dem `userIdentity`-ARN ab.
4. Melden Sie sich an der IAM-Konsole an.
   - Wählen Sie für Benutzer „Benutzer“ aus. Wählen Sie den Benutzer mit dem im vorherigen Schritt identifizierten `UserName` aus.
   - Wählen Sie für Rollen „Rollen“ aus. Suchen Sie nach dem Benutzer mit dem im vorherigen Schritt angegebenen `RoleName`.
5. Überprüfen Sie auf der Seite Details im Abschnitt Berechtigungen alle Richtlinien mit dem `sqs:`-Präfix in `Action` oder überprüfen Sie Richtlinien, in denen die Amazon-SQS-Warteschlange in `Resource` definiert ist.

Mit Amazon Alarme für Warteschlangen mit unzustellbaren Briefen erstellen CloudWatch

Sie können mithilfe von Amazon und der Metrik einen Alarm für alle Nachrichten konfigurieren, die in eine Warteschlange mit unzustellbaren Briefen verschoben CloudWatch wurden. `ApproximateNumberOfMessagesVisible` Weitere Informationen finden Sie unter CloudWatch Alarme für Amazon SQS-Metriken erstellen. Nachdem Sie eine Benachrichtigung erhalten haben, dass Nachrichten an die Warteschlange für unzustellbare Briefe gesendet wurden, können Sie die Nachrichten mithilfe von Polling überprüfen, um die Nachricht zu erhalten.

Nachrichten-Metadaten für Amazon SQS

Sie können Nachrichtenattribute verwenden, um benutzerdefinierte Metadaten an Amazon-SQS-Nachrichten für Ihre Anwendungen anzufügen. Sie können Nachrichtensystemattribute verwenden, um Metadaten für andere AWS -Services wie AWS X-Ray zu speichern.

Themen

- Amazon-SQS-Nachrichtenattribute
- Attribute des Amazon-SQS-Nachrichtensystems

Amazon-SQS-Nachrichtenattribute

Sie können mit Amazon SQS strukturierte Metadaten (wie etwa Zeitstempel, geospatiale Daten, Signaturen und Kennungen) in Nachrichten einschließen, indem Sie Nachrichtenattribute verwenden. Jede Nachricht kann bis zu 10 Attribute aufweisen. Nachrichtenattribute sind optional und separat vom Nachrichtentext (werden aber mit diesem versendet). Ihr Konsument kann Nachrichtenattribute für die Verarbeitung einer Nachricht auf eine bestimmte Weise verwenden, ohne erst den Nachrichtentext verarbeiten zu müssen. Weitere Informationen über das Senden von Nachrichten mit Attributen mithilfe der Amazon-SQS-Konsole finden Sie unter Senden einer Nachricht mit Attributen.

> **Note**
>
> Verwechseln Sie Nachrichtenattribute nicht mit Nachrichtensystemattributen: Während Sie Nachrichtenattribute verwenden können, um benutzerdefinierte Metadaten an Amazon SQS SQS-Nachrichten für Ihre Anwendungen anzuhängen, können Sie Nachrichtensystemattribute verwenden, um Metadaten für andere AWS Dienste zu speichern, z. AWS X-Ray

Themen

- Nachrichtenattributkomponenten
- Datentypen für Nachrichtenattribute
- Berechnung des MD5-Nachrichtendigests für Nachrichtenattribute

Nachrichtenattributkomponenten

> **Important**
>
> Alle Komponenten eines Nachrichtenattributs sind in der Größenbeschränkung der Nachricht von 256 KB enthalten.
> `Name`, `Type`, `Value` und der Nachrichtentext dürfen nicht leer oder null sein.

Jedes Nachrichtenattribut besteht aus den folgenden Komponenten:

- Name – Der Name des Nachrichtenattributs kann die folgenden Zeichen enthalten: `A-Z`, `a-z`, `0-9`, Unterstrich (`_`), Bindestrich (`-`) und Punkt (`.`). Beachten Sie die folgenden Einschränkungen:
  - Kann bis zu 256 Zeichen lang sein.
  - Darf nicht mit `AWS.` oder `Amazon.` (oder beliebige Varianten in Groß-/Kleinschreibung) beginnen
  - Berücksichtigt Groß- und Kleinschreibung
  - Muss unter allen Attributnamen für die Nachricht eindeutig sein
  - Darf nicht mit einem Punkt beginnen oder enden
  - Darf keine Punkte in einer Sequenz enthalten
- Typ – Der Datentyp des Nachrichtenattributs. Unterstützte Typen sind u. a.: `String`, `Number` und `Binary`. Sie können auch benutzerdefinierte Informationen für alle Datentypen hinzufügen. Der Datentyp weist dieselben Einschränkungen wie der Nachrichtentext auf (weitere Informationen finden Sie unter `SendMessage` in der Amazon-Simple-Queue-Service-API-Referenz). Darüber hinaus gelten die folgenden Einschränkungen:
  - Kann bis zu 256 Zeichen lang sein.
  - Berücksichtigt Groß- und Kleinschreibung
- Value – Der Wert des Nachrichtenattributs. Für den Datentyp `String` unterliegen die Attributwerte denselben Beschränkungen wie der Nachrichtentext.

Datentypen für Nachrichtenattribute

Über den Datentyp des Nachrichtenattributs wird Amazon SQS angewiesen, wie die entsprechenden Nachrichtenattributwerte verarbeitet werden. Beispiel: Wenn der Typ `Number` ist, überprüft Amazon SQS numerische Werte.

Amazon SQS unterstützt die logischen Datentypen `String`, `Number` und `Binary`, mit optionalen benutzerdefinierten Datentypenbezeichnungen im Format *`.custom-data-type`*.

- Zeichenfolge – `String`-Attribute können Unicode-Text unter Verwendung beliebiger gültiger XML-Zeichen speichern.
- Zahl – `Number`-Attribute können positive oder negative numerische Werte speichern. Zahlen können bis zu 38 Ziffern und Werte von 10^-128 bis 10^+126 umfassen.

> **Note**
>
> Amazon SQS entfernt voranstehende und nachgestellte Nullen.

- Binär – Binäre Attribute können beliebige binäre Daten speichern, wie etwa komprimierte Daten, verschlüsselte Daten oder Bilder.
- Benutzerdefiniert – Hängen Sie eine benutzerdefinierte Typenbezeichnung an jeden unterstützten Datentyp an, um einen benutzerdefinierten Datentyp zu erstellen. Beispielsweise:
  - `Number.byte`, `Number.short`, `Number.int` und `Number.float` können beim Unterscheiden zwischen den Zahlentypen helfen.
  - `Binary.gif` und `Binary.png` können beim Unterscheiden zwischen den Dateitypen helfen.

> **Note**
>
> Amazon SQS interpretiert, validiert oder verwendet die angefügten Daten nicht.
> Die benutzerdefinierte Typenbezeichnung unterliegt denselben Beschränkungen wie der Nachrichtentext.

Berechnung des MD5-Nachrichtendigests für Nachrichtenattribute

Wenn Sie die verwenden AWS SDK for Java, können Sie diesen Abschnitt überspringen. Die `MessageMD5ChecksumHandler`-Klasse des SDK für Java unterstützt MD5-Nachrichten-Digests für Amazon-SQS-Nachrichtenattribute.

Wenn Sie entweder die Query-API oder eines der AWS SDKs verwenden, das keine MD5-Message Digests für Amazon SQS SQS-Nachrichtenattribute unterstützt, müssen Sie die folgenden Richtlinien verwenden, um die MD5-Nachrichtendigest-Berechnung durchzuführen.

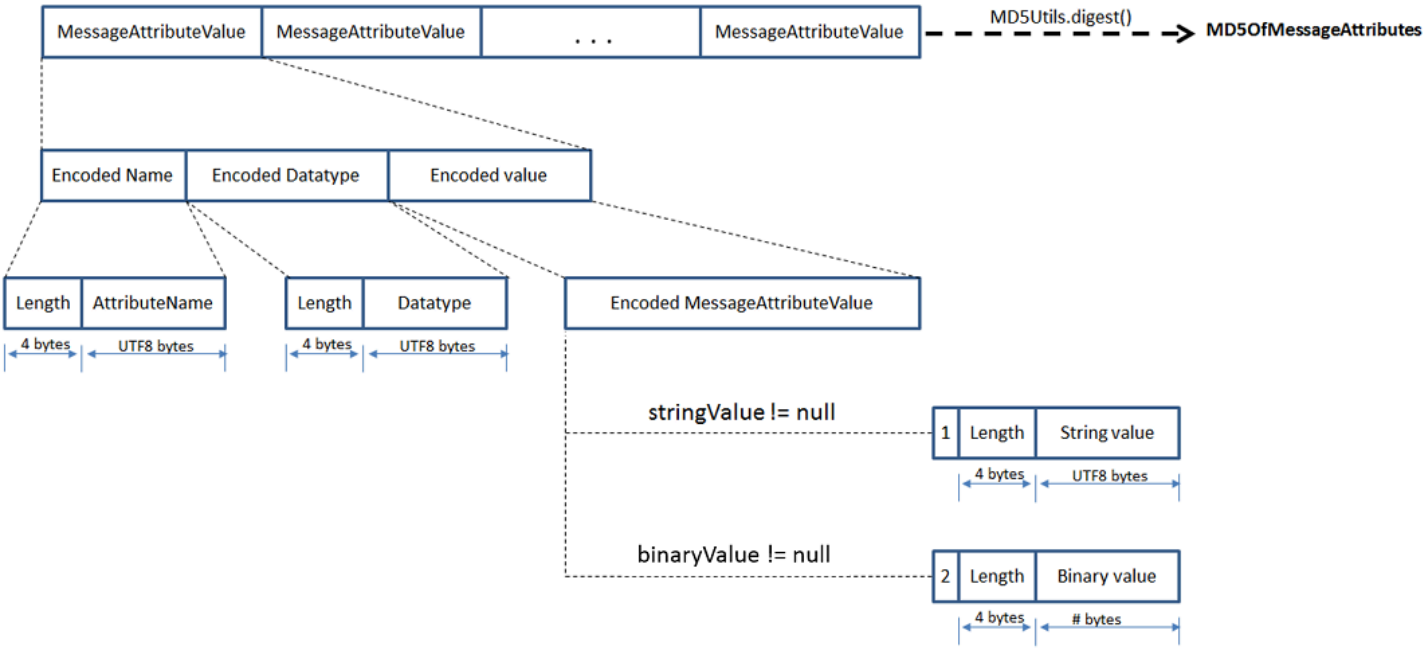
> **Note**
>
> Nehmen Sie bei der Berechnung des MD5-Nachrichten-Digest immer benutzerdefinierte Suffixe des Datentyps mit auf.

Übersicht

Im Folgenden finden Sie eine Übersicht über den MD5-Nachrichtendigest-Berechnungsalgorithmus:

1. Er sortiert alle Nachrichtenattribute in aufsteigender Reihenfolge nach ihrem Namen.
2. Er verschlüsselt die einzelnen Teile der Attribute (`Name`, `Type` und `Value`) im Puffer.
3. Er berechnet den Nachrichtendigest des gesamten Puffers.

In der folgenden Abbildung ist die Verschlüsselung des MD5-Nachrichtendigests für ein einzelnes Nachrichtenattribut dargestellt:



So verschlüsseln Sie ein einzelnes SQS-Nachrichtenattribut

1. Verschlüsselung des Namens: Länge (4 Bytes) und UTF-8-Bytes des Namens.
2. Verschlüsselung des Datentyps: Länge (4 Bytes) und UTF-8-Bytes des Datentyps.
3. Verschlüsselung des Übertragungstyps (`String` oder `Binary`) des Werts (1 Byte).

> **Note**
>
> Für die logischen Datentypen `String` und `Number` wird der Übertragungstyp `String` verwendet.
> Für den logischen Datentyp `Binary` wird der Übertragungstyp `Binary` verwendet.

   a. Verwenden Sie für den Transporttyp `String` die Verschlüsselung 1.
   b. Verwenden Sie für den Transporttyp `Binary` die Verschlüsselung 2.

4. Verschlüsselung des Attributwerts
   a. Verschlüsselung des Attributwerts für den Übertragungstyp `String`: Länge (4 Bytes) und UTF-8 Bytes des Werts.
   b. Verschlüsselung des Attributwerts für den Übertragungstyp `Binary`: Länge (4 Bytes) und Rohbytes des Werts.

Attribute des Amazon-SQS-Nachrichtensystems

Während Sie Nachrichtenattribute verwenden können, um Amazon-SQS-Nachrichten benutzerdefinierte Metadaten für Ihre Anwendungen anzufügen, können Sie Nachrichtensystemattribute verwenden, um Metadaten für andere AWS -Services wie AWS X-Ray zu speichern. Für weitere Informationen siehe den `MessageSystemAttribute`-Anforderungsparameter der API-Aktionen `SendMessage` und `SendMessageBatch`, das `AWSTraceHeader`-Attribut der API-Aktion `ReceiveMessage` sowie den `MessageSystemAttributeValue`-Datentyp in der Amazon-Simple-Queue-Service-API-Referenz.

Nachrichtensystemattribute sind genau wie Nachrichtenattribute strukturiert, mit folgenden Ausnahmen:

- Derzeit wird `AWSTraceHeader` als einziges Nachrichtensystemattribut unterstützt. Sein Typ muss sein `String` und sein Wert muss eine korrekt formatierte AWS X-Ray Trace-Header-Zeichenfolge sein.
- Die Größe eines Nachrichtensystemattributs zählt nicht zur Gesamtgröße einer Nachricht.

Für die Verarbeitung von Amazon-SQS-Nachrichten erforderliche Ressourcen

Zur Einschätzung der für die Verarbeitung der Nachrichten in der Warteschlange benötigten Ressourcen kann Amazon SQS die ungefähre Anzahl an verzögerten, sichtbaren und nicht sichtbaren Nachrichten in einer Warteschlange bestimmen. Weitere Informationen zur Sichtbarkeit finden Sie unter Amazon-SQS-Zeitbeschränkung für die Sichtbarkeit.

> **Note**
>
> Bei Standard-Warteschlangen ist das Ergebnis aufgrund der verteilten Architektur von Amazon SQS ein ungefährer Wert. In den meisten Fällen sollte die Anzahl nahe an der tatsächlichen Anzahl von Nachrichten in der Warteschlange liegen.
> Bei FIFO-Warteschlangen ist der Wert exakt.

Die folgende Tabelle enthält die Attributnamen, die mit der `GetQueueAttributes`-Aktion verwendet werden.

Aufgabe	Attributname
Abrufen der ungefähren Anzahl der Nachrichten, die von der Warteschlange abgerufen werden können.	`ApproximateNumberOfMessagesVisible`
Abrufen der ungefähren Anzahl der Nachrichten in der Warteschlange, die verzögert und zum Lesen nicht sofort verfügbar sind. Dies kann vorkommen, wenn die Warteschlange als Verzögerungswarteschlange konfiguriert ist oder eine Mitteilung mit einem Verzögerungsparameter gesendet worden ist.	`ApproximateNumberOfMessagesDelayed`
Abrufen der ungefähren Anzahl der in Übertragung befindlichen Mitteilungen. Die Mitteilungen befinden sich in Übertragung, wenn sie an einen Client gesendet, aber noch	`ApproximateNumberOfMessagesNotVisible`

Aufgabe	Attributname
nicht gelöscht worden sind oder noch nicht das Ende ihres Sichtbarkeitsfensters erreicht haben.	

Auflistung der Warteschlangenpaginierung

Die API-Methoden `listQueues` und `listDeadLetterQueues` unterstützen optionale Steuerelemente für die Paginierung. Standardmäßig geben diese API-Methoden bis zu 1 000 Warteschlangen in der Antwortnachricht zurück. Sie können den `MaxResults`-Parameter so einstellen, dass bei jeder Antwort weniger Ergebnisse zurückgegeben werden.

Setzen Sie den Parameter `MaxResults` in der Anforderung `listQueues` oder `listDeadLetterQueues`, um die maximale Anzahl von Ergebnissen anzugeben, die in der Antwort zurückgegeben werden sollen. Wenn Sie `MaxResults` nicht festlegen, enthält die Antwort maximal 1 000 Ergebnisse und der `NextToken`-Wert in der Antwort ist Null.

Wenn Sie `MaxResults` festlegen, enthält die Antwort einen Wert für `NextToken`, wenn weitere Ergebnisse zur Anzeige vorhanden sind. Verwenden Sie `NextToken` als Parameter in Ihrer nächsten Anforderung an `listQueues`, um die nächste Ergebnisseite zu erhalten. Wenn keine weiteren Ergebnisse zur Anzeige vorhanden sind, ist der `NextToken`-Wert in der Antwort Null.

Amazon-SQS-Kostenzuordnungs-Tags

Zum Strukturieren und Identifizieren Ihrer Amazon-SQS-Warteschlangen für die Kostenzuordnung können Sie Metadaten-Tags hinzufügen, die den Zweck, den Eigentümer oder die Umgebung einer Warteschlange identifizieren. Dies ist vor allem nützlich, wenn Sie viele Warteschlangen haben. Zur Konfiguration von Tags mit der Amazon-SQS-Konsole siehe the section called “Konfigurieren von Tags für eine Warteschlange”

Mithilfe von Tags zur Kostenzuweisung können Sie Ihre AWS Rechnung so organisieren, dass sie Ihrer eigenen Kostenstruktur entspricht. Melden Sie sich dazu an, damit Ihre AWS-Konto Rechnung die Tagschlüssel und -werte enthält. Weitere Informationen finden Sie unter Einrichten Ihres monatlichen Kostenzuordnungsberichts im AWS Billing Benutzerhandbuch.

Jedes Tag besteht aus einem Schlüssel-Wert-Paar, das Sie definieren. Beispielsweise können Sie auf einfache Weise Ihre Produktions- und Test- Warteschlangen identifizieren, wenn Sie Ihre Warteschlangen wie folgt kennzeichnen:

Warteschlange	Schlüssel	Wert
`MyQueueA`	`QueueType`	`Production`
`MyQueueB`	`QueueType`	`Testing`

> **Note**
>
> Beachten Sie bei der Verwendung von Warteschlangen-Tags die folgenden Richtlinien:
>
> - Es wird nicht empfohlen, einer Warteschlange mehr als 50 Tags hinzuzufügen. Tagging unterstützt Unicode-Zeichen in UTF-8.
> - Tags haben keine semantische Bedeutung. Amazon SQS interpretiert Tags als Zeichenfolgen.
> - Bei Tags muss die Groß- und Kleinschreibung beachtet werden.
> - Ein neuer Tag mit einem Schlüssel, der mit dem eines vorhandenen Tags identisch ist, überschreibt den vorhandenen Tag.
> - Tagging-Aktionen sind auf 30 TPS pro Tag begrenzt. AWS-Konto Wenn Ihre Anwendung einen höheren Durchsatz erfordert, reichen Sie eine Anfrage ein.
>
> Eine vollständige Liste von Tag-Einschränkungen finden Sie unter Kontingente.

Kurz- und Langabfragen in Amazon SQS

Amazon SQS bietet kurze und lange Abfrageoptionen für den Empfang von Nachrichten aus einer Warteschlange. Berücksichtigen Sie die Anforderungen Ihrer Anwendung an Reaktionsfähigkeit und Kosteneffizienz, wenn Sie zwischen diesen beiden Abfrageoptionen wählen:

- Kurzes Polling (Standard) — Die `ReceiveMessage`Anfrage fragt eine Untergruppe von Servern ab (basierend auf einer gewichteten Zufallsverteilung), um verfügbare Nachrichten zu finden, und sendet eine sofortige Antwort, auch wenn keine Nachrichten gefunden wurden.

- Langes Polling — `ReceiveMessage`fragt alle Server nach Nachrichten ab und sendet bis zum angegebenen Höchstwert eine Antwort, sobald mindestens eine Nachricht verfügbar ist. Eine leere Antwort wird nur gesendet, wenn die Wartezeit für die Abfrage abgelaufen ist. Diese Option kann die Anzahl leerer Antworten reduzieren und möglicherweise die Kosten senken.

In den folgenden Abschnitten werden die Details von Kurz- und Langabfragen erläutert.

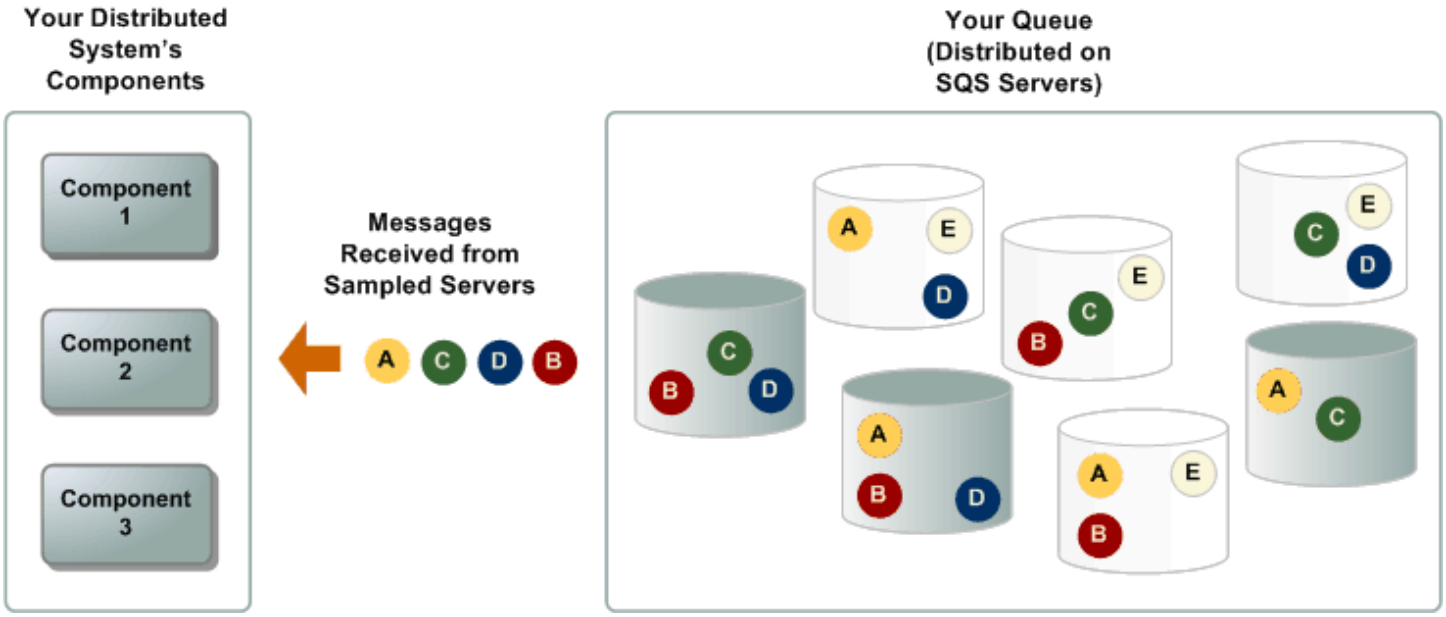
Themen

- Abrufen von Nachrichten durch Kurzabfragen
- Konsumieren von Nachrichten mithilfe von Langabfragen
- Unterschiede zwischen Lang- und Kurzabfragen

Abrufen von Nachrichten durch Kurzabfragen

Wenn Sie Nachrichten aus einer Warteschlange (FIFO oder Standard) mithilfe von Kurzabfragen verarbeiten, nimmt Amazon SQS eine Stichprobe von einer Teilmenge seiner Server (basierend auf einer gewichteten Zufallsverteilung) vor und gibt nur Nachrichten von diesen Servern zurück. Daher werden für eine bestimmte `ReceiveMessage`-Anforderung möglicherweise nicht alle Nachrichten zurückgegeben. Wenn Sie jedoch weniger als 1000 Nachrichten in Ihrer Warteschlange haben, gibt die nächste Anfrage Ihre Nachrichten zurück. Wenn Sie das Abrufen aus Ihren Warteschlangen fortsetzen, nimmt Amazon SQS Stichproben aller seiner Server und Sie erhalten alle Ihre Nachrichten.

Das folgende Diagramm zeigt das Verhalten von Nachrichten bei Kurzabfragen, die von einer Standard-Warteschlange zurückgegeben werden, nachdem eine Ihrer Systemkomponenten eine Empfangsanforderung stellt. Amazon SQS nimmt Stichproben von mehreren seiner Server (grau dargestellt) und gibt die Nachrichten A, C, D und B von diesen Servern zurück. Nachricht E wird nicht für diese Anforderung zurückgegeben, sondern für eine nachfolgende Anforderung.



Konsumieren von Nachrichten mithilfe von Langabfragen

Ist die Wartezeit für die `ReceiveMessage`-API-Aktion größer als 0, ist eine lange Abfrage wirksam. Die maximale Wartezeit für lange Abfragen beträgt 20 Sekunden. Mithilfe von Langabfragen können Sie die Kosten für die Verwendung von Amazon SQS reduzieren, indem Sie die Anzahl der leeren Antworten (wenn bei einer `ReceiveMessage`-Anfrage keine Nachrichten vorliegen) und fälschlicherweise leeren Antworten (wenn Nachrichten vorliegen, diese aber nicht in einer Antwort enthalten sind) eliminieren. Informationen zum Aktivieren von Langabfragen für eine neue oder vorhandene Warteschlange mithilfe der Amazon-SQS-Konsole finden Sie unter Konfiguration von Warteschlangenparametern mit der Amazon SQS SQS-Konsole. Bewährte Methoden finden Sie unter Einrichten von Langabfragen.

Die Langabfrage bietet die folgenden Vorteile:

- Eliminieren leerer Antworten, da Amazon SQS wartet, bis eine Nachricht in der Warteschlange vorhanden ist, bevor eine Antwort gesendet wird. Sofern die Verbindung nicht abläuft, enthält die Antwort auf die `ReceiveMessage`-Anfrage mindestens eine der verfügbaren Nachrichten bis zur in der Aktion `ReceiveMessage` definierten maximalen Anzahl an Nachrichten. In seltenen Fällen erhalten Sie möglicherweise leere Antworten, auch wenn eine Warteschlange noch Nachrichten enthält, insbesondere wenn Sie einen niedrigen Wert für den `ReceiveMessageWaitTimeSeconds`-Parameter angeben.
- Reduzieren Sie falsche Leerantworten, indem Sie alle – nicht nur eine Teilmenge – von Amazon-SQS-Servern abfragen.

- Zurückgeben von Nachrichten, sobald sie verfügbar werden.

Informationen darüber, wie Sie überprüfen, ob eine Warteschlange leer ist, finden Sie unter Bestätigen, dass eine Amazon SQS SQS-Warteschlange leer ist.

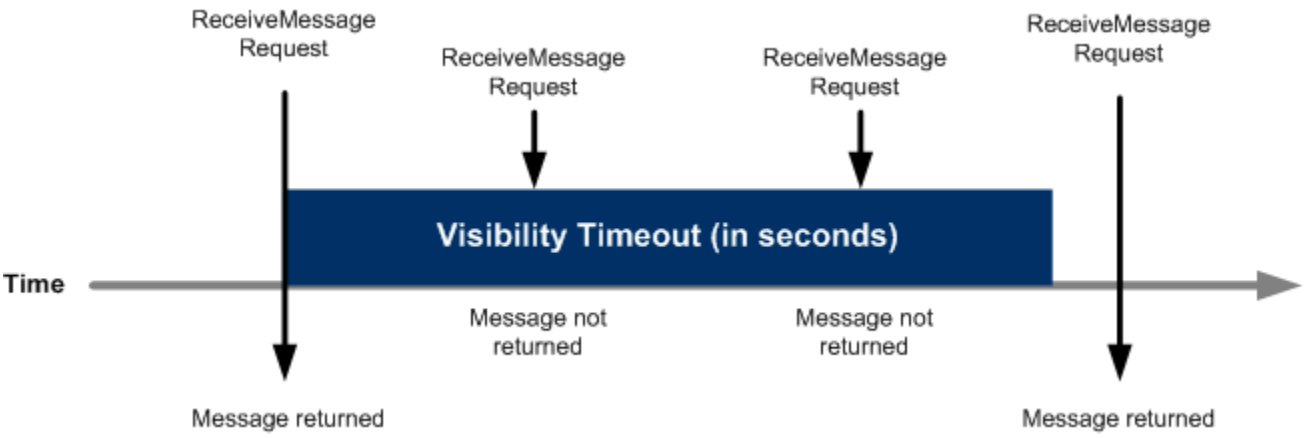
Unterschiede zwischen Lang- und Kurzabfragen

Kurzabfragen werden ausgeführt, wenn der Parameter `WaitTimeSeconds` einer `ReceiveMessage`-Anfrage mit einer der beiden folgenden Methoden auf den Wert `0` festgelegt wurde:

- Der `ReceiveMessage`-Aufruf legt `WaitTimeSeconds` auf `0` fest.
- Der `ReceiveMessage`-Aufruf legt `WaitTimeSeconds` nicht fest, aber das Warteschlangenattribut `ReceiveMessageWaitTimeSeconds` ist auf `0` festgelegt.

Amazon-SQS-Zeitbeschränkung für die Sichtbarkeit

Wenn ein Konsument eine Nachricht aus einer Warteschlange empfängt und verarbeitet, verbleibt diese Nachricht in der Warteschlange. Amazon SQS löscht die Nachricht nicht automatisch. Da es sich bei Amazon SQS um ein verteiltes System handelt, gibt es keine Garantie dafür, dass der Konsument die Nachricht tatsächlich erhält (z. B. aufgrund eines Konnektivitätsproblems oder aufgrund eines Problems in der Konsumentenanwendung). Daher muss der Konsument die Nachricht nach dem Empfang und der Verarbeitung aus der Warteschlange löschen.



Die Nachricht verbleibt direkt nach dem Empfang in der Warteschlange. Um zu verhindern, dass andere Konsumenten die Nachricht erneut verarbeiten, legt Amazon SQS eine Zeitbeschränkung für die Sichtbarkeit fest, also einen Zeitraum, während dessen Amazon SQS verhindert, dass andere Konsumenten die Nachricht empfangen und verarbeiten können. Die Standardzeitbeschränkung

für die Sichtbarkeit einer Nachricht ist 30 Sekunden. Der Mindestwert beträgt 0 Sekunden. Der Höchstwert beträgt 12 Stunden. Informationen zur Konfiguration der Zeitbeschränkung für die Sichtbarkeit für eine Warteschlange mit der Konsole finden Sie unter Konfiguration von Warteschlangenparametern mit der Amazon SQS SQS-Konsole.

> **Note**
>
> Für Standard-Warteschlangen kann auch durch die Zeitbeschränkung für die Sichtbarkeit nicht garantiert werden, dass eine Nachricht mehrmals empfangen wird. Weitere Informationen finden Sie unter Eine t-least-once Lieferung.
> FIFO-Warteschlangen ermöglichen es dem Produzenten oder Verbraucher, mehrere Wiederholungen zu versuchen:
>
> - Wenn der Produzent eine fehlgeschlagene `SendMessage`-Aktion feststellt, kann er das Senden so oft wie nötig wiederholen und dabei dieselbe Nachrichten-Deduplizierungs-ID verwenden. Unter der Annahme, dass der Produzent vor Ablauf des Deduplizierungsintervalls mindestens eine Bestätigung erhält, wirken sich mehrere Wiederholungsversuche weder auf die Reihenfolge der Nachrichten aus, noch führen sie zu Duplikaten.
> - Wenn der Verbraucher eine fehlgeschlagene `ReceiveMessage`-Aktion feststellt, kann er sie so oft wie nötig wiederholen und dabei dieselbe ID für den Versuch verwenden, die Anfrage zu empfangen. Unter der Annahme, dass der Verbraucher mindestens eine Bestätigung erhält, bevor die Sichtbarkeitszeitbeschränkung abläuft, wirken sich mehrere Wiederholungsversuche nicht auf die Reihenfolge der Nachrichten aus.
> - Wenn Sie eine Nachricht mit einer Nachrichtengruppen-ID erhalten, werden keine weiteren Nachrichten für dieselbe Nachrichtengruppen-ID zurückgegeben, es sei denn, Sie löschen die Nachricht oder sie wird sichtbar.

Themen

- In-Flight-Nachrichten
- Einrichten der Zeitbeschränkung für die Sichtbarkeit
- Ändern der Zeitbeschränkung für die Sichtbarkeit für eine Nachricht
- Beenden der Zeitbeschränkung für die Sichtbarkeit für eine Nachricht

In-Flight-Nachrichten

Eine Amazon-SQS-Nachricht hat drei grundlegende Status:

1. Von einem Produzenten an eine Warteschlange gesendet.
2. Von einem Verbraucher aus der Warteschlange empfangen.
3. Aus der Warteschlange gelöscht.

Eine Nachricht gilt als gespeichert, wenn sie von einem Produzenten an eine Warteschlange gesendet, aber noch nicht von einem Verbraucher aus der Warteschlange empfangen wurde (d. h. zwischen den Zuständen 1 und 2). Es gibt kein Kontingent für die Anzahl der gespeicherten Nachrichten. Eine Nachricht gilt als In-Flight, wenn sie von einem Produzenten aus einer Warteschlange empfangen, aber noch nicht aus der Warteschlange gelöscht wurde (d. h. zwischen den Zuständen 2 und 3). Es gibt kein Kontingent für die Anzahl der In-Flight-Nachrichten.

> **Important**
>
> Kontingente, die für In-Flight-Nachrichten gelten, haben nichts mit der unbegrenzten Anzahl an gespeicherten Nachrichten zu tun.

Bei den meisten Standard-Warteschlangen (abhängig vom Warteschlangenverkehr und dem Nachrichtenrückstand) kann es maximal etwa 120 000 In-Flight-Nachrichten geben (die von einem Verbraucher aus einer Warteschlange empfangen, aber noch nicht aus der Warteschlange gelöscht wurden). Wenn Sie dieses Kontingent während der Verwendung von Kurzabfragen erreichen, gibt Amazon SQS die Fehlermeldung `OverLimit` zurück. Wenn Sie Langabfragen verwenden, gibt Amazon SQS keine Fehlermeldungen zurück. Um zu vermeiden, dass dieses Kontingent erreicht wird, sollten Sie Nachrichten aus der Warteschlange löschen, nachdem sie verarbeitet wurden. Sie können auch die Anzahl der Warteschlangen erhöhen, die Sie zur Verarbeitung Ihrer Nachrichten verwenden. Um eine Kontingenterhöhung anzufordern, übermitteln Sie eine Support-Anforderung.

Bei FIFO-Warteschlangen können maximal 20 000 In-Flight-Nachrichten enthalten sein (die von einem Verbraucher aus einer Warteschlange empfangen, aber noch nicht aus der Warteschlange gelöscht wurden). Wenn Sie dieses Kontingent erreichen, gibt Amazon SQS keine Fehlermeldungen zurück.

> **Important**
>
> Bei der Arbeit mit FIFO-Warteschlangen schlagen `DeleteMessage`-Operationen fehl, wenn die Anfrage außerhalb der Zeitbeschränkung für die Sichtbarkeit eingeht. Wenn die Zeitbeschränkung für die Sichtbarkeit 0 Sekunden beträgt, muss die Nachricht innerhalb derselben Millisekunde gelöscht werden, in der sie gesendet wurde. Andernfalls wird sie als aufgegeben betrachtet. Dies kann dazu führen, dass Amazon SQS doppelte Nachrichten in dieselbe Antwort auf einen `ReceiveMessage`-Vorgang einbezieht, wenn der `MaxNumberOfMessages`-Parameter größer als 1 ist. Weitere Informationen finden Sie unter So funktioniert die Amazon-SQS FIFO-API.

Einrichten der Zeitbeschränkung für die Sichtbarkeit

Die Zeitbeschränkung für die Sichtbarkeit beginnt, sobald Amazon SQS eine Nachricht zurückgibt. Innerhalb dieser Zeit verarbeitet und löscht der Konsument die Nachricht. Wenn jedoch der Konsument ausfällt, bevor er die Nachricht löscht, und Ihr System nicht vor Ablauf der Zeitbeschränkung für die Sichtbarkeit die Aktion `DeleteMessage` für diese Nachricht aufruft, wird die Nachricht für andere Konsumenten wieder sichtbar und die Nachricht wird erneut empfangen. Wenn eine Nachricht nur einmal empfangen werden darf, muss Ihr Konsument sie innerhalb der Zeitbeschränkung für die Sichtbarkeit löschen.

Für jede Amazon-SQS-Warteschlange sind 30 Sekunden als Standardwert für die Zeitbeschränkung für die Sichtbarkeit vorgegeben. Sie können diese Einstellung für die gesamte Warteschlange ändern. In der Regel sollten Sie für die Zeitbeschränkung für die Sichtbarkeit die maximale Zeitdauer festlegen, wie lange Ihre Anwendung benötigt, eine Nachricht aus der Warteschlange zu verarbeiten und zu löschen. Sie können beim Empfangen von Nachrichten auch eine spezifische Zeitbeschränkung für die Sichtbarkeit für die zurückgegebenen Nachrichten festlegen, ohne die Zeitbeschränkungen der Warteschlange zu ändern. Weitere Informationen finden Sie unter den bewährten Methoden im Abschnitt Zeitnahe Verarbeitung von Nachrichten.

Wenn Sie nicht wissen, wie lange die Bearbeitung einer Nachricht dauert, erstellen Sie einen Heartbeat für Ihren Kundenprozess: Geben Sie das anfängliche Timeout für die Sichtbarkeit an (z. B. 2 Minuten) und verlängern Sie dann, solange Ihr Kunde noch an der Nachricht arbeitet, die Sichtbarkeitszeitbeschränkung jede Minute um 2 Minuten.

> **Important**
>
> Die maximale Zeitbeschränkung für die Sichtbarkeit beträgt 12 Stunden ab dem Zeitpunkt, an dem Amazon SQS die `ReceiveMessage`-Anforderung empfängt. Durch die Verlängerung der Sichtbarkeitszeitbeschränkung wird das Maximum von 12 Stunden nicht zurückgesetzt. Darüber hinaus können Sie das Timeout für eine einzelne Nachricht möglicherweise nicht auf die vollen 12 Stunden (z. B. 43 200 Sekunden) festlegen, seit die `ReceiveMessage`-Anfrage den Timer initiiert hat. Wenn Sie beispielsweise eine Nachricht erhalten und sofort das Maximum von 12 Stunden festlegen, indem Sie einen `ChangeMessageVisibility`-Aufruf mit einem `VisibilityTimeout` von 43 200 Sekunden senden, schlägt der Vorgang wahrscheinlich fehl. Die Verwendung eines Werts von 43 195 Sekunden funktioniert jedoch, sofern es nicht zu einer erheblichen Verzögerung zwischen dem Anfordern der Nachricht über `ReceiveMessage` und der Aktualisierung der Sichtbarkeitszeitbeschränkung kommt. Wenn Ihr Verbraucher länger als 12 Stunden benötigt, sollten Sie die Verwendung von Step Functions in Betracht ziehen.

Ändern der Zeitbeschränkung für die Sichtbarkeit für eine Nachricht

Wenn Sie eine Nachricht aus einer Warteschlange empfangen und mit der Verarbeitung beginnen, kann es vorkommen, dass die Zeitbeschränkung für die Sichtbarkeit für die Warteschlange nicht ausreicht (z. B. für das Verarbeiten und Löschen einer Nachricht). Sie können die Sichtbarkeit einer Nachricht verkürzen oder verlängern. Geben Sie dazu mit der Aktion `ChangeMessageVisibility` einen neuen Wert für die Zeitbeschränkung ein.

Beispiel: Die Standardzeitbeschränkung für eine Warteschleife beträgt 60 Sekunden. Seit dem Empfang der Nachricht sind bereits 15 Sekunden vergangen und Sie senden einen `ChangeMessageVisibility`-Aufruf mit dem Attribut `VisibilityTimeout` auf 10 Sekunden. Diese 10 Sekunden werden ab dem Zeitpunkt des `ChangeMessageVisibility`-Aufrufs heruntergezählt. Daher führt jeder Versuch, innerhalb von 10 Sekunden nach der Änderung der Zeitbeschränkung für die Sichtbarkeit (insgesamt 25 Sekunden) die Nachricht zu löschen, zu einem Fehler.

> **Note**
>
> Der neue Zeitbeschränkungszeitraum beginnt ab dem Aufruf der Aktion `ChangeMessageVisibility`. Darüber hinaus wird der neue Zeitbeschränkungszeitraum nur auf diesen Empfang der Nachricht angewendet. `ChangeMessageVisibility` wirkt

sich nicht auf die Zeitbeschränkung für einen späteren Empfang der Nachricht oder spätere Warteschlangen aus.

Beenden der Zeitbeschränkung für die Sichtbarkeit für eine Nachricht

Wenn Sie eine Nachricht aus einer Warteschlange empfangen, stellen Sie möglicherweise fest, dass Sie diese Nachricht nicht verarbeiten und löschen möchten. Amazon SQS ermöglicht das Beenden der Zeitbeschränkung für die Sichtbarkeit für eine bestimmte Nachricht. Dadurch wird die Nachricht sofort wieder für andere Komponenten im System sichtbar und kann verarbeitet werden.

Um die Zeitbeschränkung für die Sichtbarkeit einer Nachricht nach dem Aufrufen von `ReceiveMessage` zu beenden, rufen Sie `ChangeMessageVisibility` mit dem Attribut `VisibilityTimeout` auf, das auf 0 Sekunden festgelegt ist.

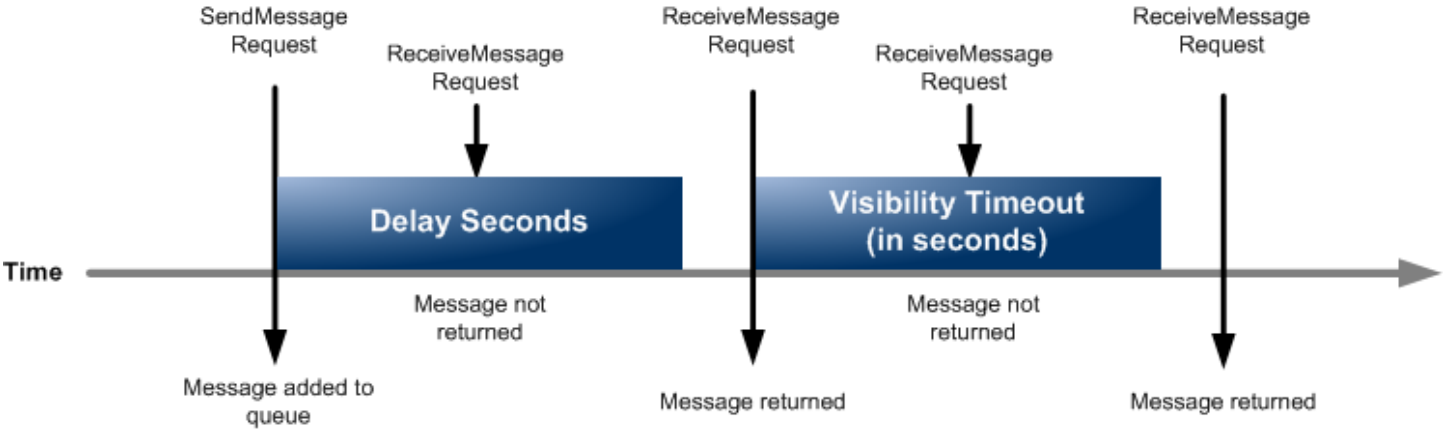
Amazon-SQS-Verzögerungswarteschlangen

Mit Verzögerungswarteschlangen können Sie die Zustellung neuer Nachrichten an Verbraucher um einige Sekunden verschieben, z. B. wenn Ihre Verbraucheranwendung zusätzliche Zeit für die Verarbeitung von Nachrichten benötigt. Wenn Sie eine Verzögerungswartschlange erstellen, bleiben an diese Warteschlange gesendete Nachrichten für Konsumenten für die Dauer des Verzögerungszeitraums unsichtbar. Die Standardverzögerung (Mindestverzögerung) für eine Warteschlange beträgt 0 Sekunden. Der Maximalwert beträgt 15 Minuten. Weitere Informationen zur Konfiguration von Verzögerungswarteschlangen mit der Konsole finden Sie unter Konfiguration von Warteschlangenparametern mit der Amazon SQS SQS-Konsole.

> **Note**
>
> Für Standard-Warteschlangen ist die Einstellung für die Verzögerung pro Warteschlange nicht retroaktiv – wenn Sie die Einstellung ändern, wirkt sich dies nicht auf die Verzögerung von Nachrichten aus, die sich bereits in der Warteschlange befinden.
> Für FIFO-Warteschlangen ist die Einstellung für die Verzögerung pro Warteschlange retroaktiv – wenn Sie die Einstellung ändern, wirkt sich dies auf die Verzögerung von Nachrichten aus, die sich bereits in der Warteschlange befinden.

Verzögerungswarteschlangen funktionieren vergleichbar mit Zeitbeschränkungen für die Sichtbarkeit, da durch beide Funktionen Nachrichten eine bestimmte Zeit lang für Konsumenten

nicht verfügbar gemacht werden. Der Unterschied zwischen beidem besteht darin, dass in Verzögerungswarteschlangen die Nachricht direkt nach dem Hinzufügen zur Warteschlange verborgen wird, bei Zeitbeschränkungen für die Sichtbarkeit hingegen erst, nachdem die Nachricht aus der Warteschlange abgerufen wurde. Das folgende Diagramm verdeutlicht die Beziehung zwischen Verzögerungswarteschlangen und Zeitbeschränkungen für die Sichtbarkeit.



Um Verzögerungssekunden für einzelne Nachrichten statt für eine ganze Warteschlange festzulegen, verwenden Sie Nachrichten-Timer, damit Amazon SQS den `DelaySeconds`-Wert des Nachrichten-Timers statt des `DelaySeconds`-Werts der Verzögerungswarteschlange verwenden kann.

Temporäre Amazon-SQS-Warteschlangen

Temporäre Warteschlangen helfen Ihnen, Entwicklungszeit und Bereitstellungskosten zu sparen, wenn Sie gängige Nachrichtenmuster wie Anfrage-Antwort verwenden. Sie können den temporären Warteschlangen-Client verwenden, um kostengünstige, anwendungsverwaltete temporäre Warteschlangen mit hohem Durchsatz zu erstellen.

Der Client ordnet mehrere temporäre Warteschlangen – anwendungsverwaltete Warteschlangen, die bei Bedarf für einen bestimmten Prozess erstellt werden – automatisch einer einzigen Amazon-SQS-Warteschlange zu. Auf diese Weise kann Ihre Anwendung weniger API-Aufrufe durchführen und einen höheren Durchsatz verzeichnen, wenn der Datenverkehr zu jeder temporären Warteschlange gering ist. Wenn eine temporäre Warteschlange nicht mehr verwendet wird, bereinigt der Client die temporäre Warteschlange automatisch. Dies gilt auch dann, wenn einige Prozesse, die den Client nutzen, nicht ordnungsgemäß heruntergefahren werden.

Im Folgenden werden die Vorteile von temporären Warteschlangen beschrieben:

- Sie dienen als einfache Kommunikationskanäle für bestimmte Threads oder Prozesse.
- Sie können erstellt und gelöscht werden, ohne dass zusätzliche Kosten anfallen.

- Diese sind API-kompatibel mit statischen (normalen) Amazon-SQS-Warteschlangen. Dies bedeutet, dass vorhandener Code, mit dem Nachrichten gesendet und empfangen werden, Nachrichten an virtuelle Warteschlangen senden und von ihnen empfangen kann.

Themen

- Virtuelle Warteschlangen
- Request-Response-Messaging-Muster (virtuelle Warteschlangen)
- Beispielszenario: Verarbeiten einer Anmeldeanforderung
  - Auf der Client-Seite
  - Auf der Server-Seite
- Bereinigen von Warteschlangen

Virtuelle Warteschlangen

Virtuelle Warteschlangen sind lokale Datenstrukturen, die vom Client temporärer Warteschlangen erstellt werden. Mit virtuellen Warteschlangen können Sie mehrere Ziele mit geringem Datenverkehr in einer einzigen Amazon-SQS-Warteschlange zusammenfassen. Bewährte Methoden finden Sie unter Vermeiden der Wiederverwendung derselben Nachrichtengruppen-ID bei virtuellen Warteschlangen.

> **Note**
>
> - Beim Erstellen einer virtuellen Warteschlange werden nur temporäre Datenstrukturen erstellten, in denen Konsumenten Nachrichten empfangen können. Da eine virtuelle Warteschlange keine API-Aufrufe an Amazon SQS sendet, entstehen für virtuelle Warteschlangen keine Kosten.
> - TPS-Kontingente gelten für alle virtuellen Warteschlangen in einer einzelnen Hostwarteschlange. Weitere Informationen finden Sie unter Amazon SQS SQS-Nachrichtenkontingente.

Die `AmazonSQSVirtualQueuesClient`-Wrapper-Klasse wurde um neue Unterstützung für Attribute im Zusammenhang mit virtuellen Warteschlangen erweitert. Um eine virtuelle Warteschlange erstellen zu können, müssen Sie die `CreateQueue`-API-Aktion mit dem Attribut `HostQueueURL`

aufrufen. Dieses Attribut gibt die vorhandene Warteschlange an, in der die virtuellen Warteschlangen gehostet werden.

Die URL einer virtuellen Warteschlange hat das folgende Format.

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

Wenn ein Produzent die API-Aktion `SendMessage` oder `SendMessageBatch` für die URL einer virtuellen Warteschlange aufruft, verfährt der Client temporärer Warteschlangen wie folgt:

1. Er extrahiert den Namen der virtuellen Warteschlange.
2. Er fügt den Namen der virtuellen Warteschlange als zusätzliches Nachrichtenattribut an.
3. Er sendet die Nachricht an die Host-Warteschlange.

Während der Produzent Nachrichten sendet, fragt ein Hintergrund-Thread die Host-Warteschlange ab und sendet die empfangenen Nachrichten gemäß den entsprechenden Nachrichtenattributen an virtuelle Warteschlangen.

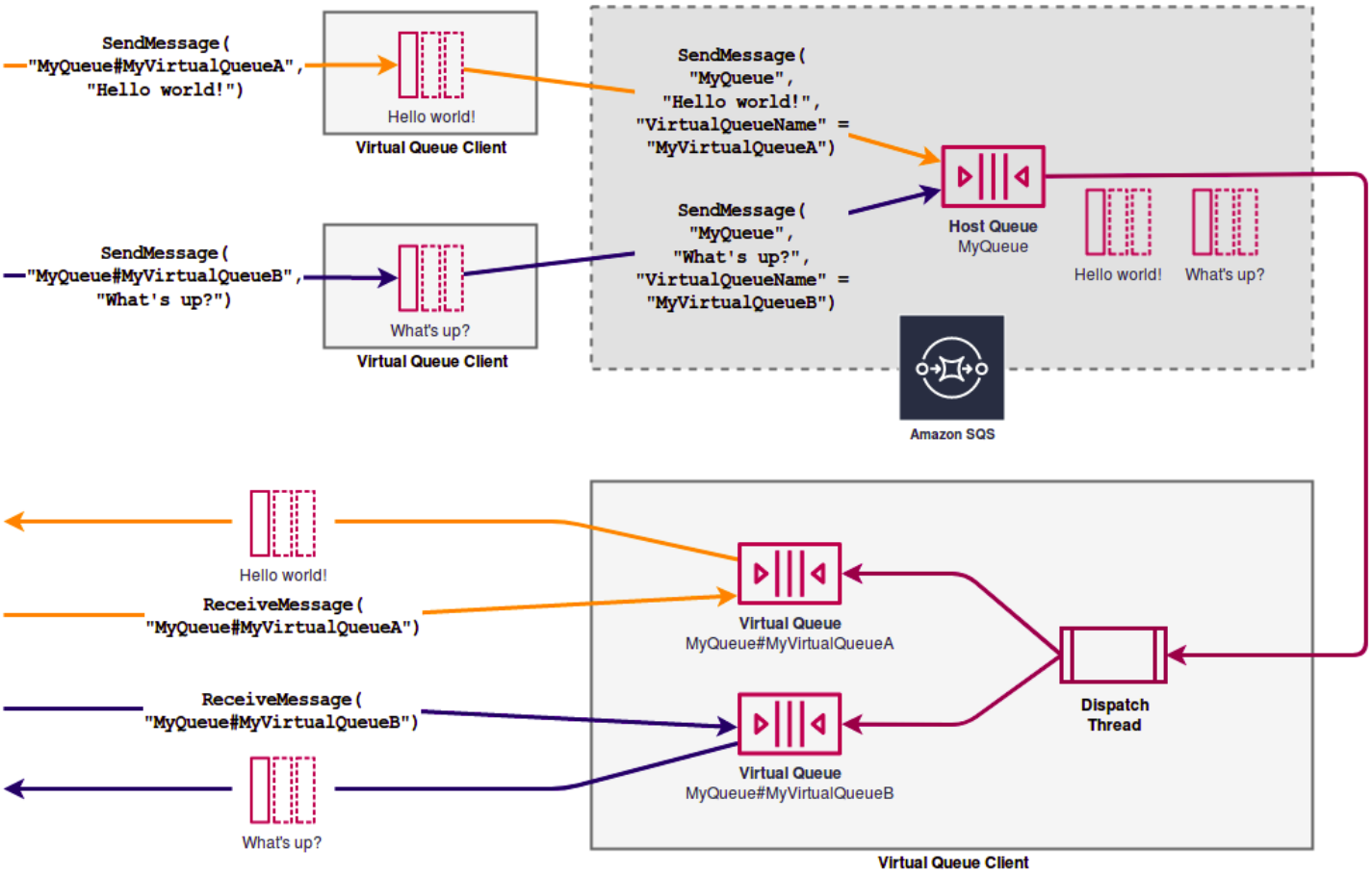
Während der Konsument die `ReceiveMessage`-API-Aktion für die URL einer virtuellen Warteschlange aufruft, blockiert der Client temporärer Warteschlangen den Aufruf lokal, bis der Hintergrund-Thread eine Nachricht in die virtuelle Warteschlange sendet. (Dieser Vorgang ist mit dem Vorabrufen von Nachrichten im gepufferten asynchronen Client vergleichbar: eine einzelne API-Aktion kann für bis zu 10 virtuelle Warteschlangen Nachrichten bereitstellen.) Wenn eine virtuelle Warteschlange gelöscht wird, werden alle Client-seitigen Ressourcen entfernt, ohne Amazon SQS selbst aufzurufen.

Die `AmazonSQSTemporaryQueuesClient`-Klasse wandelt alle von ihr erstellten Warteschlangen automatisch in temporäre Warteschlangen um. Sie erstellt außerdem Host-Warteschlangen mit denselben Warteschlangenattributen automatisch nach Bedarf. Den Namen dieser Warteschlangen ist ein konfigurierbares Präfix (standardmäßig `__RequesterClientQueues__`) gemeinsam, durch das sie als temporäre Warteschlangen identifiziert werden. Auf diese Weise kann der Client als Drop-In-Ersatz fungieren, durch den vorhandener Code optimiert wird, mit dem Warteschlangen erstellt und gelöscht werden. Der Client umfasst auch die `AmazonSQSRequester`- und `AmazonSQSResponder`-Schnittstellen, die eine bidirektionale Kommunikation zwischen Warteschlangen ermöglichen.

Request-Response-Messaging-Muster (virtuelle Warteschlangen)

Der häufigste Anwendungsfall für temporäre Warteschlangen ist das request-response-Messaging-Muster, bei dem ein Anforderer eine temporäre Warteschlange für den Empfang der einzelnen Antwortnachrichten erstellt. Um zu vermeiden, dass für jede Antwortnachricht eine Amazon-SQS-Warteschlange erstellt wird, können Sie mit dem Client für temporäre Warteschlangen mehrere temporäre Warteschlangen ohne Amazon-SQS-API-Aufrufe erstellen. Weitere Informationen finden Sie unter Implementieren von Request-Response-Systemen.

Das folgende Diagramm zeigt eine geläufige Konfiguration mit diesem Muster.



Beispielszenario: Verarbeiten einer Anmeldeanforderung

Das folgende Beispielszenario veranschaulicht, wie die `AmazonSQSRequester`- und `AmazonSQSResponder`-Schnittstellen zur Verarbeitung der Anmeldeanforderung eines Benutzers verwendet werden.

Auf der Client-Seite

```
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();
    }
}
```

Das Senden einer Anmeldeanforderung bewirkt Folgendes:

1. Erstellt eine temporäre Warteschlange.
2. Es fügt die URL der temporären Warteschlange als Attribut an die Nachricht an.
3. Es sendet die Nachricht.
4. Es empfängt eine Antwort von der temporären Warteschlange.
5. Es löscht die temporäre Warteschlange.
6. Es gibt die Antwort zurück.

Auf der Server-Seite

Im folgenden Beispiel wird davon ausgegangen, dass bei der Erstellung ein Thread zur Abfrage der Warteschlange und zum Aufruf der Methode `handleLoginRequest()` für jede Nachricht erstellt wird. Darüber hinaus handelt es sich bei `doLogin()` um eine angenommene Methode.

```
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));
    }
}
```

Bereinigen von Warteschlangen

Um sicherzustellen, dass Amazon SQS alle von virtuellen Warteschlangen beanspruchten Ressourcen im Arbeitsspeicher freigibt, wenn die Anwendung den Client für temporäre Warteschlangen nicht mehr benötigt, sollte die Methode `shutdown()` aufgerufen werden. Sie können auch die Methode `shutdown()` der die `AmazonSQSRequester`-Schnittstelle verwenden.

Der Client temporärer Warteschlangen bietet auch eine Option, mit der sich verwaiste Host-Warteschlangen vermeiden lassen. Alle Warteschlangen, die innerhalb eines Zeitraums

(standardmäßig fünf Minuten) einen API-Aufruf empfangen, werden vom Client mit der API-Aktion `TagQueue` als weiterhin genutzte Warteschlangen markiert.

> **Note**
>
> API-Aktionen, die für eine Warteschlange ausgeführt werden, markieren diese als nicht im Leerlauf befindlich. Dazu gehört auch eine `ReceiveMessage`-Aktion, die keine Nachrichten zurückgibt.

Der Hintergrund-Thread verwendet die API-Aktionen `ListQueues` und `ListTags`, um alle Warteschlange mit dem konfigurierten Präfix zu überprüfen und alle Warteschlangen zu entfernen, die nicht schon mindestens fünf Minuten lang gekennzeichnet sind. Sollte ein Client nicht ordnungsgemäß heruntergefahren werden, können die anderen aktiven Clients ihn auf diese Weise bereinigen. Um einer Verdopplung des Arbeitsaufwands entgegenzuwirken, kommunizieren alle Clients mit demselben Präfix über eine freigegebene, interne Arbeitswarteschlange, die nach dem Präfix benannt ist, miteinander.

Amazon-SQS-Nachrichten-Timer

Mit Nachrichten-Timern können Sie einen Zeitraum festlegen, innerhalb dessen eine Nachricht, die Sie zu einer Warteschlange hinzufügen, unsichtbar ist. Wenn Sie beispielsweise eine Nachricht mit dem 45-Sekunden-Timer senden, wird die Nachricht erst nach den ersten 45 Sekunden in der Warteschlange für Konsumenten sichtbar. Die Standardverzögerung (Mindestverzögerung) für eine Nachricht beträgt 0 Sekunden. Der Maximalwert beträgt 15 Minuten. Informationen zum Senden von Nachrichten mit Timern über die Konsole finden Sie unter Senden einer Nachricht.

> **Note**
>
> FIFO-Warteschlangen unterstützen keine Timer für einzelne Nachrichten.

Um einen Verzögerungszeitraum für eine ganze Warteschlange statt nur für einzelne Nachrichten festzulegen, verwenden Sie Verzögerungswarteschlangen. Eine Nachrichten-Timer-Einstellung für eine einzelne Nachricht überschreibt jeden `DelaySeconds`-Wert auf einer Amazon-SQS-Verzögerungswarteschlange.

Zugriff auf Amazon EventBridge Pipes über die Amazon SQS SQS-Konsole

Amazon EventBridge Pipes verbindet Quellen mit Zielen. Pipes sind für point-to-point Integrationen zwischen unterstützten Quellen und Zielen vorgesehen und unterstützen erweiterte Transformationen und Anreicherungen. EventBridge Pipes bieten eine hoch skalierbare Möglichkeit, Ihre Amazon SQS-Warteschlange mit AWS Services wie Step Functions, Amazon SQS und API Gateway sowie Software-as-a-Service (SaaS) -Anwendungen von Drittanbietern wie Salesforce zu verbinden.

Zum Einrichten einer Pipe wählen Sie die Quelle aus, fügen Sie optionale Filterung hinzu, definieren Sie die optionale Anreicherung und wählen Sie das Ziel für die Ereignisdaten.

Auf der Detailseite für eine Amazon-SQS-Warteschlange können Sie sich die Pipes ansehen, die diese Warteschlange als Quelle verwenden. Von dort aus können Sie auch Folgendes tun:

- Starten Sie die EventBridge Konsole, um die Pipe-Details anzuzeigen.
- Starten Sie die EventBridge Konsole, um eine neue Pipe mit der Warteschlange als Quelle zu erstellen.

Weitere Informationen zur Konfiguration einer Amazon SQS SQS-Warteschlange als Pipe-Quelle finden Sie unter Amazon SQS SQS-Warteschlange als Quelle im EventBridge Amazon-Benutzerhandbuch. Weitere Informationen zu EventBridge Pipes im Allgemeinen finden Sie unter EventBridge Pipes.

So greifen Sie auf EventBridge Pipes für eine bestimmte Amazon SQS SQS-Warteschlange zu

1. Öffnen Sie die Seite Warteschlangen der Amazon-SQS-Konsole.
2. Wählen Sie eine Warteschlange aus.
3. Wählen Sie auf der Detailseite der Warteschlange die Registerkarte EventBridge Pipes.

   Die Registerkarte EventBridge Pipes enthält eine Liste aller Pipes, die derzeit so konfiguriert sind, dass sie die ausgewählte Warteschlange als Quelle verwenden, darunter:

   - Name der Pipe
   - aktueller Status
   - Pipe-Ziel
   - wann die Pipe zuletzt geändert wurde

4. Falls gewünscht, können Sie weitere Pipe-Details anzeigen oder eine neue Pipe erstellen:

   - So greifen Sie auf weitere Details zu einer Pipe zu:

     Wählen Sie den Namen der Pipe aus.

     Dadurch wird die Seite mit den Pipe-Details der EventBridge Konsole geöffnet.

   - So erstellen Sie eine neue Pipe:

     Wählen Sie Amazon-SQS-Warteschlange mit Pipe verbinden.

     Dadurch wird die Seite „Pipe erstellen“ der EventBridge Konsole geöffnet, auf der die Amazon SQS SQS-Warteschlange als Pipe-Quelle angegeben ist. Weitere Informationen finden Sie unter Erstellen einer EventBridge Pipe im EventBridge Amazon-Benutzerhandbuch.

     > **Important**
     >
     > Eine Nachricht in einer Amazon-SQS-Warteschlange wird von einer einzelnen Pipe gelesen und dann nach der Verarbeitung aus der Warteschlange gelöscht, unabhängig davon, ob die Nachricht dem Filter entspricht, den Sie für diese Pipe konfiguriert haben. Gehen Sie vorsichtig vor, wenn Sie mehrere Pipes so konfigurieren, dass sie dieselbe Warteschlange als Quelle verwenden.

Verwalten großer Amazon SQS-Nachrichten mit der Extended Client Library und Amazon Simple Storage Service

Sie können die Amazon SQS Extended Client Library für Java und die Amazon SQS Extended Client Library für Python verwenden, um große Nachrichten zu senden. Dies ist besonders nützlich für die Nutzung großer Nachrichtennutzlasten von 256 KB bis zu 2 GB. Beide Bibliotheken speichern die Nachrichtennutzlast in einem Amazon Simple Storage Service-Bucket und senden die Referenz des gespeicherten Amazon S3-Objekts an die Amazon SQS-Warteschlange.

> **Note**
>
> Die Amazon SQS-Extended-Client-Bibliotheken sind sowohl mit Standard- als auch mit FIFO-Warteschlangen kompatibel.

Themen

- Verwaltung großer Amazon SQS SQS-Nachrichten mit Java und Amazon S3
- Verwaltung großer Amazon SQS SQS-Nachrichten mit Python und Amazon S3

Verwaltung großer Amazon SQS SQS-Nachrichten mit Java und Amazon S3

Sie können die Amazon SQS Extended Client Library für Java und Amazon Simple Storage Service (Amazon S3) verwenden, um große Amazon Simple Queue Service (Amazon SQS) -Nachrichten zu verwalten. Dies ist besonders nützlich, wenn Sie große Nachrichtennutzlasten von 256 KB bis zu 2 GB verarbeiten möchten. Die Bibliothek speichert die Nachrichtennutzlast in einem Amazon S3 S3-Bucket und sendet eine Nachricht mit einer Referenz auf das gespeicherte Amazon S3 S3-Objekt an eine Amazon SQS SQS-Warteschlange.

Sie können die Amazon SQS Extended Client Library für Java verwenden, um Folgendes zu tun:

- Festlegen, ob Nachrichten grundsätzlich oder ab einer Nachrichtengröße über 256 KB in Amazon S3 gespeichert werden
- Senden einer Nachricht, die auf ein einzelnes in einem S3-Bucket gespeichertes Nachrichtenobjekt verweist
- Rufen Sie das Nachrichtenobjekt aus einem Amazon S3 S3-Bucket ab
- Löschen Sie das Nachrichtenobjekt aus einem Amazon S3 S3-Bucket

Voraussetzungen

Das folgende Beispiel verwendet das AWS Java-SDK. Informationen zur Installation und Einrichtung des SDK finden Sie unter Setup the AWS SDK for Java im AWS SDK for Java Developer Guide.

Bevor Sie den Beispielcode ausführen, konfigurieren Sie Ihre AWS Anmeldeinformationen. Weitere Informationen finden Sie unter Einrichten von AWS Anmeldeinformationen und Region für die Entwicklung im AWS SDK for Java Entwicklerhandbuch.

Für das SDK für Java und die Amazon SQS Extended Client Library für Java ist das J2SE Development Kit 8.0 oder eine neuere Version erforderlich.

> **Note**
>
> Sie können die Amazon SQS Extended Client Library für Java verwenden, um Amazon-SQS-Nachrichten mithilfe von Amazon S3 nur mit der AWS SDK for Java zu verwalten. Sie können dies nicht mit der Amazon SQS SQS-Konsole AWS CLI, der Amazon SQS SQS-HTTP-API oder einem der anderen AWS SDKs tun.

AWS SDK for Java 2.x Beispiel: Verwendung von Amazon S3 zur Verwaltung großer Amazon SQS SQS-Nachrichten

Das folgende Beispiel für ein AWS SDK for Java 2.x erstellt einen Amazon S3 S3-Bucket mit einem zufälligen Namen und fügt eine Lebenszyklusregel hinzu, um Objekte nach 14 Tagen dauerhaft zu löschen. Es erstellt auch eine Warteschlange mit dem Namen `MyQueue` und sendet eine zufällige Nachricht, die in einem S3-Bucket gespeichert ist und mehr als 256 KB groß ist, an die Warteschlange. Schließlich ruft der Code die Nachricht ab, gibt Informationen über die Nachricht zurück und löscht die Nachricht, die Warteschlange und den Bucket.

```
/*
 * 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 com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
```

```
import com.amazonaws.services.sqs.model.*;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;

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

public class SQSExtendedClientExample {

// Create an Amazon S3 bucket with a random name.
private final static String S3_BUCKET_NAME = 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 AmazonS3 s3 = AmazonS3ClientBuilder.defaultClient();

    /*
     * 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 BucketLifecycleConfiguration.Rule expirationRule =
            new BucketLifecycleConfiguration.Rule();
    expirationRule.withExpirationInDays(14).withStatus("Enabled");
    final BucketLifecycleConfiguration lifecycleConfig =
            new BucketLifecycleConfiguration().withRules(expirationRule);

    // Create the bucket and allow message objects to be stored in the bucket.
    s3.createBucket(S3_BUCKET_NAME);
    s3.setBucketLifecycleConfiguration(S3_BUCKET_NAME, lifecycleConfig);
    System.out.println("Bucket created and configured.");

    /*
     * Set the Amazon SQS extended client configuration with large payload
     * support enabled.
     */
    final ExtendedClientConfiguration extendedClientConfig =
```

```
        new ExtendedClientConfiguration()
                .withLargePayloadSupportEnabled(s3, S3_BUCKET_NAME);

final AmazonSQS sqsExtended =
        new AmazonSQSExtendedClient(AmazonSQSClientBuilder
                .defaultClient(), extendedClientConfig);

/*
 * Create a long string of characters for the message object which will
 * be stored in the bucket.
 */
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().toString();
final CreateQueueRequest createQueueRequest =
        new CreateQueueRequest(QueueName);
final String myQueueUrl = sqsExtended
        .createQueue(createQueueRequest).getQueueUrl();
System.out.println("Queue created.");

// Send the message.
final SendMessageRequest myMessageRequest =
        new SendMessageRequest(myQueueUrl, myLongString);
sqsExtended.sendMessage(myMessageRequest);
System.out.println("Sent the message.");

// Receive the message.
final ReceiveMessageRequest receiveMessageRequest =
        new ReceiveMessageRequest(myQueueUrl);
List<Message> messages = sqsExtended
        .receiveMessage(receiveMessageRequest).getMessages();

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

```
    // Delete the message, the queue, and the bucket.
    final String messageReceiptHandle = messages.get(0).getReceiptHandle();
    sqsExtended.deleteMessage(new DeleteMessageRequest(myQueueUrl,
            messageReceiptHandle));
    System.out.println("Deleted the message.");

    sqsExtended.deleteQueue(new DeleteQueueRequest(myQueueUrl));
    System.out.println("Deleted the queue.");

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

private static void deleteBucketAndAllContents(AmazonS3 client) {

    ObjectListing objectListing = client.listObjects(S3_BUCKET_NAME);

    while (true) {
        for (S3ObjectSummary objectSummary : objectListing
                .getObjectSummaries()) {
            client.deleteObject(S3_BUCKET_NAME, objectSummary.getKey());
        }

        if (objectListing.isTruncated()) {
            objectListing = client.listNextBatchOfObjects(objectListing);
        } else {
            break;
        }
    }

    final VersionListing list = client.listVersions(
            new ListVersionsRequest().withBucketName(S3_BUCKET_NAME));

    for (S3VersionSummary s : list.getVersionSummaries()) {
        client.deleteVersion(S3_BUCKET_NAME, s.getKey(), s.getVersionId());
    }

    client.deleteBucket(S3_BUCKET_NAME);
}
}
```

AWS SDK for Java 2.x Beispiel: Verwendung von Amazon S3 zur Verwaltung großer Amazon SQS SQS-Nachrichten

Das folgende Beispiel für ein AWS SDK for Java 2.x erstellt einen Amazon S3 S3-Bucket mit einem zufälligen Namen und fügt eine Lebenszyklusregel hinzu, um Objekte nach 14 Tagen dauerhaft zu löschen. Es erstellt auch eine Warteschlange mit dem Namen `MyQueue` und sendet eine zufällige Nachricht, die in einem S3-Bucket gespeichert ist und mehr als 256 KB groß ist, an die Warteschlange. Schließlich ruft der Code die Nachricht ab, gibt Informationen über die Nachricht zurück und löscht die Nachricht, die Warteschlange und den Bucket.

```
/*
 * 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 S3_BUCKET_NAME = 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(S3_BUCKET_NAME).build());

s3.putBucketLifecycleConfiguration(PutBucketLifecycleConfigurationRequest.builder()
                .bucket(S3_BUCKET_NAME)
                .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, S3_BUCKET_NAME);

        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(S3_BUCKET_NAME).build());

        listObjectsResponse.contents().forEach(object -> {

client.deleteObject(DeleteObjectRequest.builder().bucket(S3_BUCKET_NAME).key(object.key()).bui
        });

        ListObjectVersionsResponse listVersionsResponse =
client.listObjectVersions(ListObjectVersionsRequest.builder().bucket(S3_BUCKET_NAME).build());

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

client.deleteObject(DeleteObjectRequest.builder().bucket(S3_BUCKET_NAME).key(version.key()).ve
```

```
        });


 client.deleteBucket(DeleteBucketRequest.builder().bucket(S3_BUCKET_NAME).build());
    }
}
```

Sie können Apache Maven verwenden, um Amazon SQS Extended Client für Ihr Java-Projekt zu konfigurieren und zu erstellen oder um das SDK selbst zu erstellen. Geben Sie einzelne Module aus dem SDK an, die Sie in Ihrer Anwendung verwenden.

```
<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>
```

Verwaltung großer Amazon SQS SQS-Nachrichten mit Python und Amazon S3

Sie können die Amazon Simple Queue Service Extended Client Library für Python und Amazon Simple Storage Service verwenden, um große Amazon SQS SQS-Nachrichten zu verwalten. Dies ist besonders nützlich, wenn Sie große Nachrichtennutzlasten von 256 KB bis zu 2 GB verarbeiten möchten. Die Bibliothek speichert die Nachrichtennutzlast in einem Amazon S3 S3-Bucket und sendet eine Nachricht mit einer Referenz auf das gespeicherte Amazon S3 S3-Objekt an eine Amazon Amazon SQS SQS-Warteschlange.

Sie können die Extended Client Library für Python verwenden, um Folgendes zu tun:

- Geben Sie an, ob Payloads immer in Amazon S3 oder nur in S3 gespeichert werden, wenn die Payloadgröße 256 KB überschreitet
- Senden Sie eine Nachricht, die auf ein einzelnes Nachrichtenobjekt verweist, das in einem Amazon S3 S3-Bucket gespeichert ist
- Rufen Sie das entsprechende Payload-Objekt aus einem Amazon S3 S3-Bucket ab
- Löschen Sie das entsprechende Payload-Objekt aus einem Amazon S3 S3-Bucket

Voraussetzungen

Im Folgenden sind die Voraussetzungen für die Verwendung der Amazon SQS Extended Client Library für Python aufgeführt:

- Ein AWS Konto mit den erforderlichen Anmeldeinformationen. Um ein AWS Konto zu erstellen, navigieren Sie zur AWS Startseite und wählen Sie dann AWS Konto erstellen. Folgen Sie den Anweisungen. Informationen zu Anmeldeinformationen finden Sie unter Anmeldeinformationen.
- Ein AWS SDK: Das Beispiel auf dieser Seite verwendet das AWS Python-SDK Boto3. Informationen zur Installation und Einrichtung des SDK finden Sie in der Dokumentation zum AWS SDK für Python im AWS SDK for Python Developer Guide
- Python 3.x (oder höher) und`pip`.
- Die Amazon SQS Extended Client Library für Python, erhältlich bei PyPI

> **Note**
>
> Sie können die Amazon SQS Extended Client Library for Python verwenden, um Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 nur mit dem AWS SDK für Python zu verwalten. Sie können dies nicht mit der AWS CLI, der Amazon SQS SQS-Konsole, der Amazon SQS SQS-HTTP-API oder einem der anderen AWS SDKs tun.

Konfigurieren der Nachrichtenspeicherung

Der Amazon SQS Extended Client verwendet die folgenden Nachrichtenattribute, um die Amazon S3 S3-Nachrichtenspeicheroptionen zu konfigurieren:

- `large_payload_support`: Der Amazon S3 S3-Bucket-Name zum Speichern großer Nachrichten.
- `always_through_s3`: Wenn`True`, dann werden alle Nachrichten in Amazon S3 gespeichert. Falls`False`, werden Nachrichten, die kleiner als 256 KB sind, nicht in den S3-Bucket serialisiert. Der Standardwert ist `False`.
- `use_legacy_attribute`: Falls `True` alle veröffentlichten Nachrichten das reservierte Nachrichtenattribut Legacy (`SQSLargePayloadSize`) anstelle des aktuellen reservierten Nachrichtenattributs (`ExtendedPayloadSize`) verwenden.

Verwaltung großer Amazon SQS SQS-Nachrichten mit der Extended Client Library für Python

Das folgende Beispiel erstellt einen Amazon S3 S3-Bucket mit einem zufälligen Namen. Anschließend erstellt es eine Amazon SQS SQS-Warteschlange mit dem Namen `MyQueue` und sendet eine Nachricht, die in einem S3-Bucket gespeichert ist und mehr als 256 KB groß ist, an die Warteschlange. Schließlich ruft der Code die Nachricht ab, gibt Informationen über die Nachricht zurück und löscht die Nachricht, die Warteschlange und den 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 = "S3_BUCKET_NAME"
```

```
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
```

Konfiguration von Amazon SQS SQS-Warteschlangen mit der Amazon SQS SQS-Konsole

Verwenden Sie die Amazon-QS-Konsole zum Konfigurieren und Verwalten von Warteschlangen und Features in Amazon Simple Queue Service (Amazon SQS). Sie können die Konsole auch verwenden, um Funktionen wie serverseitige Verschlüsselung zu konfigurieren, Ihrer Warteschlange eine Warteschlange zuzuordnen oder einen Trigger zum Aufrufen einer Funktion festzulegen. AWS Lambda

Themen

- Attributbasierte Zugriffskontrolle für Amazon SQS
- Konfiguration von Warteschlangenparametern mit der Amazon SQS SQS-Konsole
- Konfigurieren von Zugriffsrichtlinien
- Konfiguration der serverseitigen Verschlüsselung für eine Warteschlange mithilfe von SQS-verwalteten Verschlüsselungsschlüsseln
- Konfiguration der serverseitigen Verschlüsselung für eine Warteschlange mithilfe der Amazon SQS SQS-Konsole
- Konfiguration von Kostenzuweisungs-Tags für eine Warteschlange mithilfe der Amazon SQS SQS-Konsole
- Abonnieren einer Warteschlange für ein Amazon SNS SNS-Thema mithilfe der Amazon SQS SQS-Konsole
- Konfiguration einer Amazon SQS SQS-Warteschlange zum Auslösen einer Funktion AWS Lambda
- Automatisieren von Benachrichtigungen von AWS Services an Amazon SQS mithilfe von Amazon EventBridge
- Senden einer Nachricht mit Attributen

Attributbasierte Zugriffskontrolle für Amazon SQS

Was ist ABAC?

Die attributebasierte Zugriffskontrolle (ABAC) ist ein Autorisierungsprozess, bei dem Berechtigungen auf der Grundlage von Tags definiert werden, die Benutzern und Ressourcen zugewiesen sind.

AWS ABAC bietet eine detaillierte und flexible Zugriffskontrolle auf der Grundlage von Attributen und Werten, reduziert das Sicherheitsrisiko im Zusammenhang mit neu konfigurierten rollenbasierten Richtlinien und zentralisiert die Prüfung und Verwaltung von Zugriffsrichtlinien. Weitere Details zu ABAC finden Sie unter Was ist ABAC für AWS? im IAM-Benutzerhandbuch.

Amazon SQS unterstützt ABAC, indem es Ihnen ermöglicht, den Zugriff auf Ihre Amazon-SQS-Warteschlangen anhand der Tags und Aliase zu kontrollieren, die mit einer Amazon-SQS-Warteschlange verknüpft sind. Die Tag- und Alias-Bedingungsschlüssel, die ABAC in Amazon SQS aktivieren, autorisieren IAM-Prinzipale, Amazon-SQS-Warteschlangen zu verwenden, ohne Richtlinien zu bearbeiten oder Erteilungen zu verwalten.

Mit ABAC können Sie Tags verwenden, um IAM-Zugriffsberechtigungen und Richtlinien für Ihre Amazon-SQS-Warteschlangen zu konfigurieren, was Ihnen hilft, Ihr Berechtigungsmanagement zu skalieren. Sie können in IAM eine einzige Berechtigungsrichtlinie erstellen, indem Sie Tags verwenden, die Sie jeder Geschäftsrolle hinzufügen – ohne die Richtlinie jedes Mal aktualisieren zu müssen, wenn Sie eine neue Ressource hinzufügen. Sie können IAM-Prinzipalen auch Tags zuordnen, um eine ABAC-Richtlinie zu erstellen. Sie können ABAC-Richtlinien entwerfen, um Amazon-SQS-Operationen zuzulassen, wenn das Tag in der IAM-Benutzerrolle, die den Aufruf tätigt, mit dem Amazon-SQS-Warteschlangen-Tag übereinstimmt. Weitere Informationen zum Tagging finden Sie unter Tagging-Strategien AWS und.AWS Amazon-SQS-Kostenzuordnungs-Tags

> **Note**
>
> ABAC für Amazon SQS ist derzeit in allen AWS Handelsregionen verfügbar, in denen Amazon SQS verfügbar ist, mit den folgenden Ausnahmen:
>
> - Asien-Pazifik (Hyderabad)
> - Asien-Pazifik (Melbourne)
> - Europa (Spain)
> - Europa (Zürich)

Weshalb sollte ich ABAC in Amazon SQS verwenden?

Hier sind einige Vorteile der Verwendung von ABAC in Amazon SQS:

- ABAC für Amazon SQS erfordert weniger Berechtigungsrichtlinien. Sie müssen keine verschiedenen Richtlinien für verschiedene Job-Funktionen erstellen müssen. Sie können

Ressourcen- und Anforderungs-Tags verwenden, die für mehr als eine Warteschlange gelten, wodurch der Betriebsaufwand reduziert wird.

- Verwenden Sie ABAC, um Teams schnell zu skalieren. Berechtigungen für neue Ressourcen werden automatisch basierend auf Tags erteilt, wenn Ressourcen bei ihrer Erstellung entsprechend gekennzeichnet werden.
- Verwenden Sie Berechtigungen für den IAM-Prinzipal, um den Ressourcenzugriff einzuschränken. Sie können Tags für den IAM-Prinzipal erstellen und diese verwenden, um den Zugriff auf bestimmte Aktionen einzuschränken, die den Tags auf dem IAM-Prinzipal entsprechen. Auf diese Weise können Sie den Prozess der Erteilung von Anforderungsberechtigungen automatisieren.
- Verfolgen Sie, wer auf Ihre Ressourcen zugreift. Sie können die Identität einer Sitzung anhand der Benutzerattribute unter AWS CloudTrail ermitteln.

Themen

- ABAC-Bedingungsschlüssel für Amazon SQS
- Tagging für die Zugriffskontrolle in Amazon SQS
- Erstellen von IAM-Benutzern und Amazon-SQS-Warteschlangen
- Testen der attributbasierten Zugriffssteuerung

ABAC-Bedingungsschlüssel für Amazon SQS

Sie können die folgenden Bedingungsschlüsseln verwenden, um Funktionsaktionen zu steuern:

ABAC-Bedingungsschlüssel	Beschreibung	Richtlinientyp	Amazon-SQS-Vorgänge
als: ResourceTag	Das Tag (Schlüssel und Wert) auf der Amazon-SQS-Warteschlange entspricht dem Tag (Schlüssel und Wert) oder dem Tag-Muster in der Richtlinie	Nur IAM-Richtlinie	Ressourcenoperationen für Amazon-SQS-Warteschlangen

ABAC-Bedingungsschlüssel	Beschreibung	Richtlinientyp	Amazon-SQS-Vorgänge
war: RequestTag	Der Tag (Schlüssel und Wert) auf der Amazon-SQS-Warteschlangenressource entspricht dem Tag (Schlüssel und Wert) oder dem Tag-Muster in der Richtlinie	Warteschlangen- und IAM-Richtlinien	TagQueue, UntagQueue, CreateQueue
war: TagKeys	Die Tag-Schlüssel in der Anforderung entsprechen den Tag-Schlüsseln in der Richtlinie	Warteschlangen- und IAM-Richtlinien	TagQueue, UntagQueue, CreateQueue

Tagging für die Zugriffskontrolle in Amazon SQS

Im Folgenden finden Sie ein Beispiel für die Verwendung von Tags für die Zugriffssteuerung. Die IAM-Richtlinie beschränkt einen IAM-Benutzer auf alle Amazon-SQS-Aktionen für alle Warteschlangen, die ein Ressourcen-Tag mit der Schlüsselumgebung und der Wertproduktion enthalten. Weitere Informationen finden Sie unter Attributbasierte Zugriffskontrolle mit Stichwörtern und Organizations. AWS

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

```
        }
      }
    }
  ]
}
```

Erstellen von IAM-Benutzern und Amazon-SQS-Warteschlangen

In den folgenden Beispielen wird erklärt, wie eine ABAC-Richtlinie zur Steuerung des Zugriffs auf Amazon SQS mithilfe von und erstellt wird. AWS Management Console AWS CloudFormation

Mit dem AWS Management Console

Erstellen eines IAM-Benutzers

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im linken Navigationsbereich Benutzer aus.
3. Wählen Sie Benutzer hinzufügen und geben Sie einen Namen in das Textfeld Benutzername ein.
4. Wählen Sie das Feld Zugriffsschlüssel – Programmatischer Zugriff und Weiter: Berechtigungen aus.
5. Wählen Sie Weiter: Tags aus.
6. Fügen Sie den Tag-Schlüssel als `environment` und den Tag-Wert als `beta` hinzu.
7. Wählen Sie Weiter:Prüfung und dann Benutzer erstellen aus.
8. Speichern Sie Ihre Zugriffsschlüssel-ID und den geheimen Zugriffsschlüssel an einem sicheren Ort.

IAM-Benutzerberechtigungen hinzufügen

1. Wählen Sie den IAM-Benutzer aus, den Sie erstellt haben.
2. Wählen Sie Inline-Richtlinie hinzufügen.
3. Fügen Sie auf der Registerkarte JSON die folgende Richtlinie ein.

```
{
  "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"
        }
      }
    }
  ]
}
```

4. Wählen Sie Richtlinie prüfen.

5. Wählen Sie Richtlinie erstellen aus.

Verwenden AWS CloudFormation

Verwenden Sie die folgende AWS CloudFormation Beispielvorlage, um einen IAM-Benutzer mit einer angehängten Inline-Richtlinie und einer Amazon SQS SQS-Warteschlange zu erstellen:

```
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"
```

Testen der attributbasierten Zugriffssteuerung

Die folgenden Beispiele zeigen, wie Sie die attributbasierte Zugriffssteuerung in Amazon SQS testen können.

Erstellen einer Warteschlange, bei der der Tag-Schlüssel auf „environment“ und der Tag-Wert auf „prod“ gesetzt ist

Führen Sie diesen AWS CLI-Befehl aus, um zu testen, wie die Warteschlange erstellt wird, wobei der Tag-Schlüssel auf environment und der Tag-Wert auf prod gesetzt ist. Wenn Sie nicht über AWS CLI verfügen, können Sie es herunterladen und für Ihren Computer konfigurieren.

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

Sie erhalten eine `AccessDenied`-Fehlermeldung vom Amazon-SQS-Endpunkt:

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

Dies liegt daran, dass der Tag-Wert des IAM-Benutzers nicht mit dem im `CreateQueue`-API-Aufruf übergebenen Tag übereinstimmt. Denken Sie daran, dass wir dem IAM-Benutzer ein Tag zugewiesen haben, bei dem der Schlüssel auf `environment` und der Wert auf `beta` gesetzt ist.

Erstellen Sie eine Warteschlange, bei der der Tag-Schlüssel auf „Umgebung“ und der Tag-Wert auf „Beta“ gesetzt ist

Führen Sie diesen CLI-Befehl aus, um zu testen, wie eine Warteschlange erstellt wird, bei der der Tag-Schlüssel auf `environment` und der Tag-Wert auf `beta` gesetzt sind.

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

Sie erhalten eine Meldung, die die erfolgreiche Erstellung der Warteschlange bestätigt, ähnlich der folgenden.

```
{
"QueueUrl": "<queueUrl>“
}
```

Senden einer Mitteilung an eine Warteschlange

Führen Sie diesen CLI-Befehl aus, um das Senden einer Nachricht an eine Warteschlange zu testen.

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

Die Antwort zeigt eine erfolgreiche Nachrichtenzustellung an die Amazon-SQS-Warteschlange. Die IAM-Benutzerberechtigung ermöglicht Ihnen, eine Nachricht an eine Warteschlange zu senden, die über ein `beta`-Tag verfügt. Die Antwort beinhaltet `MD5OfMessageBody` und `MessageId` mit der Nachricht.

```
{
"MD5OfMessageBody": "<MD5OfMessageBody>",
"MessageId": "<MessageId>"
}
```

Konfiguration von Warteschlangenparametern mit der Amazon SQS SQS-Konsole

Wenn Sie eine Warteschlange erstellen oder bearbeiten, können Sie die folgenden Parameter konfigurieren:

- Sichtbarkeitszeitbeschränkung – Der Zeitraum, für den eine Nachricht, die aus einer Warteschlange (von einem Verbraucher) empfangen wurde, für die anderen Nachrichtenkonsumenten nicht sichtbar ist. Weitere Informationen finden Sie unter Sichtbarkeitszeitbeschränkung.

  > **Note**
  >
  > Wenn Sie die Sichtbarkeitszeitbeschränkung mithilfe der Konsole konfigurieren, wird der Timeout-Wert für alle Nachrichten in der Warteschlange konfiguriert. Um das Timeout für einzelne oder mehrere Nachrichten zu konfigurieren, müssen Sie eines der AWS SDKs verwenden.

- Aufbewahrungszeitraum für Nachrichten – Der Zeitraum, für den Amazon SQS Nachrichten aufbewahrt, die in der Warteschlange verbleiben. Standardmäßig werden Nachrichten in einer Warteschlange vier Tage lang beibehalten. Sie können eine Warteschlange so konfigurieren, dass sie Nachrichten bis zu 14 Tage aufbewahrt. Weitere Informationen finden Sie unter Aufbewahrungszeitraum für Nachrichten.
- Zustellungsverzögerung – Der Zeitraum, um den Amazon SQS die Zustellung der Nachricht verzögert, die der Warteschlange hinzugefügt wird. Weitere Informationen finden Sie unter Zustellungsverzögerung.
- Maximale Nachrichtengröße – Die maximale Nachrichtengröße für diese Warteschlange. Weitere Informationen finden Sie unter Maximale Nachrichtengröße.

- Wartezeit für den Empfang von Nachrichten – Die maximale Zeit, die Amazon SQS darauf wartet, dass Nachrichten verfügbar werden, nachdem die Warteschlange eine Empfangsanforderung erhalten hat. Weitere Informationen finden Sie unter Kurz- und Langabfragen in Amazon SQS.
- Aktivieren der inhaltsbasierten Deduplizierung – Amazon SQS kann automatisch Deduplizierungs-IDs basierend auf dem Nachrichtentext erstellen. Weitere Informationen finden Sie unter Erste Schritte mit FIFO-Warteschlangen in Amazon SQS.
- FIFO mit hohem Durchsatz aktivieren – Wird verwendet, um einen hohen Durchsatz für Nachrichten in der Warteschlange zu aktivieren. Wenn Sie diese Option wählen, werden die zugehörigen Optionen (Deduplizierungsbereich und FIFO-Durchsatz-Limit) auf die erforderlichen Einstellungen geändert, um einen hohen Durchsatz für FIFO-Warteschlangen zu aktivieren. Weitere Informationen finden Sie unter Hoher Durchsatz für FIFO-Warteschlangen in Amazon SQS und Amazon SQS SQS-Nachrichtenkontingente.
- Redrive-Richtlinie: definiert, welche Quellwarteschlangen diese Warteschlange als Warteschlange für unzustellbare Nachrichten verwenden können. Weitere Informationen finden Sie unter Verwenden von Warteschlangen für unzustellbare Briefe in Amazon SQS .

So konfigurieren Sie Warteschlangenparameter für eine vorhandene Warteschlange (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus. Wählen Sie eine Warteschlange aus und klicken Sie auf Bearbeiten.
3. Scrollen Sie zum Abschnitt Konfiguration.
4. Geben Sie für Sichtbarkeitszeitbeschränkung die Dauer und die Einheiten ein. Der Bereich liegt zwischen 0 Sekunden und 12 Stunden. Der Standardwert ist 30 Sekunden.
5. Geben Sie unter Aufbewahrungszeitraum für Nachrichten die Dauer und die Einheiten ein. Der gültige Bereich beträgt 1 Minute bis 14 Tage. Der Standardwert ist 4 Tage.
6. Geben Sie für eine Standard-Warteschlange einen Wert für die Wartezeit für den Empfang von Nachrichten ein. Der Bereich liegt zwischen 0 und 20 Sekunden. Der Standardwert ist 0 Sekunden, der kurze Abfragen festlegt. Jeder Wert ungleich Null führt zu einer langen Abfrage.
7. Geben Sie für Zustellungsverzögerung die Dauer und die Einheiten ein. Der Bereich liegt zwischen 0 Sekunden und 15 Minuten. Der Standardwert ist 0 Sekunden.
8. Geben Sie für Maximale Nachrichtengröße einen Wert ein. Der Bereich reicht von 1 KB bis 256 KB. Der Standardwert ist 256 KB.

9. Wählen Sie für eine FIFO-Warteschlange Aktivieren der inhaltsbasierten Deduplizierung, um die inhaltsbasierte Deduplizierung zu aktivieren. Sie Standardeinstellung ist deaktiviert.
10. (Optional) Damit eine FIFO-Warteschlange einen höheren Durchsatz für das Senden und Empfangen von Nachrichten in der Warteschlange ermöglicht, wählen Sie FIFO mit hohem Durchsatz aktivieren.

    Wenn Sie diese Option wählen, werden die zugehörigen Optionen (Deduplizierungsbereich und FIFO-Durchsatz-Limit) auf die erforderlichen Einstellungen geändert, um einen hohen Durchsatz für FIFO-Warteschlangen zu aktivieren. Wenn Sie Einstellungen ändern, die für die Verwendung von FIFO mit hohem Durchsatz erforderlich sind, ist der normale Durchsatz für die Warteschlange wirksam und die Deduplizierung erfolgt wie angegeben. Weitere Informationen finden Sie unter Hoher Durchsatz für FIFO-Warteschlangen in Amazon SQS und Amazon SQS SQS-Nachrichtenkontingente.
11. Wählen Sie für die Redrive-Zulassungsrichtlinie die Option Aktiviert aus. Wählen Sie eine der folgenden Optionen aus: Alle zulassen (Standard), Nach Warteschlange oder Alle verweigern. Wenn Sie Nach Warteschlange wählen, geben Sie eine Liste mit bis zu 10 Quellwarteschlangen nach dem Amazon-Ressourcennamen (ARN) an.
12. Wenn Sie mit der Konfiguration der Warteschlangenparameter fertig sind, wählen Sie Speichern.

Konfigurieren von Zugriffsrichtlinien

Wenn Sie eine Warteschlange bearbeiten, können Sie ihre Zugriffsrichtlinie konfigurieren.

Die Zugriffsrichtlinie definiert die Konten, Benutzer und Rollen, die auf die Warteschlange zugreifen können. Die Zugriffsrichtlinie definiert auch die Aktionen (wie `SendMessage`, `ReceiveMessage` oder `DeleteMessage`), auf die die Benutzer zugreifen können. Die Standardrichtlinie erlaubt nur dem Eigentümer der Warteschlange, Nachrichten zu senden und zu empfangen.

So konfigurieren Sie die Zugriffsrichtlinie für eine vorhandene Warteschlange (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie eine Warteschlange aus und klicken Sie auf Bearbeiten.
4. Scrollen Sie zum Abschnitt Zugriffsrichtlinie.

5. Bearbeiten Sie die Anweisungen zur Zugriffsrichtlinie im Eingabefeld. Weitere Informationen zu den Anweisungen zur Zugriffsrichtlinie finden Sie unter Identity and Access Management in Amazon SQS.
6. Wenn Sie mit der Konfiguration der Zugriffsrichtlinie fertig sind, wählen Sie Speichern.

Konfiguration der serverseitigen Verschlüsselung für eine Warteschlange mithilfe von SQS-verwalteten Verschlüsselungsschlüsseln

Zusätzlich zur Standardoption für verwaltete serverseitige Verschlüsselung (SSE) von Amazon SQS können Sie mit Amazon SQS Managed SSE (SSE-SQS) eine benutzerdefinierte verwaltete serverseitige Verschlüsselung erstellen, die von SQS verwaltete Verschlüsselungsschlüssel verwendet, um sensible Daten zu schützen, die über Nachrichtenwarteschlangen gesendet werden. Mit SSE-SQS müssen Sie keine Verschlüsselungsschlüssel erstellen und verwalten oder Ihren Code ändern, um Ihre Daten zu verschlüsseln. SSE-SQS ermöglicht Ihnen die sichere Übertragung von Daten und hilft Ihnen dabei, strenge Verschlüsselungsvorschriften und gesetzliche Anforderungen ohne zusätzliche Kosten zu erfüllen.

SSE-SQS schützt Daten im Ruhezustand mit 256-Bit Advanced Encryption Standard (AES-256). SSE verschlüsselt Nachrichten, sobald sie bei Amazon SQS eingehen. Amazon SQS speichert Nachrichten in verschlüsselter Form und entschlüsselt sie nur, wenn sie an einen autorisierten Verbraucher gesendet werden.

> **Note**
>
> - Die SSE-Standardoption ist nur wirksam, wenn Sie eine Warteschlange ohne Angabe von Verschlüsselungsattributen erstellen.
> - Mit Amazon SQS können Sie die gesamte Warteschlangenverschlüsselung ausschalten. Wenn Sie KMS-SSE ausschalten, wird SQS-SSE daher nicht automatisch aktiviert. Wenn Sie SQS-SSE nach dem Ausschalten von KMS-SSE aktivieren möchten, müssen Sie der Anfrage eine Attributänderung hinzufügen.

So konfigurieren Sie die SSE-SQS-Verschlüsselung für eine Warteschlange (Konsole)

> **Note**
>
> Jede neue Warteschlange, die mit dem HTTP-Endpunkt (ohne TLS) erstellt wurde, aktiviert standardmäßig keine SSE-SQS-Verschlüsselung. Es ist eine bewährte Sicherheitsmethode, Amazon-SQS-Warteschlangen mit HTTPS- oder Signature Version 4-Endpunkten zu erstellen.

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie eine Warteschlange und anschließend Bearbeiten aus.
4. Erweitern Sie Verschlüsselung.
5. Wählen Sie unter Serverseitige Verschlüsselung Aktivieren aus.

   > **Note**
   >
   > Wenn SSE aktiviert ist, werden anonyme `SendMessage`- und `ReceiveMessage`-Anfragen an die verschlüsselte Warteschlange abgewiesen. Die bewährten Sicherheitsmethoden von Amazon SQS raten davon ab, anonyme Anfragen zu verwenden. Wenn Sie anonyme Anfragen an eine Amazon-SQS-Warteschlange senden möchten, stellen Sie sicher, dass SSE deaktiviert ist.

6. Wählen Sie Amazon-SQS-Schlüssel (SSE-SQS) aus. Für die Nutzung dieser Option fallen keine zusätzlichen Kosten an.
7. Klicken Sie auf Speichern.

Konfiguration der serverseitigen Verschlüsselung für eine Warteschlange mithilfe der Amazon SQS SQS-Konsole

Um die Daten in den Nachrichten einer Warteschlange zu schützen, hat Amazon SQS die serverseitige Verschlüsselung (SSE) standardmäßig für alle neu erstellten Warteschlangen aktiviert. Amazon SQS ist in den Amazon Web Services Key Management Service (Amazon Web Services KMS) integriert, um KMS-Schlüssel für die serverseitige Verschlüsselung (SSE) zu verwalten. Für weitere Informationen zur Nutzung von SSE siehe Verschlüsselung im Ruhezustand in Amazon SQS.

Für den KMS-Schlüssel, den Sie Ihrer Warteschlange zuweisen, muss eine Schlüsselrichtlinie gelten, die Berechtigungen für alle Prinzipale beinhaltet, die zur Nutzung der Warteschlange berechtigt sind. Informationen finden Sie unter Schlüsselverwaltung.

Wenn Sie nicht der Besitzer des KMS-Schlüssels sind oder wenn Sie sich mit einem Konto anmelden, das über keine `kms:ListAliases`- und `kms:DescribeKey`-Berechtigungen verfügt, können Sie auf der Amazon-SQS-Konsole keine Informationen über den KMS aufrufen. Bitten Sie den Inhaber des KMS, Ihnen diese Berechtigungen zu erteilen. Weitere Informationen finden Sie unter Schlüsselverwaltung.

Wenn Sie eine Warteschlange erstellen oder bearbeiten, können Sie SSE-KMS konfigurieren.

So konfigurieren Sie SSE-KMS für eine vorhandene Warteschlange (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie eine Warteschlange und anschließend Bearbeiten aus.
4. Erweitern Sie Verschlüsselung.
5. Wählen Sie unter Serverseitige Verschlüsselung Aktivieren aus.

   > **Note**
   >
   > Wenn SSE aktiviert ist, werden anonyme `SendMessage`- und `ReceiveMessage`-Anfragen an die verschlüsselte Warteschlange abgewiesen. Die bewährten Sicherheitsmethoden von Amazon SQS raten davon ab, anonyme Anfragen zu verwenden. Wenn Sie anonyme Anfragen an eine Amazon-SQS-Warteschlange senden möchten, stellen Sie sicher, dass SSE deaktiviert ist.

6. Wählen Sie AWS Key Management Service-Schlüssel (SSE-KMS) aus.

   In der Konsole werden die Beschreibung, das Konto und der KMS-Schlüssel-ARN des KMS-Schlüssels angezeigt.

7. Geben Sie die KMS-Schlüssel-ID für die Warteschlange an. Weitere Informationen finden Sie unter Wichtige Begriffe.

   a. Wählen Sie die Option KMS-Schlüsselalias auswählen.

b. Der Standardschlüssel ist der von Amazon Web Services verwaltete KMS-Schlüssel für Amazon SQS. Um diesen Schlüssel zu verwenden, wählen Sie ihn aus der KMS-Schlüsselliste aus.

c. Um einen benutzerdefinierten KMS-Schlüssel aus Ihrem Amazon-Web-Services-Konto zu verwenden, wählen Sie ihn aus der KMS-Schlüsselliste aus. Anweisungen zum Erstellen benutzerdefinierter KMS-Schlüssel finden Sie unter Erstellen von Schlüsseln im Amazon-Web-Services-Key-Management-Service-Entwicklerhandbuch.

d. Um einen benutzerdefinierten KMS-Schlüssel, der nicht in der Liste enthalten ist, oder einen benutzerdefinierten KMS-Schlüssel von einem anderen Amazon-Web-Services-Konto zu verwenden, wählen Sie KMS-Schlüsselalias eingeben aus und geben Sie den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels ein.

8. (Optional) Geben Sie für den Zeitraum der Wiederverwendung von Datenschlüsseln einen Wert zwischen 1 Minute und 24 Stunden an. Der Standardwert ist 5 Minuten. Weitere Informationen finden Sie unter Grundlegendes zum Wiederverwendungszeitraum für den Datenschlüssel.
9. Wenn Sie mit der Konfiguration von SSE-KMS fertig sind, wählen Sie Speichern.

Konfiguration von Kostenzuweisungs-Tags für eine Warteschlange mithilfe der Amazon SQS SQS-Konsole

Sie können Ihren Amazon-SQS-Warteschlangen Kostenzuordnungs-Tags hinzufügen, um sie zu organisieren und zu identifizieren. Weitere Informationen finden Sie unter Amazon-SQS-Kostenzuordnungs-Tags.

Auf der Seite Details für eine Warteschlange werden auf der Registerkarte Tagging die Tags für die Warteschlange angezeigt.

Wenn Sie eine Warteschlange erstellen oder bearbeiten, können Sie Tags dafür konfigurieren.

So konfigurieren Sie Tags für eine vorhandene Warteschlange (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie eine Warteschlange aus und klicken Sie auf Bearbeiten.
4. Scrollen Sie zum Abschnitt Tags.
5. Fügen Sie die Warteschlangen-Tags hinzu, ändern oder entfernen Sie sie:

a. Um einen Tag hinzuzufügen, wählen Sie Neuen Tag hinzufügen, geben Sie einen Schlüssel und einen Wert ein und wählen Sie dann Änderungen anwenden.

b. Um einen Tag zu aktualisieren, ändern Sie dessen Schlüssel und Wert.

c. Wählen Sie zum Entfernen eines Tags neben einem Schlüssel-Wert-Paar Tag entfernen aus.

6. Wenn Sie mit der Konfiguration der Tags fertig sind, wählen Sie Speichern.

Abonnieren einer Warteschlange für ein Amazon SNS SNS-Thema mithilfe der Amazon SQS SQS-Konsole

Sie können eine oder mehrere Amazon-SQS-Warteschlangen für ein Amazon Simple Notification Service (Amazon SNS)-Thema abonnieren. Wenn Sie eine Nachricht in einem Thema veröffentlichen, sendet Amazon SNS die Nachricht an alle abonnierten Warteschlangen. Amazon SQS verwaltet das Abonnement und alle erforderlichen Berechtigungen. Weitere Informationen zu Amazon SNS finden Sie unter Was ist Amazon SNS? im Amazon-Simple-Notification-Service-Entwicklerhandbuch.

Wenn Sie eine Amazon-SQS-Warteschlange für ein SNS-Thema abonnieren, verwendet Amazon SNS HTTPS, um Nachrichten an Amazon SQS weiterzuleiten. Weitere Informationen zur Verwendung von Amazon SNS mit verschlüsselten Amazon-SQS-Warteschlangen finden Sie unter Konfigurieren Sie KMS-Berechtigungen für Dienste AWS.

> **Important**
>
> Amazon SQS unterstützt maximal 20 Anweisungen pro Zugriffsrichtlinie. Das Abonnieren eines Amazon-SNS-Themas fügt eine solche Anweisung hinzu. Eine Überschreitung dieser Zahl führt dazu, dass das Abonnement für das Thema nicht zugestellt werden kann.

So abonnieren Sie eine Warteschlange für ein SNS-Thema (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie in der Liste der Warteschlangen die Warteschlange aus, für die das Amazon-SNS-Thema abonniert werden soll.

4. Wählen Sie im Menü Actions (Aktionen) die Option Subscribe to Amazon SNS topic (Amazon-SNS-Thema abonnieren) aus.
5. Wählen Sie im Menü Angeben eines für dieses Warteschlangenmenü verfügbaren Amazon-SNS-Themas das SNS-Thema für Ihre Warteschlange aus.

   Wenn das SNS-Thema in der Liste nicht aufgeführt ist, wählen Sie Amazon-SNS-Thema-ARN eingeben aus und geben Sie dann den Amazon-Ressourcennamen (ARN) des Themas ein.
6. Wählen Sie Speichern.
7. Überprüfen Sie das Ergebnis des Abonnements, indem Sie eine Nachricht im Thema veröffentlichen und dann die Nachricht überprüfen, die das Thema an die Warteschlange sendet. Weitere Informationen finden Sie unter Veröffentlichen von Amazon-SNS-Nachrichten im Amazon-Simple-Notification-Service-Entwicklerhandbuch.

Wenn sich Ihre Amazon SQS SQS-Warteschlange und Ihr SNS-Thema unterscheiden AWS-Konten, muss der Eigentümer des Themas zuerst das Abonnement bestätigen. Weitere Informationen finden Sie unter bestätigen des Abonnements im Amazon-Simple-Notification-Service-Entwicklerhandbuch.

Informationen zum Abonnieren eines regionsübergreifenden SNS-Themas finden Sie unter Senden von Amazon SNS SNS-Nachrichten an eine Amazon SQS SQS-Warteschlange oder AWS Lambda - Funktion in einer anderen Region im Amazon Simple Notification Service Developer Guide

Konfiguration einer Amazon SQS SQS-Warteschlange zum Auslösen einer Funktion AWS Lambda

Sie können eine AWS Lambda Funktion verwenden, um Nachrichten in einer Amazon SQS SQS-Warteschlange zu verarbeiten. Lambda fragt die Warteschlange ab und ruft Ihre Lambda-Funktion synchron mit einem Ereignis auf, das Warteschlangennachrichten enthält. Damit die Funktion Zeit hat, jeden Batch von Datensätzen zu verarbeiten, legen Sie die Zeitbeschränkung für die Sichtbarkeit der Ausgangswarteschlange auf mindestens das Sechsfache der Zeitbeschränkung fest, die Sie für Ihre Funktion konfigurieren. Diese zusätzliche Zeit ermöglicht es Lambda, einen erneuten Versuch zu machen, wenn die Funktion gedrosselt wird, während ein früherer Batch verarbeitet wird.

Sie können eine andere Warteschlange angeben, die als Warteschlange für unzustellbare Nachrichten dient, die Ihre Lambda-Funktion nicht verarbeiten kann.

Eine Lambda-Funktion kann Elemente aus mehreren Warteschlangen verarbeiten (eine Lambda-Ereignisquelle für jede Warteschlange). Sie können dieselbe Warteschlange mit mehreren Lambda-Funktionen verwenden.

Wenn Sie eine verschlüsselte Warteschlange mit einer Lambda-Funktion verknüpfen, Lambda aber keine Nachrichten abfragt, fügen Sie die `kms:Decrypt`-Berechtigung zu Ihrer Lambda-Ausführungsrolle hinzu.

Beachten Sie die folgenden Einschränkungen:

- Ihre Warteschlange und die Lambda-Funktion müssen sich in derselben AWS Region befinden.
- Eine verschlüsselte Warteschlange, die den Standardschlüssel (AWS verwalteter KMS-Schlüssel für Amazon SQS) verwendet, kann keine Lambda-Funktion in einer anderen aufrufen. AWS-Konto

Informationen zur Implementierung der Lambda-Funktion finden Sie unter Using AWS Lambda with Amazon SQS im AWS Lambda Developer Guide.

Voraussetzungen

Um Lambda-Funktions-Auslöser zu konfigurieren, müssen Sie die folgenden Anforderungen erfüllen:

- Wenn Sie einen Benutzer verwenden, muss Ihre Amazon-SQS-Rolle die folgenden Berechtigungen einschließen:
  - `lambda:CreateEventSourceMapping`
  - `lambda:ListEventSourceMappings`
  - `lambda:ListFunctions`
- Die Lambda-Ausführungsrolle muss die folgenden Berechtigungen enthalten:
  - `sqs:DeleteMessage`
  - `sqs:GetQueueAttributes`
  - `sqs:ReceiveMessage`
- Wenn Sie eine verschlüsselte Warteschlange mit einer Lambda-Funktion verknüpfen, fügen Sie die `kms:Decrypt`-Berechtigung zur Lambda-Ausführungsrolle hinzu.

Weitere Informationen finden Sie unter Übersicht über die Zugriffsverwaltung in Amazon SQS.

So konfigurieren Sie eine Warteschlange, um eine Lambda-Funktion auszulösen (Konsole)

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie auf der Seite Warteschlange die zu konfigurierende Warteschlange aus.
4. Wählen Sie auf der Seite der Warteschlange die Registerkarte Lambda-Auslöser aus.
5. Wählen Sie auf der Seite Lambda-Auslöser einen Lambda-Auslöser aus.

   Wenn die Liste den benötigten Lambda-Auslöser nicht enthält, wählen Sie Lambda-Funktions-Auslöser konfigurieren. Geben Sie den Amazon-Ressourcennamen (ARN) der Lambda-Funktion ein oder wählen Sie eine vorhandene Ressource aus. Wählen Sie dann Speichern.
6. Wählen Sie Speichern. In der Konsole wird Konfiguration gespeichert und die Seite Details für die Warteschlange angezeigt.

   Auf der Seite Details werden auf der Registerkarte Lambda-Auslöser die Lambda-Funktion und ihr Status angezeigt. Es dauert etwa 1 Minute, bis die Lambda-Funktion der Warteschlange zugeordnet wird.
7. Zum Überprüfen der Ergebnisse der Konfiguration können Sie eine Nachricht an Ihre Warteschlange senden und dann die ausgelöste Lambda-Funktion in der Lambda-Konsole anzeigen.

Automatisieren von Benachrichtigungen von AWS Services an Amazon SQS mithilfe von Amazon EventBridge

EventBridge Mit Amazon können Sie AWS Services automatisieren und auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen reagieren. Ereignisse aus AWS Services werden EventBridge fast in Echtzeit übermittelt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind und welche automatisierten Aktionen durchgeführt werden sollen, wenn sich für ein Ereignis eine Übereinstimmung mit einer Regel ergibt.

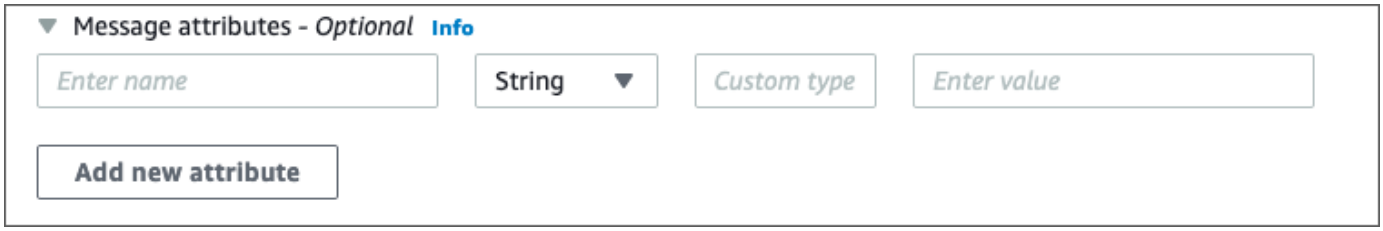
EventBridge ermöglicht es Ihnen, eine Vielzahl von Zielen — wie Amazon SQS SQS-Standard- und FIFO-Warteschlangen — festzulegen, die Ereignisse im JSON-Format empfangen. Weitere Informationen finden Sie unter EventBridgeAmazon-Ziele im EventBridge Amazon-Benutzerhandbuch.

Senden einer Nachricht mit Attributen

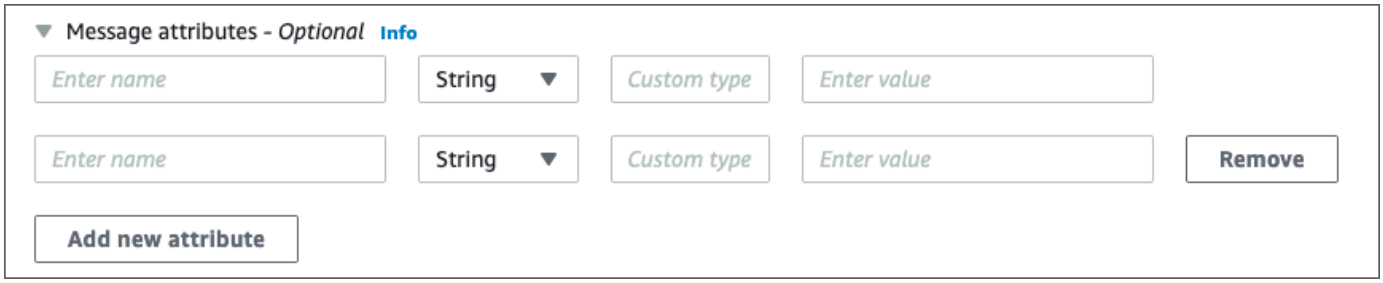
Für Standard- und FIFO-Warteschlangen können Sie strukturierte Metadaten (wie etwa Zeitstempel, geospatiale Daten, Signaturen und Kennungen) in Nachrichten einschließen. Weitere Informationen finden Sie unter Amazon-SQS-Nachrichtenattribute.

Um mit der Amazon SQS SQS-Konsole eine Nachricht mit Attributen an eine Warteschlange zu senden

1. Öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie auf der Seite Warteschlangen eine Warteschlange aus.
4. Wählen Sie Nachrichten senden und empfangen.
5. Geben Sie die Nachrichtenattributparameter ein.
   a. Geben Sie im Textfeld „Name“ einen eindeutigen Namen mit bis zu 256 Zeichen ein.
   b. Wählen Sie für den Attributtyp Zeichenfolge, Zahl oder Binär.
   c. (Optional) Geben Sie einen benutzerdefinierten Datentyp ein. Sie könnten beispielsweise **`byte`**, **`int`** oder **`float`** als benutzerdefinierte Datentypen für Zahl hinzufügen.
   d. Geben Sie den Wert des Nachrichtenattributs in das Textfeld ein.



6. Klicken Sie auf Attribut hinzufügen, um ein weiteres Nachrichtenattribut hinzuzufügen.



7. Sie können die Attributwerte jederzeit ändern, bevor die Nachricht gesendet wird.

8. Um ein Attribut zu löschen, wählen Sie Entfernen. Um das erste Attribut zu löschen, schließen Sie Nachrichtenattribute.
9. Wenn Sie mit dem Hinzufügen von Attributen zur Nachricht fertig sind, wählen Sie Nachricht senden. Wenn Ihre Nachricht gesendet wurde, zeigt die Konsole eine Erfolgsmeldung an. Um Informationen zu den Nachrichtenattributen der gesendeten Nachricht anzuzeigen, wählen Sie Details anzeigen. Wählen Sie Fertig, um das Dialogfeld mit den Nachrichtendetails zu schließen.

Bewährte Methoden für Amazon SQS

Diese bewährten Methoden können Sie dabei unterstützen, Amazon SQS optimal zu nutzen.

Themen

- Empfehlungen für Amazon-SQS-Standard- und FIFO-Warteschlangen
- Zusätzliche Empfehlungen für Amazon-SQS-FIFO-Warteschlangen

Empfehlungen für Amazon-SQS-Standard- und FIFO-Warteschlangen

Die folgenden bewährten Methoden können Ihnen dabei helfen, mit Amazon SQS Kosten zu reduzieren und Nachrichten effizient zu verarbeiten.

Themen

- Arbeiten mit Amazon-SQS-Nachrichten
- Senkung der Kosten für Amazon SQS
- Wechseln von einer Amazon-SQS-Standard- zu einer FIFO-Warteschlange

Arbeiten mit Amazon-SQS-Nachrichten

Die folgenden Richtlinien können Ihnen dabei helfen, mit Amazon SQS Nachrichten effizient zu verarbeiten.

Themen

- Zeitnahe Verarbeitung von Nachrichten
- Umgang mit Anforderungsfehlern
- Einrichten von Langabfragen
- Erfassung problematischer Nachrichten
- Die Aufbewahrung der Warteschleife für unzustellbare Briefe einrichten
- Vermeiden einer inkonsistenten Nachrichtenverarbeitung
- Implementieren von Request-Response-Systemen

Zeitnahe Verarbeitung von Nachrichten

Die Einstellung der Zeitbeschränkung für die Sichtbarkeit hängt davon ab, wie lange Ihre Anwendung braucht, um eine Nachricht zu verarbeiten und zu löschen. Benötigt Ihre Anwendung beispielsweise 10 Sekunden für die Verarbeitung einer Nachricht und Sie setzen die Zeitbeschränkung für die Sichtbarkeit auf 15 Minuten, müssen Sie relativ lange warten, bis Sie erneut versuchen können, die Nachricht zu verarbeiten, wenn der vorherige Versuch einer Verarbeitung fehlgeschlagen ist. Benötigt Ihre Anwendung alternativ 10 Sekunden für die Verarbeitung einer Nachricht, Sie setzen die Zeitbeschränkung für die Sichtbarkeit jedoch auf nur 2 Sekunden, wird eine duplizierte Nachricht von einem anderen Verbraucher empfangen, während der ursprüngliche Verbraucher die Nachricht noch verarbeitet.

Um sicherzustellen, dass ausreichend Zeit für die Verarbeitung von Nachrichten vorhanden ist, nutzen Sie eine der folgenden Strategien:

- Wenn Sie wissen (oder annähernd schätzen können), wie lange die Verarbeitung einer Nachricht dauert, erweitern Sie die Zeitbeschränkung für die Sichtbarkeit auf die maximale Zeit, die erforderlich ist, um die Nachricht zu verarbeiten und zu löschen. Weitere Informationen finden Sie unter Konfiguration der Sichtbarkeitszeitbeschränkung.
- Wenn Sie nicht wissen, wie lange die Bearbeitung einer Nachricht dauert, erstellen Sie einen Heartbeat für Ihren Kundenprozess: Geben Sie das anfängliche Timeout für die Sichtbarkeit an (z. B. 2 Minuten) und verlängern Sie dann, solange Ihr Kunde noch an der Nachricht arbeitet, die Sichtbarkeitszeitbeschränkung jede Minute um 2 Minuten.

> **Important**
>
> Die maximale Zeitbeschränkung für die Sichtbarkeit beträgt 12 Stunden ab dem Zeitpunkt, an dem Amazon SQS die `ReceiveMessage`-Anforderung empfängt. Durch die Verlängerung der Sichtbarkeitszeitbeschränkung wird das Maximum von 12 Stunden nicht zurückgesetzt.
> Darüber hinaus können Sie das Timeout für eine einzelne Nachricht möglicherweise nicht auf die vollen 12 Stunden (z. B. 43 200 Sekunden) festlegen, seit die `ReceiveMessage`-Anfrage den Timer initiiert hat. Wenn Sie beispielsweise eine Nachricht erhalten und sofort das Maximum von 12 Stunden festlegen, indem Sie einen `ChangeMessageVisibility`-Aufruf mit einem `VisibilityTimeout` von 43 200 Sekunden senden, schlägt der Vorgang wahrscheinlich fehl. Die Verwendung eines Werts von 43 195 Sekunden funktioniert jedoch, sofern es nicht zu einer erheblichen Verzögerung zwischen dem Anfordern der Nachricht über `ReceiveMessage` und der Aktualisierung der

Sichtbarkeitszeitbeschränkung kommt. Wenn Ihr Verbraucher länger als 12 Stunden benötigt, sollten Sie die Verwendung von Step Functions in Betracht ziehen.

Umgang mit Anforderungsfehlern

Für den Umgang mit Anforderungsfehlern nutzen Sie eine der folgenden Strategien:

- Wenn Sie ein AWS SDK verwenden, steht Ihnen bereits eine automatische Wiederholungs- und Backoff-Logik zur Verfügung. Weitere Informationen finden Sie unter Wiederholversuche bei Fehlern und exponentielles Backoff in AWS im Allgemeine Amazon Web Services-Referenz.
- Wenn Sie die AWS SDK-Funktionen für Wiederholungen und Backoffs nicht verwenden, warten Sie eine Pause (z. B. 200 ms), bevor Sie die ReceiveMessageAktion erneut versuchen, nachdem Sie keine Nachrichten, eine Zeitüberschreitung oder eine Fehlermeldung von Amazon SQS erhalten haben. Für die nachfolgende Verwendung der Aktion `ReceiveMessage`, mit der dieselben Ergebnisse erzielt werden, lassen Sie eine längere Pause (z. B. 400 ms) zu.

Einrichten von Langabfragen

Ist die Wartezeit für die `ReceiveMessage`-API-Aktion größer als 0, ist eine lange Abfrage wirksam. Die maximale Wartezeit für lange Abfragen beträgt 20 Sekunden. Mithilfe von Langabfragen können Sie die Kosten für die Verwendung von Amazon SQS reduzieren, indem Sie die Anzahl der leeren Antworten (wenn bei einer `ReceiveMessage`-Anfrage keine Nachrichten vorliegen) und fälschlicherweise leeren Antworten (wenn Nachrichten vorliegen, diese aber nicht in einer Antwort enthalten sind) eliminieren. Weitere Informationen finden Sie unter Kurz- und Langabfragen in Amazon SQS.

Um eine optimale Nachrichtenverarbeitung sicherzustellen, wenden Sie die folgenden Strategien an:

- In den meisten Fällen können Sie die `ReceiveMessage`-Wartezeit auf 20 Sekunden setzen. Wenn 20 Sekunden für Ihre Anwendung zu lang ist, legen Sie eine kürzere `ReceiveMessage`-Wartezeit fest (mindestens 1 Sekunde). Wenn Sie kein AWS SDK für den Zugriff auf Amazon SQS verwenden oder wenn Sie ein AWS SDK für eine kürzere Wartezeit konfigurieren, müssen Sie Ihren Amazon SQS SQS-Client möglicherweise ändern, um entweder längere Anfragen zuzulassen oder eine kürzere Wartezeit für lange Abfragen zu verwenden.
- Wenn Sie Langabfragen für mehrere Warteschlangen implementieren, verwenden Sie einen Thread für jede Warteschlange statt eines einzelnen Threads für alle Warteschlangen. Durch

die Verwendung eines einzelnen Threads für jede Warteschlange kann Ihre Anwendung die Nachrichten in jeder der Warteschlangen verarbeiten, sobald diese verfügbar sind, während bei Verwendung eines einzelnen Threads für die Abfrage mehrerer Warteschlangen dazu führen könnte, dass Ihre Anwendung die Nachrichten nicht verarbeiten kann, die in anderen Warteschlangen zur Verfügung stehen, während die Anwendung auf die Warteschlange wartet (bis zu 20 Sekunden), die keine verfügbaren Nachrichten enthält.

> **Important**
>
> Um HTTP-Fehler zu vermeiden, stellen Sie sicher, dass das HTTP-Reaktions-Timeout für `ReceiveMessage`-Anforderungen länger ist als der `WaitTimeSeconds`-Parameter. Weitere Informationen finden Sie unter. ReceiveMessage

Erfassung problematischer Nachrichten

Um alle Nachrichten zu erfassen, die nicht verarbeitet werden können, und um genaue CloudWatch Messwerte zu sammeln, konfigurieren Sie eine Warteschlange für unzustellbare Nachrichten.

- Die Redrive-Richtlinie leitet Nachrichten an eine Warteschlange für unzustellbare Nachrichten weiter, nachdem die Quell-Warteschlange eine Nachricht mehrfach nicht verarbeiten kann.
- Durch Verwenden einer Warteschlange für unzustellbare Nachrichten wird die Anzahl der Nachrichten reduziert und das Risiko von Poison-Pill-Nachrichten gesenkt, d. h. Nachrichten, die empfangen, aber nicht verarbeitet werden können.
- Die Aufnahme einer Giftpillen-Nachricht in eine Warteschlange kann die `ApproximateAgeOfOldestMessage` CloudWatch Metrik verfälschen, da das falsche Alter der Giftpillen-Nachricht angegeben wird. Das Konfigurieren einer Warteschlange für unzustellbare Nachrichten hilft, Fehlalarme bei der Verwendung dieser Metrik zu vermeiden.

Die Aufbewahrung der Warteschleife für unzustellbare Briefe einrichten

Bei Standard-Warteschlangen basiert der Ablauf einer Nachricht immer auf dem ursprünglichen Enqueue-Zeitstempel. Wenn eine Nachricht in eine Warteschlange für unzustellbare Nachrichten verschoben wird, bleibt der Enqueue-Zeitstempel unverändert. Die `ApproximateAgeOfOldestMessage`-Metrik gibt an, wann die Nachricht in die Warteschlange für unzustellbare Nachrichten verschoben wurde, und nicht, wann die Nachricht ursprünglich gesendet

wurde. Nehmen wir beispielsweise an, dass sich eine Nachricht einen Tag in der ursprünglichen Warteschlange befindet, bevor sie in eine Warteschlange für unzustellbare Nachrichten verschoben wird. Wenn die Aufbewahrungsfrist der Warteschlange für unzustellbare Nachrichten 4 Tage beträgt, wird die Nachricht nach 3 Tagen aus der Warteschlange für unzustellbare Nachrichten gelöscht und das `ApproximateAgeOfOldestMessage` ist 3 Tage. Es hat sich daher bewährt, die Aufbewahrungsdauer einer Warteschlange für unzustellbare Nachrichten immer so festzulegen, dass sie länger ist als die Aufbewahrungsdauer der ursprünglichen Warteschlange.

Bei FIFO-Warteschlangen wird der Enqueue-Zeitstempel zurückgesetzt, wenn die Nachricht in eine Warteschlange für unzustellbare Nachrichten verschoben wird. Die `ApproximateAgeOfOldestMessage`-Metrik gibt an, wann die Nachricht in die Warteschlange für unzustellbare Nachrichten verschoben wird. Im obigen Beispiel wird die Nachricht nach 4 Tagen aus der Warteschlange für unzustellbare Nachrichten gelöscht und das `ApproximateAgeOfOldestMessage` ist 4 Tage.

Vermeiden einer inkonsistenten Nachrichtenverarbeitung

Da es sich bei Amazon SQS um ein verteiltes System handelt, ist es möglich, dass ein Konsument selbst dann keine Nachricht empfängt, wenn Amazon SQS die Nachricht als übermittelt markiert, während sie erfolgreich von einem `ReceiveMessage`-API-Methodenaufruf zurückgegeben wird. In diesem Fall vermerkt Amazon SQS die Nachricht mindestens einmal als zugestellt, obwohl der Konsument sie nie erhalten hat. Da unter diesen Bedingungen keine zusätzlichen Versuche zur Zustellung von Nachrichten unternommen werden, empfehlen wir nicht, als Anzahl der maximalen Eingänge für eine Warteschlange für unzustellbare Nachrichten „1“ einzustellen.

Implementieren von Request-Response-Systemen

Beachten Sie beim Implementieren eines Request-Response- oder Remoteprozeduraufrufs (RPC)-Systems die folgenden bewährten Vorgehensweisen:

- Erstellen Sie keine Antwortwarteschlangen pro Nachricht. Erstellen Sie stattdessen beim Start pro Hersteller Antwortwarteschlangen und verwenden Sie ein Korrelations-ID-Nachrichtenattribut, um Antworten Anfragen zuzuordnen.
- Lassen Sie Ihre Produzenten keine Antwortwarteschlangen freigeben. Dies kann dazu führen, dass ein Produzent Antwortnachrichten erhält, die für einen anderen Produzenten vorgesehen sind.

Weitere Informationen zum Implementieren der request-response-Muster mit dem Client temporärer Warteschlangen finden Sie unter Request-Response-Messaging-Muster (virtuelle Warteschlangen).

Senkung der Kosten für Amazon SQS

Die folgenden bewährten Methoden können Ihnen dabei helfen, Kosten zu reduzieren und zusätzliche potenzielle Kostensenkungen sowie eine nahezu sofortige Antwort zu nutzen.

Stapelverarbeitungsaktionen für Nachrichten

Um Kosten zu reduzieren, führen Sie Ihre Nachrichtenaktionen in der Stapelverarbeitung aus:

- Zum Senden, Empfangen und Löschen von Nachrichten und zum Ändern der Zeitbeschränkung für die Sichtbarkeit für mehrere Nachrichten mit einer einzelnen Aktion verwenden Sie die Amazon-SQS-API-Stapelaktionen.
- Um die clientseitige Pufferung mit der Anforderungsstapelverarbeitung zu kombinieren, verwenden Sie Langabfragen zusammen mit dem gepufferten asynchronen Client, der in AWS SDK for Java enthalten ist.

> **Note**
>
> Der Amazon SQS Buffered Asynchronous Client unterstützt derzeit keine FIFO-Warteschlangen.

Verwendung des geeigneten Abfragemodus

- Mit einer Langabfrage können Sie Nachrichten aus Ihrer Amazon-SQS-Warteschlange nutzen, sobald diese verfügbar sind.
  - Um die Verwendungskosten für Amazon SQS zu senken und die Anzahl der leeren Empfangsvorgänge in einer leeren Warteschlange zu reduzieren, (Antworten auf die `ReceiveMessage`-Aktion, mit der keine Nachrichten zurückgegeben werden), aktivieren Sie die Langabfrage. Weitere Informationen finden Sie unter Amazon-SQS-Langabfragen.
  - Um die Effizienz beim Abfragen für mehrere Threads mit mehreren Empfangsvorgängen zu steigern, verringern Sie die Anzahl der Threads.
  - Langabfragen sind Kurzabfragen in den meisten Fällen vorzuziehen.
- Eine Kurzabfrage gibt Antworten sofort zurück, auch wenn die abgefragte Amazon-SQS-Warteschlange leer ist.
  - Um die Anforderungen einer Anwendung zu erfüllen, die sofortige Antworten auf die Abfrage `ReceiveMessage` erwartet, verwenden Sie die Kurzabfrage.

- Die Kurzabfrage wird mit denselben Kosten in Rechnung gestellt wie die Langabfrage.

Wechseln von einer Amazon-SQS-Standard- zu einer FIFO-Warteschlange

Wenn Sie den Parameter `DelaySeconds` nicht für jede Nachricht angeben, können Sie zu einer FIFO-Warteschlange wechseln, indem Sie eine Nachrichtengruppen-ID für jede gesendete Nachricht angeben.

Weitere Informationen finden Sie unter Von einer Standardwarteschlange zu einer FIFO-Warteschlange in Amazon SQS wechseln.

Zusätzliche Empfehlungen für Amazon-SQS-FIFO-Warteschlangen

Die folgenden bewährten Methoden können Ihnen dabei helfen, die Nachrichtendeduplizierungs-ID und Nachrichtengruppen-ID optimal einzusetzen. Weitere Informationen finden Sie unter den Aktionen `SendMessage` und `SendMessageBatch` in der Amazon-Simple-Queue-Service-API-Referenz.

Themen

- Verwenden der Amazon-SQS-Nachrichtendeduplizierungs-ID
- Verwenden der Amazon-SQS-Nachrichtengruppen-ID
- Verwenden der Amazon-SQS-Empfangsanforderungsversuch-ID

Verwenden der Amazon-SQS-Nachrichtendeduplizierungs-ID

Die Nachrichtendeduplizierungs-ID ist das Token, das für die Deduplizierung gesendeter Nachrichten verwendet wird. Wenn eine Nachricht mit einer bestimmten Nachrichtendeduplizierungs-ID erfolgreich gesendet wurde, werden alle Nachrichten, die mit derselben Nachrichtendeduplizierungs-ID gesendet wurden, erfolgreich akzeptiert, aber während des 5-minütigen Deduplizierungsintervalls nicht zugestellt.

> **Note**
>
> Amazon SQS verfolgt weiterhin die Nachrichtendeduplizierungs-ID, auch nachdem die Nachricht empfangen und gelöscht wurde.

Bereitstellung der Nachrichtendeduplizierungs-ID

Der Produzent sollte die Nachrichtendeduplizierungs-ID-Werte für jede gesendete Nachricht in den folgenden Szenarien angeben:

- Nachrichten, die mit identischen Nachrichtentexten gesendet werden, die von Amazon SQS als eindeutig behandelt werden müssen.
- Nachrichten, die mit identischen Inhalten, aber unterschiedlichen Nachrichtenattributen gesendet werden, die Amazon SQS als eindeutig behandeln muss.
- Nachrichten, die mit unterschiedlichen Inhalten (z. B. Wiederholungszählungen im Nachrichtentext) gesendet werden, die Amazon SQS als Duplikate behandeln muss.

Aktivieren der Deduplizierung für ein Single-Producer/Consumer-System

Wenn Sie über einen einzigen Produzenten und einen einzigen Konsumenten verfügen und die Nachrichten eindeutig sind, da eine anwendungsspezifische Nachrichten-ID im Nachrichtentext enthalten ist, wenden Sie die folgenden bewährten Methoden an:

- Aktivieren Sie die inhaltsbasierte Deduplizierung für die Warteschlange (jede Ihrer Nachrichten hat einen eindeutigen Text). Der Produzent kann die Nachrichtendeduplizierungs-ID weglassen.
- Wenn die inhaltsbasierte Deduplizierung für eine Amazon SQS FIFO-Warteschlange aktiviert ist und eine Nachricht mit einer Deduplizierungs-ID gesendet wird, überschreibt die Deduplizierungs-ID die generierte inhaltsbasierte `SendMessage` Deduplizierungs-ID.
- Obwohl der Konsument keine Empfangsanforderungsversuch-ID für jede Anforderung angeben muss, hat sich dies als Methode bewährt, da Wiederholungen nach Fehlschlägen schneller durchgeführt werden können.
- Sie können erneut versuchen, Anforderungen zu senden oder zu empfangen, da diese die Reihenfolge der Nachrichten in FIFO-Warteschlangen nicht stören.

Entwürfe für Szenarien zur Wiederherstellung nach Ausfällen

Der Deduplizierungsprozess in FIFO-Warteschlangen ist zeitkritisch. Bei der Konzeption Ihrer Anwendung stellen Sie sicher, dass sowohl der Produzent als auch der Konsument bei einem Client- oder Netzwerkausfall wiederhergestellt werden kann.

- Der Produzent muss das Deduplizierungsintervall der Warteschlange kennen. Amazon SQS verfügt über ein Deduplizierungsintervall von 5 Minuten. Wiederholungsversuche von

`SendMessage`-Anforderungen nach Ablauf des Deduplizierungsintervalls können zu doppelten Nachrichten in der Warteschlange führen. Beispiel: Ein mobiles Gerät in einem Auto sendet Nachrichten, deren Reihenfolge wichtig ist. Wenn das Auto für einen bestimmten Zeitraum die Funkverbindung verliert, bevor eine Bestätigung eingeht, kann ein Wiederholungsversuch der Anforderungen nach Wiederherstellung der Funkverbindung zu einem Duplikat führen.

- Der Konsument muss über eine Zeitbeschränkung für die Sichtbarkeit verfügen, die das Risiko minimiert, dass Nachrichten nicht verarbeitet werden können, bevor die Zeitbeschränkung für die Sichtbarkeit abgelaufen ist. Sie können die Zeitbeschränkung für die Sichtbarkeit erweitern, während die Nachrichten verarbeitet werden, indem Sie die Aktion `ChangeMessageVisibility` aufrufen. Wenn die Zeitbeschränkung für die Sichtbarkeit jedoch abgelaufen ist, kann ein anderer Konsument sofort mit der Nachrichtenverarbeitung beginnen, was dazu führt, dass eine Nachricht mehrfach verarbeitet wird. Um dieses Szenario zu vermeiden, konfigurieren Sie eine Warteschlange für unzustellbare Nachrichten.

Arbeiten mit Zeitbeschränkungen für die Sichtbarkeit

Um eine optimale Leistung zu erzielen, legen Sie das Sichtbarkeits-Timeout so fest, dass es größer ist als das AWS SDK-Lese-Timeout. Dies gilt für die Verwendung der API-Aktion `ReceiveMessage` mit Kurzabfragen gleichermaßen wie mit Langabfragen.

Verwenden der Amazon-SQS-Nachrichtengruppen-ID

`MessageGroupId` ist das Tag, das angibt, dass eine Nachricht zu einer bestimmten Nachrichtengruppe gehört. Nachrichten, die derselben Nachrichtengruppe angehören, werden immer nacheinander in einer strengen Reihenfolge in Bezug auf die Nachrichtengruppe verarbeitet (Nachrichten, die verschiedenen Nachrichtengruppen angehören, können jedoch in einer anderen Reihenfolge verarbeitet werden).

Verschränkung mehrerer geordneter Nachrichtengruppen

Um mehrere geordnete Nachrichtengruppen innerhalb einer einzigen FIFO-Warteschlange zu verschränken, verwenden Sie Nachrichtengruppen-ID-Werte (z. B. Sitzungsdaten für mehrere Benutzer). In diesem Szenario können mehrere Konsumenten die Warteschlange verarbeiten, die Sitzungsdaten der einzelnen Benutzer werden jedoch im FIFO-Verfahren verarbeitet.

> **Note**
>
> Wenn Nachrichten, die zu einer bestimmten Nachrichtengruppen-ID gehören, nicht sichtbar sind, können keine anderen Konsumenten Nachrichten mit derselben Nachrichtengruppen-ID verarbeiten.

Vermeidung der Verarbeitung von Duplikaten in einem Multiple-Producer/Consumer-System

Um die Verarbeitung von doppelten Nachrichten in einem System mit mehreren Produzenten und Konsumenten, bei dem Durchsatz und Latenz wichtiger sind als die Reihenfolge, zu vermeiden, sollte der Produzent eine eindeutige Nachrichtengruppen-ID für jede Nachricht generieren.

> **Note**
>
> In diesem Szenario werden Duplikate verhindert. Die Reihenfolge der Nachrichten kann jedoch nicht garantiert werden.
> Jedes Szenario mit mehreren Produzenten und Konsumenten erhöht das Risiko, versehentlich eine doppelte Nachricht zuzustellen, wenn ein Auftragnehmer die Nachricht nicht innerhalb der Zeitbeschränkung für die Sichtbarkeit verarbeitet und die Nachricht einem anderen Auftragnehmer zur Verfügung gestellt wird.

Vermeiden eines großen Rückstaus an Nachrichten mit derselben Nachrichtengruppen-ID

Bei FIFO-Warteschlangen können maximal 20 000 In-Flight-Nachrichten enthalten sein (die von einem Verbraucher aus einer Warteschlange empfangen, aber noch nicht aus der Warteschlange gelöscht wurden). Wenn Sie dieses Kontingent erreichen, gibt Amazon SQS keine Fehlermeldungen zurück. Eine FIFO-Warteschlange durchsucht die ersten 20 000 Nachrichten, um die verfügbaren Nachrichtengruppen zu ermitteln. Das heißt: Wenn Sie in einer einzelnen Nachrichtengruppe einen Rückstand an Nachrichten haben, können Sie Nachrichten aus anderen Nachrichtengruppen, die zu einem späteren Zeitpunkt an die Warteschlange gesendet wurden, erst verarbeiten, wenn Sie die Nachrichten aus dem Rückstand erfolgreich verarbeitet haben.

> **Note**
>
> Ein Rückstau von Nachrichten mit derselben Nachrichtengruppen-ID kann aufgrund eines Verbrauchers entstehen, der eine Nachricht nicht erfolgreich verarbeiten kann. Probleme bei der Nachrichtenverarbeitung können aufgrund eines Problems mit dem Inhalt einer Nachricht oder aufgrund eines technischen Problems mit dem Verbraucher auftreten.
> Um Nachrichten zu entfernen, die wiederholt nicht verarbeitet werden können, und um die Blockierung der Verarbeitung anderer Nachrichten mit derselben Nachrichtengruppen-ID aufzuheben, sollten Sie eine Warteschlangenrichtlinie für unzustellbare Nachrichten einrichten.

Vermeiden der Wiederverwendung derselben Nachrichtengruppen-ID bei virtuellen Warteschlangen

Um zu verhindern, dass sich Nachrichten, die mit derselben Nachrichtengruppen-ID an verschiedene virtuelle Warteschlangen mit derselben Host-Warteschlange gesandt werden, gegenseitig blockieren, vermeiden Sie es, dieselbe Nachrichtengruppen-ID bei virtuellen Warteschlangen wiederzuverwenden.

Verwenden der Amazon-SQS-Empfangsanforderungsversuch-ID

Die Empfangsanforderungsversuch-ID ist das Token, das für die Deduplizierung von `ReceiveMessage`-Aufrufen verwendet wird.

Während eines dauerhaften Netzwerkausfalls, der zu Verbindungsproblemen zwischen Ihrem SDK und Amazon SQS führt, hat es sich als Methode bewährt, die Empfangsanforderungsversuch-ID bereitzustellen und den Vorgang mit derselben Empfangsanforderungsversuch-ID erneut zu versuchen, wenn die SDK-Operation fehlschlägt.

Beispiele für Amazon SQS Java SDK

Sie können das verwenden AWS SDK for Java , um Java-Anwendungen zu erstellen, die mit Amazon Simple Queue Service (Amazon SQS) und anderen AWS Services interagieren. Zur Installation und Einrichtung des SDK siehe Erste Schritte im AWS SDK for Java 2.x -Entwicklerleitfaden.

Beispiele für grundlegende Amazon-SQS-Warteschlangenoperationen, wie z. B. das Erstellen einer Warteschlange oder das Senden einer Nachricht, finden Sie unter Arbeiten mit Amazon-SQS-Nachrichtenwarteschlangen im AWS SDK for Java 2.x -Entwicklerleitfaden.

Die Beispiele in diesem Thema veranschaulichen zusätzliche Amazon-SQS-Features, wie serverseitige Verschlüsselung (SSE), Kostenzuordnungs-Tags und Nachrichtenattribute.

Themen

- Serverseitige Verschlüsselung mit Amazon SQS SQS-Warteschlangen verwenden
- Konfiguration von Tags für eine Amazon SQS SQS-Warteschlange
- Nachrichtenattribute an eine Amazon SQS SQS-Warteschlange senden

Serverseitige Verschlüsselung mit Amazon SQS SQS-Warteschlangen verwenden

Sie können das verwenden AWS SDK for Java , um serverseitige Verschlüsselung (SSE) zu einer Amazon SQS SQS-Warteschlange hinzuzufügen. Jede Warteschlange verwendet einen AWS Key Management Service (AWS KMS) KMS-Schlüssel, um die Datenverschlüsselungsschlüssel zu generieren. In diesem Beispiel wird der AWS verwaltete KMS-Schlüssel für Amazon SQS verwendet. Weitere Informationen zum Verwenden von SSE und zur Rolle des KMS-Schlüssels finden Sie unter Verschlüsselung im Ruhezustand in Amazon SQS.

Hinzufügen von SSE zu einer vorhandenen Warteschlange

Um die serverseitige Verschlüsselung für eine vorhandene Warteschlange zu aktivieren, verwenden Sie die `SetQueueAttributes`-Methode, um das `KmsMasterKeyId`-Attribut festzulegen.

Im folgenden Codebeispiel wird der AWS KMS key als AWS verwalteter KMS-Schlüssel für Amazon SQS festgelegt. Das Beispiel legt außerdem den Zeitraum für die AWS KMS key -Wiederverwendung auf 140 Sekunden fest.

Bevor Sie den Beispielcode ausführen, stellen Sie sicher, dass Sie Ihre AWS Anmeldeinformationen festgelegt haben. Weitere Informationen finden Sie unter Einrichten von AWS Anmeldeinformationen und Region für die Entwicklung im AWS SDK for Java 2.x Entwicklerhandbuch.

```
// Create an SqsClient for the specified Region.
SqsClient sqsClient = SqsClient.builder().region(Region.US_WEST_1).build();

// Get the URL of your queue.
String myQueueName = "my queue";
GetQueueUrlResponse getQueueUrlResponse =

 sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(myQueueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();

// 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");

// Create the SetQueueAttributesRequest.
SetQueueAttributesRequest set_attrs_request = SetQueueAttributesRequest.builder()
          .queueUrl(queueUrl)
          .attributes(attributes)
          .build();

sqsClient.setQueueAttributes(set_attrs_request);
```

Deaktivieren von SSE für eine Warteschlange

Um die serverseitige Verschlüsselung für eine vorhandene Warteschlange zu deaktivieren, setzen Sie das Attribut `KmsMasterKeyId` mit der Aktion `SetQueueAttributes` auf eine leere Zeichenfolge.

> **Important**
>
> `null` ist kein gültiger Wert für `KmsMasterKeyId`.

Erstellen einer Warteschlange mit SSE

Um SSE beim Erstellen der Warteschlange zu aktivieren, fügen Sie das `KmsMasterKeyId`-Attribut der `CreateQueue`-API-Methode hinzu.

Das folgende Beispiel erstellt eine neue Warteschlange mit aktivierter SSE. Die Warteschlange verwendet den von AWS verwalteten KMS-Schlüssel für Amazon SQS. Das Beispiel legt außerdem den Zeitraum für die AWS KMS key -Wiederverwendung auf 160 Sekunden fest.

Bevor Sie den Beispielcode ausführen, stellen Sie sicher, dass Sie Ihre AWS Anmeldeinformationen festgelegt haben. Weitere Informationen finden Sie unter Einrichten von AWS Anmeldeinformationen und Region für die Entwicklung im AWS SDK for Java 2.x Entwicklerhandbuch.

```
// 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);
```

Abrufen der SSE-Attribute

Informationen zum Abrufen von Warteschlangenattributen finden Sie unter Beispiele in der Amazon-Simple-Queue-Service-API-Referenz.

Um die KMS-Schlüssel-ID oder den Zeitraum für die Wiederverwendung des Datenschlüssels für eine bestimmte Warteschlange abzurufen, führen Sie die `GetQueueAttributes`-Methode aus und rufen Sie die Werte `KmsMasterKeyId` und `KmsDataKeyReusePeriodSeconds` ab.

Konfiguration von Tags für eine Amazon SQS SQS-Warteschlange

Sie können Ihren Amazon-SQS-Warteschlangen Kostenzuordnungs-Tags hinzufügen, um sie zu organisieren und zu identifizieren. Die folgenden Beispiele veranschaulichen, wie Tags mithilfe von AWS SDK for Java konfiguriert werden. Weitere Informationen finden Sie unter Amazon-SQS-Kostenzuordnungs-Tags.

Bevor Sie den Beispielcode ausführen, stellen Sie sicher, dass Sie Ihre AWS Anmeldeinformationen festgelegt haben. Weitere Informationen finden Sie unter Einrichten von AWS Anmeldeinformationen und Region für die Entwicklung im AWS SDK for Java 2.x Entwicklerhandbuch.

Auflisten von Tags

Verwenden Sie die `ListQueueTags`-Methode, um die Tags für eine Warteschlange aufzulisten.

```
// 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() ));
```

Hinzufügen oder Aktualisieren von Tags

Verwenden Sie die `TagQueue`-Methode, um Tag-Werte für eine Warteschlange hinzuzufügen oder zu aktualisieren.

```
 // 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);
```

Entfernen von Tags

Verwenden Sie die `UntagQueue`-Methode, um ein oder mehrere Tags aus der Warteschlange zu entfernen. Im folgenden Beispiel wird das `Accounting ID`-Tag entfernt.

```

// Create the UntagQueueRequest.
final UntagQueueRequest untagQueueRequest = UntagQueueRequest.builder()
        .queueUrl(queueUrl)
        .tagKeys("Accounting ID")
        .build();

// Remove the tag from this queue.
sqsClient.untagQueue(untagQueueRequest);
```

Nachrichtenattribute an eine Amazon SQS SQS-Warteschlange senden

Sie können strukturierte Metadaten (wie etwa Zeitstempel, geospatiale Daten, Signaturen und Kennungen) in Nachrichten einschließen, indem Sie Nachrichtenattribute verwenden. Weitere Informationen finden Sie unter Amazon-SQS-Nachrichtenattribute.

Bevor Sie den Beispielcode ausführen, stellen Sie sicher, dass Sie Ihre AWS Anmeldeinformationen festgelegt haben. Weitere Informationen finden Sie unter Einrichten von AWS Anmeldeinformationen und Region für die Entwicklung im AWS SDK for Java 2.x Entwicklerhandbuch.

Definieren von Attributen

Fügen Sie zum Definieren eines Attributs für eine Nachricht den folgenden Code hinzu, der den `MessageAttributeValue`-Datentyp verwendet. Weitere Informationen finden Sie unter Nachrichtenattributkomponenten und Datentypen für Nachrichtenattribute.

Der berechnet AWS SDK for Java automatisch die Prüfsummen für den Nachrichtentext und die Nachrichtenattribute und vergleicht sie mit den Daten, die Amazon SQS zurückgibt. Weitere Informationen finden Sie im AWS SDK for Java 2.x -Entwicklerhandbuch und in Berechnung des MD5-Nachrichtendigests für Nachrichtenattribute für andere Programmiersprachen.

String

Dieses Beispiel definiert ein `String`-Attribut mit der Bezeichnung `Name` und dem Wert `Jane`.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("Name", new MessageAttributeValue()
.withDataType("String")
.withStringValue("Jane"));
```

Number

Dieses Beispiel definiert ein `Number`-Attribut mit der Bezeichnung `AccurateWeight` und dem Wert `230.0000000000000001`.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("AccurateWeight", new MessageAttributeValue()
.withDataType("Number")
```

```
.withStringValue("230.000000000000000001"));
```

Binary

Dieses Beispiel definiert ein `Binary`-Attribut mit dem Namen `ByteArray` und dem Wert eines nicht initialisierten 10-Byte-Arrays.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("ByteArray", new MessageAttributeValue()
.withDataType("Binary")
.withBinaryValue(ByteBuffer.wrap(new byte[10])));
```

String (custom)

Dieses Beispiel definiert das benutzerdefinierte Attribut `String.EmployeeId` mit der Bezeichnung `EmployeeId` und dem Wert `ABC123456`.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("EmployeeId", new MessageAttributeValue()
.withDataType("String.EmployeeId")
.withStringValue("ABC123456"));
```

Number (custom)

Dieses Beispiel definiert das benutzerdefinierte Attribut `Number.AccountId` mit der Bezeichnung `AccountId` und dem Wert `000123456`.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("AccountId", new MessageAttributeValue()
.withDataType("Number.AccountId")
.withStringValue("000123456"));
```

> **Note**
>
> Da der Basisdatentyp `Number` ist, gibt die `ReceiveMessage`-Aktion `123456` zurück.

Binary (custom)

Dieses Beispiel definiert das benutzerdefinierte Attribut `Binary.JPEG` mit dem Namen `ApplicationIcon` und dem Wert eines nicht initialisierten 10-Byte-Arrays.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("ApplicationIcon", new MessageAttributeValue()
.withDataType("Binary.JPEG")
.withBinaryValue(ByteBuffer.wrap(new byte[10])));
```

Senden einer Nachricht mit Attributen

In diesem Beispiel werden die Attribute der `SendMessageRequest` hinzugefügt, bevor die Nachricht gesendet wird.

```
// 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

Wenn Sie eine Nachricht an eine First-In-First-Out (FIFO)-Warteschlange senden, stellen Sie sicher, dass die `sendMessage`-Methode ausgeführt wird, nachdem Sie die Nachrichtengruppen-ID angeben.
Wenn Sie die `SendMessageBatch`-Aktion anstelle von `SendMessage` verwenden, müssen Sie Nachrichtenattribute für jede Nachricht in dem Stapel angeben.

Arbeiten mit Amazon-SQS-APIs

Dieser Abschnitt enthält Informationen über das Erstellen von Amazon-SQS-Endpunkten, die Durchführung von Query-API-Anforderungen mithilfe der Methoden GET und POST sowie die Verwendung von API-Stapelaktionen. Detaillierte Informationen zu Amazon-SQS-Aktionen – einschließlich Parametern, Fehlern, Beispielen und Datentypen – finden Sie in der Amazon-Simple-Queue-Service-API-Referenz.

Für den Zugriff auf Amazon SQS unter Verwendung unterschiedlichster Programmiersprachen können Sie auch AWS -SDKs verwenden, die die folgende automatische Funktionalität enthalten:

- Kryptographisches Signieren Ihrer Serviceanfragen
- Wiederholen von Anfragen
- Umgang mit Fehlerreaktionen

Informationen zum Befehlszeilen-Tool finden Sie in den Amazon-SQS-Abschnitten in der AWS CLI -Befehlsreferenz und der AWS Tools for PowerShell -Cmdlet-Referenz.

Amazon SQS SQS-APIs mit AWS JSON-Protokoll

Amazon SQS verwendet das AWS JSON-Protokoll als Transportmechanismus für alle Amazon SQS SQS-APIs auf den angegebenen AWS SDK-Versionen. AWS Das JSON-Protokoll bietet einen höheren Durchsatz, eine geringere Latenz und eine schnellere application-to-application Kommunikation. AWS Das JSON-Protokoll ist im Vergleich zum Abfrageprotokoll effizienter bei der Serialisierung/Deserialisierung von Anfragen und Antworten. AWS Wenn Sie das AWS Abfrageprotokoll weiterhin mit SQS-APIs verwenden möchten, finden Sie Welche Sprachen werden für das AWS -JSON-Protokoll unterstützt, das in Amazon-APIs verwendet wird? die AWS SDK-Versionen, die das Amazon SQS AWS SQS-Abfrageprotokoll unterstützen.

Amazon SQS verwendet das AWS JSON-Protokoll für die Kommunikation zwischen AWS SDK-Clients (z. B. Java, Python, Golang JavaScript) und dem Amazon SQS-Server. Eine HTTP-Anfrage eines Amazon-SQS-API-Vorgangs akzeptiert Eingaben im JSON-Format. Der Amazon-SQS-Vorgang wird ausgeführt und die Ausführungsantwort im JSON-Format an den SDK-Client zurückgesendet. Im Vergleich zu AWS Query ist AWS JSON einfacher, schneller und effizienter, wenn es darum geht, Daten zwischen Client und Server zu transportieren.

- AWS Das JSON-Protokoll fungiert als Vermittler zwischen dem Amazon SQS SQS-Client und dem Server.

- Der Server versteht die Programmiersprache nicht, in der der Amazon SQS SQS-Vorgang erstellt wurde, aber er versteht das AWS JSON-Protokoll.
- Das AWS JSON-Protokoll verwendet die Serialisierung (Objekt in das JSON-Format konvertieren) und die Deserialisierung (JSON-Format in Objekt konvertieren) zwischen Amazon SQS SQS-Client und -Server.

Weitere Informationen zum AWS JSON-Protokoll mit Amazon SQS finden Sie unterHäufig gestellte Fragen zum Amazon SQS AWS SQS-JSON-Protokoll.

AWS Das JSON-Protokoll ist in der angegebenen AWS SDK-Version verfügbar. Informationen zur SDK-Version und zu den Veröffentlichungsdaten der verschiedenen Sprachvarianten finden Sie in der Matrix zur Unterstützung von AWS -SDKs und Tools im Referenzhandbuch für AWS -SDKs und Tools

Themen

- Abfrage-API-Anfragen mithilfe des AWS JSON-Protokolls in Amazon SQS stellen
- Abfrage-API-Anfragen mithilfe des AWS Abfrageprotokolls in Amazon SQS stellen
- Authentifizieren von Anfragen für Amazon SQS
- Amazon-SQS-Stapelaktionen

Abfrage-API-Anfragen mithilfe des AWS JSON-Protokolls in Amazon SQS stellen

In diesem Abschnitt erfahren Sie, wie Sie einen Amazon-SQS-Endpunkt erstellen, POST-Anforderungen durchführen und die Antworten interpretieren.

> **Note**
>
> AWS Das JSON-Protokoll wird für die meisten Sprachvarianten unterstützt. Eine Liste der unterstützten Sprachvarianten finden Sie unter Welche Sprachen werden für das AWS -JSON-Protokoll unterstützt, das in Amazon-APIs verwendet wird?.

Themen

- Erstellen eines Endpunkts

- Durchführen einer POST-Anforderung
- Interpretieren von Amazon-SQS-JSON-API-Antworten
- Häufig gestellte Fragen zum Amazon SQS AWS SQS-JSON-Protokoll

Erstellen eines Endpunkts

Für die Arbeit mit Amazon-SQS-Warteschlangen müssen Sie einen Endpunkt erstellen. Informationen zu Amazon-SQS-Endpunkten finden Sie auf den folgenden Seiten im Allgemeine Amazon Web Services-Referenz:

- Regionale Endpunkte
- Endpunkte und Kontingente von Amazon Simple Queue Service

Jeder Amazon-SQS-Endpunkt ist unabhängig. Wenn beispielsweise zwei Warteschlangen benannt sind MyQueueund eine den Endpunkt und die `sqs.us-east-2.amazonaws.com` andere den Endpunkt hat`sqs.eu-west-2.amazonaws.com`, teilen die beiden Warteschlangen keine Daten miteinander.

Das folgende Beispiel zeigt einen Endpunkt, der eine Anforderung zum Erstellen einer Warteschlange stellt.

```
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**
>
> Bei den Namen der Warteschlangen und den Warteschlangen-URLs muss die Groß- und Kleinschreibung beachtet werden.
> Die Struktur von *AUTHPARAMS* hängt davon ab, wie Sie Ihre API-Anforderung signieren. Weitere Informationen finden Sie unter AWS API-Anfragen signieren in der Allgemeinen Referenz zu Amazon Web Services.

Durchführen einer POST-Anforderung

Eine Amazon-SQS-POST-Anforderung sendet Abfrageparameter als Formular im Text einer HTTP-Anforderung.

Im Folgenden finden Sie ein Beispiel für einen HTTP-Header mit der `X-Amz-Target`-Einstellung auf `AmazonSQS.<operationName>` und einen HTTP-Header mit der `Content-Type`-Einstellung auf `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",
}
```

Diese HTTP-POST-Anforderung sendet eine Nachricht an eine Amazon-SQS-Warteschlange.

> **Note**
>
> Beide HTTP-Header, `X-Amz-Target` und `Content-Type`, sind erforderlich.
> Der HTTP-Client fügt abhängig von der HTTP-Version des Clients möglicherweise weitere Elemente zur HTTP-Anforderung hinzu.

Interpretieren von Amazon-SQS-JSON-API-Antworten

Amazon SQS gibt als Antwort auf eine Aktionsanforderung eine JSON-Datenstruktur mit den Ergebnissen der Anforderung zurück. Weitere Informationen finden Sie unter den individuellen Aktionen in der Amazon-Simple-Queue-Service-API-Referenz und in Häufig gestellte Fragen zum Amazon SQS AWS SQS-JSON-Protokoll.

Themen

- Struktur einer JSON-Antwort bei Erfolg
- Struktur einer JSON-Antwort bei Fehlschlagen

Struktur einer JSON-Antwort bei Erfolg

Ist die Anfrage erfolgreich, ist das Hauptantwortelement `x-amzn-RequestId`, das den Universal Unique Identifier (UUID) der Anfrage sowie weitere angehängte Antwortfelder enthält. Beispielsweise enthält die folgende `CreateQueue`-Antwort das Feld `QueueUrl`, das wiederum die URL der erstellten Warteschlange enthält.

```
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"
}
```

Struktur einer JSON-Antwort bei Fehlschlagen

Wenn eine Anfrage nicht erfolgreich ist, gibt Amazon SQS die Hauptantwort zurück, einschließlich des HTTP-Headers und des Texts.

`x-amzn-RequestId` enthält im HTTP-Header die UUID der Anfrage. `x-amzn-query-error` enthält zwei Informationen: die Art des Fehlers, und ob es sich bei dem Fehler um einen Produzenten- oder einen Konsumentenfehler handelt.

`"__type"` gibt im Antworttext weitere Fehlerdetails an und `Message` zeigt die Fehlerbedingung in lesbarem Format an.

Im Folgenden finden Sie eine Beispielfehlerantwort im JSON-Format:

```
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."
}
```

Häufig gestellte Fragen zum Amazon SQS AWS SQS-JSON-Protokoll

Häufig gestellte Fragen zur Verwendung des AWS JSON-Protokolls mit Amazon SQS.

Was ist das AWS JSON-Protokoll und wie unterscheidet es sich von bestehenden Amazon SQS SQS-API-Anfragen und -Antworten?

JSON ist eine der am weitesten verbreiteten und akzeptierten Verbindungsmethoden für die Kommunikation zwischen heterogenen Systemen. Amazon SQS verwendet JSON als Medium für die Kommunikation zwischen einem AWS SDK-Client (z. B. Java, Python, Golang JavaScript) und einem Amazon SQS-Server. Eine HTTP-Anfrage eines Amazon-SQS-API-Vorgangs akzeptiert Eingaben in Form von JSON. Der Amazon-SQS-Vorgang wird ausgeführt und die Ausführungsantwort wird im JSON-Format an den SDK-Client zurückgesendet. Im Vergleich zu AWS -Abfragen ist JSON effizienter, wenn es darum geht, Daten zwischen Client und Server zu transportieren.

- Das Amazon SQS AWS JSON-Protokoll fungiert als Vermittler zwischen dem Amazon SQS SQS-Client und dem Server.
- Der Server versteht die Programmiersprache, in der der Amazon SQS SQS-Vorgang erstellt wurde, nicht, aber er versteht das AWS JSON-Protokoll.
- Das Amazon SQS AWS JSON-Protokoll verwendet die Serialisierung (Objekt in das JSON-Format konvertieren) und Deserialisierung (JSON-Format in Objekt konvertieren) zwischen dem Amazon SQS SQS-Client und -Server.

Wie fange ich mit AWS JSON-Protokollen für Amazon SQS an?

Um mit der neuesten AWS SDK-Version zu beginnen und schnelleres Messaging für Amazon SQS zu erreichen, aktualisieren Sie Ihr AWS SDK auf die angegebene Version oder eine nachfolgende

Version. Weitere Informationen zu SDK-Clients finden Sie in der Spalte „Leitfaden“ in der Tabelle unten.

Im Folgenden finden Sie eine Liste der SDK-Versionen in allen Sprachvarianten für das AWS JSON-Protokoll zur Verwendung mit Amazon SQS SQS-APIs:

Sprache	SDK-Client-Repository	Erforderliche SDK-Clientversion	Richtlinie
C++	aws/aws-sdk-cpp	1.11.98	AWS SDK for C++
Golang 1.x	aws/aws-sdk-go	v1.47.7	AWS SDK for Go
Golang 2.x	aws/aws-sdk-go-v2	v1.28.0	AWS SDK for Go V2
Java 1.x	aws/aws-sdk-java	1.12.585	AWS SDK für Java
Java 2.x	aws/aws-sdk-java-v2	2,21,19	AWS SDK für Java
JavaScript v2.x	aws/aws-sdk-js	v2.1492.0	JavaScript auf AWS
JavaScript v3.x	aws/aws-sdk-js-v3	v3.447.0	JavaScript auf AWS
.NET	aws/aws-sdk-net	3.7.681.0	AWS SDK for .NET
PHP	aws/aws-sdk-php	3.285.2	AWS SDK for PHP
Python-boto3	boto/boto3	1.28.82	AWS SDK for Python (Boto3)
Python-botocore	boto/botocore	1.31.82	AWS SDK for Python (Boto3)

Sprache	SDK-Client-Repository	Erforderliche SDK-Clientversion	Richtlinie
awscli	AWS CLI	1.29.82	AWS-Befehlszeilenschnittstelle
Ruby	aws/aws-sdk-ruby	1,67,0	AWS SDK for Ruby

Was sind die Risiken, wenn ich das JSON-Protokoll für meine Amazon-SQS-Workloads aktiviere?

Wenn Sie eine benutzerdefinierte Implementierung von AWS SDK oder eine Kombination aus benutzerdefinierten Clients und AWS SDK für die Interaktion mit Amazon SQS verwenden, das AWS abfragebasierte (auch bekannt als XML-basierte) Antworten generiert, ist dies möglicherweise nicht mit dem AWS JSON-Protokoll kompatibel. Wenn Sie auf Probleme stoßen, wenden Sie sich an den AWS Support.

Was ist, wenn ich bereits die neueste AWS SDK-Version verwende, aber meine Open-Source-Lösung JSON nicht unterstützt?

Sie müssen Ihre SDK-Version auf die Version ändern, die vor der von Ihnen verwendeten Version liegt. Weitere Informationen Wie fange ich mit AWS JSON-Protokollen für Amazon SQS an? finden Sie unter. AWS Die unter aufgeführten SDK-Versionen Wie fange ich mit AWS JSON-Protokollen für Amazon SQS an? verwenden das JSON-Wire-Protokoll für Amazon SQS SQS-APIs. Wenn Sie Ihr AWS SDK auf die vorherige Version ändern, verwenden Ihre Amazon SQS SQS-APIs die AWS Abfrage.

Welche Sprachen werden für das AWS -JSON-Protokoll unterstützt, das in Amazon-APIs verwendet wird?

Amazon SQS unterstützt alle Sprachvarianten, in denen AWS SDKs allgemein verfügbar sind (GA). Derzeit unterstützen wir Kotlin, Rust oder Swift nicht. Weitere Informationen zu anderen Sprachvarianten finden Sie unter Tools, auf der Grundlage von AWS.

Welche Regionen werden für das in Amazon-SQS-APIs verwendete AWS -JSON-Protokoll unterstützt?

Amazon SQS unterstützt das AWS JSON-Protokoll in allen AWS Regionen, in denen Amazon SQS verfügbar ist.

Welche Latenzverbesserungen kann ich erwarten, wenn ich ein Upgrade auf die angegebenen AWS SDK-Versionen für Amazon SQS mithilfe des AWS JSON-Protokolls durchführe?

AWS Das JSON-Protokoll ist im Vergleich zum Abfrageprotokoll effizienter bei der Serialisierung und Deserialisierung von Anfragen und Antworten. AWS Basierend auf AWS Leistungstests für eine Nachrichtennutzlast von 5 KB reduziert das JSON-Protokoll für Amazon SQS die Latenz bei der end-to-end Nachrichtenverarbeitung um bis zu 23% und reduziert die clientseitige CPU- und Speicherauslastung der Anwendung.

Wird das AWS Abfrageprotokoll veraltet sein?

AWS Das Abfrageprotokoll wird weiterhin unterstützt. Sie können das AWS Abfrageprotokoll weiterhin verwenden, solange für Ihre AWS SDK-Version eine andere Version festgelegt ist als die, die unter Wie fange ich mit AWS JSON-Protokollen für Amazon SQS an? aufgeführt ist.

Wo finde ich weitere Informationen zum AWS -JSON-Protokoll?

Weitere Informationen zum JSON-Protokoll finden Sie unter AWS -JSON-1.0-Protokoll in der Smithy-Dokumentation. Weitere Informationen zu Amazon-SQS-API-Anfragen mit dem AWS -JSON-Protokoll finden Sie unter Abfrage-API-Anfragen mithilfe des AWS JSON-Protokolls in Amazon SQS stellen.

Abfrage-API-Anfragen mithilfe des AWS Abfrageprotokolls in Amazon SQS stellen

In diesem Abschnitt erfahren Sie, wie Sie einen Amazon-SQS-Endpunkt erstellen, GET- und POST-Anforderungen durchführen und die Antworten interpretieren.

Themen

- Erstellen eines Endpunkts

- Durchführen einer GET-Anforderung
- Durchführen einer POST-Anforderung
- Interpretieren von Amazon-SQS-XML-API-Antworten

Erstellen eines Endpunkts

Für die Arbeit mit Amazon-SQS-Warteschlangen müssen Sie einen Endpunkt erstellen. Informationen zu Amazon-SQS-Endpunkten finden Sie auf den folgenden Seiten im Allgemeine Amazon Web Services-Referenz:

- Regionale Endpunkte
- Endpunkte und Kontingente von Amazon Simple Queue Service

Jeder Amazon-SQS-Endpunkt ist unabhängig. Wenn beispielsweise zwei Warteschlangen benannt sind MyQueueund eine den Endpunkt und die `sqs.us-east-2.amazonaws.com` andere den Endpunkt hat`sqs.eu-west-2.amazonaws.com`, teilen die beiden Warteschlangen keine Daten miteinander.

Das folgende Beispiel zeigt einen Endpunkt, der eine Anforderung zum Erstellen einer Warteschlange stellt.

```
https://sqs.eu-west-2.amazonaws.com/
?Action=CreateQueue
&DefaultVisibilityTimeout=40
&QueueName=MyQueue
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> Bei den Namen der Warteschlangen und den Warteschlangen-URLs muss die Groß- und Kleinschreibung beachtet werden.
> Die Struktur von *AUTHPARAMS* hängt davon ab, wie Sie Ihre API-Anforderung signieren. Weitere Informationen finden Sie unter AWS API-Anfragen signieren in der Allgemeinen Referenz zu Amazon Web Services.

Durchführen einer GET-Anforderung

Eine Amazon-SQS-GET-Anforderung ist als URL mit den folgenden Komponenten aufgebaut:

- Endpunkt – Die Ressource, für die die Anforderung ausgeführt wird (der Warteschlangenname und die URL), z. B.: `https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue`
- Aktion – Die Aktion, die Sie für den Endpunkt ausführen möchten. Ein Fragezeichen (?) trennt den Endpunkt von der Aktion, z. B.: `?Action=SendMessage&MessageBody=Your%20Message%20Text`
- Parameter – Beliebige Anforderungsparameter. Jeder Parameter ist durch ein kaufmännisches Und (&) vom Text getrennt, zum Beispiel: `&Version=2012-11-05&AUTHPARAMS`

Im Folgenden finden Sie ein Beispiel für eine GET-Anforderung, die eine Nachricht an eine Amazon-SQS-Warteschlange sendet.

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue
?Action=SendMessage&MessageBody=Your%20message%20text
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> Bei den Namen der Warteschlangen und den Warteschlangen-URLs muss die Groß- und Kleinschreibung beachtet werden.
> Da es sich bei GET-Anforderungen um URLs handelt, müssen Sie für alle Parameterwerte eine URL-Codierung durchführen. Da in URLs jedoch keine Leerzeichen zulässig sind, wird für jede Leerstelle eine URL-Codierung in `%20` durchgeführt. Für den Rest des Beispiels wurde zum Zweck der besseren Lesbarkeit keine URL-Codierung durchgeführt.

Durchführen einer POST-Anforderung

Eine Amazon-SQS-POST-Anforderung sendet Abfrageparameter als Formular im Text einer HTTP-Anforderung.

Es folgt ein Beispiel eines HTTP-Headers mit der Einstellung `application/x-www-form-urlencoded` für `Content-Type`.

```
POST /123456789012/MyQueue HTTP/1.1
Host: sqs.us-east-2.amazonaws.com
Content-Type: application/x-www-form-urlencoded
```

Auf den Header folgt eine `form-urlencoded`-GET-Anforderung, die eine Nachricht an eine Amazon-SQS-Warteschlange sendet. Jeder Parameter ist durch ein kaufmännisches Und (&) vom Text getrennt.

```
Action=SendMessage
&MessageBody=Your+Message+Text
&Expires=2020-10-15T12%3A00%3A00Z
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> Nur der `Content-Type` HTTP-Header ist erforderlich. Der *AUTHPARAMS* ist für die GET-Anforderung derselbe.
> Der HTTP-Client fügt abhängig von der HTTP-Version des Clients möglicherweise weitere Elemente zur HTTP-Anforderung hinzu.

Interpretieren von Amazon-SQS-XML-API-Antworten

Amazon SQS gibt als Antwort auf eine Aktionsanforderung eine XML-Datenstruktur mit den Ergebnissen der Anforderung zurück. Weitere Informationen finden Sie unter den individuellen Aktionen in der Amazon-Simple-Queue-Service-API-Referenz.

Themen

- Struktur einer XML-Antwort bei Erfolg
- XML-Fehler-Antwortstruktur

Struktur einer XML-Antwort bei Erfolg

Wenn die Anforderung erfolgreich ist, wird das Hauptantwortelement nach der Aktion mit dem Zusatz `Response` (zum Beispiel *ActionName*`Response`) benannt.

Dieses Element enthält die folgenden untergeordneten Elemente:

- ***ActionName*Result** – Enthält ein aktionsspezifisches Element. Das Element `CreateQueueResult` enthält das Element `QueueUrl`, das wiederum die URL der erstellten Warteschlange enthält.
- **`ResponseMetadata`** – Enthält die `RequestId`, die wiederum die UUID (Universal Unique Identifier) der Anforderung enthält.

Nachfolgend finden Sie ein Beispiel für eine Antwort bei Erfolg im XML-Format:

```
<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>
```

XML-Fehler-Antwortstruktur

Wenn eine Anforderung fehlschlägt, gibt Amazon SQS immer das Haupt-Antwort-Element `ErrorResponse` zurück. Dieses Element enthält ein `Error`- und ein `RequestId`-Element.

Das Element `Error` enthält die folgenden untergeordneten Elemente:

- **`Type`** – Gibt an, ob es sich bei dem Fehler um einen Produzenten- oder einen Konsumentenfehler handelt.
- **`Code`** – Gibt den Typ des Fehlers an.
- **`Message`** – Gibt die Fehlerbedingung in einem lesbaren Format an.
- **`Detail`** – (Optional) Gibt zusätzliche Details zu dem Fehler an.

Das Element `RequestId` enthält die UUID der Anforderung.

Nachfolgend finden Sie ein Beispiel für eine Antwort bei Fehlschlagen im XML-Format:

```
<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>
```

Authentifizieren von Anfragen für Amazon SQS

Authentifizierung bezeichnet den Prozess der Identifizierung und Überprüfung des Absenders einer Anforderung. Während der ersten Phase der Authentifizierung überprüft AWS die Identität des Produzenten, und ob der Produzent für die Verwendung von AWS registriert ist (weitere Informationen finden Sie unter Schritt 1: Erstellen Sie einen AWS-Konto und IAM-Benutzer). Halten Sie sich AWS als Nächstes an das folgende Verfahren:

1. Der Produzent (Absender) erhält die notwendigen Anmeldeinformationen.
2. Der Produzent sendet eine Anforderung mit den Anmeldeinformationen an den Konsumenten (Empfänger).
3. Der Konsument überprüft anhand der Anmeldeinformationen, ob der Produzent die Anforderung gesendet hat.
4. Es tritt einer der beiden folgenden Fälle ein:
   - Wenn die Authentifizierung erfolgreich durchgeführt wurde, wird die Anforderung vom Konsumenten verarbeitet.
   - Wenn die Authentifizierung fehlschlägt, wird die Anforderung vom Verbraucher abgelehnt und es wird eine Fehlermeldung zurückgegeben.

Themen

- Grundlegender Authentifizierungsprozess mit HMAC-SHA
- Teil 1: Die Anforderung von dem Benutzer
- Teil 2: Die Antwort von AWS

Grundlegender Authentifizierungsprozess mit HMAC-SHA

Beim Zugriff auf Amazon SQS mit der Abfrage-API müssen Sie die folgenden Elemente angeben, um die Anforderung zu authentifizieren:

- Die AWS Zugangsschlüssel-ID, mit der Sie AWS-Konto identifiziert werden und AWS anhand derer Ihr geheimer Zugriffsschlüssel nachgeschlagen wird.
- Die HMAC-SHA-Anforderungssignatur, die anhand Ihres geheimen Zugriffsschlüssels (eines freigegebenen Schlüssels, der nur Ihnen und AWS bekannt ist) berechnet wird. Weitere Informationen finden Sie unter RFC2104. Das AWS -SDK übernimmt die Signatur. Wenn Sie jedoch eine Abfrageanforderung über HTTP oder HTTPS senden, müssen Sie in jede Abfrageanforderung eine Signatur einschließen.

  1. Ableiten eines Signature Version 4-Signaturschlüssels. Weitere Informationen finden Sie unter Ableiten des Signaturschlüssels mit Java.

     > **Note**
     >
     > Amazon SQS unterstützt Signature Version 4, das gegenüber bisherigen Versionen verbesserte SHA256-basierte Sicherheit und Leistung bietet. Wenn Sie neue Anwendungen erstellen, die Amazon SQS nutzen, verwenden Sie Signature Version 4.

  2. Codieren Sie die Anforderungssignatur mit Base64. Das folgende Java-Codebeispiel führt folgende Schritte aus:

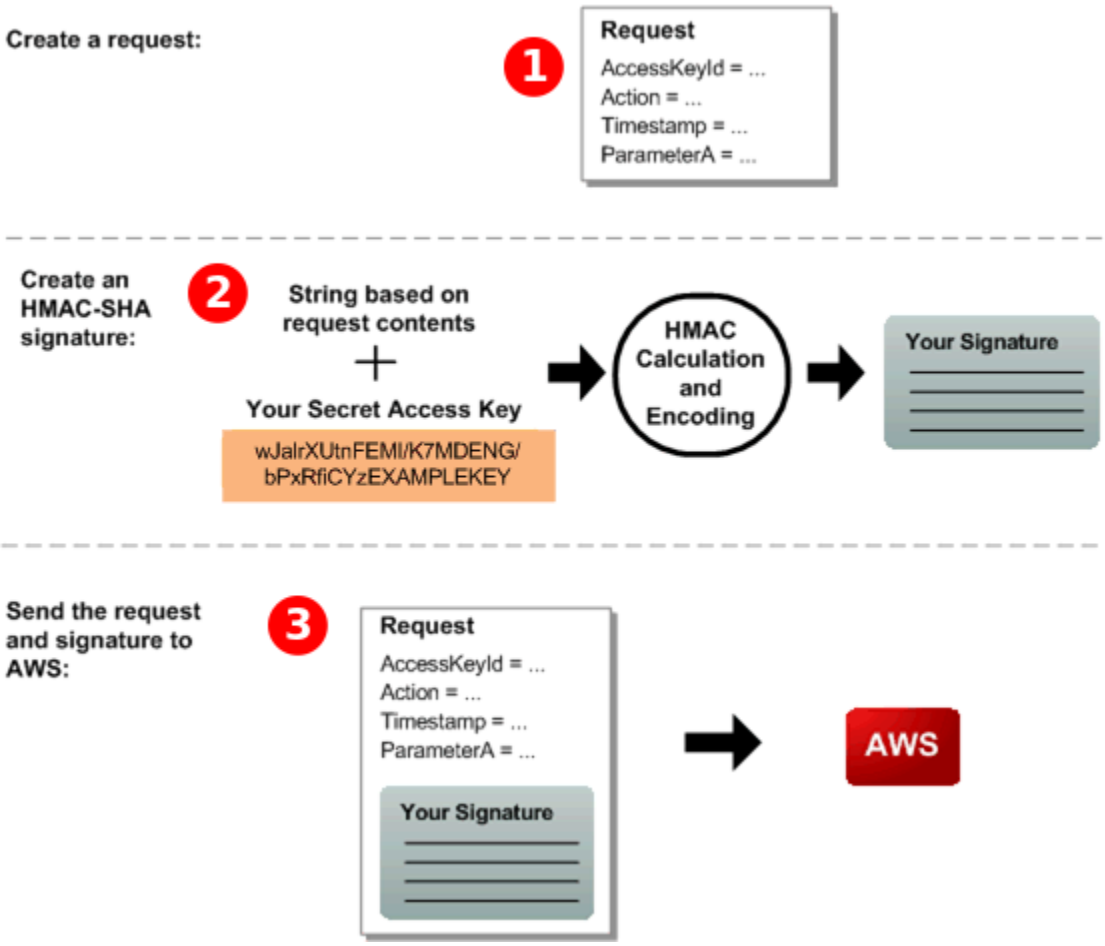
```
package amazon.webservices.common;

// 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);
    }
}
```

- Der Zeitstempel (oder die Ablaufzeit) der Anforderung. Der Zeitstempel, den Sie in der Anforderung verwenden, muss ein `dateTime`-Objekt mit dem vollständigen Datum, einschließlich Stunden, Minuten und Sekunden, sein. Beispiel: `2007-01-31T23:59:59Z` Obwohl dies nicht erforderlich ist, empfehlen wird, die Angabe des Objekts in der UTC-Zeitzone (Greenwich Mean Time).

> **Note**
>
> Stellen Sie sicher, dass Ihre Serverzeit richtig eingestellt ist. Wenn Sie einen Zeitstempel (statt eines Ablaufs) angeben, läuft die Anfrage automatisch 15 Minuten nach der angegebenen Zeit ab (Anfragen, deren Zeitstempel mehr als 15 Minuten vor der aktuellen Uhrzeit auf AWS Servern liegen, werden AWS nicht verarbeitet).
> Falls Sie .NET verwenden, dürfen Sie keine extrem spezifischen Zeitstempel senden (da unterschiedlich interpretiert wird, nach welchem Zeitraum die Anforderung verworfen werden sollte). In diesem Fall sollten Sie manuell `dateTime`-Objekte mit einer Genauigkeit von nicht mehr als einer Millisekunde erstellen.

Teil 1: Die Anforderung von dem Benutzer

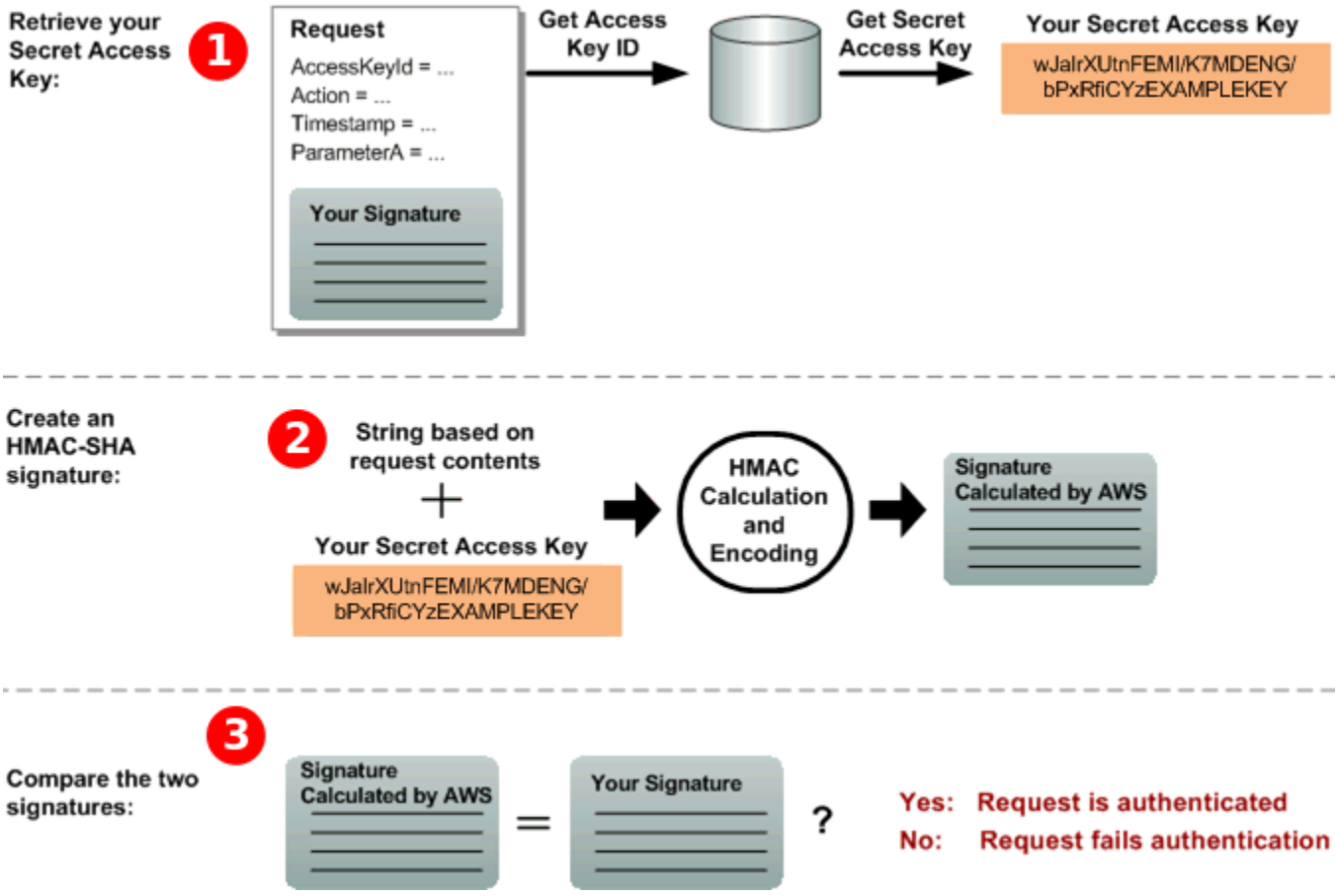
Gehen Sie wie folgt vor, um AWS Anfragen mithilfe einer HMAC-SHA-Anforderungssignatur zu authentifizieren.



1. Konstruieren Sie eine Anfrage an AWS.
2. Berechnen Sie eine verschlüsselte Hash-Signatur (HMAC-SHA) für die Nachrichtenauthentifizierung anhand des geheimen Zugriffsschlüssels.
3. Fügen Sie die Signatur und Ihre Zugriffsschlüssel-ID in die Anfrage ein und senden Sie die Anfrage dann an AWS.

Teil 2: Die Antwort von AWS

AWS startet als Antwort den folgenden Prozess.



1. AWS verwendet die Zugriffsschlüssel-ID, um nach Ihrem geheimen Zugriffsschlüssel zu suchen.
2. AWS generiert eine Signatur aus den Anforderungsdaten und dem geheimen Zugriffsschlüssel und verwendet dabei denselben Algorithmus, den Sie zur Berechnung der Signatur verwendet haben, die Sie in der Anfrage gesendet haben.
3. Es tritt einer der beiden folgenden Fälle ein:
   - Wenn die AWS generierte Signatur mit der Signatur übereinstimmt, die Sie in der Anfrage gesendet haben, wird die Anfrage als authentisch AWS betrachtet.
   - Schlägt der Vergleich fehl, wird die Anfrage verworfen und es wird ein Fehler AWS zurückgegeben.

Amazon-SQS-Stapelaktionen

Um Kosten zu senken oder mit einem einzigen Aktion bis zu 10 Nachrichten gleichzeitig zu bearbeiten, können Sie die folgenden Aktionen verwenden:

- `SendMessageBatch`
- `DeleteMessageBatch`
- `ChangeMessageVisibilityBatch`

Sie können die Batch-Funktionalität mithilfe der Query API oder eines AWS SDK nutzen, das die Amazon SQS SQS-Batch-Aktionen unterstützt.

> **Note**
>
> Die Gesamtgröße aller Nachrichten, die Sie in einem einzigen `SendMessageBatch` Anruf senden, darf 262.144 Byte (256 KiB) nicht überschreiten.
> Sie können für `SendMessageBatch`, `DeleteMessageBatch` oder `ChangeMessageVisibilityBatch` keine Berechtigungen explizit zuweisen.
> Durch das Festlegen der Berechtigungen für `SendMessage`, `DeleteMessage` oder `ChangeMessageVisibility` werden auch die Berechtigungen für die entsprechenden Stapelversionen dieser Aktionen festgelegt.
> Die Amazon-SQS-Konsole unterstützt keine Stapelaktionen.

Themen

- Aktivierung der clientseitigen Pufferung und der Batchverarbeitung von Anfragen mit Amazon SQS
- Steigerung des Durchsatzes durch horizontale Skalierung und Action-Batching mit Amazon SQS

Aktivierung der clientseitigen Pufferung und der Batchverarbeitung von Anfragen mit Amazon SQS

AWS SDK for Java enthält den `AmazonSQSBufferedAsyncClient`, der auf Amazon SQS zugreift. Dieser Client ermöglicht eine einfachere Stapelverarbeitung von Anforderungen mittels clientseitiger Pufferung, wobei vom Client durchgeführte Aufrufe zunächst gepuffert und anschließend als Stapelanforderung an Amazon SQS gesendet werden.

Mit der clientseitigen Pufferung können bis zu 10 Anforderungen zwischengespeichert und als Stapelanforderung gesendet werden. Dadurch werden Ihre Kosten für die Verwendung von Amazon SQS gesenkt und die Anzahl der gesendeten Anforderungen verringert. `AmazonSQSBufferedAsyncClient` puffert synchrone und asynchrone Aufrufe. Als Stapel

verarbeitete Anforderungen und die Unterstützung für langes Abrufen können ebenfalls zu einem erhöhten Durchsatz beitragen. Weitere Informationen finden Sie unter Steigerung des Durchsatzes durch horizontale Skalierung und Action-Batching mit Amazon SQS.

Da `AmazonSQSBufferedAsyncClient` die gleiche Schnittstelle implementiert wie `AmazonSQSAsyncClient`, sollte die Migration von `AmazonSQSAsyncClient` zu `AmazonSQSBufferedAsyncClient` in der Regel lediglich minimale Änderungen an Ihrem vorhandenen Code erfordern.

> **Note**
>
> Der Amazon SQS Buffered Asynchronous Client unterstützt derzeit keine FIFO-Warteschlangen.

Themen

- Verwenden des AmazonSQS-Clients BufferedAsync
- Konfiguration des AmazonSQS-Clients BufferedAsync

Verwenden des AmazonSQS-Clients BufferedAsync

Bevor Sie beginnen, führen Sie die Schritte in Einrichten von Amazon SQS aus.

> **Important**
>
> Das AWS SDK for Java 2.x ist derzeit nicht kompatibel mit dem`AmazonSQSBufferedAsyncClient`.

Sie können basierend auf `AmazonSQSAsyncClient` einen neuen `AmazonSQSBufferedAsyncClient` erstellen, z. B.:

```
// Create the basic Amazon SQS async client
final AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient();

// Create the buffered client
final AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync);
```

Nachdem Sie den neuen `AmazonSQSBufferedAsyncClient` erstellt haben, können Sie ihn verwenden, um mehrere Anforderungen an Amazon SQS zu senden (wie mit dem `AmazonSQSAsyncClient`), beispielsweise:

```
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);
```

Konfiguration des AmazonSQS-Clients BufferedAsync

`AmazonSQSBufferedAsyncClient` ist mit Einstellungen vorkonfiguriert, die für die meisten Anwendungsfälle verwendet werden können. Sie können `AmazonSQSBufferedAsyncClient` weiter konfigurieren, z. B.:

1. Erstellen Sie eine Instance der Klasse `QueueBufferConfig` mit den erforderlichen Konfigurationsparametern.
2. Stellen Sie die Instance dem `AmazonSQSBufferedAsyncClient`-Konstruktor zur Verfügung.

```
// 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);
```

QueueBufferConfig Konfigurationsparameter

Parameter	Standardwert	Beschreibung
`longPoll`	`true`	Wenn `longPoll` auf `true` festgelegt ist, versucht `AmazonSQSBufferedAsyncClient` , Nachrichten durch langes Abrufen abzurufen.
`longPollWaitTimeoutSeconds`	20 s	Die maximale Dauer (in Sekunden), die ein `ReceiveMessage` -Aufruf beim Warten auf das Auftauchen von Nachrichten in der Warteschlange auf dem Server blockiert wird, bevor er mit einem leeren Empfangsergebnis zurückkehrt. **Note** Wenn langes Abrufen deaktiviert ist, hat diese Einstellung hat keine Auswirkungen.
`maxBatchOpenMs`	200 ms	Die maximale Dauer (in Millisekunden), die ein ausgehender Aufruf auf andere Aufrufe desselben Typs wartet, mit denen er Nachrichten desselben Typs

Parameter	Standardwert	Beschreibung
		zu einem Stapel zusammenf asst. Je höher die Einstellung, desto weniger Stapel sind erforderl ich, um dieselbe Anzahl an Aufgaben auszuführen (der erste Aufruf eines Stapels muss jedoch länger in der Warteschlange warten). Wenn Sie diesen Parameter auf `0` einstellen, warten gesendete Anforderungen nicht auf andere Anforderu ngen, wodurch die Stapelver arbeitung effektiv deaktiviert wird.
`maxBatchSize`	10 Anforderungen pro Stapel	Die maximale Anzahl von Nachrichten, die in einer einzelnen Stapelanforderung zusammengefasst werden. Je höher die Einstellung, desto weniger Stapel müssen die gleiche Anzahl von Anforderu ngen ausführen. **Note** 10 Anforderungen pro Stapel ist der maximal zulässige Wert für Amazon SQS.

Parameter	Standardwert	Beschreibung
`maxBatchSizeBytes`	256 KiB	Die maximale Größe eines Nachrichtenstapels in Bytes, die der Client versucht, an Amazon SQS zu senden. **Note** 256 KiB ist der maximal zulässige Wert für Amazon SQS.

Parameter	Standardwert	Beschreibung
`maxDoneReceiveBatches`	10 Stapel	Die maximale Anzahl von Empfangsstapeln, die `AmazonSQSBufferedAsyncClient` vorab abruft und clientseitig speichert. Je höher die Einstellung, desto mehr Empfangsanforderungen können erfüllt werden, ohne dass ein Aufruf an Amazon SQS gestartet werden muss (je mehr Nachrichten jedoch vorab abgerufen werden, desto länger bleiben sie im Puffer, was bedeutet, dass deren Zeitbeschränkung für die Sichtbarkeit abläuft). **Note** `0`gibt an, dass der gesamte Vorabruf von Nachrichten deaktiviert ist und Nachrichten nur bei Bedarf abgerufen werden.

Parameter	Standardwert	Beschreibung
`maxInflightOutboundBatches`	5 Stapel	Die maximale Anzahl der aktiven ausgehenden Stapel, die gleichzeitig verarbeitet werden können. Je höher die Einstellung, desto schneller können ausgehende Stapel gesendet werden (in Abhängigkeit von anderen Kontingenten, z. B. durch CPU oder Bandbreite) und desto mehr Threads werden von `AmazonSQSBufferedAsyncClient` verbraucht.

Parameter	Standardwert	Beschreibung
`maxInflightReceiveBatches`	10 Stapel	Die maximale Anzahl der aktiven Empfangsstapel, die gleichzeitig verarbeitet werden können. Je höher die Einstellung, desto mehr Nachrichten können empfangen werden (in Abhängigkeit von anderen Kontingenten, z. B. durch CPU oder Bandbreite) und desto mehr Threads werden von `AmazonSQSBufferedAsyncClient` verbraucht. **Note** `0`gibt an, dass der gesamte Vorabruf von Nachrichten deaktiviert ist und Nachrichten nur bei Bedarf abgerufen werden.

Parameter	Standardwert	Beschreibung
`visibilityTimeoutSeconds`	-1	Wenn dieser Parameter auf einen positiven Wert ungleich null festgelegt ist, überschreibt die hier festgelegte Zeitbeschränkung für die Sichtbarkeit diejenige, die für die Warteschlange festgelegt ist, über die Nachrichten abgerufen werden. **Note** `-1` gibt an, dass die Standardeinstellung für die Warteschlange ausgewählt ist. Sie können als Zeitbeschränkung für die Sichtbarkeit nicht `0` festlegen.

Steigerung des Durchsatzes durch horizontale Skalierung und Action-Batching mit Amazon SQS

Amazon-SQS-Warteschlangen können einen sehr hohen Durchsatz erzielen. Weitere Informationen zu Durchsatzkontingenten finden Sie unter Amazon SQS SQS-Nachrichtenkontingente.

Um einen hohen Durchsatz zu erreichen, müssen Sie Nachrichtenproduzenten und -konsumenten horizontal skalieren (durch Hinzufügen weiterer Produzenten und Konsumenten).

Themen

- Horizontale Skalierung
- Stapelverarbeitung von Aktionen

- Funktionierendes Java-Beispiel für einzelne Operationsanforderungen und Stapelanforderungen

Horizontale Skalierung

Da Sie über ein HTTP-Anfrage-Antwort-Protokoll auf Amazon SQS zugreifen, beschränkt die Anforderungslatenz (der Zeitraum zwischen dem Initiieren einer Anforderung und dem Empfangen einer Antwort) den Durchsatz, den Sie über einen einzelnen Thread über eine einzelne Verbindung erzielen können. Wenn beispielsweise der Mittelwert der Latenz eines auf EC2 basierenden Clients zu Amazon SQS in derselben Region ca. 20 ms beträgt, liegt der Mittelwert des maximalen Durchsatzes eines einzelnen Threads über eine einzelne Verbindung bei 50 TPS.

Horizontale Skalierung bedeutet, dass die Anzahl Ihrer Nachrichtenproduzenten (die `SendMessage`-Anforderungen erstellen) und Konsumenten (die `ReceiveMessage`- und `DeleteMessage`-Anforderungen erstellen) erhöht wird, um den Gesamtdurchsatz der Warteschlange zu steigern. Sie können auf drei Arten horizontal skalieren:

- Erhöhen der Anzahl der Threads pro Client
- Hinzufügen weiterer Clients
- Erhöhen der Anzahl der Threads pro Client und Hinzufügen weiterer Clients

Wenn Sie weitere Clients hinzufügen, erzielen Sie eine wesentliche lineare Steigerung des Durchsatzes der Warteschlange. Wenn Sie die Anzahl der Clients beispielsweise verdoppeln, verdoppeln Sie auch den Durchsatz.

> **Note**
>
> Wenn Sie eine horizontale Skalierung durchführen, müssen Sie sicherstellen, dass der Amazon-SQS-Client über ausreichend Verbindungen oder Threads verfügt, um die Anzahl der gleichzeitigen Nachrichtenproduzenten und -verbraucher zu unterstützen, die Anforderungen senden und empfangen. Beispielsweise unterhalten Instances der AWS SDK for Java `AmazonSQSClient` Klasse standardmäßig maximal 50 Verbindungen zu Amazon SQS. Um zusätzliche gleichzeitige Produzenten und Konsumenten zu erstellen, müssen Sie die maximal zulässige Anzahl von Produzenten- und Konsumenten-Threads für ein `AmazonSQSClientBuilder`-Objekt anpassen, z. B.:
>
> ```
> final AmazonSQS sqsClient = AmazonSQSClientBuilder.standard()
>         .withClientConfiguration(new ClientConfiguration()
> ```

```
                .withMaxConnections(producerCount + consumerCount))
        .build();
```

Für `AmazonSQSAsyncClient` müssen Sie außerdem sicherstellen, dass ausreichend Threads verfügbar sind.
Dieses Beispiel funktioniert nur für Java v. 1.x.

Stapelverarbeitung von Aktionen

Mit der Stapelverarbeitung wird in den einzelnen Roundtrips an den Service mehr Arbeit ausgeführt (z. B. das Senden mehrerer Nachrichten mit einer einzelnen `SendMessageBatch`-Anforderung). Die Amazon-SQS-Stapelaktionen sind `SendMessageBatch`, `DeleteMessageBatch` und `ChangeMessageVisibilityBatch`. Um die Stapelverarbeitung zu nutzen, ohne Ihre Produzenten und Konsumenten zu ändern, können Sie den Amazon SQS Buffered Asynchronous Client verwenden.

> **Note**
>
> Da mit `ReceiveMessage` 10 Nachrichten gleichzeitig verarbeitet werden können, wird keine `ReceiveMessageBatch`-Aktion ausgeführt.

Die Stapelverarbeitung verteilt die Latenz der Stapelaktion über mehrere Nachrichten in einer Stapelanforderung, anstatt die gesamte Latenz für eine einzelne Nachricht (z. B. eine `SendMessage`-Anforderung) zu akzeptieren. Da jeder Roundtrip mehr Arbeit verrichtet, nutzen Stapelanforderungen Threads und Verbindungen effektiver und verbessern somit den Durchsatz.

Sie können die Stapelverarbeitung mit der horizontalen Skalierung kombinieren, um einen Durchsatz mit weniger Threads, Verbindungen und Anforderungen zu bieten, als dies bei einzelnen Nachrichtenanforderungen der Fall ist. Mit Amazon-SQS-Aktionen im Stapel können Sie bis zu 10 Nachrichten gleichzeitig senden, empfangen oder löschen. Da in Amazon SQS Gebühren nach Anforderung berechnet werden, kann die Stapelverarbeitung wesentlich zur Verringerung der Kosten beitragen.

Mit der Stapelverarbeitung kann eine gewisse Komplexität für Ihre Anwendung einhergehen (z. B. muss die Anwendung Nachrichten vor der Übermittlung sammeln oder gelegentlich länger auf eine Antwort warten). Die Stapelverarbeitung kann in folgenden Fällen jedoch weiterhin effektiv sein:

- Ihre Anwendung erstellt in kurzer Zeit viele Nachrichten, sodass es niemals zu einer sehr langen Verzögerung kommt.
- Anders als typische Nachrichtenproduzenten, die Nachrichten als Antwort auf Ereignisse senden müssen, die sie nicht steuern können, ruft ein Nachrichtenkonsument Nachrichten nach eigenem Ermessen aus einer Warteschlange ab.

> **Important**
>
> Eine Stapelanforderung kann erfolgreich ausgeführt werden, auch wenn einzelne Nachrichten im Stapel fehlgeschlagen sind. Überprüfen Sie nach einer Stapelanforderung stets, ob einzelne Nachrichten nicht zugestellt werden konnten, und führen Sie diese bei Bedarf erneut aus.

Funktionierendes Java-Beispiel für einzelne Operationsanforderungen und Stapelanforderungen

Voraussetzungen

Fügen Sie dem Pfad für Ihre Java-Build-Klasse die Pakete `aws-java-sdk-sqs.jar`, `aws-java-sdk-ec2.jar` und `commons-logging.jar` hinzu. Das folgende Beispiel zeigt diese Abhängigkeiten in der `pom.xml`-Datei eines Maven-Projekts.

```
<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

Das nachstehende Java-Code-Beispiel veranschaulicht ein einfaches Produzent-Konsument-Muster. Der Haupt-Thread ruft eine Reihe von Produzenten- und Konsumenten-Threads auf, die 1-KB-Nachrichten für eine bestimmte Zeit verarbeiten. Dieses Beispiel enthält Produzenten und Konsumenten, die einzelne Operationsanforderungen senden, und solche, die Stapelanforderungen senden.

```
/*
 * 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.
                     */
                    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.
            }
        }
    }
}
```

Überwachen von Volume-Metriken aus der Beispielausführung

Amazon SQS erstellt automatisch Volumen-Metriken für gesendete, empfangene und gelöschte Nachrichten. Sie können auf diese und andere Messwerte über den Tab Überwachung in Ihrer Warteschlange oder auf der CloudWatch Konsole zugreifen.

> **Note**
>
> Nach dem Starten der Warteschlange kann es bis zu 15 Minuten dauern, bis die Metriken verfügbar sind.

Arbeiten mit JMS und Amazon SQS

Die Amazon SQS Java Messaging Library ist eine Java-Message-Service (JMS)-Schnittstelle für Amazon SQS, mit der Sie Amazon SQS in Anwendungen nutzen können, die bereits JMS verwenden. Mithilfe der Schnittstelle können Sie mit nur wenigen Codeänderungen Amazon SQS als JMS-Anbieter verwenden. In Verbindung mit der AWS SDK for Java können Sie mithilfe der Amazon SQS Java Messaging Library JMS-Verbindungen und -Sitzungen sowie Produzenten und Konsumenten erstellen, die Nachrichten an und von Amazon-SQS-Warteschlangen senden und empfangen.

Die Bibliothek unterstützt das Senden und Empfangen von Nachrichten an eine Warteschlange (das point-to-point JMS-Modell) gemäß der JMS 1.1-Spezifikation. Die Bibliothek unterstützt das synchrone Senden von Text-, Byte- und Objektnachrichten an Amazon-SQS-Warteschlangen. Außerdem wird das synchrone und asynchrone Empfangen von Objekten unterstützt.

Weitere Informationen zu den Features der Amazon SQS Java Messaging Library, die die JMS-1.1-Spezifikation unterstützen, finden Sie unter Von Amazon SQS unterstützte JMS 1.1-Implementierungen und in den häufig gestellten Fragen zu Amazon SQS.

Themen

- Voraussetzungen für die Arbeit mit JMS und Amazon SQS
- Erste Schritte mit der Amazon SQS Java Messaging Library
- Verwenden des Java Message Service mit anderen Amazon SQS SQS-Clients
- Funktionierende Java-Beispiele für die Verwendung von JMS mit Amazon SQS SQS-Standardwarteschlangen
- Von Amazon SQS unterstützte JMS 1.1-Implementierungen

Voraussetzungen für die Arbeit mit JMS und Amazon SQS

Folgende Voraussetzungen müssen erfüllt sein, bevor Sie den Vorgang starten:

- SDK für Java

  Es gibt zwei Möglichkeiten, das SDK für Java in Ihr Projekt einzubinden:
  - Laden Sie das SDK für Java herunter und installieren Sie es.
  - Verwenden Sie Maven, um die Amazon SQS Java Messaging Library zu erhalten.

> **Note**
>
> Das SDK für Java ist als Abhängigkeit enthalten.
> Für das SDK für Java und die Amazon SQS Extended Client Library für Java ist das J2SE Development Kit 8.0 oder eine neuere Version erforderlich.

Weitere Informationen zum Herunterladen des SDK für Java finden Sie unter SDK für Java.

- Amazon SQS Java Messaging Library

  Wenn Sie Maven nicht verwenden, müssen Sie das Paket `amazon-sqs-java-messaging-lib.jar` zum Java-Erstellungspfad hinzufügen. Weitere Informationen zum Herunterladen der Bibliothek finden Sie unter Amazon SQS Java Messaging Library.

  > **Note**
  >
  > Die Amazon SQS Java Messaging Library bietet Unterstützung für Maven und das Spring Framework.
  > Codebeispiele, in denen Maven, das Spring Framework und die Amazon SQS Java Messaging Library verwendet werden, finden Sie unter Funktionierende Java-Beispiele für die Verwendung von JMS mit Amazon SQS SQS-Standardwarteschlangen.
  >
  > ```
  > <dependency>
  >   <groupId>com.amazonaws</groupId>
  >   <artifactId>amazon-sqs-java-messaging-lib</artifactId>
  >   <version>1.0.4</version>
  >   <type>jar</type>
  > </dependency>
  > ```

- Amazon-SQS-Warteschlange

  Erstellen Sie eine Warteschlange mit dem AWS Management Console für Amazon SQS, der `CreateQueue` API oder dem verpackten Amazon SQS-Client, der in der Amazon SQS Java Messaging Library enthalten ist.

  - Weitere Informationen zum Erstellen einer Warteschlange mit Amazon SQS mithilfe der AWS Management Console oder der `CreateQueue`-API finden Sie unter Erstellen einer Warteschlange.

- Weitere Informationen zur Verwendung der Amazon SQS Java Messaging Library finden Sie unter Erste Schritte mit der Amazon SQS Java Messaging Library.

Erste Schritte mit der Amazon SQS Java Messaging Library

Verwenden Sie die Codebeispiele in diesem Abschnitt, um mit der Verwendung des Java Message Service (JMS) mit Amazon SQS zu beginnen. In den folgenden Abschnitten erfahren Sie, wie Sie eine JMS-Verbindung und eine Sitzung erstellen sowie eine Nachricht senden und empfangen.

Das integrierte Amazon-SQS-Clientobjekt der Amazon SQS Java Messaging Library prüft, ob bereits eine Amazon-SQS-Warteschlange vorhanden ist. Ist dies nicht der Fall, wird sie vom Client erstellt.

Erstellen einer JMS-Verbindung

1. Erstellen Sie eine Verbindungs-Factory und rufen Sie die Methode `createConnection` für die Factory auf.

```
// 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();
```

   Die Klasse `SQSConnection` erweitert `javax.jms.Connection`. In Verbindung mit den JMS-Standardverbindungsmethoden bietet `SQSConnection` zusätzliche Methoden wie `getAmazonSQSClient` und `getWrappedAmazonSQSClient`. Mit beiden Methoden lassen sich administrative Aufgaben wie das Erstellen neuer Warteschlangen ausführen, die in der JMS-Spezifikation nicht enthalten sind. Die `getWrappedAmazonSQSClient`-Methode stellt jedoch auch eine integrierte Version des Amazon-SQS-Client bereit, die von der aktuellen Verbindung verwendet wird. Der Wrapper wandelt alle Ausnahmen des Clients in eine `JMSException` um. So können diese von bestehendem Code, in dem Vorkommen von `JMSException` erwartet werden, einfacher verwendet werden.

2. Sie können die von `getAmazonSQSClient` und `getWrappedAmazonSQSClient` zurückgegebenen Clientobjekte für administrative Aufgaben verwenden, die in der JMS-Spezifikation nicht enthalten sind (z. B. Erstellen einer neuen Amazon-SQS-Warteschlange).

   Wenn Ihr bestehender Code JMS-Ausnahmen erwartet, müssen Sie `getWrappedAmazonSQSClient` verwenden:

   - Wenn Sie `getWrappedAmazonSQSClient` verwenden, wandelt das zurückgegebene Clientobjekt alle Ausnahmen in JMS-Ausnahmen um.
   - Wenn Sie `getAmazonSQSClient` verwenden, sind alle Ausnahmen Amazon-SQS-Ausnahmen.

Erstellen einer Amazon-SQS-Warteschlange

Das integrierte Clientobjekt prüft, ob eine Amazon-SQS-Warteschlange vorhanden ist.

Ist dies nicht der Fall, wird sie vom Client erstellt. Ist bereits eine Warteschlange vorhanden, gibt die Funktion nichts zurück. Weitere Informationen finden Sie im Abschnitt "Create the queue if needed" im Beispiel TextMessageSender.java.

So erstellen Sie eine Standardwarteschlange

```
// 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");
}
```

So erstellen Sie eine FIFO-Warteschlange

```
// 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**
>
> Der Name einer FIFO-Warteschlange muss mit dem Suffix `.fifo` enden.
> Weitere Informationen zum `ContentBasedDeduplication`-Attribut finden Sie unter Exactly-Once-Verarbeitung in Amazon SQS.

Synchrones Senden von Nachrichten

1. Wenn die Verbindung und die zugrundeliegende Amazon-SQS-Warteschlange bereit sind, erstellen Sie eine nicht transaktionsorientierte JMS-Sitzung mit dem Modus `AUTO_ACKNOWLEDGE`.

   ```
   // Create the nontransacted session with AUTO_ACKNOWLEDGE mode
   Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
   ```

2. Erstellen Sie zum Senden einer Textnachricht an die Warteschlange eine JMS-Warteschlangenidentität und einen Nachrichtenproduzenten.

   ```
   // 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. Erstellen Sie eine Textnachricht und senden Sie sie an die Warteschlange.

   - Um eine Nachricht an eine Standard-Warteschlange zu senden, müssen Sie keine weiteren Parameter festlegen.

     ```
     // Create the text message
     TextMessage message = session.createTextMessage("Hello World!");

     // Send the message
     producer.send(message);
     System.out.println("JMS Message " + message.getJMSMessageID());
     ```

- Um eine Nachricht an eine FIFO-Warteschlange zu senden, müssen Sie die Nachrichtengruppen-ID festlegen. Darüber hinaus können Sie eine Nachrichtendeduplizierungs-ID festlegen. Weitere Informationen finden Sie unter Die wichtigsten Begriffe von 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"));
```

Synchrones Empfangen von Nachrichten

1. Erstellen Sie zum Empfangen von Nachrichten einen Konsumenten für dieselbe Warteschlange und rufen Sie die Methode `start` auf.

   Sie können die Methode `start` für die Verbindung jederzeit aufrufen. Der Konsument beginnt jedoch erst mit dem Abrufen von Nachrichten, wenn Sie die Methode aufrufen.

```
// Create a consumer for the 'MyQueue'
MessageConsumer consumer = session.createConsumer(queue);
// Start receiving incoming messages
connection.start();
```

2. Rufen Sie die Methode `receive` mit einer Zeitbeschränkung von 1 Sekunde auf dem Konsumenten auf und geben Sie den Inhalt der empfangenen Nachricht aus.

   - Nachdem Sie eine Nachricht aus einer Standard-Warteschlange abgerufen haben, können Sie auf den Inhalt der Nachricht zugreifen.

```
// 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());
}
```

- Nachdem Sie eine Nachricht aus einer FIFO-Warteschlange abgerufen haben, können Sie auf den Inhalt der Nachricht sowie auf andere FIFO-spezifische Nachrichtenattribute wie die Nachrichtengruppen-ID, die Nachrichtendeduplizierungs-ID und die Sequenznummer zugreifen. Weitere Informationen finden Sie unter Die wichtigsten Begriffe von 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. Schließen Sie die Verbindung und die Sitzung.

```
// Close the connection (and the session).
connection.close();
```

Die Ausgabe sieht folgendermaßen oder ähnlich aus:

```
JMS Message ID:8example-588b-44e5-bbcf-d816example2
Received: Hello World!
```

> **Note**
>
> Sie können das Spring Framework zum Initialisieren dieser Objekte verwenden. Weitere Informationen finden Sie unter `SpringExampleConfiguration.xml`, `SpringExample.java`. Informationen zu den anderen unterstützenden Klassen finden Sie unter `ExampleConfiguration.java` und `ExampleCommon.java` im Abschnitt Funktionierende Java-Beispiele für die Verwendung von JMS mit Amazon SQS SQS-Standardwarteschlangen.

Umfassende Beispiele für das Senden und Empfangen von Objekten finden Sie unter TextMessageSender.java und SyncMessageReceiver.java.

Asynchrones Empfangen von Nachrichten

Im Beispiel unter Erste Schritte mit der Amazon SQS Java Messaging Library wird eine Nachricht an `MyQueue` gesendet und synchron empfangen.

Im folgenden Beispiel wird gezeigt, wie Nachrichten mithilfe eines Listeners asynchron empfangen werden können.

1. Implementieren Sie die `MessageListener`-Schnittstelle.

```
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 (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

   Beim Empfang einer Nachricht wird die Methode `onMessage` der `MessageListener`-Schnittstelle aufgerufen. In dieser Implementierung des Listeners wird der in der Nachricht gespeicherte Text ausgegeben.

2. Statt die Methode `receive` auf dem Konsumenten explizit aufzurufen, richten Sie den Nachrichten-Listener des Konsumenten auf einer Instance der `MyListener`-Implementierung ein. Der Haupt-Thread wartet eine Sekunde lang.

```
// 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);
```

Die restlichen Schritte sind identisch wie im Beispiel Erste Schritte mit der Amazon SQS Java Messaging Library. Ein umfassendes Beispiel für einen asynchronen Konsumenten finden Sie unter `AsyncMessageReceiver.java` in den Funktionierende Java-Beispiele für die Verwendung von JMS mit Amazon SQS SQS-Standardwarteschlangen.

Die Ausgabe für dieses Beispiel sieht in etwa wie folgt aus:

```
JMS Message ID:8example-588b-44e5-bbcf-d816example2
Received: Hello World!
```

Verwenden des Client-Bestätigungsmodus

In dem Beispiel in Erste Schritte mit der Amazon SQS Java Messaging Library wird der `AUTO_ACKNOWLEDGE`-Modus verwendet, um jede empfangene Nachricht automatisch zu bestätigen (und somit aus der zugrundeliegenden Amazon-SQS-Warteschlange zu löschen).

1. Um die Nachrichten nach der Verarbeitung explizit zu bestätigen, müssen Sie die Sitzung im `CLIENT_ACKNOWLEDGE`-Modus erstellen.

```
// Create the non-transacted session with CLIENT_ACKNOWLEDGE mode.
Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
```

2. Zeigen Sie die Nachricht nach dem Empfangen an und bestätigen Sie sie explizit.

```
// 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**
>
> Wenn eine Nachricht in diesem Modus bestätigt wird, werden alle Nachrichten, die vor dieser Nachricht empfangen wurden, ebenfalls implizit bestätigt. Wenn Sie beispielsweise 10 Nachrichten empfangen haben und nur die 10. Nachricht bestätigen (in der Reihenfolge, in der die Nachrichten empfangen wurden) werden die vorherigen 9 Nachrichten ebenfalls bestätigt.

Die restlichen Schritte sind identisch wie im Beispiel Erste Schritte mit der Amazon SQS Java Messaging Library. Ein umfassendes Beispiel für einen synchronen Konsumenten im Client-Bestätigungsmodus finden Sie unter `SyncMessageReceiverClientAcknowledge.java` in den Funktionierende Java-Beispiele für die Verwendung von JMS mit Amazon SQS SQS-Standardwarteschlangen.

Die Ausgabe für dieses Beispiel sieht in etwa wie folgt aus:

```
JMS Message ID:4example-aa0e-403f-b6df-5e02example5
Received: Hello World!
Acknowledged: ID:4example-aa0e-403f-b6df-5e02example5
```

Verwenden des ungeordneten Bestätigungsmodus

Wenn Sie den `CLIENT_ACKNOWLEDGE`-Modus verwenden, werden alle Nachrichten, die vor einer explizit bestätigten Nachricht empfangen wurden, automatisch bestätigt. Weitere Informationen finden Sie unter Verwenden des Client-Bestätigungsmodus.

Die Amazon SQS Java Messaging Library stellt einen weiteren Bestätigungsmodus bereit. Wenn Sie den `UNORDERED_ACKNOWLEDGE`-Modus verwenden, müssen alle empfangenen Nachrichten

unabhängig von der Empfangsreihenfolge einzeln und explizit vom Client bestätigt werden. Erstellen Sie hierzu eine Sitzung im UNORDERED_ACKNOWLEDGE-Modus.

```
// Create the non-transacted session with UNORDERED_ACKNOWLEDGE mode.
Session session = connection.createSession(false, SQSSession.UNORDERED_ACKNOWLEDGE);
```

Die restlichen Schritte sind identisch wie im Beispiel Verwenden des Client-Bestätigungsmodus. Ein umfassendes Beispiel für einen synchronen Konsumenten im UNORDERED_ACKNOWLEDGE-Modus finden Sie unter SyncMessageReceiverUnorderedAcknowledge.java.

In diesem Beispiel sieht die Ausgabe in etwa wie folgt aus:

```
JMS Message ID:dexample-73ad-4adb-bc6c-4357example7
Received: Hello World!
Acknowledged: ID:dexample-73ad-4adb-bc6c-4357example7
```

Verwenden des Java Message Service mit anderen Amazon SQS SQS-Clients

Die Verwendung des Amazon SQS Java Message Service (JMS) -Clients mit dem AWS SDK begrenzt die Amazon SQS SQS-Nachrichtengröße auf 256 KB. Sie können jedoch mit einem beliebigen Amazon-SQS-Client einen JMS-Anbieter erstellen. Verwenden Sie beispielsweise den JMS-Client mit der Amazon SQS Extended Client Library für Java, um eine Amazon-SQS-Nachricht zu senden, die einen Verweis auf eine Nachrichtennutzlast (bis zu 2 GB) in Amazon S3 enthält. Weitere Informationen finden Sie unter Verwaltung großer Amazon SQS SQS-Nachrichten mit Java und Amazon S3.

Im folgenden Java-Codebeispiel wird der JMS-Anbieter für die erweiterte Clientbibliothek erstellt:

```
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);
```

Im folgenden Java-Codebeispiel wird die Verbindungs-Factory erstellt:

```
// 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();
```

Funktionierende Java-Beispiele für die Verwendung von JMS mit Amazon SQS SQS-Standardwarteschlangen

Das folgende Codebeispiel veranschaulicht die Verwendung des Java Message Service (JMS) mit Amazon-SQS-Standard-Warteschlangen. Weitere Informationen zur Arbeit mit FIFO-Warteschlangen finden Sie unter So erstellen Sie eine FIFO-Warteschlange, Synchrones Senden von Nachrichten und Synchrones Empfangen von Nachrichten. (Der synchrone Empfang von Nachrichten ist für Standard- und FIFO-Warteschlangen identisch. Nachrichten in FIFO-Warteschlangen enthalten jedoch mehr Attribute.)

ExampleConfiguration.java

Das folgende Codebeispiel für Java SDK v 1.x legt den Standard-Warteschlangennamen, die Region und die Anmeldeinformationen fest, die mit den anderen Java-Beispielen verwendet werden sollen.

```
/*
 * 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

Im folgenden Java-Codebeispiel wird ein Textnachrichtenproduzent erstellt.

```
/*
 * 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 JMSException {
        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 (JMSException e) {
            System.err.println( "Failed sending message: " + e.getMessage() );
            e.printStackTrace();
        }
    }
}
```

SyncMessageReceiver.java

Im folgenden Java-Codebeispiel wird ein synchroner Nachrichtenkonsument erstellt.

```
/*
 * 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 JMSException {
    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 (JMSException e) {
        System.err.println( "Error receiving from SQS: " + e.getMessage() );
        e.printStackTrace();
    }
}
}
```

AsyncMessageReceiver.java

Im folgenden Java-Codebeispiel wird ein asynchroner Nachrichtenkonsument erstellt.

```
/*
 * 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 JMSException, 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 (JMSException e) {
                System.err.println( "Error processing message: " + e.getMessage() );
                e.printStackTrace();
            }
        }
    }
}
```

SyncMessageReceiverClientAcknowledge.java

Im folgenden Java-Codebeispiel wird ein synchroner Konsument im Client-Bestätigungsmodus erstellt.

```
/*
 * 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 JMSException, 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 JMSException
     */
    private static void sendMessage(MessageProducer producer, Session session, String
messageText) throws JMSException {
        // 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 JMSException
     */
    private static void receiveMessage(MessageConsumer consumer, boolean acknowledge)
throws JMSException {
        // 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

Im folgenden Java-Codebeispiel wird ein synchroner Konsument im ungeordneten Bestätigungsmodus erstellt.

```
/*
 * 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 JMSException, 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 JMSException
     */
    private static void sendMessage(MessageProducer producer, Session session, String
messageText) throws JMSException {
        // 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 JMSException
     */
    private static void receiveMessage(MessageConsumer consumer, boolean acknowledge)
 throws JMSException {
        // 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

Das folgende XML-Codebeispiel ist eine Bean-Konfigurationsdatei für 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

Im folgenden Java-Codebeispiel wird die Bean-Konfigurationsdatei verwendet, um Ihre Objekte zu initialisieren.

```
/*
 * 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 JMSException {
        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 (JMSException e) {
            System.err.println( "Error receiving from SQS: " + e.getMessage() );
            e.printStackTrace();
        }
    }
}
```

ExampleCommon.java

Im folgenden Java-Codebeispiel wird geprüft, ob bereits eine Amazon-SQS-Warteschlange vorhanden ist, und gegebenenfalls eine neue Warteschlange erstellt. Außerdem ist Beispiel-Code für die Protokollierung enthalten.

```
/*
 * 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 JMSException {
       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 JMSException {
       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() );
       }
   }
```

```
}
```

Von Amazon SQS unterstützte JMS 1.1-Implementierungen

Die Amazon SQS Java Messaging Library unterstützt die folgenden JMS-1.1-Implementierungen. Weitere Informationen zu den unterstützten Features und Fähigkeiten der Amazon SQS Java Messaging Library finden Sie in den Häufig gestellten Fragen zu Amazon SQS.

Unterstützte gängige Schnittstellen

- `Connection`
- `ConnectionFactory`
- `Destination`
- `Session`
- `MessageConsumer`
- `MessageProducer`

Unterstützte Nachrichtentypen

- `ByteMessage`
- `ObjectMessage`
- `TextMessage`

Unterstützte Nachrichtenbestätigungsmodi

- `AUTO_ACKNOWLEDGE`
- `CLIENT_ACKNOWLEDGE`
- `DUPS_OK_ACKNOWLEDGE`
- `UNORDERED_ACKNOWLEDGE`

> **Note**
>
> Der `UNORDERED_ACKNOWLEDGE`-Modus ist nicht Bestandteil der JMS 1.1-Spezifikation. Mithilfe dieses Modus kann Amazon SQS einem JMS-Client explizit die Bestätigung einer Nachricht erlauben.

JMS-definierte Kopfzeilen und reservierte Eigenschaften

Für den Nachrichtenversand

Beim Senden von Nachrichten können Sie die folgenden Kopfzeilen und Eigenschaften für einzelne Nachrichten festlegen:

- `JMSXGroupID` (erforderlich für FIFO-Warteschlangen, nicht zulässig für Standard-Warteschlangen)
- `JMS_SQS_DeduplicationId` (optional für FIFO-Warteschlangen, nicht zulässig für Standard-Warteschlangen)

Nach dem Senden von Nachrichten legt Amazon SQS die folgenden Kopfzeilen und Eigenschaften für einzelne Nachrichten fest:

- `JMSMessageID`
- `JMS_SQS_SequenceNumber` (nur für FIFO-Warteschlangen)

Für den Nachrichtenempfang

Beim Empfang von Nachrichten legt Amazon SQS die folgenden Kopfzeilen und Eigenschaften für einzelne Nachrichten fest:

- `JMSDestination`
- `JMSMessageID`
- `JMSRedelivered`
- `JMSXDeliveryCount`
- `JMSXGroupID` (nur für FIFO-Warteschlangen)
- `JMS_SQS_DeduplicationId` (nur für FIFO-Warteschlangen)
- `JMS_SQS_SequenceNumber` (nur für FIFO-Warteschlangen)

Tutorials zu Amazon SQS

Dieser Abschnitt enthält Tutorials zum Kennenlernen von Amazon-SQS-Features und -Funktionalität.

Themen

- Erstellen einer Amazon SQS SQS-Warteschlange mit AWS CloudFormation
- Tutorial: Senden einer Nachricht an eine Amazon-SQS-Warteschlange aus Amazon Virtual Private Cloud

Erstellen einer Amazon SQS SQS-Warteschlange mit AWS CloudFormation

Sie können die AWS CloudFormation Konsole und eine JSON- (oder YAML-) Vorlage verwenden, um eine Amazon SQS SQS-Warteschlange zu erstellen. Weitere Informationen finden Sie unter Arbeiten mit AWS CloudFormation -Vorlagen und unter der `AWS::SQS::Queue`-Ressource im AWS CloudFormation -Benutzerhandbuch.

Wird verwendet, AWS CloudFormation um eine Amazon SQS SQS-Warteschlange zu erstellen.

1. Kopieren Sie den JSON-Code in eine Datei mit dem Namen `MyQueue.json`. Lassen Sie zum Erstellen einer Standard-Warteschlange die Eigenschaften `FifoQueue` und `ContentBasedDeduplication` weg. Weitere Informationen zur inhaltsbasierten Deduplizierung finden Sie unter Exactly-Once-Verarbeitung in Amazon SQS.

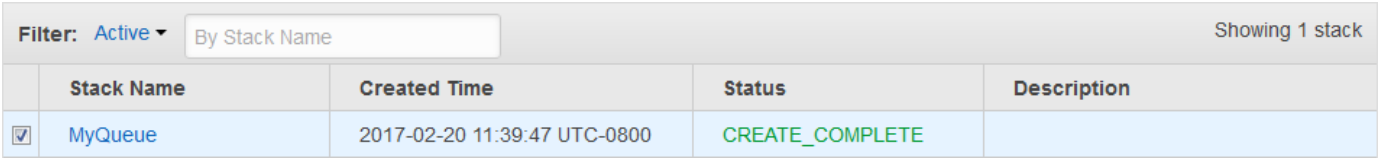
   > **Note**
   >
   > Der Name einer FIFO-Warteschlange muss mit dem Suffix `.fifo` enden.

```
{
   "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. Melden Sie sich bei der AWS CloudFormation -Konsole an und wählen Sie dann Create Stack (Stack erstellen) aus.
3. Wählen Sie im Bereich Specify Template (Vorlage angeben) die Option Upload a template file (Vorlagendatei hochladen) aus, wählen Sie Ihre `MyQueue.json`-Datei aus und klicken Sie dann auf Next (Weiter).
4. Geben Sie auf der Seite Specify Details `MyQueue` unter Stack Name ein und wählen Sie Next aus.
5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Klicken Sie auf der Seite Review auf Create.

AWS CloudFormation beginnt mit der Erstellung des `MyQueue` Stacks und zeigt den Status CREATE_IN_PROGRESS an. Wenn der Prozess abgeschlossen ist, zeigt AWS CloudFormation den Status CREATE_COMPLETE an.



7. (Optional) Um den Namen, die URL und den ARN der Warteschlange anzuzeigen, wählen Sie den Namen des Stacks aus und erweitern Sie auf der nächsten Seite den Abschnitt Outputs.

Tutorial: Senden einer Nachricht an eine Amazon-SQS-Warteschlange aus Amazon Virtual Private Cloud

In diesem Tutorial erfahren Sie, wie Sie Nachrichten über ein sicheres, privates Netzwerk zu einer Amazon-SQS-Warteschlange senden. Dieses Netzwerk besteht aus einer VPC, die eine Amazon-EC2-Instance enthält. Die Instance stellt über einen Schnittstellen-VPC-Endpunkt eine Verbindung mit Amazon SQS her, so dass Sie eine Verbindung mit der Amazon-EC2-Instance herstellen und Nachrichten an die Amazon-SQS-Warteschlange senden können, auch wenn das Netzwerk nicht mit dem öffentlichen Internet verbunden ist. Weitere Informationen finden Sie unter Endpunkte von Amazon Virtual Private Cloud für Amazon SQS.

Important

- Sie können Amazon Virtual Private Cloud nur mit HTTPS-Amazon-SQS-Endpunkten verwenden.
- Wenn Sie Amazon SQS konfigurieren, um Nachrichten von Amazon VPC zu senden, müssen Sie privates DNS aktivieren und Endpunkte im `sqs.us-east-2.amazonaws.com`-Format angeben.
- Ein privates DNS unterstützt keine Legacy-Endpunkte wie z. B. `queue.amazonaws.com` oder `us-east-2.queue.amazonaws.com`.

Themen

- Schritt 1: Erstellen eines Amazon EC2-Schlüsselpaares

- Schritt 2: Ressourcen erstellen AWS
- Schritt 3: Bestätigen, dass Ihre EC2-Instance nicht öffentlich zugänglich ist
- Schritt 4: Erstellen eines Amazon-VPC-Endpunkts für Amazon SQS
- Schritt 5: Senden einer Nachricht an Ihre Amazon-SQS-Warteschlange

Schritt 1: Erstellen eines Amazon EC2-Schlüsselpaares

Ein Schlüsselpaar ermöglicht Ihnen die Verbindung mit einer EC2-Instance. Es besteht aus einem öffentlichen Schlüssel, mit dem Ihre Anmeldeinformationen verschlüsselt werden, und einem privaten Schlüssel, mit dem sie entschlüsselt werden.

1. Melden Sie sich bei der Amazon-EC2-Konsole an.
2. Klicken Sie im Navigationsmenü unter Network & Security (Netzwerk und Sicherheit) auf Key Pairs (Schlüsselpaare).
3. Wählen Sie Create Key Pair aus.
4. Geben Sie im Dialogfeld Create Key Pair (Schlüsselpaar erstellen) im Feld Key pair name (Schlüsselpaarname) als Namen `SQS-VPCE-Tutorial-Key-Pair` ein und klicken Sie auf Create (Erstellen).
5. Ihr Browser lädt die private Schlüsseldatei `SQS-VPCE-Tutorial-Key-Pair.pem` automatisch herunter.

   > **Important**
   >
   > Speichern Sie diese Datei an einem sicheren Ort. EC2 generiert eine `.pem`-Datei für dasselbe Schlüsselpaar kein zweites Mal.

6. Um einem SSH-Client das Herstellen einer Verbindung mit Ihrer EC2-Instance zu ermöglichen, legen Sie die Berechtigungen für Ihre private Schlüsseldatei so fest, dass nur Ihrem Benutzer Leseberechtigungen für sie gewährt werden, zum Beispiel:

   ```
   chmod 400 SQS-VPCE-Tutorial-Key-Pair.pem
   ```

Schritt 2: Ressourcen erstellen AWS

Um die erforderliche Infrastruktur einzurichten, müssen Sie eine AWS CloudFormation Vorlage verwenden, bei der es sich um einen Plan für die Erstellung eines Stacks handelt, der aus AWS Ressourcen wie Amazon EC2 EC2-Instances und Amazon SQS SQS-Warteschlangen besteht.

Der Stack für dieses Tutorial enthält die folgenden Ressourcen:

- Eine VPC und die zugehörigen Netzwerkressourcen, einschließlich eines Subnetzes, einer Sicherheitsgruppe, eines Internet-Gateways und einer Routing-Tabelle.
- Eine im VPC-Subnetz gestartete Amazon-EC2-Instance
- Eine Amazon-SQS-Warteschlange

1. Laden Sie die Vorlage mit dem AWS CloudFormation Namen von herunter. `SQS-VPCE-Tutorial-CloudFormation.yaml`GitHub
2. Melden Sie sich an der AWS CloudFormation -Konsole an.
3. Wählen Sie Stapel erstellen aus.
4. Wählen Sie auf der Seite Select Template (Vorlage auswählen) die Option Upload a template to Amazon S3 (Eine Vorlage zu Amazon S3 hochladen). Wählen Sie dann die Datei `SQS-VPCE-SQS-Tutorial-CloudFormation.yaml` aus und klicken Sie auf Next (Weiter).
5. Führen Sie auf der Seite Specify DB Details (Festlegen von DB-Detail) die folgenden Schritte aus:

   a. Geben Sie unter Stack name (Stack-Name) `SQS-VPCE-Tutorial-Stack` ein.

   b. Wählen Sie für KeyNameSQS-VPCE-Tutorial-Key-Pair.

   c. Wählen Sie Weiter aus.
6. Wählen Sie auf der Seite Optionen Weiter aus.
7. Wählen Sie auf der Seite Überprüfen im Abschnitt Funktionen die Option Ich bestätige, dass möglicherweise IAM-Ressourcen mit benutzerdefinierten Namen erstellt werden. AWS CloudFormation , und wählen Sie dann Erstellen aus.

AWS CloudFormation beginnt mit der Erstellung des Stacks und zeigt den Status CREATE_IN_PROGRESS an. Wenn der Prozess abgeschlossen ist, zeigt AWS CloudFormation den Status CREATE_COMPLETE an.

Schritt 3: Bestätigen, dass Ihre EC2-Instance nicht öffentlich zugänglich ist

Ihre AWS CloudFormation Vorlage startet eine EC2-Instance, die `SQS-VPCE-Tutorial-EC2-Instance` in Ihre VPC benannt ist. Diese EC2-Instance lässt keinen ausgehenden Datenverkehr zu und kann keine Nachrichten an Amazon SQS senden. Um dies zu überprüfen, müssen Sie eine Verbindung mit der Instance herstellen. Versuchen Sie dann, eine Verbindung mit einem öffentlichen Endpunkt herzustellen und eine Nachricht an Amazon SQS zu senden.

1. Melden Sie sich bei der Amazon-EC2-Konsole an.
2. Wählen Sie im Navigationsmenü unter Instances die Option Instances.
3. Wählen Sie SQS-VPCE - aus. Tutorial-EC2Instance
4. Kopieren Sie den Hostnamen unter Public DNS (IPv4) (Öffentliches DNS (IPv4)), z. B. ec2-203-0-113-0.us-west-2.compute.amazonaws.com.
5. Stellen Sie über das Verzeichnis, in dem sich das Schlüsselpaar befindet, das Sie zuvor erstellt haben, mit dem folgenden Befehl eine Verbindung mit der Instance her, zum Beispiel:

   ```
   ssh -i SQS-VPCE-Tutorial-Key-Pair.pem ec2-user@ec2-203-0-113-0.us-east-2.compute.amazonaws.com
   ```

6. Versuchen Sie, eine Verbindung mit einem öffentlichen Endpunkt herzustellen, zum Beispiel:

   ```
   ping amazon.com
   ```

   Der Verbindungsversuch schlägt erwartungsgemäß fehl.
7. Melden Sie sich bei der Amazon-SQS-Konsole an.
8. Wählen Sie aus der Warteschlangenliste die Warteschlange aus, die mit Ihrer AWS CloudFormation Vorlage erstellt wurde, z. B. VPCE-SQS-Tutorial-Stack-CFQueue-1ABCDEFGH2IJK.
9. Kopieren Sie in der Tabelle Details die URL, zum Beispiel https://sqs.us-east-2.amazonaws.com/123456789012/.
10. Versuchen Sie, über Ihre EC2-Instance mit dem folgenden Befehl eine Nachricht an die Warteschlange zu veröffentlichen, zum Beispiel:

    ```
    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."
    ```

Der Sendeversuch schlägt erwartungsgemäß fehl.

> **Important**
>
> Später, wenn Sie einen VPC-Endpunkt für Amazon SQS erstellen, wird Ihr Sendeversuch erfolgreich sein.

Schritt 4: Erstellen eines Amazon-VPC-Endpunkts für Amazon SQS

Zum Herstellen einer Verbindung Ihrer VPC mit Amazon SQS müssen Sie einen Schnittstellen-VPC-Endpunkt definieren. Nachdem Sie den Endpunkt hinzugefügt haben, können Sie die Amazon-SQS-API von der EC2-Instance in Ihrer VPC verwenden. Auf diese Weise können Sie Nachrichten an eine Warteschlange innerhalb des AWS Netzwerks senden, ohne das öffentliche Internet zu überqueren.

> **Note**
>
> Die EC2-Instance hat immer noch keinen Zugriff auf andere AWS Dienste und Endpunkte im Internet.

1. Melden Sie sich bei der Amazon VPC-Konsole an.
2. Wählen Sie im Navigationsmenü Endpoints (Endpunkte).
3. Klicken Sie auf Endpunkt erstellen.
4. Wählen Sie auf der Seite Endpunkt erstellen für Servicename den Servicenamen für Amazon SQS aus.

   > **Note**
   >
   > Die Dienstnamen variieren je nach aktueller AWS Region. Zum Beispiel: Wenn Sie sich in USA Ost (Ohio) befinden, ist der Servicename com.amazonaws.***us-east-2***.sqs.

5. Wählen Sie für VPC die Option SQS-VPCE-Tutorial-VPC.
6. Wählen Sie für Subnets (Subnetze) das Subnetz aus, dessen Subnet ID (Subnetz-ID) den Wert SQS-VPCE-Tutorial-Subnet enthält.

7. Wählen Sie für Security group (Sicherheitsgruppe) die Option Select security groups (Sicherheitsgruppen auswählen). Wählen Sie dann die Sicherheitsgruppe aus, deren Group Name (Gruppenname) den Wert SQS VPCE Tutorial Security Group enthält.
8. Wählen Sie Endpunkt erstellen aus.

   Der Schnittstellen-VPC-Endpunkt wird erstellt und seine ID wird angezeigt, z. B. vpce-0ab1cdef2ghi3j456k.
9. Klicken Sie auf Schließen.

   Die Amazon-VPC-Konsole öffnet die Seite Endpunkte.

Amazon VPC beginnt mit der Erstellung des Endpunkts und zeigt als Status ausstehend an. Wenn der Prozess abgeschlossen ist, zeigt Amazon VPC als Status verfügbar an.

Schritt 5: Senden einer Nachricht an Ihre Amazon-SQS-Warteschlange

Da Ihre VPC jetzt einen Endpunkt für Amazon SQS erhält, können Sie sich mit Ihrer EC2-Instance verbinden und Nachrichten zu Ihrer Warteschlange senden.

1. Verbinden Sie sich erneut mit Ihrer EC2-Instance, z. B.:

   ```
   ssh -i SQS-VPCE-Tutorial-Key-Pair.pem ec2-user@ec2-203-0-113-0.us-
   east-2.compute.amazonaws.com
   ```
2. Versuchen Sie erneut, mit dem folgenden Befehl eine Nachricht in der Warteschlange zu veröffentlichen, zum Beispiel:

   ```
   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."
   ```

   Der Sendeversuch ist erfolgreich und der MD5 Digest des Nachrichtentexts und die Nachrichten-ID werden angezeigt, zum Beispiel:

   ```
   {
    "MD5OfMessageBody": "a1bcd2ef3g45hi678j90klmn12p34qr5",
    "MessageId": "12345a67-8901-2345-bc67-d890123e45fg"
   }
   ```

Informationen zum Empfangen und Löschen der Nachricht aus der Warteschlange, die mit Ihrer AWS CloudFormation Vorlage erstellt wurde (z. B. VPCE-SQS-Tutorial-Stack-CFQueue-1ABCDEFGH2IJK), finden Sie unter. Empfangen und Löschen einer Nachricht in Amazon SQS

Weitere Informationen zum Löschen Ihrer Ressourcen finden Sie unter:

- Löschen eines VPC-Endpunkts im Amazon-VPC-Benutzerhandbuch
- Löschen einer Amazon SQS SQS-Warteschlange
- Beenden Sie Ihre Instance im Amazon EC2 EC2-Benutzerhandbuch
- Löschen Ihrer VPC im Amazon-VPC-Benutzerhandbuch
- Löschen eines Stacks auf der AWS CloudFormation Konsole im AWS CloudFormation Benutzerhandbuch
- Löschen Ihres Schlüsselpaars im Amazon EC2 EC2-Benutzerhandbuch

Behebung von Problemen in Amazon SQS

Die folgenden Themen enthalten Hinweise zur Behebung häufiger Fehler und Probleme, die bei der Verwendung der Amazon SQS-Konsole, der Amazon SQS SQS-API oder anderer Tools mit Amazon SQS auftreten können. Wenn Sie auf ein Problem stoßen, das hier nicht aufgeführt ist, können Sie die Schaltfläche Feedback auf dieser Seite verwenden, um es zu melden.

Weitere Tipps zur Fehlerbehebung und Antworten auf häufig gestellte Supportfragen finden Sie im AWS - Wissenscenter.

Themen

- Behebung eines Fehlers „Zugriff verweigert“ in Amazon SQS
- Behebung von Amazon SQS SQS-API-Fehlern
- Behebung von Problemen mit der Amazon SQS SQS-Warteschlange für unzustellbare Briefe und DLQ Redrive
- Behebung von Problemen mit der FIFO-Drosselung in Amazon SQS
- Problembehandlung bei Nachrichten, die bei einem Amazon SQS ReceiveMessage SQS-API-Aufruf nicht zurückgegeben wurden
- Behebung Amazon SQS SQS-Netzwerkfehlern
- Problembehandlung bei Warteschlangen von Amazon Simple Queue Service mit AWS X-Ray

Behebung eines Fehlers „Zugriff verweigert“ in Amazon SQS

Die folgenden Themen behandeln die häufigsten Ursachen `AccessDenied` oder `AccessDeniedException` Fehler Amazon SQS SQS-API-Aufrufen. Weitere Informationen zur Behebung dieser Fehler finden Sie unter Wie behebe ich "" - oder AccessDenied "AccessDeniedAusnahme“ -Fehler Amazon SQS SQS-API-Aufrufen? im AWS Knowledge Center-Leitfaden.

Beispiele für Fehlermeldungen:

```
An error occurred (AccessDenied) when calling the SendMessage operation: Access to
        the resource https://sqs.us-east-1.amazonaws.com/ is denied.
```

- oder -

```
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.
```

Themen

- Amazon SQS SQS-Warteschlangenrichtlinie und IAM-Richtlinie
- AWS Key Management Service Berechtigungen
- VPC-Endpunktrichtlinie
- Richtlinie zur Dienstkontrolle der Organisation

Amazon SQS SQS-Warteschlangenrichtlinie und IAM-Richtlinie

Gehen Sie wie folgt vor, um zu überprüfen, ob der Anforderer über die erforderlichen Berechtigungen zur Durchführung eines Amazon SQS SQS-Vorgangs verfügt:

- Identifizieren Sie den IAM-Prinzipal, der den Amazon SQS SQS-API-Aufruf durchführt. Wenn der IAM-Principal von demselben Konto stammt, müssen entweder die Amazon SQS SQS-Warteschlangenrichtlinie oder die AWS Identity and Access Management (IAM) -Richtlinie Berechtigungen enthalten, um den Zugriff für die Aktion explizit zuzulassen.
- Wenn der Principal eine IAM-Entität ist:
  - Sie können Ihren IAM-Benutzer oder Ihre IAM-Rolle identifizieren, indem Sie die obere rechte Ecke von überprüfen oder den AWS Management Console Befehl verwenden. `aws sts get-caller-identity`
  - Prüfen Sie die IAM-Richtlinien im Zusammenhang mit dem IAM-Benutzer oder der IAM-Rolle. Sie können eine der folgenden Methoden verwenden:
    - Testen Sie IAM-Richtlinien mit dem IAM Policy Simulator.
    - Überprüfen der verschiedenen IAM-Richtlinientypen
  - Bearbeiten Sie gegebenenfalls Ihre IAM-Benutzerrichtlinie.
  - Überprüfen Sie die Warteschlangenrichtlinie und bearbeiten Sie sie gegebenenfalls.
- Wenn der Principal ein AWS Service ist, muss die Amazon SQS SQS-Warteschlangenrichtlinie den Zugriff explizit zulassen.

- Wenn es sich bei dem Principal um einen kontoübergreifenden Principal handelt, müssen sowohl die Amazon SQS SQS-Warteschlangenrichtlinie als auch die IAM-Richtlinie den Zugriff explizit zulassen.
- Wenn die Richtlinie ein Bedingungselement verwendet, überprüfen Sie, ob die Bedingung den Zugriff einschränkt.

> **Important**
>
> Eine ausdrückliche Ablehnung in einer der Richtlinien hat Vorrang vor einer ausdrücklichen Zulassung. Hier sind einige grundlegende Beispiele für Amazon SQS SQS-Richtlinien.

AWS Key Management Service Berechtigungen

Wenn in Ihrer Amazon SQS SQS-Warteschlange serverseitige Verschlüsselung (SSE) aktiviert ist und ein Kunde verwaltet wird AWS KMS key, müssen sowohl Produzenten als auch Verbrauchern Berechtigungen erteilt werden. Um zu überprüfen, ob eine Warteschlange verschlüsselt ist, können Sie das **`GetQueueAttributes`** **`KmsMasterKeyId`**API-Attribut oder die Warteschlangenkonsole unter Verschlüsselung verwenden.

- Erforderliche Berechtigungen für Produzenten:

```
{
"Effect": "Allow",
"Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
],
"Resource": "<Key ARN>"
}
```

- Erforderliche Berechtigungen für Verbraucher:

```
{
"Effect": "Allow",
"Action": [
    "kms:Decrypt"
],
"Resource": "<Key ARN>"
```

```
}
```

- Erforderliche Berechtigungen für den kontoübergreifenden Zugriff:

```
{
"Effect": "Allow",
"Action": [
    "kms:DescribeKey",
    "kms:Decrypt",
    "kms:ReEncrypt",
    "kms:GenerateDataKey"
],
"Resource": "<Key ARN>"
}
```

Sie können eine der folgenden Methoden verwenden, um die Verschlüsselung für eine Amazon SQS SQS-Warteschlange zu aktivieren:

- SSE-Amazon SQS (Verschlüsselungsschlüssel, der vom Amazon SQS-Service erstellt und verwaltet wird.)
- AWS verwalteter Standardschlüssel (alias/aws/sqs)
- Kundenverwalteter Schlüssel

Wenn Sie jedoch einen AWS-verwalteten KMS-Schlüssel verwenden, können Sie die Standardschlüsselrichtlinie nicht ändern. Verwenden Sie daher den vom Kunden verwalteten Schlüssel, um Zugriff auf andere Dienste und kontenübergreifende Konten zu gewähren. Auf diese Weise können Sie die Schlüsselrichtlinie bearbeiten.

VPC-Endpunktrichtlinie

Wenn Sie über einen Amazon Virtual Private Cloud (Amazon VPC) -Endpunkt auf Amazon SQS zugreifen, muss die Amazon SQS VPC-Endpunktrichtlinie den Zugriff zulassen. Sie können eine Richtlinie für Amazon VPC-Endpunkte für Amazon SQS erstellen, in der Sie Folgendes angeben können:

1. Prinzipal, der die Aktionen ausführen kann.
2. Aktionen, die ausgeführt werden können
3. Die Ressourcen, für die Aktionen ausgeführt werden können.

Im folgenden Beispiel legt die VPC-Endpunktrichtlinie fest, dass der IAM-Benutzer Nachrichten an die Amazon *MyUser*SQS SQS-Warteschlange senden darf. *MyQueue* Anderen Aktionen, IAM-Benutzern und Amazon SQS-Ressourcen wird der Zugriff über den VPC-Endpunkt verweigert.

```
{
   "Statement": [{
      "Action": ["sqs:SendMessage"],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Principal": {
        "AWS": "arn:aws:iam:123456789012:user/MyUser"
      }
   }]
}
```

Richtlinie zur Dienstkontrolle der Organisation

Wenn Sie zu einer Organisation AWS-Konto gehören, können AWS Organizations Richtlinien Sie daran hindern, auf Ihre Amazon SQS SQS-Warteschlangen zuzugreifen. Standardmäßig blockieren AWS Organizations Richtlinien keine Anfragen an Amazon SQS. Stellen Sie jedoch sicher, dass Ihre AWS Organizations Richtlinien nicht so konfiguriert wurden, dass sie den Zugriff auf Amazon SQS SQS-Warteschlangen blockieren. Anweisungen, wie Sie Ihre AWS Organizations Richtlinien überprüfen können, finden Sie im AWS Organizations Benutzerhandbuch unter Alle Richtlinien auflisten.

Behebung von Amazon SQS SQS-API-Fehlern

Die folgenden Themen behandeln die häufigsten Fehler, die bei Amazon SQS SQS-API-Aufrufen auftreten, und deren Behebung.

Themen

- QueueDoesNotExist Fehler
- InvalidAttributeValue Fehler
- ReceiptHandle Fehler

QueueDoesNotExist Fehler

Dieser Fehler wird zurückgegeben, wenn der Amazon SQS-Service die angegebene Warteschlange für die Amazon SQS SQS-Aktion nicht finden kann.

Mögliche Ursachen und Abhilfemaßnahmen:

- Falsche Region: Überprüfen Sie die Amazon SQS SQS-Client-Konfiguration, um sicherzustellen, dass Sie die richtige Region auf dem Client konfiguriert haben. Wenn Sie keine Region auf dem Client konfigurieren, AWS CLI wählt das SDK oder die Region aus der Konfigurationsdatei oder der Umgebungsvariablen aus. Wenn das SDK in der Konfigurationsdatei keine Region findet, setzt das SDK die Region standardmäßig auf us-east-1.
- Die Warteschlange wurde möglicherweise kürzlich gelöscht: Wenn die Warteschlange vor dem API-Aufruf gelöscht wurde, gibt der API-Aufruf diesen Fehler zurück. CloudTrailSuchen Sie nach DeleteQueueVorgängen, die vor dem Zeitpunkt des Fehlers aufgetreten sind.
- Probleme mit Berechtigungen: Wenn der anfragende Benutzer oder die Rolle AWS Identity and Access Management (IAM) nicht über die erforderlichen Berechtigungen verfügt, wird möglicherweise die folgende Fehlermeldung angezeigt:

  ```
  The specified queue does not exist or you do not have access to it.
  ```

  Überprüfen Sie die Berechtigungen und führen Sie den API-Aufruf mit den richtigen Berechtigungen durch.

Weitere Informationen zur Behebung des `QueueDoesNotExist` Fehlers finden Sie unter Wie behebe ich den QueueDoesNotExist Fehler, wenn ich API-Aufrufe an meine Amazon SQS-Warteschlange tätige? im AWS Knowledge Center-Leitfaden.

InvalidAttributeValue Fehler

Dieser Fehler wird zurückgegeben, wenn die Amazon SQS SQS-Warteschlangenressourcenrichtlinie oder Eigenschaften mit einer falschen Richtlinie oder einem falschen Prinzipal aktualisiert werden.

Mögliche Ursachen und Abhilfemaßnahmen:

- Ungültige Ressourcenrichtlinie: Stellen Sie sicher, dass die Ressourcenrichtlinie alle erforderlichen Felder enthält. Weitere Informationen finden Sie unter Referenz zu IAM-JSON-Richtlinienelementen und Validierung von IAM-Richtlinien. Sie können den IAM-

Richtliniengenerator auch verwenden, um eine Amazon SQS SQS-Ressourcenrichtlinie zu erstellen und zu testen. Stellen Sie sicher, dass die Richtlinie im JSON-Format vorliegt.

- Ungültiger Prinzipal: Stellen Sie sicher, dass das `Principal` Element in der Ressourcenrichtlinie vorhanden ist und dass der Wert gültig ist. Wenn Ihr Amazon SQS `Principal` SQS-Ressourcenrichtlinienelement eine IAM-Entität enthält, stellen Sie sicher, dass die Entität existiert, bevor Sie die Richtlinie verwenden. Amazon SQS validiert die Ressourcenrichtlinie und sucht nach der IAM-Entität. Wenn die IAM-Entität nicht existiert, erhalten Sie eine Fehlermeldung. Verwenden Sie die APIs `GetRole`und `GetUser`, um IAM-Entitäten zu bestätigen.

Weitere Informationen zur Behebung eines `InvalidAttributeValue` Fehlers finden Sie unter Wie behebe ich den QueueDoesNotExist Fehler, wenn ich API-Aufrufe an meine Amazon SQS-Warteschlange tätige? im AWS Knowledge Center-Leitfaden.

ReceiptHandle Fehler

Bei einem `DeleteMessage`API-Aufruf `InvalidParameterValue` kann der Fehler `ReceiptHandleIsInvalid` oder zurückgegeben werden, wenn die Empfangsnummer falsch oder abgelaufen ist.

- ReceiptHandleIsInvalid Fehler: Wenn das Beleg-Handle falsch ist, erhalten Sie eine Fehlermeldung, die dem folgenden Beispiel ähnelt:

```
An error occurred (ReceiptHandleIsInvalid) when calling the DeleteMessage operation:
 The input receipt handle <YOUR RECEIPT HANDLE> is not a valid receipt handle.
```

- InvalidParameterValue Fehler: Wenn das Beleg-Handle abgelaufen ist, erhalten Sie eine Fehlermeldung, die dem folgenden Beispiel ähnelt:

```
An error occurred (InvalidParameterValue) when calling the DeleteMessage operation:
 Value <YOUR RECEIPT HANDLE> for parameter ReceiptHandle is invalid. Reason: The
 receipt handle has expired.
```

Mögliche Ursachen und Abhilfemaßnahmen:

Das Empfangs-Handle wird für jede empfangene Nachricht erstellt und ist nur während des Sichtbarkeits-Timeouts gültig. Wenn das Sichtbarkeits-Timeout abläuft, wird die Nachricht in der Warteschlange für Verbraucher sichtbar. Wenn Sie die Nachricht erneut vom Verbraucher erhalten, erhalten Sie eine neue Empfangsnummer. Um Fehler bei der Verarbeitung falscher oder

abgelaufener Belege zu vermeiden, verwenden Sie die richtige Empfangsnummer, um die Nachricht innerhalb des Zeitlimits für die Sichtbarkeit der Amazon SQS SQS-Warteschlange zu löschen.

Weitere Informationen zur Behebung eines `ReceiptHandle` Fehlers finden Sie unter Wie behebe ich die Fehler "ReceiptHandleIsInvalid" und "InvalidParameterValue“, wenn ich den Amazon SQS DeleteMessage SQS-API-Aufruf verwende? im AWS Knowledge Center-Leitfaden.

Behebung von Problemen mit der Amazon SQS SQS-Warteschlange für unzustellbare Briefe und DLQ Redrive

In den folgenden Themen werden die häufigsten Ursachen für Amazon SQS DLQ- und DLQ Redrive-Probleme sowie deren Behebung behandelt. Weitere Informationen finden Sie unter Wie behebe ich Probleme mit Amazon SQS DLQ Redrive? im AWS Knowledge Center-Leitfaden.

Themen

- DLQ-Probleme
- Probleme mit DLQ-Redrive

DLQ-Probleme

Erfahren Sie mehr über häufig auftretende DLQ-Probleme und wie Sie sie lösen können.

Themen

- Durch das Anzeigen von Nachrichten mithilfe der Konsole werden Nachrichten in eine Warteschlange für unzustellbare Nachrichten verschoben
- Die Werte für NumberOfMessagesSent und NumberOfMessagesReceived für eine Warteschlange für unzustellbare Nachrichten stimmen nicht überein
- Ein Redrive für Warteschlangen mit unerlaubten Briefen erstellen und konfigurieren
- Behandlung von Fehlern bei Nachrichten in Standard- und FIFO-Warteschlangen

Durch das Anzeigen von Nachrichten mithilfe der Konsole werden Nachrichten in eine Warteschlange für unzustellbare Nachrichten verschoben

Amazon SQS zählt das Anzeigen einer Nachricht in der Konsole im Rahmen der zugehörigen Redrive-Richtlinie der Warteschlange. Wenn Sie also eine Nachricht in der Konsole so oft anzeigen,

wie es in der Redrive-Richtlinie der entsprechenden Warteschlange angegeben ist, wird die Nachricht in die Warteschlange für unzustellbare Nachrichten der entsprechenden Warteschlange verschoben.

Führen Sie einen der folgenden Schritte aus, um dieses Verhalten anzupassen:

- Erhöhen Sie die Einstellung Maximum Receives für die Redrive-Richtlinie der entsprechenden Warteschlange.
- Rufen Sie Nachrichten der betroffenen Warteschlange nicht in der Konsole auf.

Die Werte für **`NumberOfMessagesSent`** und **`NumberOfMessagesReceived`** für eine Warteschlange für unzustellbare Nachrichten stimmen nicht überein

Wenn Sie eine Nachricht manuell an eine Warteschlange für unzustellbare Nachrichten senden, wird sie in der Metrik `NumberOfMessagesSent` erfasst. Wenn eine Nachricht jedoch aufgrund eines fehlgeschlagenen Verarbeitungsversuchs an eine Warteschlange für unzustellbare Nachrichten gesendet wird, wird sie von dieser Metrik nicht erfasst. Daher ist es möglich, dass die Werte von `NumberOfMessagesSent` und `NumberOfMessagesReceived`unterschiedlich sind.

Ein Redrive für Warteschlangen mit unerlaubten Briefen erstellen und konfigurieren

Um die Warteschlange für unzustellbare Nachrichten erneut zu aktivieren, müssen Sie Amazon SQS die entsprechenden Berechtigungen für den Empfang von Nachrichten aus der Warteschlange für unzustellbare Briefe und das Senden von Nachrichten an die Zielwarteschlange einrichten. Wenn Sie nicht über die richtigen Berechtigungen verfügen, kann die Aufgabe zur Neuübertragung von Warteschlangen mit unerlaubten Briefen fehlschlagen. Sie können den Status Ihrer Aufgabe zum erneuten Versand von Nachrichten anzeigen, um die Probleme zu beheben, und es erneut versuchen.

Behandlung von Fehlern bei Nachrichten in Standard- und FIFO-Warteschlangen

Standardwarteschlangen verarbeiten weiterhin Nachrichten bis zum Ablauf der Aufbewahrungsfrist. Durch diese kontinuierliche Verarbeitung wird die Wahrscheinlichkeit minimiert, dass die Warteschlange durch nicht verbrauchte Nachrichten blockiert wird. Eine große Anzahl von Nachrichten, die der Verbraucher wiederholt nicht löscht, kann die Kosten in die Höhe treiben und die Hardware zusätzlich belasten. Um die Kosten niedrig zu halten, verschieben Sie fehlgeschlagene Nachrichten in die Warteschlange für unzustellbare Nachrichten.

Standardwarteschlangen ermöglichen auch eine hohe Anzahl von Nachrichten während des Fluges. Wenn der Großteil Ihrer Nachrichten nicht verarbeitet werden kann und nicht in eine

Warteschlange mit unzustellbaren Briefen geschickt wird, kann sich Ihre Geschwindigkeit bei der Nachrichtenverarbeitung verlangsamen. Um die Effizienz Ihrer Warteschlange aufrechtzuerhalten, stellen Sie sicher, dass Ihre Anwendung die Nachrichtenverarbeitung korrekt handhabt.

FIFO-Warteschlangen sorgen dafür, dass Nachrichten genau einmal verarbeitet werden, indem diese nacheinander einer Nachrichtengruppe entnommen werden. Daher kann der Verbraucher zwar weiterhin geordnete Nachrichten aus einer anderen Nachrichtengruppe abrufen, die erste Nachrichtengruppe ist jedoch so lange nicht verfügbar, bis die Nachricht, die die Warteschlange blockiert, erfolgreich verarbeitet oder in eine Warteschlange mit unzustellbaren Nachrichten verschoben wurde.

Darüber hinaus ermöglichen FIFO-Warteschlangen eine geringere Anzahl von Nachrichten während der Übertragung. Um zu verhindern, dass Ihre FIFO-Warteschlange durch eine Nachricht blockiert wird, stellen Sie sicher, dass Ihre Anwendung die Nachrichtenverarbeitung korrekt handhabt.

Weitere Informationen finden Sie unter Amazon SQS SQS-Nachrichtenkontingente und Arbeiten mit Amazon-SQS-Nachrichten.

Probleme mit DLQ-Redrive

Erfahren Sie mehr über häufig auftretende Probleme mit DLQ-Redrive und wie Sie diese lösen können.

Themen

- AccessDenied Problem mit der Genehmigung
- NonExistentQueue Fehler
- CouldNotDetermineMessageQuellfehler

AccessDenied Problem mit der Genehmigung

Der `AccessDenied` Fehler tritt auf, wenn das DLQ-Redrive fehlschlägt, weil die AWS Identity and Access Management (IAM-) Entität nicht über die erforderlichen Berechtigungen verfügt.

Beispiel für eine Fehlermeldung:

```
Failed to create redrive task. Error code: AccessDenied - Queue Permissions to Redrive.
```

Die folgenden API-Berechtigungen sind erforderlich, um DLQ-Redrive-Anfragen zu stellen:

Um eine Nachrichten-Redrive zu starten:

- Berechtigungen für die Warteschlange „Unzustellbarer Briefe“:
  - `sqs:StartMessageMoveTask`
  - `sqs:ReceiveMessage`
  - `sqs:DeleteMessage`
  - `sqs:GetQueueAttributes`
  - `kms:Decrypt`— Wenn entweder die Warteschlange für unzustellbare Nachrichten oder die ursprüngliche Quellwarteschlange verschlüsselt sind.
- Berechtigungen für die Zielwarteschlange:
  - `sqs:SendMessage`
  - `kms:GenerateDataKey`— Wenn die Zielwarteschlange verschlüsselt ist.
  - `kms:Decrypt` — Wenn die Zielwarteschlange verschlüsselt ist.

Um eine Nachricht abzubrechen, die gerade bearbeitet wird, wiederholen Sie den Vorgang:

- Berechtigungen für die Warteschlange „Unzustellbare Briefe“:
  - `sqs:CancelMessageMoveTask`
  - `sqs:ReceiveMessage`
  - `sqs:DeleteMessage`
  - `sqs:GetQueueAttributes`
  - `kms:Decrypt`— Wenn entweder die Warteschlange für unzustellbare Nachrichten oder die ursprüngliche Quellwarteschlange verschlüsselt sind.

So zeigen Sie den Verschiebungsstatus einer Nachricht an:

- Berechtigungen für die Warteschlange „Unzustellbare Briefe“:
  - `sqs:ListMessageMoveTasks`
  - `sqs:GetQueueAttributes`

NonExistentQueue Fehler

Der `NonExistentQueue` Fehler tritt auf, wenn die Amazon SQS SQS-Quellwarteschlange nicht existiert oder gelöscht wurde. Suchen Sie nach einer vorhandenen Amazon SQS-Warteschlange und wechseln Sie erneut zu einer vorhandenen Amazon SQS-Warteschlange.

Beispiel für eine Fehlermeldung:

```
Failed: AWS.SimpleQueueService.NonExistentQueue
```

CouldNotDetermineMessageQuellfehler

Der `CouldNotDetermineMessageSource` Fehler tritt auf, wenn Sie versuchen, einen DLQ-Redrive mit den folgenden Szenarien zu starten:

- Eine Amazon SQS SQS-Nachricht, die mit `SendMessage`API direkt an den DLQ gesendet wird.
- Eine Nachricht vom Thema oder der AWS Lambda Funktion Amazon Simple Notification Service (Amazon SNS) mit konfiguriertem DLQ.

Um diesen Fehler zu beheben, wählen Sie Redrive to a custom destination, wenn Sie den Redrive starten. Geben Sie dann den ARN der Amazon SQS SQS-Warteschlange ein, um alle Nachrichten von der DLQ in die Zielwarteschlange zu verschieben.

Beispiel für eine Fehlermeldung:

```
Failed: CouldNotDetermineMessageSource
```

Behebung von Problemen mit der FIFO-Drosselung in Amazon SQS

Standardmäßig unterstützen FIFO-Warteschlangen 300 Transaktionen pro Sekunde und API-Aktion für `SendMessage`, und `ReceiveMessage`. `DeleteMessage` Bei Anfragen über 300 TPS wird der `ThrottlingException` Fehler auch dann angezeigt, wenn Nachrichten in der Warteschlange verfügbar sind. Um dies zu beheben, können Sie die folgenden Methoden verwenden:

- Aktivieren Sie einen hohen Durchsatz für FIFO-Warteschlangen in Amazon SQS.

- Verwenden Sie die Amazon SQS SQS-API-Batch-Aktionen `SendMessageBatch` und `ChangeMessageVisibilityBatch` um das TPS-Limit von bis zu 3.000 Nachrichten pro Sekunde pro API-Aktion zu erhöhen und die Kosten zu senken. `DeleteMessageBatch` Stellen Sie für die `ReceiveMessage` API den `MaxNumberofMessages` Parameter so ein, dass er bis zu zehn Nachrichten pro Transaktion empfängt. Weitere Informationen finden Sie unter Amazon-SQS-Stapelaktionen.
- Folgen Sie bei FIFO-Warteschlangen mit hohem Durchsatz den Empfehlungen zur Optimierung der Partitionsnutzung. Senden Sie Nachrichten mit denselben Nachrichtengruppen-IDs stapelweise. Löschen Sie Nachrichten oder ändern Sie die Timeout-Werte für die Nachrichtensichtbarkeit in Stapeln mit Empfangsdaten aus denselben `ReceiveMessage` API-Anfragen.
- Erhöhen Sie die Anzahl der `MessageGroupId`Einzelwerte. Dies ermöglicht eine gleichmäßige Verteilung auf die FIFO-Warteschlangenpartitionen. Weitere Informationen finden Sie unter Verwenden der Amazon-SQS-Nachrichtengruppen-ID.

Weitere Informationen finden Sie unter Warum gibt meine Amazon SQS FIFO-Warteschlange nicht alle Nachrichten oder Nachrichten in anderen Nachrichtengruppen zurück? im AWS Knowledge Center-Leitfaden.

Problembehandlung bei Nachrichten, die bei einem Amazon SQS ReceiveMessage SQS-API-Aufruf nicht zurückgegeben wurden

In den folgenden Themen werden die häufigsten Ursachen behandelt, warum eine Amazon SQS SQS-Nachricht möglicherweise nicht an Verbraucher zurückgesendet wird, und wie diese Probleme behoben werden können. Weitere Informationen finden Sie unter Warum kann ich keine Nachrichten aus meiner Amazon SQS SQS-Warteschlange empfangen? im AWS Knowledge Center-Leitfaden.

Themen

- Leere Warteschlange
- Während des Fluges wurde das Limit erreicht
- Verzögerung der Nachricht
- Die Nachricht ist im Flug
- Abfragemethode

Leere Warteschlange

Um festzustellen, ob eine Warteschlange leer ist, rufen Sie die `ReceiveMessage`API mithilfe von langen Abfragen auf. Sie können auch die `ApproximateNumberOfMessagesDelayed` CloudWatch Metriken `ApproximateNumberOfMessagesVisible`, `ApproximateNumberOfMessagesNotVisible`, und verwenden. Wenn alle Metrikwerte für mehrere Minuten auf 0 gesetzt sind, wird die Warteschlange als leer betrachtet.

Während des Fluges wurde das Limit erreicht

Wenn Sie Long Polling verwenden und das In-Flight-Limit der Warteschlange (standardmäßig 20000 für FIFO, 120000 für Standard) überschritten wird, gibt Amazon SQS keine Fehlermeldungen zurück, die die Kontingentgrenzen überschreiten.

Verzögerung der Nachricht

Wenn die Amazon SQS SQS-Warteschlange als Verzögerungswarteschlange konfiguriert ist oder die Nachrichten mit Nachrichtentimern gesendet wurden, sind die Nachrichten erst sichtbar, wenn die Verzögerungszeit abgelaufen ist. Um zu überprüfen, ob eine Warteschlange als Verzögerungswarteschlange konfiguriert ist, verwenden Sie das `GetQueueAttributes``DelaySeconds`API-Attribut oder verwenden Sie die Warteschlangenkonsole unter Delivery Delay. Prüfen Sie `ApproximateNumberOfMessagesDelayed` CloudWatch anhand der Metrik, ob Nachrichten verzögert sind.

Die Nachricht ist im Flug

Wenn ein anderer Nutzer die Nachricht abgefragt hat, ist die Nachricht während des Sichtbarkeitszeitraums entweder im Umlauf oder unsichtbar. Bei den zusätzlichen Umfragen wird möglicherweise ein leerer Empfang zurückgegeben. Überprüfen Sie die CloudWatch Metrik ApproximateNumberOfMessagesSichtbar, um die Anzahl der Nachrichten zu ermitteln, die für den Empfang verfügbar sind. Wenn bei FIFO-Warteschlangen eine Nachricht mit der Nachrichtengruppen-ID gesendet wird, werden keine Nachrichten mehr zurückgegeben, es sei denn, Sie löschen die Nachricht oder sie wird sichtbar. Das liegt daran, dass die Reihenfolge der Nachrichten in einer FIFO-Warteschlange auf Nachrichtengruppenebene beibehalten wird.

Abfragemethode

Wenn Sie Short-Polling verwenden (WaitTimeSeconds ist 0), nimmt Amazon SQS eine Stichprobe von einer Teilmenge seiner Server vor und gibt nur Nachrichten von diesen Servern zurück. Daher erhalten Sie die Nachrichten möglicherweise nicht, auch wenn sie für den Empfang verfügbar sind. Bei nachfolgenden Umfrageanfragen werden die Nachrichten zurückgegeben.

Wenn Sie lange Abfragen verwenden, fragt Amazon SQS alle Server ab und sendet eine Antwort, nachdem mindestens eine verfügbare Nachricht erfasst wurde, und zwar bis zur angegebenen Höchstanzahl. Wenn der Wert für ReceiveMessage WaitTimeSekunden zu niedrig ist, erhalten Sie möglicherweise nicht alle verfügbaren Nachrichten.

Behebung Amazon SQS SQS-Netzwerkfehlern

Die folgenden Themen behandeln die häufigsten Ursachen für Netzwerkprobleme in Amazon SQS und deren Behebung.

Themen

- ETIMEOUT error
- UnknownHostException error

ETIMEOUT error

Der ETIMEOUT Fehler tritt auf, wenn der Client keine TCP-Verbindung zu einem Amazon SQS SQS-Endpunkt herstellen kann.

Fehlerbehebung

- Überprüfen Sie die Netzwerkverbindung

  Testen Sie Ihre Netzwerkverbindung zu Amazon SQS, indem Sie Befehle wie `telnet` ausführen.

  Example: telnet sqs.us-east-1.amazonaws.com 443
- Überprüfen Sie die Netzwerkeinstellungen
  - Stellen Sie sicher, dass Ihre lokalen Firewallregeln, Routen und Zugriffskontrolllisten (ACLs) Datenverkehr auf dem von Ihnen verwendeten Port zulassen.
  - Die Regeln der Sicherheitsgruppe für ausgehenden Datenverkehr (ausgehenden Verkehr) müssen den Datenverkehr zum Port 80 oder 443 zulassen.

  - Die Netzwerk-ACL-Regeln für ausgehenden Verkehr (ausgehenden Verkehr) müssen den Datenverkehr zum TCP-Port 80 oder 443 zulassen.
  - Die Netzwerk-ACL-Regeln für eingehenden Datenverkehr (Eingang) müssen den Datenverkehr auf den TCP-Ports 1024-65535 zulassen.
  - Amazon Elastic Compute Cloud (Amazon EC2) -Instances, die eine Verbindung zum öffentlichen Internet herstellen, müssen über eine Internetverbindung verfügen.
- Endpunkte der Amazon Virtual Private Cloud (Amazon VPC)

  Wenn Sie über einen Amazon VPC-Endpunkt auf Amazon SQS zugreifen, muss die Sicherheitsgruppe der Endgeräte eingehenden Datenverkehr zur Sicherheitsgruppe des Clients auf Port 443 zulassen. Die Netzwerk-ACL, die dem Subnetz des VPC-Endpunkts zugeordnet ist, muss diese Konfiguration haben:

  - Die Netzwerk-ACL-Regeln für ausgehenden Datenverkehr (Ausgang) müssen den Datenverkehr auf den TCP-Ports 1024-65535 (kurzlebige Ports) zulassen.
  - Die Netzwerk-ACL-Regeln für eingehenden Datenverkehr (Eingang) müssen Datenverkehr auf Port 443 zulassen.

Außerdem muss die Amazon SQS VPC-Endpoint AWS Identity and Access Management (IAM) -Richtlinie den Zugriff zulassen. Die folgende Beispiel-VPC-Endpunktrichtlinie legt fest, dass der IAM-Benutzer *MyUser*Nachrichten an die Amazon SQS SQS-Warteschlange senden darf. *MyQueue* Anderen Aktionen, IAM-Benutzern und Amazon SQS-Ressourcen wird der Zugriff über den VPC-Endpunkt verweigert.

```
{
    "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

Der UnknownHostException Fehler tritt auf, wenn die Host-IP-Adresse nicht ermittelt werden konnte.

Fehlerbehebung

Verwenden Sie das nslookup Hilfsprogramm, um die dem Hostnamen zugeordnete IP-Adresse zurückzugeben:

- Windows and Linux OS

```
nslookup sqs.<region>.amazonaws.com
```

- AWS CLI oder SDK für ältere Python-Endpunkte:

```
nslookup <region>.queue.amazonaws.com
```

Wenn Sie eine erfolglose Ausgabe erhalten haben, folgen Sie den Anweisungen unter Wie funktioniert DNS und wie behebe ich teilweise oder zeitweise auftretende DNS-Fehler? im AWS Knowledge Center-Leitfaden.

Wenn Sie eine gültige Ausgabe erhalten haben, handelt es sich wahrscheinlich um ein Problem auf Anwendungsebene. Probieren Sie die folgenden Methoden aus, um Probleme auf Anwendungsebene zu beheben:

- Starten Sie Ihre Anwendung neu.
- Vergewissern Sie sich, dass Ihre Java-Anwendung keinen fehlerhaften DNS-Cache hat. Wenn möglich, konfigurieren Sie Ihre Anwendung so, dass sie der DNS-TTL entspricht. Weitere Informationen finden Sie unter JVM-TTL für DNS-Namenssuchen einrichten.

Weitere Informationen zur Behebung von Netzwerkfehlern finden Sie unter Wie behebe ich die Amazon SQS SQS-Verbindungsfehler „ETIMEOUT“ und „UnknownHostException“? im AWS Knowledge Center-Leitfaden.

Problembehandlung bei Warteschlangen von Amazon Simple Queue Service mit AWS X-Ray

AWS X-Ray sammelt Daten über Anfragen, die Ihre Anwendung bearbeitet, und ermöglicht es Ihnen, Daten anzuzeigen und zu filtern, um potenzielle Probleme und Optimierungsmöglichkeiten zu identifizieren. Für jede verfolgte Anfrage an Ihre Anwendung können Sie detaillierte Informationen über die Anfrage, die Antwort und die Aufrufe einsehen, die Ihre Anwendung an nachgelagerte AWS Ressourcen, Microservices, Datenbanken und HTTP-Web-APIs sendet.

Um AWS X-Ray Trace-Header über Amazon SQS zu senden, können Sie einen der folgenden Schritte ausführen:

- Verwenden Sie `X-Amzn-Trace-Id`-Nachverfolgungs-Header.
- Verwenden Sie das `AWSTraceHeader`-Nachrichtensystemattribut.

Um Daten über Fehler und Latenz zu erfassen, müssen Sie den `AmazonSQS`-Client mit dem AWS -X-Ray-SDK instrumentieren.

Sie können die AWS X-Ray Konsole verwenden, um die Verbindungsübersicht zwischen Amazon SQS und anderen Services anzuzeigen, die Ihre Anwendung verwendet. Sie können mithilfe der Konsole auch Metriken, wie durchschnittliche Latenz- und Ausfallraten, anzeigen. Weitere Informationen finden Sie unter Amazon SQS und AWS X-Ray im AWS X-Ray -Entwicklerhandbuch.

Sicherheit in Amazon SQS

Dieser Abschnitt enthält Informationen zur Sicherheit, Authentifizierung und Zugriffskontrolle in Amazon SQS sowie zur Sprache der Zugriffsrichtlinie von Amazon SQS.

Themen

- Datenschutz in Amazon SQS
- Identity and Access Management in Amazon SQS
- Protokollierung und Überwachung in Amazon SQS
- Compliance-Validierung für Amazon SQS
- Ausfallsicherheit bei Amazon SQS
- Infrastruktursicherheit in Amazon SQS
- Bewährte Methoden für die Sicherheit in Amazon SQS

Datenschutz in Amazon SQS

Das AWS Modell der gilt für den Datenschutz in Amazon Simple Queue Service. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail

- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Amazon SQS oder anderen Geräten arbeiten und die Konsole AWS CLI, API oder AWS SDKs AWS-Services verwenden. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Die folgenden Abschnitte enthalten Informationen zum Datenschutz in Amazon SQS.

Themen

- Datenverschlüsselung in Amazon SQS
- Datenschutz im Netzwerkverkehr in Amazon SQS

Datenverschlüsselung in Amazon SQS

Datenschutz bezieht sich auf Daten in Übertragung (wenn sie zu Amazon SQS oder von diesem geschickt werden), ebenso wie auf ruhende Daten (die in Amazon-SQS-Rechenzentren auf Datenträgern gespeichert sind). Sie können Daten mithilfe von Secure Sockets Layer (SSL) oder einer clientseitigen Verschlüsselung während der Übertragung schützen. Standardmäßig speichert Amazon SQS Nachrichten und Dateien mit Festplattenverschlüsselung. Sie können Daten im Ruhezustand schützen, indem Sie Amazon SQS auffordern, Ihre Nachrichten zu verschlüsseln, bevor Sie sie im verschlüsselten Dateisystem in seinen Rechenzentren speichern. Amazon SQS empfiehlt die Verwendung von SSE für eine optimierte Datenverschlüsselung.

Themen

- Verschlüsselung im Ruhezustand in Amazon SQS
- Amazon SQS Schlüsselverwaltung

Verschlüsselung im Ruhezustand in Amazon SQS

Mit der serverseitigen Verschlüsselung (SSE) können Sie vertrauliche Daten in verschlüsselten Warteschlangen übermitteln. SSE schützt den Inhalt von Nachrichten in Warteschlangen mithilfe von SQS-verwalteten Verschlüsselungsschlüsseln (SSE-SQS) oder Schlüsseln, die im (SSE-KMS) verwaltet werden. AWS Key Management Service Informationen zur Verwaltung von SSE mithilfe von finden Sie im Folgenden: AWS Management Console

- Konfigurieren von SSE-SQS für eine Warteschlange (Konsole)
- Konfigurieren von SSE-KMS für eine Warteschlange (Konsole)

Informationen zur Verwaltung von SSE mithilfe der `GetQueueAttributes` Aktionen AWS SDK for Java (und `CreateQueueSetQueueAttributes`, und) finden Sie in den folgenden Beispielen:

- Serverseitige Verschlüsselung mit Amazon SQS SQS-Warteschlangen verwenden
- Konfiguration von KMS-Berechtigungen für AWS-Services

SSE verschlüsselt Nachrichten, sobald sie bei Amazon SQS eingehen. Die Nachrichten werden verschlüsselt gespeichert. Amazon SQS entschlüsselt Nachrichten nur, wenn sie an einen autorisierten Konsumenten gesendet werden.

Important

Alle Anfragen zu Warteschlangen mit aktiviertem SSE müssen HTTPS und Signature Version 4 verwenden.
Eine verschlüsselte Warteschlange, die den Standardschlüssel (AWS verwalteter KMS-Schlüssel für Amazon SQS) verwendet, kann keine Lambda-Funktion in einer anderen aufrufen. AWS-Konto
Einige Funktionen von AWS Diensten, die mithilfe der AWS Security Token Service `AssumeRole` Aktion Benachrichtigungen an Amazon SQS senden können, sind mit SSE kompatibel, funktionieren jedoch nur mit Standardwarteschlangen:

- Auto Scaling Lifecycle Hooks

- AWS Lambda Warteschlangen für unzustellbare Nachrichten

Weitere Informationen zur Kompatibilität anderer Services mit verschlüsselten Warteschlangen finden Sie unter Konfigurieren Sie KMS-Berechtigungen für Dienste AWS und in Ihrer Service-Dokumentation.

AWS KMS kombiniert sichere, hochverfügbare Hardware und Software zu einem für die Cloud skalierten Schlüsselverwaltungssystem. Wenn Sie Amazon SQS mit verwenden AWS KMS, werden die Datenschlüssel, die Ihre Nachrichtendaten verschlüsseln, ebenfalls verschlüsselt und zusammen mit den Daten gespeichert, die sie schützen.

Vorteile von AWS KMS:

- Sie können AWS KMS keys selbst erstellen und verwalten.
- Sie können auch den AWS verwalteten KMS-Schlüssel für Amazon SQS verwenden, der für jedes Konto und jede Region einzigartig ist.
- Die AWS KMS Sicherheitsstandards können Ihnen dabei helfen, die Compliance-Anforderungen im Zusammenhang mit der Verschlüsselung zu erfüllen.

Weitere Informationen finden Sie unter Was ist AWS Key Management Service? im AWS Key Management Service -Entwicklerhandbuch.

Themen

- Verschlüsselungsumfang
- Wichtige Begriffe

Verschlüsselungsumfang

SSE verschlüsselt den Nachrichtentext in einer Amazon-SQS-Warteschlange.

Folgendes wird von SSE nicht verschlüsselt:

- Metadaten der Warteschlange (Name und Attribute)
- Metadaten der Nachrichten (ID, Zeitstempel und Attribute)
- Metriken pro Warteschlange

Durch die Verschlüsselung sind die Nachrichteninhalte für nicht autorisierte oder anonyme Benutzer nicht zugänglich. Wenn SSE aktiviert ist, werden anonyme `SendMessage`- und `ReceiveMessage`-Anfragen an die verschlüsselte Warteschlange abgewiesen. Die bewährten Sicherheitsmethoden von Amazon SQS raten davon ab, anonyme Anfragen zu verwenden. Wenn Sie anonyme Anfragen an eine Amazon-SQS-Warteschlange senden möchten, stellen Sie sicher, dass Sie SSE deaktivieren. Dies wirkt sich nicht auf die normale Funktion von Amazon SQS aus:

- Eine Nachricht ist nur dann verschlüsselt, wenn sie gesendet wurde, nachdem die Verschlüsselung einer Warteschlange aktiviert wurde. Amazon SQS verschlüsselt keine Nachrichten, die sich bereits in der Queue befinden.
- Verschlüsselte Nachrichten bleiben auch dann verschlüsselt, wenn die Verschlüsselung der Warteschlange deaktiviert wird.

Das Verschieben einer Nachricht in eine Warteschlange für unzustellbare Nachrichten wirkt sich nicht auf ihre Verschlüsselung aus:

- Wenn Amazon SQS eine Nachricht aus einer verschlüsselten Quellwarteschlange in eine unverschlüsselte Warteschlange für unzustellbare Nachrichten verschiebt, bleibt die Nachricht verschlüsselt.
- Wenn Amazon SQS eine Nachricht aus einer unverschlüsselten Quellwarteschlange in eine verschlüsselte Warteschlange für unzustellbare Nachrichten verschiebt, bleibt die Nachricht unverschlüsselt.

Wichtige Begriffe

Die folgenden wichtigen Begriffe vermitteln Ihnen ein besseres Verständnis für die Funktionalität von SSE. Detaillierte Beschreibungen finden Sie in der Amazon-Simple-Queue-Service-API-Referenz.

Datenschlüssel

Der Schlüssel (DEK), der dafür zuständig ist, den Inhalt von Amazon-SQS-Nachrichten zu verschlüsseln.

Weitere Informationen finden Sie unter Datenschlüssel im AWS Key Management Service - Entwicklerhandbuch im AWS Encryption SDK -Entwicklerhandbuch.

Data key reuse period (Wiederverwendungszeitraum für den Datenschlüssel)

Die Zeitspanne in Sekunden, für die Amazon SQS einen Datenschlüssel wiederverwenden kann, um Nachrichten zu verschlüsseln oder zu entschlüsseln, bevor er erneut anruft. AWS KMS Eine Ganzzahl stellt Sekunden dar, und zwar zwischen 60 Sekunden (1 Minute) und 86 400 Sekunden (24 Stunden). Der Standardwert ist 300 (5 Minuten). Weitere Informationen finden Sie unter Grundlegendes zum Wiederverwendungszeitraum für den Datenschlüssel.

> **Note**
>
> In dem unwahrscheinlichen Fall, dass keine Verbindung hergestellt werden kann AWS KMS, verwendet Amazon SQS weiterhin den zwischengespeicherten Datenschlüssel, bis eine Verbindung wiederhergestellt ist.

KMS-Schlüssel-ID

Der Alias, Alias-ARN, die Schlüssel-ID oder der Schlüssel-ARN eines AWS verwalteten KMS-Schlüssels oder eines benutzerdefinierten KMS-Schlüssels — in Ihrem Konto oder in einem anderen Konto. Während der Alias des AWS verwalteten KMS-Schlüssels für Amazon SQS immer ist`alias/aws/sqs`, kann der Alias eines benutzerdefinierten KMS-Schlüssels beispielsweise sein`alias/MyAlias`. Sie können diese KMS-Schlüssel zum Schutz der Nachrichten in Amazon-SQS-Warteschlangen verwenden.

> **Note**
>
> Beachten Sie Folgendes:
>
> - Wenn Sie keinen benutzerdefinierten KMS-Schlüssel angeben, verwendet Amazon SQS den AWS verwalteten KMS-Schlüssel für Amazon SQS.
> - Wenn Sie AWS Management Console zum ersten Mal den AWS verwalteten KMS-Schlüssel für Amazon SQS für eine Warteschlange angeben, AWS KMS wird der AWS verwaltete KMS-Schlüssel für Amazon SQS erstellt.
> - Alternativ können Sie, wenn Sie die `SendMessageBatch` Aktion `SendMessage` oder zum ersten Mal in einer Warteschlange mit aktivierter SSE verwenden, der AWS verwaltete KMS-Schlüssel für Amazon SQS AWS KMS erstellt.

Sie können KMS-Schlüssel erstellen, die Richtlinien definieren, die steuern, wie KMS-Schlüssel verwendet werden können, und die Verwendung von KMS-Schlüsseln mithilfe des Abschnitts „Vom Kunden verwaltete Schlüssel“ der AWS KMS Konsole oder der `CreateKey` AWS KMS Aktion überprüfen. Weitere Informationen zum Erstellen von KMS-Schlüsseln finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service -Entwicklerhandbuch. Weitere Beispiele für KMS-Schlüsselkennungen finden Sie KeyIdin der AWS Key Management Service API-Referenz. Weitere Informationen zum Suchen von KMS-Schlüssel-IDs finden Sie unter Suchen der Schlüssel-ID und des ARNs im AWS Key Management Service -Entwicklerhandbuch.

> **Important**
>
> Für die Nutzung AWS KMS fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter Schätzung der Kosten AWS KMS und Preise zu AWS Key Management Service.

Envelope-Verschlüsselung

Die Sicherheit Ihrer verschlüsselten Daten hängt teilweise vom Schutz des Datenschlüssels ab, der sie entschlüsseln kann. Amazon SQS verwendet den KMS-Schlüssel, um den Datenschlüssel zu verschlüsseln. Anschließend wird der verschlüsselte Datenschlüssel zusammen mit der verschlüsselten Nachricht gespeichert. Diese Vorgehensweise der Verwendung eines KMS-Schlüssels zum Verschlüsseln von Datenschlüsseln wird als Umschlagverschlüsselung bezeichnet.

Weitere Informationen zur Envelope-Verschlüsselung finden im AWS Encryption SDK Entwicklerhandbuch.

Amazon SQS Schlüsselverwaltung

Amazon SQS lässt sich in das AWS Key Management Service (KMS) integrieren, um KMS-Schlüssel für serverseitige Verschlüsselung (SSE) zu verwalten. SSE-Informationen und Definitionen zur Schlüsselverwaltung finden Sie unter Verschlüsselung im Ruhezustand in Amazon SQS. Amazon SQS verwendet KMS-Schlüssel, um die Datenschlüssel zu validieren und zu sichern, mit denen die Nachrichten ver- und entschlüsselt werden. Die folgenden Abschnitte enthalten Informationen über das Arbeiten mit KMS-Schlüsseln und Datenschlüsseln im Amazon-SQS-Service.

Themen

- Konfigurieren von AWS KMS -Berechtigungen
- Grundlegendes zum Wiederverwendungszeitraum für den Datenschlüssel
- Schätzung der Kosten AWS KMS
- AWS KMS Fehler

Konfigurieren von AWS KMS -Berechtigungen

Jeder KMS-Schlüssel muss über eine Schlüsselrichtlinie verfügen. Beachten Sie, dass Sie die Schlüsselrichtlinie eines AWS verwalteten KMS-Schlüssels für Amazon SQS nicht ändern können. Die Richtlinie für diesen KMS-Schlüssel beinhaltet Berechtigungen für alle Prinzipale im Konto (die zur Verwendung von Amazon SQS berechtigt sind), verschlüsselte Warteschlangen zu verwenden.

Bei einem kundenverwalteten KMS-Schlüssel müssen Sie die Schlüsselrichtlinie konfigurieren, um Berechtigungen für jeden Warteschlangenproduzenten und -konsumenten hinzuzufügen. Dazu benennen Sie in der KMS-Schlüsselrichtlinie den Produzenten und den Konsumenten als Benutzer. Weitere Informationen zu AWS KMS Berechtigungen finden Sie in der Referenz zu AWS KMS Ressourcen und Vorgängen oder AWS KMS API-Berechtigungen im AWS Key Management Service Entwicklerhandbuch.

Alternativ können Sie die erforderlichen Berechtigungen in einer IAM-Richtlinie angeben, die den Prinzipalen zugewiesen ist, die verschlüsselte Nachrichten produzieren und konsumieren. Weitere Informationen finden Sie unter Verwenden von IAM-Richtlinien mit AWS KMS im AWS Key Management Service -Entwicklerhandbuch.

> **Note**
>
> Sie können zwar globale Berechtigungen für das Senden an und Empfangen von Amazon SQS konfigurieren, AWS KMS erfordert jedoch die ausdrückliche Benennung des vollständigen ARN der KMS-Schlüssel in bestimmten Regionen im `Resource` Abschnitt einer IAM-Richtlinie.

Konfigurieren Sie KMS-Berechtigungen für Dienste AWS

Verschiedene AWS Dienste dienen als Ereignisquellen, die Ereignisse an Amazon SQS SQS-Warteschlangen senden können. Damit diese Ereignisquellen mit verschlüsselten Warteschlangen arbeiten können, müssen Sie einen vom Kunden verwalteten KMS-Schlüssel erstellen und der Schlüsselrichtlinie Berechtigungen hinzufügen, damit der Service die erforderlichen AWS KMS API-

Methoden verwenden kann. Führen Sie zum Konfigurieren der Berechtigungen die folgenden Schritte durch.

> **Warning**
>
> Wenn Sie den KMS-Schlüssel für die Verschlüsselung Ihrer Amazon SQS SQS-Nachrichten ändern, beachten Sie, dass bestehende Nachrichten, die mit dem alten KMS-Schlüssel verschlüsselt wurden, mit diesem Schlüssel verschlüsselt bleiben. Um diese Nachrichten zu entschlüsseln, müssen Sie den alten KMS-Schlüssel beibehalten und sicherstellen, dass seine Schlüsselrichtlinie Amazon SQS die Berechtigungen für `kms:Decrypt` und gewährt. `kms:GenerateDataKey` Stellen Sie nach der Aktualisierung auf einen neuen KMS-Schlüssel zur Verschlüsselung neuer Nachrichten sicher, dass alle vorhandenen Nachrichten, die mit dem alten KMS-Schlüssel verschlüsselt wurden, verarbeitet und aus der Warteschlange entfernt werden, bevor Sie den alten KMS-Schlüssel löschen oder deaktivieren.

1. Erstellen Sie einen vom Kunden verwalteten KMS-Schlüssel. Weitere Informationen finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service -Entwicklerhandbuch.
2. Damit die AWS Dienstereignisquelle die Methoden `kms:GenerateDataKey` und `kms:Decrypt` API verwenden kann, fügen Sie der KMS-Schlüsselrichtlinie die folgende Anweisung hinzu.

```
{
   "Version": "2012-10-17",
      "Statement": [{
         "Effect": "Allow",
         "Principal": {
            "Service": "service.amazonaws.com"
         },
         "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
         ],
         "Resource": "*"
       }]
}
```

   Ersetzen Sie „service“ im obigen Beispiel durch den Servicenamen der Ereignisquelle. Zu den Ereignisquellen gehören die folgenden Services.

Ereignisquelle	Service-Name
CloudWatch Amazon-Veranstaltungen	`events.amazonaws.com`
Amazon-S3-Ereignis-Benachrichtigungen	`s3.amazonaws.com`
Amazon-SNS-Themenabonnements	`sns.amazonaws.com`

3. Konfigurieren Sie eine vorhandene SSE-Warteschlange mit dem ARN Ihres KMS-Schlüssels.
4. Stellen Sie der verschlüsselten Warteschlange den ARN der Ereignisquelle zur Verfügung.

AWS KMS Berechtigungen für Produzenten konfigurieren

Wenn der Zeitraum für die Wiederverwendung des Datenschlüssels abgelaufen ist, löst der nächste Aufruf des Produzenten von `SendMessage` oder `SendMessageBatch` auch Aufrufe von `kms:GenerateDataKey` und `kms:Decrypt` aus. Der Aufruf von `kms:Decrypt` dient dazu, die Integrität des neuen Datenschlüssels vor dessen Verwendung zu überprüfen. Daher muss der Produzent über die Berechtigungen `kms:GenerateDataKey` und `kms:Decrypt` für den KMS-Schlüssel verfügen.

Fügen Sie der IAM-Richtlinie des Produzenten die folgende Anweisung hinzu. Denken Sie daran, die richtigen ARN-Werte für die Schlüsselressource und die Warteschlangenressource zu verwenden.

```
{
   "Version": "2012-10-17",
      "Statement": [{
         "Effect": "Allow",
         "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
         ],
         "Resource":  "arn:aws:kms:us-
east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
         }, {
         "Effect": "Allow",
         "Action": [
            "sqs:SendMessage"
         ],
         "Resource": "arn:aws:sqs:*:123456789012:MyQueue"
      }]
```

```
}
```

AWS KMS Berechtigungen für Verbraucher konfigurieren

Wenn der Zeitraum für die Wiederverwendung des Datenschlüssels abläuft, löst der nächste Aufruf des Konsumenten von `ReceiveMessage` ebenfalls einen Auruf von `kms:Decrypt` aus, um die Integrität des neuen Datenschlüssels vor dessen Verwendung zu überprüfen. Deshalb muss der Konsument über die Berechtigung `kms:Decrypt` für alle KMS-Schlüssel verfügen, die für die Verschlüsselung von Nachrichten in der angegebenen Warteschlange verwendet werden. Wenn es sich bei der Warteschlange um eine Warteschlange für unzustellbare Nachrichten handelt, muss der Konsument zusätzlich die Berechtigung `kms:Decrypt` für jeden KMS-Schlüssel haben, mit dem die Nachrichten in der Quellwarteschlange verschlüsselt werden. Fügen Sie der IAM-Richtlinie des Konsumenten die folgende Anweisung hinzu. Denken Sie daran, die richtigen ARN-Werte für die Schlüsselressource und die Warteschlangenressource zu verwenden.

```
{
   "Version": "2012-10-17",
      "Statement": [{
         "Effect": "Allow",
         "Action": [
            "kms:Decrypt"
         ],
         "Resource": "arn:aws:kms:us-
east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
         }, {
         "Effect": "Allow",
         "Action": [
            "sqs:ReceiveMessage"
         ],
         "Resource": "arn:aws:sqs:*:123456789012:MyQueue"
      }]
}
```

Konfigurieren Sie AWS KMS Berechtigungen mit dem Schutz vor verwirrten Stellvertretern

Wenn der Prinzipal in einer Schlüsselrichtlinie ein AWS -Service-Prinzipal ist, können Sie die globalen Zustandsschlüssel `aws:SourceArn` oder `aws:SourceAccount` zum Schutz vor dem Confused Deputy Scenario verwenden. Um diese globalen Bedingungsschlüssel zu verwenden, legen Sie den Wert auf den Amazon-Ressourcennamen (ARN) oder das Konto der Ressource fest, die

verschlüsselt wird. Wenn Sie den ARN der Ressource nicht kennen, verwenden Sie stattdessen `aws:SourceAccount`.

In dieser KMS-Schlüsselrichtlinie darf eine bestimmte Ressource aus einem Service, der dem Konto `111122223333` gehört, KMS für die Aktionen `Decrypt` und `GenerateDataKey` Aktionen aufrufen, die während der SSE-Nutzung von Amazon SQS auftreten.

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Principal": {
            "Service": "<replaceable>service</replaceable>.amazonaws.com"
  },
  "Action": [
   "kms:GenerateDataKey",
   "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
   "ArnEquals": {
    "aws:SourceArn": [
     "arn:aws:service::111122223333:resource"
    ]
   }
  }
 }]
}
```

Bei Verwendung von SSE-fähigen Amazon-SQS-Warteschlangen unterstützen die folgenden Services `aws:SourceArn`:

- Amazon SNS
- Amazon S3
- CloudWatch Ereignisse
- AWS Lambda
- CodeBuild
- Amazon Connect Customer Profiles
- AWS Auto Scaling

- Amazon Chime

Grundlegendes zum Wiederverwendungszeitraum für den Datenschlüssel

Der Zeitraum für die Wiederverwendung des Datenschlüssels definiert die maximale Dauer für Amazon SQS zur Wiederverwendung desselben Datenschlüssels. Endet der Zeitraum der Datenschlüssel-Wiederverwendung, generiert Amazon SQS einen neuen Datenschlüssel. Beachten Sie die folgenden Richtlinien zum Wiederverwendungszeitraum.

- Ein kürzerer Wiederverwendungszeitraum bietet mehr Sicherheit, führt jedoch zu mehr Anrufen AWS KMS, wodurch Gebühren anfallen können, die über das kostenlose Kontingent hinausgehen.
- Obwohl der Datenschlüssel für die Verschlüsselung und Entschlüsselung separat zwischengespeichert wird, gilt der Wiederverwendungszeitraum für beide Kopien des Datenschlüssels.
- Wenn der Zeitraum für die Wiederverwendung von Datenschlüsseln endet, löst der nächste Aufruf von `SendMessage` oder `SendMessageBatch` in der Regel einen Aufruf der AWS KMS `GenerateDataKey` Methode aus, um einen neuen Datenschlüssel abzurufen. Außerdem lösen die nächsten Aufrufe jeweils einen Aufruf von aus, AWS KMS `Decrypt` um die Integrität des Datenschlüssels zu überprüfen, bevor er verwendet wird. `SendMessage` `ReceiveMessage`
- Prinzipale (AWS-Konten oder Benutzer) verwenden keine gemeinsamen Datenschlüssel (Nachrichten, die von eindeutigen Prinzipalen gesendet werden, erhalten immer eindeutige Datenschlüssel). Daher entspricht die Anzahl der Anrufe an AWS KMS ein Vielfaches der Anzahl der eindeutigen Prinzipale, die während der Wiederverwendung von Datenschlüsseln verwendet wurden.

Schätzung der Kosten AWS KMS

Um Kosten vorherzusagen und Ihre AWS Rechnung besser zu verstehen, möchten Sie vielleicht wissen, wie oft Amazon SQS Ihren KMS-Schlüssel verwendet.

> **Note**
>
> Mit der nachstehenden Formel erhalten Sie eine gute Vorstellung davon, welche Kosten auf Sie zukommen. Allerdings können die tatsächlichen Kosten aufgrund der verteilten Struktur von Amazon SQS höher liegen.

Zur Berechnung der Anzahl der API-Anfragen (R) pro Warteschlange verwenden Sie folgende Formel:

```
R = (B / D) * (2 * P + C)
```

B ist der Abrechnungszeitraum (in Sekunden).

D ist der Zeitraum für die Wiederverwendung des Datenschlüssels (in Sekunden).

P ist die Anzahl der produzierenden Prinzipale, die Nachrichten an die Amazon-SQS-Warteschlange senden.

C ist die Anzahl der konsumierenden Prinzipale, die Nachrichten aus der Amazon-SQS-Warteschlange erhalten.

> **Important**
>
> Für produzierende Prinzipale entstehen in der Regel doppelt so hohe Kosten wie für konsumierende Prinzipale. Weitere Informationen finden Sie unter Grundlegendes zum Wiederverwendungszeitraum für den Datenschlüssel.
> Die Kosten erhöhen sich, wenn der Produzent und der Verbraucher unterschiedliche - Benutzer haben.

Es folgen Beispielberechnungen: Preisinformationen finden Sie unter AWS Key Management Service -Preise.

Beispiel 1: Berechnung der Anzahl der AWS KMS API-Aufrufe für 2 Principals und 1 Warteschlange

In diesem Beispiel wird Folgendes angenommen:

- Der Abrechnungszeitraum ist 1. bis 31. Januar (2 678 400 Sekunden).
- Der Wiederverwendungszeitraum für den Datenschlüssel ist auf 5 Minuten (300 Sekunden) eingestellt.
- Es ist 1 Warteschlange vorhanden.
- Es gibt 1 produzierenden und 1 konsumierenden Prinzipal.

```
(2,678,400 / 300) * (2 * 1 + 1) = 26,784
```

Beispiel 2: Berechnung der Anzahl von AWS KMS API-Aufrufen für mehrere Produzenten und Verbraucher sowie für 2 Warteschlangen

In diesem Beispiel wird Folgendes angenommen:

- Der Abrechnungszeitraum ist 1. bis 28. Februar (2 419 200 Sekunden).
- Der Wiederverwendungszeitraum für den Datenschlüssel ist auf 24 Stunden (86 400 Sekunden) eingestellt.
- Es gibt 2 Warteschlangen.
- Die erste Warteschlange hat 3 produzierende Prinzipale und 1 konsumierenden Prinzipal.
- Die zweite Warteschlange hat 5 produzierende und 2 konsumierende Prinzipale.

```
(2,419,200 / 86,400 * (2 * 3 + 1)) + (2,419,200 / 86,400 * (2 * 5 + 2)) = 532
```

AWS KMS Fehler

Wenn Sie mit Amazon SQS und arbeiten AWS KMS, können Fehler auftreten. In den folgenden Referenzen werden die Fehler und möglichen Lösungen zur Fehlerbehebung beschrieben.

- Häufige AWS KMS -Fehler
- AWS KMS -Entschlüsselungsfehler
- AWS KMS GenerateDataKey Fehler

Datenschutz im Netzwerkverkehr in Amazon SQS

Ein Amazon Virtual Private Cloud (Amazon VPC)-Endpunkt für Amazon SQS ist eine logische Einheit innerhalb einer VPC, die nur Konnektivität mit Amazon SQS ermöglicht. Die VPC leitet Anforderungen an Amazon SQS weiter und Antworten an die VPC zurück. In den folgenden Abschnitten finden Sie Informationen zum Arbeiten mit VPC-Endpunkten und zum Erstellen von VPC-Endpunktrichtlinien.

Themen

- Endpunkte von Amazon Virtual Private Cloud für Amazon SQS
- Erstellen einer Amazon-VPC-Endpunkt-Richtlinie für Amazon SQS

Endpunkte von Amazon Virtual Private Cloud für Amazon SQS

Wenn Sie Amazon VPC zum Hosten Ihrer AWS Ressourcen verwenden, können Sie eine Verbindung zwischen Ihrer VPC und Amazon SQS herstellen. Sie können diese Verbindung zum Senden von Nachrichten an Ihre Amazon-SQS-Warteschlangen ganz ohne das öffentliche Internet verwenden.

Mit Amazon VPC können Sie AWS Ressourcen in einem benutzerdefinierten virtuellen Netzwerk starten. Mit einer VPC können Sie Netzwerkeinstellungen, wie IP-Adressbereich, Subnetze, Routing-Tabellen und Netzwerk-Gateways, steuern. Weitere Informationen zu VPCs finden Sie im Amazon-VPC-Benutzerhandbuch.

Um Ihre VPC mit Amazon SQS zu verbinden, müssen Sie zunächst einen Schnittstellen-VPC-Endpunkt definieren, der eine Verbindung zwischen Ihrer VPC mit anderen AWS -Services ermöglicht. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität zu Amazon SQS, ohne dass ein Internet-Gateway, eine Network Address Translation (NAT)-Instance oder eine VPN-Verbindung erforderlich ist. Weitere Informationen finden Sie unter Tutorial: Senden einer Nachricht an eine Amazon-SQS-Warteschlange aus Amazon Virtual Private Cloud und Beispiel 5: Verweigerung des Zugriffs, wenn dieser nicht von einem VPC-Endpunkt aus erfolgt in diesem Handbuch und unter Interface-VPC-Endpunkte (AWS PrivateLink) im Amazon-VPC-Benutzerhandbuch.

> **Important**
>
> - Sie können Amazon Virtual Private Cloud nur mit HTTPS-Amazon-SQS-Endpunkten verwenden.
> - Wenn Sie Amazon SQS konfigurieren, um Nachrichten von Amazon VPC zu senden, müssen Sie privates DNS aktivieren und Endpunkte im `sqs.us-east-2.amazonaws.com`-Format angeben.
> - Ein privates DNS unterstützt keine Legacy-Endpunkte wie z. B. `queue.amazonaws.com` oder `us-east-2.queue.amazonaws.com`.

Erstellen einer Amazon-VPC-Endpunkt-Richtlinie für Amazon SQS

Sie können eine Richtlinie für Amazon-VPC-Endpunkte für Amazon SQS erstellen, in der Sie Folgendes angeben:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können

- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon VPC-Benutzerhandbuch.

Im folgenden Beispiel für eine VPC-Endpunkt-Richtlinie wird angegeben, dass der `MyUser`-Benutzer Nachrichten an die Amazon-SQS-Warteschlange `MyQueue` senden darf.

```
{
   "Statement": [{
      "Action": ["sqs:SendMessage"],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Principal": {
        "AWS": "arn:aws:iam:123456789012:user/MyUser"
      }
   }]
}
```

Folgendes wird abgelehnt:

- Andere Amazon-SQS-API-Aktionen, z. B. `sqs:CreateQueue` und `sqs:DeleteQueue`.
- Andere -Benutzer und -Regeln, die versuchen, diesen VPC-Endpunkt zu verwenden.
- `MyUser` Senden von Nachrichten an eine andere Amazon-SQS-Warteschlange.

> **Note**
>
> Der Benutzer kann nach wie vor andere Amazon-SQS-API-Aktionen von außerhalb der VPC verwenden. Weitere Informationen finden Sie unter Beispiel 5: Verweigerung des Zugriffs, wenn dieser nicht von einem VPC-Endpunkt aus erfolgt.

Identity and Access Management in Amazon SQS

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf AWS Ressourcen sicher zu kontrollieren. IAM-Administratoren steuern, wer authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) ist, Amazon-SQS-Ressourcen zu nutzen. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie in Amazon SQS ausführen.

Service-Benutzer – Wenn Sie den Amazon-SQS-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie zur Ausführung von Aufgaben weitere Amazon-SQS-Features verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie auf ein Feature in Amazon SQS nicht zugreifen können, siehe Beheben von Identitäts- und Zugriffsfehlern bei Amazon Simple Queue Service.

Service-Administrator – Wenn Sie in Ihrem Unternehmen für die Amazon-SQS-Ressourcen zuständig sind, haben Sie wahrscheinlich vollen Zugriff auf Amazon SQS. Ihre Aufgabe besteht darin, zu bestimmen, auf welche Amazon-SQS-Features und -Ressourcen Ihre Servicebenutzer zugreifen sollten. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit Amazon SQS verwenden kann, finden Sie unter So funktioniert Amazon Simple Notification Service mit IAM.

IAM-Administrator – Wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht Details darüber erfahren, wie Sie Richtlinien zur Verwaltung des Zugriffs auf Amazon SQS erstellen können. Beispiele für identitätsbasierte Amazon-SQS-Richtlinien, die Sie in IAM verwenden können, finden Sie unter Bewährte Methoden für Richtlinien.

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAM-Benutzerhandbuch unter AWS API-Anfragen signieren.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine

Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS

CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie unter Kontenübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige verwenden Funktionen in anderen. AWS-Services AWS-Services Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon-EC2 aus oder speichert Objekte in Amazon-S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über

Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Auf Amazon EC2 ausgeführte Anwendungen — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und API-Anfragen stellen AWS CLI . AWS Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen

in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service Control Policies (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Entitäten. Root-Benutzer des AWS-Kontos

Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

Übersicht über die Zugriffsverwaltung in Amazon SQS

Jede AWS Ressource gehört einem AWS-Konto, und die Berechtigungen zum Erstellen oder Zugreifen auf eine Ressource werden durch Berechtigungsrichtlinien geregelt. Ein Kontoadministrator kann IAM-Identitäten (d. h. Benutzern, Gruppen und Rollen) Berechtigungsrichtlinien zuweisen. Manche Services (wie etwa Amazon SQS) unterstützen auch die Zuweisung von Berechtigungsrichtlinien zu Ressourcen.

> **Note**
>
> Ein Kontoadministrator (oder Administratorbenutzer) ist ein Benutzer mit Administratorberechtigungen. Weitere Informationen finden Sie unter Bewährte Methoden für IAM im IAM-Benutzerhandbuch.

Beim Erteilen von Berechtigungen geben Sie an, wer die Berechtigungen erhält, für welche Ressourcen die Berechtigungen gelten und welche Aktionen an dieser Ressource gestattet werden sollen.

Themen

- Ressourcen und Abläufe von Amazon Simple Queue Service
- Grundlegendes zum Eigentum an Ressourcen
- Verwalten des Zugriffs auf Ressourcen
- Angeben der Richtlinienelemente: Aktionen, Effekte, Ressourcen und Prinzipale

Ressourcen und Abläufe von Amazon Simple Queue Service

In Amazon SQS ist die einzige Ressource die Warteschlange. In einer Richtlinie identifizieren Sie die Ressource, für welche die Richtlinie gilt, mithilfe eines Amazon-Ressourcennamens (ARN). Der folgenden Ressource ist ein eindeutiger ARN zugewiesen:

Ressourcentyp	ARN-Format
Warteschlange	`arn:aws:sqs:` *`region`*`:`*`account_id`*`:`*`queue_name`*

Es folgen Beispiele des ARN-Formats für Warteschlangen:

- Ein ARN für eine Warteschlange mit dem Namen `my_queue` in der Region USA Ost (Ohio), die zum AWS Konto 123456789012 gehört:

  ```
  arn:aws:sqs:us-east-2:123456789012:my_queue
  ```

- Ein ARN für eine Warteschlange mit dem Namen `my_queue` in den verschiedenen Regionen, die von Amazon SQS unterstützt werden:

  ```
  arn:aws:sqs:*:123456789012:my_queue
  ```

- Eine ARN, der * oder ? als Platzhalter für den Warteschlangennamen verwendet. In den folgenden Beispielen entspricht der ARN allen Warteschlangen mit dem Präfix `my_prefix_`:

  ```
  arn:aws:sqs:*:123456789012:my_prefix_*
  ```

Sie können den ARN-Wert für eine vorhandene Warteschlange abrufen, indem Sie die Aktion `GetQueueAttributes` aufrufen. Der Wert des `QueueArn`-Attributs ist der ARN der Warteschlange. Weitere Informationen zu ARNs finden Sie unter IAM-ARNs im IAM-Benutzerhandbuch.

Amazon SQS bietet eine Reihe von Aktionen für die Arbeit mit der Warteschlangenressource. Weitere Informationen finden Sie unter Amazon-SQS-API-Berechtigungen: Referenz für Aktionen und Ressourcen.

Grundlegendes zum Eigentum an Ressourcen

Das AWS-Konto besitzt die Ressourcen, die im Konto erstellt wurden, unabhängig davon, wer die Ressourcen erstellt hat. Genauer gesagt ist der Ressourceneigentümer das AWS-Konto der Prinzipal-Entität (d. h. das Root-Konto, ein Benutzer oder eine IAM-Rolle), die die Anfrage zur Erstellung der Ressource authentifiziert. Die Funktionsweise wird anhand der folgenden Beispiele deutlich:

- Wenn Sie Ihre Root-Kontoanmeldedaten verwenden, AWS-Konto um eine Amazon SQS-Warteschlange zu erstellen, sind Sie AWS-Konto der Eigentümer der Ressource (in Amazon SQS ist die Ressource die Amazon SQS SQS-Warteschlange).
- Wenn Sie in Ihrer Datenbank einen Benutzer erstellen AWS-Konto und diesem Benutzer die Berechtigung zum Erstellen einer Warteschlange erteilen, kann der Benutzer die Warteschlange erstellen. Eigentümer der Warteschlangenressource ist jedoch Ihr AWS-Konto (zu dem der Benutzer gehört).
- Wenn Sie eine IAM-Rolle in Ihrem AWS-Konto mit den Berechtigungen zum Erstellen einer Amazon SQS SQS-Warteschlange erstellen, kann jeder, der die Rolle übernehmen kann, eine Warteschlange erstellen. Ihnen AWS-Konto (zu der die Rolle gehört) gehört die Warteschlangenressource.

Verwalten des Zugriffs auf Ressourcen

Eine Berechtigungsrichtlinie beschreibt die Konten zugewiesenen Berechtigungen. Im folgenden Abschnitt werden die verfügbaren Optionen zum Erstellen von Berechtigungsrichtlinien erläutert.

> **Note**
>
> Dieser Abschnitt behandelt die Verwendung von IAM im Zusammenhang mit Amazon SQS. Er enthält keine detaillierten Informationen über den IAM-Service. Eine umfassende IAM-Dokumentation finden Sie unter Was ist IAM? im IAM-Benutzerhandbuch. Für Informationen über die Syntax und Beschreibungen von AWS -IAM-Richtlinien lesen Sie die IAM-Richtlinienreferenz im IAM-Benutzerhandbuch.

Richtlinien, die einer IAM-Identität zugeordnet sind, werden als identitätsbasierte Richtlinien (IAM-Richtlinien) bezeichnet, während Richtlinien, die einer Ressource zugeordnet sind, ressourcenbasierte Richtlinien genannt werden.

Identitätsbasierte Richtlinien

Es gibt zwei Möglichkeiten, Ihren Benutzern Berechtigungen für Ihre Amazon-SQS-Warteschlangen zu erteilen: das Amazon-SQS-Richtliniensystem und das IAM-Richtliniensystem. Sie können entweder eines der Systeme oder beide verwenden, um Benutzern oder Rollen Richtlinien anzufügen. In den meisten Fällen erzielen Sie mit beiden Systemen dasselbe Ergebnis. Sie können z. B. Folgendes tun:

- Einem Benutzer oder einer Gruppe in Ihrem Konto eine Berechtigungsrichtlinie zuweisen – Wenn Sie einem Benutzer Berechtigungen zur Erstellung einer Amazon-SQS-Warteschlange erteilen möchten, können Sie dem Benutzer oder der Gruppe, zu der er gehört, eine Berechtigungsrichtlinie zuweisen.
- Einem Benutzer in einem anderen AWS-Konto eine Berechtigungsrichtlinie zuweisen – Zum Erteilen von Benutzerberechtigungen zum Erstellen einer Amazon-SQS-Warteschlange fügen Sie einem Benutzer in einem anderen AWS-Konto eine Amazon-SQS-Berechtigungsrichtlinie an.

  Kontoübergreifende Berechtigungen gelten nicht für die folgenden Aktionen:

  - `AddPermission`
  - `CancelMessageMoveTask`
  - `CreateQueue`
  - `DeleteQueue`
  - `ListMessageMoveTask`
  - `ListQueues`
  - `ListQueueTags`
  - `RemovePermission`
  - `SetQueueAttributes`
  - `StartMessageMoveTask`
  - `TagQueue`
  - `UntagQueue`
- Einer Rolle eine Berechtigungsrichtlinie zuweisen (kontoübergreifende Berechtigungen erteilen) – Sie können einer IAM-Rolle eine identitätsbasierte Berechtigungsrichtlinie zuweisen,

um kontoübergreifende Berechtigungen zu erteilen. Der AWS-Konto A-Administrator kann beispielsweise wie folgt eine Rolle erstellen, um AWS-Konto B (oder einem AWS Dienst) kontenübergreifende Berechtigungen zu gewähren:

- Der Administrator von Konto A erstellt eine IAM-Rolle und fügt ihr eine Berechtigungsrichtlinie an, die Berechtigungen für Ressourcen in Konto A erteilt.
- Der Administrator von Konto A weist der Rolle eine Vertrauensrichtlinie zu, die Konto B als den Prinzipal identifiziert, der die Rolle übernehmen kann.
- Administrator von Konto B delegiert die Berechtigung zur Übernahme der Rolle an Benutzer in Konto B. So können Benutzer in Konto B Warteschlangen in Konto A erstellen bzw. darauf zugreifen.

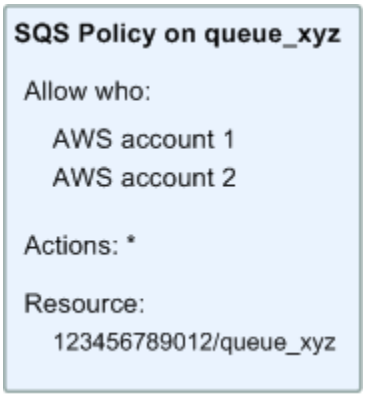
> **Note**
>
> Wenn Sie einem AWS Dienst die Berechtigung zur Übernahme der Rolle erteilen möchten, kann der Principal in der Vertrauensrichtlinie auch ein AWS Dienstprinzipal sein.

Weitere Informationen zum Delegieren von Berechtigungen mithilfe von IAM finden Sie unter Zugriffsverwaltung im IAM-Benutzerhandbuch.

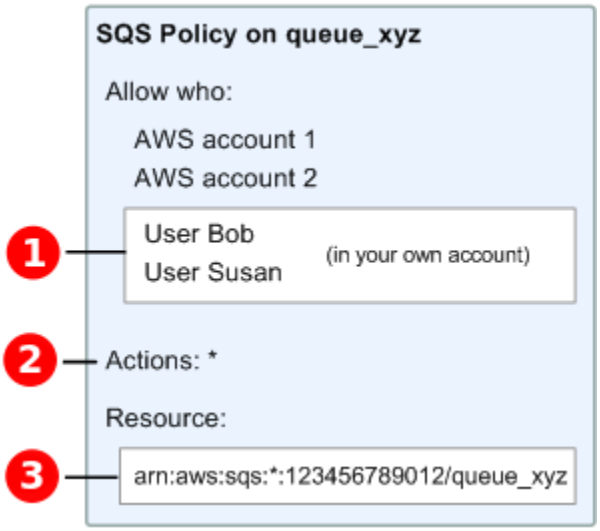
Amazon SQS arbeitet zwar mit IAM-Richtlinien, verfügt jedoch über eine eigene Richtlinieninfrastruktur. Sie können eine Amazon SQS SQS-Richtlinie mit einer Warteschlange verwenden, um anzugeben, welche AWS Konten Zugriff auf die Warteschlange haben. Sie können die Art des Zugriffs sowie Bedingungen festlegen (z. B. eine Bedingung, mit der Berechtigungen für `SendMessage` und für `ReceiveMessage` erteilt werden, wenn die Anforderung vor dem 31. Dezember 2010 gestellt wurde). Die spezifischen Aktionen, für die Sie Berechtigungen erteilen können, gelten für eine Untergruppe der gesamten Liste der Amazon-SQS-Aktionen. Wenn Sie eine Amazon-SQS-Richtlinie schreiben und für * „alle Amazon-SQS-Aktionen zulassen“ festlegen, bedeutet dies, dass alle Aktionen dieser Untergruppe von einzelnen Benutzern ausgeführt werden können.

Das folgende Diagramm veranschaulicht das Konzept einer dieser grundlegenden Amazon-SQS-Richtlinien, die für die Untergruppe der Aktionen gelten. Die Richtlinie gilt für `queue_xyz` und gibt AWS Konto 1 und AWS Konto 2 die Erlaubnis, jede der zulässigen Aktionen mit der angegebenen Warteschlange zu verwenden.

> **Note**
>
> Die Ressource in der Richtlinie ist wie folgt angegeben: `123456789012/queue_xyz` Dabei `123456789012` handelt es sich um die AWS Konto-ID des Kontos, dem die Warteschlange gehört.



Seit der Einführung von IAM und der Konzepte Benutzer und Amazon-Ressourcennamen (ARNs) gibt es einige Änderungen in Bezug auf die SQS-Richtlinien. Im folgenden Diagramm und in der Tabelle werden die Änderungen beschreiben.



1 Informa

zur Erteilung von Berechtigungen an Benutzer mit unterschiedlichen Konten finden Sie unter Tutorial: Kontenübergreifender Delegieren des Zugriffs mithilfe von IAM-Rollen im IAM-Benutzerhandbuch.
AWS

2

Die Untergruppe der Aktionen in * wurde erweitert. Eine Liste zulässiger Aktionen finden Sie unter Amazon-SQS-API-Berechtigungen: Referenz für Aktionen und Ressourcen.

3

Sie können die Ressource mithilfe des Amazon-Ressourcennamens (ARN) angeben. Dies entspricht dem standardmäßigen Verfahren zum Festlegen von Ressourcen in IAM-Richtlinien. Weitere Informationen zum ARN-Format für Amazon-SQS-Warteschlangen finden Sie unter Ressourcen und Abläufe von Amazon Simple Queue Service.

Gemäß der Amazon SQS SQS-Richtlinie im obigen Diagramm kann beispielsweise jeder, der die Sicherheitsanmeldedaten für AWS Konto 1 oder AWS Konto 2 besitzt, darauf zugreifen`queue_xyz`. Außerdem haben die Benutzer Bob und Susan in Ihrem eigenen AWS -Konto (Benutzer-ID `123456789012`) Zugriff auf die Warteschlange.

Vor der Einführung von IAM übertrug Amazon SQS automatisch die vollständige Kontrolle über die Warteschlange auf den jeweiligen Ersteller der Warteschlange (d. h. Zugriff auf alle möglichen Amazon-SQS-Aktionen für diese Warteschlange). Dies trifft nur noch zu, wenn der Ersteller AWS -Sicherheitsanmeldeinformationen verwendet. Jeder Benutzer, der über Berechtigungen zum Erstellen einer Warteschlange verfügt, muss zudem Berechtigungen zur Verwendung anderer Amazon-SQS-Aktionen haben, um Aktionen für die erstellten Warteschlangen ausführen zu können.

Es folgt ein Beispiel für eine Richtlinie, mit der ein Benutzer zwar alle Amazon-SQS-Aktionen verwenden kann, jedoch für Warteschlangen, deren Namen mit dem Präfix der Literalzeichenfolge `bob_queue_` versehen sind.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:*:123456789012:bob_queue_*"
   }]
}
```

Weitere Informationen finden Sie unter Richtlinien mit Amazon SQS verwenden und Identitäten (Benutzer, Gruppen und Rollen) im IAM-Benutzerhandbuch.

Angeben der Richtlinienelemente: Aktionen, Effekte, Ressourcen und Prinzipale

Für jede Amazon-Simple-Queue-Service-Ressource definiert der Service eine Reihe von Aktionen. Zur Erteilung von Berechtigungen für diese Aktionen definiert Amazon SQS eine Reihe von Aktionen, die Sie in einer Richtlinie angeben können.

> **Note**
>
> Für das Durchführen einer Aktion können Berechtigungen für mehrere Aktionen erforderlich sein. Bei der Erteilung von Berechtigungen für bestimmte Aktionen geben Sie auch die Ressource an, für die die Aktionen zugelassen oder verweigert werden.

Grundlegende Richtlinienelemente:

- Ressource – In einer Richtlinie wird der Amazon-Ressourcenname (ARN) zur Identifizierung der Ressource verwendet, für die die Richtlinie gilt.
- Aktion – Mit Aktionsschlüsselwörtern geben Sie die Ressourcenaktionen an, die Sie zulassen oder verweigern möchten. Die `sqs:CreateQueue`-Berechtigung erteilt dem Benutzer zum Beispiel Berechtigungen zum Ausführen der Amazon-Simple-Queue-Service-Aktion `CreateQueue`.
- Auswirkung – Die von Ihnen festgelegte Auswirkung, wenn der Benutzer die jeweilige Aktion anfordert – entweder „allow“ (Zugriffserlaubnis) oder „deny“ (Zugriffsverweigerung). Wenn Sie den Zugriff auf eine Ressource nicht ausdrücklich gestatten (""), wird er automatisch verweigert. Sie können den Zugriff auf eine Ressource auch explizit verweigern. So können Sie sicherstellen, dass Benutzer nicht darauf zugreifen können, auch wenn der Zugriff durch eine andere Richtlinie gestattet wird.
- Prinzipal – In identitätsbasierten Richtlinien (IAM-Richtlinien) ist der Benutzer, dem die Richtlinie zugewiesen ist, automatisch der Prinzipal. In ressourcenbasierten Richtlinien müssen Sie den Benutzer, das Konto, den Service oder die sonstige Entität angeben, die die Berechtigungen erhalten soll (gilt nur für ressourcenbasierte Richtlinien).

Weitere Informationen zur Syntax und zu Beschreibungen von Amazon-SQS-Richtlinien finden Sie in der AWS -IAM-Richtlinienreferenz im IAM-Benutzerhandbuch.

Eine Tabellenliste mit allen Amazon-Simple-Queue-Service-Aktionen und den Ressourcen, für die diese gelten, finden Sie unter Amazon-SQS-API-Berechtigungen: Referenz für Aktionen und Ressourcen.

So funktioniert Amazon Simple Notification Service mit IAM

Bevor Sie mit IAM den Zugriff auf Amazon SQS verwalten können, sollten Sie sich darüber informieren, welche IAM-Features Sie mit Amazon SQS verwenden können.

IAM-Features, die Sie mit Amazon Simple Queue Service verwenden können

IAM-Feature	Amazon-SQS-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Ja
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (services pezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Teilweise
Temporäre Anmeldeinformationen	Ja
Forward Access Sessions (FAS)	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie Amazon SQS und andere AWS Services mit den meisten IAM-Funktionen funktionieren, finden Sie im AWS IAM-Benutzerhandbuch unter Services, die mit IAM funktionieren.

Zugriffskontrolle

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

> **Note**
>
> Es ist wichtig zu verstehen, dass alle ihre Berechtigungen an Benutzer unter ihren Konten delegieren AWS-Konten können. Durch den kontoübergreifenden Zugriff können Sie den Zugriff auf Ihre AWS Ressourcen gemeinsam nutzen, ohne zusätzliche Benutzer verwalten zu müssen. Weitere Informationen über die Verwendung des kontenübergreifenden Zugriffs finden Sie unter Enabling Cross-Account Access im IAM-Benutzerhandbuch.
> Weitere Informationen zu inhaltsübergreifenden Berechtigungen und Bedingungsschlüsseln in den benutzerdefinierten Amazon-SQS-Richtlinien finden Sie unter Einschränkungen der benutzerdefinierten Amazon SQS SQS-Richtlinien.

Identitätsbasierte Richtlinien für Amazon SQS

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet

ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Amazon SQS

Beispiele für identitätsbasierte Amazon-SQS-Richtlinien finden Sie unter Bewährte Methoden für Richtlinien.

Ressourcenbasierte Richtlinien in Amazon SQS

Unterstützt ressourcenbasierte Richtlinien	Ja

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM-Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Kontenübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Richtlinienaktionen für Amazon SQS

Unterstützt Richtlinienaktionen	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Amazon-SQS-Aktionen finden Sie unter Von Amazon Simple Queue Service definierte Ressourcen in der Service-Autorisierungs-Referenz.

Richtlinienaktionen in Amazon SQS verwenden das folgende Präfix vor der Aktion:

```
sqs
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "sqs:action1",
      "sqs:action2"
         ]
```

Beispiele für identitätsbasierte Amazon-SQS-Richtlinien finden Sie unter Bewährte Methoden für Richtlinien.

Richtlinienressourcen für Amazon SQS

Unterstützt Richtlinienressourcen	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Amazon-SQS-Ressourcentypen und ihrer ARNs finden Sie unter Von Amazon Simple Queue Service definierte Aktionen in der Service-Autorisierungs-Referenz. Um zu erfahren, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, lesen Sie Von Amazon Simple Queue Service definierte Ressourcen.

Beispiele für identitätsbasierte Amazon-SQS-Richtlinien finden Sie unter Bewährte Methoden für Richtlinien.

Richtlinien-Bedingungsschlüssel für Amazon SQS

Unterstützt servicespezifische Richtlini enbedingungsschlüssel	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte

Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen AND-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen OR Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der Amazon-SQS-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon Simple Queue Service in der Service-Autorisierungs-Referenz. Informationen dazu, mit welchen Aktionen und Ressourcen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon Simple Queue Service definierte Ressourcen.

Beispiele für identitätsbasierte Amazon-SQS-Richtlinien finden Sie unter Bewährte Methoden für Richtlinien.

ACLs in Amazon SQS

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Amazon SQS

Unterstützt ABAC (Tags in Richtlinien)	Teilweise

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Verwenden temporärer Anmeldeinformationen mit Amazon SQS

Unterstützt temporäre Anmeldeinformationen	Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen, die mit temporären Anmeldeinformationen AWS-Services funktionieren AWS-Services , finden Sie im IAM-Benutzerhandbuch unter Diese Option funktioniert mit IAM.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Passwort anmelden. Wenn Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM-Benutzerhandbuch.

Mithilfe der AWS API AWS CLI oder können Sie temporäre Anmeldeinformationen manuell erstellen. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Zugriffssitzungen für Amazon SQS weiterleiten

Unterstützt Forward Access Sessions (FAS)	Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Amazon SQS

Unterstützt Servicerollen	Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle könnte die Funktionalität von Amazon SQS beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Amazon SQS dazu Anleitungen gibt.

Serviceverknüpfte Rollen für Amazon SQS

Unterstützt serviceverknüpfte Rollen	Nein

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Amazon SQS SQS-Updates für AWS verwaltete Richtlinien

Um Benutzern, Gruppen und Rollen Berechtigungen hinzuzufügen, ist es einfacher, von AWS verwaltete Richtlinien zu verwenden, als selbst Richtlinien zu schreiben. Es erfordert Zeit und Fachwissen, um von Kunden verwaltete IAM-Richtlinien zu erstellen, die Ihrem Team nur die benötigten Berechtigungen bieten. Um schnell loszulegen, können Sie unsere von AWS verwalteten Richtlinien verwenden. Diese Richtlinien decken häufige Anwendungsfälle ab und sind in Ihrem AWS -Konto verfügbar. Weitere Informationen zu AWS verwalteten Richtlinien finden Sie unter AWS Verwaltete Richtlinien im IAM-Benutzerhandbuch.

AWS Dienste verwalten und aktualisieren AWS verwaltete Richtlinien. Sie können die Berechtigungen in AWS verwalteten Richtlinien nicht ändern. Dienste fügen einer AWS verwalteten Richtlinie gelegentlich zusätzliche Berechtigungen hinzu, um neue Funktionen zu unterstützen. Diese Art von Update betrifft alle Identitäten (Benutzer, Gruppen und Rollen), an welche die Richtlinie angehängt ist. Es ist sehr wahrscheinlich, dass Dienste eine AWS verwaltete Richtlinie aktualisieren, wenn eine neue Funktion eingeführt wird oder wenn neue Operationen verfügbar werden. Dienste entfernen keine Berechtigungen aus einer AWS verwalteten Richtlinie, sodass durch Richtlinienaktualisierungen Ihre bestehenden Berechtigungen nicht beeinträchtigt werden.

AWS Unterstützt außerdem verwaltete Richtlinien für Jobfunktionen, die sich über mehrere Dienste erstrecken. Die AWS verwaltete ReadOnlyAccess-Richtlinie bietet beispielsweise Lesezugriff auf alle AWS Dienste und Ressourcen. Wenn ein Dienst ein neues Feature startet, werden nur

Leseberechtigungen für neue Operationen und Ressourcen AWS hinzugefügt. Eine Liste und Beschreibungen der Richtlinien für Auftragsfunktionen finden Sie in Verwaltete AWS -Richtlinien für Auftragsfunktionen im IAM-Leitfaden.

AWS verwaltete Richtlinie: AmazonSQS FullAccess

Sie können die `AmazonSQSFullAccess`-Richtlinie an Ihre Amazon-SQS-Identitäten anfügen. Diese Richtlinie gewährt Berechtigungen, die vollen Zugriff auf Amazon SQS ermöglichen.

Die Berechtigungen für diese Richtlinie finden Sie unter AmazonSQS FullAccess in der Referenz zu AWS verwalteten Richtlinien.

AWS verwaltete Richtlinie: AmazonSQS Access ReadOnly

Sie können die `AmazonSQSReadOnlyAccess`-Richtlinie an Ihre Amazon-SQS-Identitäten anfügen. Diese Richtlinie gewährt Berechtigungen, die einen schreibgeschützten Zugriff auf Amazon SQS erlauben.

Die Berechtigungen für diese Richtlinie finden Sie unter AmazonSQS ReadOnly Access in der Referenz zu AWS verwalteten Richtlinien.

Amazon SQS SQS-Updates für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Amazon SQS an, seit dieser Service begonnen hat, diese Änderungen zu verfolgen. Um automatische Warnungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der Amazon-SQS-Dokumentverlauf-Seite.

Änderung	Beschreibung	Datum
Zugriff auf Amazon ReadOnly SQS	Amazon SQS hat eine neue Aktion hinzugefügt, mit der Sie die neuesten Aufgaben zur Nachrichtenverschiebung (bis zu 10) in einer bestimmte n Quellwarteschlange auflisten können. Diese Aktion ist mit dem `ListMessageMoveTas ks` -API-Vorgang verknüpft.	9. Juni 2023

Beheben von Identitäts- und Zugriffsfehlern bei Amazon Simple Queue Service

Diagnostizieren und beheben Sie mithilfe der folgenden Informationen gängige Probleme, die bei der Verwendung von Amazon SQS und IAM auftreten können.

Themen

- Ich bin nicht autorisiert, eine Aktion in Amazon SQS auszuführen
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Amazon SQS SQS-Ressourcen ermöglichen

Ich bin nicht autorisiert, eine Aktion in Amazon SQS auszuführen

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen einer Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um die Aktion durchführen zu können.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson`-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `sqs:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sqs:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Mateo-Richtlinie aktualisiert werden, damit er mit der `sqs:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Amazon SQS übergeben zu können.

In einigen AWS-Services Fällen können Sie eine bestehende Rolle an diesen Dienst übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon SQS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Amazon SQS SQS-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob Amazon SQS diese Features unterstützt, finden Sie unter So funktioniert Amazon Simple Notification Service mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie im IAM-Benutzerhandbuch unter Kontenübergreifender Ressourcenzugriff in IAM.

Richtlinien mit Amazon SQS verwenden

In diesem Thema finden Sie Beispiele für identitätsbasierte Richtlinien, in denen ein Kontoadministrator den IAM-Identitäten (Benutzern, Gruppen und Rollen) Berechtigungsrichtlinien zuweisen kann.

> **Important**
>
> Wir empfehlen Ihnen, zunächst die einführenden Themen zu lesen, in denen die Grundkonzepte und die für Sie verfügbaren Optionen zum Verwalten des Zugriffs auf Ihre Amazon-Simple-Queue-Service-Ressourcen erläutert werden. Weitere Informationen finden Sie unter Übersicht über die Zugriffsverwaltung in Amazon SQS.
> Mit Ausnahme von `ListQueues` unterstützen alle Amazon-SQS-Aktionen Berechtigungen auf Ressourcenebene. Weitere Informationen finden Sie unter Amazon-SQS-API-Berechtigungen: Referenz für Aktionen und Ressourcen.

Themen

- Verwenden von Amazon-SQS- und IAM-Richtlinien
- Erforderliche Berechtigungen zur Verwendung der Amazon-SQS-Konsole
- Beispiele für identitätsbasierte Richtlinien für Amazon SQS
- Grundlegende Beispiele für Amazon-SQS-Richtlinien
- Verwenden von benutzerdefinierten Richtlinien mit der Sprache der Zugriffsrichtlinie von Amazon SQS

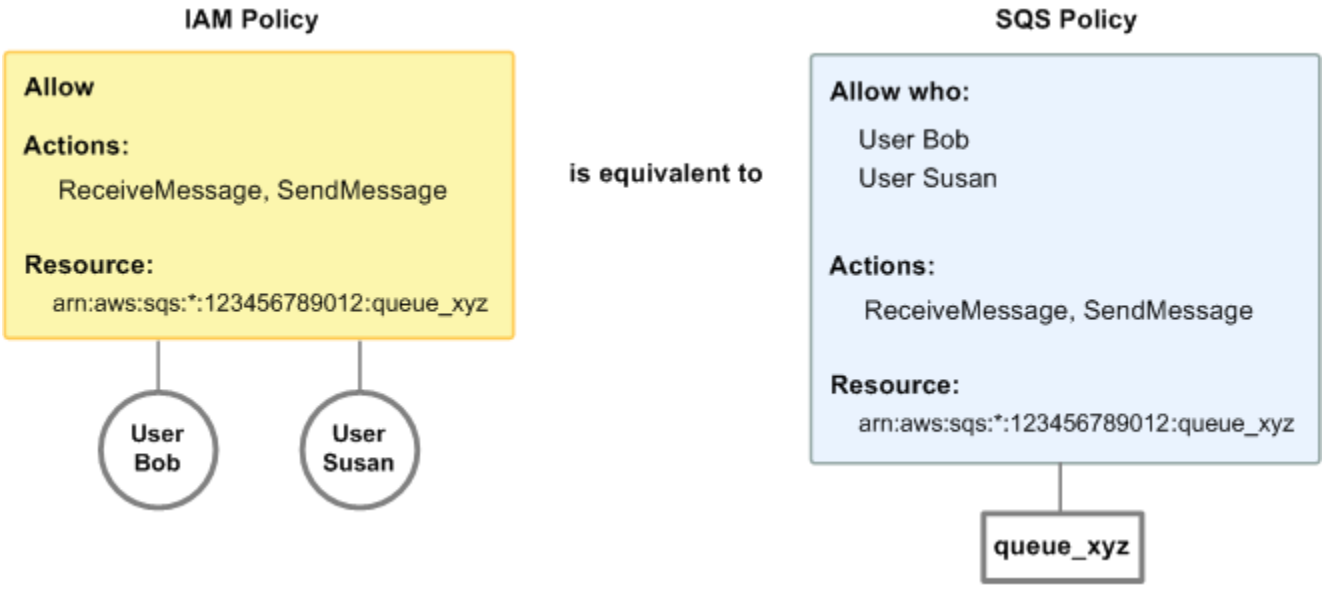
Verwenden von Amazon-SQS- und IAM-Richtlinien

Es gibt zwei Möglichkeiten, Ihren Benutzern Berechtigungen für Ihre Amazon-SQS-Ressourcen zu erteilen: das Amazon-SQS-Richtliniensystem und das IAM-Richtliniensystem. Sie können entweder das eine oder das andere oder beide Systeme verwenden. In den meisten Fällen erzielen Sie mit beiden Systemen dasselbe Ergebnis.

Die folgende Abbildung zeigt eine IAM-Richtlinie und eine entsprechende Amazon-SQS-Richtlinie. Die IAM-Richtlinie gewährt die Rechte an Amazon SQS `ReceiveMessage` und `SendMessage` Aktionen für die Warteschlange, die `queue_xyz` in Ihrem AWS Konto aufgerufen wurde, und die Richtlinie gilt für Benutzer mit den Namen Bob und Susan (Bob und Susan verfügen über die in

der Richtlinie angegebenen Berechtigungen). Diese Amazon-SQS-Richtlinie erteilt Bob und Susan Berechtigungen zum Ausführen der Aktionen `ReceiveMessage` und `SendMessage` für dieselbe Warteschlange.

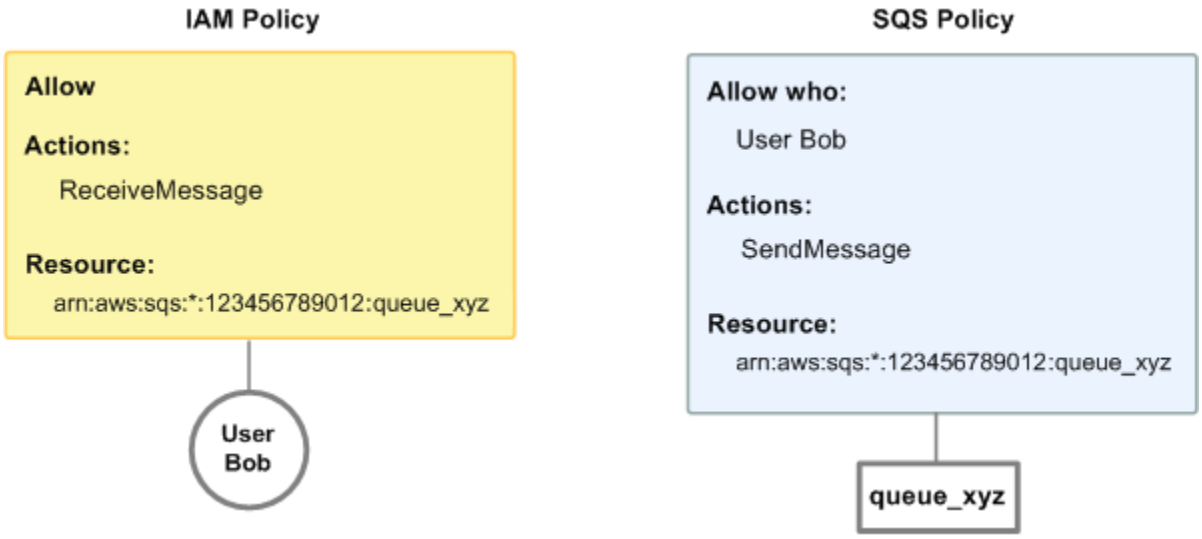
> **Note**
>
> Das folgende Beispiel zeigt einfache Richtlinien ohne Bedingungen. Sie können eine bestimmte Bedingung in einer der Richtlinien angeben und erhalten dasselbe Ergebnis.



Es gibt einen wesentlichen Unterschied zwischen IAM- und Amazon SQS SQS-Richtlinien: Mit dem Amazon SQS SQS-Richtliniensystem können Sie anderen AWS Konten Berechtigungen erteilen, während dies bei IAM nicht der Fall ist.

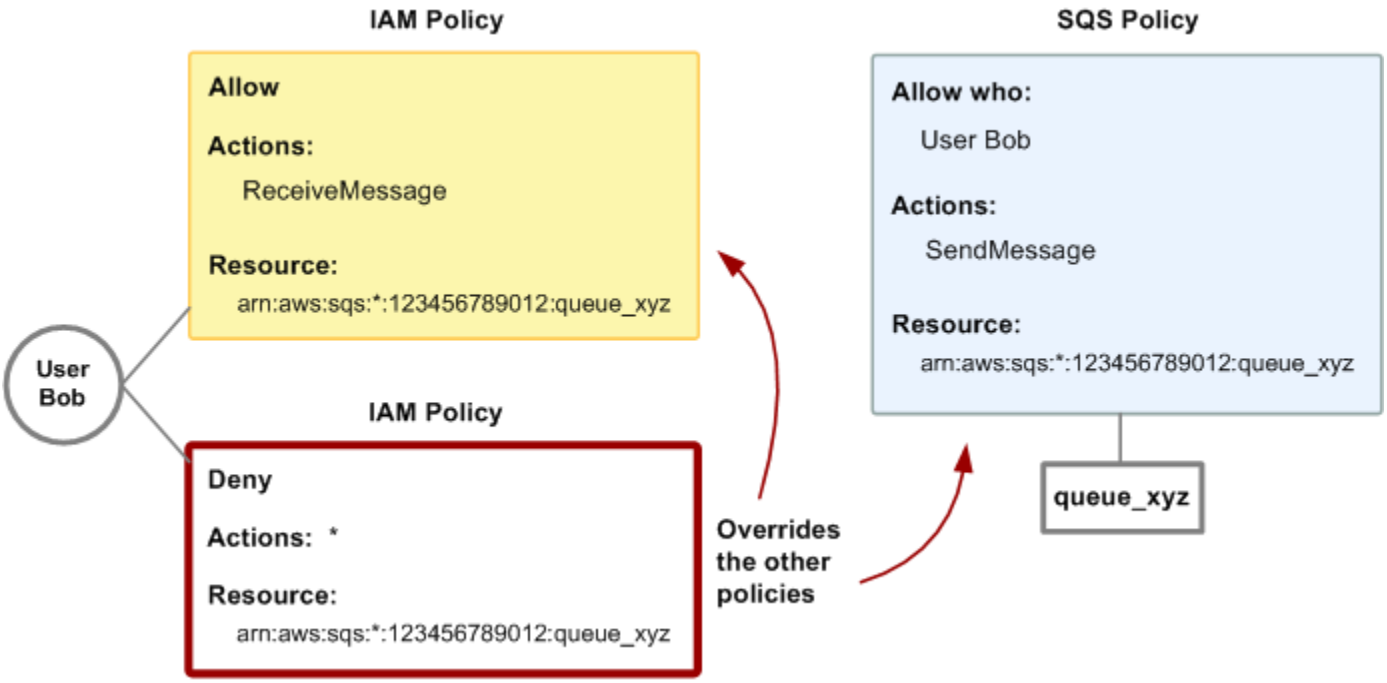
Sie entscheiden, wie Sie beide Systeme zum Verwalten von Berechtigungen verwenden. Die folgenden Beispiele zeigen, wie die beiden Richtliniensysteme zusammenarbeiten.

- Im ersten Beispiel verfügt Bob sowohl über eine IAM-Richtlinie als auch über eine Amazon-SQS-Richtlinie, die für sein Konto gelten. Die IAM-Richtlinie erteilt seinem Konto die Berechtigung für die Aktion `ReceiveMessage` für `queue_xyz`, während die Amazon-SQS-Richtlinie sein Konto berechtigt, die Aktion `SendMessage` für dieselbe Warteschlange auszuführen. Das folgende Diagramm verdeutlicht das Konzept.



Wenn Bob eine `ReceiveMessage`-Anforderung an `queue_xyz` sendet, lässt die IAM-Richtlinie die Aktion zu. Wenn Bob eine `SendMessage`-Anforderung an `queue_xyz` sendet, lässt die Amazon-SQS-Richtlinie die Aktion zu.

- Im zweiten Beispiel missbraucht Bob seinen Zugriff auf `queue_xyz`, sodass es nötig wird, seinen gesamten Zugriff auf die Warteschlange zu verweigern Der einfachste Weg ist, eine Richtlinie hinzuzufügen, die ihm den Zugriff auf alle Aktionen für die Warteschlange verweigert. Diese Richtlinie setzt die beiden anderen außer Kraft, da ein explizites `deny` ein `allow`immer überschreibt. Weitere Informationen zur Richtlinienauswertungslogik finden Sie unter Verwenden von benutzerdefinierten Richtlinien mit der Sprache der Zugriffsrichtlinie von Amazon SQS. Das folgende Diagramm verdeutlicht das Konzept.



Sie können der Amazon-SQS-Richtlinie auch eine Anweisung hinzufügen, die Bob alle Zugriffe auf die Warteschlange verweigert. Dies hat die gleiche Auswirkung wie das Hinzufügen einer IAM-Richtlinie, die Bob den Zugriff auf die Warteschlange verweigert. Beispiele von Richtlinien, die Amazon-SQS-Aktionen und -Ressourcen abdecken, finden Sie unter Grundlegende Beispiele für Amazon-SQS-Richtlinien. Weitere Informationen zum Erstellen von Amazon-SQS-Richtlinien finden Sie unter Verwenden von benutzerdefinierten Richtlinien mit der Sprache der Zugriffsrichtlinie von Amazon SQS.

Erforderliche Berechtigungen zur Verwendung der Amazon-SQS-Konsole

Ein Benutzer, der mit der Amazon-SQS-Konsole arbeiten möchte, muss über die Mindestmenge an Berechtigungen verfügen, die es ihm erlauben, die Amazon-SQS-Warteschlangen im AWS-Konto des Benutzers zu verwenden. Beispielsweise muss der Benutzer über die Berechtigung zum Aufruf der Aktion `ListQueues` verfügen, um Warteschlangen aufzulisten, oder über die Berechtigung zum Aufruf der Aktion `CreateQueue`, um Warteschlangen erstellen zu können. Zusätzlich zu den Amazon-SQS-Berechtigungen erfordert die Konsole zum Abonnieren einer Amazon-SQS-Warteschlange für ein Amazon-SNS-Thema Berechtigungen für Amazon-SNS-Aktionen.

Wenn Sie eine IAM-Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole möglicherweise nicht wie vorgesehen für Benutzer mit dieser IAM-Richtlinie.

Sie müssen Benutzern, die nur die Aktionen AWS CLI oder Amazon SQS aufrufen, keine Mindestberechtigungen für die Konsole gewähren.

Beispiele für identitätsbasierte Richtlinien für Amazon SQS

Benutzer und Rollen besitzen standardmäßig keine Berechtigungen zum Erstellen oder Ändern von Amazon-SQS-Ressourcen. Sie können auch keine Aufgaben mithilfe der AWS Management Console, AWS Command Line Interface (AWS CLI) oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von Amazon SQS definiert werden, einschließlich des Formats der ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Simple Queue Service in der Service-Authorisierungs-Referenz.

> **Note**
>
> Wenn Sie Lebenszyklus-Hooks für Amazon EC2 Auto Scaling konfigurieren, müssen Sie keine Richtlinie schreiben, um Nachrichten an eine Amazon-SQS-Warteschlange zu senden. Weitere Informationen finden Sie unter Amazon EC2 Auto Scaling Lifecycle Hooks im Amazon EC2 EC2-Benutzerhandbuch.

Themen

- Bewährte Methoden für Richtlinien
- Verwenden der Amazon-SQS-Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer
- Erlauben Sie einem Benutzer, Warteschlangen zu erstellen
- Erlauben Sie Entwicklern, Nachrichten in eine gemeinsam genutzte Warteschlange zu schreiben
- Erlauben Sie Managern, die allgemeine Größe von Warteschlangen zu ermitteln
- Erlauben Sie einem Partner, Nachrichten an eine bestimmte Warteschlange zu senden

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien können festlegen, ob jemand Amazon-SQS-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder daraus löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.

- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Amazon-SQS-Konsole

Um auf die Konsole von Amazon Simple Queue Service zugreifen zu können, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Amazon SQS SQS-Ressourcen in Ihrem AWS-Konto aufzulisten und anzuzeigen. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen weiterhin die Amazon SQS SQS-Konsole verwenden können, fügen Sie den Entitäten auch die `AmazonSQSReadOnlyAccess` AWS verwaltete Amazon SQS SQS-Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder 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": "*"
        }
    ]
}
```

Erlauben Sie einem Benutzer, Warteschlangen zu erstellen

Im folgenden Beispiel wird eine Richtlinie für den Benutzer Bob erstellt, mit der er zwar auf alle Amazon-SQS-Aktionen zugreifen kann, aber nur mit Warteschlangen, deren Namen mit dem Präfix der Literalzeichenfolge `alice_queue_` versehen sind.

Amazon SQS gewährt dem Ersteller einer Warteschlange nicht automatisch Berechtigungen zum Verwenden der Warteschlange. Daher müssen wir Bob explizit Berechtigungen zum Verwenden aller Amazon-SQS-Aktionen zusätzlich zur Aktion `CreateQueue` in der IAM-Richtlinie erteilen.

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

```
      "Resource": "arn:aws:sqs:*:123456789012:alice_queue_*"
   }]
}
```

Erlauben Sie Entwicklern, Nachrichten in eine gemeinsam genutzte Warteschlange zu schreiben

Im folgenden Beispiel erstellen wir eine Gruppe für Entwickler und fügen eine Richtlinie hinzu, die es der Gruppe ermöglicht, die Amazon SQS `SendMessage` SQS-Aktion zu verwenden, aber nur mit der Warteschlange, die zu der angegebenen gehört AWS-Konto und benannt `MyCompanyQueue` ist.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:*:123456789012:MyCompanyQueue"
   }]
}
```

Sie können * anstelle von `SendMessage` verwenden, um die folgenden Aktionen auf einen Prinzipal auf einer gemeinsamen Warteschlange zu gewähren: `ChangeMessageVisibility`, `DeleteMessageGetQueueAttributes`, `GetQueueUrl`, `ReceiveMessage` und `SendMessage`.

> **Note**
>
> Obwohl * den von den anderen Berechtigungstypen bereitgestellten Zugriff beinhaltet, werden Berechtigungen von Amazon SQS als separat betrachtet. Es ist beispielsweise möglich, einem Benutzer sowohl die Berechtigung * als auch die Berechtigung `SendMessage` zu erteilen, obwohl ein * den von `SendMessage` bereitgestellten Zugriff gewährt.
>
> Dieses Konzept gilt auch, wenn Sie eine Berechtigung entfernen. Wenn einem Prinzipal lediglich eine *-Berechtigung gewährt wurde, verfügt er durch die Anforderung zum Entfernen einer `SendMessage`-Berechtigung nicht ausschließlich über eine alles außer-Berechtigung. Stattdessen hat die Anforderung keine Auswirkungen, da der Prinzipal keine explizite `SendMessage`-Berechtigung besitzt. Wenn dem Prinzipal lediglich die `ReceiveMessage`-Berechtigung erteilt werden soll, fügen Sie die `ReceiveMessage`-Berechtigung hinzu und entfernen Sie die *-Berechtigung.

Erlauben Sie Managern, die allgemeine Größe von Warteschlangen zu ermitteln

Im folgenden Beispiel erstellen wir eine Gruppe für Manager und fügen eine Richtlinie hinzu, die es der Gruppe ermöglicht, die Amazon SQS `GetQueueAttributes` SQS-Aktion mit allen Warteschlangen zu verwenden, die zu dem angegebenen AWS Konto gehören.

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

Erlauben Sie einem Partner, Nachrichten an eine bestimmte Warteschlange zu senden

Sie können diese Aufgabe mit einer Amazon-SQS- oder einer IAM-Richtlinie ausführen. Wenn Ihr Partner über eine verfügt AWS-Konto, ist es möglicherweise einfacher, eine Amazon SQS SQS-Richtlinie zu verwenden. Jeder Benutzer im Unternehmen des Partners, der über die AWS Sicherheitsanmeldedaten verfügt, kann jedoch Nachrichten an die Warteschlange senden. Wenn Sie den Zugriff auf einen bestimmten Benutzer oder eine Anwendung beschränken möchten, müssen Sie den Partner wie einen Benutzer in Ihrem eigenen Unternehmen behandeln und eine IAM-Richtlinie anstelle einer Amazon-SQS-Richtlinie verwenden.

Dieses Beispiel führt die folgenden Aktionen aus:

1. Erstellen Sie eine Gruppe WidgetCo , die das Partnerunternehmen repräsentiert.
2. Erstellt einen Benutzer für den jeweiligen Benutzer oder die Anwendung des Unternehmens des Partners, der bzw. die Zugriff erfordert.
3. Fügen Sie den Benutzer zur Gruppe hinzu.
4. Fügt eine Richtlinie an, mit der die Gruppe nur Zugriff auf die Aktion `SendMessage` ausschließlich für die Warteschlange mit dem Namen `WidgetPartnerQueue` erhält.

```
{
   "Version": "2012-10-17",
   "Statement": [{
         "Effect": "Allow",
         "Action": "sqs:SendMessage",
```

```
         "Resource": "arn:aws:sqs:*:123456789012:WidgetPartnerQueue"
   }]
}
```

Grundlegende Beispiele für Amazon-SQS-Richtlinien

Dieser Abschnitt zeigt Richtlinienbeispiele für allgemeine Amazon-SQS-Anwendungsfälle.

Während Sie dem Benutzer die Richtlinien zuweisen, können Sie die Konsole verwenden, um die Auswirkungen der einzelnen Richtlinien zu überprüfen. Zunächst verfügt der Benutzer über keine Berechtigungen und kann in der Konsole keine Aktionen ausführen. Während Sie dem Benutzer Richtlinien zuweisen, können Sie überprüfen, ob der Benutzer die verschiedenen Aktionen in der Konsole ausführen kann.

> **Note**
>
> Es wird empfohlen, zwei Browserfenster zu verwenden: eines, um Berechtigungen zu erteilen, und das andere, um sich AWS Management Console mit den Anmeldeinformationen des Benutzers anzumelden, um die Berechtigungen zu überprüfen, während Sie sie dem Benutzer gewähren.

Beispiel 1: Erteilen Sie einem Benutzer eine Berechtigung AWS-Konto

Die folgende Beispielrichtlinie erteilt AWS-Konto number `111122223333` die `SendMessage` Berechtigung für die Warteschlange, die `444455556666/queue1` in der Region USA Ost (Ohio) benannt ist.

```
{
   "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"
```

```
   }]
}
```

Beispiel 2: Erteilen Sie einer Person zwei Berechtigungen AWS-Konto

Die folgende Beispielrichtlinie gewährt der genannten Warteschlange `111122223333` `SendMessage` sowohl die AWS-Konto Nummer als auch die `ReceiveMessage` Berechtigung`444455556666/queue1`.

```
{
   "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"
   }]
}
```

Beispiel 3: Erteilen Sie zwei Personen alle Berechtigungen AWS-Konten

Die folgende Beispielrichtlinie gewährt zwei verschiedene AWS-Konten Nummern (`111122223333`und`444455556666`) die Erlaubnis, alle Aktionen zu verwenden, für die Amazon SQS gemeinsamen Zugriff für die Warteschlange gewährt, die `123456789012/queue1` in der Region USA Ost (Ohio) benannt ist.

```
{
   "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"
   }]
}
```

Beispiel 4: Einer Rolle und einem Benutzernamen kontenübergreifende Berechtigungen erteilen

Die folgende Beispielrichtlinie gewährt `role1` einer Person `username1` unter der AWS-Konto Nummer `111122223333` kontoübergreifend die Erlaubnis, alle Aktionen zu verwenden, für die Amazon SQS gemeinsamen Zugriff auf die Warteschlange gewährt, die `123456789012/queue1` in der Region USA Ost (Ohio) benannt ist.

Kontoübergreifende Berechtigungen gelten nicht für die folgenden Aktionen:

- `AddPermission`
- `CancelMessageMoveTask`
- `CreateQueue`
- `DeleteQueue`
- `ListMessageMoveTask`
- `ListQueues`
- `ListQueueTags`
- `RemovePermission`
- `SetQueueAttributes`
- `StartMessageMoveTask`
- `TagQueue`
- `UntagQueue`

```
{
   "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"
   }]
}
```

Beispiel 5: Allen Benutzern eine Berechtigung erteilen

Mit der folgenden Beispielrichtlinie wird allen Benutzern (anonymen Benutzern) die Berechtigung `ReceiveMessage` für die Warteschlange mit dem Namen `111122223333/queue1` erteilt.

```
{
   "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"
   }]
}
```

Beispiel 6: Allen Benutzern eine zeitlich begrenzte Berechtigung erteilen

Das folgende Beispiel gewährt die Berechtigung `ReceiveMessage` allen Benutzern (anonymen Benutzern) der Warteschlange mit dem Namen `111122223333/queue1`, aber nur zwischen 12:00 Uhr und 15:00 Uhr am 31. Januar 2009.

```
{
   "Version": "2012-10-17",
   "Id": "Queue1_Policy_UUID",
   "Statement": [{
      "Sid":"Queue1_AnonymousAccess_ReceiveMessage_TimeLimit",
      "Effect": "Allow",
```

```
      "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"
         }
      }
   }]
}
```

Beispiel 7: Allen Benutzern in einem CIDR-Bereich sämtliche Berechtigungen erteilen

Die folgende Beispielrichtlinie erteilt allen Benutzern (anonymen Benutzern) die Berechtigung zur Verwendung aller möglichen Amazon-SQS-Aktionen, die für die Warteschlange mit dem Namen `111122223333/queue1` gemeinsam genutzt werden können, jedoch nur, wenn die Anfrage aus dem `192.0.2.0/24`-CIDR-Bereich kommt.

```
{
   "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"
         }
      }
   }]
}
```

Beispiel 8: Berechtigungen für Benutzer in verschiedenen CIDR-Bereichen über Zulassungslisten und Sperrlisten

Die folgende Beispielrichtlinie enthält zwei Anweisungen:

- Die erste Anweisung gewährt allen Benutzer (anonymen Benutzern) im CIDR-Bereich `192.0.2.0/24` (mit Ausnahme von `192.0.2.188`) die Berechtigung zur Verwendung der Aktion `SendMessage` für die Warteschlange mit dem Namen `111122223333/queue1`.
- Die zweite Anweisung verwehrt allen Benutzern (anonyme Benutzer) im CIDR-Bereich `12.148.72.0/23` die Nutzung der Warteschlange.

```
{
   "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"
         }
      }
   }]
}
```

Verwenden von benutzerdefinierten Richtlinien mit der Sprache der Zugriffsrichtlinie von Amazon SQS

Wenn Sie Amazon SQS SQS-Zugriff nur auf der Grundlage einer AWS-Konto ID und Grundberechtigungen (wie für `SendMessage`oder `ReceiveMessage`) gewähren möchten, müssen Sie keine eigenen Richtlinien schreiben. Sie können einfach die Amazon-SQS-Aktion `AddPermission` verwenden.

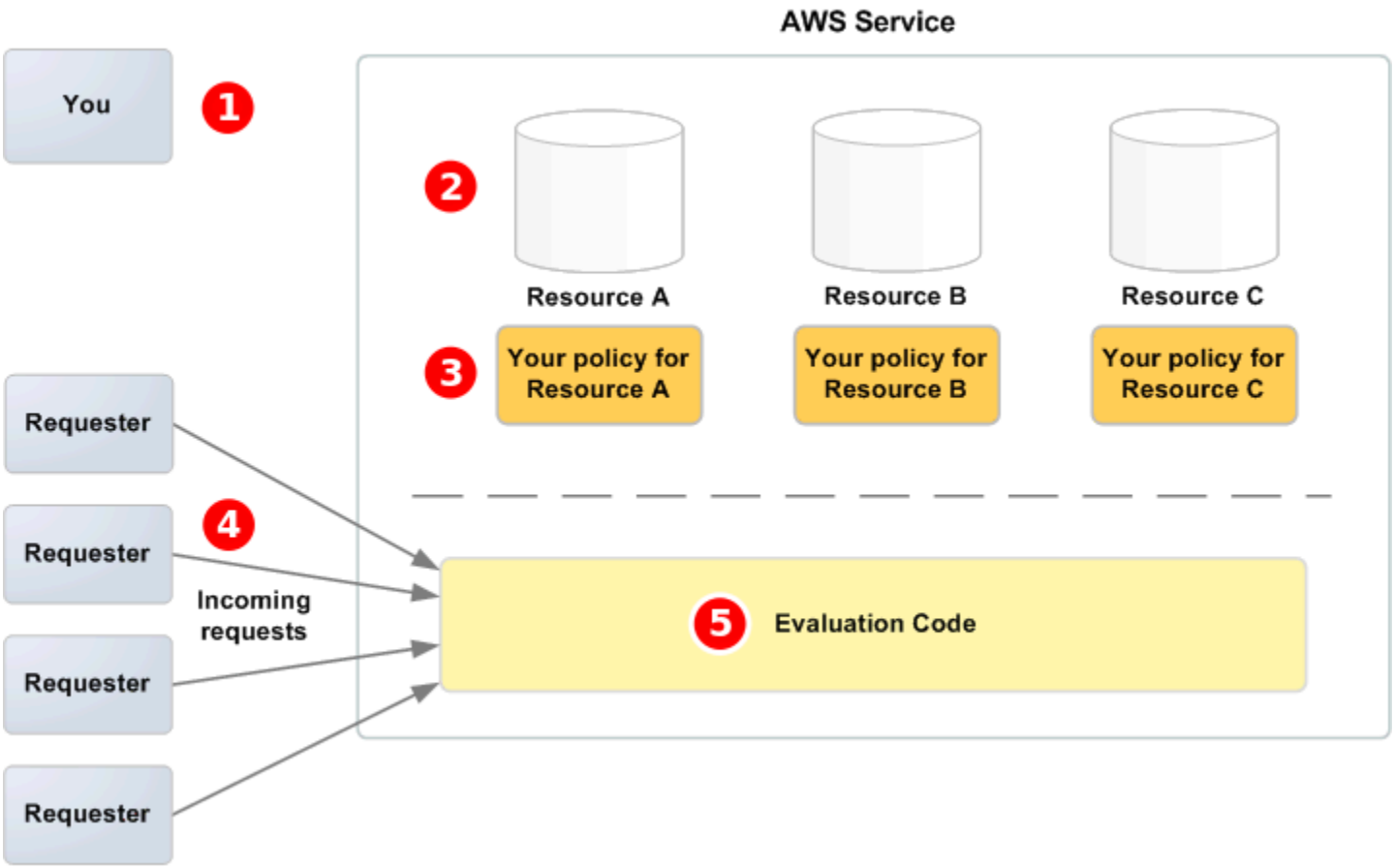
Wenn Sie den Zugriff auf der Grundlage spezifischerer Bedingungen (wie dem Zeitpunkt des Eingangs der Anfrage oder der IP-Adresse des Anfragenden) explizit verweigern oder zulassen möchten, müssen Sie Ihre eigenen Amazon SQS SQS-Richtlinien schreiben und diese mithilfe der Amazon SQS SQS-Aktion in das AWS System hochladen. `SetQueueAttributes`

Themen

- Amazon-SQS-Zugriffskontrollarchitektur
- Prozess-Workflow für die Amazon-SQS-Zugriffskontrolle
- Die wichtigsten Konzepte der Sprache der Zugriffsrichtlinie von Amazon SQS
- Bewertungslogik der Sprache der Zugriffsrichtlinie von Amazon SQS
- Beziehungen zwischen expliziten und standardmäßigen Ablehnungen in der Sprache der Zugriffsrichtlinie von Amazon SQS
- Einschränkungen der benutzerdefinierten Amazon SQS SQS-Richtlinien
- Beispiele für eine benutzerdefinierte Sprache der Zugriffsrichtlinie von Amazon SQS

Amazon-SQS-Zugriffskontrollarchitektur

Das folgende Diagramm beschreibt die Zugriffskontrolle für Ihre Amazon-SQS-Ressourcen.



1

Sie selbst als Ressourceneigentümer.

2

Ihre im AWS Service enthaltenen Ressourcen (z. B. Amazon SQS SQS-Warteschlangen).

3

Ihre Richtlinien. Es empfiehlt sich eine Richtlinie pro Ressource. Der AWS Service bietet eine API, mit der Sie Ihre Richtlinien hochladen und verwalten können.

4

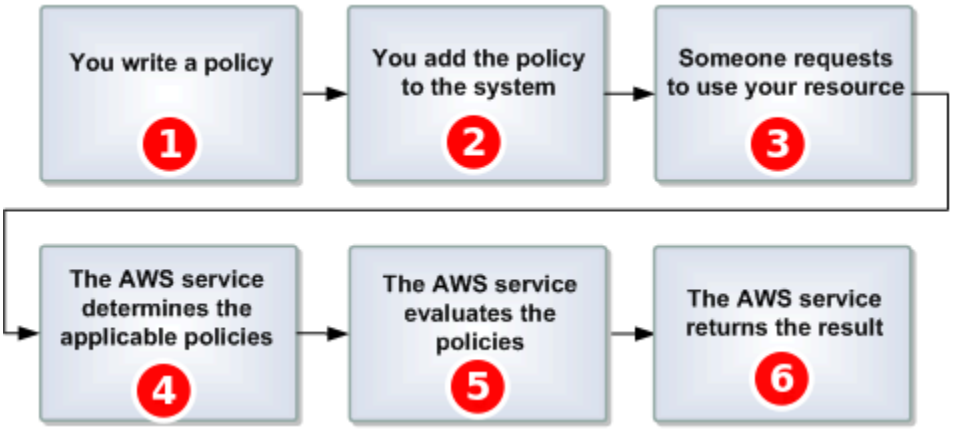
Anforderer und deren eingehende Anforderungen an den AWS -Service.

5

Der Code für die Auswertung der Sprache der Zugriffsrichtlinie. Dies ist der Codesatz innerhalb des AWS Dienstes, der eingehende Anfragen anhand der geltenden Richtlinien bewertet und bestimmt, ob dem Antragsteller Zugriff auf die Ressource gewährt wird.

Prozess-Workflow für die Amazon-SQS-Zugriffskontrolle

Das folgende Diagramm beschreibt den allgemeinen Workflow der Zugriffskontrolle der Sprache der Zugriffsrichtlinie von Amazon SQS.



1. Sie schreiben eine Amazon-SQS-Richtlinie für Ihre Warteschlange.
2. Sie laden Ihre Richtlinie auf AWS hoch. Der AWS Dienst stellt eine API bereit, die Sie zum Hochladen Ihrer Richtlinien verwenden. Sie verwenden beispielsweise die Amazon-SQS-Aktion `SetQueueAttributes` zum Hochladen einer Richtlinie für eine bestimmte Amazon-SQS-Warteschlange.
3. Jemand sendet eine Anforderung zur Verwendung Ihrer Amazon-SQS-Warteschlange.
4. Amazon SQS prüft alle verfügbaren Amazon-SQS-Richtlinien und bestimmt, welche zutreffend sind.
5. Amazon SQS bewertet die Richtlinien und bestimmt, ob der Anforderer Ihre Warteschlange verwenden darf.
6. Basierend auf dem Ergebnis der Richtlinienauswertung gibt Amazon SQS entweder einen `Access denied`-Fehler an den Anforderer zurück oder verarbeitet die Anforderung.

Die wichtigsten Konzepte der Sprache der Zugriffsrichtlinie von Amazon SQS

Zum Schreiben Ihrer eigenen Richtlinien müssen Sie mit JSON und einer Reihe von wichtigen Konzepten vertraut sein.

Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf

Das Ergebnis einer Statement, bei der Effect (Effekt) auf `allow` festgelegt ist.

Action (Aktion)

Die Aktivität, zu der Auftraggeber berechtigt ist, in der Regel eine Anforderung an AWS.

Default-deny

Das Ergebnis einer Statement, die weder über die Einstellung Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf noch Explicit-deny verfügt.

Bedingung

Jede Einschränkung oder jedes Detail zu einer Berechtigung. Bedingungen sind in der Regel auf das Datum, die Uhrzeit und IP-Adressen bezogen.

Effect (Effekt)

Das Ergebnis, die die Statement einer Richtlinie zu Auswertungszeit zurückgeben soll. Sie geben den Wert `deny` oder `allow` an, wenn Sie die Richtlinienanweisung schreiben. Bei der Richtlinienauswertung sind drei Ergebnisse möglich: Default-deny, Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf und Explicit-deny.

Explicit-deny

Das Ergebnis einer Statement, bei der Effect (Effekt) auf `deny` festgelegt ist.

Bewertung

Der Prozess, mit dem Amazon SQS ermittelt, ob eine eingehende Anforderung basierend auf einer Richtlinie verweigert oder erlaubt werden soll.

Aussteller

Der Benutzer, der eine Richtlinie erstellt, um einer Ressource Berechtigungen zu erteilen. Der Emittent ist per Definition immer der Eigentümer der Ressource. AWS erlaubt Amazon SQS SQS-Benutzern nicht, Richtlinien für Ressourcen zu erstellen, die ihnen nicht gehören.

Key (Schlüssel)

Das besondere Merkmal, das die Grundlage für die Einschränkung des Zugriffs bildet.

Berechtigung

Das Konzept, den Zugriff auf eine Ressource mit einer Bedingung und einem Key (Schlüssel) zuzulassen oder abzulehnen.

Richtlinie

Das Dokument dient als Container für eine oder mehrere Anweisungen.

Amazon SQS verwendet die Richtlinie, um zu bestimmen, ob einem Benutzer der Zugriff auf eine Ressource gewährt werden soll.

Auftraggeber

Der Benutzer, der die Berechtigung in der Richtlinie erhält.

Ressource

Das Objekt, auf das die Auftraggeber-Anforderungen zugreifen.

Statement

Die formelle Beschreibung einer einzelnen Berechtigung, die in der Sprache der Zugriffsrichtlinie als Teil eines umfassenderen Richtlinie-Dokuments verfasst ist.

Auftraggeber

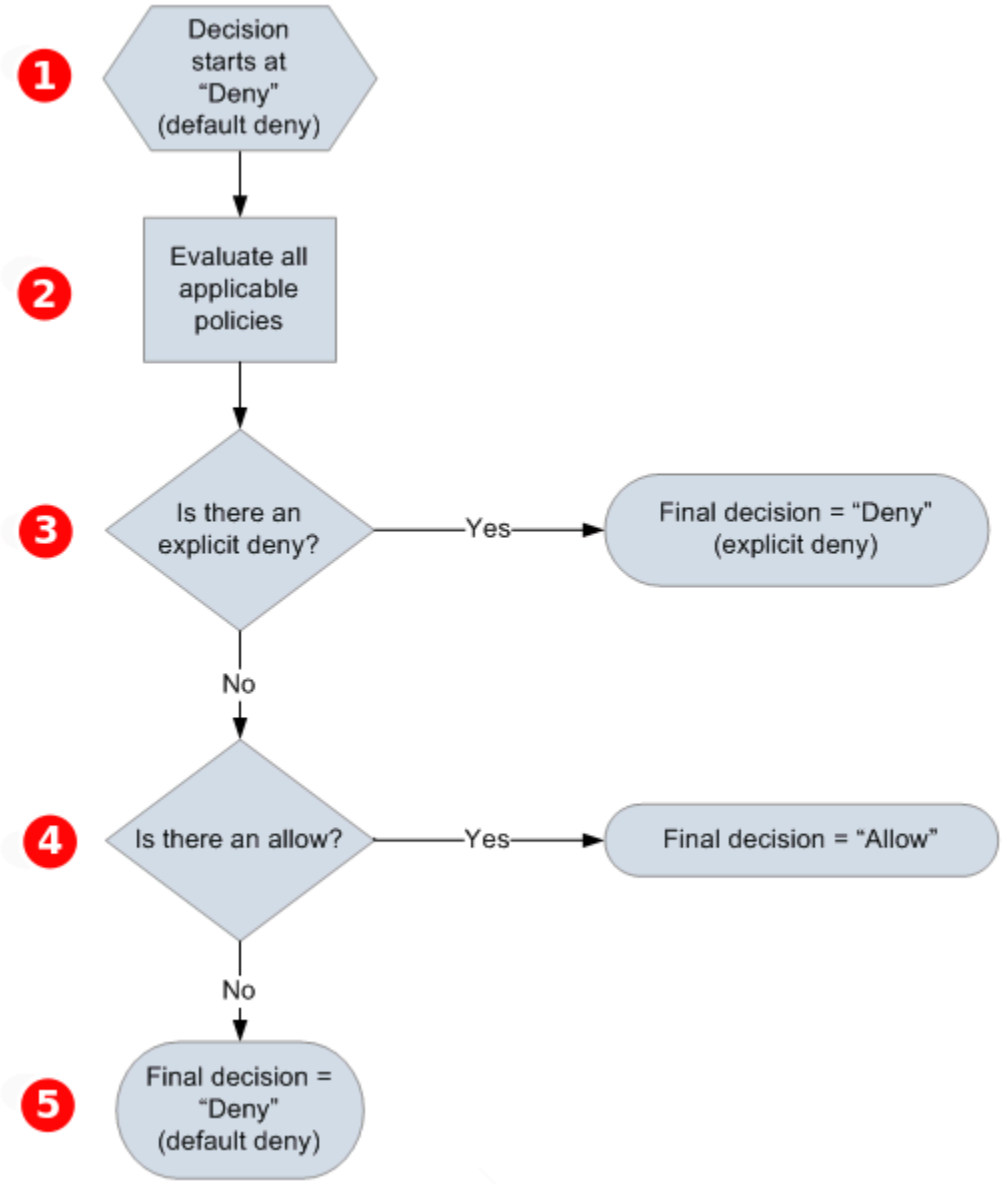
Der Benutzer, der eine Anforderung für den Zugriff auf eine Ressource sendet.

Bewertungslogik der Sprache der Zugriffsrichtlinie von Amazon SQS

Zum Zeitpunkt der Auswertung bestimmt Amazon SQS, ob eine Anforderung von einem anderen Benutzer als dem Ressourceneigentümer zugelassen oder abgelehnt werden soll. Die Auswertungslogik unterliegt mehreren Grundregeln:

- Standardmäßig werden alle Anforderungen zur Verwendung Ihrer Ressourcen, die nicht von Ihnen stammen, verweigert.
- Ein Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf setzt jedes Default-deny außer Kraft.
- Ein Explicit-deny setzt jedes allow außer Kraft.
- Es spielt keine Rolle, in welcher Reihenfolge die Richtlinien ausgewertet werden.

Im folgenden Diagramm wird detailliert beschrieben, wie Amazon SQS Entscheidungen im Hinblick auf Zugriffsberechtigungen trifft.



1. Die Entscheidung beginnt mit default-deny.

2. Es werden alle Richtlinien ausgewertet, die auf die Anforderung anwendbar sind (basierend auf der Ressource, dem Prinzipal, der Aktion und den Bedingungen). Es spielt keine Rolle, in welcher Reihenfolge der Durchführungscode die Richtlinien auswertet.

3. Der Durchführungscode sucht nach einer explicit-deny-Anweisung, die für die Anforderung gelten könnte. Wird eine Anweisung gefunden, gibt der Durchführungscode die Entscheidung deny zurück und der Prozess wird beendet.

4. Wenn keine explicit-deny-Anweisung gefunden wird, wird nach allow-Anweisungen gesucht, die auf die Anfrage zutreffen können. Wenn auch nur eine explizite Zugriffserlaubnis gefunden wird, wird allow zurückgegeben und der Prozess beendet (der Service setzt die Verarbeitung der Anforderung fort).

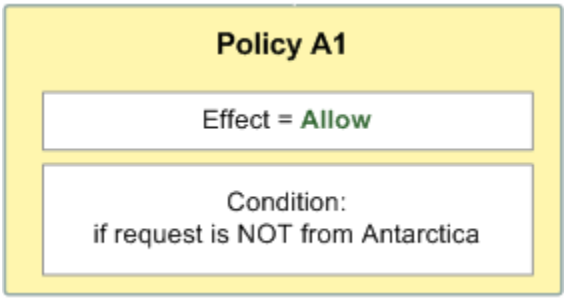
5. Wenn keine allow-Anweisung gefunden wird, ist die endgültige Entscheidung deny (da es kein explicit-deny bzw. allow gab, gilt dies als default-deny).

Beziehungen zwischen expliziten und standardmäßigen Ablehnungen in der Sprache der Zugriffsrichtlinie von Amazon SQS

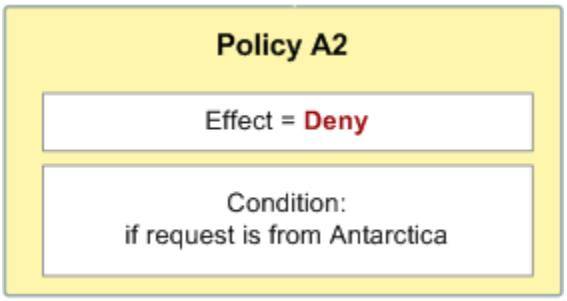
Wenn eine Amazon-SQS-Richtlinie nicht direkt auf eine Anforderung anwendbar ist, führt dies im Ergebnis zu Default-deny. Wenn ein Benutzer z. B. die Berechtigung zur Nutzung von Amazon SQS anfordert, aber die einzige für den Benutzer anwendbare Richtlinie DynamoDB verwenden kann, ist das Ergebnis default-deny.

Wenn eine Bedingung in einer Anweisung nicht erfüllt ist, hat die Anforderung default-deny zur Folge. Wenn alle Bedingungen einer Anweisung erfüllt sind, hat dies für diese Anforderung abhängig vom Wert des Elements Effect (Effekt) entweder Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf oder Explicit-deny zur Folge. In Richtlinien ist nicht festgelegt, was geschieht, wenn eine Bedingung nicht erfüllt ist, daher ist das Ergebnis in diesem Fall default-deny. Angenommen, Sie möchten Anforderungen, die aus der Antarktis stammen, ablehnen. Sie erstellen die Richtlinie A1,

um Zugriff nur dann zu gewähren, wenn Anforderungen von außerhalb der Antarktis kommen. Die Amazon-SQS-Richtlinie ist in der folgenden Abbildung dargestellt.

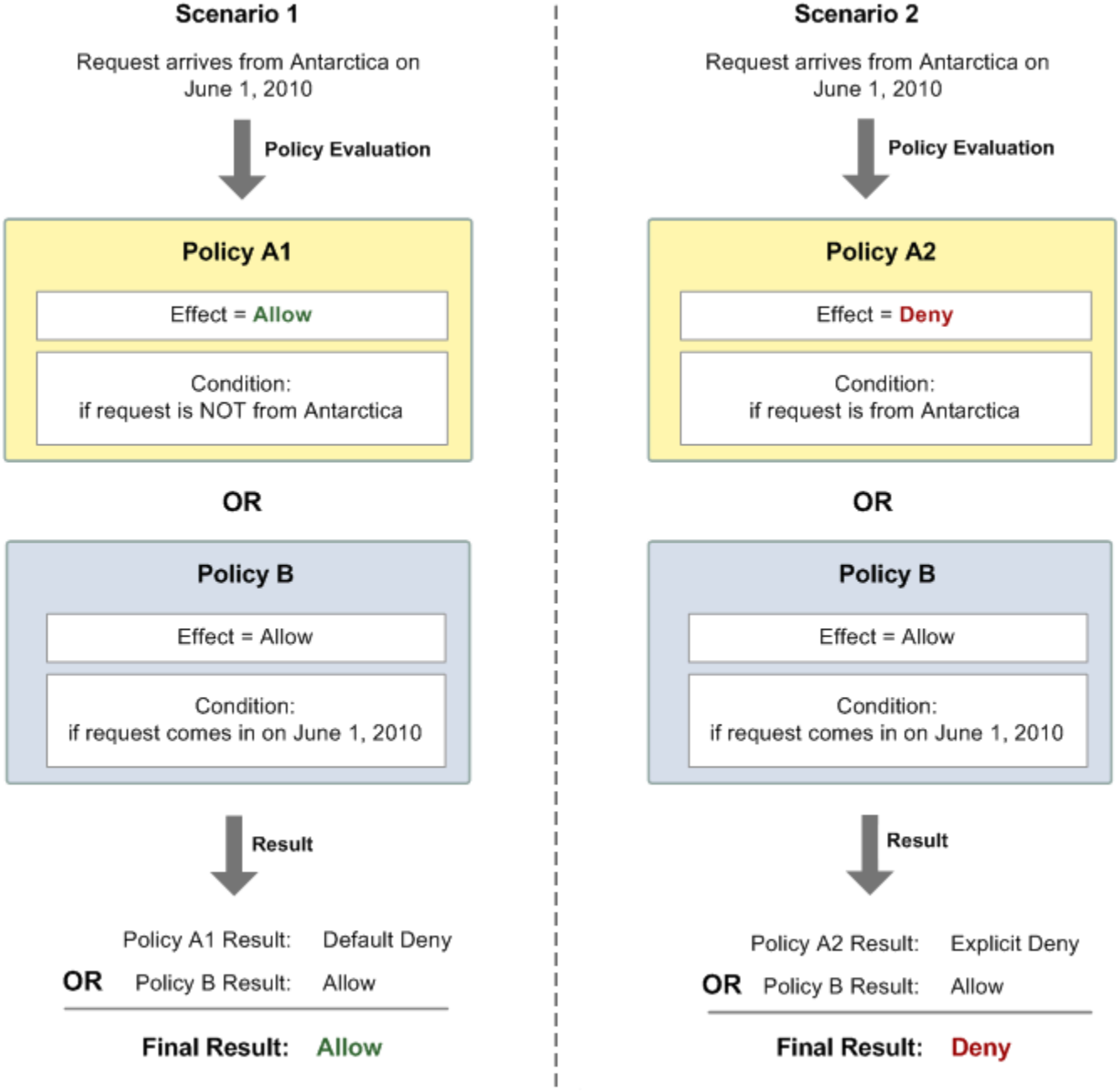


Wenn ein Benutzer eine Anforderung aus den USA sendet, wird die Bedingung erfüllt (die Anforderung stammt nicht aus der Antarktis), und die Anforderung hat allow zur Folge. Wenn ein Benutzer allerdings eine Anforderung aus der Antarktis sendet, ist die Bedingung nicht erfüllt und die Anforderung führt standardmäßig zu default-deny. Sie können das Ergebnis von default-deny ändern, indem Sie Richtlinie A2 erstellen, die einer Anforderung den Zugriff ausdrücklich verweigert, wenn sie aus der Antarktis stammt. Diese Richtlinie ist in der folgenden Abbildung dargestellt.



Wenn ein Benutzer eine Anforderung aus der Antarktis sendet, ist die Bedingung erfüllt und die Anforderung führt zu explicit-deny.

Der Unterschied zwischen default-deny und explicit-deny ist wichtig, da allow zwar eine standardmäßige Ablehnung, aber keine explizite Zugriffsverweigerung außer Kraft setzen kann. Beispiel: Richtlinie B lässt Anforderungen zu, wenn sie am 1. Juni 2010 eingehen. Das folgende Diagramm vergleicht die Kombination dieser Richtlinie mit Richtlinie A1 und Richtlinie A2.



In Szenario 1 liefert Richtlinie A1 das Ergebnis default-deny und Richtlinie B allow, weil die Richtlinie Anforderungen zulässt, die am 1. Juni 2010 eingehen. Das allow aus Richtlinie B überschreibt default-deny aus Richtlinie A1 und die Anforderung wird zugelassen.

In Szenario 2 hat Richtlinie B2 explicit-deny und Richtlinie B allow zur Folge. Das allow aus Richtlinie B wird durch explicit-deny aus Richtlinie A2 überschrieben und die Anforderung wird abgelehnt.

Einschränkungen der benutzerdefinierten Amazon SQS SQS-Richtlinien

Kontoübergreifender Zugriff

Kontoübergreifende Berechtigungen gelten nicht für die folgenden Aktionen:

- `AddPermission`
- `CancelMessageMoveTask`
- `CreateQueue`
- `DeleteQueue`
- `ListMessageMoveTask`
- `ListQueues`
- `ListQueueTags`
- `RemovePermission`
- `SetQueueAttributes`
- `StartMessageMoveTask`
- `TagQueue`
- `UntagQueue`

Bedingungsschlüssel

Derzeit unterstützt Amazon SQS nur einen eingeschränkten Teilbereich der in IAM verfügbaren Bedingungsschlüssel. Weitere Informationen finden Sie unter Amazon-SQS-API-Berechtigungen: Referenz für Aktionen und Ressourcen.

Beispiele für eine benutzerdefinierte Sprache der Zugriffsrichtlinie von Amazon SQS

Nachfolgend sind Beispiele typischer Amazon-SQS-Zugriffsrichtlinien aufgeführt.

Beispiel 1: Einem Konto eine Berechtigung erteilen

Das folgende Beispiel für eine Amazon-SQS-Richtlinie gewährt AWS-Konto 111122223333 die Berechtigung zum Senden an und zum Empfangen von `queue2`, in Eigentümerschaft von AWS-Konto 444455556666.

```
{
   "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"
   }]
}
```

Beispiel 2: Einem oder mehreren Konten eine Berechtigung erteilen

Die folgende Amazon SQS SQS-Beispielrichtlinie gewährt eine oder mehrere AWS-Konten Zugriff auf Warteschlangen, die Ihrem Konto gehören, für einen bestimmten Zeitraum. Es ist erforderlich, diese Richtlinie zu erstellen und mit der Aktion `SetQueueAttributes` zu Amazon SQS hochzuladen, da die Aktion `AddPermission` bei der Zugriffserteilung für eine Warteschlange keine Angabe einer Zeitbeschränkung erlaubt.

```
{
   "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"
         }
      }
   }]
}
```

Beispiel 3: Berechtigung für Anforderungen von Amazon-EC2-Instances erteilen

Das folgende Beispiel für eine Amazon-SQS-Richtlinie erteilt den Zugriff für Anforderungen, die von Amazon-EC2-Instances stammen. Dieses Beispiel basiert auf dem Beispiel "Beispiel 2: Einem oder mehreren Konten eine Berechtigung erteilen": Es beschränkt den Zugriff auf die Zeit vor dem 30. Juni 2009 12.00 Uhr (UTC) und auf den IP-Adressbereich `203.0.113.0/24`. Es ist erforderlich, diese Richtlinie zu erstellen und mit der Aktion `SetQueueAttributes` in Amazon SQS hochzuladen, da die Aktion `AddPermission` bei der Zugriffserteilung für eine Warteschlange keine Angabe einer IP-Adressbeschränkung erlaubt.

```
{
   "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"
         }
      }
   }]
}
```

Beispiel 4: Zugriff für ein bestimmtes Konto verweigern

Das folgende Beispiel einer Amazon SQS SQS-Richtlinie verweigert einen bestimmten AWS-Konto Zugriff auf Ihre Warteschlange. Dieses Beispiel baut auf dem Beispiel "Beispiel 1: Einem Konto eine Berechtigung erteilen" auf: Es verweigert den Zugriff auf die angegebene Datei. AWS-Konto Es ist erforderlich, diese Richtlinie zu erstellen und mit der Aktion `SetQueueAttributes` zu Amazon SQS hochzuladen, da die Aktion `AddPermission` keine Zugriffsverweigerung für eine Warteschlange erlaubt (sie lässt nur die Erteilung des Zugriffs auf eine Warteschlange zu).

```
{
   "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"
   }]
}
```

Beispiel 5: Verweigerung des Zugriffs, wenn dieser nicht von einem VPC-Endpunkt aus erfolgt

Das folgende Beispiel für eine Amazon-SQS-Richtlinie beschränkt den Zugriff auf `queue1`: 111122223333 kann die Aktionen `SendMessage` und `ReceiveMessage` nur von der VPC-Endpunkt-ID `vpce-1a2b3c4d` aus durchführen (angegeben mit der `aws:sourceVpce`-Bedingung). Weitere Informationen finden Sie unter Endpunkte von Amazon Virtual Private Cloud für Amazon SQS.

> **Note**
>
> - Die `aws:sourceVpce`-Bedingung benötigt keine ARN für die VPC-Endpunkt-Ressource, sondern nur die VPC-Endpunkt-ID.

- Sie können das folgende Beispiel so abändern, dass alle Aktionen auf einen spezifischen VPC-Endpunkt eingeschränkt werden, indem Sie in der zweiten Anweisung alle Amazon-SQS-Aktionen (`sqs:*`) verweigern. Durch eine solche Richtlinienanweisung würde jedoch festgelegt, dass alle Aktionen (einschließlich von zum Ändern von Warteschlangen-Berechtigungen erforderlichen administrativen Aktionen) über den spezifischen, in der Richtlinie definierten VPC-Endpunkt erfolgen müssen; hierdurch würde dem Benutzer in der Zukunft das Ändern von Warteschlangen-Berechtigungen unmöglich gemacht.

```
{
   "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"
            }
         }
      }
```

```
    ]
}
```

Verwenden von temporären Sicherheitsanmeldeinformationen mit Amazon SQS

Mit IAM können Sie nicht nur Benutzer mit eigenen Sicherheitsanmeldedaten erstellen, sondern auch jedem Benutzer temporäre Sicherheitsanmeldedaten zuweisen, sodass der Benutzer auf Ihre AWS Dienste und Ressourcen zugreifen kann. Sie können Benutzer verwalten, die AWS-Konten haben. Sie können auch Benutzer für Ihr System verwalten, die noch keine haben AWS-Konten (Verbundbenutzer). Darüber hinaus können Anwendungen, die Sie für den Zugriff auf Ihre AWS Ressourcen erstellen, auch als „Benutzer“ betrachtet werden.

Sie können diese temporären Sicherheitsanmeldeinformationen für das Erstellen von Anforderungen an Amazon SQS verwenden. Die API-Bibliotheken berechnen anhand dieser Anmeldeinformationen den notwendigen Signaturwert, um Ihre Anforderung zu authentifizieren. Wenn Sie beim Senden von Anfragen abgelaufene Anmeldeinformationen verwenden, lehnt Amazon SQS die Anfrage ab.

> **Note**
>
> Sie können keine Richtlinie auf der Basis von temporären Anmeldeinformationen festlegen.

Voraussetzungen

1. Verwenden Sie IAM zum Erstellen von temporären Sicherheitsanmeldeinformationen:
   - Sicherheits-Token
   - Access Key ID
   - Secret Access Key
2. Bereiten Sie die zu signierende Zeichenfolge mit der temporären Zugriffsschlüssel-ID und dem Sicherheits-Token vor.
3. Verwenden Sie den temporären geheimen Zugriffsschlüssel anstelle Ihres eigenen geheimen Zugriffsschlüssels, um Ihre Abfrage-API-Anforderung zu signieren.

> **Note**
>
> Wenn Sie die signierte Abfrage-API senden, verwenden Sie die temporäre Zugriffsschlüssel-ID anstelle Ihrer eigenen Zugriffsschlüssel-ID und schließen Sie das Sicherheits-Token ein.

Weitere Informationen zur IAM-Unterstützung für temporäre Sicherheitsanmeldedaten finden Sie unter Granting Temporary Access to Your AWS Resources im IAM-Benutzerhandbuch.

So rufen Sie eine Amazon-SQS-Abfrage-API-Aktion mit temporären Sicherheitsanmeldeinformationen auf

1. Fordern Sie ein temporäres Sicherheitstoken an mit AWS Identity and Access Management. Weitere Informationen finden Sie unter Erstellen temporärer Sicherheitsanmeldeinformationen für IAM-Benutzer im IAM-Benutzerhandbuch.

   IAM gibt ein Sicherheits-Token, eine Zugriffsschlüssel-ID und einen geheimen Zugriffsschlüssel zurück.
2. Bereiten Sie Ihre Abfrage mit der temporären Zugriffsschlüssel-ID anstelle Ihrer eigenen Zugriffsschlüssel-ID vor und schließen Sie das Sicherheits-Token ein. Signieren Sie Ihre Anforderung mit dem temporären geheimen Zugriffsschlüssel anstelle Ihres eigenen Zugriffsschlüssels.
3. Senden Sie die signierte Abfragezeichenfolge mit der temporären Zugriffsschlüssel-ID und dem Sicherheits-Token.

   Das folgende Beispiel zeigt, wie Sie temporäre Sicherheitsanmeldeinformationen zum Authentifizieren einer Amazon-SQS-Anforderung verwenden. Die Struktur von *AUTHPARAMS* hängt davon ab, wie Sie Ihre API-Anforderung signieren. Weitere Informationen finden Sie unter AWS API-Anfragen signieren in der Allgemeinen Referenz zu 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=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   &AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
   &Version=2012-11-05
   &AUTHPARAMS
   ```

   Das folgende Beispiel verwendet temporäre Sicherheitsanmeldeinformationen, um zwei Nachrichten mit der `SendMessageBatch`-Aktion zu senden.

```
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
```

Zugriffsverwaltung für verschlüsselte Amazon SQS SQS-Warteschlangen mit Richtlinien für geringste Rechte

Sie können Amazon SQS verwenden, um vertrauliche Daten zwischen Anwendungen auszutauschen, indem Sie serverseitige Verschlüsselung (SSE) verwenden, die in AWS Key Management Service (KMS) integriert ist. Mit der Integration von Amazon SQS und können Sie die Schlüssel AWS KMS, die Amazon SQS schützen, sowie die Schlüssel, die Ihre anderen AWS Ressourcen schützen, zentral verwalten.

Mehrere AWS Dienste können als Ereignisquellen dienen, die Ereignisse an Amazon SQS senden. Um einer Ereignisquelle den Zugriff auf die verschlüsselte Amazon SQS SQS-Warteschlange zu ermöglichen, müssen Sie die Warteschlange mit einem vom Kunden verwalteten Schlüssel AWS KMS konfigurieren. Verwenden Sie dann die Schlüsselrichtlinie, damit der Service die erforderlichen AWS KMS API-Methoden verwenden kann. Der Service benötigt außerdem Berechtigungen zur Authentifizierung des Zugriffs, damit die Warteschlange Ereignisse senden kann. Sie können dies erreichen, indem Sie eine Amazon-SQS-Richtlinie verwenden. Dabei handelt es sich um eine ressourcenbasierte Richtlinie, mit der Sie den Zugriff auf die Amazon-SQS-Warteschlange und ihre Daten kontrollieren können.

Die folgenden Abschnitte enthalten Informationen darüber, wie Sie den Zugriff auf Ihre verschlüsselte Amazon SQS SQS-Warteschlange mithilfe der Amazon SQS SQS-Richtlinie und der AWS KMS Schlüsselrichtlinie kontrollieren können. Die Richtlinien in diesem Handbuch helfen Ihnen dabei, die geringsten Berechtigungen zu erlangen.

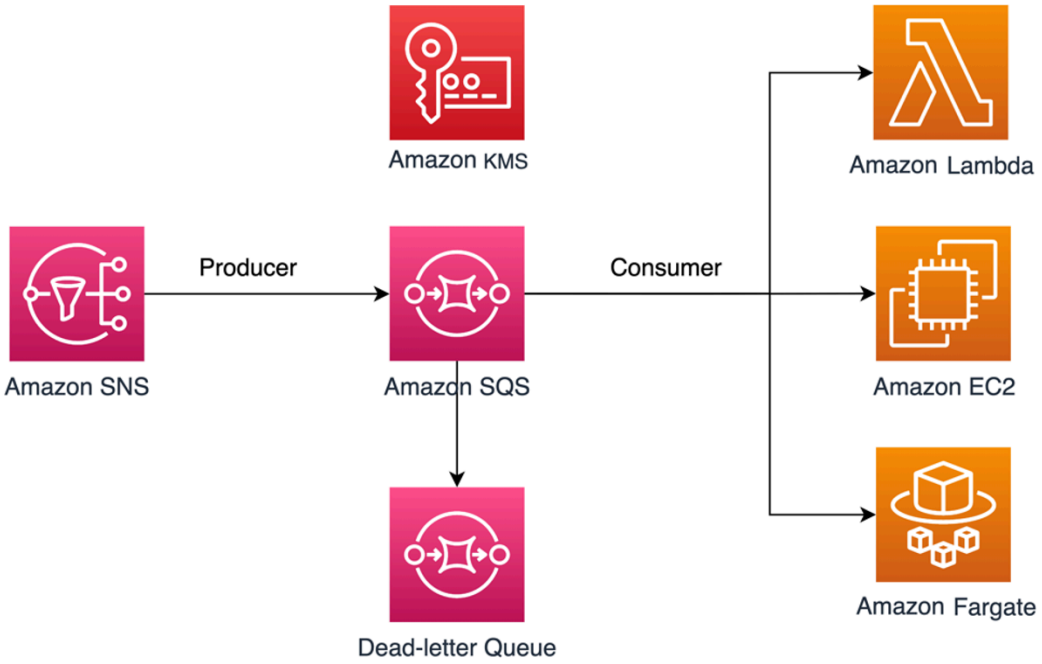
In diesem Leitfaden wird auch beschrieben, wie ressourcenbasierte Richtlinien das Confused-Deputy-Problem mithilfe der globalen IAM-Bedingungskontextschlüssel `aws:SourceArn`, `aws:SourceAccount` und `aws:PrincipalOrgID` lösen.

Themen

- Übersicht
- Schlüsselrichtlinie mit den geringsten Berechtigungen für Amazon SQS
- Amazon-SQS-Richtlinienerklärungen für die Warteschlange für unzustellbare Nachrichten
- Vermeidung des serviceübergreifenden Confused-Deputy-Problems
- Verwenden von IAM Access Analyzer, um den kontoübergreifenden Zugriff zu überprüfen

Übersicht

In diesem Thema werden wir Sie durch einen häufigen Anwendungsfall führen, um zu veranschaulichen, wie Sie die Schlüsselrichtlinie und die Amazon-SQS-Warteschlangenrichtlinie erstellen können. Dieser Anwendungsfall wird im folgenden Bild veranschaulicht.



In diesem Beispiel ist der Nachrichtenproduzent ein Amazon Simple Notification Service (SNS)-Thema, das so konfiguriert ist, dass es per Fanout Nachrichten in Ihre verschlüsselte Amazon-SQS-Warteschlange überträgt. Der Nachrichtenkonsument ist ein Computing-Service, z. B. eine AWS Lambda-Funktion, eine Amazon Elastic Compute Cloud (EC2)-Instance oder ein AWS Fargate-Container. Ihre Amazon-SQS-Warteschlange ist dann so konfiguriert, dass

fehlgeschlagene Nachrichten an eine Warteschlange für unzustellbare Nachrichten (DLQ) gesendet werden. Dies ist nützlich für das Debuggen Ihrer Anwendung oder Ihres Messaging-Systems, da Sie mit DLQs nicht konsumierte Nachrichten isolieren können, um festzustellen, warum ihre Verarbeitung nicht erfolgreich war. In der in diesem Thema definierten Lösung wird ein Computing-Service wie eine Lambda-Funktion verwendet, um Nachrichten zu verarbeiten, die in der Amazon-SQS-Warteschlange gespeichert sind. Wenn sich der Nachrichtenverbraucher in einer Virtual Private Cloud (VPC) befindet, können Sie mit der in diesem Handbuch enthaltenen `DenyReceivingIfNotThroughVPCE`-Richtlinienanweisung den Nachrichtenempfang auf diese spezifische VPC beschränken.

> **Note**
>
> Dieses Handbuch enthält nur die erforderlichen IAM-Berechtigungen in Form von Richtlinienanweisungen. Um die Richtlinie zu erstellen, müssen Sie die Anweisungen zu Ihrer Amazon SQS SQS-Richtlinie oder Ihrer AWS KMS wichtigsten Richtlinie hinzufügen. Dieses Handbuch enthält keine Anweisungen zum Erstellen der Amazon SQS SQS-Warteschlange oder des AWS KMS Schlüssels. Anweisungen zum Erstellen dieser Ressourcen finden Sie unter Erstellen einer Amazon-SQS-Warteschlange und Erstellen von Schlüsseln.
> Die in diesem Handbuch definierte Amazon-SQS-Richtlinie unterstützt nicht die direkte Weiterleitung von Nachrichten an dieselbe oder eine andere Amazon-SQS-Warteschlange.

Schlüsselrichtlinie mit den geringsten Berechtigungen für Amazon SQS

In diesem Abschnitt beschreiben wir die erforderlichen Berechtigungen mit den geringsten Rechten AWS KMS für den vom Kunden verwalteten Schlüssel, den Sie zum Verschlüsseln Ihrer Amazon SQS SQS-Warteschlange verwenden. Mit diesen Genehmigungen können Sie den Zugriff nur auf die vorgesehenen Entitäten beschränken und gleichzeitig die geringsten Berechtigungen implementieren. Die Schlüsselrichtlinie muss aus den folgenden Richtlinienanweisungen bestehen, die wir im Folgenden ausführlich beschreiben:

- Erteilen Sie Administratorrechte für den Schlüssel AWS KMS
- Gewährt Lesezugriff auf die wichtigsten Metadaten
- Gewährt Amazon SNS KMS-Berechtigungen, um Nachrichten für die Warteschlange zu veröffentlichen
- Konsumenten gestatten, Nachrichten aus der Warteschlange zu entschlüsseln

Erteilen Sie Administratorrechte für den Schlüssel AWS KMS

Um einen AWS KMS Schlüssel zu erstellen, müssen Sie AWS KMS Administratorberechtigungen für die IAM-Rolle bereitstellen, die Sie für die Bereitstellung des AWS KMS Schlüssels verwenden. Diese Administratorberechtigungen sind in der folgenden `AllowKeyAdminPermissions`-Richtlinienanweisung definiert. Wenn Sie diese Aussage zu Ihrer AWS KMS Schlüsselrichtlinie hinzufügen, achten Sie darauf, sie durch **<admin-role ARN>**den Amazon-Ressourcennamen (ARN) der IAM-Rolle zu ersetzen, die für die Bereitstellung des AWS KMS Schlüssels, die Verwaltung des AWS KMS Schlüssels oder beides verwendet wurde. Dies kann die IAM-Rolle Ihrer Bereitstellungs-Pipeline oder die Administratorrolle für Ihre Organisation in Ihren AWS -Organisationen sein.

```
{
  "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**
>
> In einer AWS KMS Schlüsselrichtlinie muss der Wert des `Resource` Elements sein`*`, was „dieser AWS KMS Schlüssel“ bedeutet. Das Sternchen (`*`) kennzeichnet den AWS KMS Schlüssel, dem die Schlüsselrichtlinie zugeordnet ist.

Gewährt Lesezugriff auf die wichtigsten Metadaten

Um anderen IAM-Rollen schreibgeschützten Zugriff auf Ihre wichtigsten Metadaten zu gewähren, fügen Sie die `AllowReadAccessToKeyMetaData`-Anweisung zu Ihrer Schlüsselrichtlinie hinzu. Mit der folgenden Anweisung können Sie beispielsweise alle AWS KMS Schlüssel in Ihrem Konto zu Prüfungszwecken auflisten. Diese Anweisung gewährt dem AWS Root-Benutzer nur Lesezugriff auf die Schlüsselmetadaten. Daher kann jeder IAM-Prinzipal in dem Konto auf die Schlüsselmetadaten zugreifen, wenn seine identitätsbasierten Richtlinien über die in der folgenden Anweisung aufgeführten Berechtigungen verfügen: `kms:Describe*`, `kms:Get*` und `kms:List*`. Ersetzen Sie *`<account-ID>`* durch Ihre eigenen Informationen.

```
{
  "Sid": "AllowReadAcesssToKeyMetaData",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::<accountID>:root"
    ]
  },
  "Action": [
    "kms:Describe*",
    "kms:Get*",
    "kms:List*"
  ],
  "Resource": "*"
}
```

Gewährt Amazon SNS KMS-Berechtigungen, um Nachrichten für die Warteschlange zu veröffentlichen

Damit Ihr Amazon-SNS-Thema Nachrichten in Ihrer verschlüsselten Amazon-SQS-Warteschlange veröffentlichen kann, fügen Sie die `AllowSNSToSendToSQS`-Richtlinienanweisung zu Ihrer Schlüsselrichtlinie hinzu. Diese Erklärung erteilt Amazon SNS die Erlaubnis, den AWS KMS

Schlüssel zur Veröffentlichung in Ihrer Amazon SQS SQS-Warteschlange zu verwenden. Ersetzen Sie *<account-ID>* durch Ihre eigenen Informationen.

> **Note**
>
> Die `Condition` in der Erklärung angegebene Einschränkung beschränkt den Zugriff nur auf den Amazon SNS SNS-Service auf demselben AWS Konto.

```
{
  "Sid": "AllowSNSToSendToSQS",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "sns.amazonaws.com"
    ]
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "<account-id>"
    }
  }
}
```

Konsumenten gestatten, Nachrichten aus der Warteschlange zu entschlüsseln

Die folgende `AllowConsumersToReceiveFromTheQueue`-Anweisung gewährt dem Amazon-SQS-Nachrichtenkonsumenten die erforderlichen Berechtigungen zum Entschlüsseln von Nachrichten, die aus der verschlüsselten Amazon-SQS-Warteschlange empfangen wurden. Wenn Sie die Richtlinienanweisung anfügen, ersetzen Sie *<consumer's runtime role ARN>* durch den ARN der IAM-Laufzeitrolle des Nachrichtenkonsumenten.

```
{
  "Sid": "AllowConsumersToReceiveFromTheQueue",
  "Effect": "Allow",
  "Principal": {
```

```
    "AWS": [
      "<consumer's execution role ARN>"
    ]
  },
  "Action": [
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

Richtlinie für Amazon SQS mit den geringsten Berechtigungen

Dieser Abschnitt führt Sie durch die Amazon-SQS-Warteschlangenrichtlinien mit den geringsten Berechtigungen für den in diesem Handbuch behandelten Anwendungsfall (z. B. Amazon SNS zu Amazon SQS). Die definierte Richtlinie soll unbeabsichtigten Zugriff verhindern, indem eine Kombination aus den beiden Anweisungen `Deny` und `Allow` verwendet wird. Die `Allow`-Anweisungen gewähren Zugriff auf die intendierte(n) Entität(en). Die `Deny`-Anweisungen verhindern, dass andere unbeabsichtigte Entitäten auf die Amazon-SQS-Warteschlange zugreifen, während die beabsichtigte Entität innerhalb der Richtlinienbedingung ausgeschlossen wird.

Die Amazon-SQS-Richtlinie umfasst die folgenden Anweisungen, die wir im Folgenden ausführlich beschreiben:

- Beschränken der Amazon-SQS-Verwaltungsberechtigungen
- Beschränken der Amazon-SQS-Warteschlangenaktionen der angegebenen Organisation
- Erteilen von Amazon-SQS-Berechtigungen für Konsumenten
- Erzwingen der Verschlüsselung von Daten während der Übertragung
- Einschränkung der Nachrichtenübertragung auf ein bestimmtes Amazon-SNS-Thema
- (Optional) Einschränken des Nachrichtenempfangs auf einen bestimmten VPC-Endpunkt

Beschränken der Amazon-SQS-Verwaltungsberechtigungen

Die folgende `RestrictAdminQueueActions`-Richtlinienanweisung beschränkt die Amazon-SQS-Verwaltungsberechtigungen nur auf die IAM-Rolle(n), die Sie für die Bereitstellung der Warteschlange, die Verwaltung der Warteschlange oder für beide verwenden. Stellen Sie sicher, dass Sie die *`<placeholder values>`* durch Ihre eigenen Informationen ersetzen. Geben Sie den ARN der IAM-Rolle an, die für die Bereitstellung der Amazon-SQS-Warteschlange verwendet

wurde, sowie die ARNs aller Administratorrollen, die über Amazon-SQS-Verwaltungsberechtigungen verfügen sollten.

```
{
  "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>"
      ]
    }
  }
}
```

Beschränken der Amazon-SQS-Warteschlangenaktionen der angegebenen Organisation

Verwenden Sie die folgende Anweisungen, um Ihre Amazon-SQS-Ressourcen vor externem Zugriff (Zugriff durch eine Entität außerhalb Ihrer AWS -Organisation) zu schützen. Diese Anweisung beschränkt den Zugriff auf die Amazon-SQS-Warteschlange auf die Organisation, die Sie in der `Condition` angeben. Stellen Sie sicher, den *<SQS queue ARN>* durch den ARN der IAM-Rolle zu ersetzen, die für die Bereitstellung der Amazon-SQS-Warteschlange verwendet wurde, sowie die *<org-id>* durch die ID Ihrer Organisation.

```
{
  "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>"
      ]
    }
  }
}
```

Erteilen von Amazon-SQS-Berechtigungen für Konsumenten

Um Nachrichten aus der Amazon-SQS-Warteschlange zu empfangen, müssen Sie dem Nachrichtenverbraucher die erforderlichen Berechtigungen erteilen. Die folgende Richtlinienanweisung gewährt dem von Ihnen angegebenen Konsumenten die erforderlichen Berechtigungen, um Nachrichten aus der Amazon-SQS-Warteschlange zu konsumieren. Achten Sie beim Hinzufügen der Anweisung zu Ihrer Amazon-SQS-Richtlinie darauf, *<consumer's IAM runtime role ARN>* durch den ARN der vom Konsumenten verwendeten IAM-Laufzeitrolle und *<SQS queue ARN>* durch den ARN der IAM-Rolle zu ersetzen, die für die Bereitstellung der Amazon-SQS-Warteschlange verwendet wurde.

```
{
  "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>"
}
```

Um zu verhindern, dass andere Entitäten Nachrichten aus der Amazon-SQS-Warteschlange erhalten, fügen Sie die `DenyOtherConsumersFromReceiving`-Anweisung zu der Amazon-SQS-Warteschlangenrichtlinie hinzu. Diese Anweisung beschränkt den Nachrichtenverbrauch auf den von Ihnen angegebenen Konsumenten, so dass keine anderen Konsumenten Zugriff haben, selbst wenn ihnen ihre Identitätsberechtigungen Zugriff gewähren würden. Stellen Sie sicher, dass Sie *<SQS queue ARN>* und *<consumer's runtime role ARN>* durch Ihre eigenen Informationen ersetzen.

```
{
  "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>"
    }
  }
}
```

Erzwingen der Verschlüsselung von Daten während der Übertragung

Die folgende `DenyUnsecureTransport`-Richtlinienanweisung verpflichtet die Konsumenten und Produzenten, sichere Kanäle (TLS-Verbindungen) zu verwenden, um Nachrichten aus der Amazon-SQS-Warteschlange zu senden und zu empfangen. Stellen Sie sicher, dass Sie *<SQS queue ARN>* durch den ARN der IAM-Rolle ersetzen, die für die Bereitstellung der Amazon-SQS-Warteschlange verwendet wurde.

```
{
  "Sid": "DenyUnsecureTransport",
```

```
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:ReceiveMessage",
    "sqs:SendMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "Bool": {
      "aws:SecureTransport": "false"
    }
  }
}
```

Einschränkung der Nachrichtenübertragung auf ein bestimmtes Amazon-SNS-Thema

Die folgende `AllowSNSToSendToTheQueue`-Richtlinienanweisung ermöglicht dem angegebenen Amazon-SNS-Thema, Nachrichten an die Amazon-SQS-Warteschlange zu senden. Achten Sie darauf, *<SQS queue ARN>* durch den ARN der IAM-Rolle zu ersetzen, die für die Bereitstellung der Amazon-SQS-Warteschlange verwendet wurde, sowie *<SNS topic ARN>* durch den ARN des Amazon-SNS-Themas.

```
{
  "Sid": "AllowSNSToSendToTheQueue",
  "Effect": "Allow",
  "Principal": {
    "Service": "sns.amazonaws.com"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn": "<SNS topic ARN>"
    }
  }
}
```

Die folgende `DenyAllProducersExceptSNSFromSending`-Richtlinienanweisung verhindert, dass andere Produzenten Nachrichten an die Warteschlange senden. Ersetzen Sie *<SQS queue ARN>* und *<SNS topic ARN>* durch Ihre eigenen Informationen.

```
{
  "Sid": "DenyAllProducersExceptSNSFromSending",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "ArnNotLike": {
      "aws:SourceArn": "<SNS topic ARN>"
    }
  }
}
```

(Optional) Einschränken des Nachrichtenempfangs auf einen bestimmten VPC-Endpunkt

Um den Empfang von Nachrichten nur auf einen bestimmten VPC-Endpunkt zu beschränken, fügen Sie Ihrer Amazon-SQS-Warteschlangenrichtlinie die folgende Richtlinienanweisung hinzu. Diese Anweisung verhindert, dass ein Nachrichtenkonsument Nachrichten aus der Warteschlange empfängt, es sei denn, die Nachrichten stammen vom gewünschten VPC-Endpunkt. Ersetzen Sie *<SQS queue ARN>* durch den ARN der IAM-Rolle, die für die Bereitstellung der Amazon-SQS-Warteschlange verwendet wurde, und *<vpce_id>* durch die ID des VPC-Endpunkts.

```
{
  "Sid": "DenyReceivingIfNotThroughVPCE",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "StringNotEquals": {
      "aws:sourceVpce": "<vpce id>"
    }
```

```
  }
}
```

Amazon-SQS-Richtlinienerklärungen für die Warteschlange für unzustellbare Nachrichten

Fügen Sie Ihrer DLQ-Zugriffsrichtlinie die folgenden Richtlinienerklärungen hinzu, die anhand ihrer Anweisungs-ID gekennzeichnet sind:

- `RestrictAdminQueueActions`
- `DenyQueueActionsOutsideOrg`
- `AllowConsumersToReceiveFromTheQueue`
- `DenyOtherConsumersFromReceiving`
- `DenyUnsecureTransport`

Zusätzlich zum Hinzufügen der obigen Richtlinienanweisungen zu Ihrer DLQ-Zugriffsrichtlinie sollten Sie auch eine Anweisung hinzufügen, um die Nachrichtenübertragung an Amazon-SQS-Warteschlangen einzuschränken, wie im folgenden Abschnitt beschrieben.

Einschränken der Nachrichtenübertragung auf Amazon-SQS-Warteschlangen

Um den Zugriff nur auf Amazon-SQS-Warteschlangen von demselben Konto aus zu beschränken, fügen Sie der DLQ-Warteschlangenrichtlinie die folgende `DenyAnyProducersExceptSQS`-Richtlinienanweisung hinzu. Diese Anweisung beschränkt die Nachrichtenübertragung nicht auf eine bestimmte Warteschlange, da Sie die DLQ bereitstellen müssen, bevor Sie die Hauptwarteschlange erstellen, so dass Sie den Amazon-SQS-ARN nicht kennen, wenn Sie die DLQ erstellen. Wenn Sie den Zugriff auf nur eine Amazon-SQS-Warteschlange beschränken möchten, ändern Sie den `aws:SourceArn` in der `Condition` zu dem ARN Ihrer Amazon-SQS-Quellwarteschlange, wenn Sie diesen kennen.

```
{
  "Sid": "DenyAnyProducersExceptSQS",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS DLQ ARN>",
  "Condition": {
    "ArnNotLike": {
```

```
      "aws:SourceArn": "arn:aws:sqs:<region>:<account-id>:*"
    }
  }
}
```

> **Important**
>
> Die in diesem Handbuch definierten Amazon-SQS-Warteschlangenrichtlinien beschränken die `sqs:PurgeQueue`-Aktion nicht auf eine oder mehrere bestimmte IAM-Rolle(n). Die `sqs:PurgeQueue`-Aktion ermöglicht Ihnen, alle Nachrichten in der Amazon-SQS-Warteschlange zu löschen. Sie können diese Aktion auch verwenden, um Änderungen am Nachrichtenformat vorzunehmen, ohne die Amazon-SQS-Warteschlange zu ersetzen. Beim Debuggen einer Anwendung können Sie die Amazon-SQS-Warteschlange leeren, um potenziell fehlerhafte Nachrichten zu entfernen. Beim Testen der Anwendung können Sie ein hohes Nachrichtenvolumen durch die Amazon-SQS-Warteschlange leiten und dann die Warteschlange leeren, um neu zu beginnen, bevor Sie mit der Produktion beginnen. Der Grund dafür, dass diese Aktion nicht auf eine bestimmte Rolle beschränkt wird, liegt darin, dass diese Rolle bei der Bereitstellung der Amazon-SQS-Warteschlange möglicherweise nicht bekannt ist. Sie müssen diese Berechtigung zur identitätsbasierten Richtlinie der Rolle hinzufügen, um die Warteschlange löschen zu können.

Vermeidung des serviceübergreifenden Confused-Deputy-Problems

Das Confused-Deputy-Problem ist ein Sicherheitsproblem, bei dem eine Entität, die nicht über die Berechtigung zum Ausführen einer Aktion verfügt, eine Entität mit größeren Rechten zwingen kann, die Aktion auszuführen. Um dies zu verhindern, AWS stellt Tools bereit, mit denen Sie Ihr Konto schützen können, wenn Sie Dritten (auch als kontoübergreifender Dienst bezeichnet) oder anderen AWS Diensten (bekannt als dienstübergreifender Zugriff) Zugriff auf Ressourcen in Ihrem Konto gewähren. Die Richtlinienanweisungen in diesem Abschnitt können Ihnen helfen, das serviceübergreifende Confused-Deputy-Problem zu vermeiden.

Ein serviceübergreifender Identitätswechsel kann auftreten, wenn ein Service (der Anruf-Service) einen anderen Service anruft (den aufgerufenen Service). Der Aufruf-Service kann so manipuliert werden, dass er seine Berechtigungen verwendet, um auf die Ressourcen eines anderen Kunden zu reagieren, auf die er sonst nicht zugreifen dürfte. Um sich vor diesem Problem zu schützen, verwenden die in diesem Beitrag definierten ressourcenbasierten Richtlinien die globalen IAM-Bedingungsschlüssel `aws:SourceArn`, `aws:SourceAccount` und `aws:PrincipalOrgID`.

Dadurch werden die Berechtigungen eines Dienstes auf eine bestimmte Ressource, ein bestimmtes Konto oder eine bestimmte Organisation in AWS Organizations beschränkt.

Verwenden von IAM Access Analyzer, um den kontoübergreifenden Zugriff zu überprüfen

Sie können AWS IAM Access Analyzer verwenden, um Ihre Amazon SQS SQS-Warteschlangenrichtlinien und AWS KMS Schlüsselrichtlinien zu überprüfen und Sie zu benachrichtigen, wenn eine Amazon SQS SQS-Warteschlange oder ein AWS KMS Schlüssel Zugriff auf eine externe Entität gewährt. IAM Access Analyzer hilft bei der Identifizierung von Ressourcen in Ihrer Organisation und Ihren Konten, die außerhalb Ihrer Vertrauenszone weitergegeben werden. Bei dieser Vertrauenszone kann es sich um ein AWS Konto oder die Organisation innerhalb von AWS Organizations handeln, die Sie bei der Aktivierung von IAM Access Analyzer angeben.

IAM Access Analyzer identifiziert Ressourcen, die mit externen Prinzipalen gemeinsam genutzt werden, indem er die ressourcenbasierten Richtlinien in Ihrer Umgebung anhand von logischer Argumentation analysiert. AWS Für jede Instance einer Ressource, die außerhalb Ihrer Zone gemeinsam genutzt wird, erstellt Access Analyzer eine Erkenntnis. Die Ergebnisse enthalten Informationen über den Zugang und den externen Prinzipal, dem dieser gewährt wurde. Sie können die Ergebnisse prüfen, um festzustellen, ob der Zugriff beabsichtigt und sicher oder ob er unbeabsichtigt ist und ein Sicherheitsrisiko darstellt. Überprüfen Sie bei unbeabsichtigtem Zugriff die betroffene Richtlinie und korrigieren Sie diese. In diesem Blogbeitrag finden Sie weitere Informationen darüber, wie AWS IAM Access Analyzer unbeabsichtigte Zugriffe auf Ihre Ressourcen erkennt. AWS

Weitere Informationen zu AWS IAM Access Analyzer finden Sie in der AWS IAM Access Analyzer-Dokumentation.

Amazon-SQS-API-Berechtigungen: Referenz für Aktionen und Ressourcen

Wenn Sie Zugriffskontrolle einrichten und Berechtigungsrichtlinien erstellen, die Sie einer IAM-Identität anfügen können, verwenden Sie die folgende Tabelle als Referenz. Die enthält jede Amazon Simple Queue Service-Aktion, die entsprechenden Aktionen, für die Sie Berechtigungen zur Ausführung der Aktion erteilen können, und die AWS Ressource, für die Sie die Berechtigungen erteilen können.

Die Aktionen geben Sie im Feld `Action` und den Wert für die Ressource im Feld `Resource` der Richtlinie an. Um eine Aktion anzugeben, verwenden Sie das Präfix `sqs:` gefolgt vom Namen der Aktion (z. B. `sqs:CreateQueue`).

Derzeit unterstützt Amazon SQS die in IAM verfügbaren globalen Bedingungskontextschlüssel.

Amazon-Simple-Queue-Service-API und erforderliche Berechtigungen für Aktionen

AddPermission

Aktion(en): `sqs:AddPermission`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

ChangeMessageSichtbarkeit

Aktion(en): `sqs:ChangeMessageVisibility`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

ChangeMessageVisibilityBatch

Aktion(en): `sqs:ChangeMessageVisibilityBatch`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

CreateQueue

Aktion(en): `sqs:CreateQueue`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

DeleteMessage

Aktion(en): `sqs:DeleteMessage`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

DeleteMessageBatch

Aktion(en): `sqs:DeleteMessageBatch`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

DeleteQueue

Aktion(en): `sqs:DeleteQueue`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

GetQueueAttribute

Aktion(en): `sqs:GetQueueAttributes`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

GetQueueUrl

Aktion(en): `sqs:GetQueueUrl`

Ressource: `arn:aws:sqs:`*`region`*`:`*`account_id`*`:`*`queue_name`*

ListDeadLetterSourceWarteschlangen

Aktion(en): `sqs:ListDeadLetterSourceQueues`

Ressource: `arn:aws:sqs:`*`region`*`:`*`account_id`*`:`*`queue_name`*

ListQueues

Aktion(en): `sqs:ListQueues`

Ressource: `arn:aws:sqs:`*`region`*`:`*`account_id`*`:`*`queue_name`*

ListQueueSchlagworte

Aktion(en): `sqs:ListQueueTags`

Ressource: `arn:aws:sqs:`*`region`*`:`*`account_id`*`:`*`queue_name`*

PurgeQueue

Aktion(en): `sqs:PurgeQueue`

Ressource: `arn:aws:sqs:`*`region`*`:`*`account_id`*`:`*`queue_name`*

ReceiveMessage

Aktion(en): `sqs:ReceiveMessage`

Ressource: `arn:aws:sqs:`*`region`*`:`*`account_id`*`:`*`queue_name`*

RemovePermission

Aktion(en): `sqs:RemovePermission`

Ressource: `arn:aws:sqs:`*`region`*`:`*`account_id`*`:`*`queue_name`*

SendMessageund SendMessageBatch

Aktion(en): `sqs:SendMessage`

Ressource: `arn:aws:sqs:`*`region`*`:`*`account_id`*`:`*`queue_name`*

SetQueueAttribute

Aktion(en): `sqs:SetQueueAttributes`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

TagQueue

Aktion(en): `sqs:TagQueue`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

UntagQueue

Aktion(en): `sqs:UntagQueue`

Ressource: `arn:aws:sqs:region:account_id:queue_name`

Protokollierung und Überwachung in Amazon SQS

Dieser Abschnitt enthält Informationen zu den Protokollierungs- und Überwachungsoptionen für Amazon SQS, einschließlich Informationen zur Erfassung von API-Aufrufen sowie CloudTrail zu CloudWatch Metriken, um Einblicke in die Warteschlangenaktivität und -leistung zu gewinnen.

Themen

- Protokollieren von Amazon-SQS-API-Aufrufen mit AWS CloudTrail
- Überwachen Amazon SQS SQS-Warteschlangen mit CloudWatch

Protokollieren von Amazon-SQS-API-Aufrufen mit AWS CloudTrail

Amazon SQS ist integriert AWS CloudTrail , um die Amazon SQS SQS-Aufrufe von einem Benutzer, einer Rolle oder AWS einem Service aufzuzeichnen. CloudTrail erfasst API-Aufrufe im Zusammenhang mit Amazon SQS SQS-Standard- und FIFO-Warteschlangen als Ereignisse, einschließlich Interaktionen, die über die Amazon SQS SQS-Konsole sowie programmgesteuert über Aufrufe der Amazon SQS SQS-APIs initiiert wurden.

Themen

- Amazon SQS SQS-Informationen in CloudTrail
- Verwaltungsereignisse in CloudTrail
- Datenereignisse in CloudTrail
- Beispiele: CloudTrail Verwaltungsereignisse für Amazon SQS
- Beispiele: CloudTrail Datenereignisse für Amazon SQS

Amazon SQS SQS-Informationen in CloudTrail

CloudTrail ist standardmäßig aktiviert, wenn Sie Ihr Konto erstellen. AWS Wenn eine unterstützte Amazon SQS SQS-Ereignisaktivität auftritt, wird sie zusammen mit anderen CloudTrail AWS Serviceereignissen in der Ereignishistorie als Ereignis aufgezeichnet. Sie können aktuelle Ereignisse für Ihr AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Amazon SQS SQS-APIs, die Warteschlangenverwaltungsvorgänge aufrufen, `AddPermission` werden beispielsweise als Verwaltungsereignisse kategorisiert und sind CloudTrail standardmäßig angemeldet. Amazon SQS SQS-APIs, bei denen es sich um umfangreiche Operationen handelt, die in einer Amazon SQS SQS-Warteschlange ausgeführt `SendMessage` werden, z. B. als Datenereignisse kategorisiert und protokolliert werden, nachdem Sie sich für angemeldet haben. CloudTrail

Anhand der CloudTrail gesammelten Informationen können Sie eine bestimmte Anfrage an eine Amazon SQS SQS-API, die IP-Adresse oder Identität des Anfragenden sowie Datum und Uhrzeit der Anfrage identifizieren. Wenn Sie einen CloudTrail Trail konfigurieren, können Sie kontinuierlich CloudTrail Ereignisse an einen Amazon S3 S3-Bucket mit einer optionalen Übermittlung an Amazon CloudWatch Logs und senden AWS EventBridge. Wenn Sie keinen Trail konfigurieren, können Sie den Ereignisverlauf der Verwaltungsereignisse nur unter Ereignisse in der CloudTrail Konsole einsehen. Weitere Informationen finden Sie unter Übersicht zum Erstellen eines Trails im AWS CloudTrail Benutzerhandbuch.

Verwaltungsereignisse in CloudTrail

Amazon SQS protokolliert die folgenden API-Aktionen als Verwaltungsereignisse:

- `AddPermission`
- `CreateQueue`
- `CancelMessageMoveTask`
- `DeleteQueue`
- `ListMessageMoveTasks`
- `PurgeQueue`
- `RemovePermission`
- `SetQueueAttributes`
- `StartMessageMoveTask`

- TagQueue
- UntagQueue

Die folgenden Amazon SQS SQS-APIs werden für die CloudTrail Protokollierung nicht unterstützt:

- GetQueueAttributes
- GetQueueUrl
- ListDeadLetterSourceQueues
- ListQueueTags
- ListQueues

Datenereignisse in CloudTrail

Datenereignisse liefern Informationen über die Ressourcenoperationen, die auf oder in einer Ressource ausgeführt werden, z. B. das Senden oder Empfangen einer Amazon-SQS-Nachricht an und aus einer Amazon-SQS-Warteschlange. Datenereignisse sind umfangreiche Aktivitäten, die standardmäßig CloudTrail nicht protokolliert werden. Sie können die API-Aktionsprotokollierung für Datenereignisse für Ihre SQS-Warteschlange mithilfe CloudTrail von APIs aktivieren. Weitere Informationen finden Sie unter Protokollieren von Datenereignissen im Benutzerhandbuch für AWS CloudTrail .

Mit CloudTrail können Sie mithilfe erweiterter Event-Selektoren entscheiden, welche Amazon SQS SQS-API-Aktivitäten protokolliert und aufgezeichnet werden. Wenn Sie Amazon-SQS-Datenereignisse protokollieren möchten, müssen Sie den Ressourcentyp `AWS::SQS::Queue` angeben. Sobald dies festgelegt ist, können Sie Ihre Protokollierungseinstellungen weiter verfeinern, indem Sie bestimmte Datenereignisse für die Aufzeichnung auswählen, z. B. die Verwendung des Filters `eventName` zum Nachverfolgen von `SendMessage`-Ereignissen. Weitere Informationen finden Sie unter `AdvancedEventSelector` in der AWS CloudTrail -API-Referenz.

Amazon-SQS-Datenereignisse:

- SendMessage
- SendMessageBatch
- ReceiveMessage
- DeleteMessage
- DeleteMessageBatch

- ChangeMessageVisibility
- ChangeMessageVisibilityBatch

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen finden Sie unter AWS CloudTrail-Preisgestaltung.

Beispiele: CloudTrail Verwaltungsereignisse für Amazon SQS

Die folgenden Beispiele zeigen CloudTrail Protokolleinträge für unterstützte APIs:

AddPermission

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für einen `AddPermission` API-Aufruf.


```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Alice",
         "accountId": "123456789012",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "userName": "Alice"
       },
       "eventTime": "2018-06-28T22:23:46Z",
       "eventSource": "sqs.amazonaws.com",
       "eventName": "AddPermission",
       "awsRegion": "us-east-2",
       "sourceIPAddress": "203.0.113.0",
       "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
       "requestParameters": {
         "actions": [
           "SendMessage"
         ],
         "AWSAccountIds": [
           "123456789012"
         ],
         "label": "MyLabel",
```

```
        "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
      },
      "responseElements": null,
      "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
      "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
    }
  ]
}
```

CreateQueue

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für einen `CreateQueue` API-Aufruf.

```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Alejandro",
         "accountId": "123456789012",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "userName": "Alejandro"
       },
       "eventTime": "2018-06-28T22:23:46Z",
       "eventSource": "sqs.amazonaws.com",
       "eventName": "CreateQueue",
       "awsRegion": "us-east-2",
       "sourceIPAddress": "203.0.113.1",
       "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
       "requestParameters": {
         "queueName": "MyQueue"
       },
       "responseElements": {
         "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
       },
       "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
       "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
     }
   ]
}
```

DeleteQueue

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für einen `DeleteQueue` API-Aufruf.


```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Carlos",
         "accountId": "123456789012",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "userName": "Carlos"
       },
       "eventTime": "2018-06-28T22:23:46Z",
       "eventSource": "sqs.amazonaws.com",
       "eventName": "DeleteQueue",
       "awsRegion": "us-east-2",
       "sourceIPAddress": "203.0.113.2",
       "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
       "requestParameters": {
         "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
       },
       "responseElements": null,
       "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
       "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
     }
   ]
 }
```


RemovePermission

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für einen `RemovePermission` API-Aufruf.


```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Jane",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Jane"
      },
      "eventTime": "2018-06-28T22:23:46Z",
      "eventSource": "sqs.amazonaws.com",
      "eventName": "RemovePermission",
      "awsRegion": "us-east-2",
      "sourceIPAddress": "203.0.113.3",
      "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
      "requestParameters": {
        "label": "label",
        "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
      },
      "responseElements": null,
      "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
      "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
    }
  ]
 }
```

SetQueueAttributes

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für`SetQueueAttributes`:

```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Maria",
         "accountId": "123456789012",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "userName": "Maria"
       },
       "eventTime": "2018-06-28T22:23:46Z",
       "eventSource": "sqs.amazonaws.com",
       "eventName": "SetQueueAttributes",
```

```
      "awsRegion": "us-east-2",
      "sourceIPAddress": "203.0.113.4",
      "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
Firefox/24.0",
      "requestParameters": {
        "attributes": {
          "VisibilityTimeout": "100"
        },
        "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
      },
      "responseElements": null,
      "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
      "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
    }
  ]
}
```

Beispiele: CloudTrail Datenereignisse für Amazon SQS

Im Folgenden finden Sie Beispiele für CloudTrail Ereignisse, die für Amazon SQS SQS-Datenereignis-APIs spezifisch sind:

SendMessage

Das folgende Beispiel zeigt ein CloudTrail Datenereignis für`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": {
    "mD5OfMessageBody": "9a4e3f7a614d9dd9f8722092dbda17a2",
    "mD5OfMessageSystemAttributes": "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"
  }
```

ReceiveMessage

Das folgende Beispiel zeigt ein CloudTrail Datenereignis für`ReceiveMessage`.

```
    {
  "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": "2023-11-07T23:59:24Z",
  "eventSource": "sqs.amazonaws.com",
  "eventName": "ReceiveMessage",
  "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",
    "maxNumberOfMessages": 10
  },
  "responseElements": null,
```

```
  "requestID": "8b4d4643-8f49-52cd-a6e8-1b875ed54b99",
  "eventID": "f3f23ab7-b0a4-4b71-afc0-141209c49206",
  "readOnly": true,
  "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"
  }
}
```

DeleteMessageBatch

Das folgende Beispiel zeigt ein CloudTrail Datenereignis für`DeleteMessageBatch`.

```
    {
  "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": "2023-11-07T23:59:24Z",
  "eventSource": "sqs.amazonaws.com",
  "eventName": "DeleteMessageBatch",
  "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",
    "entries": [
      {
        "id": "0",
        "receiptHandle": "AQEBefxM1O4zyZGF87DehbRbmri91w2W7mMdD0GrBjQa8e/
hpb4RbXHPZ9tLBVleECbChQIE5NtaDuoZhZPOkTy0eN46EyRR4jXDzE3AlkbPlX1mA9f2fUuTrXx8aeCoCA3I3woNg3f
hlLS94tjAZqV2krc4BaC2pYgjyHWcW019HwIV8T/bjNMIeZoQwOM5V
+o9vHPfewz5QGr5SKpDo7uE7Umyk5n5CJZvcn1efp/
mrwtaCIb9M7cCQUYcZm2ZmZDnIO9XpGTAi3m2dQ0M83pnNh0nvDfpkHpoa+hX1TrUmxCupCWHJwA8HFJ1O/
CCJsodMNFthLBA9S57dkBZCsw41G8jAmgQ0MkvZ0UL5mg0OFQQd1Yrw0zvthjCgiwdzn0yXoMzxIZMBxkY14E4nVVZ7N
h8oRk2C7gByzg2kYJ0LnUvLJFT8DQE28JZppEC9klvrdR/BWiPT7asc="
      }
    ]
  },
  "responseElements": {
    "successful": [
      {
        "id": "0"
      }
    ],
    "failed": []
  },
  "requestID": "fe423091-5642-5ba5-9256-6d5587de52f1",
  "eventID": "88c8020d-d769-4985-8ecb-ee0b59acc418",
  "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"
  }
}
```

ChangeMessageVisibilityBatch

Das folgende Beispiel zeigt ein CloudTrail Datenereignis für`ChangeMessageVisibilityBatch`.

```
    {
  "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": "2023-11-07T23:59:01Z",
  "eventSource": "sqs.amazonaws.com",
  "eventName": "ChangeMessageVisibilityBatch",
```

```
  "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": {
    "visibilityTimeout": 0,
    "entries": [
      {
        "id": "0",
        "receiptHandle":
 "AQEB2M5cVYg5gslhWME6537hdjcaPnOYPA5M0W460TTb0DzPle631yPWm8qxd4O1hDj/
B4ntTMnsgBTa95t14tNx7Vn96jKJ5rIoZ7iI8TRmkT1caKodKIPs8w9yndZq50c2FPQxtyH+2L3UHf/
abV3szqVWXOLZR4PwX8zZkWVQGNCNnY2q2lGCG586F8QwvrOFYoXNwB8ymd1t77e1PDPknq1Io3JFuzkEsndkkETy4fV
l5PHX17nXxaC+DURVlMPXOuSFACGmWqAoyk50HKwGOjLQgpySL/
TcnQXClvFq8kNXGwyVzJsbwHpOHxI7oce69vaD6DaWFP75d3hx+PJeG9pauQCKzVP3skt3Hw/
zDC7YfKcALD3aCwMmeNDwT3w0BUG6XZdG5lYhtFtTQYV7YuS3i/
Jh3HShGbtm07JKOEFiPkxv2+XNaAX3gFEpbng6zamTanfyMXCJIiglAEqiyWHQ=",
        "visibilityTimeout": 2271
      }
    ],
    "queueUrl": "https://sqs.ap-southeast-4.amazonaws.com/123456789012/MyQueue"
  },
  "responseElements": {
    "successful": [
      {
        "id": "0"
      }
    ]
  },
  "requestID": "d49ab65f-9dc7-54b8-875c-eb9b4c42988b",
  "eventID": "ca16c8c2-c4ba-4eb5-a54c-e650a10266d4",
  "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"
  }
}
```

Überwachen Amazon SQS SQS-Warteschlangen mit CloudWatch

Amazon SQS und Amazon CloudWatch sind integriert, sodass Sie CloudWatch Metriken für Ihre Amazon SQS SQS-Warteschlangen anzeigen und analysieren können. Sie können die Metriken Ihrer Warteschlangen über die Amazon SQS SQS-Konsole, die CloudWatch Konsole, mithilfe der oder mithilfe der AWS CLICloudWatch API anzeigen und analysieren. Sie können auch CloudWatch Alarme für Amazon SQS-Metriken einrichten.

CloudWatch Metriken für Ihre Amazon SQS SQS-Warteschlangen werden automatisch erfasst und in Intervallen von einer CloudWatch Minute weitergeleitet. Diese Metriken werden für alle Warteschlangen erfasst, die den CloudWatch Richtlinien für aktive Benutzer entsprechen. CloudWatchbetrachtet eine Warteschlange als bis zu sechs Stunden aktiv, wenn sie Nachrichten enthält oder wenn eine Aktion auf sie zugreift.

Wenn eine Amazon SQS-Warteschlange länger als sechs Stunden inaktiv ist, gilt der Amazon SQS-Service als inaktiv und stellt die Übermittlung von Metriken an den CloudWatch Service ein. Fehlende Daten oder Daten, die Null darstellen, können in den CloudWatch Kennzahlen für Amazon SQS für den Zeitraum, in dem Ihre Amazon SQS SQS-Warteschlange inaktiv war, nicht visualisiert werden.

> **Note**
>
> - Eine Amazon SQS SQS-Warteschlange kann aktiviert werden, wenn der Benutzer, der eine API für die Warteschlange aufruft, nicht autorisiert ist und die Anfrage fehlschlägt.
> - Die Amazon SQS SQS-Konsole führt einen `GetQueueAttributes` API-Aufruf durch, wenn die Seite der Warteschlange geöffnet wird. Die `GetQueueAttributes` API-Anfrage aktiviert die Warteschlange.
> - Bei CloudWatch Metriken kommt es zu einer Verzögerung von bis zu 15 Minuten, wenn eine Warteschlange aus einem inaktiven Zustand heraus aktiviert wird.

- Für die unter angegebenen Amazon SQS-Metriken fallen keine Gebühren an CloudWatch. Sie werden im Rahmen des Amazon-SQS-Service bereitgestellt.
- CloudWatch Metriken werden sowohl für Standard- als auch für FIFO-Warteschlangen unterstützt.

Themen

- Zugreifen auf CloudWatch Metriken für Amazon SQS
- CloudWatch Alarme für Amazon SQS-Metriken erstellen
- Verfügbare CloudWatch Metriken für Amazon SQS

Zugreifen auf CloudWatch Metriken für Amazon SQS

Amazon SQS und Amazon CloudWatch sind integriert, sodass Sie CloudWatch Metriken für Ihre Amazon SQS SQS-Warteschlangen anzeigen und analysieren können. Sie können die Metriken Ihrer Warteschlangen über die Amazon SQS SQS-Konsole, die CloudWatch Konsole, mithilfe der oder mithilfe der AWS CLICloudWatch API anzeigen und analysieren. Sie können auch CloudWatch Alarme für Amazon SQS-Metriken einrichten.

Amazon-SQS-Konsole

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Aktivieren Sie in der Liste der Warteschlangen die Kontrollkästchen für die Warteschlangen, auf deren Metriken Sie zugreifen möchten. Sie können Metriken für bis zu 10 Warteschlangen anzeigen.
3. Wählen Sie die Registerkarte Überwachung.

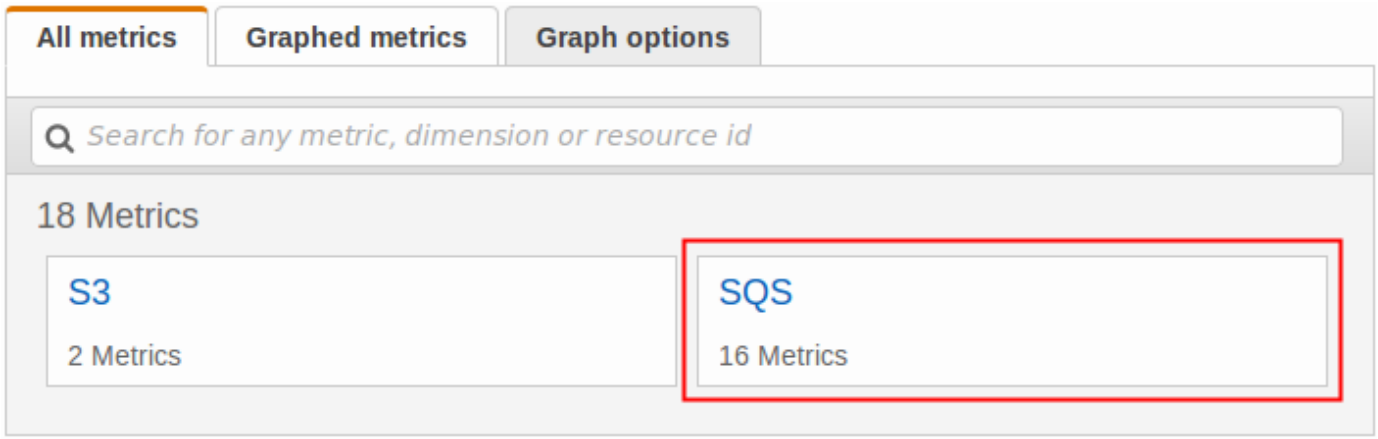
   Im Bereich SQS metrics werden verschiedene Diagramme angezeigt.
4. Wenn Sie wissen möchten, was ein bestimmtes Diagramm darstellt, zeigen Sie mit der Maus auf 

   neben dem gewünschten Diagramm. Weitere Informationen finden Sie auch unter Verfügbare CloudWatch Metriken für Amazon SQS.
5. Um den Zeitraum für alle Diagramme gleichzeitig zu ändern, wählen Sie für Time Range den gewünschten Zeitraum aus (z. B. Last Hour).

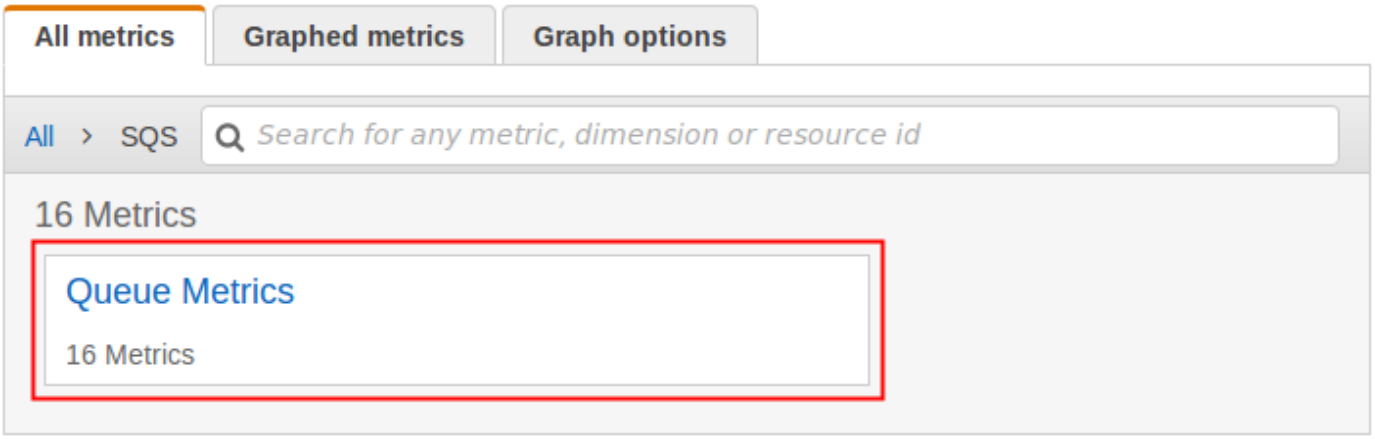
6. Zum Anzeigen zusätzlicher Statistiken für ein einzelnes Diagramm wählen Sie das Diagramm aus.
7. Wählen Sie im Dialogfeld „CloudWatch Überwachungsdetails“ eine Statistik aus (z. B. Summe). Eine Liste der unterstützten Statistiken finden Sie unter Verfügbare CloudWatch Metriken für Amazon SQS.
8. Zum Ändern des Zeitraums und Zeitintervalls, die ein einzelnes Diagramm anzeigt (z. B. einen Zeitraum der letzten 24 Stunden anstelle der letzten 5 Minuten oder einen Zeitraum für jede Stunde anstatt alle 5 Minuten) wählen Sie bei aktivem Diagrammdialogfeld für Time Range den gewünschten Zeitraum aus (z. B. Last 24 Hours). Wählen Sie für Period den gewünschten Zeitraum innerhalb des angegebenen Zeitraums aus (z. B. 1 Hour). Wenn Sie mit dem Diagramm fertig sind, klicken Sie auf Close.
9. (Optional) Um mit zusätzlichen CloudWatch Funktionen zu arbeiten, wählen Sie auf der Registerkarte Überwachung die Option Alle CloudWatch Metriken anzeigen aus, und folgen Sie dann den Anweisungen im CloudWatch Amazon-Konsole Verfahren.

CloudWatch Amazon-Konsole

1. Melden Sie sich an der CloudWatch-Konsole an.
2. Wählen Sie im Navigationsbereich Metriken aus.
3. Wählen Sie den Namespace der SQS-Metrik aus.

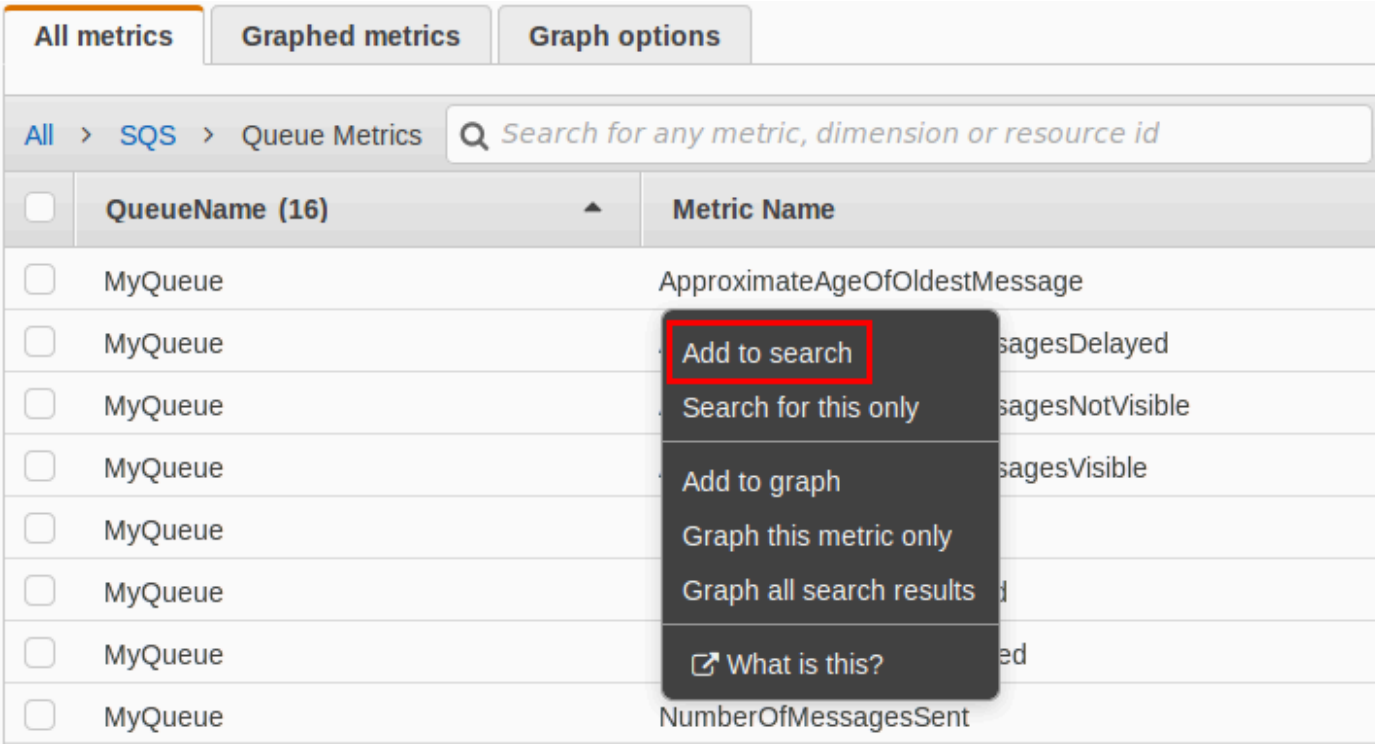


4. Wählen Sie die Metrikdimension Queue Metrics aus.



5. Sie können Ihre Amazon-SQS-Metriken jetzt analysieren:

   - Verwenden Sie die Spaltenüberschrift, um die Metriken zu sortieren.
   - Um eine Metrik grafisch darzustellen, müssen Sie das Kontrollkästchen neben der Metrik aktivieren.
   - Um nach Metrik zu filtern, müssen Sie den Metriknamen und anschließend Zur Suche hinzufügen auswählen.



Weitere Informationen und zusätzliche Optionen finden Sie unter Graph Metrics und Using Amazon CloudWatch Dashboards im CloudWatch Amazon-Benutzerhandbuch.

AWS Command Line Interface

Um mit dem auf Amazon SQS-Metriken zuzugreifen AWS CLI, führen Sie den `get-metric-statistics` Befehl aus.

Weitere Informationen finden Sie unter Get Statistics for a Metric im CloudWatch Amazon-Benutzerhandbuch.

CloudWatch API

Verwenden Sie die `GetMetricStatistics` Aktion, um mithilfe der CloudWatch API auf Amazon SQS-Metriken zuzugreifen.

Weitere Informationen finden Sie unter Get Statistics for a Metric im CloudWatch Amazon-Benutzerhandbuch.

CloudWatch Alarme für Amazon SQS-Metriken erstellen

CloudWatch ermöglicht das Auslösen von Alarmen auf der Grundlage eines metrischen Schwellenwerts. Sie können beispielsweise einen Alarm für die `NumberOfMessagesSent`-Metrik erstellen. Beispiel: Wenn in einer Stunde mehr als 100 Nachrichten an die `MyQueue`-Warteschlange gesendet werden, wird eine E-Mail-Benachrichtigung gesendet. Weitere Informationen finden Sie unter CloudWatch Amazon-Alarme erstellen im CloudWatch Amazon-Benutzerhandbuch.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Alarms und dann Create Alarm.
3. Wählen Sie im Abschnitt Select Metric (Metrik auswählen) im Dialogfeld Create Alarm (Alarm erstellen) die Optionen Browse Metrics (Metrik durchsuchen) und SQSaus.
4. Wählen Sie unter SQS > Queue Metrics den Metriknamen QueueNameund den Metriknamen aus, für den Sie einen Alarm einrichten möchten, und klicken Sie dann auf Weiter. Eine Liste der verfügbaren Metriken finden Sie unter Verfügbare CloudWatch Metriken für Amazon SQS.

   Im folgenden Beispiel handelt es sich bei der Auswahl um einen Alarm für die `NumberOfMessagesSent`-Metrik der `MyQueue`-Warteschlange. Der Alarm wird ausgelöst, wenn die Anzahl der gesendeten Nachrichten 100 überschreitet.

5. Gehen Sie im Abschnitt Define Alarm (Alarm festlegen) des Dialogfelds Create Alarm (Alarm erstellen) wie folgt vor:

   a. Geben Sie im Feld Alarm Threshold (Alarmschwellenwert) Name und Description (Beschreibung) für den Alarm ein.
   b. Legen Sie is auf > 100 fest.
   c. Setzen Sie for (für) auf den Wert 1 out of 1 datapoints (1 von 1 Datenpunkten).
   d. Setzen Sie unter Alarm preview (Alarmvorschau) Period (Zeitraum) auf 1 Hour (1 Stunde).
   e. Setzen Sie Statistic (Statistik) auf Standard, Sum (Summe).
   f. Setzen Sie unter Actions (Aktionen) Whenever this alarm (Wann immer dieser Alarm ausgegeben wird) auf State is ALARM (Status lautet ALARM).

      Wenn Sie eine Benachrichtigung senden CloudWatch möchten, wenn der Alarm ausgelöst wird, wählen Sie ein vorhandenes Amazon SNS SNS-Thema aus oder wählen Sie Neue Liste und geben Sie durch Kommas getrennte E-Mail-Adressen ein.

      > **Note**
      >
      > Wenn Sie ein neues Amazon-SNS-Thema erstellen, müssen die E-Mail-Adressen verifiziert werden, bevor die Empfänger Benachrichtigungen erhalten. Wenn es zu einer Änderung des Alarmstatus kommt, bevor die E-Mail-Adressen überprüft wurden, werden die Benachrichtigungen nicht versendet.

6. Wählen Sie Alarm erstellen aus.

   Der Alarm ist erstellt.

Verfügbare CloudWatch Metriken für Amazon SQS

Amazon SQS sendet die folgenden Metriken an CloudWatch.

> **Note**
>
> Bei Standard-Warteschlangen ist das Ergebnis aufgrund der verteilten Architektur von Amazon SQS ein ungefährer Wert. In den meisten Fällen sollte die Anzahl nahe an der tatsächlichen Anzahl von Nachrichten in der Warteschlange liegen.
> Bei FIFO-Warteschlangen ist der Wert exakt.

Amazon-SQS-Metriken

Der `AWS/SQS`-Namespace enthält die folgenden Metriken.

Metrik	Beschreibung
`ApproximateAgeOfOldestMessage`	Das ungefähre Alter der ältesten nicht gelöschten Mitteilung in der Warteschlange. **Note** • Eine Nachricht wird zurück in die Warteschlange verschoben, nachdem sie dreimal (oder häufiger) empfangen, aber nicht verarbeitet wurde, und die `ApproximateAgeOfOldestMessage` -Metrik verweist auf die zweitälteste Nachricht, die nicht mehr als dreimal empfangen wurde. Diese Aktion tritt auch dann auf, wenn die Warteschlange über eine Richtlinie für erneute Ausführung verfügt. • Da eine einzelne Poison-Pill-Nachricht (mehrfach empfangen, aber nie gelöscht) diese Metrik verzerren kann, ist das Alter einer Poison-Pill-Nachricht erst in der Metrik enthalten, wenn die Poison-Pill-Nachricht erfolgreich verbraucht wurde.

Metrik	Beschreibung
	• Wenn die Warteschlange über eine Redrive-Richtlinie verfügt, wird die Nachricht nach der konfigurierten maximalen Anzahl von Empfängen in eine Warteschlange für unzustellbare Nachrichten verschoben. Wenn die Nachricht in die Warteschlange für unzustellbare Nachrichten verschoben wird, steht die `ApproximateAgeOfOldestMessage` -Metrik der Warteschlange für unzustellbare Nachrichten für den Zeitpunkt, zu dem die Nachricht in die Warteschlange für unzustellbare Nachrichten verschoben wurde (nicht den ursprünglichen Zeitpunkt, an dem die Nachricht gesendet wurde). • Bei FIFO-Warteschlangen wird die Nachricht nicht an das Ende der Warteschlange verschoben, da dadurch die FIFO-Reihenfolgengarantie verletzt wird. Die Nachricht wird stattdessen an die DLQ gesendet, sofern eine konfiguriert ist. Andernfalls wird die Nachrichtengruppe blockiert, bis sie erfolgreich

Metrik	Beschreibung
	gelöscht wurde oder bis sie abläuft. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Sekunden Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt)
`ApproximateNumberOfMessagesDelayed`	Anzahl der Mitteilungen in der Warteschlange, die verzögert sind und nicht sofort gelesen werden können. Dies kann vorkommen, wenn die Warteschlange als Verzögerungswarteschlange konfiguriert ist oder eine Mitteilung mit einem Verzögerungsparameter gesendet worden ist. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Anzahl Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt)

Metrik	Beschreibung
`ApproximateNumberOfMessagesNotVisible`	Maximale Anzahl der in Übertragung befindlichen Mitteilungen. Die Mitteilungen befinden sich in Übertragung, wenn sie an einen Client gesendet, aber noch nicht gelöscht worden sind oder noch nicht das Ende ihres Sichtbarkeitsfensters erreicht haben. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Anzahl Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt)
`ApproximateNumberOfMessagesVisible`	Die Anzahl der Nachrichten, die verarbeitet werden sollen. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Anzahl Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt) Die Anzahl der Nachrichten an Prozesse ist unbegrenzt, Sie können diesen Rückstand jedoch einer Aufbewahrungsfrist unterwerfen.

Metrik	Beschreibung
`NumberOfEmptyReceives` [1]	Anzahl der `ReceiveMessage` -API-Aufrufe, die keine Mitteilung zurückgegeben haben. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Anzahl Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt)

Metrik	Beschreibung
`NumberOfMessagesDeleted` [1]	Anzahl der aus der Warteschlange gelöschten Mitteilungen. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Anzahl Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt) Amazon SQS übermittelt die `NumberOfMessagesDeleted` -Metrik für jeden erfolgreichen Löschvorgang, der eine gültige Empfangsmitteilung verwendet, einschließlich der Löschung von Duplikaten. Folgende Szenarien können zu einem höheren Metrikwert von `NumberOfMessagesDeleted` führen als erwartet: • Aufruf der Aktion `DeleteMessage` für verschiedene Empfangs-Mitteilungen, die zu derselben Mitteilung gehören: Wenn die Mitteilung nicht verarbeitet wird, bevor die Zeitbeschränkung für die Sichtbarkeit abläuft, ist die Mitteilung für andere Verbraucher verfügbar, die diese verarbeiten und erneut löschen können. Dabei wird der Wert der Metrik `NumberOfMessagesDeleted` erhöht. •

Metrik	Beschreibung
	Aufruf der Aktion `DeleteMessage` für dieselbe Empfangs-Mitteilung: Wenn die Mitteilung verarbeitet und gelöscht wird, Sie aber die Aktion `DeleteMessage` erneut unter Verwendung derselben Empfangs-Mitteilung aufrufen, wird ein Erfolgsstatus zurückgegeben und der Wert der Metrik `NumberOfMessagesDeleted` erhöht.
`NumberOfMessagesReceived` [1]	Anzahl der Mitteilungen, die infolge von Aufrufen der `ReceiveMessage` -Aktion zurückgegeben worden sind. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Anzahl Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt)

Metrik	Beschreibung
`NumberOfMessagesSent` [1]	Anzahl der einer Warteschlange hinzugefügten Mitteilungen. Wenn Sie eine Nachricht manuell an eine Warteschlange für unzustellbare Nachrichten senden, wird sie in der Metrik `NumberOfMessagesSent` erfasst. Wenn eine Nachricht jedoch aufgrund eines fehlgeschlagenen Verarbeitungsversuchs an eine Warteschlange mit unzustellbaren Nachrichten gesendet wird, wird sie von dieser Metrik nicht erfasst. Daher können die Werte von `NumberOfMessagesSent` und `NumberOfMessagesReceived` voneinander abweichen. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Anzahl Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt)

Metrik	Beschreibung
`SentMessageSize` [1]	Größe der einer Warteschlange hinzugefügten Mitteilungen. Berichtskriterien: Wenn die Warteschlange aktiv ist, wird ein nicht negativer Wert gemeldet. Einheiten: Byte Gültige Statistiken: Durchschnitt, Minimum, Maximum, Summe, Datenstichproben (wird als SampleCount in der Amazon-SQS-Konsole angezeigt) **Note** `SentMessageSize` wird in der CloudWatch Konsole erst als verfügbare Metrik angezeigt, wenn mindestens eine Nachricht an die entsprechende Warteschlange gesendet wurde.

[1] Diese Metriken werden aus Sicht des Services berechnet und können Wiederholungsversuche beinhalten. Verlassen Sie sich nicht auf die absoluten Werte dieser Messwerte und verwenden Sie sie auch nicht, um den aktuellen Warteschlangenstatus einzuschätzen.

Dimensionen für Amazon-SQS-Metriken

Die einzige Dimension, an die Amazon SQS sendet, CloudWatch ist`QueueName`. Dies bedeutet, dass alle verfügbaren Statistiken nach `QueueName` gefiltert werden.

Compliance-Validierung für Amazon SQS

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Basisumgebungen beschrieben AWS , bei denen Sicherheit und Compliance im Mittelpunkt stehen.
- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-fähige Anwendungen erstellen AWS können.

  > **Note**
  >
  > AWS-Services Nicht alle sind HIPAA-fähig. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.

- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.
- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Ausfallsicherheit bei Amazon SQS

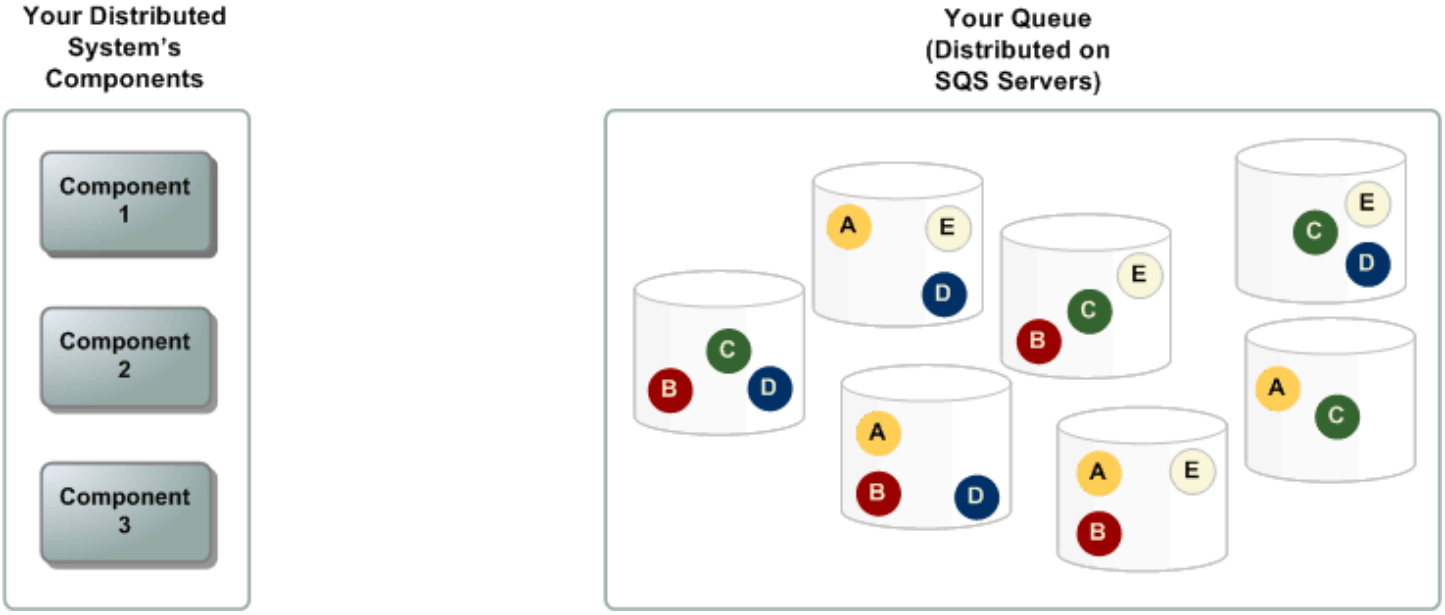
Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren. Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur AWS globalen Infrastruktur bietet Amazon SQS verteilte Warteschlangen.

Verteilte Warteschlangen

Es gibt drei Hauptkomponenten in einem verteilten Messaging-System: die Komponenten Ihres verteilten Systems, Ihre Warteschlange (auf Amazon-SQS-Server verteilt) und die Nachrichten in der Warteschlange.

Im folgenden Szenario hat Ihr System mehrere Produzenten (Komponenten, die Nachrichten an die Warteschlange senden) und Konsumenten (Komponenten, die Nachrichten aus der Warteschlange empfangen). Die Warteschlange (die die Nachrichten A bis E enthält) speichert die Nachrichten redundant auf mehreren Amazon-SQS-Servern.



Infrastruktursicherheit in Amazon SQS

Als verwalteter Service ist Amazon SQS durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben sind.

Sie verwenden AWS veröffentlichte API-Aktionen, um über das Netzwerk auf Amazon SQS zuzugreifen. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Clients müssen außerdem Cipher Suites mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen.

Anforderungen müssen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, die mit einem IAM-Prinzipal verknüpft sind. Alternativ können Sie die AWS Security Token Service (AWS STS) verwenden, um temporäre Anmeldeinformationen für Signaturanforderungen zu generieren.

Sie können API-Aktionen von jedem Netzwerkstandort aus aufrufen. Amazon SQS unterstützt jedoch ressourcenbasierte Zugriffsrichtlinien, die Einschränkungen auf der Basis der Quell-IP-Adresse enthalten können. Sie können auch Amazon-SQS-Richtlinien verwenden, um den Zugriff über bestimmte Amazon-VPC-Endpunkte oder bestimmte VPCs zu steuern. Dadurch wird der Netzwerkzugriff auf eine bestimmte Amazon SQS SQS-Warteschlange effektiv nur von der spezifischen VPC innerhalb des Netzwerks isoliert. AWS Weitere Informationen finden Sie unter Beispiel 5: Verweigerung des Zugriffs, wenn dieser nicht von einem VPC-Endpunkt aus erfolgt.

Bewährte Methoden für die Sicherheit in Amazon SQS

AWS bietet viele Sicherheitsfunktionen für Amazon SQS, die Sie im Zusammenhang mit Ihrer eigenen Sicherheitsrichtlinie überprüfen sollten. Im Folgenden werden bewährte vorbeugende Sicherheitsmethoden für Amazon SQS beschrieben.

> **Note**
>
> Die spezifischen Implementierungshinweise sind für häufige Anwendungsfälle und Implementierungen vorgesehen. Wir empfehlen Ihnen, diese bewährten Methoden im Kontext Ihres spezifischen Anwendungsfalls, der Architektur und des Bedrohungsmodells zu betrachten.

Themen

- Sicherstellen, dass Warteschlangen nicht öffentlich zugänglich sind
- Implementieren der geringstmöglichen Zugriffsrechte
- Verwenden Sie IAM-Rollen für Anwendungen und AWS Services, für die Amazon SQS SQS-Zugriff erforderlich ist
- Implementieren serverseitiger Verschlüsselung
- Erzwingen der Verschlüsselung von Daten während der Übertragung
- Erwägen der Verwendung von VPC-Endpunkten für den Zugriff auf Amazon SQS

Sicherstellen, dass Warteschlangen nicht öffentlich zugänglich sind

Sofern Sie nicht ausdrücklich verlangen, dass jemand im Internet Ihre Amazon SQS SQS-Warteschlange lesen oder in sie schreiben kann, sollten Sie sicherstellen, dass Ihre Warteschlange nicht öffentlich zugänglich ist (für jeden auf der Welt oder für jeden authentifizierten AWS Benutzer zugänglich).

- Vermeiden Sie das Erstellen von Richtlinien mit auf `""` festgelegtem `Principal`.
- Vermeiden Sie die Verwendung eines Platzhalters (`*`). Benennen Sie stattdessen einen oder mehrere bestimmte Benutzer.

Implementieren der geringstmöglichen Zugriffsrechte

Wenn Sie Berechtigungen erteilen, entscheiden Sie, wer sie erhält, für welche Warteschlangen die Berechtigungen gelten, und welche bestimmten API-Aktionen für diese Warteschlangen zugelassen werden. Die Implementierung von geringsten Privilegien ist wichtig, um Sicherheitsrisiken zu verringern und die Auswirkungen von Fehlern oder böswilligen Absichten zu reduzieren.

Folgen Sie den standardmäßigen Sicherheitshinweisen zur Erteilung von geringsten Privilegien. Das heißt: erteilen Sie nur die Berechtigungen, die zum Ausführen einer bestimmten Aufgabe erforderlich sind. Sie können dies mithilfe einer Kombination mehrerer Sicherheitsrichtlinien implementieren.

Amazon SQS verwendet das Producer-Consumer-Modell, für das drei Arten von Benutzerkontenzugriff erforderlich sind:

- Administratoren – Zugriff auf das Erstellen, Ändern und Löschen von Warteschlangen. Administratoren steuern auch Warteschlangenrichtlinien.
- Produzenten – Zugriff auf das Senden von Nachrichten an Warteschlangen.
- Konsumenten – Zugriff auf das Empfangen und Löschen von Nachrichten aus Warteschlangen.

Weitere Informationen finden Sie in den folgenden Abschnitten:

- Identity and Access Management in Amazon SQS
- Amazon-SQS-API-Berechtigungen: Referenz für Aktionen und Ressourcen
- Verwenden von benutzerdefinierten Richtlinien mit der Sprache der Zugriffsrichtlinie von Amazon SQS

Verwenden Sie IAM-Rollen für Anwendungen und AWS Services, für die Amazon SQS SQS-Zugriff erforderlich ist

Damit Anwendungen oder AWS Dienste wie Amazon EC2 auf Amazon SQS SQS-Warteschlangen zugreifen können, müssen sie in ihren AWS API-Anfragen gültige AWS Anmeldeinformationen verwenden. Da diese Anmeldeinformationen nicht automatisch rotiert werden, sollten Sie die AWS Anmeldeinformationen nicht direkt in der Anwendung oder EC2-Instance speichern.

Sie sollten mithilfe einer IAM-Rolle temporäre Anmeldeinformationen für Anwendungen und Services verwalten, die Zugriff auf Amazon SQS benötigen. Wenn Sie eine Rolle verwenden, müssen Sie keine langfristigen Anmeldeinformationen (wie einen Benutzernamen, ein Passwort und

Zugriffsschlüssel) an eine EC2-Instance oder einen AWS EC2-Dienst wie verteilen. AWS Lambda Stattdessen stellt die Rolle temporäre Berechtigungen bereit, die Anwendungen verwenden können, wenn sie andere AWS Ressourcen aufrufen.

Weitere Informationen finden Sie unter IAM-Rollen und Gängige Szenarien für Rollen: Benutzer, Anwendungen und Services im IAM Benutzerhandbuch.

Implementieren serverseitiger Verschlüsselung

Probleme durch Datenlecks lassen sich verringern, indem Sie die Verschlüsselung im Ruhezustand verwenden. Dabei verschlüsseln Sie Ihre Nachrichten mithilfe eines Schlüssels, der an einem anderen Speicherort gespeichert ist als Ihre Nachrichten. Serverseitige Verschlüsselung (SSE) bietet Datenverschlüsselung im Ruhezustand. Amazon SQS verschlüsselt Ihre Daten auf Nachrichtenebene bei der Speicherung und entschlüsselt die Nachrichten für Sie bei Zugriff darauf. SSE verwendet Schlüssel, die in AWS Key Management Service verwaltet werden. Solange Sie Ihre Anfrage authentifizieren und Zugriffsberechtigungen haben, gibt es keinen Unterschied zwischen dem Zugriff auf verschlüsselte und unverschlüsselte Warteschlangen.

Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand in Amazon SQS und Amazon SQS Schlüsselverwaltung.

Erzwingen der Verschlüsselung von Daten während der Übertragung

Ohne HTTPS (TLS) kann ein netzwerkbasierter Angreifer den Netzwerkverkehr abhören oder ihn manipulieren, indem er einen Angriff wie verwendet. man-in-the-middle Erlauben Sie nur verschlüsselte Verbindungen über HTTPS (TLS) unter Verwendung der`aws:SecureTransport`-Bedingung in der Warteschlangenrichtlinie, um zu erzwingen, dass Anforderungen SSL verwenden.

Erwägen der Verwendung von VPC-Endpunkten für den Zugriff auf Amazon SQS

Verwenden Sie bei Warteschlangen, mit denen eine Interaktion erforderlich ist, die jedoch dem Internet absolut nicht zugänglich gemacht werden dürfen, VPC-Endpunkte, um den Zugriff nur auf die Hosts innerhalb einer bestimmten VPC in die Warteschlange festzulegen. Mit Warteschlangenrichtlinien können Sie den Zugriff auf Warteschlangen von bestimmten Amazon-VPC-Endpunkten oder von bestimmten VPCs aus steuern.

Amazon-SQS-VPC-Endpunkte bieten zwei Möglichkeiten zur Kontrolle des Zugriffs auf Ihre Nachrichten:

- Sie können steuern, welche Anfragen, Benutzer oder Gruppen durch einen spezifischen VPC-Endpunkt erlaubt sind.
- Mithilfe einer Warteschlangenrichtlinie können Sie steuern, welche VPCs oder VPC-Endpunkte Zugriff auf Ihre Warteschlange haben.

Weitere Informationen finden Sie unter Endpunkte von Amazon Virtual Private Cloud für Amazon SQS und Erstellen einer Amazon-VPC-Endpunkt-Richtlinie für Amazon SQS.

Verwandte Amazon-SQS-Ressourcen

Die folgende Tabelle enthält verwandte Ressourcen, die für die Arbeit mit diesem Service nützlich sind.

Ressource	Beschreibung
Amazon Simple Queue Service – API-Referenz	Beschreibungen der Aktionen, Parameter und Datentypen sowie eine Liste von Fehlern, die der Service zurückgibt.
Amazon SQS in der AWS CLI - Befehlsreferenz	Beschreibungen der AWS CLI Befehle, mit denen Sie mit Warteschlangen arbeiten können.
Regionen und Endpunkte	Informationen zu Amazon-SQS-Regionen und -Endpunkten
Produktseite	Hauptwebsite für Informationen zu Amazon SQS.
Diskussionsforum	Ein auf der Community basierendes Forum, das für Entwickler eingerichtet wurde, um technische Fragen zu Amazon SWS zu klären.
AWS Informationen zum Premium-Support	Die wichtigste Webseite mit Informationen über AWS Premium Support, einen persönlichen, schnell reagierenden Support-Kanal, der Sie bei der Entwicklung und Ausführung von Anwendungen auf AWS Infrastrukturdiensten unterstützt.

Dokumentationsverlauf

In der folgenden Tabelle werden wichtige Änderungen am Amazon-Simple-Queue-Service-Entwicklerleitfaden seit Januar 2019 beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie den RSS-Feed abonnieren.

Servicefunktionen werden manchmal schrittweise in den AWS Regionen eingeführt, in denen ein Service verfügbar ist. Wir aktualisieren diese Dokumentation nur für die erste Version. Wir stellen keine Informationen über die Verfügbarkeit von Regionen zur Verfügung und kündigen auch keine späteren Rollouts von Regionen an. Informationen zur regionalen Verfügbarkeit von Servicefunktionen und zum Abonnieren von Benachrichtigungen über Updates finden Sie unter Was gibt's Neues bei AWS? .

Änderung	Beschreibung	Datum
AWS JSON-Protokoll	Stellen Sie API-Anfragen mithilfe des AWS JSON-Protokolls.	27. Juli 2023
Neuer Abschnitt, in dem AWS verwaltete Richtlinien für Amazon SQS und Aktualisierungen dieser Richtlinien beschrieben werden	Amazon SQS hat eine neue Aktion hinzugefügt, mit der Sie die neuesten Aufgaben zur Nachrichtenverschiebung (bis zu 10) in einer bestimmten Quellwarteschlange auflisten können. Diese Aktion ist mit dem `ListMessageMoveTasks` -API-Vorgang verknüpft.	7. Juni 2023
Redrive einer Warteschlange für unzustellbare Nachrichten mit APIs	Konfigurieren von Redrives für Warteschlangen für unzustellbare Nachrichten mithilfe von Amazon-SQS-APIs.	7. Juni 2023
ABAC für Amazon SQS	Attributbasierte Zugriffskontrolle (ABAC) mit Warteschlangen-Tags für flexible und	10. November 2022

	skalierbare Zugriffsberechtigungen.	
Das FIFO-Limit für hohen Durchsatz wird erhöht	Erhöhung der Standardkontingente für den FIFO-Hochdurchsatzmodus in kommerziellen Regionen sowie FIFO-Dokumentenoptimierung mit hohem Durchsatz.	20. Oktober 2022
Standardmäßige serverseitige Verschlüsselung (SSE) ist verfügbar	Serverseitige Verschlüsselung (SSE) mit standardmäßiger SQS-eigener Verschlüsselung (SSE-SQS).	26. September 2022
Unterstützung für Amazon SQS Confused Deputy Protection ist verfügbar	Confused Deputy Protection ermöglicht Ihnen, neue Header in Anfragen anzugeben, die bei Verwendung von Amazon SQS Managed SSE anhand der Bedingungen in der KMS-Richtlinie überprüft werden.	29. Dezember 2021
Managed SSE ist verfügbar	Amazon SQS Managed SSE (SSE-SQS) ist eine verwaltete serverseitige Verschlüsselung, die Amazon-SQS-eigene Verschlüsselungsschlüssel verwendet, um sensible Daten zu schützen, die über Nachrichtenwarteschlangen gesendet werden.	23. November 2021
Redrive von Warteschlangen für unzustellbare Nachrichten ist verfügbar	Amazon SQS unterstützt das Redrive von Warteschlangen für unzustellbare Nachrichten für Standard-Warteschlangen.	10. November 2021

Hoher Durchsatz für Nachrichten in FIFO-Warteschlangen ist verfügbar	Ein hoher Durchsatz für Amazon-SQS-FIFO-Warteschlangen ermöglicht eine höhere Anzahl von Transaktionen pro Sekunde (TPS) für Nachrichten in FIFO-Warteschlangen. Informationen zu Durchsatzkontingenten finden Sie unter Kontingente für Nachrichten.	27. Mai 2021
Hoher Durchsatz für Nachrichten in FIFO-Warteschlangen ist in der Vorschauversion verfügbar	Amazon-SQS-FIFO-Warteschlangen mit hohem Durchsatz sind als Vorschauversion verfügbar können kann sich noch ändern. Dieses Feature bietet eine höhere Anzahl von Transaktionen pro Sekunde (TPS) für Nachrichten in FIFO-Warteschlangen. Informationen zu Durchsatzkontingenten finden Sie unter Kontingente für Nachrichten.	17. Dezember 2020
Neues Amazon-SQS-Konsolendesign	Um Entwicklungs- und Produktionsabläufe zu vereinfachen, bietet die Amazon-SQS-Konsole eine neue Benutzerumgebung.	8. Juli 2020
Amazon SQS unterstützt die Paginierung für ListQueues und listDeadLetter SourceQueues	Sie können die maximale Anzahl von Ergebnissen angeben, die von einer ListQueues - oder DeadLetterSourceQueuesListenanforderung zurückgegeben werden sollen.	22. Juni 2020

Amazon SQS unterstützt CloudWatch 1-Minuten-Amazon-Metriken in allen AWS Regionen außer den Regionen AWS GovCloud (USA)	Die CloudWatch Ein-Minuten-Metrik für Amazon SQS ist in allen Regionen außer den AWS GovCloud (US) Regionen verfügbar.	9. Januar 2020
Amazon SQS unterstützt CloudWatch 1-Minuten-Metriken	Die einminütige CloudWatch Metrik für Amazon SQS ist derzeit nur in den folgenden Regionen verfügbar: USA Ost (Ohio), Europa (Irland), Europa (Stockholm) und Asien-Pazifik (Tokio).	25. November 2019
AWS Lambda Auslöser für Amazon SQS FIFO-Warteschlangen sind verfügbar	Sie können die in einer FIFO-Warteschlange Nachrichten so konfigurieren, dass eine Lambda-Funktion ausgelöst wird.	25. November 2019
Serverseitige Verschlüsselung (SSE) für Amazon SQS ist in den chinesischen Regionen verfügbar	SSE für Amazon SQS ist in den chinesischen Regionen verfügbar.	13. November 2019
FIFO-Warteschlangen sind in der Region Naher Osten (Bahrain) verfügbar	FIFO-Warteschlangen sind in der Region Naher Osten (Bahrain) verfügbar.	10. Oktober 2019
Amazon Virtual Private Cloud (Amazon VPC) - Endpunkte für Amazon SQS sind in den Regionen AWS GovCloud (USA-Ost) und AWS GovCloud (US-West) verfügbar	Sie können Nachrichten von Amazon VPC in den Regionen AWS GovCloud (USA-Ost) und AWS GovCloud (US-West) an Ihre Amazon SQS-Warteschlangen senden.	5. September 2019

Amazon SQS ermöglicht die Fehlerbehebung von Warteschlangen AWS X-Ray mithilfe von Nachrichtensystemattributen	Sie können mithilfe von X-Ray Fehler bei Nachrichten beheben, die über Amazon-SQS-Warteschlangen weitergeleitet werden. Diese Version fügt `MessageSystemAttribute` -Anforderungsparameter zu den API-Aktionen `SendMessage` und `SendMessageBatch` (mit denen Sie X-Ray-Ablaufverfolgungs-Header über Amazon SQS senden können), das `AWSTraceHeader` -Attribut zu der API-Aktion `ReceiveMessage` und den Datentyp `MessageSystemAttributeValue` hinzu.	28. August 2019
Sie können Amazon-SQS-Warteschlangen bei der Erstellung mit Tags markieren	Sie können einen einzigen Amazon SQS SQS-API-Aufruf, eine AWS SDK-Funktion oder einen AWS Command Line Interface (AWS CLI) -Befehl verwenden, um gleichzeitig eine Warteschlange zu erstellen und ihre Tags anzugeben. Darüber hinaus unterstützt Amazon SQS die Schlüssel `aws:TagKeys` und `aws:RequestTag` AWS Identity and Access Management (IAM).	22. August 2019

Der temporäre Warteschlangen-Client für Amazon SQS ist jetzt verfügbar	Temporäre Warteschlangen helfen Ihnen, Entwicklungszeit und Bereitstellungskosten zu sparen, wenn Sie gängige Nachrichtenmuster wie Anfrage-Antwort verwenden . Sie können den Temporären Warteschlangen-Client verwenden, um kostengünstige, anwendungsverwaltete temporäre Warteschlangen mit hohem Durchsatz zu erstellen.	25. Juli 2019
SSE für Amazon SQS ist in der Region AWS GovCloud (USA-Ost) verfügbar	Serverseitige Verschlüsselung (SSE) für Amazon SQS ist in der Region AWS GovCloud (USA-Ost) verfügbar.	20. Juni 2019
FIFO-Warteschlangen sind in den Regionen Asien-Pazifik (Hongkong), China (Peking), (USA Ost) und AWS GovCloud (USA West) verfügbar AWS GovCloud	FIFO-Warteschlangen sind in den Regionen Asien-Pazifik (Hongkong), China (Peking), (USA-Ost) und AWS GovCloud (US-West) verfügbar. AWS GovCloud	15. Mai 2019
Amazon-VPC-Endpunkt-Richtlinien sind für Amazon SQS verfügbar	Sie können Amazon-VPC-Endpunkt-Richtlinien für Amazon SQS erstellen.	4. April 2019
FIFO-Warteschlangen sind in den Regionen Europa (Stockholm) und China (Ningxia) verfügbar	FIFO-Warteschlangen sind in den Regionen Europa (Stockholm) und China (Ningxia) verfügbar.	14. März 2019

FIFO-Warteschlangen sind in allen Regionen verfügbar, in denen Amazon SQS verfügbar ist	FIFO-Warteschlangen sind in den Regionen USA Ost (Ohio), USA Ost (Nord-Virginia), USA West (Nordkalifornien), USA West (Oregon), Asien-Pazifik (Mumbai), Asien-Pazifik (Seoul), Asien-Pazifik (Singapur), Asien-Pazifik (Sydney), Asien-Pazifik (Tokio), Kanada (Zentral), Europa (Frankfurt), Europa (Irland), Europa (London), Europa (Paris) und Südamerika (São Paulo) verfügbar.	7. Februar 2019

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.