aws

개발자 가이드

Amazon Simple Queue Service

Amazon Simple Queue Service: 개발자 가이드

Table of Contents

Amazon 심플 큐 서비스란?

Amazon Simple Queue Service(Amazon SQS)는 내구력 있고 가용성이 뛰어난 보안 호스팅 대기열을 제공하며 이를 통해 분산 소프트웨어 시스템과 구성 요소를 통합 및 분리할 수 있습니다. Amazon SQS는 배달 못한 편지 대기열 및 비용 할당 태그와 같은 공용 구성을 제공합니다. AWS SDK가 지원하는 모든 프로그래밍 언어를 사용하여 액세스할 수 있는 일반 웹 서비스 API를 제공합니다.

주제

Amazon SQS 사용의 이점

- 보안 - Amazon SQS 대기열에 메시지를 보내고 받을 수 있는 사람을 제어합니다. 기본 Amazon SQS 관리형 서버 측 암호화(SSE)를 사용하거나 AWS Key Management Service (AWS KMS)에서 관리되는 사용자 지정 SSE 키를 사용하여 대기열에 있는 메시지의 콘텐츠를 보호함으로써 민감한 데이터를 전송하도록 선택할 수 있습니다.
- 내구성 – 메시지의 안전을 위해 Amazon SQS는 메시지를 여러 서버에 저장합니다. 표준 대기열은 at-least-once 메시지 전송을 지원하고 FIFO 대기열은 정확히 한 번의 메시지 처리 및 높은 처리량 모드를 지원합니다.
- 가용성 – Amazon SQS는 중복 인프라를 사용하여 메시지에 대한 고도의 동시 액세스와 메시지 생성 및 소비에 대한 고가용성을 제공합니다.
- 확장성 – Amazon SQS는 버퍼링된 요청을 각각 독립적으로 처리하여 프로비저닝 지침 없이도 로드 증가 또는 급증을 처리하기 위해 투명하게 확장할 수 있습니다.
- 신뢰성 – Amazon SQS는 처리 중에 메시지를 잠그므로 여러 생산자와 소비자가 동시에 메시지를 전송 및 수신할 수 있습니다.
- 사용자 지정 – 대기열이 똑같을 필요는 없습니다. 예를 들어 대기열에서 기본 지연 시간을 설정할 수 있습니다. Amazon S3 객체에 대한 포인터를 보유하는 Amazon SQS를 통해 Amazon DynamoDB 또는 Amazon Simple Storage Service(S3)를 사용하여 256KB보다 큰 메시지 콘텐츠를 저장하거나 큰 메시지를 더 작은 메시지로 분할할 수 있습니다.

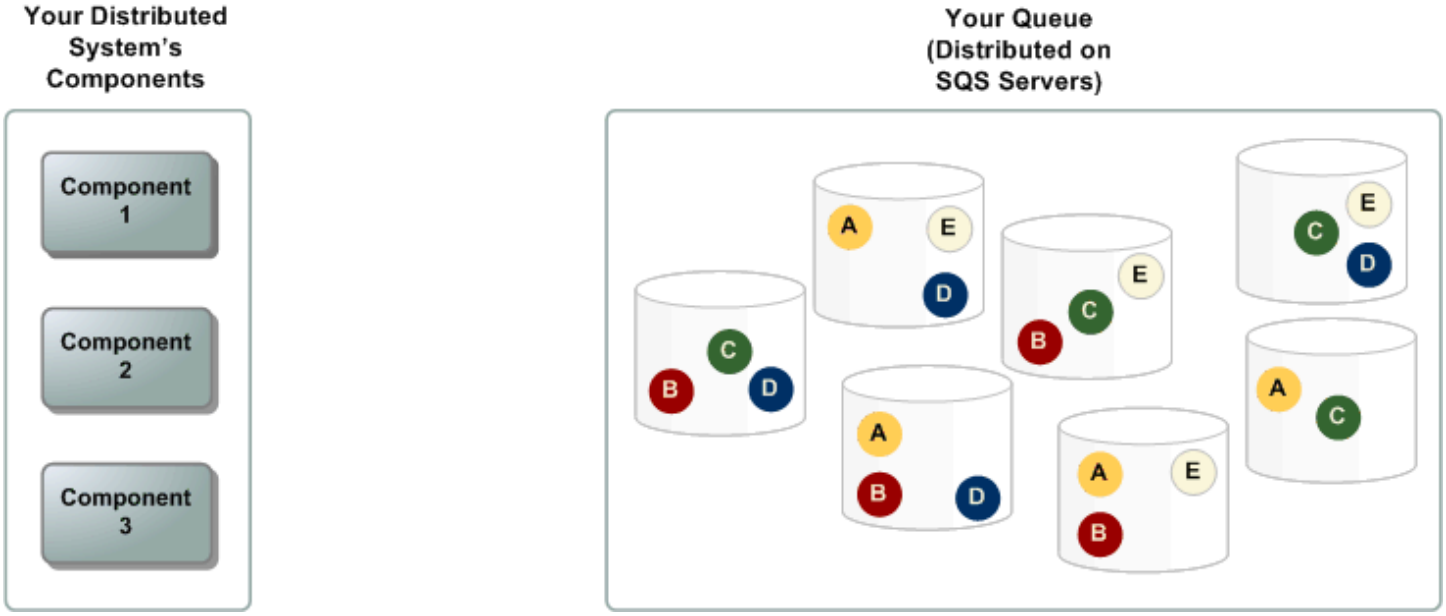
Amazon SQS 기본 아키텍처

이 섹션에서는 분산 메시징 시스템의 주요 부분에 대해 간략히 소개하고 Amazon SQS 메시지의 수명 주기에 대해 설명합니다.

분산 대기열

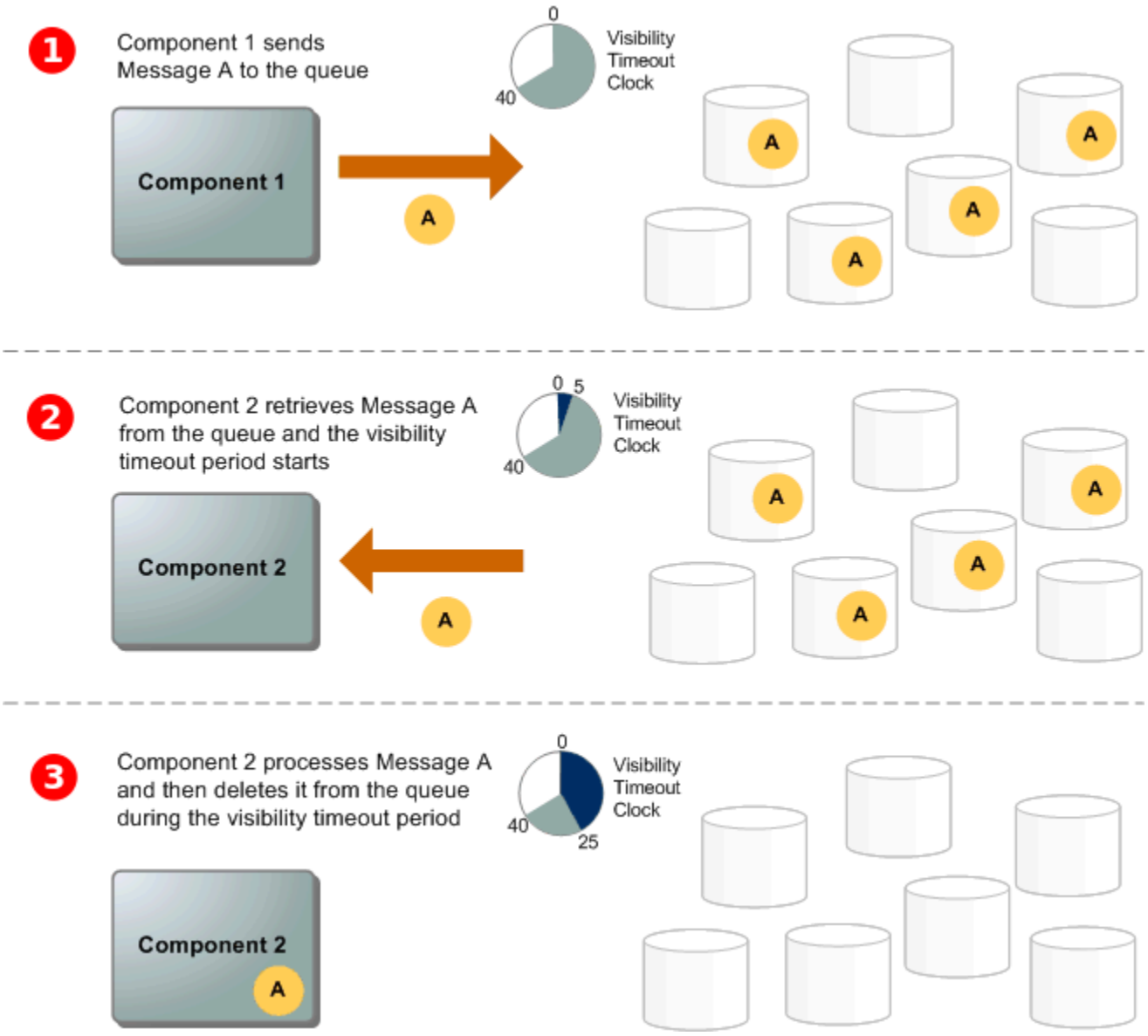
분산 메시징 시스템에는 세 가지 주요 부분, 즉 분산 시스템의 구성 요소, 대기열(Amazon SQS 서버에 분산됨), 대기열의 메시지가 있습니다.

다음 시나리오에서 시스템에는 여러 생산자(메시지를 대기열로 전송하는 구성 요소) 및 소비자(대기열의 메시지를 수신하는 구성 요소)가 있습니다. 대기열(메시지 A~E 유지)은 여러 Amazon SQS 서버에서 메시지를 중복으로 저장합니다.



메시지 수명 주기

다음 시나리오에서는 대기열에서 생성부터 삭제까지 Amazon SQS 메시지의 수명 주기를 설명합니다.



1

생산자(구성 요소 1)는 메시지 A를 대기열로 전송하고 이 메시지는 Amazon SQS 서버에서 중복 분산됩니다.

2

소비자(구성 요소 2)는 메시지를 처리할 준비가 되면 대기열에서 메시지를 소비하고 메시지 A가 반환됩니다. 메시지 A는 처리되는 동안 대기열에 그대로 남아 있고 제한 시간 초과가 지속되는 동안 후속 수신 요청으로 반환되지 않습니다.

3

소

비자(구성 요소 2)는 대기열에서 메시지 A를 삭제하여 제한 시간 초과가 만료되면 이 메시지가 수신되어 다시 처리되지 못하도록 합니다.

> **Note**
>
> Amazon SQS는 최대 메시지 보존 기간 넘게 대기열에 유지된 메시지를 자동으로 삭제합니다. 기본 메시지 보존 기간은 4일입니다. 그러나 `SetQueueAttributes` 작업을 사용하면 메시지 보존 기간을 60초에서 1,209,600초(14일) 사이로 설정할 수 있습니다.

Amazon SQS, Amazon MQ 및 Amazon SNS 간의 차이점

Amazon SQS, Amazon SNS 및 Amazon MQ는 확장성이 뛰어나고 easy-to-use 관리되는 메시징 서비스를 제공하며, 각 서비스는 분산 시스템 내에서 특정 역할을 위해 설계되었습니다. 다음은 이러한 서비스 간의 차이점에 대한 향상된 개요입니다.

Amazon SQS는 분산 소프트웨어 시스템과 구성 요소를 대기열 서비스로서 분리하고 확장합니다. 일반적으로 단일 구독자를 통해 메시지를 처리하므로 주문 및 손실 방지가 중요한 워크플로에 적합합니다. 더 넓은 배포를 위해 Amazon SQS를 Amazon SNS와 통합하면 팬아웃 메시징 패턴이 구현되어 한 번에 여러 구독자에게 메시지를 효과적으로 푸시할 수 있습니다.

Amazon SNS를 사용하면 게시자가 주제를 통해 여러 구독자에게 메시지를 보낼 수 있으며, 이는 커뮤니케이션 채널 역할을 합니다. 구독자는 Amazon SQS, Lambda, HTTP, 이메일 Amazon Data Firehose, 모바일 푸시 알림, 모바일 문자 메시지 (SMS) 등 지원되는 엔드포인트 유형을 사용하여 게시된 메시지를 수신합니다. 이 서비스는 실시간 사용자 참여 또는 경보 시스템과 같이 즉각적인 알림이 필요한 시나리오에 적합합니다. 구독자가 오프라인 상태일 때 메시지 손실을 방지하기 위해 Amazon SNS를 Amazon SQS 대기열 메시지와 통합하면 일관된 전송이 보장됩니다.

Amazon MQ는 Apache ActiveMQ 및 RabbitMQ와 함께 AMQP 및 MQTT와 같은 표준 메시징 프로토콜을 지원하여 기존 메시지 브로커에서 마이그레이션하려는 기업에 가장 적합합니다. 대폭적인 재구성 없이 안정적이고 신뢰할 수 있는 메시징이 필요한 레거시 시스템과의 호환성을 제공합니다.

다음 차트는 각 서비스의 리소스 유형에 대한 개요를 제공합니다.

리소스 유형	Amazon SNS	Amazon SQS	Amazon MQ
동기식	아니요	아니요	예

리소스 유형	Amazon SNS	Amazon SQS	Amazon MQ
비동기식	예	예	예
대기열	아니요	예	예
게시자-구독자 메시징	예	아니요	예
메시지 브로커	아니요	아니요	예

Amazon SQS와 Amazon SNS는 무제한에 가까운 확장성과 간편한 API를 활용할 수 있는 새로운 애플리케이션에 사용하면 좋습니다. 이들은 일반적으로 가격 면에서도 대용량 애플리케이션을 위한 더 비용 효율적인 솔루션을 제공합니다. pay-as-you-go JMS와 같은 API 또는 고급 메시지 큐 프로토콜 (AMQP), MQTT, OpenWire 단순 텍스트 지향 메시지 프로토콜 (STOMP) 과 같은 프로토콜과의 호환성을 사용하는 기존 메시지 브로커에서 애플리케이션을 마이그레이션할 때는 Amazon MQ를 사용하는 것이 좋습니다.

Amazon SQS 설정

Amazon SQS를 처음 사용하려면 먼저 다음 단계를 완료해야 합니다.

주제

1단계: AWS 계정 및 IAM 사용자 생성

AWS 서비스에 액세스하려면 먼저 제품을 사용할 수 있는 Amazon.com 계정을 AWS 계정만들어야 합니다. AWS 를 사용하여 활동 및 사용 보고서를 보고 인증 및 액세스를 관리할 수 있습니다. AWS 계정

Amazon SQS 작업에 AWS 계정 루트 사용자를 사용하지 않으려면 Amazon SQS에 대한 관리 액세스 권한이 필요한 각 사용자에 대해 IAM 사용자를 생성하는 것이 가장 좋습니다.

가입하십시오. AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요. AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

2단계: 프로그래밍 방식 액세스 권한 부여

Amazon SQS 작업 (예: Java 사용 또는 사용 AWS Command Line Interface) 을 사용하려면 액세스 키 ID와 보안 액세스 키가 필요합니다.

> **Note**
>
> 액세스 키 ID와 보안 액세스 키는 전용입니다. AWS Identity and Access Management Amazon EC2 키 페어와 같은 다른 AWS 서비스의 자격 증명과 혼동하지 마십시오.

AWS 외부 사용자와 상호 작용하려는 사용자는 프로그래밍 방식의 액세스가 필요합니다. AWS Management Console프로그래밍 방식의 액세스 권한을 부여하는 방법은 액세스하는 사용자 유형에 따라 다릅니다. AWS

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명할 수 있습니다. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI 에 대한 내용은 사용 설명서의 AWS CLI사용을 AWS IAM Identity Center위한 구성을 참조하십시오.AWS Command Line Interface • AWS SDK, 도구 및 AWS API의 경우 AWS SDK 및 도구 참조 안내서의 IAM ID 센터 인증을 참조하십시오.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 방식 요청에 서명할 수 있습니다. AWS	IAM 사용 설명서의 AWS 리소스와 함께 임시 자격 증명 사용의 지침을 따르십시오.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDK 또는 API에 대한 프로그래밍 요청에 서명하십시오. AWS	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 에 대한 내용은 사용 설명서의 IAM 사용자 자격 증명을 사용한 인증을 참조하십시오. AWS CLIAWS Command Line Interface • AWS SDK 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용한 인증을 참조하십시오.AWS • AWS API의 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하십시오.

3단계: 예제 코드 사용 준비

이 안내서에는 Java용 AWS SDK를 사용하는 예제가 포함되어 있습니다. 예제 코드를 실행하려면 Java 2.0용 AWS SDK 시작하기에 나와 있는 설정 지침을 따르세요.

Go, PythonJavaScript, Ruby와 같은 다른 프로그래밍 언어로 AWS 애플리케이션을 개발할 수 있습니다. 자세한 내용은 AWS빌드 기반 도구를 참조하십시오.

> **Note**
>
> AWS Command Line Interface (AWS CLI) 또는 Windows와 같은 도구를 사용하여 코드를 작성하지 않고도 Amazon SQS를 탐색할 수 있습니다. PowerShell AWS CLI 명령 참조의 Amazon SQS 섹션에서 AWS CLI 예제를 찾을 수 있습니다. AWS Tools for PowerShell

Cmdlet 참조의 Amazon 심플 큐 서비스 섹션에서 Windows PowerShell 예제를 찾을 수 있습니다.

다음 단계

이제 AWS Management Console을 사용하여 Amazon SQS 대기열 및 메시지 관리를 시작할 준비가 되었습니다.

Amazon SQS 시작하기

이 단원에서는 Amazon SQS 콘솔을 사용하여 표준 또는 FIFO 대기열을 생성하는 방법을 알아봅니다.

주제

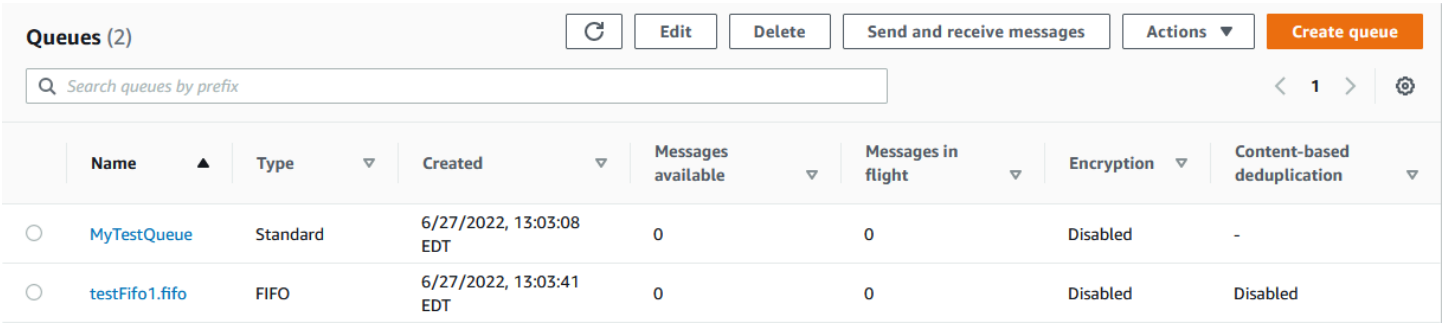
필수 조건

시작하기 전에 Amazon SQS 설정의 단계를 완료해야 합니다.

Amazon SQS 콘솔 이해

Amazon SQS 콘솔을 열고 탐색 창에서 [Queues] 를 선택합니다. 대기열 페이지는 활성 리전의 모든 대기열에 대한 정보를 제공합니다.

각 대기열 항목은 대기열의 유형 및 주요 속성을 포함하여 대기열에 대한 필수 정보를 제공합니다. 최대 처리량과 최선의 메시지 정렬을 위해 최적화된 표준 대기열은 엄격한 메시지 시퀀싱이 필요한 응용 프로그램에서 메시지 순서와 고유성을 우선시하는 FIFO (First-In-First-Out) 대기열과 구별됩니다.

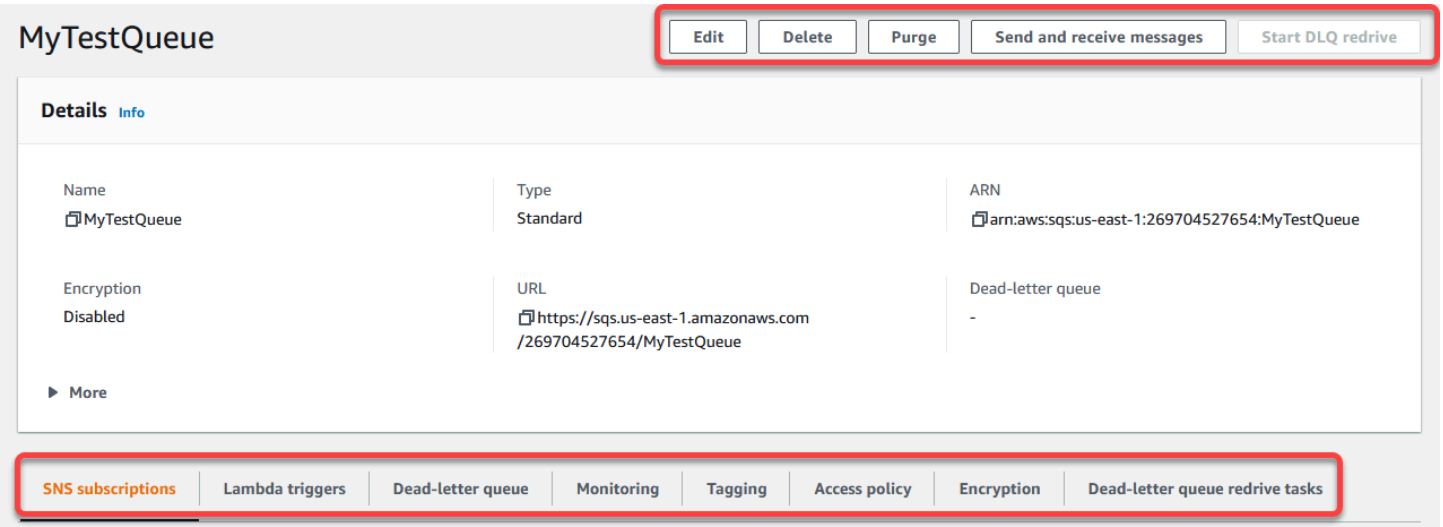


대화형 요소 및 작업

대기열 페이지에서 대기열을 관리하기 위한 여러 옵션을 사용할 수 있습니다.

1. 빠른 작업 - 각 대기열 이름 옆의 드롭다운 메뉴를 통해 메시지 전송, 메시지 보기 또는 삭제, 트리거 구성, 대기열 자체 삭제와 같은 일반적인 작업에 빠르게 액세스할 수 있습니다.

2. 세부 보기 및 구성 — 큐 이름을 클릭하면 세부 정보 페이지가 열리고 큐 설정 및 구성을 자세히 살펴볼 수 있습니다. 여기에서 메시지 보존 기간, 가시성 제한 시간, 최대 메시지 크기 등의 매개변수를 조정하여 애플리케이션 요구 사항에 맞게 대기열을 조정할 수 있습니다.



지역 선택 및 리소스 태그

대기열에 효과적으로 액세스하고 관리할 수 AWS 리전 있도록 올바른 위치에 있는지 확인하세요. 또한 리소스 태그를 사용하여 대기열을 구성 및 분류하여 공유 환경 내에서 더 나은 리소스 관리, 비용 할당 및 액세스 제어를 가능하게 하는 것도 고려해 보세요. AWS

Amazon SQS 콘솔에서 제공하는 기능을 활용하면 메시징 인프라를 효율적으로 관리하고 대기열 성능을 최적화하며 애플리케이션에 안정적인 메시지 전송을 보장할 수 있습니다.

Amazon SQS 대기열 유형

Amazon SQS는 표준 대기열과 FIFO 대기열이라는 두 가지 유형의 대기열을 지원합니다. 다음 표의 정보를 사용하여 상황에 맞는 대기열을 선택하세요. Amazon SQS 대기열에 대해 자세히 알아보려면 Amazon SQS 표준 대기열 시작하기 및 Amazon SQS에서 FIFO 대기열을 사용하기 시작하기 섹션을 참조하세요.

표준 대기열	FIFO 대기열
무제한 처리량 - 표준 대기열은 API 작업(`SendMessage` , `ReceiveMessage` 또는	높은 처리량 - 배치 처리를 사용하는 경우 FIFO 대기열은 API 메서드(`SendMessageBatch` , `ReceiveMessage` 또는 `DeleteMes`

표준 대기열	FIFO 대기열
DeleteMessage)별 초당 거의 무제한의 API 호출 수를 지원합니다. 최소 1회 전달 - 메시지가 최소한 한 번 전달되고, 가끔 2개 이상의 메시지 복사본이 전달될 수 있습니다. 최선의 정렬 가끔 메시지가 전송된 순서와 다르게 전달됩니다.	sageBatch)별 초당 최대 3,000개의 메시지를 지원합니다. 초당 3,000개의 메시지는 300개의 API 호출을 나타내며, 각각 10개의 메시지를 배치 처리합니다. 할당량 증가를 요청하려면 지원 요청을 제출하십시오. 배치 처리가 없으면 FIFO 대기열은 API 메서드(SendMessage , ReceiveMessage 또는 DeleteMessage)별 초당 최대 300개의 API 호출을 지원합니다. 정확히 1회 처리 - 메시지가 한 번 전달되고 소비자가 이를 처리 및 삭제할 때까지 유지됩니다. 중복 항목을 대기열에 삽입하지 않습니다. 선입선출 전송 - 메시지가 전송되고 수신되는 순서가 엄격하게 지켜집니다.
	
다음 예와 같이 처리량이 중요할 때 애플리케이션 간에 데이터를 전송합니다. • 실시간 사용자 요청을 폭넓은 배경 작업과 분리: 미디어 크기를 조정하거나 인코딩하는 동안 미디어를 업로드할 수 있습니다. • 작업을 여러 작업자 노드에 할당: 대량의 신용카드 확인 요청을 처리합니다. • 이후의 처리를 위해 메시지를 배치 처리합니다. 다수의 항목이 데이터베이스에 추가되도록 예약합니다.	다음 예와 같이 이벤트 순서가 중요할 때 애플리케이션 간에 데이터를 전송합니다. • 사용자가 입력한 명령이 올바른 순서로 실행되도록 보장합니다. • 가격 수정을 올바른 순서로 전송하여 올바른 제품 가격 표시. • 학생이 계정 등록 전에 과정에 등록하지 못하도록 차단.

Amazon SQS 표준 대기열 생성 및 메시지 전송

Amazon SQS용 표준 대기열을 생성하는 방법입니다.

Amazon SQS 콘솔을 사용하여 대기열을 생성합니다.

Amazon SQS 콘솔을 사용하여 표준 대기열을 만들 수 있습니다. 콘솔은 대기열 이름을 제외한 모든 설정의 기본값을 제공합니다.

> **Important**
>
> 2022년 8월 17일에 기본 서버 측 암호화(SSE)가 모든 Amazon SQS 대기열에 적용되었습니다.
> 개인 식별 정보(PII)나 기타 기밀 정보 또는 민감한 정보를 대기열 이름에 추가하지 마세요. 대금 청구 및 CloudWatch 로그를 비롯한 많은 Amazon Web Services에서 대기열 이름에 액세스할 수 있습니다. 대기열 이름은 개인 데이터나 민감한 데이터에 사용하기 위한 것이 아닙니다.

Amazon SQS 대기열을 생성하려면

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 대기열 생성을 선택합니다.
3. 유형의 경우 표준 대기열 유형이 기본적으로 설정됩니다.

   > **Note**
   >
   > 대기열을 생성한 후에는 대기열 유형을 변경할 수 없습니다.

4. 대기열의 이름을 입력합니다.
5. (선택 사항) 콘솔은 대기열 구성 파라미터의 기본값을 설정합니다. 구성에서 다음 파라미터에 새 값을 설정할 수 있습니다.

   a. 제한 시간 초과에는 기간과 단위를 입력합니다. 범위는 0초~12시간입니다. 기본값은 30초입니다.

   b. 메시지 보존 기간에는 기간과 단위를 입력합니다. 범위는 1분~14일입니다. 기본값은 4일입니다.

   c. 전송 지연에는 기간과 단위를 입력합니다. 범위는 0초~15분입니다. 기본값은 0초입니다.

   d. 최대 메시지 크기에는 값을 입력합니다. 범위는 1~256KB입니다. 기본값은 256KB입니다.

e. 메시지 수신 대기 시간은 값을 입력합니다. 범위는 0~20초입니다. 기본값은 0초이며 짧은 폴링을 설정합니다. 0이 아닌 값은 모두 긴 폴링을 설정합니다.

6. (선택 사항) 액세스 정책을 정의합니다. 액세스 정책은 대기열에 액세스할 수 있는 계정, 사용자 및 역할을 정의합니다. 또한 액세스 정책은 사용자가 액세스할 수 있는 작업(예: `SendMessage`, `ReceiveMessage` 또는`DeleteMessage`)을 정의합니다. 기본 정책에서는 대기열 소유자만 메시지를 보내고 받을 수 있도록 허용합니다.

   액세스 정책을 정의하려면 다음 중 하나를 수행합니다.

   - 대기열에 메시지를 보낼 수 있는 사람과 대기열에서 메시지를 받을 수 있는 사람을 구성하려면 기본을 선택합니다. 콘솔은 사용자의 선택에 따라 정책을 생성하고 결과 액세스 정책을 읽기 전용 JSON 창에 표시합니다.
   - JSON 액세스 정책을 직접 수정하려면 고급을 선택합니다. 이를 통해 각 주체(계정, 사용자 또는 역할)가 수행할 수 있는 사용자 지정 작업 집합을 지정할 수 있습니다.

7. 리드라이브 허용 정책의 경우 활성화를 선택합니다. 모두 허용, 대기열 기준 또는 모두 거부 중 하나를 선택합니다. 대기열 기준 선택 시 Amazon 리소스 이름(ARN)으로 최대 10개 소스 대기열의 목록을 지정합니다.

8. Amazon SQS는 기본적으로 관리형 서버 측 암호화를 제공합니다. 암호화 키 유형을 선택하거나 Amazon SQS 관리형 서버 측 암호화를 비활성화하려면 암호화를 확장합니다. 암호화 키 유형에 대한 자세한 내용은 SQS에서 관리하는 암호화 키를 사용하여 대기열에 대한 서버 측 암호화 구성 및 Amazon SQS 콘솔을 사용하여 대기열에 대한 서버 측 암호화 구성 섹션을 참조하세요.

   > **Note**
   >
   > SSE를 활성화하면 암호화된 대기열에 대한 익명 `SendMessage` 및 `ReceiveMessage` 요청이 거부됩니다. Amazon SQS 보안 모범 사례에서는 익명 요청을 사용하지 말 것을 권장합니다. Amazon SQS 대기열로 익명 요청을 보내려면 SSE를 비활성화해야 합니다.

9. (선택 사항) 배달되지 않은 메시지를 수신하도록 DLQ(Dead Letter Queue)를 구성하려면 DLQ(Dead Letter Queue)를 펼칩니다.

10. (선택 사항) 대기열에 태그를 추가하려면 태그를 펼칩니다.

11. 대기열 생성을 선택합니다. Amazon SQS가 대기열을 생성하고 대기열의 세부 정보 페이지를 표시합니다.

Amazon SQS는 새 대기열에 대한 정보를 시스템 전체에 전파합니다. Amazon SQS는 분산 시스템이므로 콘솔이 대기열 페이지에 대기열을 표시할 때까지 약간의 지연이 발생할 수 있습니다.

메시지 전송

대기열을 생성한 후 그 대기열에 메시지를 보낼 수 있습니다.

1. 왼쪽 탐색 창에서 대기열을 선택합니다. 대기열 목록에서 자신이 생성한 대기열을 선택합니다.
2. 작업에서 메시지 전송 및 수신을 선택합니다.

   콘솔에 메시지 전송 및 수신 페이지가 표시됩니다.
3. 메시지 본문에 메시지 텍스트를 입력합니다.
4. 표준 대기열의 경우 전송 지연 시간 값을 입력하고 단위를 선택할 수 있습니다. 예를 들어 60 입력 후 초를 선택합니다. 자세한 정보는 Amazon SQS 메시지 타이머을 참조하세요.
5. 메시지 전송을 선택합니다.

   메시지를 전송하면 콘솔에 성공 메시지가 표시됩니다. 전송한 메시지에 대한 정보를 표시하려면 세부 정보 보기를 선택합니다.

Amazon SQS FIFO 대기열 생성 및 메시지 전송

Amazon SQS용 FIFO 대기열을 생성하는 방법입니다.

대기열 생성

Amazon SQS 콘솔을 사용하여 FIFO 대기열을 만들 수 있습니다. 콘솔은 대기열 이름을 제외한 모든 설정의 기본값을 제공합니다.

> **Important**
>
> 2022년 8월 17일에 기본 서버 측 암호화(SSE)가 모든 Amazon SQS 대기열에 적용되었습니다.
> 개인 식별 정보(PII)나 기타 기밀 정보 또는 민감한 정보를 대기열 이름에 추가하지 마세요. 대금 청구 및 CloudWatch 로그를 비롯한 많은 Amazon Web Services에서 대기열 이름에 액세스할 수 있습니다. 대기열 이름은 개인 데이터나 민감한 데이터에 사용하기 위한 것이 아닙니다.

Amazon SQS FIFO 대기열 생성

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.

2. 대기열 생성을 선택합니다.
3. 유형의 경우 표준 대기열 유형이 기본적으로 설정됩니다. FIFO 대기열을 만들려면 FIFO를 선택합니다.

> **Note**
>
> 대기열을 생성한 후에는 대기열 유형을 변경할 수 없습니다.

4. 대기열의 이름을 입력합니다.

   FIFO 대기열의 이름은 `.fifo` 접미사로 끝나야 합니다. 접미사는 문자 80개의 대기열 이름 할당량에 포함됩니다. 해당 대기열이 접미사로 끝나는지 확인하여 FIFO 대기열인지 여부를 알 수 있습니다.

5. (선택 사항) 콘솔은 대기열 구성 파라미터의 기본값을 설정합니다. 구성에서 다음 파라미터에 새 값을 설정할 수 있습니다.

   a. 제한 시간 초과에는 기간과 단위를 입력합니다. 범위는 0초~12시간입니다. 기본값은 30초입니다.

   b. 메시지 보존 기간에는 기간과 단위를 입력합니다. 범위는 1분~14일입니다. 기본값은 4일입니다.

   c. 전송 지연에는 기간과 단위를 입력합니다. 범위는 0초~15분입니다. 기본값은 0초입니다.

   d. 최대 메시지 크기에는 값을 입력합니다. 범위는 1~256KB입니다. 기본값은 256KB입니다.

   e. 메시지 수신 대기 시간은 값을 입력합니다. 범위는 0~20초입니다. 기본값은 0초이며 짧은 폴링을 설정합니다. 0이 아닌 값은 모두 긴 폴링을 설정합니다.

   f. FIFO 대기열의 경우 콘텐츠 기반 중복 제거를 선택하여 콘텐츠 기반 중복 제거를 활성화합니다. 기본값은 비활성화입니다.

   g. (선택 사항) FIFO 대기열에서 메시지를 보내고 받을 때 처리량을 높이려면 높은 처리량 FIFO 활성화를 선택합니다.

      이 옵션을 선택하면 관련 옵션(중복 제거 범위 및 FIFO 처리량 한도)이 FIFO 대기열의 높은 처리량을 활성화하는 데 필요한 설정으로 변경됩니다. 높은 처리량 FIFO를 사용하는 데 필요한 설정을 변경하면 대기열에 일반 처리량이 적용되고 지정된 대로 중복 제거가 수행됩니다. 자세한 내용은 Amazon SQS의 FIFO 대기열에 대한 높은 처리량 및 Amazon SQS 메시지 할당량 섹션을 참조하세요.

6. (선택 사항) 액세스 정책을 정의합니다. 액세스 정책은 대기열에 액세스할 수 있는 계정, 사용자 및 역할을 정의합니다. 또한 액세스 정책은 사용자가 액세스할 수 있는 작업(예: `SendMessage`,

`ReceiveMessage` 또는`DeleteMessage`)을 정의합니다. 기본 정책에서는 대기열 소유자만 메시지를 보내고 받을 수 있도록 허용합니다.

액세스 정책을 정의하려면 다음 중 하나를 수행합니다.

- 대기열에 메시지를 보낼 수 있는 사람과 대기열에서 메시지를 받을 수 있는 사람을 구성하려면 기본을 선택합니다. 콘솔은 사용자의 선택에 따라 정책을 생성하고 결과 액세스 정책을 읽기 전용 JSON 창에 표시합니다.
- JSON 액세스 정책을 직접 수정하려면 고급을 선택합니다. 이를 통해 각 주체(계정, 사용자 또는 역할)가 수행할 수 있는 사용자 지정 작업 집합을 지정할 수 있습니다.

7. 리드라이브 허용 정책의 경우 활성화를 선택합니다. 모두 허용, 대기열 기준 또는 모두 거부 중 하나를 선택합니다. 대기열 기준 선택 시 Amazon 리소스 이름(ARN)으로 최대 10개 소스 대기열의 목록을 지정합니다.
8. Amazon SQS는 기본적으로 관리형 서버 측 암호화를 제공합니다. 암호화 키 유형을 선택하거나 Amazon SQS 관리형 서버 측 암호화를 비활성화하려면 암호화를 확장합니다. 암호화 키 유형에 대한 자세한 내용은 SQS에서 관리하는 암호화 키를 사용하여 대기열에 대한 서버 측 암호화 구성 및 Amazon SQS 콘솔을 사용하여 대기열에 대한 서버 측 암호화 구성 섹션을 참조하세요.

> **Note**
>
> SSE를 활성화하면 암호화된 대기열에 대한 익명 `SendMessage` 및 `ReceiveMessage` 요청이 거부됩니다. Amazon SQS 보안 모범 사례에서는 익명 요청을 사용하지 말 것을 권장합니다. Amazon SQS 대기열로 익명 요청을 보내려면 SSE를 비활성화해야 합니다.

9. (선택 사항) 배달되지 않은 메시지를 수신하도록 DLQ(Dead Letter Queue)를 구성하려면 DLQ(Dead Letter Queue)를 펼칩니다.
10. (선택 사항) 대기열에 태그를 추가하려면 태그를 펼칩니다.
11. 대기열 생성을 선택합니다. Amazon SQS가 대기열을 생성하고 대기열의 세부 정보 페이지를 표시합니다.

Amazon SQS는 새 대기열에 대한 정보를 시스템 전체에 전파합니다. Amazon SQS는 분산 시스템이므로 콘솔이 대기열 페이지에 대기열을 표시할 때까지 약간의 지연이 발생할 수 있습니다.

대기열을 생성한 후에는 대기열에 메시지를 보내고 메시지를 수신 및 삭제할 수 있습니다. 대기열 유형을 제외한 모든 대기열 구성 설정을 편집할 수도 있습니다.

메시지 전송

대기열을 생성한 후 그 대기열에 메시지를 보낼 수 있습니다.

1. 왼쪽 탐색 창에서 대기열을 선택합니다. 대기열 목록에서 자신이 생성한 대기열을 선택합니다.
2. 작업에서 메시지 전송 및 수신을 선택합니다.

   콘솔에 메시지 전송 및 수신 페이지가 표시됩니다.
3. 메시지 본문에 메시지 텍스트를 입력합니다.
4. 선입선출(FIFO) 대기열의 경우 메시지 그룹 ID를 입력합니다. 자세한 정보는 Amazon SQS의 FIFO 대기열 전송 로직을 참조하세요.
5. (선택 사항) FIFO 대기열의 경우 메시지 중복 제거 ID를 입력할 수 있습니다. 대기열에 콘텐츠 기반 중복 제거를 활성화한 경우에는 메시지 중복 제거 ID가 필요하지 않습니다. 자세한 정보는 Amazon SQS의 FIFO 대기열 전송 로직을 참조하세요.
6. FIFO 대기열은 개별 메시지의 타이머를 지원하지 않습니다. 자세한 정보는 Amazon SQS 메시지 타이머을 참조하세요.
7. 메시지 전송을 선택합니다.

   메시지를 전송하면 콘솔에 성공 메시지가 표시됩니다. 전송한 메시지에 대한 정보를 표시하려면 세부 정보 보기를 선택합니다.

Amazon SQS 대기열 관리

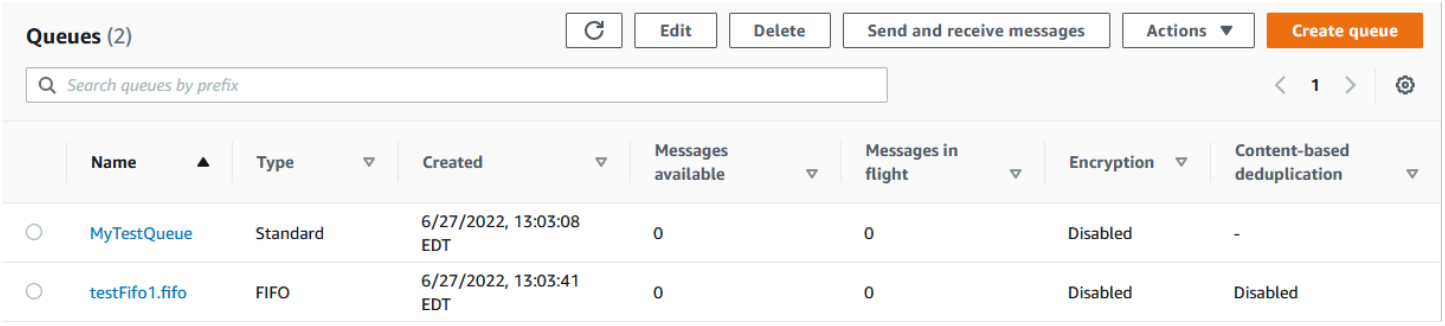
이 섹션에서는 Amazon SQS에 더 익숙해질 수 있도록 Amazon SQS 콘솔을 사용하여 대기열과 메시지를 관리하는 방법을 보여줍니다.

사전 조건

시작하기 전에 Amazon SQS 설정의 단계를 완료해야 합니다.

Amazon SQS 콘솔 이해

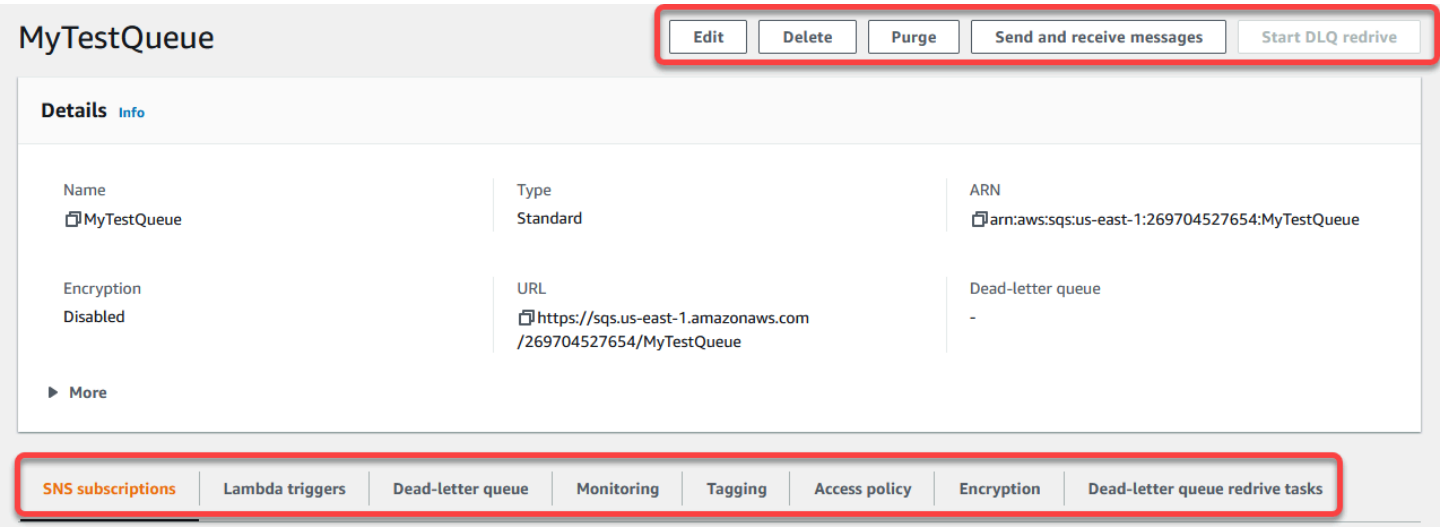
콘솔을 열 때 탐색 창에서 대기열을 선택하여 대기열 페이지를 표시합니다. 대기열 페이지는 활성 리전의 모든 대기열에 대한 정보를 제공합니다.



각 대기열 항목에는 대기열 유형과 대기열에 대한 기타 정보가 표시됩니다. 유형 열을 사용하면 표준 대기열과 FIFO(First-In-First Out) 대기열을 한 눈에 구분할 수 있습니다.

대기열 페이지에서는 두 가지 방법으로 대기열에 대한 작업을 수행할 수 있습니다. 대기열 이름 옆에 있는 옵션을 선택한 다음 대기열에서 수행하려는 작업을 선택할 수 있습니다.

대기열 이름을 선택하여 대기열의 세부 정보 페이지를 열 수도 있습니다. 세부 정보 페이지에는 대기열 페이지와 동일한 작업이 포함됩니다. 또한 세부 정보 섹션 아래의 탭 중 하나를 선택하여 추가 구성 세부 정보 및 작업을 볼 수 있습니다.



콘솔을 사용하여 Amazon SQS 대기열 편집

Amazon SQS 콘솔을 사용하여 대기열 구성 파라미터(대기열 유형 제외)를 편집하고 대기열 기능을 추가 또는 제거할 수 있습니다.

Amazon SQS 대기열을 편집하려면(콘솔)

1. Amazon SQS 콘솔의 대기열 페이지를 엽니다.
2. 대기열을 선택한 다음 편집을 선택합니다.
3. (선택 사항) 구성에서 대기열의 구성 파라미터를 업데이트합니다.
4. (선택 사항) 액세스 정책을 업데이트하려면 액세스 정책에서 JSON 정책을 수정합니다.
5. (선택 사항) 배달 못한 편지 대기열 리드라이브 허용 정책을 업데이트하려면 리드라이브 허용 정책을 확장합니다.
6. (선택 사항) 암호화를 업데이트하거나 제거하려면 암호화를 확장합니다.
7. (선택 사항) 배달 못한 편지 대기열을 추가, 업데이트 또는 제거하려면(전달할 수 없는 메시지를 수신할 수 있음) 배달 못한 편지 대기열을 확장합니다.
8. (선택 사항) 대기열에 태그를 추가, 업데이트 또는 제거하려면 태그를 확장합니다.
9. 저장을 선택합니다.

   콘솔에 대기열에 대한 세부 정보 페이지가 표시됩니다.

Amazon SQS에서 메시지 수신 및 삭제

Amazon SQS 대기열로 메시지를 전송한 후 메시지를 수신하고 삭제할 수 있습니다. 대기열에서 메시지를 요청할 때는 개별 메시지를 지정할 수 없습니다. 대신 검색하려는 최대 메시지 수 (최대 10개) 를 결정합니다.

Amazon SQS는 분산 시스템으로 작동하므로 메시지가 거의 없는 대기열에서 메시지를 검색할 때 가끔 빈 응답이 발생할 수 있습니다. 이 경우 요청을 다시 실행하기만 하면 됩니다. 메시지 검색을 최적화하고 빈 응답을 최소화하려면 긴 폴링을 사용하는 것이 좋습니다. 폴링이 길면 메시지를 사용할 수 있게 되거나 폴링 제한 시간이 초과될 때까지 응답이 지연되므로 불필요한 폴링 비용이 줄어들고 효율성이 향상됩니다.

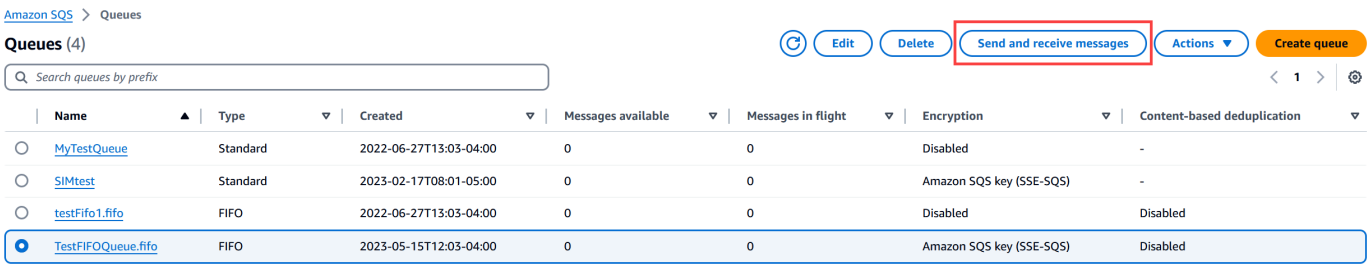
Amazon SQS에서는 애플리케이션 문제 또는 네트워크 장애와 같은 처리 실패로 인해 메시지에 대한 액세스 권한을 잃지 않기 때문에 메시지가 검색된 후에도 자동으로 삭제되지 않습니다. 대기열에서 메시지를 영구적으로 제거하려면 메시지를 처리한 후 명시적으로 삭제 요청을 보내 성공적으로 수신 및 처리되었는지 확인해야 합니다.

Amazon SQS 콘솔을 통해 메시지를 검색하면 다시 검색할 수 있도록 즉시 다시 표시됩니다. 이 기본 동작은 수동 작업 중에 실수로 메시지가 손실되는 일이 없도록 해주지만 반복 처리로 이어질 수 있습니다. 자동화된 환경에서는 가시성 제한 시간 설정을 조정하여 메시지가 검색된 후 다른 소비자가 볼 수 없는 상태로 유지되는 기간을 제어하십시오. 이 설정은 여러 소비자의 메시지 처리를 조정하고 메시지가 한 번만 처리되도록 하는 데 매우 중요합니다.

메시지 수신 및 삭제에 대한 자세한 작업은 Amazon SQS API 참조 안내서를 참조하십시오. 이 안내서는 복잡한 메시지 처리 시나리오를 효과적으로 관리하는 파라미터를 포함하여 API 엔드포인트에 대한 포괄적인 정보를 제공합니다.

콘솔을 사용하여 메시지를 수신 및 삭제하려면

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 큐 페이지에서 큐를 선택한 다음 메시지 보내기 및 받기를 선택합니다.



4. 메시지 보내기 및 받기 페이지에서 메시지 설문 조사를 선택합니다.

   Amazon SQS가 대기열에서 메시지를 폴링하기 시작합니다. 메시지 수신 섹션 오른쪽에 있는 진행률 표시줄에 폴링 기간이 표시됩니다.

   메시지 섹션에는 수신된 메시지 목록이 표시됩니다. 목록에는 각 메시지에 대한 메시지 ID, 전송 날짜, 크기 및 수신 개수가 표시됩니다.
5. 메시지를 삭제하려면 삭제하려는 메시지를 선택한 다음 삭제를 선택합니다.
6. [메시지 삭제] 대화 상자에서 [삭제] 를 선택합니다.

Amazon SQS 대기열이 비어 있는지 확인

대부분의 경우 긴 폴링을 사용하여 대기열이 비어 있는지 확인할 수 있습니다. 드문 경우지만, 대기열에 여전히 메시지가 있는 경우에도 빈 응답을 받을 수 있습니다. 특히 큐를 만들 때 메시지 수신 대기 시간 값을 낮게 지정한 경우 더욱 그렇습니다. 이 섹션에서는 대기열이 비어 있는지 확인하는 방법을 설명합니다.

대기열이 비어 있는지 확인하는 방법(콘솔)

1. 모든 생산자가 메시지를 보내지 못하게 합니다.
2. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
3. 탐색 창에서 대기열을 선택합니다.
4. 대기열 페이지에서 대기열을 선택합니다.
5. 모니터링 탭을 선택합니다.
6. 모니터링 대시보드의 오른쪽 상단에서 새로 고침 기호 옆에 있는 아래쪽 화살표를 선택합니다. 드롭다운 메뉴에서 자동 새로 고침을 선택합니다. 새로 고침 간격을 1분으로 유지합니다.
7. 다음 대시보드를 살펴봅니다.

   - 지연된 메시지의 대략적인 수
   - 표시되지 않은 메시지의 대략적인 수
   - 표시된 메시지의 대략적인 수

   몇 분 동안 모든 항목의 값이 0으로 표시되면 대기열이 비어 있는 것입니다.

대기열이 비어 있는지 확인하려면 (AWS CLI, AWS API)

1. 모든 생산자가 메시지를 보내지 못하게 합니다.
2. 다음 명령 중 하나를 반복적으로 실행합니다.
   - AWS CLI: `get-queue-attributes`
   - AWS API: `GetQueueAttributes`
3. 지표를 관찰하여 다음 속성이 있는지 확인합니다.
   - `ApproximateNumberOfMessagesDelayed`
   - `ApproximateNumberOfMessagesNotVisible`
   - `ApproximateNumberOfMessagesVisible`

   몇 분 동안 모든 항목이 `0`으로 표시되면 대기열이 비어 있는 것입니다.

Amazon CloudWatch 메트릭을 사용하는 경우 대기열이 비어 있는 것으로 간주하기 전에 연속된 0 데이터 포인트가 여러 개 있는지 확인하십시오. CloudWatch 지표에 대한 자세한 내용은 을 참조하십시오Amazon SQS에서 사용할 수 있는 CloudWatch 지표.

Amazon SQS 대기열 삭제

Amazon SQS 대기열을 더 이상 사용하지 않고 가까운 시일 내에 사용할 계획이 없다면 삭제하는 것이 좋습니다.

> **Tip**
> 대기열을 삭제하기 전에 대기열이 비어 있는지 확인하려면 Amazon SQS 대기열이 비어 있는지 확인 섹션을 참조하세요.

대기열은 비어있지 않은 경우에도 삭제할 수 있습니다. 대기열의 메시지는 삭제하지만 대기열 자체는 삭제하지 않을 경우 대기열을 제거합니다.

대기열을 삭제하려면(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.

3. 대기열 페이지에서 삭제할 대기열을 선택합니다.
4. 삭제를 선택합니다.
5. 대기열 삭제 대화 상자에 **delete**를 입력하여 삭제를 확인합니다.
6. 삭제를 선택합니다.

대기열 (AWS CLI 및 API) 을 삭제하려면

다음 명령 중 하나를 사용하여 대기열을 삭제할 수 있습니다.

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

Amazon SQS 콘솔을 사용하여 대기열에서 메시지 삭제

Amazon SQS 대기열을 삭제하지 않고 이 대기열에서 모든 메시지를 삭제할 경우, 이 대기열을 제거합니다. 메시지 삭제 작업은 최대 60초 걸립니다. 대기열 크기와 상관없이 60초 정도 기다리는 것이 좋습니다.

> **Important**
> 대기열을 제거할 경우 삭제된 메시지를 검색할 수 없습니다.

대기열을 제거하려면(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열 페이지에서 제거할 대기열을 선택합니다.
4. 작업에서 제거를 선택합니다.
5. 대기열 제거 대화 상자에서 **purge**라고 입력하고 제거를 선택하여 제거를 확인합니다.

   대기열에서 모든 메시지가 제거됩니다. 콘솔에 확인 배너가 표시됩니다.

Amazon SQS 시작을 위한 일반적인 작업

이제 대기열을 생성했고 메시지 전송, 수신 및 삭제 방법과 대기열 삭제 방법을 알았으므로, 다음 작업을 실시해야 합니다.

- Lambda 함수를 트리거하려면 함수를 트리거하도록 Amazon SQS 대기열 구성 AWS Lambda 단원을 참조하십시오.
- SSE 및 기타 기능을 포함하여 대기열을 구성하는 방법을 알아봅니다.
- 속성과 함께 메시지를 전송하는 방법을 알아봅니다.
- VPC에서 메시지를 전송하는 방법을 알아봅니다.
- Amazon SQS의 기능 및 아키텍처에 대해 알아보려면 Amazon SQS 대기열 유형 및 Amazon SQS 기본 아키텍처 섹션을 참조합니다.
- Amazon SQS를 최대한 활용하는 데 도움이 될 지침 및 경고에 대해 알아보려면 Amazon SQS의 모범 사례 섹션을 참조합니다.
- 개발자 안내서와 같은 AWS SDK 중 하나에 대한 Amazon SQS 예제를 살펴보십시오.AWS SDK for Java 2.x
- Amazon SQS AWS CLI 명령에 대해 자세히 알아보려면 명령 참조를 참조하십시오.AWS CLI
- Amazon SQS 작업에 대해 자세히 알아보려면 Amazon Simple Queue Service API 참조를 참조합니다.
- 프로그래밍 방식으로 Amazon SQS와 상호 작용하는 방법을 알아보십시오. API 사용을 읽고 개발 센터에 대해 알아보십시오.AWS
  - Java
  - JavaScript
  - PHP
  - Python
  - Ruby
  - Windows 및 .NET
- Amazon SQS에서의 문제 해결 단원에서 비용과 리소스를 모니터링하는 방법에 대해 알아봅니다.
- 보안 단원에서 데이터 및 데이터 액세스를 보호하는 방법에 대해 알아봅니다.
- Amazon SQS 워크플로에 대한 자세한 내용은 Amazon SQS 액세스 제어 프로세스 워크플로 섹션을 참조하십시오.

Amazon SQS 표준 대기열 시작하기

Amazon SQS에서는 표준을 기본 대기열 유형으로 제공합니다. 표준 대기열은 API 작업(`SendMessage`, `ReceiveMessage` 또는 `DeleteMessage`)별 초당 거의 무제한의 API 호출 수를 지원합니다. 표준 대기열은 메시지 전달을 지원합니다. at-least-once 하지만 이따금(거의 무제한의 처리량을 지원하는 고도로 분산된 아키텍처로 인해) 두 개 이상의 메시지 사본이 순서대로 전송되지 않을 수 있습니다. 표준 대기열은 완벽하지는 않지만 최선의 정렬을 통해 일반적으로 메시지가 전송한 순서와 동일한 순서로 전송되도록 보장합니다.

Amazon SQS는 `SendMessage`가 승인되기 전에 둘 이상의 가용 영역(AZ)에 메시지를 중복 저장합니다. 메시지 사본은 여러 AZ에 저장되므로 단일 컴퓨터, 네트워크 또는 AZ 장애로 인해 메시지에 액세스할 수 없게 되는 상황은 발생하지 않습니다.

Amazon SQS 콘솔을 사용하여 대기열을 생성 및 구성하는 방법에 대한 자세한 내용은 Amazon SQS 콘솔을 사용하여 대기열을 생성합니다. 섹션을 참조하세요. Java 예제는 Amazon SQS Java SDK 예제 섹션을 참조하세요.

다양한 시나리오에서 표준 메시지 대기열을 사용할 수 있으며, 이는 애플리케이션에서 두 번 이상 순서에 맞지 않게 도착한 메시지를 처리하는 경우에 한합니다. 그 예는 다음과 같습니다.

- 실시간 사용자 요청을 폭넓은 배경 작업과 분리 - 미디어 크기를 조정하거나 인코딩하는 동안 미디어를 업로드할 수 있습니다.
- 작업을 여러 작업자 노드에 할당 - 대량의 신용카드 확인 요청을 처리합니다.
- 이후의 처리를 위해 메시지를 배치 처리 - 다수의 항목이 데이터베이스에 추가되도록 예약합니다.

표준 대기열 관련 할당량은 할당량 섹션을 참조하세요.

표준 대기열 작업에 대한 모범 사례는 Amazon SQS 표준 및 FIFO 대기열에 대한 권장 사항 섹션을 참조하세요.

메시지 정렬

표준 대기열을 통해 메시지 순서를 최대한 보존할 수 있지만 두 개 이상의 메시지 사본이 순서대로 전송되지 않을 수 있습니다. 시스템에서 순서를 보존해야 하는 경우 수신 시 메시지를 재정렬할 수 있도록 FIFO(First-In-First-Out) 대기열을 사용하거나 각 메시지에 순서 정보를 추가하는 것이 좋습니다.

하나 이상의 전송

Amazon SQS는 중복성과 고가용성을 위해 여러 대의 서버에 메시지 사본을 저장합니다. 드물게는 메시지 사본을 받거나 삭제할 때 메시지 사본을 저장하는 서버 중 하나를 사용할 수 없을 수도 있습니다.

이 경우 사용할 수 없는 서버에서는 메시지 사본이 삭제되지 않으므로 메시지를 받을 때 해당 메시지 복사본을 다시 가져올 수 있습니다. 따라서 애플리케이션이 idempotent가 되도록 설계해야 합니다(다시 말해 동일한 메시지를 두 번 이상 처리할 경우 부정적인 영향을 받지 않아야 함).

Amazon SQS 대기열 및 메시지 식별자

이 섹션에서는 표준 및 FIFO 대기열의 식별자에 대해 설명합니다. 이러한 식별자는 특정 대기열과 메시지를 찾고 조작하는 데 도움이 될 수 있습니다.

Amazon SQS 표준 대기열의 식별자

다음 식별자에 대한 자세한 내용은 Amazon Simple Queue Service API 참조를 참조하세요.

대기열 이름 및 URL

새 대기열을 생성할 때는 AWS 계정 및 리전에 고유한 대기열 이름을 지정해야 합니다. Amazon SQS는 사용자가 생성한 각 대기열에 대기열 이름과 기타 Amazon SQS 구성 요소가 포함된 대기열 URL이라는 식별자를 할당합니다. 대기열에서 작업을 실시할 때마다 대기열 URL을 지정합니다.

다음은 AWS 계정 번호 123456789012를 갖는 사용자가 소유하는 MyQueue라는 이름의 대기열의 대기열 URL입니다.

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

대기열을 나열하고 계정 번호 뒤에 오는 문자열을 구문 분석하여 프로그래밍 방식으로 대기열의 URL을 검색할 수 있습니다. 자세한 정보는 `ListQueues`을 참조하세요.

메시지 ID

각 메시지는 Amazon SQS가 `SendMessage` 응답으로 반환하는 시스템 할당 메시지 ID를 수신합니다. 이 식별자는 메시지를 식별하는 데 유용합니다. 메시지 ID의 최대 길이는 100자입니다.

수신 핸들

대기열에서 메시지를 수신할 때마다 해당 메시지의 수신 핸들을 수신합니다. 이 핸들은 메시지 자체가 아닌 메시지 수신 작업과 관련되어 있습니다. 메시지를 삭제하거나 메시지 표시를 변경하려면, (메시지

ID가 아닌) 수신 핸들을 지정해야 합니다. 또한, 메시지를 수신한 후 언제든지 삭제할 수 있습니다(메시지를 대기열에 배치한 후 회수할 수 없음). 수신 핸들의 최대 길이는 1024자입니다.

> **Important**
>
> 메시지를 두 번 이상 받으면, 받을 때마다 다른 수신 핸들을 받습니다. 메시지 삭제를 요청할 때에는 가장 최근에 받은 수신 핸들을 제공해야 합니다(그렇지 않으면 메시지가 삭제될 수 있음).

다음은 수신 핸들 예제입니다(세 줄로 나뉨).

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

할당량

다음 표에는 표준 대기열과 관련된 할당량이 나열되어 있습니다.

할당량	설명
지연 대기열	대기열의 기본(최소) 지연 시간은 0초입니다. 최대값은 15분입니다.
대기열 나열	`ListQueues` 요청당 대기열 1,000개
긴 폴링 대기 시간	긴 폴링 대기 시간의 최대값은 20초입니다.
대기열당 메시지(백로그)	Amazon SQS 대기열이 저장할 수 있는 메시지 수에는 제한이 없습니다.
대기열당 메시지(이동 중)	대부분의 표준 대기열(대기열 트래픽 및 메시지 백로그에 따라 다름)의 경우 최대 약 120,000개의 이동 중 메시지(소비자가 대기열에서 수신했지만 대기열에서 아직 삭제되지 않은 메시지)가 있을 수 있습니다. 짧은 폴링을 사용하는 동안 이 할당량에 도달하면 Amazon SQS에서 `OverLimit` 오류 메시지를 반환합니다. 긴 폴링을 사용하는 경우에는 Amazon SQS에서 오류 메시지를 반환하지

할당량	설명
	않습니다. 이 할당량에 도달하지 않도록 하려면 메시지를 처리한 후 대기열에서 삭제해야 합니다. 또한 메시지를 처리하는 데 사용하는 대기열의 수를 늘릴 수 있습니다. 할당량 증가를 요청하려면 지원 요청을 제출하십시오.
대기열 이름	대기열 이름은 최대 80자까지 가능합니다. 허용되는 문자는 영숫자, 하이픈(-), 밑줄(_)입니다. **Note** 대기열 이름은 대/소문자를 구분합니다(예: `Test-queue` 와 `test-queue` 는 다른 대기열임).
대기열 태그	대기열에 50개 이상의 태그를 추가하지 않는 것이 좋습니다. 태그 지정 시 UTF-8 형식의 유니코드 문자를 사용할 수 있습니다.
	`Key` 태그는 필수지만 `Value` 태그는 선택 사항입니다.
	`Key` 태그 및 `Value` 태그는 대소문자를 구분합니다.
	`Key` 태그 및 `Value` 태그에는 UTF-8 형식의 유니코드 알파벳 문자와 공백이 포함될 수 있습니다. 다음 특수 문자가 허용됩니다. `_ . : / = + - @`
	`Key` 태그 또는 `Value` 태그에 예약된 접두사 `aws:`가 포함되며 안 됩니다(이 접두사가 있는 태그 키 또는 값은 삭제할 수 없음).
	`Key` 태그의 최대 길이는 UTF-8 형식의 유니코드 문자 128자입니다. `Key` 태그는 비어 있거나 null이면 안 됩니다.
	`Value` 태그의 최대 길이는 UTF-8 형식의 유니코드 문자 256자입니다. `Value` 태그는 비어 있거나 null일 수 있습니다.

할당량	설명
	태깅 작업은 1회당 30TPS로 제한됩니다. AWS 계정애플리케이션에 더 많은 처리량이 필요한 경우 요청을 제출하십시오.

Amazon SQS에서 FIFO 대기열을 사용하기 시작하기

FIFO(First-In-First-Out) 대기열은 표준 대기열의 모든 기능을 갖추고 있지만 작업 및 이벤트의 순서가 중요하거나 중복이 허용되지 않을 때 애플리케이션 간의 메시징이 향상되도록 설계되었습니다.

FIFO 대기열을 사용할 수 있는 상황의 예는 다음과 같습니다.

1. 주문이 중요한 전자 상거래 주문 관리 시스템
2. 이벤트를 순서대로 처리해야 하는 타사 시스템과의 통합
3. 사용자가 입력한 내용을 입력한 순서대로 처리
4. 통신 및 네트워킹 - 데이터와 정보를 동일한 순서로 전송 및 수신
5. 컴퓨터 시스템 - 사용자가 입력한 명령이 올바른 순서로 실행되도록 보장
6. 교육 기관 - 학생이 계정 등록 전에 과정에 등록하지 못하도록 차단
7. 온라인 티켓팅 시스템 - 티켓이 선착순으로 배포됨

> **Note**
>
> 또한 FIFO 대기열은 정확히 1회 처리를 제공하지만 초당 트랜잭션(TPS) 수가 제한적입니다. FIFO 대기열에서 Amazon SQS 높은 처리량 모드를 사용하면 트랜잭션 한도를 늘릴 수 있습니다. 높은 처리량 모드 사용에 대한 자세한 내용은 Amazon SQS의 FIFO 대기열에 대한 높은 처리량 섹션을 참조하세요. 처리량 할당량에 대한 자세한 내용은 the section called “메시지 할당량” 섹션을 참조하세요.

Amazon SQS를 사용할 수 있는 모든 리전에서 Amazon SQS FIFO 대기열을 사용할 수 있습니다.

순서가 복잡한 FIFO 대기열 사용에 대한 자세한 내용은 Solving Complex Ordering Challenges with Amazon SQS FIFO Queues 항목을 참조하세요.

Amazon SQS 콘솔을 사용하여 대기열을 생성 및 구성하는 방법에 대한 자세한 내용은 Amazon SQS 콘솔을 사용하여 대기열을 생성합니다. 섹션을 참조하세요. Java 예제는 Amazon SQS Java SDK 예제 섹션을 참조하세요.

FIFO 대기열 작업에 대한 모범 사례는 Amazon SQS FIFO 대기열에 대한 추가 권장 사항 및 Amazon SQS 표준 및 FIFO 대기열에 대한 권장 사항 섹션을 참조하세요.

Amazon SQS의 FIFO 대기열 전송 로직

다음 개념은 FIFO와의 메시지 교환을 보다 정확하게 이해하는 데 유용합니다.

메시지 전송

여러 메시지를 FIFO 대기열에 연속으로 전송할 때 각 대기열에 고유한 메시지 중복 제거 ID가 있는 경우 Amazon SQS는 메시지를 저장하고 전송을 확인합니다. 그리고 나면, 각 메시지는 메시지가 전송된 순서와 정확히 같은 순서로 수신 및 처리될 수 있습니다.

FIFO 대기열에서 메시지는 메시지 그룹 ID를 기준으로 정렬됩니다. 여러 호스트(또는 동일한 호스트에 있는 다른 스레드)가 메시지 그룹 ID가 동일한 여러 메시지를 FIFO 대기열에 전송하는 경우 Amazon SQS는 도착하는 순서대로 메시지를 저장한 후 처리합니다. Amazon SQS가 메시지 전송 및 수신 순서를 유지하려면 각 생산자가 고유한 메시지 그룹 ID를 사용하여 모든 메시지를 전송해야 합니다.

FIFO 대기열 로직은 메시지 그룹 ID별로만 적용됩니다. 각 메시지 그룹 ID는 Amazon SQS 대기열에서 고유하게 정렬된 메시지 그룹을 나타냅니다. 각각의 메시지 그룹 ID의 경우, 모든 메시지가 엄격한 순서로 전송되고 수신됩니다. 그렇지만, 메시지 그룹 ID 값이 다른 메시지는 다른 순서로 전송 및 수신될 수 있습니다. 메시지 그룹 ID를 메시지와 연결해야 합니다. 사용자가 메시지 그룹 ID를 지정하지 않으면 작업에 실패합니다. 단일 그룹의 정렬된 메시지가 필요한 경우 FIFO 대기열로 전송된 메시지에 동일한 메시지 그룹 ID를 지정합니다.

메시지 수신

특정 메시지 그룹 ID를 사용하는 메시지를 수신하도록 요청할 수 없습니다.

메시지 그룹 ID가 여러 개 있는 FIFO 대기열에서 메시지를 수신할 경우 Amazon SQS는 먼저 메시지 그룹 ID가 동일한 메시지를 가능한 한 많이 반환하려고 시도합니다. 따라서 다른 소비자들은 메시지 그룹 ID가 다른 메시지를 처리할 수 있습니다. 메시지 그룹 ID가 있는 메시지를 수신한 경우, 동일한 메시지 그룹 ID에 대한 메시지는 그 메시지를 삭제하거나 메시지가 표시되는 경우가 아니라면 더 이상 반환되지 않습니다.

> **Note**
>
> `MaxNumberOfMessages` 작업의 `ReceiveMessage` 요청 파라미터를 사용하여 단일 호출로 최대 10개 메시지를 받을 수 있습니다. 이러한 메시지는 FIFO 순서를 유지하고 동일한 메시지 그룹 ID를 사용할 수 있습니다. 따라서 동일한 메시지 그룹 ID를 사용할 수 있는 메시지가 10개 미만인 경우, 다른 메시지 그룹 ID로부터 메시지를 받을 수 있으며 동일한 10개 메시지 배치에서 여전히 FIFO 순서를 유지합니다.

다수의 재시도 횟수

FIFO 대기열에서는 다음과 같이 생산자 또는 소비자가 여러 번 재시도할 수 있도록 허용합니다.

- 생성자가 실패한 SendMessage 작업을 감지한 경우, 동일한 메시지 중복 제거 ID를 사용하여 필요한 횟수 만큼 여러 번 재전송을 시도할 수 있습니다. 생성자가 중복 제거 간격이 만료되기 전에 최소 한 개의 승인을 수신한다고 가정하면 여러 번 재시도를 해도 메시지의 순서에 영향을 미치지도 복제를 도입하지도 않습니다.
- 소비자가 실패한 ReceiveMessage 작업을 감지한 경우, 동일한 수신 요청 시도 ID를 사용하여 필요한 횟수 만큼 여러 번 재시도할 수 있습니다. 소비자가 제한 시간 초과가 만료되기 전에 최소 한 개의 승인을 수신한다고 가정하면 여러 번 재시도를 해도 메시지의 순서에는 영향을 미치지 않습니다.
- 메시지 그룹 ID가 있는 메시지를 수신한 경우, 동일한 메시지 그룹 ID에 대한 메시지는 그 메시지를 삭제하거나 메시지가 표시되는 경우가 아니라면 더 이상 반환되지 않습니다.

아마존 SQS에서의 FIFO 큐 메시지 주문

FIFO 대기열은 표준 대기열을 개선하고 보완합니다. 이 대기열 유형의 가장 중요한 기능은 FIFO(First-In-First-Out) 전송과 정확히 1회 처리입니다.

- 메시지를 보내고 받는 순서는 엄격하게 보존되며 메시지는 한 번 전달되며 소비자가 처리하고 삭제할 때까지 사용할 수 없습니다.
- 중복 항목을 대기열에 삽입하지 않습니다.

또한 FIFO 대기열은 단일 대기열에 정렬된 여러 메시지 그룹을 허용하는 메시지 그룹을 지원합니다. FIFO 대기열 내 메시지 그룹의 수에는 할당량이 없습니다.

Amazon SQS에서 정확히 한 번의 프로세싱

표준 대기열과 달리 FIFO 대기열에는 중복 메시지가 추가되지 않습니다. FIFO 대기열을 사용하면 대기열에 중복 메시지가 전송되는 것을 방지할 수 있습니다. 5분 중복 제거 간격 내에 SendMessage 작업을 재시도하면, Amazon SQS가 대기열에 중복 항목을 추가하지 않습니다.

중복 제거를 구성하려면 다음 중 하나를 실시해야 합니다.

- 콘텐츠 기반 중복 제거 활성화. 이 작업을 통해 Amazon SQS는 SHA-256 해시를 사용한 후 메시지 본문을 사용하여 메시지 중복 제거 ID를 생성하지만 메시지 속성은 생성하지 않습니다. 자세한

내용은 Amazon Simple Queue Service API 참조에서 `CreateQueue`, `GetQueueAttributes` 및 `SetQueueAttributes` 작업에 대한 설명서를 참조하세요.

- 메시지에 메시지 중복 제거 ID 값을 명시적으로 지정하거나 시퀀스 번호를 표시합니다. 자세한 내용은 Amazon Simple Queue Service API 참조에서 `SendMessage`, `SendMessageBatch` 및 `ReceiveMessage` 작업에 대한 설명서를 참조하세요.

Amazon SQS의 표준 대기열에서 FIFO 대기열로 이동

표준 대기열을 사용한 기존 애플리케이션을 사용하고 있으면서 FIFO 대기열의 정렬 또는 정확히 1회 처리 기능을 활용하려는 경우 대기열과 애플리케이션을 올바르게 구성해야 합니다.

> **Note**
>
> 기존 표준 대기열을 FIFO 대기열로 변환할 수는 없습니다. 이동하려면 애플리케이션의 새 FIFO 대기열을 생성하거나 기존의 표준 대기열을 삭제하고 FIFO 대기열로 다시 생성해야 합니다.

애플리케이션이 FIFO 대기열에서 올바르게 작동하는지 확인하려면 다음 체크리스트를 사용합니다.

- 처리량을 높이려면 FIFO에 권장되는 높은 처리량 모드를 사용합니다. 메시징 할당량에 대해 자세히 알아보려면 Amazon SQS 메시지 할당량 섹션을 참조하세요.
- FIFO 대기열은 메시지당 지연을 지원하지 않고 대기열당 지연만 지원합니다. 애플리케이션에서 각 메시지마다 동일한 `DelaySeconds` 파라미터 값을 설정하는 경우, 대신 애플리케이션을 수정하여 메시지당 지연을 제거하고 전체 대기열에서 `DelaySeconds`를 설정해야 합니다.
- 메시지 그룹은 고객이 각각의 순서를 유지하면서 메시지를 병렬로 처리할 수 있도록 해주는 고유한 FIFO 기능입니다. 고객은 메시지 그룹 ID를 지정하여 메시지를 메시지 그룹으로 구성합니다. 메시지 그룹은 지정된 워크로드에 대한 비즈니스 차원을 기반으로 하는 경우가 많습니다. FIFO 대기열을 통해 더 효과적으로 확장하려면 메시지 ID에 더 세분화된 비즈니스 차원을 사용하세요. 메시지를 배포하는 메시지 그룹 ID가 많을수록 FIFO에서 사용할 수 있는 메시지 수도 많아집니다.
- FIFO 대기열로 메시지를 전송하기 전에 다음 사항을 확인하세요.
  - 애플리케이션에서 본문이 동일한 메시지를 전송할 수 있는 경우, 애플리케이션을 수정하여 전송한 각 메시지별로 고유한 메시지 중복 제거 ID를 지정합니다.
  - 애플리케이션에서 고유한 본문 메시지를 전송하는 경우, 콘텐츠 기반 중복 제거를 활성화할 수 있습니다.

- 소비자에게 맞게 코드를 변경하지 않아도 됩니다. 그렇지만, 메시지를 처리하는 데 오랜 시간이 걸리고 제한 시간 초과를 높은 값으로 설정하는 경우, 각 `ReceiveMessage` 작업에 수신 요청 시도 ID를 추가하는 것이 좋습니다. 그러면 네트워킹 오류가 발생한 경우에도 여러 번 수신 시도를 할 수 있고 실패한 수신 시도로 인해 대기열이 일시 중지되지 않습니다.

자세한 내용은 Amazon Simple Queue Service API 참조를 참조하세요.

Amazon SQS의 FIFO 대기열에 대한 높은 처리량

Amazon SQS의 높은 처리량 FIFO 대기열은 엄격한 메시지 순서를 유지하면서 높은 메시지 처리량을 효율적으로 관리하여 수많은 메시지를 처리하는 애플리케이션의 안정성과 확장성을 보장합니다. 이 솔루션은 높은 처리량과 순차적인 메시지 전송을 모두 요구하는 시나리오에 적합합니다.

엄격한 메시지 순서가 중요하지 않고 수신 메시지의 양이 비교적 적거나 산발적인 시나리오에서는 Amazon SQS 처리량이 높은 FIFO 대기열이 필요하지 않습니다. 예를 들어, 메시지를 자주 처리하지 않거나 비순차적인 메시지를 처리하는 소규모 애플리케이션을 사용하는 경우 처리량이 높은 FIFO 대기열과 관련된 복잡성과 비용이 추가되는 것은 정당화되지 않을 수 있습니다. 또한 애플리케이션에 높은 처리량의 FIFO 대기열이 제공하는 향상된 처리량 기능이 필요하지 않은 경우 표준 Amazon SQS 대기열을 선택하는 것이 더 비용 효율적이고 관리가 간단할 수 있습니다.

처리량이 많은 FIFO 대기열의 요청 용량을 향상시키려면 메시지 그룹 수를 늘리는 것이 좋습니다. 처리량이 많은 메시지 할당량에 대한 자세한 내용은 Amazon Web Services 일반 참조의 Amazon SQS 서비스 할당량을 참조하세요.

대기열별 할당량 및 데이터 배포 전략에 대한 자세한 내용은 및 을 참조하십시오. Amazon SQS 메시지 할당량 SQS FIFO 대기열의 높은 처리량을 위한 파티션 및 데이터 배포

주제

- Amazon SQS FIFO 대기열의 높은 처리량을 위한 사용 사례
- SQS FIFO 대기열의 높은 처리량을 위한 파티션 및 데이터 배포
- Amazon SQS에서 FIFO 대기열의 높은 처리량을 활성화할 수 있습니다.

Amazon SQS FIFO 대기열의 높은 처리량을 위한 사용 사례

다음 사용 사례는 처리량이 높은 FIFO 대기열의 다양한 애플리케이션을 강조하여 산업 및 시나리오 전반에 걸친 효율성을 보여줍니다.

1. 실시간 데이터 처리: 이벤트 처리 또는 원격 측정 데이터 수집과 같은 실시간 데이터 스트림을 처리하는 애플리케이션은 처리량이 높은 FIFO 대기열을 활용하여 정확한 분석을 위해 순서를 유지하면서 지속적으로 유입되는 메시지를 처리할 수 있습니다.
2. 전자 상거래 주문 처리: 고객 거래의 순서를 유지하는 것이 중요한 전자 상거래 플랫폼에서 처리량이 높은 FIFO 대기열을 사용하면 쇼핑 성수기에도 주문이 지연 없이 순차적으로 처리됩니다.
3. 금융 서비스: 빈도가 높은 거래 또는 거래 데이터를 처리하는 금융 기관은 메시지 주문에 대한 엄격한 규제 요구 사항을 준수하면서 최소의 지연 시간으로 시장 데이터와 거래를 처리하기 위해 높은 처리량의 FIFO 대기열을 사용합니다.
4. 미디어 스트리밍: 스트리밍 플랫폼 및 미디어 배포 서비스는 처리량이 높은 FIFO 대기열을 활용하여 미디어 파일 및 스트리밍 콘텐츠의 전송을 관리하여 올바른 콘텐츠 전송 순서를 유지하면서 사용자의 원활한 재생 환경을 보장합니다.

SQS FIFO 대기열의 높은 처리량을 위한 파티션 및 데이터 배포

Amazon SQS는 FIFO 대기열 데이터를 파티션에 저장합니다. 파티션은 대기열에 대한 스토리지 할당으로, 지역 내 여러 가용 영역에 자동으로 복제됩니다. AWS 사용자는 파티션을 관리하지 않습니다. 대신 Amazon SQS가 파티션 관리를 처리합니다.

FIFO 대기열의 경우 Amazon SQS는 다음과 같은 상황에서 대기열의 파티션 수를 수정합니다.

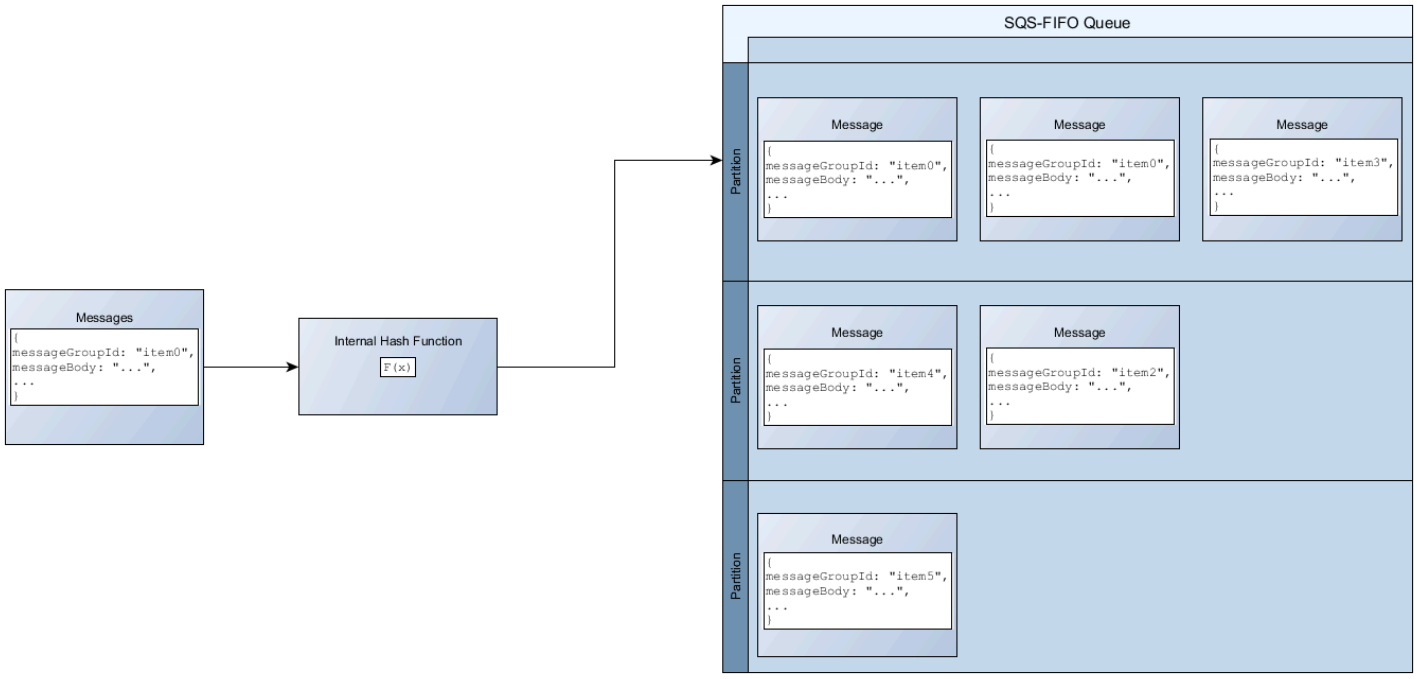
- 현재 요청 속도가 기존 파티션이 지원할 수 있는 속도에 근접하거나 초과하면 대기열이 리전별 할당량에 도달할 때까지 추가 파티션이 할당됩니다. 할당량에 대한 자세한 내용은 Amazon SQS 메시지 할당량 섹션을 참조하세요.
- 현재 파티션의 사용률이 낮으면 파티션 수가 줄어들 수 있습니다.

파티션 관리는 백그라운드에서 자동으로 이루어지므로 애플리케이션에는 표시되지 않습니다. 대기열과 메시지를 항상 사용할 수 있습니다.

메시지 그룹 ID별로 데이터 배포

FIFO 대기열에 메시지를 추가하기 위해 Amazon SQS는 각 메시지의 메시지 그룹 ID 값을 내부 해시 함수의 입력으로 사용합니다. 해시 함수 출력 값은 메시지를 저장할 파티션을 결정합니다.

다음 다이어그램은 여러 파티션에 걸쳐 있는 대기열을 보여줍니다. 대기열의 메시지 그룹 ID는 항목 번호를 기반으로 합니다. Amazon SQS는 해시 함수를 사용하여 새 항목을 저장할 위치를 결정합니다. 이 경우 문자열 `item0`의 해시 값이 기준으로 사용됩니다. 항목은 대기열에 추가되는 것과 동일한 순서로 저장된다는 점에 유의하세요. 각 항목의 위치는 메시지 그룹 ID의 해시 값으로 결정됩니다.



> **Note**
>
> Amazon SQS는 파티션 수에 관계없이 FIFO 대기열의 파티션 전체에 항목을 균일하게 배포하도록 최적화되었습니다. AWS 고유 값이 많을 수 있는 메시지 그룹 ID를 사용할 것을 권장합니다.

파티션 사용률 최적화

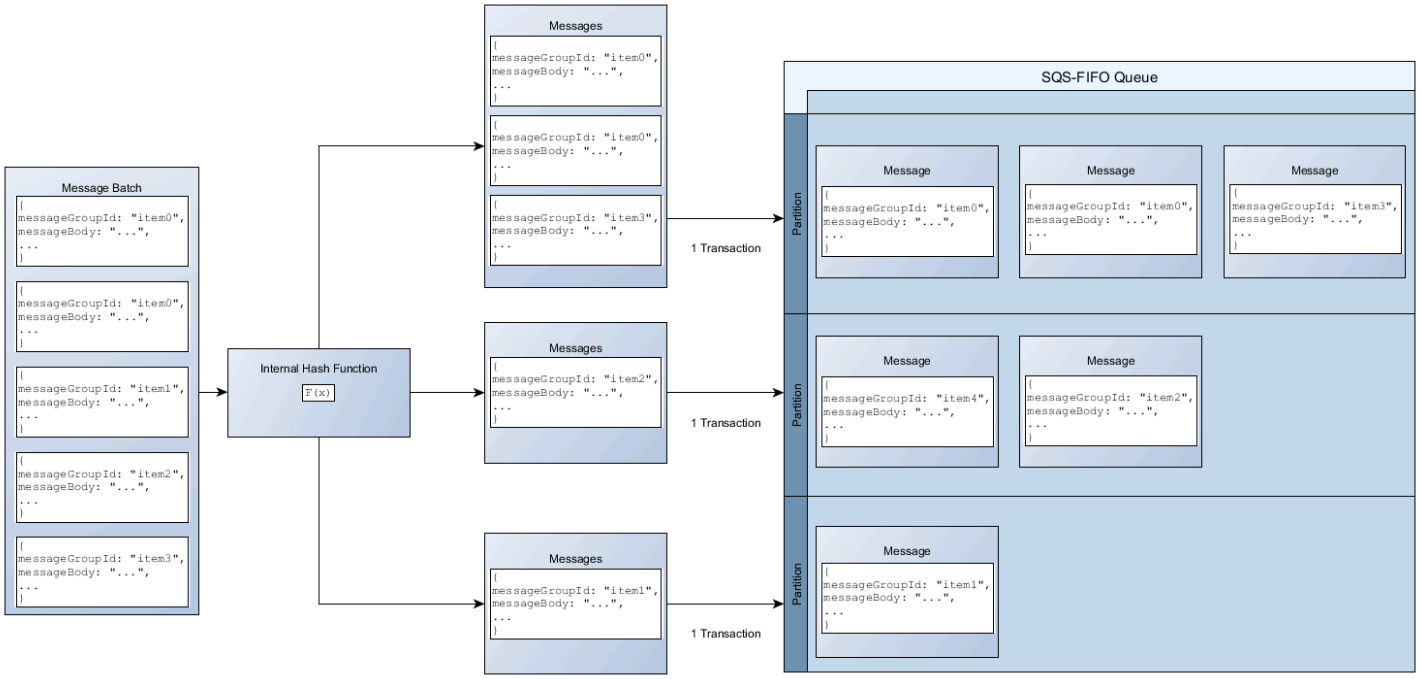
각 파티션은 배치 처리 시 초당 최대 3,000개의 메시지를 지원하거나 지원되는 리전에서 전송, 수신 및 삭제 작업의 경우 초당 최대 300개의 메시지를 지원합니다. 처리량이 많은 메시지 할당량에 대한 자세한 내용은 Amazon Web Services 일반 참조의 Amazon SQS 서비스 할당량을 참조하세요.

배치 API를 사용하는 경우 각 메시지는 메시지 그룹 ID별로 데이터 배포에 설명된 프로세스에 따라 라우팅됩니다. 동일한 파티션으로 라우팅되는 메시지는 그룹화되어 단일 트랜잭션으로 처리됩니다.

`SendMessageBatch`API의 파티션 사용률을 최적화하려면 가능하면 메시지 그룹 ID가 동일한 메시지를 일괄 처리하는 것이 좋습니다. AWS

`DeleteMessageBatch`및 `ChangeMessageVisibilityBatch` API의 파티션 사용률을 최적화하려면 `MaxNumberOfMessages` 파라미터가 10으로 설정된 `ReceiveMessage` 요청을 사용하고 단일 요청에서 반환된 수신 핸들을 일괄 처리하는 것이 좋습니다. AWS `ReceiveMessage`

다음 예에서는 다양한 메시지 그룹 ID를 가진 메시지 배치 처리가 전송됩니다. 배치 처리는 세 그룹으로 나뉘며, 각 그룹은 파티션의 할당량에 포함됩니다.



> **Note**
>
> Amazon SQS는 동일한 메시지 그룹 ID의 내부 해시 함수를 가진 메시지가 배치 요청 내에서만 그룹화되도록 보장합니다. 내부 해시 함수의 출력과 파티션 수에 따라 메시지 그룹 ID가 다른 메시지가 그룹화될 수 있습니다. 해시 함수 또는 파티션 수는 언제든지 변경될 수 있으므로 한 지점에서 그룹화된 메시지가 나중에 그룹화되지 않을 수 있습니다.

Amazon SQS에서 FIFO 대기열의 높은 처리량을 활성화할 수 있습니다.

신규 또는 기존 FIFO 대기열에 높은 처리량을 활성화할 수 있습니다. 이 기능에는 FIFO 대기열을 만들고 편집할 때 사용할 수 있는 새로운 옵션 세 가지가 포함되어 있습니다.

- 높은 처리량 FIFO 활성화 - 현재 FIFO 대기열의 메시지에 높은 처리량을 사용합니다.
- 중복 제거 범위 - 중복 제거를 대기열 또는 메시지 그룹 수준에서 수행할지를 지정합니다.
- FIFO 처리량 한도 - FIFO 대기열에 있는 메시지의 처리량 할당량을 대기열 또는 메시지 그룹 수준에서 설정할지를 지정합니다.

FIFO 대기열의 높은 처리량을 활성화하려면(콘솔)

1. FIFO 대기열 생성 또는 편집을 시작합니다.
2. 대기열 옵션을 지정할 때는 높은 처리량 FIFO 활성화를 선택합니다.

   FIFO 대기열에 높은 처리량을 활성화하면 관련 옵션이 다음과 같이 설정됩니다.

   - 중복 제거 범위가 메시지 그룹으로 설정되어 있으며, 이는 FIFO 대기열에 높은 처리량을 사용하는 데 필요한 설정입니다.
   - FIFO 처리량 한도가 메시지 그룹 ID당으로 설정되어 있으며, 이는 FIFO 대기열에 높은 처리량을 사용하는 데 필요한 설정입니다.

   FIFO 대기열에 높은 처리량을 사용하는 데 필요한 설정을 변경하면 대기열에 일반 처리량이 적용되고 지정된 대로 중복 제거가 수행됩니다.
3. 대기열의 모든 옵션을 계속 지정합니다. 지정을 완료하면 대기열 생성 또는 저장을 선택합니다.

FIFO 대기열을 만들거나 편집한 후에는 이 대기열에 메시지를 보내고 메시지를 수신 및 삭제할 수 있으며, 이 모든 작업을 더 높은 TPS로 수행할 수 있습니다. 높은 처리량 할당량에 대해서는 Amazon SQS 메시지 할당량의 메시지 처리량을 참조하세요.

아마존 SQS 주요 용어

다음 주요 용어 설명은 FIFO 대기열 기능을 보다 정확하게 이해하는 데 도움이 될 수 있습니다. 자세한 내용은 Amazon Simple Queue Service API 참조를 참조하세요.

메시지 중복 제거 ID

전송된 메시지의 중복 제거에 사용되는 토큰입니다. 특정 메시지 중복 제거 ID가 있는 메시지를 성공적으로 전송한 경우, 메시지 중복 제거 ID가 동일한 모든 메시지는 성공적으로 수신되지만 중복 제거 간격인 5분 동안은 전달되지 않습니다.

> **Note**
>
> Amazon SQS는 메시지가 수신되어 삭제된 후에도 메시지 중복 제거 ID를 계속 추적합니다.

메시지 그룹 ID

메시지가 특정 메시지 그룹에 속하도록 지정하는 태그입니다. 동일한 메시지 그룹에 속하는 메시지는 항상 메시지 그룹을 기준으로 엄격한 순서에 따라 하나씩 처리됩니다(다른 메시지 그룹에 속하는 메시지는 순서가 뒤바뀌어 처리될 수 있음).

수신 요청 시도 ID

`ReceiveMessage` 호출의 중복 제거에 사용되는 토큰입니다.

시퀀스 번호

Amazon SQS가 각 메시지에 할당하는 연속되지 않은 큰 숫자입니다.

아마존 SQS에서의 FIFO 호환성

클라이언트

Amazon SQS의 버퍼링된 비동기식 클라이언트는 현재 FIFO 대기열을 지원하지 않습니다.

서비스

애플리케이션에서 여러 AWS 서비스를 사용하거나 외부 서비스를 혼합하여 사용하는 경우 FIFO 대기열을 지원하지 않는 서비스 기능을 이해하는 것이 중요합니다. AWS

Amazon SQS로 알림을 전송하는 일부 AWS 또는 외부 서비스는 FIFO 대기열을 대상으로 설정할 수 있음에도 불구하고 FIFO 대기열과 호환되지 않을 수 있습니다.

다음 AWS 서비스 기능은 현재 FIFO 대기열과 호환되지 않습니다.

- Amazon S3 이벤트 알림
- Auto Scaling 수명 주기 후크
- AWS IoT 규칙 조치
- AWS Lambda 데드레터 대기열

기타 서비스와 FIFO 대기열의 호환성에 관한 자세한 내용은 서비스 설명서를 참조하세요.

Amazon SQS의 FIFO 대기열 및 메시지 식별자

이 섹션에서는 FIFO 대기열의 식별자에 대해 설명합니다. 이러한 식별자는 특정 대기열과 메시지를 찾고 조작하는 데 도움이 될 수 있습니다.

주제

Amazon SQS의 FIFO 대기열에 대한 식별자

다음 식별자에 대한 자세한 내용은 Amazon Simple Queue Service API 참조를 참조하세요.

대기열 이름 및 URL

새 대기열을 생성할 때는 AWS 계정 및 리전에 고유한 대기열 이름을 지정해야 합니다. Amazon SQS는 사용자가 생성한 각 대기열에 대기열 이름과 기타 Amazon SQS 구성 요소가 포함된 대기열 URL이라는 식별자를 할당합니다. 대기열에서 작업을 실시할 때마다 대기열 URL을 지정합니다.

FIFO 대기열의 이름은 `.fifo` 접미사로 끝나야 합니다. 접미사는 문자 80개의 대기열 이름 할당량에 포함됩니다. 해당 대기열이 접미사로 끝나는지 확인하여 FIFO 대기열인지 여부를 알 수 있습니다.

다음은 AWS 계정 번호를 `123456789012` 가진 사용자가 `MyQueue` 소유한 FIFO 대기열의 대기열 URL입니다.

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

대기열을 나열하고 계정 번호 뒤에 오는 문자열을 구문 분석하여 프로그래밍 방식으로 대기열의 URL을 검색할 수 있습니다. 자세한 정보는 `ListQueues`을 참조하세요.

메시지 ID

각 메시지는 Amazon SQS가 `SendMessage` 응답으로 반환하는 시스템 할당 메시지 ID를 수신합니다. 이 식별자는 메시지를 식별하는 데 유용합니다. 메시지 ID의 최대 길이는 100자입니다.

수신 핸들

대기열에서 메시지를 수신할 때마다 해당 메시지의 수신 핸들을 수신합니다. 이 핸들은 메시지 자체가 아닌 메시지 수신 작업과 관련되어 있습니다. 메시지를 삭제하거나 메시지 표시를 변경하려면, (메시지 ID가 아닌) 수신 핸들을 지정해야 합니다. 또한, 메시지를 수신한 후 언제든지 삭제할 수 있습니다(메시지를 대기열에 배치한 후 회수할 수 없음). 수신 핸들의 최대 길이는 1024자입니다.

> **Important**
>
> 메시지를 두 번 이상 받으면, 받을 때마다 다른 수신 핸들을 받습니다. 메시지 삭제를 요청할 때에는 가장 최근에 받은 수신 핸들을 제공해야 합니다(그렇지 않으면 메시지가 삭제될 수 있음).

다음은 수신 핸들 예제입니다(세 줄로 나뉨).

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

Amazon SQS FIFO 대기열의 추가 식별자

다음 식별자에 대한 자세한 내용은 Amazon SQS에서 정확히 한 번의 프로세싱 및 Amazon Simple Queue Service API 참조를 참조하세요.

메시지 중복 제거 ID

전송된 메시지의 중복 제거에 사용되는 토큰입니다. 특정 메시지 중복 제거 ID가 있는 메시지를 성공적으로 전송한 경우, 메시지 중복 제거 ID가 동일한 모든 메시지는 성공적으로 수신되지만 중복 제거 간격인 5분 동안은 전달되지 않습니다.

메시지 그룹 ID

메시지가 특정 메시지 그룹에 속하도록 지정하는 태그입니다. 동일한 메시지 그룹에 속하는 메시지는 항상 메시지 그룹을 기준으로 엄격한 순서에 따라 하나씩 처리됩니다(다른 메시지 그룹에 속하는 메시지는 순서가 뒤바뀌어 처리될 수 있음).

시퀀스 번호

Amazon SQS가 각 메시지에 할당하는 연속되지 않은 큰 숫자입니다.

Amazon SQS 할당량

이 주제에서는 Amazon Simple Queue Service(Amazon SQS) 내의 할당량을 나열합니다.

주제

아마존 SQS FIFO 대기열 할당량

Amazon SQS 할당량

다음 표에는 FIFO 대기열과 관련된 할당량이 나열되어 있습니다.

할당량	설명
지연 대기열	대기열의 기본(최소) 지연 시간은 0초입니다. 최대값은 15분입니다.
대기열 나열	`ListQueues` 요청당 대기열 1,000개
긴 폴링 대기 시간	긴 폴링 대기 시간의 최대값은 20초입니다.
메시지 그룹	FIFO 대기열 내 메시지 그룹의 수에는 할당량이 없습니다.
대기열당 메시지(백로그)	Amazon SQS 대기열이 저장할 수 있는 메시지 수에는 제한이 없습니다.
대기열당 메시지(이동 중)	FIFO 대기열의 경우 최대 20,000개의 이동 중인 메시지(소비자가 대기열에서 수신했지만 대기열에서 아직 삭제되지 않은 메시지)가 있을 수 있습니다. 이 할당량에 도달해도 Amazon SQS에서는 오류 메시지를 반환하지 않습니다.
대기열 이름	FIFO 대기열의 이름은 `.fifo` 접미사로 끝나야 합니다. 접미사는 문자 80개의 대기열 이름 할당량에 포함됩니다. 대

할당량	설명
	기열 이름이 접미사로 끝나는지 확인하여 FIFO 대기열인지 여부를 알 수 있습니다.
대기열 태그	대기열에 50개 이상의 태그를 추가하지 않는 것이 좋습니다. 태그 지정 시 UTF-8 형식의 유니코드 문자를 사용할 수 있습니다.
	Key 태그는 필수지만 Value 태그는 선택 사항입니다.
	Key 태그 및 Value 태그는 대소문자를 구분합니다.
	Key 태그 및 Value 태그에는 UTF-8 형식의 유니코드 알파벳 문자와 공백이 포함될 수 있습니다. 다음 특수 문자가 허용됩니다. _ . : / = + - @
	Key 태그 또는 Value 태그에 예약된 접두사 aws:가 포함되며 안 됩니다(이 접두사가 있는 태그 키 또는 값은 삭제할 수 없음).
	Key 태그의 최대 길이는 UTF-8 형식의 유니코드 문자 128자입니다. Key 태그는 비어 있거나 null이면 안 됩니다.
	Value 태그의 최대 길이는 UTF-8 형식의 유니코드 문자 256자입니다. Value 태그는 비어 있거나 null일 수 있습니다.
	태깅 작업은 1인당 30TPS로 제한됩니다. AWS 계정애플리케이션에 더 많은 처리량이 필요한 경우 요청을 제출하십시오.

Amazon SQS 메시지 할당량

다음 표에 메시지와 관련된 할당량이 나열되어 있습니다.

할당량	설명
일괄 메시지 ID	일괄 메시지 ID는 최대 80자까지 가능합니다. 허용되는 문자는 영숫자, 하이픈(-), 밑줄(_) 입니다.
메시지 속성	메시지 하나에는 최대 10개의 메타데이터 속성이 있을 수 있습니다.
메시지 배치	단일 메시지 배치 요청에는 최대 10개 메시지가 포함될 수 있습니다. 자세한 내용은 Amazon SQS 배치 작업 섹션의 BufferedAsyncAmazonSQS 클라이언트 구성 섹션을 참조하세요.
메시지 콘텐츠	메시지에는 XML, JSON 및 포맷되지 않은 텍스트만이 포함될 수 있습니다. 다음 유니코드 문자가 허용됩니다. #x9 | #xA | #xD | #x20 ~ #xD7FF | #xE000 to #xFFFD | #x10000 ~ #x10FFFF 이 목록에 포함되지 않은 문자는 거부됩니다. 자세한 내용은 문자에 대한 W3C 사양을 참조하세요.
메시지 그룹 ID	백로그의 메시지를 사용하여 메시지 그룹 ID가 동일한 메시지의 대규모 백로그가 누적되는 것을 방지합니다. MessageGroupId 는 FIFO 대기열에 필요합니다. 표준 대기열에는 사용할 수 없습니다. 비어 있지 않은 MessageGroupId 를 메시지와 연결해야 합니다. MessageGroupId 를 제공하지 않으면 작업에 실패합니다. MessageGroupId 의 최대 길이는 128자입니다. 유효한 값은 영숫자 및 문장 부호(!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~) 입니다.
메시지 보존	기본적으로 메시지는 4일간 보존됩니다. 최소 기간은 60초(1분)입니다. 최소 시간은 1,209,600초(14일)입니다.

할당량	설명
메시지 처리량	표준 대기열은 API 작업(SendMessage , ReceiveMessage 또는 DeleteMessage)별 초당 거의 무제한의 API 호출 수를 지원합니다.
	FIFO 대기열 • FIFO 대기열은 API 작업(SendMessage , ReceiveMessage 및 DeleteMessage)별 초당 300개의 트랜잭션 할당량을 지원합니다. • 배치 처리를 사용하면 FIFO 대기열은 API 작업(SendMessage , ReceiveMessage 또는 DeleteMessage)별 초당 최대 3,000개의 메시지를 지원합니다. 초당 3,000개의 메시지는 300개의 API 호출을 나타내며, 각각 10개의 메시지를 배치 처리합니다.

할당량	설명
	FIFO 대기열의 높은 처리량 • 미국 동부(버지니아 북부), 미국 서부(오레곤) 및 유럽(아일랜드) 리전에는 배치 처리(SendMessage , ReceiveMessage 및 DeleteMessage) 없이도 API 작업별 초당 최대 70,000개의 많은 트랜잭션을 처리하는 FIFO 대기열이 있습니다. • 미국 동부(오하이오) 및 유럽(프랑크푸르트) 리전의 경우 기본 처리량은 API 작업별 초당 18,000개의 트랜잭션입니다. • 아시아 태평양(뭄바이), 아시아 태평양(싱가포르), 아시아 태평양(시드니) 및 아시아 태평양(도쿄) 리전의 기본 처리량은 API 작업별 초당 9,000개 트랜잭션입니다. • 유럽(런던) 및 남아메리카(상파울루)의 기본 처리량은 API 작업별 초당 4,500개 트랜잭션입니다. • 처리량을 최대화하려면 배치 처리하지 않고 보내는 메시지에 사용하는 메시지 그룹 ID의 수를 늘리세요. • 미국 동부(버지니아 북부), 미국 서부(오레곤) 및 유럽(아일랜드) 리전에서 배치 처리 API(SendMessageBatch 및 DeleteMessageBatch)를 사용하여 처리량을 초당 최대 700,000개 메시지로 늘릴 수 있습니다. 초당 700,000개의 메시지는 초당 70,000개의 트랜잭션을 나타내며, 각각 10개의 메시지를 배치 처리합니다. 유럽(프랑크푸르트) 및 미국 동부(오하이오) 리전의 경우 배치 처리 API를 사용하여 초당 최대 180,000개의 메시지를 처리할 수 있습니다. 초당 180,000개의 메시지는 초당 18,000개의 트랜잭션을 나타내며, 각각 10개의 메시지를 배치 처리합니다. 아시아 태평양(뭄바이), 아시아 태평양(싱가포르), 아시아 태평양(시드니) 및 아시아 태평양(도쿄) 리전의 경우 배치 처리를 통해 초당 최대 90,000개의 메시지를 처리할 수 있습니다. SendMessageBatch 및 DeleteMes

할당량	설명
	sageBatch 를 사용할 때 처리량을 최대화하려면 배치 요청의 모든 메시지가 동일한 메시지 그룹 ID를 사용해야 합니다. • 유럽(런던) 및 남아메리카(상파울루) 리전의 경우 배치 처리를 통해 초당 최대 45,000개의 메시지를 처리할 수 있습니다. SendMessageBatch 및 DeleteMessageBatch 를 사용할 때 처리량을 최대화하려면 배치 요청의 모든 메시지가 동일한 메시지 그룹 ID를 사용해야 합니다. • 다른 모든 AWS 지역의 최대 처리량은 API 작업당 초당 메시지 2,400개 (일괄 처리 제외) 또는 24,000개 (일괄 처리 사용 시) 입니다. • 지역 한도를 초과하여 할당량을 늘리도록 요청하려면 지원 요청을 제출하십시오. • 자세한 정보는 SQS FIFO 대기열의 높은 처리량을 위한 파티션 및 데이터 배포을 참조하세요.
메시지 타이머	메시지의 기본(최소) 지연 시간은 0초입니다. 최대값은 15분입니다.
메시지 크기	최소 메시지 크기는 1바이트(1자리)입니다. 최대 크기는 262,144바이트(256KiB)입니다. 256KiB보다 큰 메시지를 보내려면 Java용 Amazon SQS 확장 클라이언트 라이브러리와 Python용 Amazon SQS 확장 클라이언트 라이브러리를 사용할 수 있습니다. 이 라이브러리를 통해 Amazon S3에 메시지 페이로드에 대한 참조를 포함하는 Amazon SQS 메시지를 전송할 수 있습니다. 최대 페이로드 크기는 2GB입니다. **Note** 이 확장 라이브러리는 동기 클라이언트에서만 작동합니다.

할당량	설명
메시지 제한 시간 초과	메시지의 제한 시간 초과는 30초입니다. 최소 시간은 0초입니다. 최대 시간은 12시간입니다.
정책 정보	최대 할당량은 8,192바이트, 20개 문, 주체 50개 또는 조건 10개입니다. 자세한 내용은 Amazon SQS 정책 할당량을(를) 참조하세요.

Amazon SQS 정책 할당량

다음 표에 정책과 관련된 할당량이 나열되어 있습니다.

명칭	Maximum
바이트	8,192
조건	10
Principal	50
Statement	20
명령문당 작업	7

Amazon SQS의 특징 및 기능

Amazon SQS는 다음과 같은 특징 및 기능을 제공합니다.

주제

Amazon SQS에서 데드레터 대기열 사용

Amazon SQS는 소스 대기열이 성공적으로 처리되지 않은 메시지를 대상으로 지정할 수 있는 데드레터 대기열 (DLQ) 을 지원합니다. DLQ는 사용되지 않은 메시지를 분리하여 처리가 실패한 이유를 확인할 수 있기 때문에 애플리케이션을 디버깅하는 데 유용합니다. 최적의 성능을 위해서는 소스 대기열과 DLQ를 동일한 지역 내에 유지하는 것이 가장 좋습니다. AWS 계정 메시지가 데드레터 대기열에 들어가면 다음과 같은 작업을 수행할 수 있습니다.

- 메시지가 DLQ(Dead Letter Queue)로 이동되게 만든 예외적인 상황의 로그를 검사합니다.
- 데드레터 대기열로 이동된 메시지의 내용을 분석하여 애플리케이션 문제를 진단할 수 있습니다.
- 소비자에게 메시지를 처리하기에 충분한 시간을 주었는지 확인합니다.
- 데드레터 큐 재드라이브를 사용하여 데드레터 큐 밖으로 메시지를 이동할 수 있습니다.

데드레터 큐로 구성하려면 먼저 새 큐를 만들어야 합니다. Amazon SQS 콘솔을 사용하여 DLQ(Dead Letter Queue)를 구성하는 방법에 대한 자세한 내용은 Amazon SQS 콘솔을 사용하여 데드레터 대기열을 구성하는 방법을 알아보십시오. 섹션을 참조하세요. 데드레터 큐로 이동된 메시지에 대한 경보를 구성하는 방법 등 데드레터 큐에 대한 도움이 필요하면 을 참조하십시오. Amazon을 사용하여 데드레터 대기열에 대한 알람 생성 CloudWatch

데드레터 대기열에 대한 정책 사용

리드라이브 정책을 사용하여 지정합니다. `maxReceiveCount` `maxReceiveCount`데드레터 대기열로 이동하기 전에 소비자가 소스 대기열로부터 메시지를 수신할 수 있는 횟수입니다. 예를 들어 를 1과 `maxReceiveCount` 같이 낮은 값으로 설정하면 메시지를 한 번 수신하지 못하면 메시지가 데드레터 대기열로 이동하게 됩니다. 시스템이 오류에 대해 복원력을 갖도록 하려면 충분한 재시도를 허용할 수 있을 정도로 `maxReceiveCount`를 높게 설정하세요.

리드라이브 허용 정책은 DLQ(Dead Letter Queue)에 액세스할 수 있는 소스 대기열을 지정합니다. 모든 소스 대기열을 허용할지, 특정 소스 대기열을 허용할지 또는 데드레터 대기열의 모든 소스 대기열 사용을 거부할지 선택할 수 있습니다. 기본값에서는 모든 소스 대기열에서 데드레터 대기열을 사용할 수 있습니다. byQueue옵션을 사용하여 특정 대기열을 허용하도록 선택한 경우 소스 대기열 Amazon Resource Name (ARN) 을 사용하여 소스 대기열을 최대 10개까지 지정할 수 있습니다. `denyAll`을 지정하는 경우, 대기열을 DLQ(Dead Letter Queue)로 사용할 수 없습니다.

데드레터 대기열의 메시지 보존 기간 이해

표준 대기열의 경우, 메시지 만료는 항상 메시지가 원래 대기열에 추가된 타임스탬프를 기준으로 합니다. 메시지가 DLQ(Dead Letter Queue)로 이동하면 대기열에 추가 타임스탬프는 변경되지 않습니다. `ApproximateAgeOfOldestMessage`지표는 메시지가 처음 전송된 시기가 아니라 메시지가 데드레터 대기열로 이동한 시기를 나타냅니다. 예를 들어, 메시지가 DLQ(Dead Letter Queue)로 이동하기 전에 원래 대기열에서 1일을 보낸다고 가정합니다. DLQ(Dead Letter Queue)의 보존 기간이 4일인 경우 메시지는 3일 후에 DLQ(Dead Letter Queue)에서 삭제되며 `ApproximateAgeOfOldestMessage`는 3일입니다. 따라서 배달 못한 편지 대기열의 보존 기간을 항상 원래 대기열의 보존 기간보다 더 길게 설정하는 것이 좋습니다.

FIFO 대기열의 경우 메시지가 DLQ(Dead Letter Queue)로 이동하면 대기열에 추가 타임스탬프가 재설정됩니다. `ApproximateAgeOfOldestMessage` 지표는 메시지가 DLQ(Dead Letter Queue)로 이동한 시간을 나타냅니다. 위의 동일한 예에서 메시지는 4일 후에 배달 못한 편지 대기열에서 삭제되며 `ApproximateAgeOfOldestMessage`는 4일입니다.

Amazon SQS 콘솔을 사용하여 데드레터 대기열을 구성하는 방법을 알아보십시오.

데드레터 큐는 성공적으로 처리되지 않은 메시지를 소스 큐에서 대상으로 지정할 수 있는 큐입니다. 자세한 정보는 Amazon SQS에서 데드레터 대기열 사용 을 참조하세요.

Amazon SQS는 배달 못한 편지 대기열을 자동으로 생성하지 않습니다. 대기열을 배달 못한 편지 대기열로 사용하기 전에 먼저 대기열을 생성해야 합니다. 데드레터 큐로 사용할 큐를 만드는 방법에 대한 지침은 을 참조하십시오. Amazon SQS 콘솔을 사용하여 대기열을 생성합니다.

FIFO 대기열의 배달 못한 편지 대기열 역시 FIFO 대기열이어야 합니다. 이와 마찬가지로 표준 대기열의 배달 못한 편지 대기열 역시 표준 대기열이어야 합니다.

대기열을 생성 또는 편집할 때 배달 못한 편지 대기열을 구성할 수 있습니다.

기존 대기열에 대한 배달 못한 편지 대기열을 구성하려면(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열을 선택하고 편집을 선택합니다.
4. 배달 못한 편지 대기열 섹션으로 스크롤하고 활성화됨을 선택합니다.
5. 이 소스 대기열과 연결하려는 기존 배달 못한 편지 대기열의 Amazon 리소스 이름(ARN)을 선택합니다.
6. 메시지를 배달 못한 편지 대기열로 보내기 전까지 시도할 수신 횟수를 구성하기 위해 최대 수신 수를 1에서 1,000 사이의 값으로 설정합니다.
7. 배달 못한 편지 대기열 구성을 마쳤으면 저장을 선택합니다.

   대기열을 저장하면 콘솔에 대기열의 세부 정보 페이지가 표시됩니다. 세부 정보 페이지의 배달 못한 편지 대기열 탭에는 배달 못한 편지 대기열의 최대 수신 수 및 배달 못한 편지 대기열 ARN이 표시됩니다.

Amazon SQS에서 데드레터 대기열 재드라이브를 구성하는 방법을 알아보십시오.

데드레터 대기열 재드라이브를 사용하여 사용하지 않은 메시지를 기존 데드레터 대기열 밖으로 이동할 수 있습니다. 기본적으로 DLQ(Dead Letter Queue) 리드라이브는 메시지를 DLQ(Dead Letter Queue)에서 소스 대기열로 이동합니다. 그러나 두 대기열이 동일한 유형인 경우 다른 대기열을 리드라

이브 대상으로 구성할 수도 있습니다. 예를 들어 DLQ(Dead Letter Queue)가 FIFO 대기열인 경우 리드라이브 대상 대기열도 FIFO 대기열이어야 합니다. 또한 Amazon SQS가 메시지를 이동하는 속도를 설정하도록 리드라이브 속도를 구성할 수 있습니다.

> **Note**
>
> 메시지가 FIFO 대기열에서 FIFO DLQ로 이동되면 원본 메시지의 중복 제거 ID가 원본 메시지의 ID로 대체됩니다. 이는 중복 제거 ID를 공유하는 두 개의 독립적인 메시지가 DLQ 중복 제거로 인해 저장되지 않도록 하기 위한 것입니다.

데드레터 큐는 가장 오래된 메시지부터 시작하여 수신된 순서대로 메시지를 다시 전송합니다. 그러나 대상 대기열은 수신 순서에 따라 리드라이브된 메시지뿐만 아니라 다른 생산자의 새 메시지도 수집합니다. 예를 들어 생산자가 소스 FIFO 대기열로 메시지를 보내는 동시에 데드레터 대기열에서 다시 구동된 메시지를 수신하는 경우 재구동된 메시지는 생산자가 보낸 새 메시지와 뒤섞이게 됩니다.

> **Note**
>
> 리드라이브 작업은 보존 기간을 재설정합니다. 모든 재제어 메시지는 새 메시지와 새 메시지로 간주되며 다시 제어된 메시지에 `messageID` 할당됩니다. `enqueueTime`

주제

- Amazon SQS API를 사용하여 기존 표준 대기열에 데드레터 대기열 재드라이브 구성
- Amazon SQS 콘솔을 사용하여 기존 표준 대기열에 데드레터 대기열 재드라이브 구성
- 배달 못한 편지 대기열 리드라이브에 대한 대기열 권한 구성

Amazon SQS API를 사용하여 기존 표준 대기열에 데드레터 대기열 재드라이브 구성

, 및 API 작업을 사용하여 데드레터 큐 재드라이브를 구성할 수 있습니다`SendMessageBatch`. `ReceiveMessage` `DeleteMessageBatch`

API 작업	설명
`StartMessageMoveTask`	지정된 소스 대기열에서 지정된 대상 대기열로 메시지를 이동하는 비동기 작업을 시작합니다.

API 작업	설명
ListMessageMoveTasks	특정 소스 대기열에서 가장 최근의 메시지 이동 작업(최대 10개)을 가져옵니다.
CancelMessageMoveTask	지정된 메시지 이동 작업을 취소합니다. 메시지 이동은 현재 상태가 실행 중일 때만 취소할 수 있습니다.

Amazon SQS 콘솔을 사용하여 기존 표준 대기열에 데드레터 대기열 재드라이브 구성

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 배달 못한 편지 대기열로 구성한 대기열의 이름을 선택합니다.
4. DLQ 리드라이브 시작을 선택합니다.
5. 리드라이브 구성에서 메시지 대상에 대해 다음 중 하나를 수행합니다.
   - 메시지를 소스 대기열로 리드라이브하려면 소스 대기열로 리드라이브를 선택합니다.
   - 메시지를 다른 대기열로 리드라이브하려면 사용자 지정 대상으로 리드라이브를 선택합니다. 그런 다음 기존 대상 대기열의 Amazon 리소스 이름(ARN)을 입력합니다.
6. 속도 제어 설정에서 다음 중 하나를 선택합니다.
   - 시스템 최적화 - 배달 못한 편지 대기열 메시지를 초당 최대 메시지 수로 리드라이브합니다.
   - 사용자 지정 최고 속도 - 배달 못한 편지 대기열 메시지를 초당 최대 사용자 지정 메시지 속도로 리드라이브합니다. 최대 허용 속도는 초당 500개의 메시지입니다.
     - 사용자 지정 최고 속도를 작은 값으로 시작하고 소스 대기열에 메시지가 넘치지 않는지 확인하는 것이 좋습니다. 그런 다음 사용자 지정 최대 속도 값을 점진적으로 늘려 소스 대기열의 상태를 계속 모니터링합니다.
7. 배달 못한 편지 대기열 리드라이브 구성을 마쳤으면 메시지 리드라이브를 선택합니다.

> **Important**
>
> Amazon SQS는 배달 못한 편지 대기열에서 메시지를 리드라이브하는 동안 메시지를 필터링하고 수정하는 기능을 지원하지 않습니다.

DLQ(Dead Letter Queue) 리드라이브 작업은 최대 36시간까지 실행될 수 있습니다. Amazon SQS는 계정당 최대 100개의 활성 리드라이브 작업을 지원합니다.

8. 메시지 리드라이브 작업을 취소하려면 대기열의 세부 정보 페이지에서 DLQ 리드라이브 취소를 선택합니다. 진행 중인 메시지 리드라이브를 취소하면 이미 이동 대상 대기열로 이동된 모든 메시지는 대상 대기열에 남아 있게 됩니다.

배달 못한 편지 대기열 리드라이브에 대한 대기열 권한 구성

정책에 권한을 추가하여 사용자에게 특정 배달 못한 편지 대기열 작업에 대한 액세스 권한을 부여할 수 있습니다. 배달 못한 편지 대기열 리드라이브에 필요한 최소 권한은 다음과 같습니다.

최소 권한	필수 API 메서드
메시지 리드라이브 시작	• 배달 못한 편지 대기열의 `sqs:StartMessageMoveTask` , `sqs:ReceiveMessage` , `sqs:DeleteMessage` 및 `sqs:GetQueueAttributes` 를 추가합니다. 배달 못한 편지 대기열 또는 원본 소스 대기열이 암호화된 경우(SSE 대기열이라고도 함) 메시지를 암호화하는 데 사용된 모든 KMS 키에 대한 `kms:Decrypt` 도 필요합니다. • 대상 대기열의 `sqs:SendMessage` 를 추가합니다. 대상 대기열이 암호화된 경우 `kms:GenerateDataKey` 및 `kms:Decrypt` 도 필요합니다.
진행 중인 메시지 리드라이브 취소	• 배달 못한 편지 대기열의 `sqs:CancelMessageMoveTask` , `sqs:ReceiveMessage` , `sqs:DeleteMessage` 및 `sqs:GetQueueAttributes` 를 추가합니다. 배달 못한 편지 대기열이 암호화된 경우(SSE 대기열이라고도 함) `kms:Decrypt` 도 필요합니다.
메시지 이동 상태 표시	• 배달 못한 편지 대기열의 `sqs:ListMessageMoveTasks` 및 `sqs:GetQueueAttributes` 를 추가합니다.

암호화된 대기열 페어(배달 못한 편지 대기열이 있는 소스 대기열)에 대한 권한을 구성하려면

다음 단계를 사용하여 배달 못한 편지 대기열 리드라이브에 대한 최소 권한을 구성합니다.

1. 에 AWS Management Console 로그인하고 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 다음 권한이 있는 정책을 생성하고 로그인 IAM 사용자 또는 역할에 연결합니다.

   - `sqs:StartMessageMoveTask`
   - `sqs:CancelMessageMoveTask`
   - `sqs:ListMessageMoveTasks`
   - `sqs:ListDeadLetterSourceQueues`
   - `sqs:ReceiveMessage`
   - `sqs:DeleteMessage`
   - `sqs:GetQueueAttributes`
   - 배달 못한 편지 대기열의 `Resource` ARN(예: "arn:aws:sqs:*<DLQ_region>*:*<DLQ_accountId>*:*<DLQ_name>*").
   - `sqs:SendMessage`
   - 대상 대기열의 `Resource` ARN (예: “arn:aws:sqs: *< DestQueue _region>: < _accountId>: < _name>* “) DestQueue DestQueue
   - `kms:Decrypt` - 암호 해독 작업을 허용합니다.
   - `kms:GenerateDataKey`
   - 원본 소스 대기열에 있는 메시지를 암호화하는 데 사용된 KMS 암호화 키의 `Resource` ARN(예: "arn:aws:kms:*<region>*:*<accountId>*:key/*<keyId_used to encrypt the message body>*").
   - 리드라이브 대상 대기열에 사용되는 KMS 암호화 키의 리소스 ARN(예: "arn:aws:kms:*<region>*:*<accountId>*:key/*<keyId_used for the destination queue>*").

   액세스 정책은 다음과 유사합니다.

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

암호화되지 않은 대기열 페어(배달 못한 편지 대기열이 있는 소스 대기열)을 사용하여 권한을 구성하려면

다음 단계를 사용하여 표준의 암호화되지 않은 배달 못한 편지 대기열에 대한 최소 권한을 구성합니다. 필요한 최소 권한은 배달 못한 편지 대기열에서 속성을 수신, 삭제 및 가져오고 소스 대기열로 속성을 전송하는 것입니다.

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 다음 권한이 있는 정책을 생성하고 로그인 IAM 사용자 또는 역할에 연결합니다.

   - `sqs:StartMessageMoveTask`
   - `sqs:CancelMessageMoveTask`

- sqs:ListMessageMoveTasks
- sqs:ListDeadLetterSourceQueues
- sqs:ReceiveMessage
- sqs:DeleteMessage
- sqs:GetQueueAttributes
- 배달 못한 편지 대기열의 Resource ARN(예: "arn:aws:sqs:*<DLQ_region>*:*<DLQ_accountId>*:*<DLQ_name>*") .
- sqs:SendMessage
- *## #### Resource ARN (#: “arn:aws:sqs: < DestQueue _region>: < _accountId>: < _name> “) DestQueue DestQueue*

액세스 정책은 다음과 유사합니다.

```
{
"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 Amazon SQS 데드레터 큐 리드라이브에 대한 업데이트 및 권한 요구 사항

2023년 6월 8일, Amazon SQS는 SDK AWS 및 (CLI) 용 데드레터 큐 (DLQ) 재드라이브를 도입했습니다. AWS Command Line Interface 이 기능은 이미 지원되는 콘솔용 DLQ 리드라이브에 추가된 기능입니다. AWS 이전에 AWS 콘솔을 사용하여 데드레터 큐 메시지를 다시 구동한 적이 있다면 다음 변경 사항의 영향을 받을 수 있습니다.

- CloudTrail 데드레터 대기열 재드라이브의 이벤트 이름 변경
- DLQ(Dead Letter Queue) 재드라이브에 대한 권한 업데이트

CloudTrail 이벤트 이름 변경

2023년 10월 15일에 Amazon SQS 콘솔에서 데드레터 큐 재드라이브의 CloudTrail 이벤트 이름이 변경됩니다. 이러한 CloudTrail 이벤트에 대해 경보를 설정한 경우 지금 업데이트해야 합니다. DLQ redrive의 새 CloudTrail 이벤트 이름은 다음과 같습니다.

이전 이벤트 이름	새로운 이벤트 이름
`CreateMoveTask`	`StartMessageMoveTask`
`CancelMoveTask`	`CancelMessageMoveTask`

업데이트된 권한

SDK 및 CLI 릴리스에 포함된 Amazon SQS는 보안 모범 사례를 준수하기 위해 DLQ 리드라이브에 대한 대기열 권한도 업데이트했습니다. 다음 대기열 권한 유형을 사용하여 DLQ에서 메시지를 리드라이브하세요.

1. 작업 기반 권한(DLQ API 작업에 대한 업데이트)
2. 관리형 Amazon SQS 정책 권한
3. sqs:* 와일드카드를 사용하는 권한 정책

> **Important**
>
> SDK 또는 CLI용 DLQ 리드라이브를 사용하려면 위의 옵션 중 하나와 일치하는 DLQ 리드라이브 권한 정책이 있어야 합니다.

DLQ 리드라이브에 대한 대기열 권한이 위의 옵션 중 하나와 일치하지 않는 경우 2023년 8월 31일까지 권한을 업데이트해야 합니다. 지금부터 2023년 8월 31일까지는 이전에 DLQ 리드라이브를 사용한 리전에서만 AWS 콘솔을 통해 구성한 권한을 사용하여 메시지를 리드라이브할 수 있습니다. 예를 들어, us-east-1과 eu-west-1에 모두 '계정 A'가 있다고 가정해 보겠습니다. '계정 A'는 2023년 6월 8일 이전에 us-east-1의 AWS 콘솔에서 메시지를 재전송하는 데 사용되었지만 eu-west-1에서는 사용되지 않았습니다. 2023년 6월 8일부터 2023년 8월 31일 사이에 “계정 A의” 정책 권한이 위의 옵션 중 하나와 일치하지 않는 경우, AWS 콘솔의 us-east-1에서는 메시지를 재전송하는 데만 사용할 수 있으며 eu-west-1에서는 사용할 수 없습니다.

> **Important**
>
> 2023년 8월 31일 이후에 DLQ 리드라이브 권한이 이러한 옵션 중 하나와 일치하지 않는 경우 계정은 더 이상 AWS 콘솔을 사용하여 DLQ 메시지를 리드라이브할 수 없게 됩니다.
> 하지만 2023년 8월에 AWS 콘솔에서 DLQ 리드라이브 기능을 사용한 경우 2023년 10월 15일까지 이러한 옵션 중 하나에 따라 새 권한을 채택할 수 있습니다.
> 자세한 정보는 the section called “영향을 받는 정책 식별”을 참조하세요.

다음은 각 DLQ 리드라이브 옵션에 대한 대기열 권한 예제입니다. 서버 측 암호화 (SSE) 대기열을 사용하는 경우 해당 키 권한이 필요합니다. AWS KMS

작업 기반

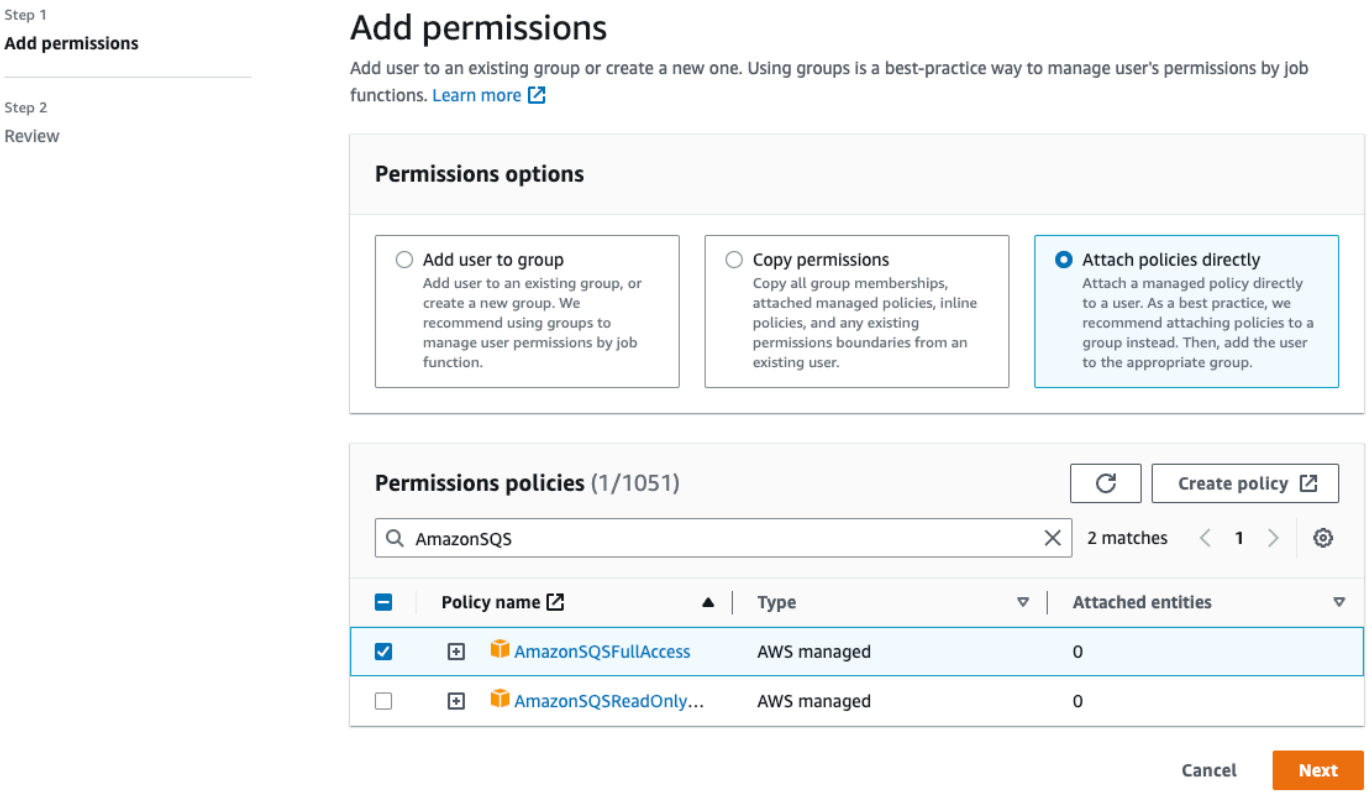
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:StartMessageMoveTask",
        "sqs:ListMessageMoveTasks",
```

```
        "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>"
    }
  ]
}
```

관리형 정책

다음 관리형 정책에는 업데이트된 필수 권한이 포함됩니다.

- AmazonSQS FullAccess — 데드레터 대기열 재구동 작업 (시작, 취소, 목록 표시) 이 포함됩니다.
- AmazonSQS ReadOnly Access — 읽기 전용 액세스를 제공하며 데드 레터 대기열 목록 재드라이브 작업을 포함합니다.



sqs* 와일드카드를 사용하는 권한 정책

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

영향을 받는 정책 식별

고객 관리형 정책 (CMP) 을 사용하는 AWS CloudTrail 경우 IAM을 사용하여 대기열 권한 업데이트의 영향을 받는 정책을 식별할 수 있습니다.

> **Note**
>
> `AmazonSQSFullAccess` 및 `AmazonSQSReadOnlyAccess`를 사용하는 경우 추가 조치가 필요하지 않습니다.

1. 콘솔에 로그인합니다. AWS CloudTrail
2. 이벤트 기록 페이지의 속성 조회에서 드롭다운 메뉴를 사용하여 이벤트 이름을 선택합니다. 그런 다음 `CreateMoveTask`를 검색합니다.
3. 세부 정보 페이지를 열려면 이벤트를 선택합니다. 이벤트 레코드 섹션에서 `userIdentity` ARN의 `UserName` 또는 `RoleName`을 검색합니다.
4. IAM 콘솔에 로그인합니다.
   - 사용자의 경우 사용자를 선택합니다. 이전 단계에서 식별된 `UserName` 사용자를 선택합니다.
   - 역할에서 역할을 선택합니다. 이전 단계에서 식별된 `RoleName` 사용자를 검색합니다.
5. 세부 정보 페이지의 권한 섹션에서 `Action`의 접두사가 `sqs:`인 정책을 검토하거나 `Resource`에서 정의된 Amazon SQS 대기열이 있는 정책을 검토합니다.

Amazon을 사용하여 데드레터 대기열에 대한 알람 생성 CloudWatch

CloudWatchAmazon과 지표를 사용하여 데드레터 대기열로 이동된 모든 메시지에 대해 경보를 구성할 수 있습니다. `ApproximateNumberOfMessagesVisible` 자세한 정보는 Amazon CloudWatch SQS 지표에 대한 경보 생성을 참조하세요. 메시지가 데드레터 대기열로 전송되었다는 알림을 받은 후 폴링을 사용하여 메시지를 검토하여 메시지를 수신할 수 있습니다.

Amazon SQS용 메시지 메타데이터

메시지 속성을 사용하여 사용자 지정 메타데이터를 애플리케이션에 대한 Amazon SQS 메시지에 연결할 수 있습니다. 메시지 시스템 속성을 사용하여 AWS X-Ray와 같은 다른 AWS 서비스에 대한 메타데이터를 저장할 수 있습니다.

주제

- Amazon SQS 메시지 속성
- Amazon SQS 메시지 시스템 속성

Amazon SQS 메시지 속성

Amazon SQS는 메시지 속성을 사용하여 메시지에 구조화된 메타데이터(예: 타임스탬프, 지리 공간 데이터, 서명 및 식별자)를 포함할 수 있습니다. 각 메시지에는 최대 10개의 속성이 있을 수 있습니다. 메시지 속성은 선택 사항이며 메시지 본문과 분리됩니다(단, 메시지 본문과 함께 전송됨). 소비자는 메시지 본문을 먼저 처리하지 않고도 메시지 속성을 사용하여 특정 방식으로 메시지를 처리할 수 있습니다. Amazon SQS 콘솔을 사용하여 속성이 있는 메시지를 전송하는 방법에 대한 자세한 내용은 속성을 포함하는 메시지 전송 섹션을 참조하세요.

> **Note**
>
> 메시지 속성을 메시지 시스템 속성과 혼동하지 마십시오. 메시지 속성을 사용하여 애플리케이션용 Amazon SQS 메시지에 사용자 지정 메타데이터를 첨부할 수 있는 반면, 메시지 시스템 속성을 사용하여 다음과 같은 AWS 다른 서비스에 대한 메타데이터를 저장할 수 있습니다.
> AWS X-Ray

주제

- 메시지 속성 구성 요소

- 메시지 속성 데이터 형식
- 메시지 속성의 MD5 메시지 다이제스트 계산

메시지 속성 구성 요소

> **Important**
>
> 메시지 속성의 모든 구성 요소는 256KB의 메시지 크기 제한에 포함됩니다.
> `Name`, `Type`, `Value`, 메시지 본문은 비어있거나 null이면 안됩니다.

각 메시지 속성은 다음 구성 요소로 구성됩니다.

- 이름 - 메시지 속성 이름에 사용할 수 있는 문자는 `A-Z`, `a-z`, `0-9`, 밑줄(`_`), 하이픈(`-`), 마침표(`.`)입니다. 다음과 같은 제한 사항이 있습니다.
  - 최대 256자일 수 있음
  - `AWS.` 또는 `Amazon.`(또는 모든 대소문자 변형)으로 시작할 수 없음
  - 대소문자 구분
  - 메시지의 모든 속성 이름 중에서 고유해야 함
  - 마침표로 시작하거나 끝나지 않아야 함
  - 시퀀스에 마침표가 없어야 함
- 형식 - 메시지 속성 데이터 형식입니다. 지원되는 형식은 `String`, `Number`, `Binary`입니다. 또한 데이터 형식에 대한 사용자 지정 정보를 추가할 수 있습니다. 데이터 유형에는 메시지 본문과 동일한 제한 사항이 있습니다(자세한 내용은 Amazon Simple Queue Service API 참조의 `SendMessage` 참조). 또한 다음과 같은 제한 사항이 적용됩니다.
  - 최대 256자일 수 있음
  - 대소문자 구분
- 값 - 메시지 속성 값입니다. `String` 데이터 형식의 경우 속성 값은 메시지 본문과 동일한 제한 사항이 있습니다.

메시지 속성 데이터 형식

메시지 속성 데이터 형식은 해당 메시지 속성 값의 처리 방법을 Amazon SQS에 지시합니다. 예를 들어 형식이 `Number`인 경우 Amazon SQS에서 숫자 값을 확인합니다.

Amazon SQS는 `String`, `Number`, `Binary` 논리 데이터 형식과 *`.custom-data-type`* 형식의 선택적 사용자 지정 데이터 형식 레이블을 지원합니다.

- 문자열 - `String` 속성은 모든 유효한 XML 문자를 사용하여 유니코드 텍스트를 저장할 수 있습니다.
- 숫자 - `Number` 속성은 양수 또는 음수 값을 저장할 수 있습니다. 숫자는 최대 38자릿수의 정밀도를 갖출 수 있으며 10^-128에서 10^+126 사이일 수 있습니다.

> **Note**
>
> Amazon SQS는 앞과 끝의 0을 제거합니다.

- 이진 - 이진 속성은 모든 이진 데이터를 저장할 수 있습니다(예: 압축된 데이터, 암호화된 데이터 또는 이미지).
- 사용자 지정 - 사용자 지정 데이터 형식을 생성하려면 사용자 지정 형식 레이블을 데이터 형식에 추가합니다. 예:
  - `Number.byte`, `Number.short`, `Number.int`, `Number.float`는 숫자 형식을 구별하는 데 도움이 될 수 있습니다.
  - `Binary.gif`와 `Binary.png`는 파일 형식을 구별하는 데 도움이 될 수 있습니다.

> **Note**
>
> Amazon SQS는 추가된 데이터를 해석, 확인 또는 사용하지 않습니다.
> 사용자 지정 형식 레이블은 메시지 본문과 동일한 제한 사항이 있습니다.

메시지 속성의 MD5 메시지 다이제스트 계산

를 사용하는 경우 이 AWS SDK for Java섹션을 건너뛸 수 있습니다. Java용 SDK의 `MessageMD5ChecksumHandler` 클래스는 Amazon SQS 메시지 속성에서 MD5 메시지 다이제스트를 지원합니다.

Query API 또는 Amazon SQS 메시지 속성에 대해 MD5 메시지 다이제스트를 지원하지 않는 AWS SDK 중 하나를 사용하는 경우 다음 지침을 사용하여 MD5 메시지 다이제스트 계산을 수행해야 합니다.

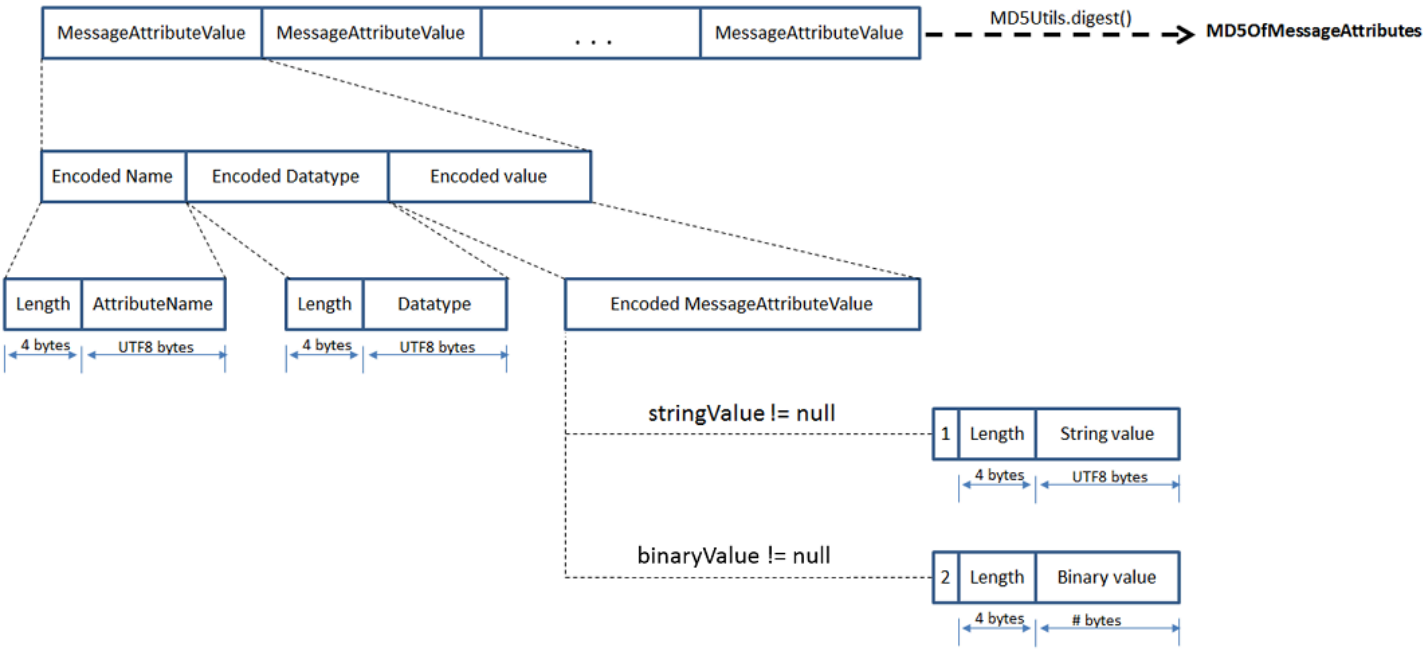
> **Note**
>
> MD5 메시지 다이제스트 계산에는 항상 사용자 지정 데이터 유형 접미사를 포함하세요.

개요

다음은 MD5 메시지 다이제스트 계산 알로리즘의 개요입니다.

1. 모든 메시지 속성을 이름에 따라 오름차순으로 정렬합니다.
2. 각 속성의 개별 부분(Name, Type, Value)을 버퍼로 인코딩합니다.
3. 전체 버퍼의 메시지 다이제스트를 계산합니다.

다음 다이어그램은 단일 메시지 속성에서 MD5 메시지의 인코딩을 나타냅니다.



단일 Amazon SQS 메시지 속성을 인코딩하려면

1. 이름(이름의 길이(4바이트) 및 UTF-8바이트)을 인코딩합니다.
2. 데이터 형식(데이터 형식의 길이(4바이트) 및 UTF-8바이트)을 인코딩합니다.
3. 값(1바이트)의 전송 형식(String 또는 Binary)을 인코딩합니다.

> **Note**
>
> `String` 및 `Number` 논리 데이터 형식은 `String` 전송 형식을 사용합니다.
> `Binary` 논리 데이터 형식은 `Binary` 전송 형식을 사용합니다.

   a. `String` 전송 형식은 1을 인코딩합니다.
   b. `Binary` 전송 형식은 2를 인코딩합니다.

4. 속성 값을 인코딩합니다.
   a. `String` 전송 형식은 속성 값(값의 길이(4바이트) 및 UTF-8바이트)을 인코딩합니다.
   b. `Binary` 전송 형식은 속성 값(값의 길이(4바이트) 및 원시 바이트)을 인코딩합니다.

Amazon SQS 메시지 시스템 속성

메시지 속성을 사용하여 사용자 지정 메타데이터를 응용 프로그램에 대한 Amazon SQS 메시지에 연결할 수 있는 반면, 메시지 시스템 속성을 사용하여 AWS X-Ray같은 다른 AWS 서비스에 대한 메타데이터를 지정할 수 있습니다. 자세한 내용은 Amazon Simple Queue Service API 참조의 `SendMessage` 및 `SendMessageBatch` API 작업의 `MessageSystemAttribute` 요청 파라미터, `ReceiveMessage` API 작업의 `AWSTraceHeader` 속성, `MessageSystemAttributeValue` 데이터 유형을 참조하세요.

메시지 시스템 속성은 다음을 제외하고 메시지 시스템과 똑같이 구조화됩니다.

- 현재 유일하게 지원되는 메시지 시스템 속성은 `AWSTraceHeader`입니다. 유형은 이어야 `String` 하고 값은 올바른 형식의 AWS X-Ray 추적 헤더 문자열이어야 합니다.
- 메시지 시스템 속서으이 크기는 총 메시지 크기에 합산되지 않습니다.

Amazon SQS 메시지를 처리하는 데 필요한 리소스

대기된 메시지를 처리해야 하는 리소스를 예측할 때 도움이 될 수 있도록 Amazon SQS에서는 대기열에 있는 지연된 메시지, 표시되는 메시지, 표시되지 않는 메시지의 대략적인 수를 확인할 수 있습니다. 표시에 대한 자세한 정보는 Amazon SQS 제한 시간 초과 단원을 참조하십시오.

> **Note**
>
> 표준 대기열의 경우, Amazon SQS의 분산 아키텍처로 인해 그 결과는 거의 비슷합니다. 대부분의 경우 그 수는 대기열에 있는 메시지의 실제 개수에 근접해야 합니다.
> FIFO 대기열의 경우, 그 결과는 정확합니다.

다음 표에는 GetQueueAttributes 작업에 사용할 속성 이름이 나열되어 있습니다:

작업	속성 이름
대기열로부터 검색이 가능한 메시지의 대략적인 수를 확인합니다.	ApproximateNumberOfMessages Visible
대기열 중 지연되고 즉시 읽기가 불가능한 메시지의 대략적인 수를 확인합니다. 이러한 경우는 대기열이 지연 대기열로 구성되거나 메시지가 지연 파라미터와 함께 전송되었을 때 발생할 수 있습니다.	ApproximateNumberOfMessages Delayed
이동 중인 메시지의 대략적인 수를 확인합니다. 클라이언트에게 전송되었으나 아직 삭제되지 않았거나 가시성 창 말단에 이르지 않은 경우 이동 중인 것으로 간주됩니다.	ApproximateNumberOfMessages NotVisible

목록 대기열 페이지 매김

listQueues 및 listDeadLetterQueues API 메서드는 선택적 페이지 매김 제어를 지원합니다. 기본적으로 이러한 API 메서드는 응답 메시지에 최대 1000개의 대기열을 반환합니다. 각 응답에서 반환되는 결과 수가 적도록 MaxResults 파라미터를 설정할 수 있습니다.

listQueues 또는 listDeadLetterQueues 요청에서 MaxResults 파라미터를 설정하여 응답에서 반환할 최대 결과 수를 지정합니다. MaxResults를 설정하지 않으면 응답에는 최대 1,000개의 결과가 포함되며 응답의 NextToken 값은 null입니다.

MaxResults를 설정한 경우, 표시할 추가 결과가 있으면 응답에 NextToken에 대한 값이 포함됩니다. listQueues에 대한 다음 요청에서 NextToken을 파라미터로 사용하여 다음 결과 페이지를 수신합니다. 표시할 추가 결과가 없으면 응답의 NextToken 값은 null입니다.

Amazon SQS 비용 할당 태그

비용 할당에 대한 Amazon SQS 대기열을 구성하고 식별하려면 대기열의 목적, 소유자 또는 환경을 식별하는 메타데이터 태그를 추가할 수 있습니다. 이 기능은 대기열이 많을 때 특히 유용합니다. Amazon SQS 콘솔을 사용하여 태그를 구성하려면 the section called “대기열에 대해 태그 구성” 섹션을 참조하세요.

비용 할당 태그를 사용하여 자체 비용 구조를 반영하도록 AWS 청구서를 구성할 수 있습니다. 이렇게 하려면 가입하여 AWS 계정 청구서에 태그 키와 값이 포함되도록 하세요. 자세한 내용은 AWS Billing 사용 설명서에서 월간 비용 할당 보고서 설정을 참조하세요.

각 태그는 사용자가 정의하는 키-값 페어로 구성됩니다. 예를 들어, 다음과 같이 대기열에 태그를 지정하면 프로덕션 및 테스트 대기열을 쉽게 식별할 수 있습니다.

대기열	키	값
MyQueueA	QueueType	Production
MyQueueB	QueueType	Testing

> **Note**
>
> 대기열 태그를 사용할 경우 다음 지침을 유념해야 합니다.
>
> - 대기열에 50개 이상의 태그를 추가하지 않는 것이 좋습니다. 태그 지정 시 UTF-8 형식의 유니코드 문자를 사용할 수 있습니다.
> - 태그에는 의미론적 의미가 없습니다. Amazon SQS는 태그를 문자열로 해석합니다.
> - 태그는 대/소문자를 구분합니다.
> - 키가 기존 태그와 동일한 새 태그는 기존 태그를 덮어씁니다.
> - 태깅 작업은 1인당 30TPS로 제한됩니다. AWS 계정애플리케이션에 더 많은 처리량이 필요한 경우 요청을 제출하십시오.
>
> 태그 제한 사항 전체 목록은 할당량 단원을 참조하십시오.

Amazon SQS 짧은 폴링 및 긴 폴링

Amazon SQS는 대기열에서 메시지를 수신하기 위한 단기 및 장기 폴링 옵션을 제공합니다. 다음 두 폴링 옵션 중에서 선택할 때는 응답성 및 비용 효율성에 대한 애플리케이션의 요구 사항을 고려하십시오.

- 짧은 폴링 (기본값) - `ReceiveMessage`요청은 서버의 하위 집합 (가중치 기반 무작위 분포 기반) 을 쿼리하여 사용 가능한 메시지를 찾고 메시지가 없는 경우에도 즉시 응답을 보냅니다.
- 긴 폴링 — 모든 서버에 메시지를 `ReceiveMessage`쿼리하여 하나 이상의 메시지를 사용할 수 있게 되면 지정된 최대값까지 응답을 보냅니다. 폴링 대기 시간이 만료된 경우에만 빈 응답이 전송됩니다. 이 옵션을 사용하면 빈 응답 수를 줄이고 잠재적으로 비용을 절감할 수 있습니다.

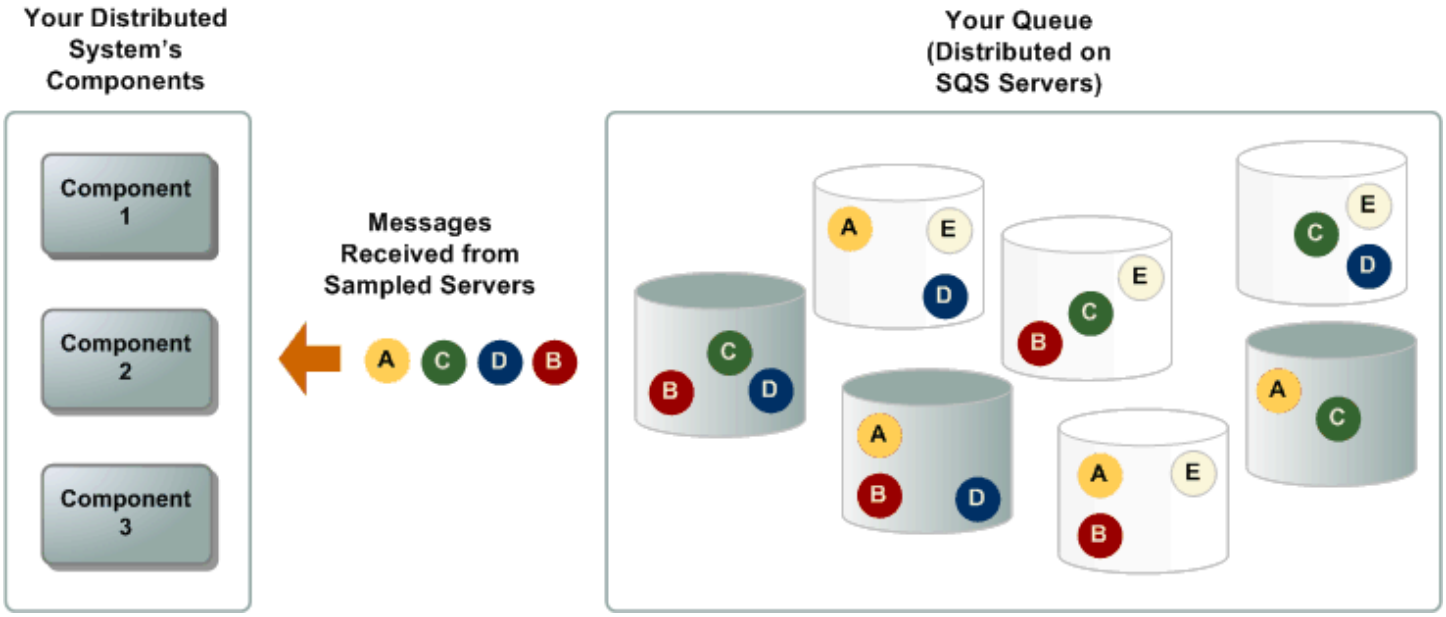
다음 섹션에서는 짧은 폴링과 긴 폴링의 세부 사항을 설명합니다.

주제

짧은 폴링을 사용한 메시지 사용

단축 폴링을 사용하여 대기열 (FIFO 또는 표준) 의 메시지를 소비하면 Amazon SQS는 가중치 기반 무작위 배포를 기반으로 해당 서버의 하위 집합을 샘플링하여 해당 서버에서만 메시지를 반환합니다. 따라서 특정 `ReceiveMessage` 요청은 모든 메시지를 반환하지 않을 수도 있습니다. 하지만 대기열의 메시지 수가 1,000개 미만인 경우 이후 요청은 메시지를 반환합니다. 대기열에서 계속 사용할 경우 Amazon SQS가 서버를 모두 샘플링하며, 사용자는 메시지를 모두 받게 됩니다.

다음 다이어그램은 시스템 구성 요소 중 하나가 수신 요청을 한 후 표준 대기열에서 반환된 메시지의 짧은 폴링 동작을 보여줍니다. Amazon SQS는 여러 서버(회색)를 샘플링하고 이러한 서버에서 메시지 A, C, D, B를 반환합니다. 메시지 E는 이 요청에서 반환되지 않지만 이후 요청에서 반환됩니다.



긴 폴링을 사용한 메시지 사용

`ReceiveMessage` API 작업에 대한 대기 시간이 0보다 큰 경우 긴 폴링이 유효합니다. 긴 폴링 대기 시간의 최대값은 20초입니다. 긴 폴링은 빈 응답 수를 줄이고(`ReceiveMessage` 요청에 사용 가능한 메시지가 없는 경우) 거짓의 빈 응답을 제거하여(메시지를 사용할 수 있지만 응답에 포함되지 않은 경우) Amazon SQS 사용 비용을 줄여 줍니다. Amazon SQS 콘솔을 사용하여 새 대기열 또는 기존 대기열에 긴 폴링을 활성화하는 방법에 대한 자세한 내용은 Amazon SQS 콘솔을 사용하여 대기열 파라미터 구성 섹션을 참조하세요. 모범 사례는 긴 폴링 설정 단원을 참조하세요.

긴 폴링은 다음과 같은 이점이 있습니다.

- 응답을 전송하기 전에 대기열에서 메시지를 사용할 수 있을 때까지 Amazon SQS를 대기시켜 빈 응답을 줄입니다. 연결 시간이 초과되지 않은 경우, `ReceiveMessage` 요청에 대한 응답은 하나 이상의 사용 가능한 메시지부터 `ReceiveMessage` 작업에 지정된 최대 메시지 개수까지 포함합니다. 드문 경우지만, 대기열에 여전히 메시지가 있는 경우에도 빈 응답을 받을 수 있습니다. 특히 `ReceiveMessageWaitTimeSeconds` 파라미터의 값을 낮게 지정한 경우 더욱 그렇습니다.
- Amazon SQS 서버의 하위 집합이 아닌 모든 서버를 쿼리하여 잘못된 빈 응답을 줄이세요.
- 사용 가능한 즉시 메시지를 반환합니다.

대기열이 비어 있는지 확인하는 방법에 대한 자세한 내용은 Amazon SQS 대기열이 비어 있는지 확인 섹션을 참조하세요.

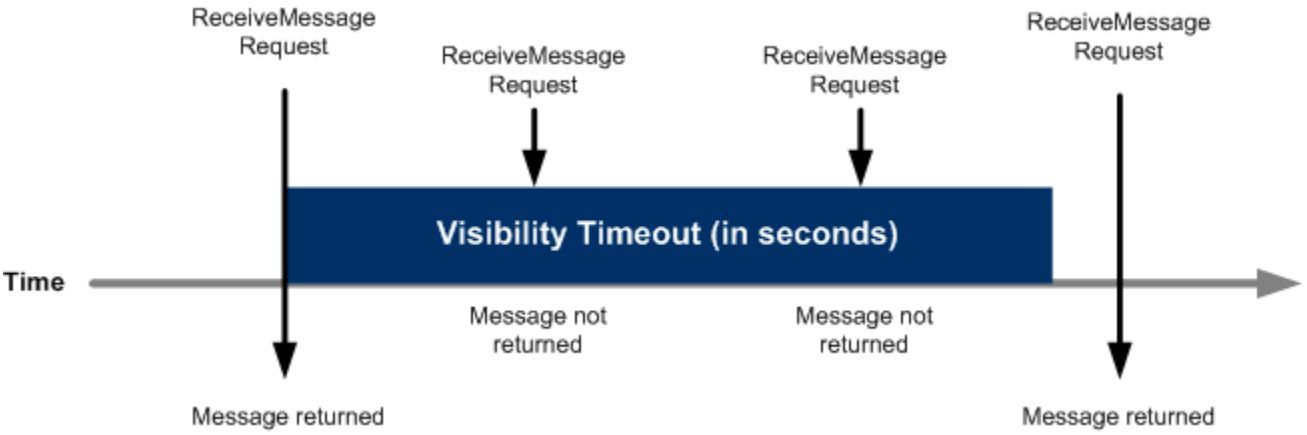
긴 폴링과 짧은 폴링의 차이점

짧은 폴링은 `WaitTimeSeconds` 요청의 `ReceiveMessage` 파라미터가 다음 두 가지 방식 중 하나에서 `0`으로 설정될 때 나타납니다.

- `ReceiveMessage` 호출이 `WaitTimeSeconds`를 `0`으로 설정합니다.
- `ReceiveMessage` 호출이 `WaitTimeSeconds`를 설정하지 않지만 대기열 속성 `ReceiveMessageWaitTimeSeconds`이 `0`으로 설정되어 있습니다.

Amazon SQS 제한 시간 초과

소비자가 대기열에서 메시지를 수신하고 처리하면 메시지는 계속 대기열에 있습니다. Amazon SQS는 메시지를 자동으로 삭제하지 않습니다. Amazon SQS는 분산 시스템이므로 소비자가 메시지를 실제로 받는지 보장할 수 없습니다(예를 들어, 연결 문제 또는 소비자 애플리케이션 문제로 인해). 또한, 소비자는 메시지를 수신하고 처리한 후 대기열에서 이 메시지를 삭제해야 합니다.



메시지를 수신한 직후에는 메시지가 대기열에 그대로 있습니다. 다른 소비자가 메시지를 다시 처리하는 것을 방지하기 위해, Amazon SQS에서 모든 소비자가 메시지를 수신하고 처리할 수 없도록 차단하는 기간인 제한 시간 초과를 설정합니다. 메시지의 제한 시간 초과는 30초입니다. 최소 시간은 0초입니다. 최대 시간은 12시간입니다. 콘솔을 사용하여 대기열의 제한 시간 초과를 구성하는 방법에 대한 자세한 내용은 Amazon SQS 콘솔을 사용하여 대기열 파라미터 구성 섹션을 참조하세요.

> **Note**
>
> 표준 대기열에서 제한 시간 초과는 메시지가 두 번 수신되지 않도록 보장하지 않습니다. 자세한 정보는 하나 이상의 전송을 참조하세요.
> FIFO 대기열에서는 다음과 같이 생산자 또는 소비자가 여러 번 재시도할 수 있도록 허용합니다.

- 생성자가 실패한 SendMessage 작업을 감지한 경우, 동일한 메시지 중복 제거 ID를 사용하여 필요한 횟수 만큼 여러 번 재전송을 시도할 수 있습니다. 생성자가 중복 제거 간격이 만료되기 전에 최소 한 개의 승인을 수신한다고 가정하면 여러 번 재시도를 해도 메시지의 순서에 영향을 미치지도 복제를 도입하지도 않습니다.
- 소비자가 실패한 ReceiveMessage 작업을 감지한 경우, 동일한 수신 요청 시도 ID를 사용하여 필요한 횟수 만큼 여러 번 재시도할 수 있습니다. 소비자가 제한 시간 초과가 만료되기 전에 최소 한 개의 승인을 수신한다고 가정하면 여러 번 재시도를 해도 메시지의 순서에는 영향을 미치지 않습니다.
- 메시지 그룹 ID가 있는 메시지를 수신한 경우, 동일한 메시지 그룹 ID에 대한 메시지는 그 메시지를 삭제하거나 메시지가 표시되는 경우가 아니라면 더 이상 반환되지 않습니다.

주제

처리 중인 메시지

Amazon SQS 메시지에는 세 가지 기본 상태가 있습니다.

1. 생산자가 대기열로 전송.
2. 소비자가 대기열에서 수신.
3. 대기열에서 삭제.

메시지는 생산자가 대기열로 전송했지만 소비자가 대기열에서 아직 수신하지 않은 후에(즉, 상태 1과 2 사이) 저장된 것으로 간주됩니다. 저장된 메시지 수에는 할당량이 없습니다. 메시지는 소비자가 대기열에서 수신했지만 아직 대기열에서 삭제되지 않은 경우(즉, 상태 2와 3 사이) 처리 중인 것으로 간주됩니다. 처리 중인 메시지 수에는 할당량이 있습니다.

> **Important**
>
> 처리 중인 메시지에 적용되는 할당량은 저장된 메시지 수의 무제한과 관련이 없습니다.

대부분의 표준 대기열(대기열 트래픽 및 메시지 백로그에 따라 다름)의 경우 최대 약 120,000개의 이동 중 메시지(소비자가 대기열에서 수신했지만 대기열에서 아직 삭제되지 않은 메시지)가 있을 수 있습니다. 짧은 폴링을 사용하는 동안 이 할당량에 도달하면 Amazon SQS에서 `OverLimit` 오류 메시지를 반환합니다. 긴 폴링을 사용하는 경우에는 Amazon SQS에서 오류 메시지를 반환하지 않습니다. 이 할당량에 도달하지 않도록 하려면 메시지를 처리한 후 대기열에서 삭제해야 합니다. 또한 메시지를 처리하는 데 사용하는 대기열의 수를 늘릴 수 있습니다. 할당량 증가를 요청하려면 지원 요청을 제출하십시오.

FIFO 대기열의 경우 최대 20,000개의 이동 중인 메시지(소비자가 대기열에서 수신했지만 대기열에서 아직 삭제되지 않은 메시지)가 있을 수 있습니다. 이 할당량에 도달해도 Amazon SQS에서는 오류 메시지를 반환하지 않습니다.

> **Important**
>
> FIFO 대기열로 작업할 때 제한 시간 초과 기간 외에 요청이 수신되면 `DeleteMessage` 작업이 실패합니다. 제한 시간 초과가 0초인 경우 메시지는 전송된 밀리초 이내에 삭제되어야 하며, 그렇지 않으면 중단된 것으로 간주됩니다. 이로 인해 `MaxNumberOfMessages` 파라미터가 1보다 큰 경우 Amazon SQS가 `ReceiveMessage` 작업에 대한 동일한 응답에 중복 메시지를 포함할 수 있습니다. 자세한 내용은 Amazon SQS FIFO API 작동 방식을 참조하세요.

제한 시간 초과 설정

제한 시간 초과는 Amazon SQS가 메시지를 반환할 때 시작됩니다. 이 시간 동안 소비자는 메시지를 처리하고 삭제합니다. 하지만 소비자가 메시지를 삭제하기 전에 실패할 경우 제한 시간 초과가 만료되기 전에 시스템에서 해당 메시지에 대한 `DeleteMessage` 작업을 호출하지 않으면 다른 소비자가 메시지를 볼 수 있게 되고 메시지가 다시 수신됩니다. 메시지가 한 번만 수신되어야 하는 경우 소비자는 제한 시간 초과 기간 내에 메시지를 삭제해야 합니다.

모든 Amazon SQS 대기열에는 30초의 기본 제한 시간 초과 설정이 있습니다. 전체 대기열에 대해 이 설정을 변경할 수 있습니다. 일반적으로 애플리케이션에서 대기열의 메시지를 처리하고 삭제하는 데 소요되는 최대 시간으로 제한 시간 초과를 설정해야 합니다. 메시지를 수신할 때 전체 대기열 제한 시간을 변경하지 않고도 반환된 메시지에 대해 특별한 제한 시간 초과를 설정할 수도 있습니다. 자세한 내용은 적시에 메시지 처리 단원의 모범 사례를 참조하십시오.

메시지를 처리하는 데 걸리는 시간을 모르는 경우 소비자 프로세스에 대한 하트비트를 생성합니다. 초기 표시 제한 시간(예: 2분)을 지정한 다음 소비자가 메시지에서 계속 작업하는 경우 표시 제한 시간을 1분마다 2분씩 연장합니다.

Important

최대 표시 제한 시간은 Amazon SQS가 `ReceiveMessage` 요청을 수신한 시간으로부터 12시간입니다. 표시 제한 시간을 연장해도 최대 12시간이 재설정되지는 않습니다.
또한 `ReceiveMessage` 요청으로 타이머가 시작되기 때문에 개별 메시지의 제한 시간을 전체 12시간(예: 43,200초)으로 설정하지 못할 수도 있습니다. 예를 들어 메시지를 수신한 후 `VisibilityTimeout`이 43,200초인 `ChangeMessageVisibility` 호출을 전송하여 즉시 최대 12시간을 설정하면 실패할 가능성이 높습니다. 하지만 43,195초 값을 사용하면 `ReceiveMessage`를 통해 메시지를 요청하는 것과 표시 제한 시간을 업데이트하는 것 사이에 상당한 지연이 있는 경우가 아니면 작동합니다. 소비자가 12시간 이상 필요한 경우 Step Functions 사용을 고려하세요.

메시지에 대한 제한 시간 초과 변경

대기열에서 메시지를 수신하여 처리하기 시작하다 보면 이 대기열의 표시 제한 시간이 부족해질 수 있습니다(처리한 메시지를 삭제해야 하는 경우도 있음). 메시지의 제한 시간을 단축하거나 늘리기 위해 `ChangeMessageVisibility` 작업을 사용하여 새 제한 시간 값을 지정할 수 있습니다.

예를 들어 대기열의 시간 제한이 60초이고 메시지를 받은 뒤로 15초가 흘렀는데 `VisibilityTimeout`을 10초로 설정하여 `ChangeMessageVisibility` 호출을 보내면, `ChangeMessageVisibility` 호출 시점으로부터 10초를 세기 시작합니다. 따라서 표시 제한 시간(총 25초)을 처음 변경하고 10초가 지난 뒤에 이를 변경하려 하거나 메시지를 삭제하려고 하면 오류가 발생합니다.

Note

새로운 제한 시간은 `ChangeMessageVisibility` 작업을 호출한 시간부터 적용됩니다. 또한, 새 제한 시간은 특정 메시지 수신에만 적용됩니다. `ChangeMessageVisibility`는 메시지 수신이나 이후 대기열의 제한 시간에 영향을 미치지 않습니다.

메시지에 대한 제한 시간 초과 종료

대기열에서 메시지를 수신할 때 실제로 해당 메시지를 처리 및 삭제하지 않아도 됨을 알 수 있습니다. Amazon SQS를 통해 특정 메시지의 제한 시간 초과를 종료할 수 있습니다. 그리고 나면 메시지가 시스템의 다른 구성 요소에 즉시 표시되어 처리할 수 있습니다.

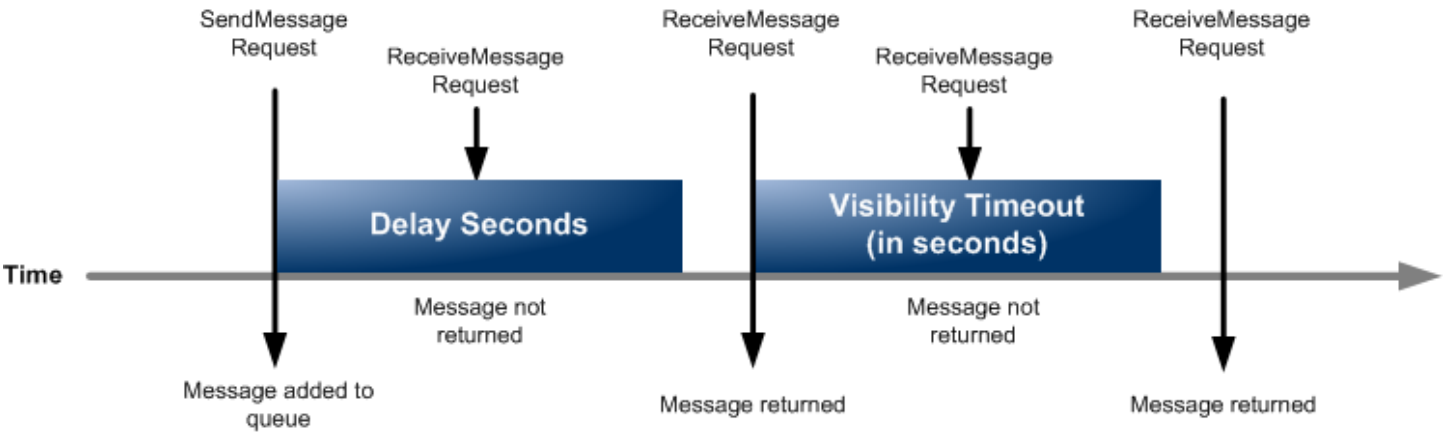
`ReceiveMessage`를 호출한 후 메시지의 제한 시간 초과를 종료하려면 `VisibilityTimeout`을 0초로 설정하여 `ChangeMessageVisibility`를 호출하세요.

Amazon SQS 지연 대기열

지연 대기열을 사용하면 새 메시지를 소비자에게 전송하는 것을 특정 시간(초) 동안 연기할 수 있습니다(예: 소비자 애플리케이션에서 메시지를 처리하기 위해 추가 시간이 필요한 경우). 지연 대기열을 생성할 경우, 이 대기열로 전송하는 모든 메시지는 지연 기간 동안 소비자에게 표시되지 않습니다. 대기열의 기본(최소) 지연 시간은 0초입니다. 최대값은 15분입니다. 콘솔을 사용해 지연 대기열을 구성하는 방법에 대한 자세한 내용은 Amazon SQS 콘솔을 사용하여 대기열 파라미터 구성 단원을 참조하세요.

> **Note**
>
> 표준 대기열에서 대기열당 지연 설정은 소급 적용되지 않습니다. 설정을 변경할 경우, 이것은 대기열에 이미 있는 메시지의 지연에 영향을 주지 않습니다.
> FIFO 대기열에서 대기열당 지연 설정은 소급 적용됩니다. 설정을 변경할 경우, 이것은 대기열에 이미 있는 메시지의 지연에 영향을 줍니다.

지연 대기열은 두 기능 모두가 소비자가 특정 기간 동안 사용하지 못하도록 하기 때문에 제한 시간 초과와 비슷합니다. 두 속성의 차이는 지연 대기열은 처음 대기열에 추가될 때 메시지가 숨겨지지만, 제한 시간 초과는 대기열에서 사용된 후에만 메시지가 숨겨진다는 것입니다. 다음 그림은 지연 대기열과 제한 시간 초과의 관계를 잘 보여줍니다.



전체 대기열이 아니라 개별 메시지에 대해 지연 시간(초)을 설정하려면 메시지 타이머를 사용하여 Amazon SQS가 지연 대기열의 `DelaySeconds` 값 대신 메시지 타이머의 `DelaySeconds` 값을 사용하도록 허용합니다.

Amazon SQS 임시 대기열

임시 대기열을 사용하면 request-response와 같은 공통 메시지 패턴을 사용할 때 개발 시간 및 배포 비용을 줄일 수 있습니다. 임시 대기열 클라이언트를 사용하여 대량고속처리가 가능하고 비용 효율적인 애플리케이션 관리 임시 대기열을 생성할 수 있습니다.

클라이언트는 여러 개의 임시 대기열(특정 프로세스에 대한 요구 시 생성되는 애플리케이션 관리형 대기열)을 하나의 Amazon SQS 대기열로 자동으로 매핑합니다. 따라서 애플리케이션에서 API 호출 수를 줄이고 각 임시 대기열에 대한 트래픽이 낮을 때도 높은 처리량을 얻을 수 있습니다. 임시 대기열을 더 이상 사용하지 않을 때는 클라이언트를 사용하는 프로세스가 명확하게 종료된 경우라 하더라도 클라이언트가 임시 대기열을 자동으로 정리합니다.

임시 대기열은 다음과 같은 이점이 있습니다.

- 임시 대기열은 특정 스레드 또는 프로세스에 대해 경량 통신 채널 역할을 합니다.
- 또한 추가 비용 없이도 생성 및 삭제가 가능합니다.
- 이는 정적(정상) Amazon SQS 대기열과 API 호환이 가능합니다. 따라서 메시지를 전송 및 수신하는 기존 코드가 가상 대기열과 메시지를 주고 받을 수 있습니다.

주제

가상 대기열

가상 대기열은 임시 대기열 클라이언트가 생성하는 로컬 데이터 구조입니다. 가상 대기열을 사용하면 Amazon SQS 대기열에 트래픽이 낮은 대상을 여러 개 결합할 수 있습니다. 모범 사례는 가상 대기열에서 동일한 메시지 그룹 ID 재사용 방지 단원을 참조하세요.

> **Note**
>
> - 가상 대기열을 생성하면 소비자가 들어오는 메시지를 수신할 수 있는 임시 데이터 구조만 생성됩니다. 가상 대기열은 Amazon SQS에 대해 API 호출을 수행하지 않기 때문에 어떤 비용도 발생하지 않습니다.
> - 한 호스트 대기열 내의 모든 가상 대기열에는 TPS 할당량이 적용됩니다. 자세한 정보는 Amazon SQS 메시지 할당량을 참조하세요.

`AmazonSQSVirtualQueuesClient` 래퍼 클래스는 가상 대기열과 관련된 속성 지원을 추가합니다. 가상 대기열을 생성하려면 `HostQueueURL` 속성을 사용하여 `CreateQueue` API 작업을 호출해야 합니다. 이 속성은 가상 대기열을 호스팅하는 기존 대기열을 지정합니다.

가상 대기열의 URL은 다음 형식을 따릅니다.

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

생산자가 가상 대기열 URL에서 `SendMessage` 또는 `SendMessageBatch` API 작업을 호출하면 임시 대기열 클라이언트가 다음과 같은 작업을 수행합니다.

1. 가상 대기열 이름을 추출합니다.
2. 가상 대기열 이름을 추가 메시지 속성으로 연결합니다.
3. 호스트 대기열에 이 메시지를 전송합니다.

생산자가 메시지를 전송할 때 백그라운드 스레드는 호스트 대기열을 폴링하고, 해당되는 메시지 속성에 따라 가상 대기열에 수신한 메시지를 전송합니다.

소비자가 가상 대기열 URL에서 `ReceiveMessage` API 작업을 호출하는 동안 임시 대기열 클라이언트는 백그라운드 스레드가 가상 대기열에 메시지를 전송할 때까지 로컬 호출을 차단합니다. (이 프로세스는 버퍼링된 비동기식 클라이언트에서의 메시지 미리 가져오기와 비슷한데, 단일 API 작업이 최대 10개의 가상 대기열에 메시지를 전송하는 것이 가능합니다). 가상 대기열을 삭제하면 Amazon SQS 자체를 호출하지 않고 클라이언트측 리소스가 제거됩니다.

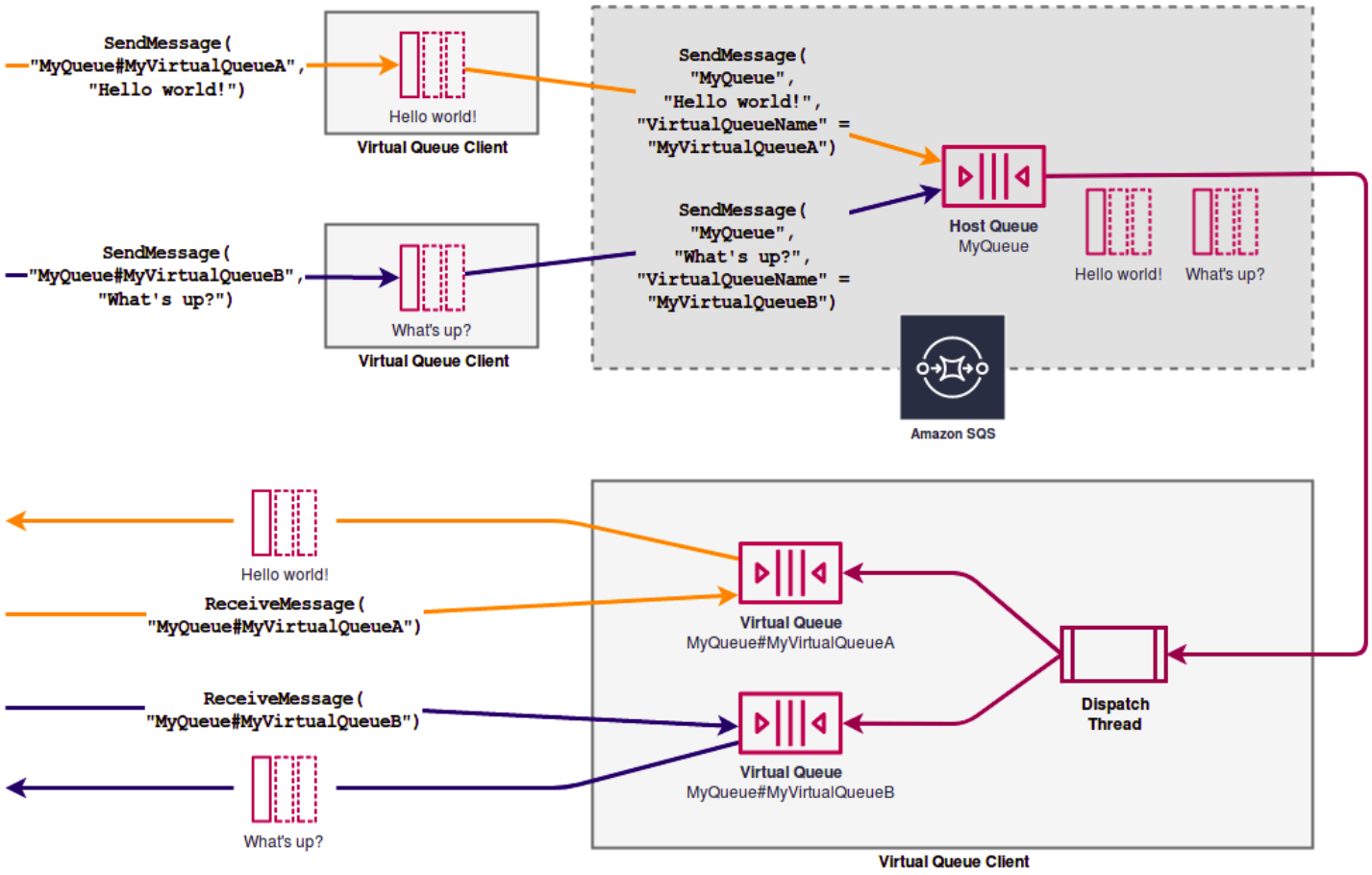
`AmazonSQSTemporaryQueuesClient` 클래스는 생성한 모든 대기열을 임시 대기열로 자동 전환합니다. 또한 요구 시 같은 대기열 속성을 가진 호스트 대기열을 자동 생성합니다. 이러한 대기열의 이름은 임시 대기열로 식별하는 공통된 구성 가능 접두사(기본적으로 `__RequesterClientQueues__`)

를 공유합니다. 따라서 클라이언트는 대기열을 생성 및 삭제하는 기존 코드를 최적화하는 즉각적인 대체물 역할을 할 수 있습니다. 또한 클라이언트에는 대기열 간에 양방향 통신을 가능하게 하는 `AmazonSQSRequester` 및 `AmazonSQSResponder` 인터페이스가 포함되어 있습니다.

요청-응답 메시징 패턴(가상 대기열)

임시 대기열의 가장 일반적인 사용 사례는 요청-응답 메시징 패턴인데, 여기에서는 요청이 각 응답 메시지를 수신하기 위해 임시 대기열을 생성합니다. 각 응답 메시지에 대해 Amazon SQS 대기열을 생성하는 것을 방지하기 위해 임시 대기열 클라이언트는 어떤 Amazon SQS API도 호출하지 않고 여러 개의 임시 대기열을 생성 및 삭제할 수 있도록 해줍니다. 자세한 정보는 요청-응답 시스템 구현을 참조하세요.

다음 다이어그램은 이 패턴을 사용하는 공통 구성을 보여줍니다.



예제 시나리오: 로그인 요청 처리

다음 예제 시나리오는 `AmazonSQSRequester` 및 `AmazonSQSResponder` 인터페이스를 사용하여 사용자의 로그인 요청을 처리할 수 있는 방법을 보여줍니다.

클라이언트 측

```
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();
    }
}
```

로그인 요청을 전송하는 과정은 다음과 같습니다.

1. 임시 대기열을 생성합니다.
2. 임시 대기열의 URL을 메시지에 속성으로써 연결합니다.
3. 이 메시지를 전송합니다.
4. 임시 대기열에서 응답을 수신합니다.
5. 임시 대기열을 삭제합니다.
6. 응답을 반환합니다.

서버 측

이 예제에서는 구성 시 대기열을 폴링하고 모든 메시지에 대해 handleLoginRequest() 메서드를 호출하기 위해 스레드가 생성된다고 가정합니다. 게다가 doLogin()는 위임된 메서드입니다.

```
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));
    }
}
```

대기열 정리

Amazon SQS가 이 가상 대기열에서 사용되는 인 메모리 리소스를 회수하는지 확인하려면 애플리케이션에서 임시 대기열 클라이언트가 더 이상 필요하지 않을 때 shutdown() 메서드를 호출해야 합니다. 또한 AmazonSQSRequester 인터페이스의 shutdown() 메서드를 사용할 수도 있습니다.

임시 대기열 클라이언트는 사용하지 않는 호스트 대기열을 제거할 수 있는 방법도 제공합니다. 일정 기간(기본적으로 5분) 동안 API 호출을 수신하는 각 대기열에서 클라이언트는 TagQueue API 작업을 사용하여 사용 중인 상태로 남아 있는 대기열에 태그 지정을 합니다.

> **Note**
>
> `ReceiveMessage` 작업을 포함하여 대기열에서 이루어진 모든 API 작업은 대기열을 비유휴 상태로 표시합니다.

백그라운드 스레드는 `ListQueues` 및 `ListTags` API 작업을 사용하여 접두사가 구성된 모든 대기열을 확인하고 최소 5분 동안 태그 지정이 되지 않은 대기열을 모두 삭제합니다. 이 과정에서 한 클라이언트가 명확하게 종료되지 않으면 다른 활성 클라이언트가 종료 이후에 정리합니다. 작업 중복을 줄이기 위해 같은 접두사를 가진 모든 클라이언트가 접두사 뒤에 명명된 내부의 공유 대기열을 통해 통신합니다.

Amazon SQS 메시지 타이머

메시지 타이머를 사용하여 대기열에 추가되는 메시지의 초기 미표시 기간을 지정합니다. 예를 들어, 45초 타이머를 사용하여 메시지를 전송할 경우에 이 메시지가 대기열에 머무르는 처음 45초 동안 소비자에게 표시되지 않습니다. 메시지의 기본(최소) 지연 시간은 0초입니다. 최대값은 15분입니다. 콘솔을 사용하여 타이머가 있는 메시지를 전송하는 방법에 대한 자세한 내용은 메시지 전송 섹션을 참조하세요.

> **Note**
>
> FIFO 대기열은 개별 메시지의 타이머를 지원하지 않습니다.

개별 메시지가 아니라 전체 대기열에 지연을 설정하려면 지연 대기열을 사용하십시오. 개별 메시지에 대한 타이머 설정은 Amazon SQS 지연 대기열의 `DelaySeconds` 값을 재정의합니다.

Amazon SQS EventBridge 콘솔을 통해 아마존 파이프에 액세스

Amazon EventBridge Pipes는 소스를 타겟에 연결합니다. 파이프는 고급 변환 및 강화를 지원하여 지원되는 소스와 대상 간의 point-to-point 통합을 위한 것입니다. EventBridge 파이프는 Amazon SQS 대기열을 Step Functions, Amazon SQS, API Gateway와 같은 AWS 서비스뿐만 아니라 Salesforce와 같은 타사 서비스형 소프트웨어 (SaaS) 애플리케이션에 연결할 수 있는 확장성이 뛰어난 방법을 제공합니다.

파이프를 설정하려면 소스를 선택하고, 선택적 필터링을 추가하고, 선택적 보강을 정의하고, 이벤트 데이터의 대상을 선택합니다.

Amazon SQS 대기열의 세부 정보 페이지에서 해당 대기열을 소스로 사용하는 파이프를 볼 수 있습니다. 다음을 수행할 수도 있습니다.

- EventBridge 콘솔을 실행하여 파이프 세부 정보를 확인하십시오.
- EventBridge 콘솔을 실행하여 큐를 소스로 사용하여 새 파이프를 생성합니다.

Amazon SQS 대기열을 파이프 소스로 구성하는 방법에 대한 자세한 내용은 Amazon 사용 설명서의 Amazon SQS 대기열을 소스로 구성하는 방법을 참조하십시오. EventBridge 일반적으로 EventBridge 파이프에 대한 자세한 내용은 파이프를 참조하십시오EventBridge .

지정된 Amazon SQS 대기열의 EventBridge 파이프에 액세스하려면

1. Amazon SQS 콘솔의 대기열 페이지를 엽니다.
2. 대기열을 선택합니다.
3. 대기열 세부 정보 페이지에서 EventBridge 파이프 탭을 선택합니다.

   EventBridge 파이프 탭에는 선택한 큐를 소스로 사용하도록 현재 구성된 파이프 목록이 포함되어 있습니다. 여기에는 다음이 포함됩니다.

   - 파이프 이름
   - 현재 상태
   - 파이프 대상
   - 파이프를 마지막으로 수정한 시점
4. 원하는 경우 더 많은 파이프 세부 정보를 보거나 새 파이프를 만듭니다.

   - 파이프에 대한 추가 세부 정보에 액세스:

     파이프 이름을 선택합니다.

     그러면 EventBridge 콘솔의 파이프 세부정보 페이지가 열립니다.
   - 새 파이프를 생성하려면:

     파이프에 Amazon SQS 대기열 연결을 선택합니다.

     그러면 Amazon SQS 대기열이 파이프 소스로 지정된 EventBridge 콘솔의 Create pipe 페이지가 시작됩니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 EventBridge 파이프 생성을 참조하십시오.

> **Important**
>
> Amazon SQS 대기열의 메시지는 메시지가 해당 파이프에 대해 구성할 수 있는 필터와 일치하는지 여부에 관계없이 단일 파이프로 읽힌 다음 처리된 후 대기열에서 삭제됩니다. 동일한 대기열을 소스로 사용하도록 여러 파이프를 구성할 때는 주의하세요.

확장 클라이언트 라이브러리 및 Amazon 심플 스토리지 서비스를 통한 대용량 Amazon SQS 메시지 관리

Java용 Amazon SQS 확장 클라이언트 라이브러리와 Python용 Amazon SQS 확장 클라이언트 라이브러리를 사용하여 대용량 메시지를 전송할 수 있습니다. 이는 256KB에서 최대 2GB에 이르는 대용량 메시지 페이로드를 소비하는 데 특히 유용합니다. 두 라이브러리 모두 메시지 페이로드를 Amazon 심플 스토리지 서비스 버킷에 저장하고 저장된 Amazon S3 객체의 참조를 Amazon SQS 대기열로 보냅니다.

> **Note**
>
> Amazon SQS 확장 클라이언트 라이브러리는 표준 및 FIFO 대기열 모두와 호환됩니다.

주제

- 자바와 아마존 S3를 사용하여 대용량 Amazon SQS 메시지 관리
- Python과 Amazon S3를 사용하여 대용량 Amazon SQS 메시지 관리

자바와 아마존 S3를 사용하여 대용량 Amazon SQS 메시지 관리

자바용 Amazon SQS 확장 클라이언트 라이브러리와 아마존 심플 스토리지 서비스 (Amazon S3) 를 사용하여 대용량 아마존 심플 큐 서비스 (Amazon SQS) 메시지를 관리할 수 있습니다. 이는 256KB에서 최대 2GB에 이르는 대용량 메시지 페이로드를 소비할 때 특히 유용합니다. 라이브러리는 메시지 페이로드를 Amazon S3 버킷에 저장하고 저장된 Amazon S3 객체의 참조가 포함된 메시지를 Amazon SQS 대기열로 보냅니다.

Java용 Amazon SQS 확장 클라이언트 라이브러리를 사용하여 다음을 수행할 수 있습니다.

- 메시지를 항상 Amazon S3에만 저장할지를 지정하거나, 메시지의 크기가 256KB를 초과할 때만 저장할지를 지정합니다.
- S3 버킷에 저장된 단일 메시지 객체를 참조하는 메시지를 전송합니다.
- Amazon S3 버킷에서 메시지 객체를 검색합니다.
- Amazon S3 버킷에서 메시지 객체를 삭제합니다.

사전 조건

다음 예시에서는 AWS Java SDK를 사용합니다. SDK를 설치하고 설정하려면 개발자 안내서의 Java용 AWS SDK 설정을 참조하십시오.AWS SDK for Java

예제 코드를 실행하기 전에 자격 증명을 구성하십시오. AWS 자세한 내용은 AWS SDK for Java 개발자 안내서의 개발용 AWS 자격 증명 및 지역 설정을 참조하십시오.

Java용 SDK 및 Java용 Amazon SQS 확장 클라이언트 라이브러리를 사용하려면 J2SE 개발 키트 8.0 이상이 필요합니다.

> **Note**
>
> Java용 Amazon SQS 확장 클라이언트 라이브러리를 사용하면 Amazon S3에 AWS SDK for Java만 사용하여 Amazon SQS 메시지를 관리할 수 있습니다. Amazon SQS 콘솔 AWS CLI, Amazon SQS HTTP API 또는 기타 SDK로는 이 작업을 수행할 수 없습니다. AWS

AWS 자바 2.x용 SDK 예제: Amazon S3를 사용하여 대용량 아마존 SQS 메시지 관리

다음 Java 2.x용AWS SDK 예제는 임의의 이름으로 Amazon S3 버킷을 생성하고 14일 후에 객체를 영구적으로 삭제하는 수명 주기 규칙을 추가합니다. 또한 이름이 MyQueue인 대기열을 생성하고 S3 버킷에 저장된 256KB 이상의 무작위 메시지를 대기열로 보냅니다. 마지막으로 이 코드는 메시지를 검색하고, 메시지에 대한 정보를 반환한 후 메시지, 대기열 및 버킷을 삭제합니다.

```
/*
 * 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 자바 2.x용 SDK 예제: Amazon S3를 사용하여 대용량 아마존 SQS 메시지 관리

다음 Java 2.x용AWS SDK 예제는 임의의 이름으로 Amazon S3 버킷을 생성하고 14일 후에 객체를 영구적으로 삭제하는 수명 주기 규칙을 추가합니다. 또한 이름이 MyQueue인 대기열을 생성하고 S3 버킷에 저장된 256KB 이상의 무작위 메시지를 대기열로 보냅니다. 마지막으로 이 코드는 메시지를 검색하고, 메시지에 대한 정보를 반환한 후 메시지, 대기열 및 버킷을 삭제합니다.

```
/*
 * 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());
    }
}
```

Apache Maven을 사용하여 자바 프로젝트용 Amazon SQS 확장 클라이언트를 구성 및 구축하거나 SDK 자체를 구축할 수 있습니다. 애플리케이션에서 사용하는 SDK의 개별 모듈을 지정하십시오.

```
<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>
```

Python과 Amazon S3를 사용하여 대용량 Amazon SQS 메시지 관리

Python용 Amazon 심플 큐 서비스 확장 클라이언트 라이브러리와 Amazon 심플 스토리지 서비스를 사용하여 대용량 Amazon SQS 메시지를 관리할 수 있습니다. 이는 256KB에서 최대 2GB에 이르는 대용량 메시지 페이로드를 소비하는 데 특히 유용합니다. 라이브러리는 메시지 페이로드를 Amazon S3 버킷에 저장하고 저장된 Amazon S3 객체의 참조가 포함된 메시지를 Amazon Amazon SQS 대기열로 보냅니다.

Python용 확장 클라이언트 라이브러리를 사용하여 다음을 수행할 수 있습니다.

- 페이로드를 항상 Amazon S3에 저장할지, 아니면 페이로드 크기가 256KB를 초과하는 경우에만 S3에 저장할지를 지정합니다.
- Amazon S3 버킷에 저장된 단일 메시지 객체를 참조하는 메시지 전송
- Amazon S3 버킷에서 해당 페이로드 객체를 검색합니다.
- Amazon S3 버킷에서 해당 페이로드 객체를 삭제합니다.

사전 조건

Python용 Amazon SQS 확장 클라이언트 라이브러리를 사용하기 위한 사전 요구 사항은 다음과 같습니다.

- 필요한 자격 AWS 증명이 있는 계정. AWS 계정을 만들려면 AWS 홈페이지로 이동한 다음 AWS 계정 만들기를 선택합니다. 지침을 따릅니다. 자격 증명에 대한 자세한 내용은 자격 증명을 참조하십시오.
- AWS SDK: 이 페이지의 예시에서는 AWS Python SDK Boto3를 사용합니다. SDK를 설치하고 설정하려면 Python용 SDK 개발자 AWS 안내서의 AWS Python용 SDK 설명서를 참조하십시오.
- Python 3.x (또는 그 이상) 및 `pip`
- PyPI에서 사용 가능한 Python용 Amazon SQS 확장 클라이언트 라이브러리

> **Note**
>
> Python용 Amazon SQS 확장 클라이언트 라이브러리를 사용하면 Amazon S3를 사용하는 경우에만 Python용 AWS SDK를 사용하여 Amazon SQS 메시지를 관리할 수 있습니다. AWS CLI, Amazon SQS 콘솔, Amazon SQS HTTP API 또는 기타 SDK로는 이 작업을 수행할 수 없습니다. AWS

메시지 스토리지 구성

Amazon SQS 확장 클라이언트는 다음 메시지 속성을 사용하여 Amazon S3 메시지 스토리지 옵션을 구성합니다.

- `large_payload_support`: 대용량 메시지를 저장하기 위한 Amazon S3 버킷 이름입니다.
- `always_through_s3`: `True` 그렇다면 모든 메시지가 Amazon S3에 저장됩니다. 경우`False`, 256KB보다 작은 메시지는 s3 버킷으로 직렬화되지 않습니다. 기본값은 `False`입니다.
- `use_legacy_attribute`: 경우`True`, 게시된 모든 메시지는 현재 예약된 메시지 속성 (`SQSLargePayloadSize`) 대신 레거시 예약 메시지 속성 (`ExtendedPayloadSize`) 을 사용합니다.

Python용 확장 클라이언트 라이브러리를 사용하여 대규모 Amazon SQS 메시지 관리

다음 예제는 임의의 이름을 가진 Amazon S3 버킷을 생성합니다. 그런 다음 이름이 지정된 `MyQueue` Amazon SQS 대기열을 생성하고 S3 버킷에 저장된 256KB가 넘는 메시지를 해당 대기열로 보냅니다. 마지막으로 이 코드는 메시지를 검색하고, 메시지에 대한 정보를 반환한 후 메시지, 대기열 및 버킷을 삭제합니다.

```
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
```

Amazon SQS 콘솔을 사용하여 Amazon SQS 대기열을 구성합니다.

Amazon SQS 콘솔을 사용하여 Amazon Simple Queue Service(Amazon SQS) 대기열 및 기능을 구성하고 관리할 수 있습니다. 콘솔을 사용하여 서버 측 암호화와 같은 기능을 구성하거나 데드레터 큐를 큐에 연결하거나 함수를 호출하도록 트리거를 설정할 수도 있습니다. AWS Lambda

주제

Amazon SQS의 속성 기반 액세스 제어

ABAC란 무엇입니까?

속성 기반 액세스 제어 (ABAC) 는 사용자 및 리소스에 연결된 태그를 기반으로 권한을 정의하는 권한 부여 프로세스입니다. AWS ABAC는 속성 및 값을 기반으로 세분화되고 유연한 액세스 제어를 제공하고 재구성된 역할 기반 정책과 관련된 보안 위험을 줄이며 감사 및 액세스 정책 관리를 중앙 집중화합니다. ABAC에 대한 자세한 내용은 IAM 사용 설명서의 AWS용 ABAC란 무엇입니까? 섹션을 참조하세요.

Amazon SQS는 Amazon SQS 대기열과 연결된 태그 및 별칭을 기반으로 Amazon SQS 대기열에 대한 액세스를 제어할 수 있도록 하여 ABAC를 지원합니다. Amazon SQS에서 ABAC를 활성화하는 태그 및 별칭 조건 키는 정책을 편집하거나 권한 부여를 관리하지 않고도 IAM 보안 주체가 Amazon SQS 대기열을 사용할 수 있는 권한을 부여합니다.

ABAC를 사용하면 태그를 사용하여 Amazon SQS 대기열에 대한 IAM 액세스 권한 및 정책을 구성할 수 있으며, 이는 권한 관리의 규모를 조정하는 데 도움이 됩니다. 새 리소스를 추가할 때마다 정책을 업데이트할 필요 없이 각 비즈니스 역할에 추가하는 태그를 사용하여 IAM에서 단일 권한 정책을 생성할 수 있습니다. 또한 IAM 보안 주체에 태그를 추가하여 ABAC 정책을 생성할 수 있습니다. 호출하는 IAM 사용자 역할의 태그가 Amazon SQS 대기열 태그와 일치할 때 Amazon SQS 작업을 허용하도록 ABAC 정책을 설계할 수 있습니다. 태그 지정에 대한 자세한 내용은 태깅 전략 AWS및 을 참조하십시오AWS . Amazon SQS 비용 할당 태그

> **Note**
>
> Amazon SQS용 ABAC는 현재 Amazon SQS를 사용할 수 있는 AWS 모든 상업 지역에서 사용할 수 있습니다. 단, 다음과 같은 예외가 있습니다.
>
> - 아시아 태평양(하이데라바드)
> - 아시아 태평양(멜버른)
> - 유럽(스페인)
> - 유럽(취리히)

Amazon SQS에 ABAC를 사용해야 하는 이유는 무엇입니까?

Amazon SQS에서 ABAC를 사용할 때 얻을 수 있는 몇 가지 이점은 다음과 같습니다.

- Amazon SQS의 ABAC에는 필요한 권한 정책이 적습니다. 각 직무에 서로 다른 정책을 생성할 필요가 없습니다. 둘 이상의 대기열에 적용되는 리소스 및 요청 태그를 사용할 수 있으므로 운영 오버헤드가 줄어듭니다.
- ABAC를 사용하여 팀을 빠르게 확장할 수 있습니다. 리소스 생성 중에 리소스에 태그가 적절하게 지정되면 새 리소스에 대한 권한이 태그에 따라 자동으로 부여됩니다.
- IAM 보안 주체에 대한 권한을 사용하여 리소스 액세스를 제한할 수 있습니다. IAM 보안 주체용 태그를 생성하고 이를 사용하여 IAM 보안 주체의 태그와 일치하는 특정 작업에 대한 액세스를 제한할 수 있습니다. 이렇게 하면 요청 권한 부여 프로세스를 자동화할 수 있습니다.
- 누가 리소스에 액세스하는지 추적할 수 있습니다. AWS CloudTrail에서 사용자 속성을 살펴봄으로써 세션의 ID를 확인할 수 있습니다.

주제

- Amazon SQS에 사용되는 ABAC 조건 키

- Amazon SQS에서의 액세스 제어를 위한 태깅
- IAM 사용자 및 Amazon SQS 대기열 생성
- 속성 기반 액세스 제어 테스트

Amazon SQS에 사용되는 ABAC 조건 키

다음 조건 키를 사용하여 함수 작업을 제어할 수 있습니다.

ABAC 조건 키	설명	정책 유형	Amazon SQS 작업
aws: ResourceTag	Amazon SQS 대기열의 태그(키 및 값)가 정책의 태그 (키 및 값) 또는 태그 패턴과 일치합니다.	IAM 정책만	Amazon SQS 대기열 리소스 작업
예: RequestTag	Amazon SQS 대기열 리소스 작업의 태그(키 및 값)가 정책의 태그 (키 및 값) 또는 태그 패턴과 일치합니다.	대기열 정책 및 IAM 정책	TagQueue, UntagQueue, CreateQueue
예: TagKeys	요청의 태그 키가 정책의 태그 키와 일치합니다.	대기열 정책 및 IAM 정책	TagQueue, UntagQueue, CreateQueue

Amazon SQS에서의 액세스 제어를 위한 태깅

다음은 액세스 제어에 태그를 사용하는 방법의 예입니다. IAM 정책은 IAM 사용자를 키 'environment' 및 값 'production'인 리소스 태그를 포함하는 모든 대기열에 대한 모든 Amazon SQS 작업으로 제한합니다. 자세한 내용은 태그 및 AWS Organizations를 사용한 속성 기반 액세스 제어를 참조하십시오.

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

```
      "Effect": "Deny",
      "Action": "sqs:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/environment": "prod"
        }
      }
    }
  ]
}
```

IAM 사용자 및 Amazon SQS 대기열 생성

다음 예제는 및 를 사용하여 AWS Management Console Amazon SQS에 대한 액세스를 제어하는 ABAC 정책을 생성하는 방법을 설명합니다. AWS CloudFormation

사용: AWS Management Console

IAM 사용자 생성

1. https://console.aws.amazon.com/iam/ 에서 AWS Management Console 로그인하고 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 사용자를 선택합니다.
3. 사용자 추가를 선택하고 사용자 이름 텍스트 상자에 이름을 입력합니다.
4. 액세스 키 - 프로그래밍 방식 액세스를 선택하고 다음: 권한을 선택합니다.
5. 다음: 태그를 선택합니다.
6. 태그 키를 `environment`로 추가하고 태그 값을 `beta`로 추가합니다.
7. 다음: 검토를 선택한 후 사용자 생성을 선택합니다.
8. 액세스 키 ID와 비밀 액세스 키를 복사하여 안전한 위치에 저장합니다.

IAM 사용자 권한 추가

1. 생성한 IAM 사용자를 선택합니다.
2. 인라인 정책 추가(Add inline policy)를 선택합니다.
3. JSON 탭에 다음 정책을 붙여넣습니다.

```
{
```

```
"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. 정책 검토를 선택합니다.
5. 정책 생성을 선택합니다.

사용 AWS CloudFormation

다음 샘플 AWS CloudFormation 템플릿을 사용하여 인라인 정책이 연결되어 있고 Amazon SQS 대기열이 있는 IAM 사용자를 생성합니다.

```
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"
```

속성 기반 액세스 제어 테스트

다음 예제는 Amazon SQS에 속성 기반 액세스 제어를 테스트하는 방법을 보여줍니다.

태그 키를 environment로 설정하고 태그 값을 prod로 설정하여 대기열을 생성합니다.

이 AWS CLI 명령을 실행하여 태그 키를 환경으로 설정하고 태그 값을 prod로 설정하여 대기열 생성을 테스트합니다. AWS CLI가 없는 경우 컴퓨터에 맞게 CLI를 다운로드하여 구성할 수 있습니다.

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

Amazon SQS 엔드포인트에서 다음과 같은 AccessDenied 오류가 발생합니다.

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

이는 IAM 사용자의 태그 값이 CreateQueue API 호출에서 전달된 태그와 일치하지 않기 때문입니다. 키가 environment로 설정되고 값이 beta로 설정된 태그를 IAM 사용자에게 적용했다는 점을 기억하세요.

태그 키를 environment로 설정하고 태그 값을 beta로 설정하여 대기열을 생성합니다.

이 CLI 명령을 실행하여 태그 키가 environment로 설정되고 태그 값이 beta로 설정된 대기열 생성을 테스트합니다.

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

아래와 비슷한 방법으로 대기열이 성공적으로 생성되었음을 확인하는 메시지가 나타납니다.

```
{
"QueueUrl": "<queueUrl>“
}
```

대기열로 메시지 전송

이 CLI 명령을 실행하여 대기열에 메시지 전송을 테스트합니다.

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

응답은 Amazon SQS 대기열로 메시지가 성공적으로 전송되었음을 보여줍니다. IAM 사용자 권한을 사용하면 beta 태그가 있는 대기열에 메시지를 보낼 수 있습니다. 응답에는 메시지가 포함된 MD5OfMessageBody 및 MessageId가 포함됩니다.

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

Amazon SQS 콘솔을 사용하여 대기열 파라미터 구성

대기열을 생성 또는 편집할 때 다음 파라미터를 구성할 수 있습니다.

- 표시 제한 시간 - 한 소비자가 대기열에서 수신한 메시지가 다른 메시지 소비자에게 표시되지 않는 시간입니다. 자세한 내용은 표시 제한 시간을 참조하세요.

  > **Note**
  >
  > 콘솔을 사용하여 표시 제한 시간을 구성하면 대기열에 있는 모든 메시지에 대한 제한 시간 값이 구성됩니다. 단일 또는 다중 메시지의 제한 시간을 구성하려면 SDK 중 하나를 사용해야 합니다. AWS

- 메시지 보존 기간 - Amazon SQS가 대기열에 남아 있는 메시지를 유지하는 기간입니다. 기본적으로 대기열의 메시지는 4일 동안 유지됩니다. 메시지를 최대 14일 동안 유지하도록 대기열을 구성할 수 있습니다. 자세한 내용은 메시지 보존 기간을 참조하세요.
- 전송 지연 - Amazon SQS가 대기열에 추가된 메시지를 전송하기 전에 지연하는 시간입니다. 자세한 내용은 전송 지연을 참조하세요.
- 최대 메시지 크기 - 이 대기열의 최대 메시지 크기입니다. 자세한 내용은 최대 메시지 크기를 참조하세요.
- 메시지 수신 대기 시간 - 대기열이 수신 요청을 받은 후 Amazon SQS가 메시지를 사용할 수 있을 때까지 기다리는 최대 시간입니다. 자세한 정보는 Amazon SQS 짧은 폴링 및 긴 폴링을 참조하세요.
- 콘텐츠 기반 중복 제거 활성화 - Amazon SQS는 메시지 본문을 기반으로 중복 제거 ID를 자동으로 생성할 수 있습니다. 자세한 정보는 Amazon SQS에서 FIFO 대기열을 사용하기 시작하기을 참조하세요.
- 높은 처리량 FIFO 활성화 - 대기열의 메시지에 대해 높은 처리량을 활성화하는 데 사용합니다. 이 옵션을 선택하면 관련 옵션(중복 제거 범위 및 FIFO 처리량 한도)이 FIFO 대기열의 높은 처리량을 활성화하는 데 필요한 설정으로 변경됩니다. 자세한 내용은 Amazon SQS의 FIFO 대기열에 대한 높은 처리량 및 Amazon SQS 메시지 할당량 섹션을 참조하세요.
- 리드라이브 허용 정책: 이 대기열을 배달 못한 편지 대기열로 사용할 수 있는 소스 대기열을 정의합니다. 자세한 정보는 Amazon SQS에서 데드레터 대기열 사용 을 참조하세요.

기존 대기열에 대한 대기열 파라미터를 구성하려면(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다. 대기열을 선택하고 편집을 선택합니다.
3. 구성 섹션으로 스크롤합니다.
4. 표시 제한 시간에 기간과 단위를 입력합니다. 범위는 0초~12시간입니다. 기본값은 30초입니다.
5. 메시지 보존 기간에 기간과 단위를 입력합니다. 범위는 1분~14일입니다. 기본값은 4일입니다.
6. 표준 대기열의 경우 메시지 수신 대기 시간 값을 입력합니다. 범위는 0~20초입니다. 기본값은 0초이며 짧은 폴링을 설정합니다. 0이 아닌 값은 모두 긴 폴링을 설정합니다.
7. 전송 지연에 기간과 단위를 입력합니다. 범위는 0초~15분입니다. 기본값은 0초입니다.
8. 최대 메시지 크기에 값을 입력합니다. 범위는 1~256KB입니다. 기본값은 256KB입니다.
9. FIFO 대기열의 경우 콘텐츠 기반 중복 제거 활성화를 선택하여 콘텐츠 기반 중복 제거를 활성화합니다. 기본 설정은 비활성화입니다.
10. (선택 사항) FIFO 대기열에서 메시지를 보내고 받을 때 처리량을 높이려면 높은 처리량 FIFO 활성화를 선택합니다.

    이 옵션을 선택하면 관련 옵션(중복 제거 범위 및 FIFO 처리량 한도)이 FIFO 대기열의 높은 처리량을 활성화하는 데 필요한 설정으로 변경됩니다. 높은 처리량 FIFO를 사용하는 데 필요한 설정을 변경하면 대기열에 일반 처리량이 적용되고 지정된 대로 중복 제거가 수행됩니다. 자세한 내용은 Amazon SQS의 FIFO 대기열에 대한 높은 처리량 및 Amazon SQS 메시지 할당량 섹션을 참조하세요.
11. 리드라이브 허용 정책의 경우 활성화됨을 선택합니다. 모두 허용(기본값), 대기열 기준 또는 모두 거부 중에서 선택합니다. 대기열 기준 선택 시 Amazon 리소스 이름(ARN)별로 최대 10개의 소스 대기열 목록을 지정합니다.
12. 대기열 파라미터 구성을 마쳤으면 저장을 선택합니다.

액세스 정책 구성

대기열을 편집할 때 해당 액세스 정책을 구성할 수 있습니다.

액세스 정책은 대기열에 액세스할 수 있는 계정, 사용자 및 역할을 정의합니다. 또한 액세스 정책은 사용자가 액세스할 수 있는 작업(예: `SendMessage`, `ReceiveMessage` 또는`DeleteMessage`)을 정의합니다. 기본 정책에서는 대기열 소유자만 메시지를 보내고 받을 수 있도록 허용합니다.

기존 대기열에 대해 액세스 정책을 구성하려면(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열을 선택하고 편집을 선택합니다.
4. 액세스 정책 섹션으로 스크롤합니다.
5. 입력 상자에서 액세스 정책 설명을 편집합니다. 액세스 정책 설명에 대한 자세한 내용은 Amazon SQS의 Identity and Access Management 섹션을 참조하세요.
6. 액세스 정책 구성을 마쳤으면 저장을 선택합니다.

SQS에서 관리하는 암호화 키를 사용하여 대기열에 대한 서버 측 암호화 구성

기본 Amazon SQS 관리형 서버 측 암호화(SSE) 옵션 외에도 Amazon SQS 관리형 SSE(SSE-SQS)를 사용하면 SQS 관리형 암호화 키 기반의 사용자 지정 관리형 서버 측 암호화를 생성하여 메시지 대기열을 통해 전송되는 민감한 데이터를 보호할 수 있습니다. SSE-SQS를 사용하면 암호화 키를 생성 및 관리하거나 데이터 암호화를 위해 코드를 수정할 필요가 없습니다. SSE-SQS를 사용하면 데이터를 안전하게 전송하는 한편, 추가 비용 없이 엄격한 암호화 규정 준수 및 규제 요구 사항을 충족할 수 있습니다.

SSE-SQS는 256비트 Advanced Encryption Standard(AES-256) 암호화를 사용하여 저장 데이터를 보호합니다. Amazon SQS가 메시지를 수신하면 SSE가 메시지를 즉시 암호화합니다. Amazon SQS는 메시지를 암호화된 형태로 저장하고 권한 있는 소비자에게 메시지를 보낼 때만 암호를 해독합니다.

> **Note**
>
> - 기본 SSE 옵션은 암호화 속성을 지정하지 않고 대기열을 생성할 때만 유효합니다.
> - Amazon SQS는 모든 대기열 암호화를 끌 수 있습니다. 따라서 KMS-SSE를 끄면 SQS-SSE가 자동으로 활성화되지 않습니다. KMS-SSE를 끈 후 SQS-SSE를 활성화하려면 요청에 속성 변경을 추가해야 합니다.

대기열에 대해 SSE-SQS 암호화를 구성하려면(콘솔)

> **Note**
>
> HTTP(비 TLS) 엔드포인트를 사용하여 만든 새 대기열에서는 기본적으로 SSE-SQS 암호화가 활성화되지 않습니다. HTTPS 또는 서명 버전 4 엔드포인트를 사용하여 Amazon SQS 대기열을 생성하는 것이 보안 모범 사례입니다.

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열을 선택한 다음 편집을 선택합니다.
4. 암호화를 확장합니다.
5. 서버 측 암호화에서 활성화(기본)를 선택합니다.

   > **Note**
   >
   > SSE를 활성화하면 암호화된 대기열에 대한 익명 `SendMessage` 및 `ReceiveMessage` 요청이 거부됩니다. Amazon SQS 보안 모범 사례에서는 익명 요청을 사용하지 말 것을 권장합니다. Amazon SQS 대기열로 익명 요청을 보내려면 SSE를 비활성화해야 합니다.

6. Amazon SQS 키(SSE-SQS)를 선택합니다. 이 옵션 사용에 대한 추가 요금은 부과되지 않습니다.
7. 저장을 선택합니다.

Amazon SQS 콘솔을 사용하여 대기열에 대한 서버 측 암호화 구성

대기열 메시지의 데이터를 보호하기 위해 Amazon SQS는 새로 생성된 모든 대기열에 대해 기본적으로 서버 측 암호화(SSE)를 사용하도록 설정되어 있습니다. Amazon SQS는 Amazon Web Services Key Management Service(Amazon Web Services KMS)와 통합하여 서버 측 암호화(SSE)용 KMS 키를 관리합니다. SSE 사용에 대한 자세한 내용은 Amazon SQS의 저장 중 암호화 섹션을 참조하세요.

대기열에 할당하는 KMS 키에는 대기열을 사용할 수 있는 권한이 있는 모든 주체에 대한 권한이 포함된 키 정책이 있어야 합니다. 자세한 내용은 키 관리를 참조하세요.

해당 KMS 키의 소유자가 아니거나 `kms:ListAliases` 및 `kms:DescribeKey` 권한이 없는 계정으로 로그인하는 경우 Amazon SQS 콘솔에서 해당 KMS 키에 대한 정보를 볼 수 없습니다. KMS 키의 소유자에게 이 권한을 부여해 달라고 요청해야 합니다. 자세한 내용은 키 관리를 참조하세요.

대기열을 만들 때 또는 편집할 때 SSE-KMS를 구성할 수 있습니다.

기존 대기열에 대해 SSE-KMS 구성(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열을 선택한 다음 편집을 선택합니다.
4. 암호화를 확장합니다.
5. 서버 측 암호화에서 활성화(기본)를 선택합니다.

> **Note**
>
> SSE를 활성화하면 암호화된 대기열에 대한 익명 `SendMessage` 및 `ReceiveMessage` 요청이 거부됩니다. Amazon SQS 보안 모범 사례에서는 익명 요청을 사용하지 말 것을 권장합니다. Amazon SQS 대기열로 익명 요청을 보내려면 SSE를 비활성화해야 합니다.

6. AWS 키 관리 서비스 키(SSE-KMS)를 선택합니다.

   콘솔에 설명, 계정 및 KMS 키의 KMS 키 ARN이 표시됩니다.
7. 대기열의 KMS 키 ID를 지정합니다. 자세한 정보는 주요 용어을 참조하세요.

   a. KMS 키 별칭 선택 옵션을 선택합니다.

   b. 기본 키는 Amazon SQS용 Amazon Web Services 관리형 KMS 키입니다. 이 키를 사용하려면 KMS 키 목록에서 키를 선택합니다.

   c. Amazon Web Services 계정에서 사용자 지정 KMS 키를 사용하려면 KMS 키 목록에서 해당 키를 선택합니다. 사용자 지정 KMS 키를 만드는 방법에 대한 지침은 Amazon Web Services Key Management Service 개발자 가이드에서 키 만들기를 참조하세요.

   d. 목록에 없는 사용자 지정 KMS 키 또는 다른 Amazon Web Services 계정의 사용자 지정 KMS 키를 사용하려면 KMS 키 별칭 입력을 선택하고 KMS 키 Amazon 리소스 이름(ARN)을 입력합니다.
8. (선택 사항) 데이터 키 재사용 기간에서 1분에서 24시간 사이의 값을 지정합니다. 기본값은 5분입니다. 자세한 정보는 데이터 키 재사용 기간 이해을 참조하세요.
9. SSE-KMS 구성을 마치면 저장을 선택합니다.

Amazon SQS 콘솔을 사용하여 대기열에 대한 비용 할당 태그 구성

Amazon SQS 대기열을 구성하고 식별하는 데 도움이 되도록 대기열에 비용 할당 태그를 추가할 수 있습니다. 자세한 정보는 Amazon SQS 비용 할당 태그을 참조하세요.

대기열의 세부 정보 페이지에 있는 태그 지정 탭에는 대기열의 태그가 표시됩니다.

대기열을 만들 때 또는 편집할 때 대기열에 대한 태그를 구성할 수 있습니다.

기존 대기열에 대해 태그 구성(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열을 선택하고 편집을 선택합니다.
4. 스크롤하여 태그 섹션으로 이동합니다.
5. 다음과 같이 대기열 태그를 추가, 수정 또는 제거합니다.
   a. 태그를 추가하려면 새 태그 추가를 선택하고 키 및 값을 입력한 다음 새 태그 추가를 선택합니다.
   b. 태그를 업데이트하려면 해당 태그의 키와 값을 변경합니다.
   c. 태그를 제거하려면 키-값 페어 옆에 있는 제거를 선택합니다.
6. 태그 구성을 마치면 저장을 선택합니다.

Amazon SQS 콘솔을 사용하여 Amazon SNS 주제 대기열 구독하기

하나 이상의 Amazon SQS 대기열이 Amazon Simple Notification Service(Amazon SNS) 주제를 구독할 수 있습니다. 메시지를 주제에 게시하면 Amazon SNS는 구독 중인 각 대기열에 그 메시지를 전송합니다. Amazon SQS는 구독 및 필요한 모든 권한을 관리합니다. Amazon SNS에 관한 자세한 내용은 Amazon Simple Notification Service Developer 안내서의 Amazon SNS란 무엇인가요?를 참조하세요.

Amazon SQS 대기열에서 SNS 주제를 구독하면 Amazon SNS는 HTTPS를 사용하여 Amazon SQS에 메시지를 전달합니다. 암호화된 Amazon SQS 대기열과 함께 Amazon SNS를 사용하는 방법에 대한 자세한 내용은 서비스에 대한 KMS 권한을 구성합니다 AWS . 섹션을 참조하세요.

> **Important**
>
> Amazon SQS는 액세스 정책당 최대 20개의 명령문을 지원합니다. Amazon SNS 주제를 구독하면 이러한 명령문이 하나 추가됩니다. 이 개수를 초과하면 주제 구독 전송이 실패합니다.

대기열이 SNS 주제를 구독하려면(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열 목록에서 SNS 주제를 구독할 대기열을 선택합니다.
4. 작업(Actions)에서 Amazon SNS 주제 구독(Subscribe to Amazon SNS topic)을 선택합니다.
5. 이 대기열 메뉴에 사용 가능한 Amazon SNS 주제 지정에서 대기열에 대해 SNS 주제를 선택합니다.

   메뉴에 SNS 주제가 없는 경우, Amazon SNS 주제 ARN 입력을 선택한 다음 주제의 Amazon 리소스 이름(ARN)을 입력합니다.
6. 저장을 선택합니다.
7. 구독 결과를 확인하기 위해 주제에 게시한 다음 주제가 대기열에 전송하는 메시지를 확인할 수 있습니다. 자세한 내용은 Amazon Simple Notification Service 개발자 안내서의 Amazon SNS 메시지 게시를 참조하세요.

Amazon SQS 대기열과 SNS 주제가 다른 AWS 계정경우 주제 소유자가 먼저 구독을 확인해야 합니다. 자세한 내용은 Amazon Simple Notification Service 개발자 가이드의 구독 확인 섹션을 참조하세요.

지역 간 SNS 주제 구독에 대한 자세한 내용은 Amazon 단순 알림 서비스 개발자 안내서의 다른 지역의 Amazon SQS 대기열 AWS Lambda 또는 함수에 Amazon SNS 메시지 전송을 참조하십시오.

함수를 트리거하도록 Amazon SQS 대기열 구성 AWS Lambda

AWS Lambda 함수를 사용하여 Amazon SQS 대기열의 메시지를 처리할 수 있습니다. Lambda는 대기열을 폴링하고 대기열 메시지를 포함한 이벤트와 동기적으로 Lambda 함수를 호출합니다. 함수 시간이 각 레코드 배치를 처리할 수 있도록 하려면 소스 대기열의 가시성 시간제한을 함수에 구성한 시간 제한의 6배 이상으로 설정하세요. 이전 배치를 처리하는 동안 함수가 제한되는 경우 추가 시간을 통해 Lambda가 재시도할 수 있습니다.

Lambda 함수가 처리할 수 없는 메시지에 대해 배달 못한 편지 대기열 역할을 하도록 다른 대기열을 지정할 수 있습니다.

Lambda 함수는 여러 대기열의 항목(각 대기열에 대해 하나의 Lambda 이벤트 소스 사용)을 처리할 수 있습니다. 여러 Lambda 함수에서 동일한 대기열을 사용할 수 있습니다.

암호화된 대기열을 Lambda 함수와 연결하지만 Lambda가 메시지를 폴링하지 않는 경우 Lambda 실행 역할에 `kms:Decrypt` 권한을 추가합니다.

다음과 같은 제한 사항이 있습니다.

- 대기열과 Lambda 함수는 동일한 리전에 있어야 합니다. AWS
- 기본 키 (Amazon SQS의AWS 관리형 KMS 키) 를 사용하는 암호화된 대기열은 다른 키에서 Lambda 함수를 호출할 수 없습니다. AWS 계정

Lambda 함수 구현에 대한 자세한 내용은 개발자 안내서의 AWS Lambda Amazon SQS와 함께 사용을 참조하십시오.AWS Lambda

필수 조건

Lambda 함수 트리거를 구성하려면 다음 요구 사항을 충족해야 합니다.

- 사용자를 사용하는 경우 Amazon SQS 역할에 다음 권한이 포함되어야 합니다.
  - `lambda:CreateEventSourceMapping`
  - `lambda:ListEventSourceMappings`
  - `lambda:ListFunctions`
- Lambda 실행 역할에는 다음 권한이 포함되어야 합니다.
  - `sqs:DeleteMessage`
  - `sqs:GetQueueAttributes`
  - `sqs:ReceiveMessage`
- 암호화된 대기열을 Lambda 함수와 연결하는 경우 Lambda 실행 역할에 `kms:Decrypt` 권한을 추가합니다.

자세한 정보는 Amazon SQS의 액세스 관리 개요을 참조하세요.

Lambda 함수를 트리거하는 대기열을 구성하려면(콘솔)

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열 페이지에서 구성할 대기열을 선택합니다.
4. 대기열 페이지에서 Lambda 트리거 탭을 선택합니다.
5. Lambda 트리거 페이지에서 Lambda 트리거를 선택합니다.

   목록에 필요한 Lambda 트리거가 포함되어 있지 않은 경우 Lambda 함수 트리거 구성을 선택합니다. Lambda 함수의 Amazon 리소스 이름(ARN)을 입력하거나 기존 리소스를 선택합니다. 그런 다음 저장을 선택합니다.
6. 저장을 선택합니다. 콘솔에서 구성을 저장하고 대기열의 세부 정보 페이지를 표시합니다.

   세부 정보 페이지의 Lambda 트리거 탭에는 Lambda 함수와 해당 상태가 표시됩니다. Lambda 함수가 대기열에 연결되기까지 약 1분 정도 소요됩니다.
7. 구성 결과를 확인하려면 대기열에 메시지를 전송한 후 Lambda 콘솔에서 트리거된 Lambda 함수를 확인합니다.

Amazon을 사용하여 AWS 서비스에서 Amazon SQS로 보내는 알림 자동화 EventBridge

EventBridge Amazon을 사용하면 AWS 서비스를 자동화하고 애플리케이션 가용성 문제 또는 리소스 변경과 같은 시스템 이벤트에 대응할 수 있습니다. AWS 서비스에서 발생하는 이벤트는 EventBridge 거의 실시간으로 전송됩니다. 원하는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 태스크를 지정할 수 있습니다.

EventBridge Amazon SQS 표준 및 FIFO 대기열과 같이 JSON 형식으로 이벤트를 수신하는 다양한 대상을 설정할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge 대상을 참조하십시오.

속성을 포함하는 메시지 전송

표준 대기열 및 FIFO 대기열의 경우, 메시지에 구조화된 메타데이터(예: 타임스탬프, 지리 공간 데이터, 서명 및 식별자)를 포함할 수 있습니다. 자세한 정보는 Amazon SQS 메시지 속성을 참조하세요.

Amazon SQS 콘솔을 사용하여 속성이 포함된 메시지를 대기열로 보내려면

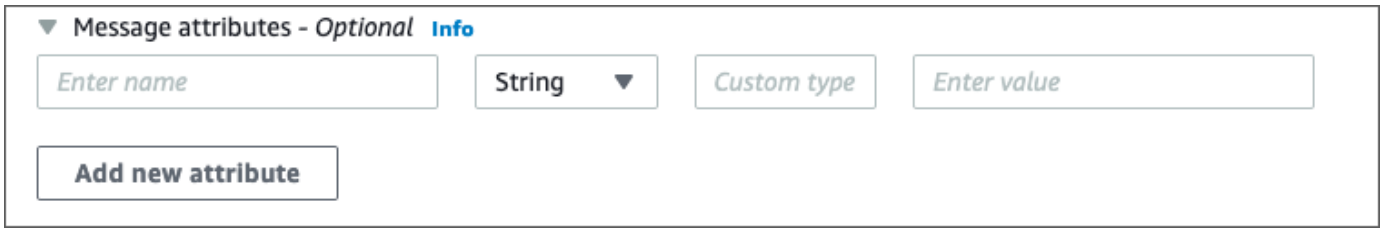
1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 탐색 창에서 대기열을 선택합니다.
3. 대기열 페이지에서 대기열을 선택합니다.
4. [메시지 전송 및 수신(Send and receive messages)]을 선택합니다.
5. 메시지 속성 파라미터를 입력합니다.

   a. 이름 텍스트 상자에 최대 256자의 고유한 이름을 입력합니다.

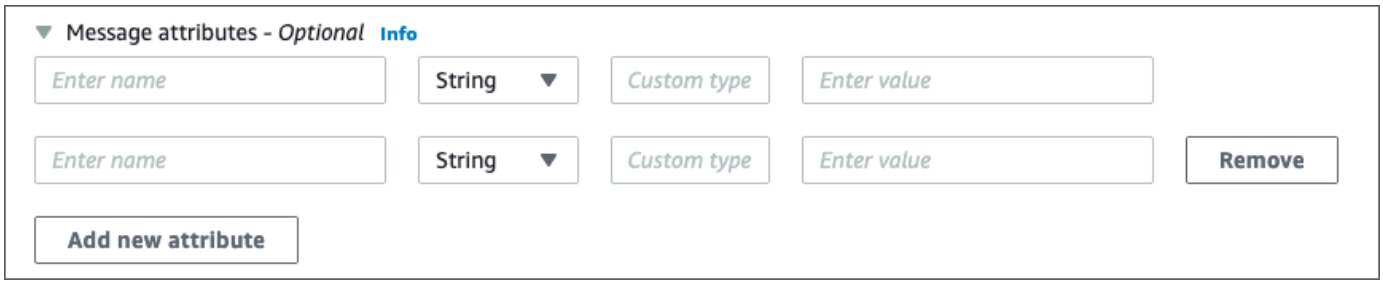
   b. 속성 유형으로는 문자열, 숫자 또는 2진수를 선택합니다.

   c. (선택 사항) 사용자 지정 데이터 유형을 입력합니다. 예를 들어, 번호에 **`byte`**, **`int`** 또는 **`float`**를 사용자 지정 데이터 유형으로 추가할 수 있습니다.

   d. 값 텍스트 상자에 메시지 속성 값을 입력합니다.



6. 다른 메시지 속성을 추가하려면 새 속성 추가를 선택합니다.



7. 메시지를 전송하기 전에 언제든지 속성 값을 수정할 수 있습니다.
8. 속성을 삭제하려면 제거를 선택합니다. 첫 번째 속성을 삭제하려면 메시지 속성을 닫습니다.
9. 속성을 메시지에 추가했으면 메시지 전송을 선택합니다. 메시지가 전송되고 콘솔에 성공 메시지가 표시됩니다. 보낸 메시지의 메시지 속성에 대한 정보를 보려면 세부 정보 보기를 선택합니다. 완료를 선택하여 메시지 세부 정보 대화 상자를 닫습니다.

Amazon SQS의 모범 사례

이 모범 사례를 통해 Amazon SQS를 최대로 활용할 수 있습니다.

주제

Amazon SQS 표준 및 FIFO 대기열에 대한 권장 사항

다음 모범 사례를 통해 Amazon SQS를 사용하여 비용을 절감하고 메시지를 효율적으로 처리할 수 있습니다.

주제

Amazon SQS 메시지 작업

다음 지침을 통해 Amazon SQS를 사용하여 메시지를 효율적으로 처리할 수 있습니다.

주제

적시에 메시지 처리

제한 시간 초과 설정은 애플리케이션에서 메시지를 처리하고 삭제하는 데 얼마나 걸리느냐에 따라 다릅니다. 예를 들어 애플리케이션에서 메시지를 처리하려면 10초가 필요한데 제한 시간 초과 값을 15분으로 설정한다면, 이전의 메시지 처리 시도가 실패할 경우 다시 메시지 처리를 시도하기 위해 비교적 오랜 시간을 기다려야 합니다. 또한 애플리케이션에서 메시지를 처리하려면 10초가 필요한데 제한 시간 초과 값을 불과 2초로 설정한다면, 원래 소비자가 아직 메시지를 처리하는 동안 다른 소비자로부터 중복 메시지를 받을 수 있습니다.

메시지 처리 시간을 충분히 확보하려면 다음 전략 중 하나를 사용하세요.

- 메시지 처리 소요 시간을 알고 있거나 합리적으로 추정할 수 있는 경우, 메시지의 제한 시간 초과를 메시지 처리와 삭제에 소요되는 최대 시간으로 연장합니다. 자세한 내용은 표시 제한 시간 구성을 참조하세요.
- 메시지를 처리하는 데 걸리는 시간을 모르는 경우 소비자 프로세스에 대한 하트비트를 생성합니다. 초기 표시 제한 시간(예: 2분)을 지정한 다음 소비자가 메시지에서 계속 작업하는 경우 표시 제한 시간을 1분마다 2분씩 연장합니다.

> **Important**
>
> 최대 표시 제한 시간은 Amazon SQS가 `ReceiveMessage` 요청을 수신한 시간으로부터 12시간입니다. 표시 제한 시간을 연장해도 최대 12시간이 재설정되지는 않습니다.
> 또한 `ReceiveMessage` 요청으로 타이머가 시작되기 때문에 개별 메시지의 제한 시간을 전체 12시간(예: 43,200초)으로 설정하지 못할 수도 있습니다. 예를 들어 메시지를 수신한 후 `VisibilityTimeout`이 43,200초인 `ChangeMessageVisibility` 호출을 전송하여 즉시 최대 12시간을 설정하면 실패할 가능성이 높습니다. 하지만 43,195초 값을 사용하면 `ReceiveMessage`를 통해 메시지를 요청하는 것과 표시 제한 시간을 업데이트하는 것 사이에 상당한 지연이 있는 경우가 아니면 작동합니다. 소비자에게 12시간 이상 필요한 경우 Step Functions 사용을 고려하세요.

요청 오류 처리

요청 오류를 처리하려면 다음 전략 중 하나를 사용하십시오.

- AWS SDK를 사용하는 경우 이미 자동 재시도 및 백오프 로직을 마음대로 사용할 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS의 오류 재시도 및 지수 백오프 단원을 참조하세요.

- 재시도 및 백오프에 AWS SDK 기능을 사용하지 않는 경우 Amazon SQS로부터 메시지 없음, 시간 초과 또는 오류 메시지를 수신한 후 ReceiveMessage작업을 재시도하기 전에 일시 중지 (예: 200ms)를 허용하십시오. 그 다음에 동일한 결과를 반환하는 ReceiveMessage를 사용하려면, 더 긴 일시 중지 시간(예: 400ms)을 둡니다.

긴 폴링 설정

ReceiveMessage API 작업에 대한 대기 시간이 0보다 큰 경우 긴 폴링이 유효합니다. 긴 폴링 대기 시간의 최대값은 20초입니다. 긴 폴링은 빈 응답 수를 줄이고(ReceiveMessage 요청에 사용 가능한 메시지가 없는 경우) 거짓의 빈 응답을 제거하여(메시지를 사용할 수 있지만 응답에 포함되지 않은 경우) Amazon SQS 사용 비용을 줄여 줍니다. 자세한 정보는 Amazon SQS 짧은 폴링 및 긴 폴링을 참조하세요.

메시지 처리를 최적화하려면 다음 전략을 사용하세요.

- 대부분의 경우 ReceiveMessage 대기 시간을 20초로 설정할 수 있습니다. 애플리케이션에 20초가 너무 긴 경우 ReceiveMessage 대기 시간을 더 짧게 설정할 수 있습니다(최소값은 1초). Amazon SQS에 액세스하는 데 AWS SDK를 사용하지 않거나 대기 시간을 단축하도록 SDK를 구성하는 AWS 경우, 더 긴 요청을 허용하거나 긴 폴링을 위해 더 짧은 대기 시간을 사용하도록 Amazon SQS 클라이언트를 수정해야 할 수 있습니다.
- 여러 대기열에 대해 긴 폴링을 구현할 경우에는 모든 대기열에 단일 스레드를 사용하지 말고 각 대기열마다 하나의 스레드를 사용하는 것이 좋습니다. 각 대기열마다 하나의 스레드를 사용하면 애플리케이션이 각 대기열에서 메시지를 사용 가능한 즉시 처리할 수 있지만, 하나의 스레드를 사용하여 여러 대기열을 폴링하면 애플리케이션이 사용 가능한 메시지가 없는 대기열을 대기하느라(최대 20초) 다른 대기열에서 사용 가능한 메시지를 처리하지 못하게 될 수 있습니다.

> **Important**
>
> HTTP 오류를 방지하려면 ReceiveMessage 요청에 대한 HTTP 응답 제한 시간이 WaitTimeSeconds 파라미터보다 긴지 확인합니다. 자세한 내용은 을 참조하십시오. ReceiveMessage

문제가 있는 메시지 캡처

처리할 수 없는 모든 메시지를 캡처하고 정확한 CloudWatch 지표를 수집하려면 데드레터 큐를 구성하세요.

- 소스 대기열이 지정된 횟수까지 메시지를 처리하지 못하고 나면 리드라이브 정책에 따라 메시지를 배달 못한 편지 대기열로 리디렉션합니다.
- 배달 못한 편지 대기열을 사용하면 메시지 개수가 감소하고 사용자가 독약 메시지(수신하지만 처리 불가능한 메시지)를 받을 가능성이 낮아집니다.
- 대기열에 포이즌 필 메시지를 포함시키면 포이즌 필 메시지의 유효 기간을 잘못 지정하여 `ApproximateAgeOfOldestMessage` CloudWatch 메트릭이 왜곡될 수 있습니다. 배달 못한 편지 대기열을 구성하면 이 측정치를 사용할 때 거짓 경보를 방지할 수 있습니다.

데드레터 대기열 보존 설정

표준 대기열의 경우, 메시지 만료는 항상 메시지가 원래 대기열에 추가된 타임스탬프를 기준으로 합니다. 메시지가 DLQ(Dead Letter Queue)로 이동하면 대기열에 추가 타임스탬프는 변경되지 않습니다. `ApproximateAgeOfOldestMessage` 지표는 메시지가 원래 전송된 시간이 아니라 메시지가 DLQ(Dead Letter Queue)로 이동한 시간을 나타냅니다. 예를 들어, 메시지가 DLQ(Dead Letter Queue)로 이동하기 전에 원래 대기열에서 1일을 보낸다고 가정합니다. DLQ(Dead Letter Queue)의 보존 기간이 4일인 경우 메시지는 3일 후에 DLQ(Dead Letter Queue)에서 삭제되며 `ApproximateAgeOfOldestMessage`는 3일입니다. 따라서 배달 못한 편지 대기열의 보존 기간을 항상 원래 대기열의 보존 기간보다 더 길게 설정하는 것이 좋습니다.

FIFO 대기열의 경우 메시지가 DLQ(Dead Letter Queue)로 이동하면 대기열에 추가 타임스탬프가 재설정됩니다. `ApproximateAgeOfOldestMessage` 지표는 메시지가 DLQ(Dead Letter Queue)로 이동한 시간을 나타냅니다. 위의 동일한 예에서 메시지는 4일 후에 배달 못한 편지 대기열에서 삭제되며 `ApproximateAgeOfOldestMessage`는 4일입니다.

일관되지 않은 메시지 처리 방지

Amazon SQS는 분산 시스템이므로 메시지가 `ReceiveMessage` API 메서드 호출에서 성공적으로 반환되고 Amazon SQS에서 배달된 것으로 표시되더라도 소비자가 해당 메시지를 받지 못할 수 있습니다. 이 경우 Amazon SQS는 소비자가 수신한 적이 없는 메시지를 한 번 이상 배달된 것으로 기록합니다. 이 상태에서는 메시지 배달이 추가로 시도되지 않으므로 배달 못한 편지 대기열의 최대 수신 수를 1로 설정하지 않는 것이 좋습니다.

요청-응답 시스템 구현

요청 응답 또는 RPC(원격 프로시저 호출) 시스템을 구현할 경우에는 다음 모범 사례에 유의하십시오.

- 메시지별로 응답 대기열을 생성하지 마십시오. 대신 시작 시 생산자별로 응답 대기열을 만들고 상관 관계 ID 메시지 속성을 사용하여 회신을 요청에 매핑하십시오.

- 생산자가 응답 대기열을 공유할 수 없도록 하십시오. 이로 인해 생산자는 다른 생산자를 위한 응답 메시지를 수신할 수 있습니다.

임시 대기열 클라이언트를 사용하여 요청-응답 패턴을 구현하는 방법에 대한 자세한 내용은 요청-응답 메시징 패턴(가상 대기열) 단원을 참조하십시오.

Amazon SQS 비용 절감

다음 모범 사례를 통해 비용을 절감할 수 있으며 추가적인 비용 절감 가능성과 거의 즉각적인 응답을 활용할 수 있습니다.

메시지 작업 일괄 처리

비용을 절감하려면 메시지 작업을 일괄 처리합니다.

- 메시지를 전송, 수신 및 삭제하고 한 번의 작업으로 여러 메시지에 대한 메시지 표시 시간 제한을 변경하려면 Amazon SQS 배치 API 작업을 사용합니다.
- 클라이언트 측 버퍼링과 요청 배치 처리를 결합하려면 AWS SDK for Java에 포함된 버퍼링된 비동기식 클라이언트와 함께 긴 폴링을 사용합니다.

> **Note**
>
> Amazon SQS의 버퍼링된 비동기식 클라이언트는 현재 FIFO 대기열을 지원하지 않습니다.

적절한 폴링 모드 사용

- 긴 폴링은 Amazon SQS 대기열의 메시지가 사용 가능 상태가 되자마자 사용할 수 있게 해줍니다.
  - Amazon SQS 사용 비용을 절감하고 빈 대기열로의 빈 수신(반환되는 메시지가 없는 `ReceiveMessage` 작업의 응답) 횟수를 줄이려면 긴 폴링을 활성화합니다. 자세한 내용은 Amazon SQS 긴 폴링을 참조하세요.
  - 여러 번의 수신으로 여러 스레드를 폴링할 때 효율성을 높이려면, 스레드 수를 줄입니다.
  - 대다수의 경우에 긴 폴링이 짧은 폴링보다 선호됩니다.
- 폴링된 Amazon SQS 대기열이 비어 있더라도 짧은 폴링은 즉시 응답을 반환합니다.
  - `ReceiveMessage` 요청에 대한 즉각적인 응답이 예상되는 애플리케이션의 요구 사항을 충족하려면, 짧은 폴링을 사용합니다.
  - 긴 폴링과 마찬가지로 짧은 폴링도 요금이 청구됩니다.

Amazon SQS 표준 대기열에서 FIFO 대기열로 이동

각 메시지에서 `DelaySeconds` 파라미터를 설정하지 않는 경우, 전송된 모든 메시지에 메시지 그룹 ID를 지정하여 FIFO 대기열로 이동할 수 있습니다.

자세한 내용은 Amazon SQS의 표준 대기열에서 FIFO 대기열로 이동을(를) 참조하세요.

Amazon SQS FIFO 대기열에 대한 추가 권장 사항

다음 모범 사례를 통해 메시지 중복 제거 ID와 메시지 그룹 ID를 최적으로 활용할 수 있습니다. 자세한 내용은 Amazon Simple Queue Service API 참조의 `SendMessage` 및 `SendMessageBatch` 작업을 참조하세요.

주제

- Amazon SQS 메시지 중복 제거 ID 사용
- Amazon SQS 메시지 그룹 ID 사용
- Amazon SQS 수신 요청 시도 ID 사용

Amazon SQS 메시지 중복 제거 ID 사용

메시지 중복 제거 ID는 전송된 메시지의 중복 제거에 사용되는 토큰입니다. 특정 메시지 중복 제거 ID가 있는 메시지를 성공적으로 전송한 경우, 메시지 중복 제거 ID가 동일한 모든 메시지는 성공적으로 수신되지만 중복 제거 간격인 5분 동안은 전달되지 않습니다.

> **Note**
>
> Amazon SQS는 메시지가 수신되어 삭제된 후에도 메시지 중복 제거 ID를 계속 추적합니다.

메시지 중복 제거 ID 제공

생산자는 다음 시나리오에서 각 메시지에 대해 메시지 중복 제거 ID 값을 제공해야 합니다.

- Amazon SQS가 고유하게 처리해야 하는 동일한 메시지 본문이 포함된 메시지 전송.
- Amazon SQS가 고유하게 처리해야 하는 동일한 콘텐츠가 포함되어 있지만 메시지 속성이 다른 메시지 전송.
- Amazon SQS가 중복으로 처리해야 하는 다른 콘텐츠(예: 메시지 본문에 재시도 횟수 포함)가 포함된 메시지 전송.

단일 생산자/소비자 시스템에 대한 중복 제거 활성화

애플리케이션별 메시지 ID가 메시지 본문에 포함되어 있기 때문에 단일 생산자, 단일 소비자, 고유 메시지가 있으면 다음 모범 사례를 따르십시오.

- 대기열에서 콘텐츠 기반 중복 제거 기능을 활성화합니다(각 메시지에는 고유한 본문이 있음). 생성자는 메시지 중복 제거 ID를 생략할 수 있습니다.
- Amazon SQS FIFO 대기열에 콘텐츠 기반 중복 제거가 활성화되고 메시지가 중복 제거 ID와 함께 전송되면 중복 제거 ID가 생성된 콘텐츠 기반 중복 제거 ID를 SendMessage 재정의합니다.
- 소비자가 각 요청별 수신 요청 시도 ID를 제공하지 않아도 되지만, 이렇게 하면 실패-재시도 시퀀스가 빠르게 실행되기 때문에 이는 권장되는 모범 사례입니다.
- 전송 또는 수신 요청은 FIFO 대기열에서 메시지 정렬을 방해하지 않기 때문에 이 요청을 다시 시도할 수 있습니다.

중단 복구 시나리오 설계

FIFO 대기열의 중복 제거 프로세스는 시간에 민감합니다. 애플리케이션을 설계할 때에는 클라이언트나 네트워크 작동이 중단되는 경우에 네트워크 생산자와 소비자 모두가 복구할 수 있도록 해야 합니다.

- 생산자는 대기열의 중복 제거 간격을 알고 있어야 합니다. Amazon SQS의 중복 제거 간격은 5분입니다. 중복 제거 간격이 만료된 이후에 SendMessage 요청을 재시도하면 대기열에 중복 메시지가 생성될 수 있습니다. 예를 들어, 자동차 안의 모바일 디바이스에서 순서가 중요한 메시지를 전송할 수 있습니다. 승인을 받기 전에 일정 기간 동안 자동차의 셀룰러 연결이 끊길 경우, 셀룰러가 다시 연결되고 나서 요청을 다시 시도하면 중복이 생성될 수 있습니다.
- 제한 시간 초과가 만료되기 전에 메시지를 처리하지 못할 위험을 최소화해 주는 제한 시간 초과를 소비자에게 적용해야 합니다. 메시지를 처리하는 동안 제한 시간 초과를 연장하려면 ChangeMessageVisibility 작업을 호출합니다. 그렇지만, 제한 시간 초과가 만료된 경우 다른 소비자가 메시지를 바로 처리할 수 있고, 이렇게 되면 한 메시지를 여러 번 처리하게 됩니다. 이 시나리오를 방지하려면 배달 못한 편지 대기열을 구성합니다.

제한 시간 초과 작업

최적의 성능을 위해 가시성 타임아웃을 AWS SDK 읽기 타임아웃보다 크게 설정하세요. 이는 짧은 폴링 또는 긴 폴링과 함께 ReceiveMessage API 작업을 사용하는 경우에 적용됩니다.

Amazon SQS 메시지 그룹 ID 사용

`MessageGroupId`는 메시지가 특정 메시지 그룹에 속하도록 지정하는 태그입니다. 동일한 메시지 그룹에 속하는 메시지는 항상 메시지 그룹을 기준으로 엄격한 순서에 따라 하나씩 처리됩니다(다른 메시지 그룹에 속하는 메시지는 순서가 뒤바뀌어 처리될 수 있음).

여러 개의 정렬된 메시지 그룹 인터리브

단일 FIFO 대기열 내에서 정렬된 여러 메시지 그룹을 인터리브하려면 메시지 그룹 ID 값(예: 여러 사용자의 세션 데이터)을 사용해야 합니다. 이 시나리오에서 여러 소비자가 대기열을 처리할 수 있지만, 각 사용자의 세션 데이터는 FIFO 방식으로 처리됩니다.

> **Note**
>
> 특정 메시지 그룹 ID에 속한 메시지가 표시되지 않으면, 다른 어떤 소비자도 동일한 메시지 그룹 ID로 메시지를 처리할 수 없습니다.

여러 생산자/소비자 시스템에서 중복 처리 방지

여러 생성자와 소비자에게 처리량과 지연 시간이 정렬보다 더 중요한 경우에 시스템에서 중복 메시지를 처리하지 않으려면, 생성자가 각 메시지별로 고유 메시지 그룹 ID를 생성해야 합니다.

> **Note**
>
> 이 시나리오에서 중복 항목이 제거됩니다. 그러나 명령 실행 정렬은 보장되지 않습니다. 작업자가 제한 시간 초과 내에 메시지를 처리하지 않고 이 메시지를 다른 작업자가 사용할 수 있게 되면 여러 생성자와 소비자가 있는 시나리오에서는 중복 메시지를 부적절하게 전송할 위험이 높아집니다.

메시지 그룹 ID가 동일한 메시지의 대규모 백로그 방지

FIFO 대기열의 경우 최대 20,000개의 이동 중인 메시지(소비자가 대기열에서 수신했지만 대기열에서 아직 삭제되지 않은 메시지)가 있을 수 있습니다. 이 할당량에 도달해도 Amazon SQS에서는 오류 메시지를 반환하지 않습니다. FIFO 대기열은 처음 2만 개의 메시지를 검토하여 사용 가능한 메시지 그룹을 결정합니다. 즉, 단일 메시지 그룹에 메시지 백로그가 있는 경우 백로그의 메시지를 성공적으로 사용할 때까지 나중에 대기열로 전송된 다른 메시지 그룹의 메시지를 사용할 수 없습니다.

> **Note**
>
> 메시지 그룹 ID가 동일한 메시지의 백로그는 사용자가 메시지를 처리할 수 없기 때문에 누적될 수도 있습니다. 메시지의 내용 문제 또는 사용자의 기술 문제로 인해 메시지 처리 문제가 발생할 수 있습니다.
> 처리할 수 없는 메시지를 반복적으로 제거하고 메시지 그룹 ID가 동일한 다른 메시지의 처리를 차단 해제하려면 배달 못한 편지 대기열 정책 설정을 고려하십시오.

가상 대기열에서 동일한 메시지 그룹 ID 재사용 방지

호스트 대기열이 동일한 서로 다른 가상 대기열로 전송된 동일한 메시지 그룹 ID를 가진 메시지가 서로 차단되는 것을 방지하려면 가상 대기열에서 동일한 메시지 그룹 ID를 재사용하는 것을 피해야 합니다.

Amazon SQS 수신 요청 시도 ID 사용

수신 요청 시도 ID는 `ReceiveMessage` 호출 중복 제거에 사용되는 토큰입니다.

오래 지속되는 네트워크 중단 기간 중에는 SDK와 Amazon SQS 사이의 연결 문제가 발생하므로, SDK 작업이 실패하면 수신 요청 시도 ID를 제공하고 동일한 수신 요청 시도 ID로 다시 시도하는 것이 권장되는 모범 사례입니다.

Amazon SQS Java SDK 예제

를 사용하여 Amazon Simple Queue Service (Amazon SQS) 및 기타 서비스와 상호 작용하는 Java 애플리케이션을 구축할 수 있습니다. AWS SDK for Java AWS SDK를 설치하고 설정하려면 AWS SDK for Java 2.x 개발자 안내서의 시작하기를 참조하세요.

대기열을 생성하거나 메시지를 전송하는 방법과 같은 기본적인 Amazon SQS 대기열 작업의 예는 AWS SDK for Java 2.x 개발자 안내서의 Amazon SQS 메시지 대기열 작업을 참조하세요.

이 주제의 예제는 서버 측 암호화(SSE), 비용 할당 태그, 메시지 속성과 같은 추가적인 Amazon SQS 기능을 보여줍니다.

주제

- Amazon SQS 대기열에서 서버 측 암호화 사용
- Amazon SQS 대기열에 대한 태그 구성
- Amazon SQS 대기열에 메시지 속성 전송

Amazon SQS 대기열에서 서버 측 암호화 사용

를 사용하여 Amazon SQS AWS SDK for Java 대기열에 서버 측 암호화 (SSE) 를 추가할 수 있습니다. 각 대기열은 AWS Key Management Service (AWS KMS) KMS 키를 사용하여 데이터 암호화 키를 생성합니다. 이 예시에서는 Amazon AWS SQS의 관리형 KMS 키를 사용합니다. SSE 사용 및 KMS 키의 역할에 대한 자세한 정보는 Amazon SQS의 저장 중 암호화 섹션을 참조하세요.

기존 대기열에 SSE 추가

기존 대기열에 대해 서버 측 암호화를 활성화하려면 `SetQueueAttributes` 메서드를 사용하여 `KmsMasterKeyId` 속성을 설정합니다.

다음 코드 예제는 를 Amazon AWS KMS key SQS의 AWS 관리형 KMS 키로 설정합니다. 또한 이 예제에서는 AWS KMS key 재사용 기간을 140초로 설정합니다.

예제 코드를 실행하기 전에 자격 증명을 설정했는지 확인하십시오. AWS 자세한 내용은 AWS SDK for Java 2.x 개발자 안내서의 개발용 AWS 자격 증명 및 지역 설정을 참조하십시오.

```
// 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);
```

대기열에 SSE 비활성화

기존 대기열의 서버 측 암호화를 비활성화하려면 `SetQueueAttributes` 메서드를 사용하여 `KmsMasterKeyId` 속성을 빈 문자열로 설정하세요.

> **Important**
>
> `null`은 유효한 `KmsMasterKeyId` 값이 아닙니다.

SSE를 사용하여 대기열 생성

대기열을 생성할 때 SSE를 활성화하려면 `CreateQueue` API 메서드에 `KmsMasterKeyId` 속성을 추가합니다.

다음 예제에서는 SSE가 활성화된 새 대기열을 만듭니다. 대기열은 Amazon SQS에 AWS 관리형 KMS 키를 사용합니다. 또한 이 예제에서는 AWS KMS key 재사용 기간을 160초로 설정합니다.

예제 코드를 실행하기 전에 AWS 자격 증명을 설정했는지 확인하십시오. 자세한 내용은 AWS SDK for Java 2.x 개발자 안내서의 개발용 AWS 자격 증명 및 지역 설정을 참조하십시오.

```
// 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);
```

SSE 속성 검색

대기열 속성 검색에 대한 자세한 내용은 Amazon Simple Queue Service API 참조의 예를 참조합니다.

특정 대기열의 KMS 키 ID 또는 데이터 키 재사용 기간을 검색하려면 `GetQueueAttributes` 메서드를 실행하고 `KmsMasterKeyId` 및 `KmsDataKeyReusePeriodSeconds` 값을 검색합니다.

Amazon SQS 대기열에 대한 태그 구성

비용 할당 태그를 사용하여 Amazon SQS 대기열을 구성하고 식별할 수 있습니다. 다음 예제에서는 AWS SDK for Java를 사용하여 태그를 구성하는 방법을 보여줍니다. 자세한 정보는 Amazon SQS 비용 할당 태그을 참조하세요.

예제 코드를 실행하기 전에 AWS 자격 증명을 설정했는지 확인하십시오. 자세한 내용은 AWS SDK for Java 2.x 개발자 안내서의 개발용 AWS 자격 증명 및 지역 설정을 참조하십시오.

태그 나열

대기열의 태그를 나열하려면 ListQueueTags 메서드를 사용합니다.

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

태그 추가 또는 업데이트

대기열에 태그 값을 추가하거나 업데이트하려면 TagQueue 메서드를 사용합니다.

```
 // 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);
```

태그 제거

대기열에서 하나 이상의 태그를 삭제하려면 `UntagQueue` 메서드를 사용합니다. 다음 예제에서는 `Accounting ID` 태그를 제거합니다.

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

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

Amazon SQS 대기열에 메시지 속성 전송

메시지 속성을 사용하여 메시지에 구조화된 메타데이터(예: 타임스탬프, 지리 공간 데이터, 서명 및 식별자)를 포함할 수 있습니다. 자세한 정보는 Amazon SQS 메시지 속성을 참조하세요.

예제 코드를 실행하기 전에 AWS 자격 증명을 설정했는지 확인하십시오. 자세한 내용은 AWS SDK for Java 2.x 개발자 안내서의 개발용 AWS 자격 증명 및 지역 설정을 참조하십시오.

속성 정의

메시지에 대한 속성을 정의하려면 `MessageAttributeValue` 데이터 유형을 사용하는 다음 코드를 추가합니다. 자세한 내용은 메시지 속성 구성 요소 및 메시지 속성 데이터 형식 섹션을 참조하세요.

는 메시지 본문과 메시지 속성 체크섬을 AWS SDK for Java 자동으로 계산하여 Amazon SQS가 반환하는 데이터와 비교합니다. 자세한 내용은 AWS SDK for Java 2.x 개발자 안내서 및 다른 프로그래밍 언어의 메시지 속성의 MD5 메시지 다이제스트 계산을 참조하세요.

String

이 예제에서는 값 Jane을 사용하여 Name이라는 String 속성을 정의합니다.

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

Number

이 예제에서는 값 230.00000000000000001을 사용하여 AccurateWeight이라는 Number 속성을 정의합니다.

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("AccurateWeight", new MessageAttributeValue()
.withDataType("Number")
.withStringValue("230.00000000000000001"));
```

Binary

이 예제에서는 초기화되지 않은 10바이트 어레이의 값을 사용하여 ByteArray라는 Binary 속성을 정의합니다.

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

String (custom)

이 예제에서는 값 ABC123456을 사용하여 EmployeeId라는 사용자 지정 속성 String.EmployeeId를 정의합니다.

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

```
.withDataType("String.EmployeeId")
.withStringValue("ABC123456"));
```

Number (custom)

이 예제에서는 값 `000123456`을 사용하여 `AccountId`라는 사용자 지정 속성 `Number.AccountId`를 정의합니다.

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

> **Note**
>
> 기본 데이터 입력이 `Number`이므로 `ReceiveMessage` 메서드가 `123456`을 반환합니다.

Binary (custom)

이 예제에서는 초기화되지 않은 10바이트 어레이의 값을 사용하여 `ApplicationIcon`이라는 사용자 지정 속성 `Binary.JPEG`를 정의합니다.

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

속성을 포함하는 메시지 전송

이 예에서는 메시지를 보내기 전에 `SendMessageRequest`에 속성을 추가합니다.

```
// 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

선입선출(FIFO) 대기열로 메시지를 보내는 경우 메시지 그룹 ID를 제공한 후 `sendMessage` 메서드가 실행되는지 확인하세요.
`SendMessage` 대신 `SendMessageBatch` 메서드를 사용하는 경우 배치의 개별 메시지에 대해 메시지 속성을 지정해야 합니다.

Amazon SQS API 작업

이 섹션에서는 Amazon SQS 엔드포인트 생성, GET 및 POST 메서드를 사용하여 쿼리 API 요청 만들기, 배치 API 작업 사용에 대해 설명합니다. 파라미터, 오류, 예제, 데이터 유형 등 Amazon SQS 작업에 대한 자세한 내용은 Amazon Simple Queue Service API 참조를 참조하세요.

다양한 프로그래밍 언어를 사용하여 Amazon SQS에 액세스하려는 경우 다음과 같은 자동 기능을 제공하는 AWS SDK를 사용할 수도 있습니다.

- 서비스 요청에 대한 암호화 서명
- 요청 재시도
- 오류 응답 처리

명령줄 도구 정보는 AWS CLI 명령 참조 및 AWS Tools for PowerShell Cmdlet 참조의 Amazon SQS 섹션을 참조하세요.

JSON 프로토콜을 사용하는 Amazon SQS API AWS

Amazon SQS는 지정된 SDK AWS 버전의 모든 Amazon SQS API에 대한 전송 메커니즘으로 JSON 프로토콜을 사용합니다.AWS AWS JSON 프로토콜은 더 높은 처리량, 더 짧은 지연 시간, 더 빠른 통신을 제공합니다. application-to-application AWS JSON 프로토콜은 쿼리 프로토콜에 비해 요청 및 응답의 직렬화/역직렬화에 더 효율적입니다. AWS 여전히 AWS 쿼리 프로토콜을 SQS API와 함께 사용하고 싶다면 Amazon SQS 쿼리 프로토콜을 AWS 지원하는 SDK 버전을 Amazon SQS API에서 사용되는 AWS JSON 프로토콜에는 어떤 언어가 지원되나요? 참조하십시오. AWS

Amazon SQS는 AWS JSON 프로토콜을 사용하여 AWS SDK 클라이언트 (예: 자바, Python, Golang 등) 와 Amazon SQS 서버 간에 JavaScript 통신합니다. Amazon SQS API 작업의 HTTP 요청은 JSON 형식의 입력을 수락합니다. Amazon SQS 작업이 실행되고 실행 응답이 JSON 형식으로 SDK 클라이언트에 다시 전송됩니다. AWS 쿼리에 비해 AWS JSON은 클라이언트와 서버 간에 데이터를 전송하는 데 더 간단하고 빠르며 효율적입니다.

- AWS JSON 프로토콜은 Amazon SQS 클라이언트와 서버 사이에서 중재자 역할을 합니다.
- 서버는 Amazon SQS 작업이 생성되는 프로그래밍 언어를 이해하지 못하지만 JSON 프로토콜은 이해합니다 AWS .
- AWS JSON 프로토콜은 Amazon SQS 클라이언트와 서버 간의 직렬화 (객체를 JSON 형식으로 변환) 및 역직렬화 (JSON 형식을 객체로 변환) 를 사용합니다.

Amazon SQS의 AWS JSON 프로토콜에 대한 자세한 내용은 을 참조하십시오. Amazon SQS AWS JSON 프로토콜 FAQ

AWS JSON 프로토콜은 지정된AWS SDK 버전에서 사용할 수 있습니다. 언어 변형에 따른 SDK 버전 및 릴리스 날짜를 검토하려면AWS SDK 및 도구 참조 안내서의 AWS SDK 및 도구 버전 지원 매트릭스를 참조하세요.

주제

- Amazon SQS에서 AWS JSON 프로토콜을 사용하여 쿼리 API 요청 생성
- Amazon SQS에서 AWS 쿼리 프로토콜을 사용하여 쿼리 API 요청 생성
- Amazon SQS에 대한 요청 인증
- Amazon SQS 배치 작업

Amazon SQS에서 AWS JSON 프로토콜을 사용하여 쿼리 API 요청 생성

이 섹션에서는 Amazon SQS 엔드포인트를 생성하고, POST 요청을 만들고 응답을 해석하는 방법을 설명합니다.

> **Note**
>
> AWS JSON 프로토콜은 대부분의 언어 변형에서 지원됩니다. 지원되는 언어 변형의 전체 목록은 Amazon SQS API에서 사용되는 AWS JSON 프로토콜에는 어떤 언어가 지원되나요? 섹션을 참조하세요.

주제

- 엔드포인트 구성
- POST 요청 만들기
- 아마존 SQS JSON API 응답 해석
- Amazon SQS AWS JSON 프로토콜 FAQ

엔드포인트 구성

Amazon SQS 대기열로 작업하려면 엔드포인트를 생성해야 합니다. Amazon SQS 엔드포인트에 대한 자세한 내용은 Amazon Web Services 일반 참조의 다음 페이지를 참조하세요.

- 리전 엔드포인트
- Amazon Simple Queue Service 엔드포인트 및 할당량

모든 Amazon SQS 엔드포인트는 독립적입니다. 예를 들어 대기열 두 개에 이름이 지정되고 한 대기열에 엔드포인트가 MyQueuesqs.us-east-2.amazonaws.com있고 다른 대기열에 엔드포인트가 `sqs.eu-west-2.amazonaws.com` 있는 경우 두 대기열은 서로 데이터를 공유하지 않습니다.

다음은 대기열을 생성하는 요청을 만드는 엔드포인트의 예제입니다.

```
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**
>
> 대기열 이름과 대기열 URL은 대소문자를 구분합니다.
>
> *AUTHPARAMS*의 구조는 API 요청의 서명에 따라 달라집니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS API 요청 서명을 참조하십시오.

POST 요청 만들기

Amazon SQS POST 요청은 쿼리 파라미터를 HTTP 요청 본문에 포함시켜 보냅니다.

다음은 `X-Amz-Target`이 `AmazonSQS.<operationName>`으로 설정된 HTTP 헤더와 `Content-Type`이 `application/x-amz-json-1.0`으로 설정된 HTTP 헤더의 예입니다.

```
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",
}
```

이 HTTP POST 요청은 Amazon SQS 대기열로 메시지를 전송합니다.

> **Note**
>
> HTTP 헤더 `X-Amz-Target` 및 `Content-Type` 모두 필수입니다.
> HTTP 클라이언트는 클라이언트의 HTTP 버전에 따라 HTTP 요청에 다른 항목을 추가할 수 있습니다.

아마존 SQS JSON API 응답 해석

작업 요청에 대한 응답으로 Amazon SQS는 요청 결과를 포함하는 JSON 데이터 구조를 반환합니다. 자세한 내용은 Amazon Simple Queue Service API 참조의 개별 작업 및 Amazon SQS AWS JSON 프로토콜 FAQ를 참조하세요.

주제

- 성공적인 JSON 응답 구조
- JSON 오류 응답 구조

성공적인 JSON 응답 구조

요청이 성공하면 기본 응답 요소는 요청의 범용 고유 식별자(UUID)와 기타 추가된 응답 필드를 포함하는 x-amzn-RequestId입니다. 예를 들어 다음 CreateQueue 응답에는 생성된 대기열의 URL이 포함된 QueueUrl 필드가 포함되어 있습니다.

```
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"
}
```

JSON 오류 응답 구조

요청이 실패하면 Amazon SQS는 HTTP 헤더와 본문을 포함한 기본 응답을 반환합니다.

HTTP 헤더에서 x-amzn-RequestId에는 요청의 UUID가 포함됩니다. x-amzn-query-error에는 오류 유형, 오류가 생산자 오류인지 소비자 오류인지 여부라는 두 가지 정보가 포함되어 있습니다.

응답 본문에서 "__type"은 기타 오류 세부 정보를 나타내고, Message는 오류 조건을 읽을 수 있는 형식으로 나타냅니다.

다음은 JSON 형식의 오류 응답 예제입니다.

```
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."
}
```

Amazon SQS AWS JSON 프로토콜 FAQ

Amazon SQS에서 AWS JSON 프로토콜을 사용하는 방법에 대해 자주 묻는 질문입니다.

AWS JSON 프로토콜이란 무엇이며 기존 Amazon SQS API 요청 및 응답과 어떻게 다릅니까?

JSON은 이기종 시스템 간 통신에 가장 널리 사용되고 수용되는 연결 방법 중 하나입니다. Amazon SQS는 JSON을 매체로 사용하여 AWS SDK 클라이언트 (예: 자바, Python, Golang 등) 와 JavaScript Amazon SQS 서버 간에 통신합니다. Amazon SQS API 작업의 HTTP 요청은 JSON 형식의 입력을 수락합니다. Amazon SQS 작업이 실행되고 실행 응답이 JSON 형식으로 SDK 클라이언트에 다시 공유됩니다. JSON은 AWS 쿼리에 비해 클라이언트와 서버 간에 데이터를 전송하는 데 효율적입니다.

- Amazon SQS AWS JSON 프로토콜은 Amazon SQS 클라이언트와 서버 사이에서 중재자 역할을 합니다.
- 서버는 Amazon SQS 작업이 생성되는 프로그래밍 언어를 이해하지 못하지만 JSON 프로토콜은 이해합니다 AWS .
- Amazon SQS AWS JSON 프로토콜은 Amazon SQS 클라이언트와 서버 간의 직렬화 (객체를 JSON 형식으로 변환) 및 역직렬화 (JSON 형식을 객체로 변환) 를 사용합니다.

Amazon SQS용 AWS JSON 프로토콜을 시작하려면 어떻게 해야 합니까?

Amazon SQS에서 메시징 속도를 높이기 위해 최신 AWS SDK 버전을 시작하려면 SDK를 지정된 버전 또는 후속 버전으로 AWS 업그레이드하십시오. SDK 클라이언트에 대해 자세히 알아보려면 아래 표의 안내서 열을 참조하세요.

다음은 Amazon SQS API와 함께 사용할 수 있는 AWS JSON 프로토콜의 여러 언어 변형에 대한 SDK 버전 목록입니다.

언어	SDK 클라이언트 리포지토리	필수 SDK 클라이언트 버전	안내서
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 for Java

언어	SDK 클라이언트 리포지토리	필수 SDK 클라이언트 버전	안내서
Java 2.x	aws/aws-sdk-java-v2	2.21.19	AWS SDK for Java
JavaScript v2.x	aws/aws-sdk-js	v2.1492.0	JavaScript 켜져 있습니다 AWS
JavaScript v3.x	aws/aws-sdk-js-v3	v3.447.0	JavaScript 켜져 있습니다 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)
awscli	AWS CLI	1.29.82	AWS Command Line Interface
Ruby	aws/aws-sdk-ruby	1.67.0	AWS SDK for Ruby

Amazon SQS 워크로드에 JSON 프로토콜을 활성화하면 어떤 위험이 있나요?

사용자 지정 AWS SDK 구현 또는 사용자 지정 클라이언트와 AWS SDK의 조합을 사용하여 쿼리 기반 (일명 XML 기반) 응답을 AWS 생성하는 Amazon SQS와 상호 작용하는 경우 JSON 프로토콜과 호환되지 않을 수 있습니다. AWS 문제가 발생하는 경우 AWS Support에 문의하세요.

이미 최신 AWS SDK 버전을 사용하고 있는데 오픈소스 솔루션이 JSON을 지원하지 않으면 어떻게 되나요?

SDK 버전을 사용 중인 것보다 이전 버전으로 변경해야 합니다. 자세한 내용은 Amazon SQS용 AWS JSON 프로토콜을 시작하려면 어떻게 해야 합니까? 을 참조하십시오. AWS 에 나열된 SDK 버전은 Amazon SQS API용 JSON 유선 프로토콜을 Amazon SQS용 AWS JSON 프로토콜을 시작하려면 어떻게 해야 합니까? 사용합니다. AWS SDK를 이전 버전으로 변경하면 Amazon SQS API가 쿼리를 사용합니다. AWS

Amazon SQS API에서 사용되는 AWS JSON 프로토콜에는 어떤 언어가 지원되나요?

Amazon SQS는 AWS SDK가 일반적으로 사용 가능한 모든 언어 변형 (GA) 을 지원합니다. 현재 Kotlin, Rust 또는 Swift는 지원하지 않습니다. 다른 언어 변형에 대해 자세히 알아보려면 AWS에서의 빌드를 위한 도구를 참조하세요.

Amazon SQS API에서 사용되는 AWS JSON 프로토콜이 지원되는 리전

Amazon SQS는 Amazon SQS를 사용할 수 있는 모든 AWS 지역에서 AWS JSON 프로토콜을 지원합니다.

JSON 프로토콜을 사용하여 AWS Amazon SQS용 특정 AWS SDK 버전으로 업그레이드할 때 어떤 지연 시간 개선을 기대할 수 있습니까?

AWS JSON 프로토콜은 쿼리 프로토콜에 비해 요청 및 응답의 직렬화 및 역직렬화에 더 효율적입니다. AWS 5KB 메시지 페이로드에 대한 AWS 성능 테스트에 따르면 Amazon SQS용 JSON 프로토콜은 메시지 처리 지연 시간을 최대 23% end-to-end 줄이고 애플리케이션 클라이언트 측 CPU 및 메모리 사용량을 줄입니다.

AWS 쿼리 프로토콜은 더 이상 사용되지 않나요?

AWS 쿼리 프로토콜은 계속 지원될 예정입니다. Amazon SQS용 AWS JSON 프로토콜을 시작하는 방법에 나열된 버전 이외의 이전 버전을 AWS SDK 버전으로 설정하면 AWS 쿼리 프로토콜을 계속 사용할 수 있습니다.

AWS JSON 프로토콜에 대한 자세한 정보는 어디에서 찾을 수 있나요?

Smithy 설명서의 AWS JSON 1.0 프로토콜에서 JSON 프로토콜에 대한 자세한 내용을 확인할 수 있습니다. AWS JSON 프로토콜을 사용하는 Amazon SQS API 요청에 대한 자세한 내용은 Amazon SQS에서 AWS JSON 프로토콜을 사용하여 쿼리 API 요청 생성 섹션을 참조하세요.

Amazon SQS에서 AWS 쿼리 프로토콜을 사용하여 쿼리 API 요청 생성

이 섹션에서는 Amazon SQS 엔드포인트를 생성하고, GET 및 POST 요청을 만들고 응답을 해석하는 방법을 설명합니다.

주제

엔드포인트 구성

Amazon SQS 대기열을 사용하려면 엔드포인트를 생성해야 합니다. Amazon SQS 엔드포인트에 대한 자세한 내용은 Amazon Web Services 일반 참조의 다음 페이지를 참조하세요.

- 리전 엔드포인트
- Amazon Simple Queue Service 엔드포인트 및 할당량

모든 Amazon SQS 엔드포인트는 독립적입니다. 예를 들어, 대기열 두 개에 이름이 지정되고 한 대기열에 엔드포인트가 MyQueuesqs.us-east-2.amazonaws.com있고 다른 대기열에 엔드포인트가 sqs.eu-west-2.amazonaws.com 있는 경우 두 대기열은 서로 데이터를 공유하지 않습니다.

다음은 대기열을 생성하는 요청을 만드는 엔드포인트의 예제입니다.

```
https://sqs.eu-west-2.amazonaws.com/
?Action=CreateQueue
&DefaultVisibilityTimeout=40
&QueueName=MyQueue
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> 대기열 이름과 대기열 URL은 대소문자를 구분합니다.

*AUTHPARAMS*의 구조는 API 요청의 서명에 따라 달라집니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS API 요청 서명을 참조하십시오.

GET 요청 만들기

Amazon SQS GET 요청은 다음으로 구성된 URL로 생성됩니다.

- 엔드포인트 - 요청이 작용하는 리소스(대기열 이름 및 URL)입니다. 예: `https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue`
- 작업 - 엔드포인트에서 수행하려는 작업입니다. 물음표(?)는 엔드포인트와 작업을 구분합니다. 예: `?Action=SendMessage&MessageBody=Your%20Message%20Text`
- 파라미터 - 요청 파라미터입니다. 각 파라미터는 앰퍼샌드(&)로 구분됩니다. 예: `&Version=2012-11-05&AUTHPARAMS`

다음은 Amazon SQS 대기열로 메시지를 보내는 GET 요청의 예제입니다.

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue
?Action=SendMessage&MessageBody=Your%20message%20text
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> 대기열 이름과 대기열 URL은 대소문자를 구분합니다.
> GET 요청은 URL 형식이므로 모든 파라미터 값을 URL 인코딩해야 합니다. URL에 공백이 허용되지 않기 때문에 각 공백은 %20으로 URL 인코딩됩니다. 나머지 예제는 URL 인코딩되지 않아 쉽게 판독할 수 있습니다.

POST 요청 만들기

Amazon SQS POST 요청은 쿼리 파라미터를 HTTP 요청 본문에 포함시켜 보냅니다.

다음은 `Content-Type`이 `application/x-www-form-urlencoded`로 설정된 HTTP 헤더의 예제입니다.

```
POST /123456789012/MyQueue HTTP/1.1
```

```
Host: sqs.us-east-2.amazonaws.com
Content-Type: application/x-www-form-urlencoded
```

헤더 뒤에 Amazon SQS 대기열로 메시지를 보내는 form-urlencoded GET 요청이 옵니다. 각 파라미터는 앰퍼샌드(&)로 구분됩니다.

```
Action=SendMessage
&MessageBody=Your+Message+Text
&Expires=2020-10-15T12%3A00%3A00Z
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> Content-Type HTTP 헤더만 필수 항목입니다. *AUTHPARAMS*는 GET 요청과 동일합니다. HTTP 클라이언트는 클라이언트의 HTTP 버전에 따라 HTTP 요청에 다른 항목을 추가할 수 있습니다.

Amazon SQS XML API 응답 해석

작업 요청에 대한 응답으로 Amazon SQS는 요청 결과를 포함하는 XML 데이터 구조를 반환합니다. 자세한 내용은 Amazon Simple Queue Service API 참조의 개별 작업을 참조하세요.

주제

- 성공적인 XML 응답 구조
- XML 오류 응답 구조

성공적인 XML 응답 구조

요청이 성공하면 주 응답 요소는 작업의 이름을 따서 이름이 지정되고 Response가 추가됩니다(예: *ActionName*Response).

이 요소에는 다음과 같은 하위 요소가 포함되어 있습니다.

- ***ActionName*Result** - 작업별 요소를 포함합니다. 예를 들어 CreateQueueResult 요소는 QueueUrl 요소를 포함하며, 후자는 생성된 대기열의 URL을 포함합니다.

- **ResponseMetadata** - RequestId를 포함하며, 결과적으로 요청의 Universal Unique Identifier(UUID)를 포함합니다.

다음은 XML 형식의 성공적인 응답의 예입니다.

```
<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 오류 응답 구조

요청이 실패한 경우 Amazon SQS는 항상 주 응답 요소 ErrorResponse를 반환합니다. 이 요소에는 Error 요소와 RequestId 요소가 포함되어 있습니다.

Error 요소에는 다음과 같은 하위 요소가 포함되어 있습니다.

- **Type** - 오류가 생성자 오류였는지 소비자 오류였는지 지정합니다.
- **Code** - 오류 유형을 지정합니다.
- **Message** - 오류 조건을 읽을 수 있는 형식으로 지정합니다.
- **Detail** - (선택 사항) 오류에 대한 추가 세부 정보를 지정합니다.

RequestId 요소에는 요청의 UUID가 포함되어 있습니다.

다음은 XML 형식의 오류 응답의 예입니다.

```
<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>
```

Amazon SQS에 대한 요청 인증

인증은 요청을 보내는 당사자를 식별하고 확인하는 프로세스입니다. 인증의 첫 단계에서 AWS 는 생성자의 자격 증명과 생성자가 AWS를 사용하도록 등록되었는지 여부를 확인합니다(자세한 내용은 1단계: AWS 계정 및 IAM 사용자 생성 참조). 다음으로, 다음 AWS 절차를 준수합니다.

1. 생성자(발신자)는 필요한 자격 증명을 획득합니다.
2. 생성자는 요청과 자격 증명은 소비자(수신자)에게 전송합니다.
3. 소비자는 자격 증명을 사용하여 생성자가 요청을 전송했는지 여부를 확인합니다.
4. 다음 중 하나가 발생합니다.
   - 인증에 성공할 경우, 소비자는 요청을 처리합니다.
   - 인증에 실패할 경우 소비자는 요청을 거부하고 오류를 반환합니다.

주제

- HMAC-SHA를 사용한 기본 인증 프로세스
- 1부: 사용자의 요청
- 2부: 응답 양식 AWS

HMAC-SHA를 사용한 기본 인증 프로세스

쿼리 API를 사용하여 Amazon SQS에 액세스할 때는 요청을 인증할 수 있도록 다음 항목을 제공해야 합니다.

- 사용자를 식별하는 AWS 액세스 키 ID로 AWS 계정, 보안 액세스 키를 조회하는 데 AWS 사용됩니다.
- 비밀 액세스 키(사용자와 AAA에게만 알려진 공유 비밀 - 자세한 내용은 RFC2104 참조)를 사용하여 계산된 HMAC-SHA 요청 서명입니다. AWS SDK는 서명 프로세스를 처리합니다. 그러나 HTTP 또는 HTTPS를 통해 쿼리 요청을 제출하는 경우 모든 쿼리 요청에 서명을 포함시켜야 합니다.

1. 서명 버전 4 서명 키를 생성합니다. 자세한 내용은 Java를 사용하여 서명 키 생성을 참조하십시오.

> **Note**
>
> Amazon SQS는 이전 버전보다 SHA256 기반 보안과 성능이 강화된 서명 버전 4를 지원합니다. Amazon SQS를 사용하는 새 애플리케이션을 만들 때 서명 버전 4를 사용합니다.

2. 요청 서명을 base64로 인코딩합니다. 다음 샘플 Java 코드는 이를 수행합니다.

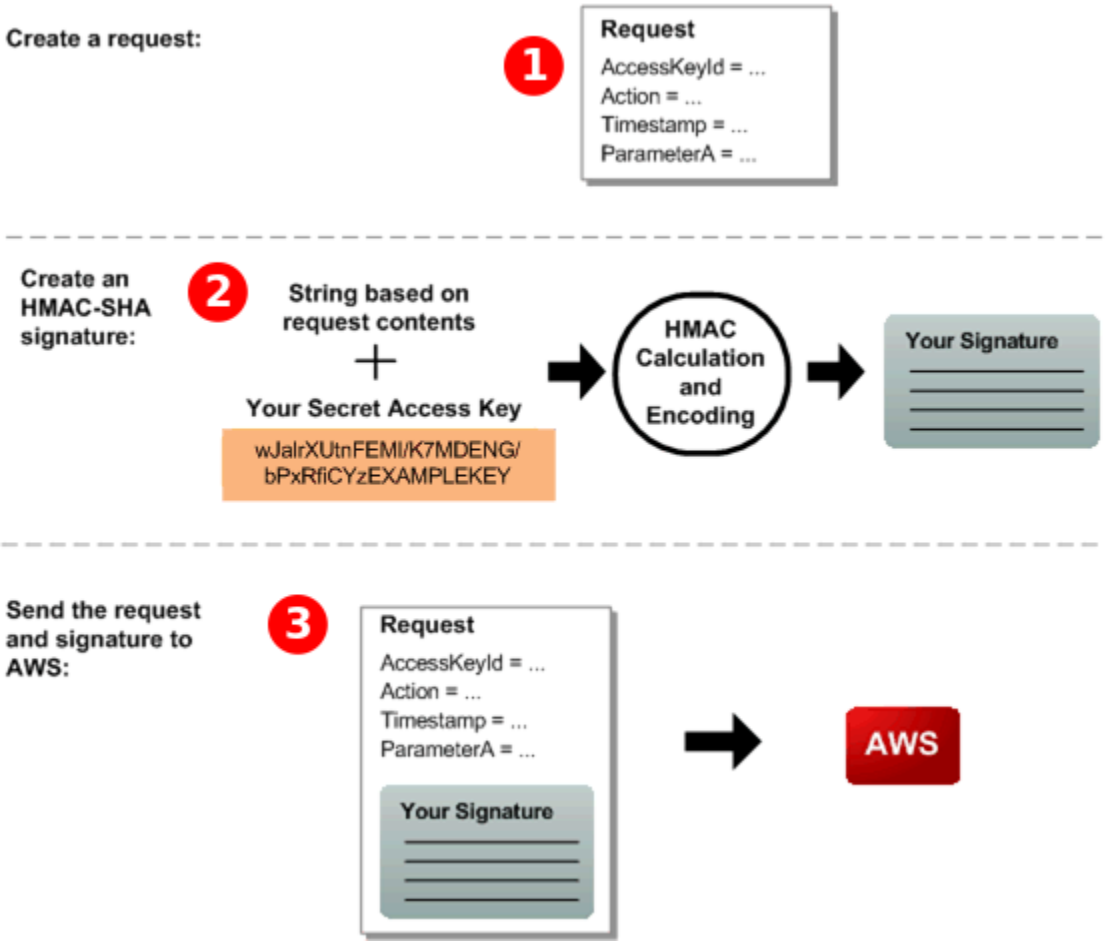
```
package amazon.webservices.common;

// 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);
    }
}
```

- 요청의 타임스탬프(또는 만료)입니다. 요청에 사용하는 타임스탬프는 시간, 분, 초를 포함한 전체 날짜가 포함된 dateTime 객체여야 합니다. 예를 들면 2007-01-31T23:59:59Z입니다. 필수는 아니지만 협정 세계시(그리니치 표준시) 시간대를 사용하여 객체를 제공하는 것이 좋습니다.

> **Note**
>
> 서버 시간을 올바르게 설정했는지 확인하십시오. 만료일이 아닌 타임스탬프를 지정하는 경우 요청은 지정된 시간이 지난 후 15분 후에 자동으로 만료됩니다 (서버에서 현재 시간보다 15분 이상 빠른 타임스탬프가 있는 요청은 처리하지AWS 않음). AWS .NET을 사용하는 경우, 시간 정밀도의 유효 자릿수에 대한 해석이 다양하기 때문에 너무 자세한 타임스탬프를 전송하지 않아야 합니다. 이 경우 정밀도가 1밀리초 이하인 dateTime 객체를 수동으로 생성해야 합니다.

1부: 사용자의 요청

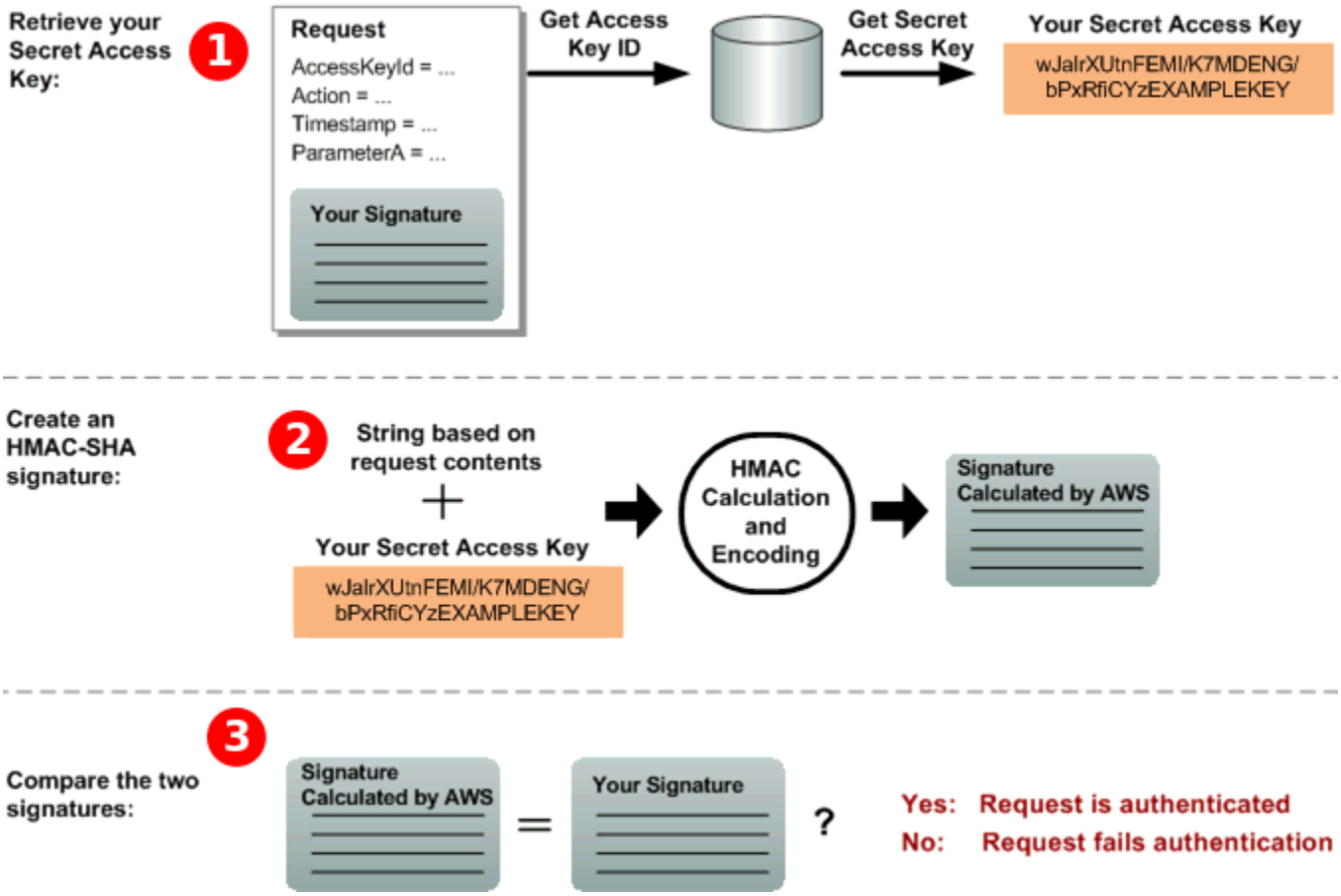
다음은 HMAC-SHA 요청 서명을 사용하여 AWS 요청을 인증하기 위해 따라야 하는 프로세스입니다.



1. 로 AWS요청을 구성하십시오.
2. 보안 액세스 키를 사용하여 키가 추가된 해시 메시지 인증 코드(HMAC-SHA) 서명을 계산합니다.
3. 요청에 서명과 액세스 키 ID를 포함한 다음 으로 요청을 보내십시오 AWS.

2부: 응답 양식 AWS

AWS 이에 대한 응답으로 다음 프로세스를 시작합니다.



1. AWS 액세스 키 ID를 사용하여 보안 액세스 키를 조회합니다.
2. AWS 요청에서 전송한 서명을 계산하는 데 사용한 것과 동일한 알고리즘을 사용하여 요청 데이터와 보안 액세스 키에서 서명을 생성합니다.
3. 다음 중 하나가 발생합니다.
   - AWS 생성된 서명이 요청에서 보낸 서명과 일치하면 요청을 인증한 AWS 것으로 간주합니다.
   - 비교에 실패하면 요청이 삭제되고 오류가 AWS 반환됩니다.

Amazon SQS 배치 작업

비용을 절감하거나 단일 작업으로 최대 10개 메시지를 조작하려면 다음 작업을 사용하면 됩니다.

- SendMessageBatch
- DeleteMessageBatch
- ChangeMessageVisibilityBatch

쿼리 API 또는 Amazon SQS 배치 작업을 지원하는 AWS SDK를 사용하여 배치 기능을 활용할 수 있습니다.

> **Note**
>
> 단일 SendMessageBatch 호출로 보내는 모든 메시지의 총 크기는 262,144바이트 (256KiB)를 초과할 수 없습니다.
>
> SendMessageBatch, DeleteMessageBatch 또는 ChangeMessageVisibilityBatch에 대한 권한을 명시적으로 설정할 수 없습니다. SendMessage, DeleteMessage 또는 ChangeMessageVisibility에 대한 권한을 설정하면 작업의 해당 배치 버전에 대한 권한도 설정됩니다.
>
> Amazon SQS 콘솔은 배치 작업을 지원하지 않습니다.

주제

Amazon SQS를 통한 클라이언트 측 버퍼링 및 요청 일괄 처리 지원

AWS SDK for Java는 Amazon SQS에 액세스하는 AmazonSQSBufferedAsyncClient를 포함합니다. 이 클라이언트는 클라이언트 측 버퍼링을 사용하여 요청 배치 처리를 간소화합니다. 즉 클라이언트에서 한 호출을 먼저 버퍼링한 후 배치 요청 형태로 Amazon SQS로 전송합니다.

클라이언트 측 버퍼링을 통해 최대 10개의 요청을 버퍼링하여 배치 요청으로 전송할 수 있으므로 Amazon SQS 사용 비용이 절감되고 전송되는 요청 수가 줄어듭니다. AmazonSQSBufferedAsyncClient 버퍼는 동기식 및 비동기식 호출을 모두 버퍼링합니다. 배치 처리된 요청과 긴 폴링 지원을 통해 처리량도 높일 수 있습니다. 자세한 정보는 Amazon SQS를 통한 수평 조정 및 작업 일괄 처리를 사용하여 처리량 증가을 참조하세요.

AmazonSQSBufferedAsyncClient는 AmazonSQSAsyncClient와 동일한 인터페이스를 구현하므로 AmazonSQSAsyncClient에서 AmazonSQSBufferedAsyncClient로 마이그레이션하려면 일반적으로 기존 코드를 최소한만 변경하면 됩니다.

> **Note**
>
> Amazon SQS의 버퍼링된 비동기식 클라이언트는 현재 FIFO 대기열을 지원하지 않습니다.

주제

AmazonSQS 클라이언트 BufferedAsync 사용

시작하기 전에 Amazon SQS 설정의 단계를 완료해야 합니다.

> **Important**
>
> 는 AWS SDK for Java 2.x 현재 와 호환되지 않습니다. `AmazonSQSBufferedAsyncClient`

`AmazonSQSAsyncClient`를 기반으로 새 `AmazonSQSBufferedAsyncClient`를 만들 수 있습니다. 예를 들면 다음과 같습니다.

```
// Create the basic Amazon SQS async client
final AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient();

// Create the buffered client
final AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync);
```

새 `AmazonSQSBufferedAsyncClient`를 생성한 후 이것을 이용해서 Amazon SQS에 다중 요청을 전송할 수 있습니다(`AmazonSQSAsyncClient`와 마찬가지로). 예를 들면 다음과 같습니다.

```
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);
```

BufferedAsyncAmazonSQS 클라이언트 구성

AmazonSQSBufferedAsyncClient는 대부분의 사용 사례에 적용되는 설정으로 사전 구성됩니다. AmazonSQSBufferedAsyncClient를 추가로 구성할 수 있습니다. 예를 들면 다음과 같습니다.

1. 필요한 구성 파라미터로 QueueBufferConfig 클래스의 인스턴스를 만듭니다.
2. AmazonSQSBufferedAsyncClient 생성자에 인스턴스를 제공합니다.

```
// 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 구성 파라미터

파라미터	기본값	설명
longPoll	true	longPoll이 true로 설정된 경우 AmazonSQSBufferedAsyncClient 는 메시지를 사용할 때 긴 폴링을 사용하려고 시도합니다.
longPollWaitTimeoutSeconds	20s	ReceiveMessage 호출이 서버에서 차단되고 빈 수신 결과를 반환하기 전에 메시지가 대기열에 나타나기까지 대기하는 최대 시간(초)입니다.

파라미터	기본값	설명
		Note 긴 폴링을 비활성화하면 이 설정은 영향을 주지 않습니다.
`maxBatchOpenMs`	200ms	발신 호출이 동일한 유형의 메시지를 배치 처리할 다른 호출을 대기하는 최대 시간(밀리초)입니다. 설정이 높을수록 동일한 작업량을 수행하는 데 필요한 배치 수가 감소합니다(그러나 배치의 첫 번째 호출을 대기하는 시간이 길어짐). 이 파라미터를 `0`으로 설정하면, 제출된 요청은 다른 요청을 기다리지 않으므로 사실상 일괄 처리하지 않게 됩니다.
`maxBatchSize`	배치당 요청 10개	단일 요청에서 배치 방식으로 함께 처리되는 최대 메시지 수입니다. 설정이 높을수록 동일한 요청 수를 수행하는 데 필요한 배치 수가 감소합니다. **Note** 배치당 요청 10개는 Amazon SQS에 허용되는 최대값입니다.

파라미터	기본값	설명
`maxBatchSizeBytes`	256KiB	클라이언트가 Amazon SQS에 전송하려고 시도하는 메시지 배치의 최대 크기(바이트 단위)입니다. **Note** 256KiB는 Amazon SQS에 허용되는 최대 값입니다.
`maxDoneReceiveBatches`	배치 10개	`AmazonSQSBufferedAsyncClient` 가 클라이언트 측에서 미리 가져와서 저장하는 수신 배치의 최대 개수입니다. 설정이 높을수록 Amazon SQS에 호출을 전송하지 않고도 만족할 수 있는 수신 요청 개수가 늘어납니다(그러나 미리 가져오는 메시지가 많을수록 메시지가 버퍼에 있는 시간이 길어지므로 표시 제한 시간이 만료될 수 있음). **Note** `0`모든 메시지 프리페치가 비활성화되고 메시지가 요청 시에만 소비됨을 나타냅니다.

파라미터	기본값	설명
`maxInflightOutboundBatches`	배치 5개	동시에 처리될 수 있는 활성 아웃바운드 배치의 최대 개수입니다. 설정이 높을수록 아웃바운드 배치 전송 속도가 빨라지고(CPU 또는 대역폭과 같은 할당량도 영향을 줌), `AmazonSQSBufferedAsyncClient` 에서 사용하는 스레드 수가 많아집니다.
`maxInflightReceiveBatches`	배치 10개	동시에 처리될 수 있는 활성 수신 배치의 최대 개수입니다. 설정이 높을수록 수신 가능한 메시지가 많아지고(CPU 또는 대역폭과 같은 할당량도 영향을 줌), `AmazonSQSBufferedAsyncClient` 에서 사용하는 스레드 수가 많아집니다. **Note** `0`모든 메시지 프리페치가 비활성화되고 메시지가 요청 시에만 소비됨을 나타냅니다.

파라미터	기본값	설명
visibilityTimeoutSeconds	-1	이 파라미터를 0이 아닌 양수 값으로 설정하면, 여기에 설정된 제한 시간 초과는 메시지가 사용되는 대기열에서 설정된 제한 시간 초과를 재정의합니다. **Note** -1은 대기열에서 기본 설정이 선택되어 있음을 나타냅니다. 제한 시간 초과를 0으로 설정할 수 없습니다.

Amazon SQS를 통한 수평 조정 및 작업 일괄 처리를 사용하여 처리량 증가

Amazon SQS 대기열은 매우 높은 처리량을 제공할 수 있습니다. 처리량 할당량에 대한 자세한 내용은 Amazon SQS 메시지 할당량 섹션을 참조하세요.

높은 처리량을 달성하려면 메시지 생성자와 소비자를 수평적으로 조정해야 합니다(더 많은 생성자와 소비자 추가).

주제

- 수평 크기 조정
- 작업 일괄 처리
- 단일 작업 및 배치 요청에 대한 Java 사용 예제

수평 크기 조정

HTTP 요청-응답 프로토콜을 통해 Amazon SQS에 액세스하기 때문에 요청 지연 시간(요청을 시작해서 응답을 받기까지의 시간 간격)은 단일 연결을 사용하여 단일 스레드에서 확보할 수 있는 처리량을

제한합니다. 예를 들어, 동일한 리전에서 Amazon EC2 기반 클라이언트부터 Amazon SQS까지 지연 시간이 평균 20ms인 경우 단일 연결을 통한 단일 스레드의 최대 처리량은 평균 50TPS입니다.

수평적 조정은 메시지 생성자(SendMessage 요청 전송)와 소비자(ReceiveMessage 및 DeleteMessage 요청 전송) 수를 늘리는 것으로 전체 대기열 처리량을 높여줍니다. 다음 세 가지 방법으로 수평 확장할 수 있습니다.

- 클라이언트당 스레드 수 늘리기
- 클라이언트 추가
- 클라이언트당 스레드 수를 늘리고 클라이언트 추가

클라이언트를 더 추가하면 대기열 처리량이 선형적으로 증가합니다. 예를 들어 클라이언트 개수를 2배 늘리면 처리량도 2배가 됩니다.

> **Note**
>
> 수평적으로 확장할 경우 요청을 전송하고 응답을 수신하는 동시 메시지 생산자와 소비자의 수를 지원할 수 있도록 Amazon SQS 클라이언트의 연결이나 스레드가 충분한지 확인해야 합니다. 예를 들어 기본적으로 AWS SDK for Java AmazonSQSClient 클래스의 인스턴스는 Amazon SQS와의 연결을 최대 50개 유지합니다. 동시 생성자와 소비자를 추가로 만들려면 AmazonSQSClientBuilder 객체에 있는 허용되는 생성자 및 소비자 스레드의 최대 개수를 조정해야 합니다. 예를 들면 다음과 같습니다.
>
> ```
> final AmazonSQS sqsClient = AmazonSQSClientBuilder.standard()
>         .withClientConfiguration(new ClientConfiguration()
>                 .withMaxConnections(producerCount + consumerCount))
>         .build();
> ```
>
> AmazonSQSAsyncClient의 경우 충분한 스레드를 사용할 수 있는지도 확인해야 합니다. 이 예제는 Java v. 1.x에서만 작동합니다.

작업 일괄 처리

일괄 처리는 서비스까지 왕복할 때마다 더 많은 작업을 수행합니다(예: SendMessageBatch 요청 하나로 여러 메시지를 전송하는 경우). Amazon SQS 배치 작업은 SendMessageBatch, DeleteMessageBatch 및 ChangeMessageVisibilityBatch입니다. 생산자 또는 소비자를 변경

하지 않고 배치 처리를 활용하려면 Amazon SQS의 버퍼링된 비동기식 클라이언트를 사용하면 됩니다.

> **Note**
>
> `ReceiveMessage`는 한 번에 메시지를 10개 처리할 수 있으므로 `ReceiveMessageBatch` 작업이 없습니다.

일괄 처리는 메시지 하나의 전체 지연 시간을 수락하는 것이 아니라, 배치 작업의 지연 시간을 배치 요청의 여러 메시지로 분산시키는 것입니다(예: `SendMessage` 요청). 각 왕복마다 더 많은 작업을 수행하기 때문에 배치 요청은 스레드와 연결을 보다 효율적으로 사용하므로 처리량이 개선됩니다.

일괄 처리와 수평적 조정을 함께 사용하면 개별 메시지 요청보다 스레드, 연결 및 요청 수가 적은 처리량을 얻습니다. 배치 처리된 Amazon SQS 작업을 사용하여 한 번에 최대 10개 메시지를 전송, 수신 또는 삭제할 수 있습니다. Amazon SQS는 요청별로 요금을 부과하기 때문에 배치 처리 시 비용을 대폭 절감할 수 있습니다.

일괄 처리를 사용하면 애플리케이션에 어느 정도 복잡성이 수반됩니다(예를 들어 애플리케이션이 메시지를 누적한 후 메시지를 전송하고 때로는 응답을 받기까지 대기 시간이 길어짐). 그러나 다음과 같은 상황에서는 일괄 처리가 효과적일 수 있습니다.

- 애플리케이션이 단시간에 많은 메시지를 생성하지만 지연 시간이 매우 길지 않은 경우.
- 제어하지 못하는 이벤트에 대응하여 메시지를 전송해야 하는 일반적인 메시지 생성자와 달리 메시지 소비자가 단독 재량으로 대기열에서 메시지를 가져오는 경우.

> **Important**
>
> 배치의 개별 메시지에 오류가 발생하더라도 배치 요청이 성공할 수 있습니다. 배치 요청 후에는 항상 개별 메시지의 오류 여부를 확인하고 필요에 따라 작업을 다시 시도해야 합니다.

단일 작업 및 배치 요청에 대한 Java 사용 예제

사전 조건

`aws-java-sdk-sqs.jar`, `aws-java-sdk-ec2.jar`, `commons-logging.jar` 패키지를 Java 빌드 클래스 경로에 추가합니다. 다음 예제는 Maven 프로젝트의 `pom.xml` 파일 내에 존재하는 이러한 종속성을 보여줍니다.

```
<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

다음 Java 코드 예제는 단순한 생성자-소비자 패턴을 구현한 것입니다. 기본 스레드는 지정된 시간 동안 1KB 메시지를 처리하는 생성자와 소비자 스레드를 다수 생성합니다. 이 예제에는 단일 작업을 요청하는 생성자와 소비자뿐 아니라 배치 요청을 하는 생성자와 소비자가 포함되어 있습니다.

```
/*
 * 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.
        }
    }
}
```

```
}
```

예제 실행에서 볼륨 측정치 모니터링

Amazon SQS는 전송, 수신 및 삭제된 메시지에 대한 볼륨 지표를 자동으로 생성합니다. 대기열의 모니터링 탭 또는 CloudWatch 콘솔을 통해 해당 지표 및 기타 지표에 액세스할 수 있습니다.

> **Note**
>
> 이 측정치의 경우 대기열을 사용할 수 있을 때까지 최대 15분이 걸릴 수 있습니다.

JMS 및 Amazon SQS에서 작업

Amazon SQS Java 메시징 라이브러리는 이미 JMS를 사용하고 있는 애플리케이션에서 Amazon SQS를 활용하도록 해주는 Amazon SQS용 Java 메시지 서비스(JMS) 인터페이스입니다. 이 인터페이스를 통해 코드 변경을 최소화하면서 Amazon SQS를 JMS 공급자로서 사용할 수 있습니다. AWS SDK for Java와 함께 Amazon SQS Java 메시징 라이브러리를 통해 JMS 연결과 세션뿐 아니라 Amazon SQS 대기열과 메시지를 주고 받는 생성자 및 소비자를 생성할 수 있습니다.

라이브러리는 JMS 1.1 사양에 따라 대기열 (JMS point-to-point 모델) 에 메시지를 보내고 받을 수 있도록 지원합니다. 이 라이브러리는 Amazon SQS 대기열로의 문자, 바이트 또는 객체 메시지 동기식 전송을 지원합니다. 또한, 동기 또는 비동기식 객체 수신도 지원합니다.

JMS 1.1 사양을 지원하는 Amazon SQS Java 메시징 라이브러리의 기능에 대한 자세한 내용은 아마존 SQS는 JMS 1.1 구현을 지원했습니다. 및 Amazon SQS FAQ를 참조하세요.

주제

JMS 및 Amazon SQS로 작업하기 위한 사전 요구 사항

시작하기 전에 다음 사전 요구 사항을 완비해야 합니다.

- Java용 SDK

  프로젝트에 Java용 SDK를 포함시키는 두 가지 방법이 있습니다.

  - Java용 SDK 다운로드 및 설치
  - Maven을 사용하여 Amazon SQS Java 메시징 라이브러리를 다운로드합니다.

  > **Note**
  >
  > Java용 SDK는 종속적으로 포함되어 있습니다.

Java용 SDK 및 Java용 Amazon SQS 확장 클라이언트 라이브러리를 사용하려면 J2SE 개발 키트 8.0 이상이 필요합니다.

Java용 SDK 다운로드에 대한 자세한 내용은 Java용 SDK를 참조하세요.

- Amazon SQS Java 메시징 라이브러리

  Maven을 사용하지 않을 경우, `amazon-sqs-java-messaging-lib.jar` 패키지를 Java 빌드 클래스 경로에 추가해야 합니다. 라이브러리 다운로드에 대한 자세한 내용은 Amazon SQS Java 메시징 라이브러리를 참조하세요.

  > **Note**
  >
  > Amazon SQS Java 메시징 라이브러리에는 Maven과 Spring 프레임워크에 대한 지원이 포함되어 있습니다.
  > Maven, Spring Framework, Amazon SQS Java 메시징 라이브러리를 사용하는 코드 샘플은 Amazon SQS 표준 대기열에서 JMS를 사용하기 위한 작업 자바 예제 섹션을 참조하세요.
  >
  > ```
  > <dependency>
  >   <groupId>com.amazonaws</groupId>
  >   <artifactId>amazon-sqs-java-messaging-lib</artifactId>
  >   <version>1.0.4</version>
  >   <type>jar</type>
  > </dependency>
  > ```

- Amazon SQS 대기열

  Amazon AWS Management Console SQS용, `CreateQueue` API 또는 Amazon SQS Java 메시징 라이브러리에 포함된 래핑된 Amazon SQS 클라이언트를 사용하여 대기열을 생성합니다.

  - AWS Management Console 또는 `CreateQueue` API를 사용하여 Amazon SQS에서 대기열을 만드는 방법에 대한 정보는 대기열 만들기를 참조하세요.
  - Amazon SQS Java 메시징 라이브러리 사용에 대한 자세한 내용은 Amazon SQS Java 메시징 라이브러리 시작 섹션을 참조하세요.

Amazon SQS Java 메시징 라이브러리 시작

Amazon SQS에서 Java 메시지 서비스(JMS) 사용을 시작하려면 이 섹션의 코드 예제를 사용합니다. 다음 섹션에는 JMS 연결 및 세션 생성 방법과 메시지 전송 및 수신 방법이 나와 있습니다.

Amazon SQS Java 메시징 라이브러리에 포함된 래핑된 Amazon SQS 클라이언트 객체는 Amazon SQS 대기열이 존재하는지 확인합니다. 대기열이 존재하지 않는 경우, 클라이언트가 이 대기열을 생성합니다.

JMS 연결 생성

1. 연결 팩토리를 생성하고 이 팩토리에 대한 `createConnection` 메서드를 호출합니다.

   ```
   // 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();
   ```

   `SQSConnection` 클래스는 `javax.jms.Connection`을 확장합니다. JMS 표준 연결 메서드와 더불어 `SQSConnection`은 `getAmazonSQSClient` 및 `getWrappedAmazonSQSClient`와 같은 추가 메서드를 제공합니다. 이 두 메서드를 통해 새로운 대기열 생성과 같이 JMS 사양에 포함되지 않은 관리 작업을 수행할 수 있습니다. 그렇지만, `getWrappedAmazonSQSClient` 메서드는 최신 연결을 통해 사용되는 래핑된 버전의 Amazon SQS 클라이언트도 제공합니다. 래퍼는 클라이언트에서 `JMSException`으로 모든 예외를 변환하므로, `JMSException` 발생이 예상되는 기존 코드에서 보다 쉽게 사용할 수 있습니다.

2. `getAmazonSQSClient` 및 `getWrappedAmazonSQSClient`에서 반환되는 클라이언트 객체를 사용하여 JMS 사양에 포함되지 않은 관리 작업(예: Amazon SQS 대기열 생성 가능)을 수행할 수 있습니다.

   JMS 예외가 예상되는 기존 코드가 있는 경우, `getWrappedAmazonSQSClient`를 사용해야 합니다.

   - `getWrappedAmazonSQSClient`를 사용한 경우, 반환된 클라이언트 객체는 모든 예외를 JMS 예외로 변환합니다.

- `getAmazonSQSClient`를 사용하는 경우, 예외는 모두 Amazon SQS 예외입니다.

Amazon SQS 대기열 생성

래핑된 클라이언트 객체는 Amazon SQS 대기열이 있는지를 확인합니다.

대기열이 존재하지 않는 경우, 클라이언트가 이 대기열을 생성합니다. 대기열이 존재하지 않는 경우, 이 함수는 어느 항목도 반환하지 않습니다. 자세한 내용은 TextMessageSender.java 예제에서 "필요한 경우 대기열 생성"을 참조하십시오.

표준 대기열을 생성하려면

```
// 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");
}
```

FIFO 대기열을 생성하려면

```
// 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**
>
> FIFO 대기열의 이름은 `.fifo` 접미사로 끝나야 합니다.
> `ContentBasedDeduplication` 속성에 대한 자세한 내용은 Amazon SQS에서 정확히 한 번의 프로세싱를 참조하십시오.

메시지 동기식 전송

1. 연결 및 기본 Amazon SQS 대기열이 준비가 되면, AUTO_ACKNOWLEDGE 모드로 트랜잭션 처리되지 않는 JMS 세션을 생성합니다.

```
// Create the nontransacted session with AUTO_ACKNOWLEDGE mode
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
```

2. 대기열에 문자 메시지를 전송하려면, JMS 대기열 자격 증명과 메시지 생성자를 생성합니다.

```
// 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. 문자 메시지를 작성하고 대기열로 전송합니다.

   - 메시지를 표준 대기열로 전송하는 경우 추가 파라미터를 설정할 필요가 없습니다.

```
// Create the text message
TextMessage message = session.createTextMessage("Hello World!");

// Send the message
producer.send(message);
System.out.println("JMS Message " + message.getJMSMessageID());
```

   - 메시지를 FIFO 대기열로 전송하는 경우 메시지 그룹 ID를 설정해야 합니다. 메시지 중복 제거 ID를 설정할 수도 있습니다. 자세한 정보는 아마존 SQS 주요 용어을 참조하세요.

```
// 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"));
```

메시지 동기식 수신

1. 메시지를 수신하려면 동일한 대기열에 대한 소비자를 생성하고 `start` 메서드를 호출합니다.

   언제든지 연결에서 `start` 메서드를 호출할 수 있습니다. 그렇지만, 소비자는 사용자가 메시지를 호출할 때까지 메시지 수신을 시작하지 않습니다.

   ```
   // Create a consumer for the 'MyQueue'
   MessageConsumer consumer = session.createConsumer(queue);
   // Start receiving incoming messages
   connection.start();
   ```

2. 제한 시간을 1초로 설정한 소비자에서 `receive` 메서드를 호출한 다음 수신된 메시지의 내용을 인쇄합니다.

   - 표준 대기열에서 메시지를 수신한 후 해당 메시지의 내용에 액세스할 수 있습니다.

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

   - FIFO 대기열에서 메시지를 수신한 후 메시지 그룹 ID, 메시지 중복 제거 ID 및 시퀀스 번호와 같은 메시지 내용 및 기타 FIFO 관련 메시지 속성에 액세스할 수 있습니다. 자세한 정보는 아마존 SQS 주요 용어을 참조하세요.

     ```
     // 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. 연결과 세션을 종료합니다.

```
// Close the connection (and the session).
connection.close();
```

출력 결과는 다음과 비슷합니다:

```
JMS Message ID:8example-588b-44e5-bbcf-d816example2
Received: Hello World!
```

> **Note**
>
> Spring Framework를 사용하여 이 객체를 초기화할 수 있습니다.
> 추가 정보는 `SpringExampleConfiguration.xml`, `SpringExample.java` 및 `ExampleConfiguration.java` 단원의 `ExampleCommon.java`와 Amazon SQS 표준 대기열에서 JMS를 사용하기 위한 작업 자바 예제에 있는 기타 헬퍼 클래스를 참조하십시오.

객체 전송 및 수신 예제 전체를 보려면, TextMessageSender.java 및 SyncMessageReceiver.java를 참조하십시오.

메시지 비동기식 수신

Amazon SQS Java 메시징 라이브러리 시작의 예제에서 메시지는 `MyQueue`로 전송되고 동기식으로 수신됩니다.

다음 예제는 리스너를 통해 비동기식으로 메시지를 수신하는 방법을 나타냅니다.

1. `MessageListener` 인터페이스를 구현합니다.

```
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();
        }
    }
}
```

메시지를 수신하면 MessageListener 인터페이스의 onMessage 메서드가 호출됩니다. 이 리스너 구현에서 메시지에 저장된 텍스트가 인쇄됩니다.

2. 소비자에서 receive 메서드를 명시적으로 호출하지 않고, 소비자의 메시지 리스너를 MyListener 구현의 인스턴스로 설정합니다. 기본 스레드는 1초간 대기합니다.

```
// 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);
```

나머지 단계는 Amazon SQS Java 메시징 라이브러리 시작 예제의 단계와 동일합니다. 비동기식 소비자의 예제 전체는 AsyncMessageReceiver.java의 Amazon SQS 표준 대기열에서 JMS를 사용하기 위한 작업 자바 예제를 참조하십시오.

이 예제의 출력은 다음과 비슷합니다.

```
JMS Message ID:8example-588b-44e5-bbcf-d816example2
Received: Hello World!
```

클라이언트 승인 모드 사용

Amazon SQS Java 메시징 라이브러리 시작의 예제는 모든 수신 메시지가 자동으로 승인되고 나서 기본 Amazon SQS 대기열에서 삭제되는 AUTO_ACKNOWLEDGE 모드를 사용합니다.

1. 메시지를 처리한 후 명시적으로 승인하려면, CLIENT_ACKNOWLEDGE 모드로 세션을 생성해야 합니다.

```
// Create the non-transacted session with CLIENT_ACKNOWLEDGE mode.
Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
```

2. 메시지를 수신하면 메시지를 표시한 후 명시적으로 승인합니다.

```
// 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**
>
> 이 모드에서 메시지를 승인하면, 이 메시지를 이전에 받은 모든 메시지도 명시적으로 승인됩니다. 예를 들어, 메시지 10개를 수신하고 10번째(메시지 수신 순서) 메시지만을 승인하면, 이전의 9개 메시지 전체도 승인됩니다.

나머지 단계는 Amazon SQS Java 메시징 라이브러리 시작 예제의 단계와 동일합니다. 클라이언트 승인 모드를 사용하는 비동기식 소비자의 예제 전체는 SyncMessageReceiverClientAcknowledge.java의 Amazon SQS 표준 대기열에서 JMS를 사용하기 위한 작업 자바 예제를 참조하십시오.

이 예제의 출력은 다음과 비슷합니다.

```
JMS Message ID:4example-aa0e-403f-b6df-5e02example5
Received: Hello World!
Acknowledged: ID:4example-aa0e-403f-b6df-5e02example5
```

정렬되어 있지 않은 승인 모드 사용

CLIENT_ACKNOWLEDGE 모드를 사용하면, 명시적으로 승인한 메시지 이전에 받은 모든 메시지가 자동으로 승인됩니다. 자세한 정보는 클라이언트 승인 모드 사용을 참조하세요.

Amazon SQS Java 메시징 라이브러리는 또 다른 승인 모드를 제공합니다. UNORDERED_ACKNOWLEDGE 모드를 사용하면, 수신한 모든 메시지가 수신 순서에 상관 없이 개별적 및 명시적으로 승인됩니다. 이렇게 하려면 UNORDERED_ACKNOWLEDGE 모드를 사용하여 세션을 생성합니다.

```
// Create the non-transacted session with UNORDERED_ACKNOWLEDGE mode.
Session session = connection.createSession(false, SQSSession.UNORDERED_ACKNOWLEDGE);
```

나머지 단계는 클라이언트 승인 모드 사용 예제의 단계와 동일합니다. UNORDERED_ACKNOWLEDGE 모드에서 동기식 소비자의 예제 전체는 SyncMessageReceiverUnorderedAcknowledge.java를 참조하십시오.

이 예제의 출력은 다음과 비슷합니다.

```
JMS Message ID:dexample-73ad-4adb-bc6c-4357example7
Received: Hello World!
Acknowledged: ID:dexample-73ad-4adb-bc6c-4357example7
```

Java 메시지 서비스를 다른 Amazon SQS 클라이언트와 함께 사용

Amazon SQS Java 메시지 서비스 (JMS) 클라이언트를 AWS SDK와 함께 사용하면 Amazon SQS 메시지 크기가 256KB로 제한됩니다. 그러나 Amazon SQS 클라이언트를 사용하여 JMS 공급자를 생성할 수 있습니다. 예를 들어, JMS 클라이언트와 함께 Java용 Amazon SQS 확장 클라이언트 라이브러리를 사용하여 Amazon S3의 메시지 페이로드(최대 2GB)에 대한 참조가 들어 있는 Amazon SQS 메시지를 전송할 수 있습니다. 자세한 정보는 자바와 아마존 S3를 사용하여 대용량 Amazon SQS 메시지 관리을 참조하세요.

다음의 Java 코드 예제에서는 확장 클라이언트 라이브러리에 대한 JMS 공급자를 생성합니다.

```
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);
```

다음의 Java 코드 예제에서는 연결 팩토리를 생성합니다.

```
// 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();
```

Amazon SQS 표준 대기열에서 JMS를 사용하기 위한 작업 자바 예제

다음 코드 예시에서는 Amazon SQS 표준 대기열과 함께 Java 메시지 서비스(JMS)를 사용하는 방법을 보여줍니다. FIFO 대기열 작업에 대한 자세한 내용은 FIFO 대기열을 생성하려면, 메시지 동기식 전송 및 메시지 동기식 수신 섹션을 참조하세요. (메시지를 동기적으로 수신하는 것은 표준 대기열과 FIFO 대기열에서 동일합니다. 그러나 FIFO 대기열의 메시지에는 더 많은 속성이 포함되어 있습니다.)

ExampleConfiguration.java

다음 Java SDK v 1.x 코드 예제는 다른 Java 예제와 함께 사용할 기본 대기열 이름, 지역 및 자격 증명을 설정합니다.

```
/*
 * 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

다음 Java 코드 예제는 문자 메시지 생성자를 생성합니다.

```
/*
 * 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

다음 Java 코드 예제는 동기식 메시지 소비자를 생성합니다.

```
/*
 * 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

다음 Java 코드 예제는 비동기식 메시지 소비자를 생성합니다.

```
/*
 * 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

다음 Java 코드 예제는 클라이언트 승인 모드를 사용하는 비동기식 소비자를 생성합니다.

```
/*
 * 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

다음 Java 코드 예제는 정렬되어 있지 않은 승인 모드를 사용하는 비동기식 소비자를 생성합니다.

```
/*
 * 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

다음 XML 코드 예제는 SpringExample.java의 빈(bean) 구성 파일입니다.

```
<!--
    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

다음 Java 코드 예제는 객체를 초기화하는 빈(bean) 구성 파일을 사용합니다.

```
/*
 * 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

다음 Java 코드 예제는 Amazon SQS 대기열이 존재하는지 확인하고 나서 존재하지 않으면 대기열 하나를 생성합니다. 또한 예제 로깅 코드를 포함합니다.

```
/*
 * 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() );
        }
    }
}
```

아마존 SQS는 JMS 1.1 구현을 지원했습니다.

Amazon SQS 자바 메시징 라이브러리는 다음과 같은 JMS 1.1 구현을 지원합니다. Amazon SQS Java 메시징 라이브러리의 지원 기능 및 역량에 대한 자세한 내용은 Amazon SQS FAQ를 참조하세요.

지원되는 공통 인터페이스

- `Connection`
- `ConnectionFactory`
- `Destination`
- `Session`
- `MessageConsumer`
- `MessageProducer`

지원되는 메시지 유형

- `ByteMessage`
- `ObjectMessage`
- `TextMessage`

지원되는 메시지 승인 모드

- `AUTO_ACKNOWLEDGE`
- `CLIENT_ACKNOWLEDGE`
- `DUPS_OK_ACKNOWLEDGE`
- `UNORDERED_ACKNOWLEDGE`

> **Note**
>
> `UNORDERED_ACKNOWLEDGE` 모드는 JMS 1.1 사양의 일부가 아닙니다. 이 모드를 통해 Amazon SQS에서 JMS 클라이언트가 메시지를 명시적으로 승인하도록 허용합니다.

JMS 정의된 헤더 및 예약된 속성

메시지 전송의 경우

메시지를 전송하는 경우 사용자는 각 메시지에 대해 다음과 같은 헤더 및 속성을 설정할 수 있습니다.

- `JMSXGroupID`(FIFO 대기열에서 필수, 표준 대기열에서는 허용되지 않음)
- `JMS_SQS_DeduplicationId`(FIFO 대기열에서 선택 사항, 표준 대기열에서는 허용되지 않음)

사용자가 메시지를 전송한 후 Amazon SQS는 각 메시지에 대해 다음과 같은 헤더 및 속성을 설정합니다.

- `JMSMessageID`
- `JMS_SQS_SequenceNumber`(FIFO 대기열에만 해당)

메시지 수신의 경우

사용자가 메시지를 수신한 후 Amazon SQS는 각 메시지에 대해 다음과 같은 헤더 및 속성을 설정합니다.

- `JMSDestination`
- `JMSMessageID`
- `JMSRedelivered`
- `JMSXDeliveryCount`
- `JMSXGroupID`(FIFO 대기열에만 해당)
- `JMS_SQS_DeduplicationId`(FIFO 대기열에만 해당)
- `JMS_SQS_SequenceNumber`(FIFO 대기열에만 해당)

Amazon SQS 자습서

이 섹션에서는 Amazon SQS 기능에 대해 살펴보기 위해 사용할 수 있는 자습서를 제공합니다.

주제

- 를 사용하여 Amazon SQS 대기열 생성 AWS CloudFormation
- 자습서: Amazon Virtual Private Cloud에서 Amazon SQS 대기열로 메시지 보내기

를 사용하여 Amazon SQS 대기열 생성 AWS CloudFormation

AWS CloudFormation 콘솔과 JSON (또는 YAML) 템플릿을 사용하여 Amazon SQS 대기열을 생성할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 템플릿으로 작업하기 및 `AWS::SQS::Queue` 리소스를 참조하세요.

Amazon SQS 대기열을 생성하는 AWS CloudFormation 데 사용합니다.

1. 다음 JSON 코드를 `MyQueue.json`이라는 파일에 복사합니다. 표준 대기열을 생성하려면 `FifoQueue` 및 `ContentBasedDeduplication` 속성을 생략합니다. 콘텐츠 기반 중복 제거에 대한 자세한 내용은 Amazon SQS에서 정확히 한 번의 프로세싱 단원을 참조하십시오.

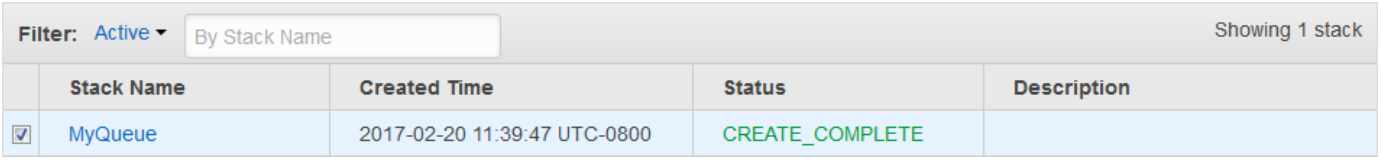
   > **Note**
   >
   > FIFO 대기열의 이름은 `.fifo` 접미사로 끝나야 합니다.

```
{
   "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. AWS CloudFormation 콘솔에 로그인한 후 스택 생성을 선택합니다.
3. 템플릿 지정 패널에서 템플릿 파일 업로드를 선택하고, `MyQueue.json` 파일을 선택한 후 다음을 선택합니다.
4. 세부 정보 지정 페이지에서 스택 이름에 `MyQueue`를 입력한 후 다음을 선택합니다.
5. 옵션 페이지에서 다음을 선택합니다.
6. 검토(Review) 페이지에서 생성(Create)을 선택합니다.

AWS CloudFormation `MyQueue`스택 생성을 시작하고 CREATE_IN_PROGRESS 상태를 표시합니다. 이 과정이 완료되면 AWS CloudFormation 에는 CREATE_COMPLETE 상태가 표시됩니다.



7. (선택 사항) 대기열의 이름, URL 및 ARN을 표시하려면, 스택의 이름을 선택하고 나서 다음 페이지에서 [Outputs] 섹션을 확장합니다.

자습서: Amazon Virtual Private Cloud에서 Amazon SQS 대기열로 메시지 보내기

이 자습서에서는 메시지를 안전한 프라이빗 네트워크를 통해 Amazon SQS 대기열로 보내는 방법을 알아봅니다. 이 네트워크는 Amazon EC2 인스턴스를 포함하는 VPC로 구성됩니다. 인스턴스는 인터페이스 VPC 엔드포인트를 통해 Amazon SQS에 연결되므로 네트워크가 퍼블릭 인터넷에서 연결이 끊긴 경우에도 Amazon EC2 인스턴스에 연결하고 Amazon SQS 대기열에 메시지를 보낼 수 있습니다. 자세한 정보는 Amazon SQS용 Amazon Virtual Private Cloud 엔드포인트을 참조하세요.

> **Important**
>
> - Amazon Virtual Private Cloud는 HTTPS Amazon SQS 엔드포인트에서만 사용할 수 있습니다.
> - Amazon VPC에서 메시지를 보내도록 Amazon SQS를 구성할 때는 프라이빗 DNS를 활성화하고 엔드포인트를 `sqs.us-east-2.amazonaws.com` 형식으로 지정해야 합니다.
> - 프라이빗 DNS는 `queue.amazonaws.com` 또는 `us-east-2.queue.amazonaws.com` 같은 레거시 엔드포인트를 지원하지 않습니다.

주제

- 1단계: Amazon EC2 키 페어 생성
- 2단계: 리소스 생성 AWS
- 3단계: EC2 인스턴스에 공개적으로 액세스할 수 없는지 확인
- 4단계: Amazon SQS용 Amazon VPC 엔드포인트 생성
- 5단계: Amazon SQS 대기열에 메시지 전송

1단계: Amazon EC2 키 페어 생성

키 페어는 Amazon EC2 인스턴스에 연결할 수 있게 해줍니다. 키 페어는 로그인 정보를 암호화하는 퍼블릭 키와 로그인 정보 암호를 해독하는 프라이빗 키로 구성됩니다.

1. Amazon EC2 콘솔에 로그인합니다.
2. 탐색 창의 네트워크 및 보안에서 키 페어를 선택합니다.
3. 키 페어 생성(Create Key Pair)을 선택합니다.
4. 키 페어 생성 대화 상자에서 키 페어 이름에`SQS-VPCE-Tutorial-Key-Pair`를 입력한 후 생성을 선택합니다.
5. 브라우저가 프라이빗 키 파일 `SQS-VPCE-Tutorial-Key-Pair.pem`을 자동으로 다운로드합니다.

   > **Important**
   >
   > 이 파일을 안전한 위치에 저장합니다. EC2는 동일한 키 페어에 대해 `.pem` 파일을 두 번 생성하지 않습니다.

6. SSH 클라이언트가 EC2 인스턴스에 연결하도록 허용하려면 최종 사용자만 읽기 권한을 가질 수 있도록 프라이빗 키 파일에 대한 권한을 설정합니다. 예:

   ```
   chmod 400 SQS-VPCE-Tutorial-Key-Pair.pem
   ```

2단계: 리소스 생성 AWS

필요한 인프라를 설정하려면 Amazon EC2 인스턴스 및 Amazon SQS 대기열과 같은 AWS 리소스로 구성된 스택을 생성하기 위한 청사진인 AWS CloudFormation 템플릿을 사용해야 합니다.

이 자습서에서 사용할 스택으로 다음과 같은 리소스가 있습니다.

- VPC 및 이에 연결된 네트워킹 리소스(예: 서브넷, 보안 그룹, 인터넷 게이트웨이, 라우팅 테이블)
- VPC 서브넷으로 시작된 Amazon EC2 인스턴스
- Amazon SQS 대기열

1. 에서 이름이 지정된 템플릿을 다운로드하십시오. AWS CloudFormation `SQS-VPCE-Tutorial-CloudFormation.yaml`GitHub
2. AWS CloudFormation 콘솔에 로그인합니다.
3. 스택 생성을 선택합니다.
4. 템플릿 선택 페이지에서 Amazon S3에 템플릿 업로드를 선택하고 `SQS-VPCE-SQS-Tutorial-CloudFormation.yaml` 파일을 선택한 후 다음을 선택합니다.

5. 세부 정보 지정 페이지에서 다음 작업을 수행합니다.

   a. 스택 이름에 SQS-VPCE-Tutorial-Stack을 입력합니다.

   b. 또는 KeyNameSQS-VPC-튜토리얼-키 페어를 선택하십시오.

   c. 다음을 선택합니다.

6. 옵션 페이지에서 다음을 선택합니다.

7. 검토 페이지의 기능 섹션에서 사용자 지정 이름으로 IAM 리소스를 생성할 수 있다고 인정함을 선택합니다. AWS CloudFormation 그런 다음 [Create] 를 선택합니다.

AWS CloudFormation 스택 생성을 시작하고 CREATE_IN_PROGRESS 상태를 표시합니다. 이 과정이 완료되면 AWS CloudFormation 에는 CREATE_COMPLETE 상태가 표시됩니다.

3단계: EC2 인스턴스에 공개적으로 액세스할 수 없는지 확인

AWS CloudFormation 템플릿은 VPC에 이름이 지정된 SQS-VPCE-Tutorial-EC2-Instance EC2 인스턴스를 시작합니다. 이 EC2 인스턴스는 아웃바운드 트래픽을 허용하지 않으며 Amazon SQS로 메시지를 전송할 수 없습니다. 이를 확인하려면, 인스턴스에 연결하고 퍼블릭 엔드포인트에 연결을 시도한 후 Amazon SQS에 메시지를 게시해 보아야 합니다.

1. Amazon EC2 콘솔에 로그인합니다.

2. 탐색 메뉴의 인스턴스에서 인스턴스를 선택합니다.

3. SQS-VPCE-를 선택합니다. Tutorial-EC2Instance

4. 퍼블릭 DNS(IPv4)에서 호스트 이름을 복사합니다(예: ec2-203-0-113-0.us-west-2.compute.amazonaws.com).

5. 앞서 생성한 키 페어를 포함하는 디렉터리에서, 다음과 같은 명령을 사용하여 인스턴스에 연결합니다.

   ```
   ssh -i SQS-VPCE-Tutorial-Key-Pair.pem ec2-user@ec2-203-0-113-0.us-east-2.compute.amazonaws.com
   ```

6. 임의의 퍼블릭 엔드포인트에 연결해 봅니다. 예:

   ```
   ping amazon.com
   ```

   예상대로 연결 시도가 실패합니다.

7. Amazon SQS 콘솔에 로그인합니다.

8. 대기열 목록에서 AWS CloudFormation 템플릿으로 생성한 대기열을 선택합니다 (예: VPCE-SQS-Tutorial-stack-CFQueue-1ABCDEFGH2ijk).
9. 세부 정보 테이블에서 URL을 복사합니다(예: https://sqs.us-east-2.amazonaws.com/123456789012/).
10. EC2 인스턴스에서 다음과 같은 명령을 사용하여 대기열에 메시지를 게시해 봅니다.

```
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."
```

예상대로 전송 시도가 실패합니다.

> **Important**
>
> 나중에 Amazon SQS용 VPC 엔드포인트를 생성하면 전송 시도가 성공할 것입니다.

4단계: Amazon SQS용 Amazon VPC 엔드포인트 생성

VPC를 Amazon SQS에 연결하려면 인터페이스 VPC 엔드포인트를 정의해야 합니다. 엔드포인트를 추가한 후에는 VPC의 EC2 인스턴스에서 Amazon SQS API를 사용할 수 있습니다. 이렇게 하면 공용 인터넷을 거치지 않고도 AWS 네트워크 내 대기열로 메시지를 보낼 수 있습니다.

> **Note**
>
> EC2 인스턴스는 여전히 인터넷상의 다른 AWS 서비스 및 엔드포인트에 액세스할 수 없습니다.

1. Amazon VPC 콘솔에 로그인합니다.
2. 탐색 메뉴에서 엔드포인트를 선택합니다.
3. 엔드포인트 생성을 선택합니다.
4. 엔드포인트 생성 페이지의 서비스 이름에서 Amazon SQS의 서비스 이름을 선택합니다.

> **Note**
>
> 서비스 이름은 현재 AWS 지역에 따라 달라집니다. 예를 들어, 미국 동부(오하이오)에서는 서비스 이름이 com.amazonaws.***us-east-2***.sqs입니다.

5. VPC에서 SQS-VPCE-Tutorial-VPC를 선택합니다.
6. 서브넷에서 서브넷 ID에 SQS-VPCE-Tutorial-Subnet이 포함된 서브넷을 선택합니다.
7. 보안 그룹에서 Select security groups(보안 그룹 선택)를 선택한 다음 그룹 이름에 SQS VPCE Tutorial Security Group이 포함된 보안 그룹을 선택합니다.
8. Create endpoint(엔드포인트 생성)을 선택합니다.

   인터페이스 VPC 엔드포인트가 생성되고 ID가 표시됩니다. 예: vpce-0ab1cdef2ghi3j456k.
9. 닫기를 선택하세요.

   Amazon VPC 콘솔에서 엔드포인트 페이지가 열립니다.

Amazon VPC가 엔드포인트를 생성하기 시작하고 대기 중 상태를 표시합니다. 이 과정이 완료되면 Amazon VPC에는 사용 가능 상태가 표시됩니다.

5단계: Amazon SQS 대기열에 메시지 전송

이제 VPC에 Amazon SQS용 엔드포인트가 포함되므로 EC2 인스턴스에 연결하여 대기열로 메시지를 보낼 수 있습니다.

1. EC2 인스턴스에 다시 연결합니다. 예:

   ```
   ssh -i SQS-VPCE-Tutorial-Key-Pair.pem ec2-user@ec2-203-0-113-0.us-
   east-2.compute.amazonaws.com
   ```
2. 다음과 같은 명령을 사용하여 다시 대기열에 메시지를 게시해 봅니다.

   ```
   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."
   ```

   전송 시도가 성공하고 메시지 본문의 MD5 다이제스트와 메시지 ID가 표시됩니다. 예:

   ```
   {
   ```

```
 "MD5OfMessageBody": "a1bcd2ef3g45hi678j90klmn12p34qr5",
 "MessageId": "12345a67-8901-2345-bc67-d890123e45fg"
}
```

AWS CloudFormation 템플릿으로 생성된 대기열에서 메시지를 수신하고 삭제하는 방법에 대한 자세한 내용은 (예: VPCE-SQS-Tutorial-Stack-CFQueue-1ABCDFGH2ijk) 을 참조하십시오. Amazon SQS에서 메시지 수신 및 삭제

리소스 삭제에 대한 자세한 내용은 다음을 참조하십시오.

- Amazon VPC 사용 설명서의 VPC 엔드포인트 삭제
- Amazon SQS 대기열 삭제
- Amazon EC2 사용 설명서에서 인스턴스 종료
- Amazon VPC 사용 설명서의 VPC 삭제
- AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 삭제
- Amazon EC2 사용 설명서에서 키 페어 삭제

Amazon SQS에서의 문제 해결

다음 항목에서는 Amazon SQS 콘솔, Amazon SQS API 또는 Amazon SQS와 함께 기타 도구를 사용할 때 발생할 수 있는 일반적인 오류 및 문제에 대한 문제 해결 조언을 제공합니다. 여기에 나열되지 않은 문제를 발견하는 경우 이 페이지의 [Feedback] 버튼을 사용하여 해당 문제를 보고할 수 있습니다.

문제 해결 조언과 일반적인 지원 질문에 대한 답변은 AWS 지식 센터를 참조하세요.

주제

Amazon SQS의 액세스 거부 오류 문제 해결

다음 항목에서는 Amazon SQS API 호출의 가장 일반적인 원인 `AccessDenied` 또는 `AccessDeniedException` 오류에 대해 다룹니다. 이러한 오류를 해결하는 방법에 대한 자세한 내용은 Amazon SQS API 호출에서 발생하는 "" 또는 AccessDenied "AccessDenied예외" 오류를 해결하려면 어떻게 해야 합니까? 를 참조하십시오. AWS 지식 센터 안내서에서

오류 메시지 예제:

```
An error occurred (AccessDenied) when calling the SendMessage operation: Access to
        the resource https://sqs.us-east-1.amazonaws.com/ is denied.
```

- 또는 -

```
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.
```

주제

Amazon SQS 대기열 정책 및 IAM 정책

요청자가 Amazon SQS 작업을 수행할 수 있는 적절한 권한을 가지고 있는지 확인하려면 다음을 수행하십시오.

- Amazon SQS API를 호출하는 IAM 보안 주체를 식별하십시오. IAM 보안 주체가 동일한 계정의 사용자인 경우 Amazon SQS 대기열 정책 또는 AWS IAM (ID 및 액세스 관리) 정책에 작업에 대한 액세스를 명시적으로 허용하는 권한이 포함되어야 합니다.
- 보안 주체가 IAM 개체인 경우:
  - 오른쪽 상단을 확인하거나 명령을 사용하여 IAM 사용자 또는 역할을 식별할 수 있습니다. AWS Management Console`aws sts get-caller-identity`
  - 요청을 수행하는 IAM 사용자 또는 역할과 관련된 IAM 정책을 확인합니다. 다음 방법 중 하나를 사용할 수 있습니다.
    - IAM 정책 시뮬레이터로 IAM 정책을 테스트하십시오.
    - 다양한 IAM 정책 유형을 검토합니다.
  - 필요한 경우 IAM 사용자 정책을 편집합니다.
  - 대기열 정책을 확인하고 필요한 경우 편집하십시오.
- 보안 주체가 AWS 서비스인 경우 Amazon SQS 대기열 정책에서 액세스를 명시적으로 허용해야 합니다.
- 보안 주체가 교차 계정 보안 주체인 경우 Amazon SQS 대기열 정책과 IAM 정책 모두 액세스를 명시적으로 허용해야 합니다.
- 정책에서 조건 요소를 사용하는 경우 조건이 액세스를 제한하는지 확인하십시오.

> **Important**
>
> 두 정책 중 하나의 명시적 거부는 명시적 허용보다 우선합니다. 다음은 Amazon SQS 정책의 몇 가지 기본 예시입니다.

AWS Key Management Service 권한

Amazon SQS 대기열에 고객이 관리하는 AWS KMS key서버 측 암호화 (SSE) 가 켜져 있는 경우 생산자와 소비자 모두에게 권한을 부여해야 합니다. 대기열이 암호화되었는지 확인하려면 대기열 콘솔의 암호화에서 **GetQueueAttributes**API **KmsMasterKeyId** 속성을 사용하거나 대기열 콘솔에서 사용할 수 있습니다.

- 프로듀서에게 필요한 권한:

```
{
"Effect": "Allow",
"Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
],
"Resource": "<Key ARN>"
}
```

- 소비자에게 필요한 권한:

```
{
"Effect": "Allow",
"Action": [
    "kms:Decrypt"
],
"Resource": "<Key ARN>"
}
```

- 계정 간 액세스에 필요한 권한:

```
{
"Effect": "Allow",
"Action": [
    "kms:DescribeKey",
    "kms:Decrypt",
    "kms:ReEncrypt",
    "kms:GenerateDataKey"
],
"Resource": "<Key ARN>"
}
```

다음 중 하나를 사용하여 Amazon SQS 대기열에 대한 암호화를 활성화할 수 있습니다.

- SSE-Amazon SQS (Amazon SQS 서비스에서 생성 및 관리하는 암호화 키)
- AWS 관리형 기본 키 (alias/aws/sqs)
- 고객 관리형 키

하지만 AWS-managed KMS 키를 사용하는 경우 기본 키 정책을 수정할 수 없습니다. 따라서 다른 서비스 및 교차 계정에 대한 액세스를 제공하려면 고객 관리 키를 사용하십시오. 이렇게 하면 키 정책을 편집할 수 있습니다.

VPC 엔드포인트 정책

Amazon 가상 사설 클라우드 (Amazon VPC) 엔드포인트를 통해 Amazon SQS에 액세스하는 경우, Amazon SQS VPC 엔드포인트 정책에서 액세스를 허용해야 합니다. Amazon SQS용 Amazon VPC 엔드포인트에 대한 정책을 생성하여 다음을 지정할 수 있습니다.

1. 작업을 수행할 수 있는 보안 주체.
2. 수행할 수 있는 작업.
3. 작업을 수행할 수 있는 리소스.

다음 예제에서 VPC 엔드포인트 정책은 IAM 사용자가 Amazon SQS *MyUser*대기열에 메시지를 보낼 수 있도록 지정합니다. *MyQueue* 다른 작업, IAM 사용자 및 Amazon SQS 리소스는 VPC 엔드포인트를 통한 액세스가 거부됩니다.

```
{
   "Statement": [{
      "Action": ["sqs:SendMessage"],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Principal": {
        "AWS": "arn:aws:iam:123456789012:user/MyUser"
      }
   }]
}
```

조직 서비스 제어 정책

조직에 AWS 계정 속해 있는 경우 AWS Organizations 정책에 따라 Amazon SQS 대기열에 액세스하지 못할 수 있습니다. 기본적으로 AWS Organizations 정책은 Amazon SQS에 대한 요청을 차단하지 않습니다. 하지만 Amazon SQS 대기열에 대한 액세스를 차단하도록 AWS Organizations 정책을 구성하지 않았는지 확인하십시오. AWS Organizations 정책을 확인하는 방법에 대한 지침은 사용 설명서의 모든 정책 나열을 참조하십시오.AWS Organizations

Amazon SQS API 오류 문제 해결

다음 주제에서는 Amazon SQS API 호출 시 반환되는 가장 일반적인 오류와 이러한 오류를 해결하는 방법을 다룹니다.

주제

- QueueDoesNotExist 오류
- InvalidAttributeValue 오류
- ReceiptHandle 오류

QueueDoesNotExist 오류

Amazon SQS 서비스가 Amazon SQS 작업에 대해 언급된 대기열을 찾을 수 없는 경우 이 오류가 반환됩니다.

가능한 원인 및 완화 방법:

- 잘못된 지역: Amazon SQS 클라이언트 구성을 검토하여 클라이언트에 올바른 지역을 구성했는지 확인하십시오. 클라이언트에서 지역을 구성하지 않으면 SDK가 구성 파일 또는 환경 변수에서 지역을 선택하거나 해당 지역을 AWS CLI 선택합니다. SDK가 구성 파일에서 지역을 찾지 못하는 경우 SDK는 기본적으로 지역을 us-east-1로 설정합니다.
- 대기열이 최근에 삭제되었을 수 있음: API 호출 전에 대기열을 삭제한 경우 API 호출 시 이 오류가 반환됩니다. 오류가 발생하기 전에 DeleteQueue작업이 있는지 확인하십시오. CloudTrail
- 권한 문제: 요청하는 AWS Identity and Access Management (IAM) 사용자 또는 역할에 필요한 권한이 없는 경우 다음 오류가 발생할 수 있습니다.

  ```
  The specified queue does not exist or you do not have access to it.
  ```

  권한을 확인하고 올바른 권한으로 API 호출을 실행하세요.

QueueDoesNotExist오류 문제 해결에 대한 자세한 내용은 Amazon SQS 대기열에 API를 호출할 때 QueueDoesNotExist 오류를 해결하려면 어떻게 해야 합니까? 를 참조하십시오. AWS 지식 센터 안내서에서.

InvalidAttributeValue 오류

이 오류는 Amazon SQS 대기열 리소스 정책 또는 잘못된 정책 또는 보안 주체가 있는 속성을 업데이트하면 반환됩니다.

가능한 원인 및 완화 방법:

- 잘못된 리소스 정책: 리소스 정책에 필수 필드가 모두 있는지 확인하세요. 자세한 내용은 IAM JSON 정책 요소 참조 및 IAM 정책 검증을 참조하십시오. 또한 IAM 정책 생성기를 사용하여 Amazon SQS 리소스 정책을 생성하고 테스트할 수 있습니다. 정책이 JSON 형식인지 확인하십시오.
- 잘못된 주체: Principal 요소가 리소스 정책에 존재하고 값이 유효한지 확인하십시오. Amazon SQS 리소스 정책 Principal 요소에 IAM 엔티티가 포함된 경우 정책을 사용하기 전에 해당 엔티티가 존재하는지 확인하십시오. Amazon SQS는 리소스 정책을 검증하고 IAM 엔티티를 확인합니다. IAM 개체가 존재하지 않는 경우 오류가 발생합니다. IAM 엔티티를 확인하려면 GetRole및 GetUserAPI를 사용하십시오.

InvalidAttributeValue오류를 해결하는 방법에 대한 추가 정보는 Amazon SQS 대기열에 API를 호출할 때 QueueDoesNotExist 오류를 해결하려면 어떻게 해야 합니까? 를 참조하십시오. AWS 지식 센터 안내서에서.

ReceiptHandle 오류

DeleteMessageAPI 호출 시 수신 핸들이 잘못되었거나 만료된 경우 오류 ReceiptHandleIsInvalid 또는 오류가 InvalidParameterValue 반환될 수 있습니다.

- ReceiptHandleIsInvalid 오류: 영수증 핸들이 올바르지 않으면 다음 예와 비슷한 오류가 표시됩니다.

```
An error occurred (ReceiptHandleIsInvalid) when calling the DeleteMessage operation:
 The input receipt handle <YOUR RECEIPT HANDLE> is not a valid receipt handle.
```

- InvalidParameterValue 오류: 영수증 핸들이 만료되면 다음 예와 비슷한 오류 메시지가 나타납니다.

```
An error occurred (InvalidParameterValue) when calling the DeleteMessage operation:
 Value <YOUR RECEIPT HANDLE> for parameter ReceiptHandle is invalid. Reason: The
 receipt handle has expired.
```

가능한 원인 및 해결 방법:

수신 핸들은 수신된 모든 메시지에 대해 생성되며 가시성 제한 기간 동안만 유효합니다. 가시성 제한 기간이 만료되면 소비자가 대기열에 메시지를 볼 수 있게 됩니다. 소비자로부터 메시지를 다시 받으면 새 수신 핸들을 받게 됩니다. 부정확하거나 만료된 수신 핸들 오류를 방지하려면 올바른 수신 핸들을 사용하여 Amazon SQS 대기열 표시 제한 기간 내에 메시지를 삭제하십시오.

`ReceiptHandle`오류 문제 해결 방법에 대한 추가 정보는 Amazon DeleteMessage SQS API 호출을 사용할 때 “" 및 ReceiptHandle IsInvalid "InvalidParameterValue” 오류를 해결하려면 어떻게 해야 합니까? 를 참조하십시오. AWS 지식 센터 안내서에서.

Amazon SQS 데드레터 대기열 및 DLQ 리드라이브 문제 해결

다음 항목에서는 Amazon SQS DLQ 및 DLQ 재드라이브 문제의 가장 일반적인 원인과 문제 해결 방법을 다룹니다. 자세한 내용은 Amazon SQS DLQ 재드라이브 문제를 해결하려면 어떻게 해야 합니까? 를 참조하십시오. 지식 센터 안내서에서.AWS

주제

- DLQ 문제
- DLQ-리드라이브 문제

DLQ 문제

일반적인 DLQ 문제와 해결 방법에 대해 알아보십시오.

주제

- 콘솔을 사용하여 메시지를 보면 메시지가 DLQ(Dead Letter Queue)로 이동할 수 있음
- 배달 못한 편지 대기열에 대한 NumberOfMessagesSent 및 NumberOfMessagesReceived가 일치하지 않음
- 데드레터 큐 리드라이브 생성 및 구성
- 표준 및 FIFO 큐 메시지 실패 처리

콘솔을 사용하여 메시지를 보면 메시지가 DLQ(Dead Letter Queue)로 이동할 수 있음

Amazon SQS는 콘솔에서 메시지를 보는 것을 해당 대기열의 리드라이브 정책을 위반하는 것으로 간주합니다. 따라서 해당 대기열의 재구동 정책에 지정된 횟수만큼 콘솔에서 메시지를 보는 경우 메시지는 해당 대기열의 데드레터 대기열로 이동됩니다.

이 동작을 조정하려면 다음 중 하나를 수행할 수 있습니다.

- 해당 대기열의 리드라이브 정책에 맞게 [Maximum Receives] 설정을 늘립니다.
- 콘솔에서 해당 대기열의 메시지를 보는 것을 피합니다.

배달 못한 편지 대기열에 대한 **NumberOfMessagesSent** 및 **NumberOfMessagesReceived**가 일치하지 않음

메시지를 배달 못한 편지 대기열에 수동으로 전송하는 경우, 이것은 NumberOfMessagesSent 측정치로 캡처됩니다. 그러나 처리 실패로 인해 메시지가 배달 못한 편지 대기열로 전송되는 경우에는 이 지표로 캡처되지 않습니다. 따라서 NumberOfMessagesSent 와 의 값이 다를 NumberOfMessagesReceived수 있습니다.

데드레터 큐 리드라이브 생성 및 구성

데드레터 대기열 재드라이브를 사용하려면 Amazon SQS에서 데드레터 대기열에서 메시지를 수신하고 대상 대기열로 메시지를 전송할 수 있는 적절한 권한을 설정해야 합니다. 올바른 권한이 없는 경우 데드레터 대기열 재구동 작업이 실패할 수 있습니다. 메시지 재구동 작업의 상태를 보고 문제를 해결하고 다시 시도할 수 있습니다.

표준 및 FIFO 큐 메시지 실패 처리

표준 대기열은 보존 기간이 만료될 때까지 메시지를 계속 처리합니다. 이렇게 계속 처리하면 사용되지 않은 메시지로 인해 큐가 차단될 가능성이 최소화됩니다. 소비자가 반복적으로 삭제하지 않는 메시지가 많으면 비용이 증가하고 하드웨어에 추가 부하가 발생할 수 있습니다. 비용을 절감하려면 실패한 메시지를 데드레터 대기열로 옮기십시오.

또한 표준 대기열에서는 많은 수의 기내 메시지를 보낼 수 있습니다. 대부분의 메시지를 사용할 수 없고 데드레터 대기열로 전송되지 않으면 메시지 처리 속도가 느려질 수 있습니다. 대기열의 효율성을 유지하려면 애플리케이션이 메시지 처리를 올바르게 처리하는지 확인하세요.

FIFO 대기열은 메시지 그룹의 메시지를 순서에 따라 소비함으로써 한 번만 처리되도록 합니다. 따라서 소비자가 다른 메시지 그룹에서 순서가 지정된 메시지를 계속 검색할 수는 있지만 대기열을 차단하는 메시지가 성공적으로 처리되거나 데드레터 대기열로 이동하기 전까지는 첫 번째 메시지 그룹을 사용할 수 없습니다.

또한 FIFO 대기열을 사용하면 전송 중인 메시지 수를 줄일 수 있습니다. FIFO 대기열이 메시지에 의해 차단되지 않도록 하려면 애플리케이션이 메시지 처리를 올바르게 처리하는지 확인하십시오.

자세한 내용은 Amazon SQS 메시지 할당량 및 Amazon SQS 메시지 작업 섹션을 참조하세요.

DLQ-리드라이브 문제

일반적인 DLQ-Redrive 문제 및 해결 방법에 대해 알아보십시오.

주제

AccessDenied 권한 문제

이 `AccessDenied` 오류는 AWS Identity and Access Management (IAM) 개체에 필요한 권한이 없어 DLQ 재드라이브가 실패하면 발생합니다.

오류 메시지 예시:

```
Failed to create redrive task. Error code: AccessDenied - Queue Permissions to Redrive.
```

DLQ 리드라이브 요청을 하려면 다음 API 권한이 필요합니다.

메시지 리드라이브를 시작하려면:

- 데드레터 큐 권한:
  - `sqs:StartMessageMoveTask`
  - `sqs:ReceiveMessage`
  - `sqs:DeleteMessage`
  - `sqs:GetQueueAttributes`
  - `kms:Decrypt`— 데드레터 큐 또는 원본 소스 큐가 암호화된 경우
- 대상 대기열 권한:
  - `sqs:SendMessage`
  - `kms:GenerateDataKey`— 대상 대기열이 암호화된 경우.
  - `kms:Decrypt` — 대상 대기열이 암호화된 경우

진행 중인 메시지 재드라이브를 취소하려면:

- 데드레터 대기열 권한:

- `sqs:CancelMessageMoveTask`
- `sqs:ReceiveMessage`
- `sqs:DeleteMessage`
- `sqs:GetQueueAttributes`
- `kms:Decrypt`— 데드레터 큐 또는 원본 소스 큐가 암호화된 경우

메시지 이동 상태를 표시하려면:

- 데드레터 대기열 권한:
  - `sqs:ListMessageMoveTasks`
  - `sqs:GetQueueAttributes`

NonExistentQueue 오류

Amazon SQS 소스 대기열이 없거나 삭제된 경우 `NonExistentQueue` 오류가 발생합니다. Amazon SQS 대기열이 있는지 확인한 다음 해당 대기열로 다시 이동하십시오.

오류 메시지 예시:

```
Failed: AWS.SimpleQueueService.NonExistentQueue
```

CouldNotDetermineMessage소스 오류

다음 시나리오에서 DLQ 재드라이브를 시작하려고 하면 `CouldNotDetermineMessageSource` 오류가 발생합니다.

- API를 사용하여 DLQ로 직접 전송된 Amazon SQS 메시지입니다. `SendMessage`
- DLQ가 구성된 아마존 심플 알림 서비스 (Amazon SNS) 주제 AWS Lambda 또는 함수의 메시지입니다.

이 오류를 해결하려면 리드라이브를 시작할 때 사용자 지정 목적지로 다시 드라이브를 선택하십시오. 그런 다음 Amazon SQS 대기열 ARN을 입력하여 DLQ의 모든 메시지를 대상 대기열로 이동합니다.

오류 메시지 예시:

```
Failed: CouldNotDetermineMessageSource
```

Amazon SQS의 FIFO 스로틀링 문제 해결

기본적으로 FIFO 대기열은, 및 에 대한 SendMessageAPI 작업당 초당 300개의 트랜잭션을 지원합니다. ReceiveMessageDeleteMessage 300TPS를 초과하는 요청은 대기열에 있는 메시지를 사용할 수 있더라도 ThrottlingException 오류가 발생합니다. 이를 완화하기 위해 다음과 같은 방법을 사용할 수 있습니다.

- Amazon SQS에서 FIFO 대기열의 높은 처리량을 활성화할 수 있습니다..
- Amazon SQS API 배치 작업을 SendMessageBatch 사용하여 API 작업당 초당 최대 3,000 개의 메시지로 TPS 한도를 늘리고 비용을 절감할 수 있습니다. DeleteMessageBatch ChangeMessageVisibilityBatch ReceiveMessageAPI의 경우 트랜잭션당 최대 10개의 메시지를 수신하도록 MaxNumberofMessages 파라미터를 설정합니다. 자세한 정보는 Amazon SQS 배치 작업을 참조하세요.
- 처리량이 높은 FIFO 대기열의 경우 권장 사항에 따라 파티션 사용률을 최적화하십시오. 메시지 그룹 ID가 동일한 메시지를 일괄적으로 전송합니다. 동일한 ReceiveMessage API 요청의 수신 핸들을 사용하여 메시지를 삭제하거나 메시지 가시성 제한 시간 값을 일괄적으로 변경하세요.
- 고유 MessageGroupId값 수를 늘리십시오. 이렇게 하면 FIFO 큐 파티션 전체에 균등하게 분배할 수 있습니다. 자세한 정보는 Amazon SQS 메시지 그룹 ID 사용을 참조하세요.

자세한 내용은 Amazon SQS FIFO 대기열이 모든 메시지 또는 다른 메시지 그룹의 메시지를 반환하지 않는 이유는 무엇입니까? 를 참조하십시오. AWS 지식 센터 가이드에서.

Amazon ReceiveMessage SQS API 호출에 대해 메시지가 반환되지 않는 문제 해결

다음 주제에서는 Amazon SQS 메시지가 소비자에게 반환되지 않는 가장 일반적인 원인과 이러한 문제를 해결하는 방법을 다룹니다. 자세한 내용은 Amazon SQS 대기열에서 메시지를 수신할 수 없는 이유는 무엇입니까? 를 참조하십시오. AWS 지식 센터 가이드에서

주제

- 빈 대기열
- 비행 한도에 도달했습니다.
- 메시지 지연
- 메시지가 전송 중입니다.

- 폴링 방법

빈 대기열

대기열이 비어 있는지 확인하려면 긴 폴링을 사용하여 `ReceiveMessage`API를 호출하십시오. `ApproximateNumberOfMessagesVisible`,`ApproximateNumberOfMessagesNotVisible`, `ApproximateNumberOfMessagesDelayed` CloudWatch 지표를 사용할 수도 있습니다. 몇 분 동안 모든 지표 값이 0으로 설정된 경우 대기열은 비어 있는 것으로 간주됩니다.

비행 한도에 도달했습니다.

긴 폴링을 사용하고 대기열의 비행 제한 (FIFO의 경우 20000, 표준은 120000) 을 위반하는 경우 Amazon SQS는 할당량 한도를 초과하는 오류 메시지를 반환하지 않습니다.

메시지 지연

Amazon SQS 대기열이 지연 대기열로 구성되어 있거나 메시지가 메시지 타이머와 함께 전송된 경우 지연 시간이 끝날 때까지 메시지가 표시되지 않습니다. 대기열이 지연 대기열로 구성되어 있는지 확인하려면 `GetQueueAttributes`API `DelaySeconds` 속성을 사용하거나 대기열 콘솔의 전송 지연에서 사용하십시오. `ApproximateNumberOfMessagesDelayed` CloudWatch 지표를 확인하여 지연된 메시지가 있는지 파악하십시오.

메시지가 전송 중입니다.

다른 소비자가 메시지를 폴링한 경우 메시지가 전송 중이거나 표시 제한 시간 동안 보이지 않게 됩니다. 추가 설문 조사에서는 빈 수신이 반환될 수 있습니다. ApproximateNumberOfMessagesVisible CloudWatch 지표를 확인하여 수신할 수 있는 메시지 수를 파악하십시오. FIFO 대기열의 경우 메시지 그룹 ID가 있는 메시지가 전송 중이면 메시지를 삭제하거나 메시지가 표시되기 전까지 더 이상 메시지가 반환되지 않습니다. FIFO 대기열의 메시지 그룹 수준에서 메시지 순서가 유지되기 때문입니다.

폴링 방법

짧은 폴링을 사용하는 경우 (WaitTime초: 0) Amazon SQS는 해당 서버의 하위 집합을 샘플링하여 해당 서버에서만 메시지를 반환합니다. 따라서 메시지를 수신할 수 있더라도 메시지를 받지 못할 수 있습니다. 후속 폴링 요청은 메시지를 반환합니다.

긴 폴링을 사용하는 경우 Amazon SQS는 사용 가능한 메시지를 하나 이상 수집한 후 모든 서버를 폴링하고 지정된 최대 수까지 응답을 보냅니다. ReceiveMessage WaitTime초 값이 너무 낮으면 사용 가능한 메시지를 모두 받지 못할 수 있습니다.

Amazon SQS 네트워크 오류 문제 해결

다음 주제에서는 Amazon SQS에서 발생하는 네트워크 문제의 가장 일반적인 원인과 문제 해결 방법을 다룹니다.

주제

ETIMEOUT error

이 ETIMEOUT 오류는 클라이언트가 Amazon SQS 엔드포인트에 대한 TCP 연결을 설정할 수 없을 때 발생합니다.

문제 해결:

- 네트워크 연결을 확인하십시오.

  와 같은 명령을 실행하여 Amazon SQS로의 네트워크 연결을 테스트합니다. `telnet`

  Example: telnet sqs.us-east-1.amazonaws.com 443
- 네트워크 설정을 확인하십시오.
  - 로컬 방화벽 규칙, 경로 및 ACL (액세스 제어 목록) 이 사용하는 포트의 트래픽을 허용하는지 확인하십시오.
  - 보안 그룹 아웃바운드 (이그레스) 규칙은 포트 80 또는 443으로의 트래픽을 허용해야 합니다.
  - 네트워크 ACL 아웃바운드 (이그레스) 규칙은 TCP 포트 80 또는 443으로의 트래픽을 허용해야 합니다.
  - 네트워크 ACL 인바운드 (인그레스) 규칙은 TCP 포트 1024-65535에서의 트래픽을 허용해야 합니다.
  - 퍼블릭 인터넷에 연결되는 Amazon Elastic Compute Cloud (Amazon EC2) 인스턴스는 인터넷에 연결되어 있어야 합니다.
- 아마존 가상 사설 클라우드 (Amazon VPC) 엔드포인트

  Amazon VPC 엔드포인트를 통해 Amazon SQS에 액세스하는 경우, 엔드포인트 보안 그룹은 포트 443에서 클라이언트 보안 그룹으로 향하는 인바운드 트래픽을 허용해야 합니다. VPC 엔드포인트의 서브넷과 연결된 네트워크 ACL에는 다음과 같은 구성이 있어야 합니다.

- 네트워크 ACL 아웃바운드 (이그레스) 규칙은 TCP 포트 1024-65535 (임시 포트) 에서의 트래픽을 허용해야 합니다.
- 네트워크 ACL 인바운드 (인그레스) 규칙은 포트 443에서의 트래픽을 허용해야 합니다.

또한 Amazon SQS VPC 엔드포인트 AWS Identity and Access Management (IAM) 정책에서 액세스를 허용해야 합니다. 다음 예제 VPC 엔드포인트 정책은 IAM 사용자가 Amazon SQS *MyUser*대기열에 메시지를 보낼 수 있도록 지정합니다. *MyQueue* 다른 작업, IAM 사용자 및 Amazon SQS 리소스는 VPC 엔드포인트를 통한 액세스가 거부됩니다.

```
{
    "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

호스트 IP 주소를 확인할 수 없을 때 UnknownHostException 오류가 발생합니다.

문제 해결:

nslookup유틸리티를 사용하여 호스트 이름과 관련된 IP 주소를 반환합니다.

- Windows and Linux OS

```
nslookup sqs.<region>.amazonaws.com
```

- AWS CLI 또는 Python 레거시 엔드포인트용 SDK:

```
nslookup <region>.queue.amazonaws.com
```

실패한 출력을 받은 경우 DNS는 어떻게 작동하며 부분적 또는 간헐적인 DNS 장애 문제를 해결하려면 어떻게 하나요? 의 지침을 따르세요. 지식 센터 가이드에서AWS

유효한 출력을 받았다면 애플리케이션 수준의 문제일 가능성이 높습니다. 응용 프로그램 수준 문제를 해결하려면 다음 방법을 시도해 보십시오.

- 애플리케이션을 다시 시작합니다.
- Java 애플리케이션에 잘못된 DNS 캐시가 없는지 확인하십시오. 가능하면 DNS TTL을 준수하도록 애플리케이션을 구성하십시오. 자세한 내용은 DNS 이름 조회를 위한 JVM TTL 설정을 참조하십시오.

네트워크 오류를 해결하는 방법에 대한 추가 정보는 Amazon SQS “ETIMEOUT” 및 UnknownHost “예외” 연결 오류 문제를 해결하려면 어떻게 해야 합니까? 를 참조하십시오. 지식 센터 안내서에서.AWS

AWS X-Ray를 사용한 Amazon Simple Queue Service 대기열 문제 해결

AWS X-Ray 애플리케이션이 처리하는 요청에 대한 데이터를 수집하고 데이터를 보고 필터링하여 잠재적 문제와 최적화 기회를 식별할 수 있도록 합니다. 애플리케이션에 대한 추적된 요청의 경우 요청, 응답 및 애플리케이션이 다운스트림 AWS 리소스, 마이크로서비스, 데이터베이스 및 HTTP 웹 API에 대해 수행하는 호출에 대한 세부 정보를 볼 수 있습니다.

Amazon SQS를 통해 AWS X-Ray 추적 헤더를 전송하려면 다음 중 하나를 수행할 수 있습니다.

- `X-Amzn-Trace-Id` 트레이스 헤더를 사용합니다.
- `AWSTraceHeader` 메시지 시스템 속성을 사용합니다.

오류 및 지연 시간에 대한 데이터를 수집하려면 AWS X-Ray SDK를 사용하여 `AmazonSQS` 클라이언트를 계측해야 합니다.

AWS X-Ray 콘솔을 사용하여 Amazon SQS와 애플리케이션이 사용하는 다른 서비스 간의 연결 맵을 볼 수 있습니다. 또한 콘솔을 사용하여 평균 대기 시간 및 실패율과 같은 메트릭을 볼 수 있습니다. 자세한 정보는 AWS X-Ray 개발자 안내서의 Amazon SQS 및 AWS X-Ray를 참조하세요.

Amazon SQS의 보안

이 섹션에서는 Amazon SQS 보안, 인증 및 액세스 제어, Amazon SQS 액세스 정책 언어에 대한 정보를 제공합니다.

주제

Amazon SQS에서의 데이터 보호

AWS 공동 책임 모델 공동 책임 모델 이 모델에 설명된 대로 AWS 은 (는) 모두를 실행하는 글로벌 인프라를 보호하는 역할을 AWS 클라우드합니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하십시오.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신하세요. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.

- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하십시오.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon SQS 또는 기타 콘솔 AWS CLI, API 또는 AWS 서비스 SDK를 사용하여 작업하는 경우가 포함됩니다. AWS 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

다음 단원에서는 Amazon SQS의 데이터 보호에 대해 설명합니다.

주제

- Amazon SQS의 데이터 암호화
- Amazon SQS의 인터네트워크 트래픽 프라이버시

Amazon SQS의 데이터 암호화

데이터 보호란 전송 중(Amazon SQS 안팎으로 데이터가 이동 중)과 유휴 시(Amazon SQS 데이터 센터의 디스크에 데이터가 저장된 동안) 데이터를 보호하는 것을 말합니다. Secure Sockets Layer(SSL) 또는 클라이언트 측 암호화를 사용하여 전송 중인 데이터를 보호할 수 있습니다. 기본적으로 Amazon SQS는 디스크 암호화를 사용하여 메시지와 파일을 저장합니다. 메시지를 데이터 센터의 암호화된 파일 시스템에 저장하기 전에 Amazon SQS에 메시지를 암호화하도록 요청하여 저장된 데이터를 보호할 수 있습니다. Amazon SQS는 최적화된 데이터 암호화를 위해 SSE를 사용할 것을 권장합니다.

주제

- Amazon SQS의 저장 중 암호화
- 아마존 SQS 키 관리

Amazon SQS의 저장 중 암호화

서버 측 암호화(SSE)를 사용하면 암호화된 대기열에서 민감한 데이터를 전송할 수 있습니다. SSE는 SQS에서 관리하는 암호화 키 (SSE-SQS) 또는 SSE-KMS) 에서 관리되는 키를 사용하여 대기열에 있는 메시지의 콘텐츠를 보호합니다. AWS Key Management Service 를 사용하여 SSE를 관리하는 방법에 대한 자세한 내용은 다음을 참조하십시오. AWS Management Console

- 대기열에 대해 SSE-SQS 구성(콘솔)

- 대기열에 대해 SSE-KMS 구성(콘솔)

AWS SDK for Java (및 `CreateQueueSetQueueAttributes`, 및 `GetQueueAttributes` 작업) 을 사용하여 SSE를 관리하는 방법에 대한 자세한 내용은 다음 예를 참조하십시오.

- Amazon SQS 대기열에서 서버 측 암호화 사용
- KMS 권한 구성 대상 AWS 서비스

Amazon SQS가 메시지를 수신하면 SSE가 메시지를 즉시 암호화합니다. 메시지는 암호화된 형식으로 저장되며 Amazon SQS는 권한 있는 소비자에게 전송된 경우에만 메시지를 해독합니다.

> **Important**
>
> SSE를 사용할 수 있는 대기열에 대한 모든 요청에서는 HTTPS 및 서명 버전 4를 사용해야 합니다.
> 기본 키 (Amazon SQS의AWS 관리형 KMS 키) 를 사용하는 암호화된 대기열은 다른 키에서 Lambda 함수를 호출할 수 없습니다. AWS 계정
> AWS Security Token Service `AssumeRole`작업을 사용하여 Amazon SQS로 알림을 보낼 수 있는 일부 AWS 서비스 기능은 SSE와 호환되지만 표준 대기열에서만 작동합니다.
>
> - Auto Scaling 수명 주기 후크
> - AWS Lambda 데드레터 대기열
>
> 기타 서비스의 암호화 대기열과의 호환성에 관한 정보는 서비스에 대한 KMS 권한을 구성합니다 AWS . 및 서비스 설명서를 참조하십시오.

AWS KMS 안전한 고가용성 하드웨어와 소프트웨어를 결합하여 클라우드에 맞게 확장된 키 관리 시스템을 제공합니다. Amazon SQS를 와 함께 AWS KMS사용하면 메시지 데이터를 암호화하는 데이터 키도 암호화되어 보호되는 데이터와 함께 저장됩니다.

AWS KMS를 사용하면 다음과 같은 이점이 있습니다.

- AWS KMS keys를 직접 생성하고 관리할 수 있습니다.
- Amazon SQS용 AWS 관리형 KMS 키를 사용할 수도 있습니다. 이 키는 계정 및 지역별로 고유합니다.

- AWS KMS 보안 표준은 암호화 관련 규정 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다.

자세한 정보는 AWS Key Management Service개발자 안내서.의 AWS Key Management Service 란 무엇입니까?를 참조하세요.

주제

- 암호화 범위
- 주요 용어

암호화 범위

SSE는 Amazon SQS 대기열에서 메시지의 본문을 암호화합니다.

SSE는 다음을 암호화하지 않습니다.

- 대기열 메타데이터(대기열 이름과 속성)
- 메시지 메타데이터(메시지 ID, 타임스탬프 및 속성)
- 대기열당 측정치

메시지를 암호화하면 권한이 없는 사용자나 익명 사용자가 해당 내용을 사용할 수 없게 됩니다. SSE를 활성화하면 암호화된 대기열에 대한 익명 `SendMessage` 및 `ReceiveMessage` 요청이 거부됩니다. Amazon SQS 보안 모범 사례에서는 익명 요청을 사용하지 말 것을 권장합니다. Amazon SQS 대기열로 익명 요청을 보내려면 SSE를 비활성화해야 합니다. 다음과 같은 Amazon SQS의 정상 작동에는 영향을 주지 않습니다.

- 대기열 암호화가 활성화된 후 전송되는 경우에만 메시지가 암호화됩니다. Amazon SQS는 백로그된 메시지를 암호화하지 않습니다.
- 암호화된 메시지는 해당 대기열 암호화가 비활성화된 경우에만 암호화된 상태를 유지합니다.

메시지를 배달 못한 편지 대기열로 이동하는 것은 다음과 같이 해당 암호화에 영향을 주지 않습니다.

- Amazon SQS가 암호화된 소스 대기열에서 암호화되지 않은 DLQ(Dead Letter Queue)로 메시지를 이동하는 경우, 메시지는 암호화된 상태를 유지합니다.
- Amazon SQS가 암호화되지 않은 소스 대기열에서 암호화된 DLQ(Dead Letter Queue)로 메시지를 이동하는 경우, 메시지는 암호화되지 않은 상태를 유지합니다.

주요 용어

다음 주요 용어 설명은 SSE 기능을 보다 정확하게 이해하는 데 도움이 될 수 있습니다. 자세한 내용은 Amazon Simple Queue Service API 참조를 참조하세요.

데이터 키

Amazon SQS 메시지 내용의 암호화를 담당하는 키(DEK)입니다.

자세한 내용은 AWS Encryption SDK 개발자 안내서의 AWS Key Management Service 개발자 안내서에서 데이터 키를 참조하세요.

데이터 키 재사용 기간

Amazon SQS가 다시 호출하기 전에 데이터 키를 재사용하여 메시지를 암호화하거나 복호화할 수 있는 시간 (초). AWS KMS 초 단위의 정수로, 60초(1분)에서 86,400초(24시간) 사이입니다. 기본값은 300(5분)입니다. 자세한 정보는 데이터 키 재사용 기간 이해을 참조하세요.

> **Note**
>
> 드물긴 AWS KMS하지만 연결할 수 없는 경우에도 Amazon SQS는 연결이 다시 설정될 때까지 캐시된 데이터 키를 계속 사용합니다.

KMS 키 ID

사용자 계정이나 다른 계정에 있는 AWS 관리형 KMS 키 또는 사용자 지정 KMS 키의 별칭, 별칭 ARN, 키 ID 또는 키 ARN. Amazon SQS의 AWS 관리형 KMS 키 별칭은 `alias/aws/sqs` 항상 같지만 사용자 지정 KMS 키의 별칭은 예를 들어 다음과 같을 수 있습니다. `alias/MyAlias` 이러한 KMS 키를 사용하여 Amazon SQS 대기열의 메시지를 보호할 수 있습니다.

> **Note**
>
> 다음 사항에 유의하십시오:
>
> - 사용자 지정 KMS 키를 지정하지 않는 경우 Amazon SQS는 Amazon SQS용 관리형 KMS 키를 사용합니다 AWS .
> - 를 사용하여 대기열에 AWS KMS 대해 Amazon SQS의 AWS 관리형 KMS 키를 처음 지정하면 Amazon SQS용 관리형 KMS 키가 생성됩니다 AWS . AWS Management Console

- 또는 SSE가 활성화된 대기열에서 `SendMessage` 또는 `SendMessageBatch` 작업을 처음 사용할 때 Amazon SQS용 AWS 관리형 KMS 키가 AWS KMS 생성됩니다.

콘솔의 고객 관리 키 섹션 또는 작업을 사용하여 KMS 키를 생성하고, KMS 키 사용 방법을 제어하는 정책을 정의하고, KMS 키 사용을 감사할 수 있습니다. AWS KMS `CreateKey` AWS KMS 자세한 내용은 AWS Key Management Service 개발자 안내서의 KMS 키 및 키 생성을 참조하세요. KMS 키 식별자의 추가 예는 API 참조를 참조하십시오 KeyId.AWS Key Management Service KMS 키 식별자 찾기에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 ID 및 ARN 찾기를 참조하세요.

> **Important**
>
> 사용 시 추가 요금이 부과됩니다. AWS KMS자세한 내용은 비용 추정 AWS KMS 및 AWS Key Management Service 요금을 참조하세요.

봉투 암호화

암호화된 데이터의 보안은 부분적으로 암호를 해독할 수 있는 데이터 키를 보호하는 데 달려 있습니다. Amazon SQS는 KMS 키를 사용하여 데이터 키를 암호화하며, 암호화된 데이터 키는 암호화된 메시지와 함께 저장됩니다. KMS 키를 사용하여 데이터 키를 암호화하는 이러한 방법을 봉투 암호화라고 합니다.

자세한 내용은 AWS Encryption SDK 개발자 안내서에서 봉투 암호화를 참조하세요.

아마존 SQS 키 관리

Amazon SQS는 (KMS) 와 통합되어 서버 측 암호화 AWS Key Management Service (SSE) 를 위한 KMS 키를 관리합니다. SSE 정보 및 키 관리 정의는 Amazon SQS의 저장 중 암호화 섹션을 참조하세요. Amazon SQS는 KMS 키를 사용하여 메시지를 암호화하고 복호화하는 데이터 키를 검증하고 보호합니다. 다음 단원에서는 Amazon SQS 서비스에서 KMS 키 및 데이터 키를 사용하는 작업에 대해 설명합니다.

주제

- AWS KMS 권한 구성
- 데이터 키 재사용 기간 이해
- 비용 추정 AWS KMS

- AWS KMS 오류

AWS KMS 권한 구성

모든 KMS 키에는 키 정책이 있어야 합니다. Amazon SQS의 AWS 관리형 KMS 키의 키 정책은 수정할 수 없다는 점에 유의하십시오. 이 KMS 키의 정책에는 해당 계정의 모든 주체(Amazon SQS를 사용할 권한이 있는 주체)가 암호화된 대기열을 사용할 수 있는 권한이 포함되어 있습니다.

고객 관리형 KMS 키의 경우 각 대기열 생산자 및 소비자에 대한 권한을 추가하도록 키 정책을 구성해야 합니다. 이렇게 하려면 KMS 키 정책에서 생산자와 소비자를 사용자로 지정하면 됩니다. AWS KMS 권한에 대한 자세한 내용은 AWS Key Management Service 개발자AWS KMS 안내서의 리소스 및 운영 또는 AWS KMS API 권한 참조를 참조하십시오.

또는 필요한 권한을 암호화된 메시지를 생산 및 소비하는 보안 주체에게 할당된 IAM 정책에 지정해도 됩니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 AWS KMS로 IAM 정책 사용을 참조하세요.

> **Note**
>
> Amazon SQS에서 보내고 받을 수 있는 글로벌 권한을 구성할 수 있지만, IAM 정책 섹션에서 특정 지역의 KMS 키 전체 ARN 이름을 명시적으로 AWS KMS 지정해야 합니다. `Resource`

서비스에 대한 KMS 권한을 구성합니다 AWS .

여러 AWS 서비스가 Amazon SQS 대기열로 이벤트를 전송할 수 있는 이벤트 소스 역할을 합니다. 이러한 이벤트 소스가 암호화된 대기열과 함께 작동하도록 하려면 고객 관리형 KMS 키를 생성하고 서비스가 필요한 API 방법을 사용할 수 있는 권한을 키 정책에 추가해야 합니다. AWS KMS 권한을 구성하려면 다음 단계를 수행합니다.

> **Warning**
>
> Amazon SQS 메시지 암호화를 위해 KMS 키를 변경할 때는 이전 KMS 키로 암호화된 기존 메시지가 해당 키로 암호화된 상태로 유지된다는 점에 유의하십시오. 이러한 메시지를 해독하려면 이전 KMS 키를 유지하고 키 정책이 Amazon SQS에 및 에 대한 권한을 부여하는지 확인해야 합니다. `kms:Decrypt` `kms:GenerateDataKey` 새 메시지 암호화를 위해 새 KMS 키로 업데이트한 후에는 이전 KMS 키를 삭제하거나 비활성화하기 전에 이전 KMS 키로 암호화된 기존 메시지를 모두 처리하고 대기열에서 제거해야 합니다.

1. 고객 관리형 KMS 키를 생성합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 키 생성을 참조하세요.
2. AWS 서비스 이벤트 소스가 kms:GenerateDataKey 및 kms:Decrypt API 메서드를 사용할 수 있도록 허용하려면 KMS 키 정책에 다음 명령문을 추가하십시오.

```
{
   "Version": "2012-10-17",
      "Statement": [{
         "Effect": "Allow",
         "Principal": {
            "Service": "service.amazonaws.com"
         },
         "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
         ],
         "Resource": "*"
       }]
}
```

위의 예에서 “service”를 이벤트 소스의 서비스 이름으로 바꿉니다. 이벤트 소스에는 다음 서비스가 포함됩니다.

이벤트 소스	서비스 이름
아마존 CloudWatch 이벤트	events.amazonaws.com
Amazon S3 이벤트 알림	s3.amazonaws.com
Amazon SNS 주제 구독	sns.amazonaws.com

3. KMS 키의 ARN을 사용하여 기존 SSE 대기열을 구성합니다.
4. 암호화된 대기열의 ARN을 이벤트 소스에 제공합니다.

생산자를 위한 AWS KMS 권한 구성

데이터 키 재사용 기간이 만료되면 다음에 생산자가 SendMessage 또는 SendMessageBatch를 호출했을 때 kms:GenerateDataKey 및 kms:Decrypt 호출도 트리거됩니다. kms:Decrypt 호출은

새 데이터 키를 사용하기 전에 이 데이터 키의 무결성을 확인하기 위한 것입니다. 따라서 생산자에게 KMS 키의 `kms:GenerateDataKey` 및 `kms:Decrypt` 권한이 있어야 합니다.

생산자의 IAM 정책에 다음 문을 추가합니다. 키 리소스와 대기열 리소스에 올바른 ARN 값을 사용해야 합니다.

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

소비자를 위한 AWS KMS 권한 구성

데이터 키 재사용 기간이 만료되면 다음에 소비자가 `ReceiveMessage`를 호출했을 때 `kms:Decrypt` 호출도 트리거되며, 이는 새 데이터 키를 사용하기 전에 이 데이터 키의 무결성을 확인하기 위한 것입니다. 따라서 지정된 대기열에서 메시지 암호화에 사용한 KMS 키의 `kms:Decrypt` 권한이 소비자에게 있어야 합니다. 이 대기열이 DLQ(Dead Letter Queue)인 경우, 소비자는 소스 대기열에서 메시지 암호화에 사용한 KMS 키에 대한 `kms:Decrypt` 권한도 가지고 있어야 합니다. 소비자의 IAM 정책에 다음 문을 추가합니다. 키 리소스와 대기열 리소스에 올바른 ARN 값을 사용해야 합니다.

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

혼동된 대리인 보호를 사용하여 AWS KMS 권한을 구성하십시오.

키 정책문의 보안 주체가 AWS 서비스 보안 주체인 경우 aws:SourceArn 또는 aws:SourceAccount 전역 조건 키를 사용하여 혼동된 대리자 문제를 방지할 수 있습니다. 이러한 조건 키를 사용하려면 값을 암호화되는 리소스의 Amazon 리소스 이름(ARN)으로 설정합니다. 리소스의 ARN을 모르면 aws:SourceAccount를 대신 사용합니다.

이 KMS 키 정책에서는 계정별로 소유한 서비스의 특정 리소스가 Amazon SQS의 SSE 사용 중에 발생하는 111122223333 및 Decrypt 작업을 위해 KMS를 호출할 수 있도록 허용합니다.

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

SSE가 활성화된 Amazon SQS 대기열을 사용하는 경우 다음 서비스에서 `aws:SourceArn`을 지원합니다.

- Amazon SNS
- Amazon S3
- CloudWatch 이벤트
- AWS Lambda
- CodeBuild
- Amazon Connect Customer Profiles
- AWS Auto Scaling
- Amazon Chime

데이터 키 재사용 기간 이해

데이터 키 재사용 기간은 Amazon SQS가 한 데이터 키를 재사용할 수 있는 최대 기간을 정의합니다. 데이터 키 재사용 기간이 종료되면 Amazon SQS에서 새 데이터 키를 생성합니다. 재사용 기간에 대한 다음 지침에 유의하십시오.

- 재사용 기간이 짧을수록 보안은 향상되지만 더 많은 호출이 발생하여 프리 AWS KMS티어를 초과하여 요금이 부과될 수 있습니다.
- 데이터 키는 암호화 및 암호 해독에 대해 별도로 캐시되지만 재사용 기간은 데이터 키의 두 복사본 모두에 적용됩니다.
- 데이터 키 재사용 기간이 끝나면 `SendMessageBatch` 일반적으로 다음 번 호출이 `SendMessage` 발생하거나 새 데이터 키를 가져오기 위한 AWS KMS `GenerateDataKey` 메서드 호출이 트리거됩니다. 또한 다음 호출에서는 데이터 키를 사용하기 전에 데이터 키의 무결성을 AWS KMS `Decrypt` 확인하기 위한 호출을 각각 트리거합니다. `SendMessage` `ReceiveMessage`
- 보안 주체 (AWS 계정 또는 사용자) 는 데이터 키를 공유하지 않습니다. 즉, 고유한 보안 주체가 보낸 메시지는 항상 고유한 데이터 키를 받습니다. 따라서 호출량은 데이터 키 재사용 AWS KMS 기간 동안 사용 중인 고유 보안 주체 수의 배수입니다.

비용 추정 AWS KMS

비용을 예측하고 AWS 청구서를 더 잘 이해하려면 Amazon SQS가 KMS 키를 얼마나 자주 사용하는지 알고 싶을 수 있습니다.

> **Note**
>
> 다음 공식으로 예상되는 비용을 거의 정확하게 짐작할 수 있지만, Amazon SQS의 분산 특성상 실제 비용은 더 높을 수 있습니다.

대기열당 API 요청 수(R)를 계산하려면 다음 수식을 사용하세요.

```
R = (B / D) * (2 * P + C)
```

여기서 B는 청구 기간(초)입니다.

D는 데이터 키 재사용 기간(초)입니다.

P는 Amazon SQS 대기열로 전송하는 생성 보안 주체의 수입니다.

C는 Amazon SQS 대기열로부터 받는 소비 보안 주체의 수입니다.

> **Important**
>
> 일반적으로, 생산 보안 주체에서 발생하는 비용이 소비 보안 주체의 두 배입니다. 자세한 정보는 데이터 키 재사용 기간 이해을 참조하세요.
> 생산자와 소비자의 사용자가 서로 다른 경우, 비용이 증가합니다.

다음은 계산 예제입니다. 정확한 요금 정보는 AWS Key Management Service 요금을 참조하세요.

예 1: 보안 주체 2명과 대기열 1개에 대한 AWS KMS API 호출 수 계산

이 예에서는 다음과 같이 가정합니다.

- 청구 기간은 1월 1일부터 31일까지입니다(2,678,400초).
- 데이터 키 재사용 기간은 5분(300초)로 설정되어 있습니다.
- 대기열은 1개입니다.
- 생산 보안 주체 1개와 소비 보안 주체 1개가 있습니다.

```
(2,678,400 / 300) * (2 * 1 + 1) = 26,784
```

예 2: 여러 생산자와 소비자, 대기열 2개에 대한 AWS KMS API 호출 수 계산

이 예에서는 다음과 같이 가정합니다.

- 청구 기간은 2월 1일부터 28일까지입니다(2,419,200초).
- 데이터 키 재사용 기간은 24시간(86,400초)으로 설정되어 있습니다.
- 대기열은 2개입니다.
- 첫 번째 대기열에 생산 보안 주체 3개와 소비 보안 주체 1개가 있습니다.
- 두 번째 대기열에 생산 보안 주체 5개와 소비 보안 주체 2개가 있습니다.

```
(2,419,200 / 86,400 * (2 * 3 + 1)) + (2,419,200 / 86,400 * (2 * 5 + 2)) = 532
```

AWS KMS 오류

Amazon SQS로 작업할 때 오류가 발생할 수 있습니다. AWS KMS아래 참조에서 이러한 오류와 문제 해결 방법을 설명합니다.

- 일반적인 AWS KMS 오류
- AWS KMS 암호화 해제 오류
- AWS KMS GenerateDataKey 오류

Amazon SQS의 인터네트워크 트래픽 프라이버시

Amazon SQS용 Amazon Virtual Private Cloud(VPC) 엔드포인트는 VPC 내의 논리적 엔터티로서, Amazon SQS에만 연결을 허용합니다. VPC는 Amazon SQS로 요청을 라우팅하고, 응답을 다시 VPC로 라우팅합니다. 다음 섹션에서는 VPC 엔드포인트 작업 및 VPC 엔드포인트 정책 생성에 대해 설명합니다.

주제

- Amazon SQS용 Amazon Virtual Private Cloud 엔드포인트
- Amazon SQS용 Amazon VPC 엔드포인트 정책 생성

Amazon SQS용 Amazon Virtual Private Cloud 엔드포인트

Amazon VPC를 사용하여 AWS 리소스를 호스팅하는 경우 VPC와 Amazon SQS를 연결할 수 있습니다. 이 연결을 사용하여 퍼블릭 인터넷을 통하지 않고 Amazon SQS 대기열에 메시지를 보낼 수 있습니다.

Amazon VPC를 사용하면 사용자 지정 가상 네트워크에서 AWS 리소스를 시작할 수 있습니다. VPC를 사용하여 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 등의 네트워크 설정을 제어할 수 있습니다. VPC에 대한 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

VPC를 Amazon SQS에 연결하려면 먼저 VPC를 다른 AWS 서비스에 연결할 수 있는 인터페이스 VPC 엔드포인트를 정의해야 합니다. 이 엔드포인트를 이용하면 인터넷 게이트웨이나 Network Address Translation(NAT) 인스턴스 또는 VPN 연결 없이도 Amazon SQS에 안정적이고 확장 가능하게 연결됩니다. 자세한 내용은 이 안내서의 자습서: Amazon Virtual Private Cloud에서 Amazon SQS 대기열로 메시지 보내기 및 예제 5: VPC 엔드포인트에서 온 액세스가 아닌 경우 액세스 거부와 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.

> **Important**
>
> - Amazon Virtual Private Cloud는 HTTPS Amazon SQS 엔드포인트에서만 사용할 수 있습니다.
> - Amazon VPC에서 메시지를 보내도록 Amazon SQS를 구성할 때는 프라이빗 DNS를 활성화하고 엔드포인트를 `sqs.us-east-2.amazonaws.com` 형식으로 지정해야 합니다.
> - 프라이빗 DNS는 `queue.amazonaws.com` 또는 `us-east-2.queue.amazonaws.com` 같은 레거시 엔드포인트를 지원하지 않습니다.

Amazon SQS용 Amazon VPC 엔드포인트 정책 생성

Amazon SQS에 대한 Amazon VPC 엔드포인트 정책을 생성하여 다음을 지정할 수 있습니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

다음 예제 VPC 엔드포인트 정책은 MyUser 사용자가 Amazon SQS 대기열 MyQueue에 메시지를 게시할 수 있도록 지정합니다.

```
{
   "Statement": [{
      "Action": ["sqs:SendMessage"],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Principal": {
        "AWS": "arn:aws:iam:123456789012:user/MyUser"
      }
   }]
}
```

다음 작업은 거부됩니다.

- sqs:CreateQueue 및 sqs:DeleteQueue와 같은 다른 Amazon SQS API 작업.
- 다른 사용자 및 규칙이 이 VPC 엔드포인트를 사용.
- MyUser가 다른 Amazon SQS 대기열에 메시지 전송.

> **Note**
>
> 사용자는 VPC 외부에서도 다른 Amazon SQS API 작업을 계속 사용할 수 있습니다. 자세한 내용은 예제 5: VPC 엔드포인트에서 온 액세스가 아닌 경우 액세스 거부을(를) 참조하세요.

Amazon SQS의 Identity and Access Management

AWS Identity and Access Management (IAM) 은 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와줍니다. IAM 관리자는 어떤 사용자가 Amazon SQS 리소스를 사용할 수 있도록 인증(로그인)되고 권한이 부여(권한 있음)될 수 있는지 제어합니다. IAM은 추가 AWS 서비스 비용 없이 사용할 수 있습니다.

고객

사용 방법 AWS Identity and Access Management (IAM) 은 Amazon SQS에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 – Amazon SQS 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 Amazon SQS 기능을 사용하여 작업를 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon SQS의 기능에 액세스할 수 없다면 Amazon Simple Queue Service ID 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 – 회사에서 Amazon SQS 리소스를 책임지고 있다면 Amazon SQS에 대한 완전한 액세스 권한이 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon SQS 기능과 리소스를 결정합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 Amazon SQS에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amazon Simple Queue Service가 IAM으로 작동하는 방식 섹션을 참조하세요.

IAM 관리자 - IAM 관리자라면 Amazon SQS에 대한 액세스 관리 정책 작성 방법을 자세히 알고 싶을 수도 있습니다. IAM에서 사용할 수 있는 Amazon SQS 자격 증명 기반 정책(identity-based policies)의 예제를 확인하시려면 자격 증명 기반 정책 예제 정책 모범 사례 섹션을 참조하세요.

ID를 통한 인증

인증은 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 페더레이션 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK) 와 명령줄 인터페이스 (CLI) 를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하십시오.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 태스크를 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 작업을 참조하십시오.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 비롯한 수동 AWS 서비스 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 액세스하도록 하는 것입니다.

페더레이션 ID는 기업 사용자 디렉토리, 웹 ID 공급자, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 자격 증명을 사용하여 액세스하는 AWS 서비스 모든 사용자를 말합니다. AWS Directory Service페더레이션 ID에 AWS 계정액세스하면 이들이 역할을 맡고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center(을)를 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 모든 사용자 및 애플리케이션에서 사용할 수 있습니다. AWS 계정 IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까?를 참조하십시오.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한을 AWS 계정 가진 사용자 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 보안 인증이 있는 IAM 사용자를 생성하는 대신 임시 보안 인증을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 보안 인증이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하십시오.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지

고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하십시오.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 사용을 참조하십시오.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 서드 파티 ID 공급자의 역할 생성 단원을 참조하십시오. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하십시오.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 계정 간 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하십시오.
- 서비스 간 액세스 — 일부는 다른 기능을 사용합니다. AWS 서비스 AWS 서비스예를 들어 서비스에서 직접적 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 태스크를 수행할 수 있습니다.
  - 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오.
  - 서비스 연결 역할 — 서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하십시오.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하십시오.

정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요를 참조하십시오.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

보안 인증 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

보안 인증 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하십시오.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우, 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 가이드의 ACL(액세스 제어 목록) 개요를 참조하십시오.

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 자격 증명 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 보안 인증 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 내용은 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계를 참조하십시오.
- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책 (SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 구성원 계정의 엔티티 (각 엔티티 포함) 에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하십시오.
- 세션 정책 - 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 보안 인증 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하십시오.

여러 정책 타입

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련되어 있을 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

Amazon SQS의 액세스 관리 개요

모든 AWS 리소스는 가 소유하며 AWS 계정, 리소스를 생성하거나 액세스할 수 있는 권한은 권한 정책에 따라 관리됩니다. 계정 관리자는 IAM 자격 증명(즉, 사용자, 그룹, 역할)에 권한 정책을 연결할 수 있고, 일부 서비스(예: Amazon SQS)에서는 리소스에 대한 권한 정책 연결도 지원합니다.

> **Note**
>
> 계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하십시오.

권한을 부여하려면 권한을 부여 받을 사용자, 권한 대상이 되는 리소스, 이 리소스에서 허용할 특정 작업을 지정합니다.

주제

- Amazon Simple Queue Service 리소스 및 작업
- 리소스 소유권 이해
- 리소스 액세스 관리
- 정책 요소 지정: 작업, 효과, 리소스, 보안 주체

Amazon Simple Queue Service 리소스 및 작업

Amazon SQS에서 유일한 리소스는 대기열입니다. 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다. 다음 리소스에는 연결된 고유한 ARN이 있습니다.

리소스 유형	ARN 형식
대기열	`arn:aws:sqs:` *`region`*`:`*`account_id`*`:`*`queue_name`*

다음은 대기열 ARN 형식의 몇 가지 예입니다.

- AWS 계정 123456789012에 속하는 미국 동부 (오하이오) my_queue 지역에서 이름이 지정된 대기열에 대한 ARN:

```
arn:aws:sqs:us-east-2:123456789012:my_queue
```

- 다음은 Amazon SQS가 지원하는 여러 리전 각각에서 my_queue라고 하는 대기열의 ARN입니다.

```
arn:aws:sqs:*:123456789012:my_queue
```

- 다음은 * 또는 ?를 대기열 이름에서 와일드카드로 사용하는 ARN입니다. 다음 예제에서 ARN은 my_prefix_로 시작하는 모든 대기열과 일치합니다.

```
arn:aws:sqs:*:123456789012:my_prefix_*
```

`GetQueueAttributes` 작업을 호출하여 기존 대기열의 ARN 값을 가져올 수 있습니다. `QueueArn` 속성 값은 대기열의 ARN입니다. ARN에 대한 자세한 내용은 IAM 사용 설명서의 IAM ARN 단원을 참조하세요.

Amazon SQS는 대기열 리소스를 사용하는 여러 작업을 제공합니다. 자세한 정보는 Amazon SQS API 권한: 작업 및 리소스 참조을 참조하세요.

리소스 소유권 이해

누가 리소스를 생성했는지에 상관없이 계정에서 생성된 리소스는 에서 AWS 계정 소유합니다. 특히, 리소스 소유자는 리소스 생성 요청을 인증하는 보안 주체 엔터티(즉, 루트 계정, 사용자 또는 IAM 역할)의 AWS 계정 입니다. 다음 예에서는 이러한 작동 방식을 설명합니다.

- 의 루트 계정 자격 증명을 사용하여 Amazon SQS 대기열을 생성하는 경우 사용자는 리소스의 소유자입니다 (Amazon SQS에서는 리소스가 Amazon SQS 대기열임). AWS 계정 AWS 계정
- 에서 사용자를 생성하고 사용자에게 대기열을 생성할 권한을 부여하면 사용자가 대기열을 생성할 수 있습니다. AWS 계정 하지만 (사용자가 속한) AWS 계정 은 대기열 리소스를 소유합니다.
- Amazon SQS 대기열을 생성할 권한이 AWS 계정 있는 IAM 역할을 생성하는 경우 해당 역할을 수임할 수 있는 사람은 누구나 대기열을 생성할 수 있습니다. AWS 계정 (역할이 속한) 사용자가 대기열 리소스를 소유합니다.

리소스 액세스 관리

권한 정책은 계정에 부여된 권한을 설명합니다. 다음 섹션에서는 권한 정책을 생성하는 데 사용할 수 있는 옵션에 대해 설명합니다.

> **Note**
>
> 이 섹션에서는 Amazon SQS의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. IAM 설명서 전체 내용은 IAM 사용 설명서의 IAM이란 무엇인가요? 단원을 참조하세요. IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

IAM 자격 증명에 연결된 정책을 자격 증명 기반 정책(IAM 정책)이라 하고, 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다.

보안 인증 기반 정책

사용자에게 Amazon SQS 대기열 권한을 부여하는 방법에는 Amazon SQS 정책 시스템을 사용하는 방법과 IAM 정책 시스템을 사용하는 방법이 있습니다. 두 시스템 중 하나 또는 둘 다를 사용하여 정책을 사용자 또는 역할에 연결할 수 있습니다. 대다수의 경우에 시스템을 사용하여 같은 결과를 얻을 수 있습니다. 예를 들어 다음을 수행할 수 있습니다.

- 계정 내 사용자 또는 그룹에 권한 정책 연결 - Amazon SQS 대기열 생성 권한을 사용자에게 부여하려면, 권한 정책을 사용자 또는 사용자가 속한 그룹에 연결합니다.
- 권한 정책을 다른 AWS 계정의 사용자에게 연결 - Amazon SQS 대기열 생성 권한을 사용자에게 부여하려면, Amazon SQS 권한 정책을 다른 AWS 계정의 사용자에게 연결합니다.

  다음 작업에는 교차 계정 권한이 적용되지 않습니다.

  - AddPermission
  - CancelMessageMoveTask
  - CreateQueue
  - DeleteQueue
  - ListMessageMoveTask
  - ListQueues
  - ListQueueTags
  - RemovePermission
  - SetQueueAttributes
  - StartMessageMoveTask
  - TagQueue
  - UntagQueue
- 역할에 권한 정책 연결(교차 계정 권한 부여) - 교차 계정 권한을 부여하려면, 자격 증명 기반 권한 정책을 IAM 역할에 연결합니다. 예를 들어 AWS 계정 A 관리자는 다음과 같이 AWS 계정 B (또는 AWS 서비스) 에 계정 간 권한을 부여하는 역할을 만들 수 있습니다.
  - 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 권한 정책을 역할에 연결합니다.
  - 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
  - 계정 B 관리자는 역할을 담당할 권한을 계정 B의 사용자에게 위임합니다. 이렇게 하면 계정 B의 사용자가 계정 A에서 대기열을 생성하거나 액세스할 수 있습니다.

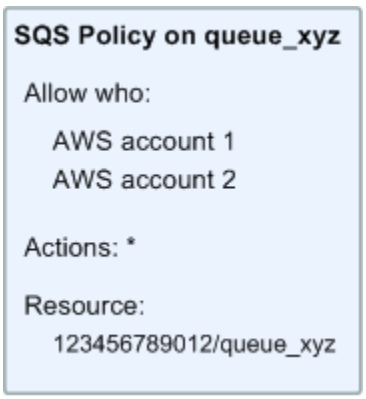
> **Note**
>
> 서비스에 역할을 수임할 권한을 부여하려는 경우 신뢰 정책의 보안 주체가 AWS 서비스 AWS 주체일 수도 있습니다.

IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 설명은 IAM 사용자 가이드의 액세스 관리 섹션을 참조하십시오.

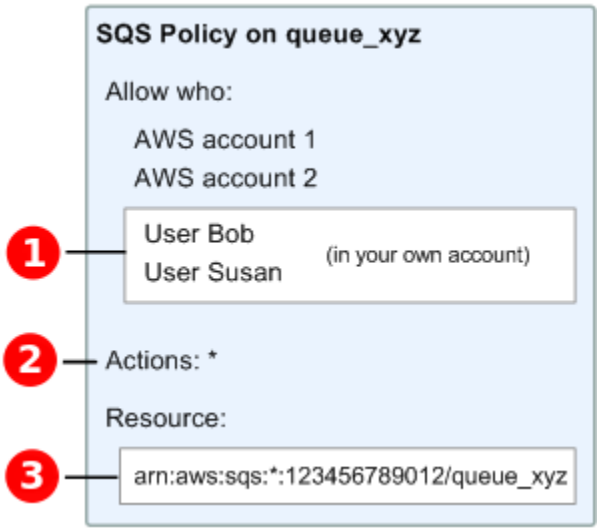
Amazon SQS가 IAM 정책을 사용하는 동안에는 자체 정책 인프라를 보유합니다. 대기열과 함께 Amazon SQS 정책을 사용하여 대기열에 액세스할 수 있는 AWS 계정을 지정할 수 있습니다. 액세스 및 조건 유형을 지정할 수 있습니다(예: 2010년 12월 31일 이전에 요청한 경우 `SendMessage`, `ReceiveMessage`를 사용할 권한을 부여하는 조건). 권한을 부여할 수 있는 특정 작업은 전체 Amazon SQS 작업 목록의 하위 집합입니다. Amazon SQS 정책을 작성하고 *를 지정하여 "모든 Amazon SQS 작업을 허용"하도록 지정하면 이것은 사용자가 이 하위 집합에서 모든 작업을 수행할 수 있음을 의미합니다.

다음 다이어그램은 작업의 하위 집합과 관련된 이 기본 Amazon SQS 정책 중 하나의 개념을 보여줍니다. 이 정책은 `queue_xyz` 용이며 AWS 계정 1과 AWS 계정 2에 지정된 대기열에서 허용된 모든 작업을 사용할 수 있는 권한을 부여합니다.

> **Note**
>
> 정책의 리소스는 다음과 같이 `123456789012/queue_xyz` 지정됩니다. 여기서 `123456789012` 은 큐를 소유한 계정의 AWS 계정 ID입니다.



IAM 소개 및 사용자와 Amazon 리소스 이름(ARN)의 개념에서 SQS 정책에 대한 몇 가지 사항이 바뀌었습니다. 다음 다이어그램과 표에는 이 변경 사항에 대한 설명이 나와 있습니다.



1 여러 계정의 사용자에게 권한을 부여하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 자습서: IAM 역할을 사용한 AWS 계정 간 액세스 위임을 참조하십시오.

2 *에 포함된 작업의 하위 집합이 확장되었습니다. 허용된 작업 목록에 대해서는 Amazon SQS API 권한: 작업 및 리소스 참조을 참조하십시오.

3 IAM 정책에서 리소스를 지정하는 기본 수단인 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정할 수 있습니다. Amazon SQS 대기열의 ARN 형식에 대한 정보는 Amazon Simple Queue Service 리소스 및 작업 섹션을 참조하세요.

예를 들어 위 다이어그램의 Amazon SQS 정책에 따르면 AWS 계정 1 또는 AWS 계정 2의 보안 자격 증명을 소유한 사람은 누구나 액세스할 수 있습니다. queue_xyz 또한, 자체 AWS 계정(ID 123456789012)에 있는 사용자 Bob과 Susan은 대기열에 액세스할 수 있습니다.

IAM 소개에 앞서 Amazon SQS는 대기열의 생성자에게 대기열에 대한 전체 제어권을 자동으로 부여합니다(즉, 이 대기열에서 가능한 Amazon SQS 작업 전체에 대한 액세스). 생성자가 AWS 보안 자격 증명을 사용하지 않을 경우 이것은 더 이상 적용되지 않습니다. 대기열 생성 권한을 가진 사용자는 누구나 생성된 대기열에서 어떤 작업이든 할 수 있도록 다른 Amazon SQS 작업을 사용할 권한이 있어야 합니다.

다음은 한 사용자에게 모든 Amazon SQS 작업을 사용하도록 허용하는 정책 예제이지만, 리터럴 문자열 bob_queue_로 시작하는 이름을 가진 대기열에만 적용됩니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:*:123456789012:bob_queue_*"
   }]
}
```

자세한 내용은 Amazon SQS에서 정책 사용 및 IAM 사용 설명서의 자격 증명(사용자, 그룹 및 역할) 섹션을 참조하세요.

정책 요소 지정: 작업, 효과, 리소스, 보안 주체

각 Amazon Simple Queue Service 리소스마다 서비스는 일련의 작업을 정의합니다. 이러한 작업에 대한 권한을 부여하기 위해 Amazon SQS는 정책에서 지정할 수 있는 작업을 정의합니다.

> **Note**
>
> 작업을 실시하려면 둘 이상의 작업에 대한 권한이 필요할 수 있습니다. 특정 작업에 대한 권한을 부여할 때 해당 작업이 허용되거나 거부되는 리소스도 식별합니다.

다음은 가장 기본적인 정책 요소입니다.

- 리소스 – 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다.
- 작업 - 작업 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들어, `sqs:CreateQueue` 권한은 사용자에게 Amazon Simple Queue Service `CreateQueue` 작업 수행을 허용합니다.
- 결과 – 사용자가 특정 작업을 요청하는 경우의 결과를 지정합니다. 이는 허용 또는 거부 중에 하나가 될 수 있습니다. 리소스에 대한 액세스 권한을 명시적으로 부여하지 않으면 액세스가 암시적으로 거부됩니다. 한 리소스에 대한 액세스를 명시적으로 거부할 수도 있으며, 다른 정책에 따라 액세스 권한이 부여되더라도 사용자가 이 리소스에 액세스하지 못하도록 조치를 취할 수 있습니다.
- Principal – 자격 증명 기반 정책(IAM 정책)에서 정책이 연결되는 사용자는 암시적인 보안 주체입니다. 리소스 기반 정책의 경우, 사용자, 계정, 서비스 또는 권한의 수신자인 기타 개체를 지정합니다(리소스 기반 정책에만 해당).

Amazon SQS 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 AWS IAM 정책 참조 섹션을 참조하세요.

모든 Amazon Simple Queue Service 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 Amazon SQS API 권한: 작업 및 리소스 참조 섹션을 참조하세요.

Amazon Simple Queue Service가 IAM으로 작동하는 방식

IAM을 사용하여 Amazon SQS에 대한 액세스를 관리하기 전에 Amazon SQS에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

Amazon SQS와 함께 사용할 수 있는 IAM 기능

IAM 특성	Amazon SQS 지원
ID 기반 정책	예
리소스 기반 정책	예
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACLs	아니요
ABAC(정책 내 태그)	부분
임시 보안 인증	예
전달 액세스 세션(FAS)	예
서비스 역할	예
서비스 연결 역할	아니요

Amazon SQS 및 AWS 기타 서비스가 대부분의 IAM 기능과 어떻게 작동하는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 함께 작동하는 서비스를AWS 참조하십시오.

액세스 제어

ACL(액세스 통제 목록)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 가이드의 ACL(액세스 제어 목록) 개요를 참조하십시오.

> **Note**
>
> 누구나 자신의 계정에 속한 사용자에게 권한을 AWS 계정 위임할 수 있다는 점을 이해하는 것이 중요합니다. 교차 계정 액세스를 사용하면 추가 사용자를 관리할 필요 없이 AWS 리소스에 대한 액세스를 공유할 수 있습니다. 교차 계정 액세스 사용에 대한 자세한 정보는 IAM 사용 설명서의 교차 계정 액세스 사용을 참조하세요.
> Amazon SQS 사용자 지정 정책 내의 콘텐츠 간 권한 및 조건 키에 대한 자세한 내용은 Amazon SQS 사용자 지정 정책의 제한 섹션을 참조하세요.

Amazon SQS의 자격 증명 기반 정책

보안 인증 기반 정책 지원	예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 보안 인증 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

Amazon SQS의 자격 증명 기반 정책 예

Amazon SQS 자격 증명 기반 정책 예제를 보려면 정책 모범 사례 섹션을 참조하세요.

Amazon SQS 내의 리소스 기반 정책

리소스 기반 정책 지원	예

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우, 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 등이 포함될 수 있습니다. AWS 서비스

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 다른 AWS 계정경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 개체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하십시오.

Amazon SQS의 정책 작업

정책 작업 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

Amazon SQS 작업의 목록을 보려면 서비스 권한 부여 참조의 Amazon Simple Queue Service에서 정의한 작업을 참조하세요.

Amazon SQS의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
sqs
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "sqs:action1",
      "sqs:action2"
         ]
```

Amazon SQS 자격 증명 기반 정책 예제를 보려면 정책 모범 사례 섹션을 참조하세요.

Amazon SQS의 정책 리소스

정책 리소스 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 태스크를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon SQS 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조의 Amazon Simple Queue Service에서 정의한 작업을 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Simple Queue Service에서 정의한 리소스를 참조하세요.

Amazon SQS 자격 증명 기반 정책 예제를 보려면 정책 모범 사례 섹션을 참조하세요.

Amazon SQS의 정책 조건 키

서비스별 정책 조건 키 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 연산을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예컨대, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하십시오.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 AWS 설명서의 글로벌 조건 컨텍스트 키를 참조하십시오.

Amazon SQS 조건 키 목록을 보려면 서비스 권한 부여 참조의 Amazon Simple Queue Service에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Simple Queue Service에서 정의한 리소스를 참조하세요.

Amazon SQS 자격 증명 기반 정책 예제를 보려면 정책 모범 사례 섹션을 참조하세요.

Amazon SQS의 ACL

ACL 지원	아니요

ACL(액세스 통제 목록)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon SQS의 ABAC

ABAC(정책 내 태그) 지원	부분

ABAC(속성 기반 액세스 통제)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔티티 (사용자 또는 역할) 및 여러 AWS 리소스에 태그를 첨부할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 정보는 IAM 사용 설명서의 ABAC란 무엇입니까?를 참조하십시오. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하십시오.

Amazon SQS에서 임시 자격 증명 사용

임시 보안 인증 지원	예

임시 자격 증명을 사용하여 로그인하면 작동하지 AWS 서비스 않는 것도 있습니다. 임시 자격 증명을 사용하는 방법을 AWS 서비스 비롯한 추가 정보는 IAM 사용 설명서의 IAM과AWS 서비스 연동되는 내용을 참조하십시오.

사용자 이름과 암호를 제외한 다른 방법을 AWS Management Console 사용하여 로그인하면 임시 자격 증명을 사용하는 것입니다. 예를 들어 회사의 SSO (Single Sign-On) 링크를 AWS 사용하여 액세스하는 경우 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하십시오.

또는 API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다 AWS CLI . AWS 그런 다음 해당 임시 자격 증명을 사용하여 액세스할 수 AWS있습니다. AWS 장기 액세스 키를 사용하는 대신 임시 자

격 증명을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하십시오.

Amazon SQS용 포워드 액세스 세션

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 작업을 수행하는 AWS경우 사용자는 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 서비스에 AWS 서비스 요청하라는 요청과 결합하여 사용합니다. AWS 서비스 FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Amazon SQS의 서비스 역할

서비스 역할 지원	예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon SQS 기능이 중단될 수 있습니다. Amazon SQS에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Amazon SQS의 서비스 연결 역할

서비스 연결 역할 지원	아니요

서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되

며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하십시오. 서비스 연결 역할 열에서 Yes(이)가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

관리형 정책에 대한 AWS Amazon SQS 업데이트

사용자, 그룹 또는 역할에 권한을 추가할 때 정책을 직접 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 더욱 편리합니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하려면 시간과 전문 지식이 필요합니다. 빨리 시작하려면 AWS 관리형 정책을 사용할 수 있습니다. 이러한 정책은 일반적인 사용 사례에 적용되며 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 AWS 설명서의 관리형 정책을 참조하십시오.

AWS 서비스는 AWS 관리형 정책을 유지 관리하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스가 새 기능을 지원하기 위해 AWS 관리형 정책에 권한을 추가하는 경우가 있습니다. 이 유형의 업데이트는 정책이 연결된 모든 ID(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새 기능이 출시되거나 새 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 높습니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한 여러 서비스에 걸친 작업 기능에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 ReadOnly액세스 AWS 관리형 정책은 모든 AWS 서비스와 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면 새 작업 및 리소스에 대한 읽기 전용 권한이 AWS 추가됩니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한AWS 관리형 정책을 참조하십시오.

AWS 관리형 정책: AmazonSQS FullAccess

`AmazonSQSFullAccess` 정책을 Amazon SQS 자격 증명에 연결할 수 있습니다. 이 정책은 Amazon SQS에 대한 전체 액세스를 허용하는 권한을 부여합니다.

이 정책에 대한 권한을 보려면 AWS 관리형 정책 FullAccess 참조의 AmazonSQS를 참조하십시오.

AWS 관리형 정책: AmazonSQS 액세스 ReadOnly

`AmazonSQSReadOnlyAccess` 정책을 Amazon SQS 자격 증명에 연결할 수 있습니다. 이 정책은 Amazon SQS에 대한 읽기 전용 액세스를 허용하는 권한을 부여합니다.

이 정책에 대한 권한을 보려면 AWS 관리형 정책 참조의 AmazonSQS ReadOnly Access를 참조하십시오.

관리형 정책에 대한 AWS Amazon SQS 업데이트

이 서비스가 변경 사항을 추적하기 시작한 이후 Amazon SQS의 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인하십시오. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon SQS 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AmazonSQS 액세스 ReadOnly	Amazon SQS는 특정 소스 대기열 아래에 가장 최근의 메시지 이동 작업(최대 10개)을 나열할 수 있는 새 작업을 추가했습니다. 이 작업은 `ListMessageMoveTasks` API 작업과 연결되어 있습니다.	2023년 6월 9일

Amazon Simple Queue Service ID 및 액세스 문제 해결

다음 정보를 사용하여 Amazon SQS 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon SQS에서 작업을 수행할 권한이 없음
- 저는 IAM을 수행할 권한이 없습니다. PassRole
- 외부 사용자가 내 Amazon SQS 리소스에 액세스할 AWS 계정 수 있도록 허용하고 싶습니다.

Amazon SQS에서 작업을 수행할 권한이 없음

작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 `mateojackson` 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `sqs:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sqs:GetWidget on resource: my-example-widget
```

이 경우 Mateo의 정책은 `sqs:GetWidget` 작업을 사용하여 *my-example-widget* 리소스에 액세스하도록 허용하도록 업데이트해야 합니다.

도움이 필요하면 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

저는 IAM을 수행할 권한이 없습니다. PassRole

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon SQS에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부는 새 서비스 역할 또는 서비스 연결 역할을 만드는 대신 기존 역할을 해당 서비스에 전달할 수 AWS 서비스 있도록 합니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 Amazon SQS에서 태스크를 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요하면 관리자에게 문의하세요. AWS 관리자는 로그인 자격 증명을 제공한 사람입니다.

외부 사용자가 내 Amazon SQS 리소스에 액세스할 AWS 계정 수 있도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하십시오.

- Amazon SQS에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon Simple Queue Service가 IAM으로 작동하는 방식 섹션을 참조하세요.

- 소유하고 AWS 계정 있는 모든 리소스에 대한 액세스를 제공하는 방법을 알아보려면 IAM 사용 설명서의 다른 AWS 계정 IAM 사용자에게 액세스 권한 제공을 참조하십시오.
- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정AWS 계정 소유에 대한 액세스 제공을 참조하십시오.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 페더레이션)을 참조하십시오.
- 교차 계정 액세스에 대한 역할 사용과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 교차 계정 리소스 액세스를 참조하십시오.

Amazon SQS에서 정책 사용

이 주제에서는 계정 관리자가 IAM 자격 증명(사용자, 그룹, 역할)에 권한 정책을 연결할 수 있는 자격 증명 기반 정책 예제를 제시합니다.

> **Important**
>
> Amazon Simple Queue Service 리소스에 대한 액세스 관리를 위해 제공되는 기본 개념과 옵션 설명에 대한 소개 주제 부분을 먼저 읽어 보는 것이 좋습니다. 자세한 정보는 Amazon SQS의 액세스 관리 개요을 참조하세요.
>
> `ListQueues`를 제외한 모든 Amazon SQS 작업은 리소스 수준 권한을 지원합니다. 자세한 정보는 Amazon SQS API 권한: 작업 및 리소스 참조을 참조하세요.

주제

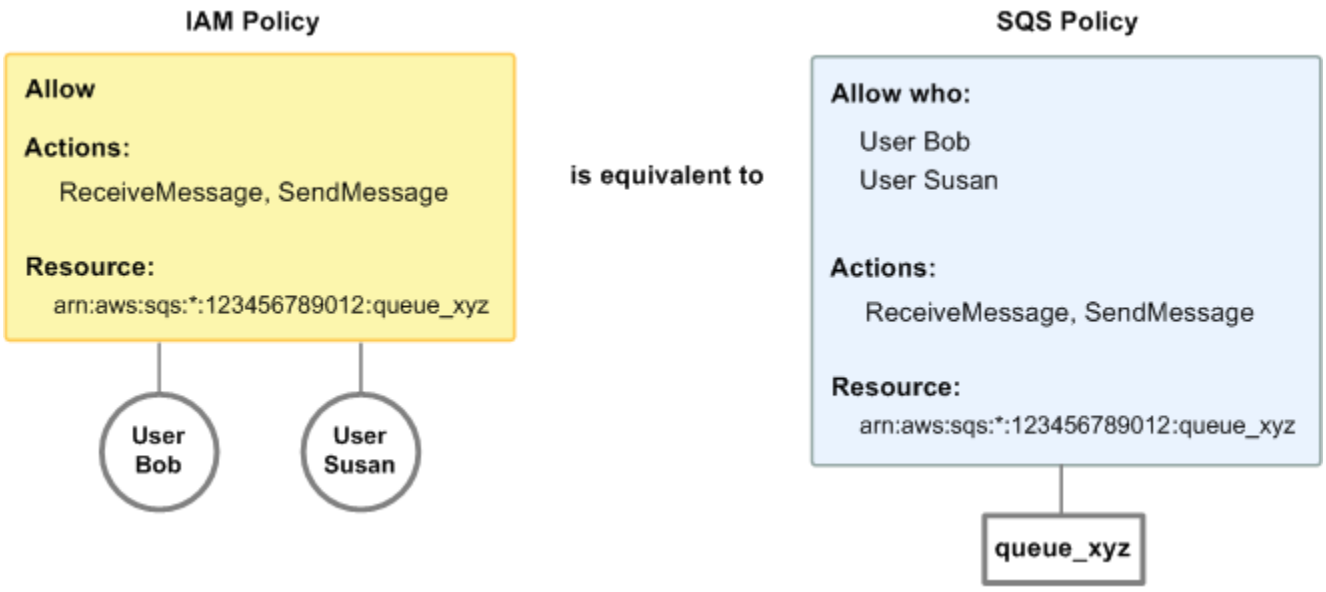
- Amazon SQS 및 IAM 정책 사용
- Amazon SQS 콘솔 사용에 필요한 권한
- Amazon SQS의 자격 증명 기반 정책 예
- Amazon SQS 정책의 기본 예제
- Amazon SQS 액세스 정책 언어와 함께 사용자 지정 정책 사용

Amazon SQS 및 IAM 정책 사용

사용자에게 Amazon SQS 리소스 권한을 부여하는 방법에는 Amazon SQS 정책 시스템을 사용하는 방법과 IAM 정책 시스템을 사용하는 방법이 있습니다. 전자나 후자 또는 둘 다 사용할 수 있습니다. 대개는 어떤 방법으로도 같은 결과를 얻을 수 있습니다.

예를 들어, 다음 다이어그램은 동일한 IAM 정책과 Amazon SQS 정책을 보여줍니다. IAM 정책은 Amazon ReceiveMessage SQS에 대한 SendMessage 권한과 AWS 계정에서 queue_xyz 호출된 대기열에 대한 작업을 부여하며, 정책은 Bob and Susan이라는 사용자에게 연결됩니다 (Bob과 Susan은 정책에 명시된 권한을 가짐). 이 Amazon SQS 정책 역시 동일한 대기열에서 ReceiveMessage 및 SendMessage 작업에 대한 권한을 Bob과 Susan에게 부여합니다.

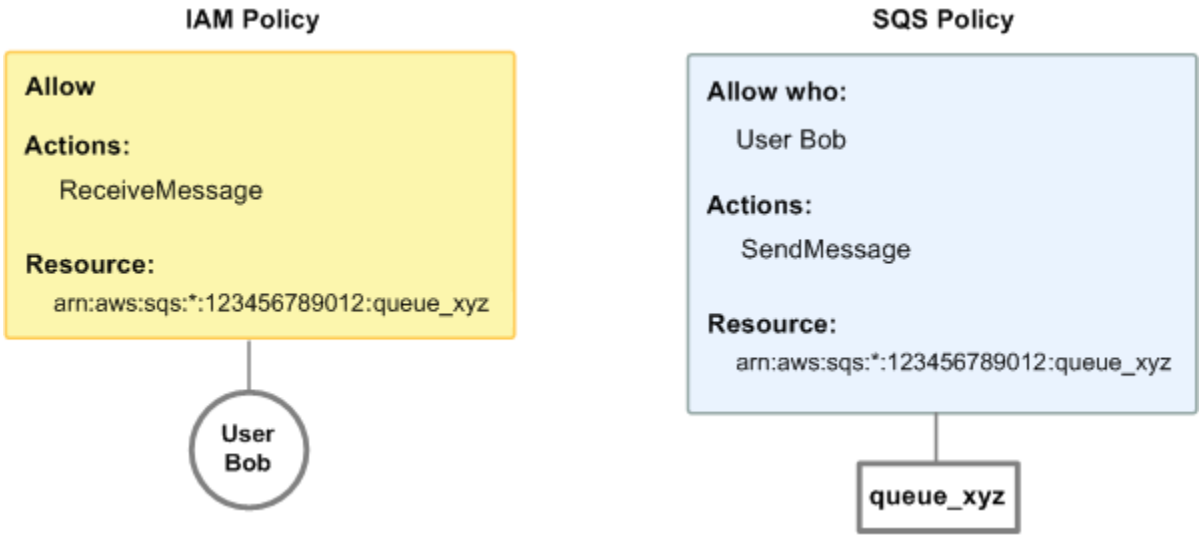
> **Note**
>
> 다음 예는 조건이 없는 간단한 정책을 보여줍니다. 두 정책 중 어느 쪽에서도 특정 조건을 지정하고 동일한 결과를 얻을 수 있습니다.



IAM과 Amazon SQS 정책 간에는 한 가지 주요 차이점이 있습니다. Amazon SQS 정책 시스템에서는 AWS 다른 계정에 권한을 부여할 수 있지만 IAM은 그렇지 않다는 것입니다.

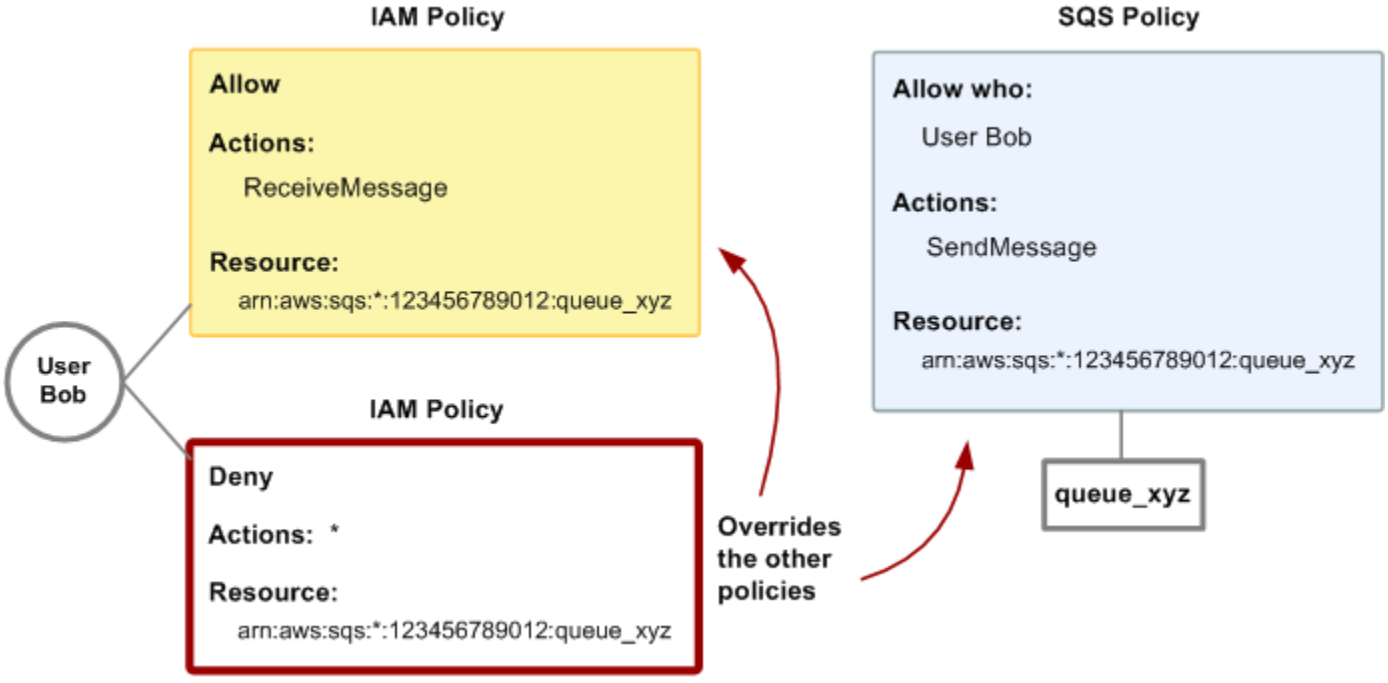
사용자의 필요에 따라 두 시스템을 함께 사용하여 권한을 관리해야 합니다. 다음 예는 두 정책 시스템이 어떻게 연계하는지 보여줍니다.

- 첫 번째 예에서 Bob은 IAM 정책과 Amazon SQS 정책 모두를 자신의 계정에 적용했습니다. IAM 정책은 queue_xyz에서 ReceiveMessage 작업에 대한 계정 권한을 부여하지만, Amazon SQS 정책은 동일한 대기열에서 SendMessage 작업에 대한 계정 권한을 부여합니다. 다음 다이어그램에서 관련 개념을 설명합니다.



Bob이 ReceiveMessage 요청을 queue_xyz에 전송하면, IAM 정책은 이 작업을 허용합니다. Bob이 SendMessage 요청을 queue_xyz에 전송하면, Amazon SQS 정책은 이 작업을 허용합니다.

- 두 번째 예에서 Bob은 queue_xyz에 대한 액세스 권한을 침해하므로, 해당 대기열에 대한 전체 액세스 권한을 제거해야 합니다. 가장 쉬운 방법은 모든 대기열 작업에 대한 Bob의 액세스를 거부하는 정책을 추가하는 것입니다. 명시적 deny는 항상 allow를 무시하기 때문에 이 정책은 다른 2개 정책을 무시합니다. 정책 평가 로직에 대한 자세한 내용은 Amazon SQS 액세스 정책 언어와 함께 사용자 지정 정책 사용를 참조하십시오. 다음 다이어그램에서 관련 개념을 설명합니다.



Bob이 어떤 방식으로 대기열에 액세스하든 간에 거부하는 문을 Amazon SQS 정책에 추가할 수도 있습니다. 이것은 Bob의 대기열 액세스를 거부하는 IAM 정책을 추가하는 것과 동일한 효과가 있습

니다. Amazon SQS 작업 및 리소스에 대한 정책의 예는 Amazon SQS 정책의 기본 예제에서 확인하세요. Amazon SQS 정책 작성에 대한 자세한 내용은 Amazon SQS 액세스 정책 언어와 함께 사용자 지정 정책 사용 섹션을 참조하세요.

Amazon SQS 콘솔 사용에 필요한 권한

사용자가 Amazon SQS 콘솔로 작업하려면 사용자의 AWS 계정에서 Amazon SQS 대기열로 작업할 수 있는 최소 개수의 권한이 있어야 합니다. 예를 들어, 사용자에게 `ListQueues` 작업을 호출할 권한이 있어 대기열을 나열할 수 있거나 `CreateQueue` 작업을 호출할 권한이 있어 대기열을 생성할 수 있어야 합니다. Amazon SQS 권한 외에도 Amazon SQS 대기열에서 Amazon SNS 주제를 구독하려면 콘솔에 Amazon SNS 작업에 대한 권한도 있어야 합니다.

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 콘솔은 해당 IAM 정책에 연결된 사용자에 대해 의도대로 작동하지 않을 수 있습니다.

AWS CLI 또는 Amazon SQS 작업만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다.

Amazon SQS의 자격 증명 기반 정책 예

기본적으로 사용자 및 역할은 Amazon SQS 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하십시오.

각 리소스 유형에 대한 ARN 형식을 포함하여 Amazon SQS에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조에서 Amazon Simple Queue Service에 대한 작업, 리소스 및 조건 키를 참조하세요.

> **Note**
>
> Amazon EC2 Auto Scaling의 수명 주기 후크를 구성할 때 Amazon SQS 대기열로 메시지를 전송하는 정책을 작성하지 않아도 됩니다. 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 Auto Scaling 라이프사이클 후크를 참조하십시오.

주제

- 정책 모범 사례
- Amazon SQS 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 사용자가 대기열을 생성할 수 있도록 허용
- 개발자가 공유 대기열에 메시지를 쓸 수 있도록 허용
- 관리자가 대기열의 일반적인 크기를 알 수 있도록 허용
- 파트너가 특정 대기열로 메시지를 보낼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon SQS 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책으로 시작하고 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반적인 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하십시오. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 직무에 대한AWS 관리형 정책을 참조하십시오.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM의 정책 및 권한을 참조하십시오.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하십시오.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하십시오.
- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정

책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA 보호 API 액세스 구성을 참조하십시오.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하십시오.

Amazon SQS 콘솔 사용

Amazon Simple Queue Service 콘솔에 액세스하려면 최소한의 권한 집합이 있어야 합니다. 이러한 권한을 통해 내 Amazon SQS 리소스를 나열하고 세부 정보를 볼 수 있어야 합니다. AWS 계정최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 엔터티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 Amazon SQS 콘솔을 계속 사용할 수 있도록 하려면 Amazon `AmazonSQSReadOnlyAccess` AWS SQS 관리형 정책도 엔티티에 연결하십시오. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "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": "*"
        }
    ]
}
```

사용자가 대기열을 생성할 수 있도록 허용

다음 예제에서는 Bob이 모든 Amazon SQS 작업에 액세스할 수 있도록 정책을 생성하지만, 리터럴 문자열 `alice_queue_`로 시작하는 이름을 가진 대기열에만 해당됩니다.

Amazon SQS는 대기열 생성자에게 대기열 사용 권한을 자동으로 부여하지 않습니다. 따라서 IAM 정책의 `CreateQueue` 작업 외에도 모든 Amazon SQS 작업을 사용할 수 있는 권한을 Bob에게 명시적으로 부여해야 합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:*:123456789012:alice_queue_*"
   }]
}
```

개발자가 공유 대기열에 메시지를 쓸 수 있도록 허용

다음 예시에서는 개발자용 그룹을 생성하고, 그룹이 지정된 항목에 속하고 이름이 지정된 AWS 계정 대기열만 사용하여 Amazon SQS `SendMessage` 작업을 사용하도록 허용하는 정책을 연결합니다. `MyCompanyQueue`

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:*:123456789012:MyCompanyQueue"
   }]
}
```

SendMessage 대신에 *를 사용하여 공유 대기열에서 보안 주체에게 ChangeMessageVisibility, DeleteMessage, GetQueueAttributes, GetQueueUrl, ReceiveMessage 및 SendMessage 작업을 허용할 수 있습니다.

> **Note**
>
> *에는 다른 권한 유형에서 제공하는 액세스 권한이 포함되나 Amazon SQS는 권한을 개별적으로 고려합니다. 예를 들어, *에 SendMessage에서 제공하는 액세스 권한이 포함되어 있더라도 * 및 SendMessage 권한을 사용자에게 부여할 수 있습니다.
> 이 개념은 권한을 제거할 때도 적용됩니다. 보안 주체가 * 권한만 있는 경우, SendMessage 권한을 제거하도록 요청해도 보안 주체에게 everything-but 권한이 남는 것은 아닙니다. 그 대신 보안 주체가 명시적 SendMessage 권한을 소유하지 않았기 때문에 이 요청은 아무런 영향도 주지 않습니다. ReceiveMessage 권한만 보안 주체에게 남기려면 먼저 ReceiveMessage 권한을 추가한 후 * 권한을 제거합니다.

관리자가 대기열의 일반적인 크기를 알 수 있도록 허용

다음 예시에서는 관리자용 그룹을 만들고 그룹이 지정된 계정에 속하는 모든 대기열과 함께 Amazon SQS GetQueueAttributes 작업을 사용할 수 있도록 하는 정책을 연결합니다. AWS

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

파트너가 특정 대기열로 메시지를 보낼 수 있도록 허용

Amazon SQS 정책 또는 IAM 정책을 사용하여 이 작업을 완료할 수 있습니다. 파트너에게 Amazon SQS 정책이 AWS 계정있는 경우 Amazon SQS 정책을 사용하는 것이 더 쉬울 수 있습니다. 하지만 AWS 보안 자격 증명을 소유한 파트너 회사의 모든 사용자는 대기열에 메시지를 보낼 수 있습니다. 액세스를 특정 사용자나 애플리케이션으로 제한하려면, 파트너를 본인 회사의 사용자처럼 다루면서 Amazon SQS 정책이 아닌 IAM 정책을 사용해야 합니다.

이 예제에서는 다음 작업을 수행합니다.

1. 파트너 회사를 WidgetCo 대표할 그룹을 만드십시오.
2. 파트너사에서 액세스 권한을 필요로 하는 특정 사용자나 애플리케이션을 위해 사용자를 생성합니다.
3. 사용자를 그룹에 추가합니다.
4. `WidgetPartnerQueue`라는 대기열에서만 `SendMessage` 작업에 유일하게 액세스할 수 있는 권한을 이 그룹에 부여하는 정책을 연결합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
         "Effect": "Allow",
         "Action": "sqs:SendMessage",
         "Resource": "arn:aws:sqs:*:123456789012:WidgetPartnerQueue"
   }]
}
```

Amazon SQS 정책의 기본 예제

이 단원에서는 일반 Amazon SQS 사용 사례에 대한 정책 예제를 보여줍니다.

콘솔을 사용하여 정책을 사용자에게 연결할 때 각 정책의 효과를 확인할 수 있습니다. 처음에 사용자는 권한을 가지고 있지 않으며 콘솔에서 어떠한 작업도 수행할 수 없습니다. 정책을 사용자에게 연결하면 사용자가 콘솔에서 다양한 작업을 수행할 수 있는지 확인할 수 있습니다.

> **Note**
>
> 두 개의 브라우저 창을 사용하는 것이 좋습니다. 하나는 권한을 부여하는 창이고 다른 하나는 사용자에게 권한을 부여할 때 사용자 자격 증명을 AWS Management Console 사용하여 권한을 확인하는 데 로그인하는 창입니다.

예 1: 하나에 권한 1개 부여 AWS 계정

다음 예제 정책은 미국 동부 (오하이오) 444455556666/queue1 지역에서 이름이 지정된 대기열에 대한 SendMessage 권한을 AWS 계정 111122223333 번호에 부여합니다.

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

예 2: 하나에 두 개의 권한 부여 AWS 계정

다음 예제 정책은 이름이 지정된 대기열에 AWS 계정 SendMessage 번호와 ReceiveMessage 권한을 111122223333 모두 444455556666/queue1 부여합니다.

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

예 3: 모든 권한을 2명에게 부여 AWS 계정

다음 예제 정책은 Amazon SQS가 미국 동부 (오하이오444455556666) 123456789012/queue1 지역에서 이름이 지정된 대기열에 대한 공유 액세스를 허용하는 모든 작업을 사용할 수 있는 두 개의 다른 AWS 계정 번호 (111122223333및) 권한을 부여합니다.

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

예제 4: 역할과 사용자 이름에 교차 계정 권한 부여

다음 예제 정책은 Amazon SQS가 미국 동부 (오하이오) 지역에서 123456789012/queue1 이름이 지정된 대기열에 대한 공유 액세스를 허용하는 모든 작업을 사용할 수 있는 AWS 계정 번호 111122223333 교차 계정 권한을 role1 부여합니다. username1

다음 작업에는 교차 계정 권한이 적용되지 않습니다.

- 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"
   }]
}
```

예제 5: 모든 사용자에게 권한 부여

다음 정책 예제에서는 111122223333/queue1이라는 대기열에 대한 ReceiveMessage 권한을 모든 사용자(익명 사용자)에게 부여합니다.

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

예제 6: 모든 사용자에게 시간 제한적인 권한 부여

다음 정책 예제는 모든 사용자(익명 사용자)에게 111122223333/queue1라는 이름의 대기열에 대한 ReceiveMessage 권한을 부여하지만, 2009년 1월 31일에는 오후 12시(정오)부터 오후 3시까지만 가능합니다.

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

예제 7: CIDR 범위의 모든 사용자에게 모든 권한 부여

다음 정책 예제에서는 모든 사용자(익명 사용자)에게 111122223333/queue1이라는 대기열에서 공유 가능한 모든 Amazon SQS 작업을 사용할 수 있는 권한을 부여하지만, 이것은 요청이 192.0.2.0/24 CIDR 범위에서 들어오는 경우에 한합니다.

```
{
```

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

예제 8: 서로 다른 CIDR 범위의 사용자에게 권한 허용(화이트리스트) 및 차단(블랙리스트)

다음의 정책 예제에는 문이 2개입니다.

- 첫 번째 문은 192.0.2.0/24 CIDR 범위(192.0.2.188 제외)에 있는 모든 사용자(익명 사용자)에게 111122223333/queue1이라는 대기열에서 SendMessage 작업을 사용할 수 있는 권한을 부여합니다.
- 두 번째 설명은 12.148.72.0/23 CIDR 범위의 모든 사용자(익명 사용자)가 대기열을 사용하지 못하도록 차단합니다.

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

Amazon SQS 액세스 정책 언어와 함께 사용자 지정 정책 사용

AWS 계정 ID 및 기본 권한 (예: for SendMessageor ReceiveMessage) 을 기반으로 Amazon SQS 액세스를 허용하려는 경우 자체 정책을 작성할 필요가 없습니다. Amazon SQS AddPermission 작업만 사용해도 됩니다.

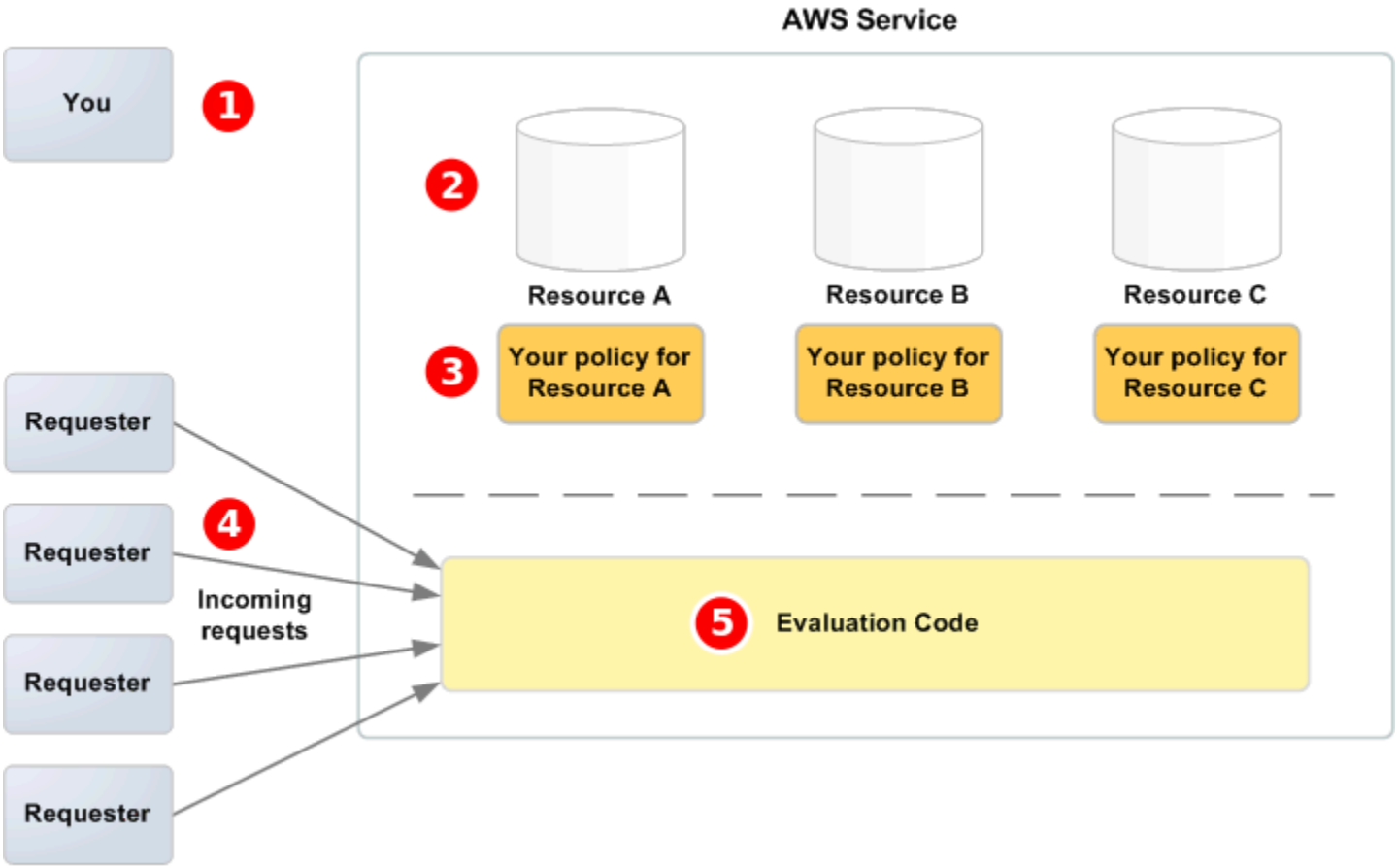
보다 구체적인 조건 (예: 요청이 들어오는 시간 또는 요청자의 IP 주소) 에 따라 액세스를 명시적으로 거부하거나 허용하려면 Amazon SQS 정책을 작성하고 Amazon SQS 작업을 사용하여 AWS 시스템에 업로드해야 합니다. SetQueueAttributes

주제

- Amazon SQS 액세스 제어 아키텍처
- Amazon SQS 액세스 제어 프로세스 워크플로
- Amazon SQS 액세스 정책 언어 주요 개념
- Amazon SQS 액세스 정책 언어 평가 로직
- Amazon SQS 액세스 정책 언어의 명시적 거부와 기본 거부 간의 관계
- Amazon SQS 사용자 지정 정책의 제한
- 사용자 지정 Amazon SQS 액세스 정책 언어 예제

Amazon SQS 액세스 제어 아키텍처

다음 그림에서는 Amazon SQS 리소스의 액세스 제어에 대해 설명합니다.



1

리소스 소유자

2

AWS 서비스에 포함된 리소스 (예: Amazon SQS 대기열)

3

정책. 리소스당 하나의 정책을 가지는 것이 좋습니다. 이 AWS 서비스는 정책을 업로드하고 관리하는 데 사용하는 API를 제공합니다.

4

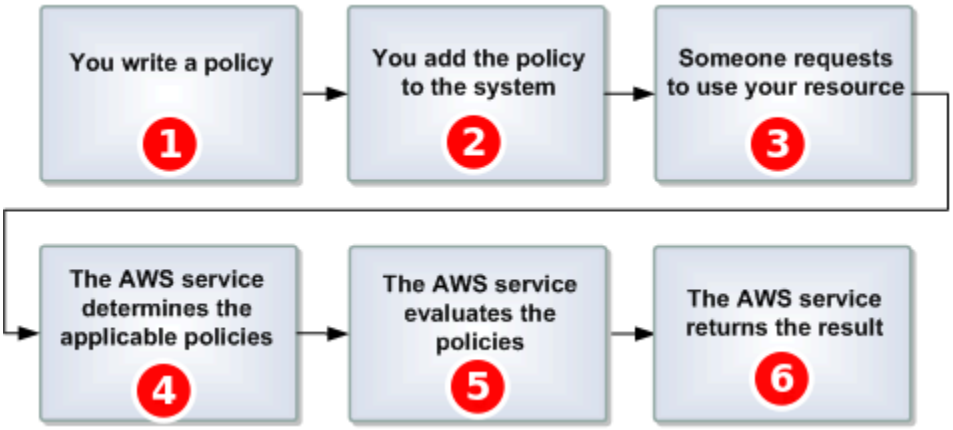
요청자 및 요청자가 AWS 서비스에서 수신하는 요청.

5

액세스 정책 언어 평가 코드. 이는 수신 요청을 적용 가능한 정책에 따라 평가하고 요청자가 리소스에 액세스할 수 있는지 여부를 결정하는 AWS 서비스 내 코드 세트입니다.

Amazon SQS 액세스 제어 프로세스 워크플로

다음 그림에서는 Amazon SQS 액세스 정책 언어를 사용한 일반적인 액세스 제어 워크플로를 설명합니다.



1

대기열에 대한 Amazon SQS 정책을 작성합니다.

2

에 AWS정책을 업로드합니다. 이 AWS 서비스는 정책을 업로드하는 데 사용할 수 있는 API를 제공합니다. 예를 들어 Amazon SQS `SetQueueAttributes` 작업을 사용하여 특정 Amazon SQS 대기열에 대한 정책을 업로드합니다.

3

누군가가 Amazon SQS 대기열을 사용하기 위해 요청을 보냅니다.

4

Amazon SQS에서는 사용 가능한 모든 Amazon SQS 정책을 검사하고 어느 정책을 적용할 수 있는지 결정합니다.

5

Amazon SQS에서는 정책을 평가하고 요청자의 대기열 사용을 허용할지 여부를 결정합니다.

6

이를테면 정책 평가 결과에 따라 Amazon SQS가 `Access denied` 오류를 요청자에게 반환하거나 계속 요청을 처리합니다.

Amazon SQS 액세스 정책 언어 주요 개념

고유한 정책을 작성하려면 JSON과 많은 주요 개념에 대해 잘 알고 있어야 합니다.

허용

`allow`로 설정된 효과이(가) 있는 Statement의 결과.

작업

보안 주체가 일반적으로 AWS에 대한 요청을 수행할 권한이 있는 활동입니다.

Default-deny

허용 또는 Explicit-deny 설정이 없는 Statement의 결과입니다.

Condition

권한에 대한 제한 또는 세부 정보입니다. 일반적인 조건은 날짜 및 시간과 IP 주소와 관련되어 있습니다.

효과

정책의 Statement이(가) 평가 시간에 반환하는 결과. 정책 설명을 작성할 때 `deny` 또는 `allow` 값을 지정합니다. 정책 평가 시간에 Default-deny, 허용 및 Explicit-deny 결과가 나올 수 있습니다.

Explicit-deny

`deny`로 설정된 효과이(가) 있는 Statement의 결과.

평가

Amazon SQS에서 수신한 요청을 정책에 따라 거부할지 또는 허용할지 결정하는 데 이용하는 프로세스입니다.

발행자

리소스에 권한을 부여하기 위해 정책을 작성하는 사용자입니다. 발급자는 정의상 항상 리소스 소유자입니다. AWS Amazon SQS 사용자는 자신이 소유하지 않은 리소스에 대한 정책을 생성할 수 없습니다.

Key(키)

액세스 제한의 기준이 되는 구체적 특성입니다.

권한

Condition 및 Key(키)를 사용하여 리소스에 대한 액세스를 허용하거나 허용하지 않는 개념입니다.

정책

하나 이상의 설명에 대한 컨테이너 역할을 하는 문서입니다.

Amazon SQS는 사용자에게 리소스에 대한 액세스 권한을 부과할지 결정하는 정책을 사용합니다.

보안 주체

정책에서 권한을 수신하는 사용자입니다.

리소스

보안 주체 요청이 액세스하는 객체입니다.

Statement

단일 권한에 대한 공식적인 설명이며, 보다 폭넓은 정책 문서의 일환으로서 액세스 정책 언어로 작성됩니다.

요청자

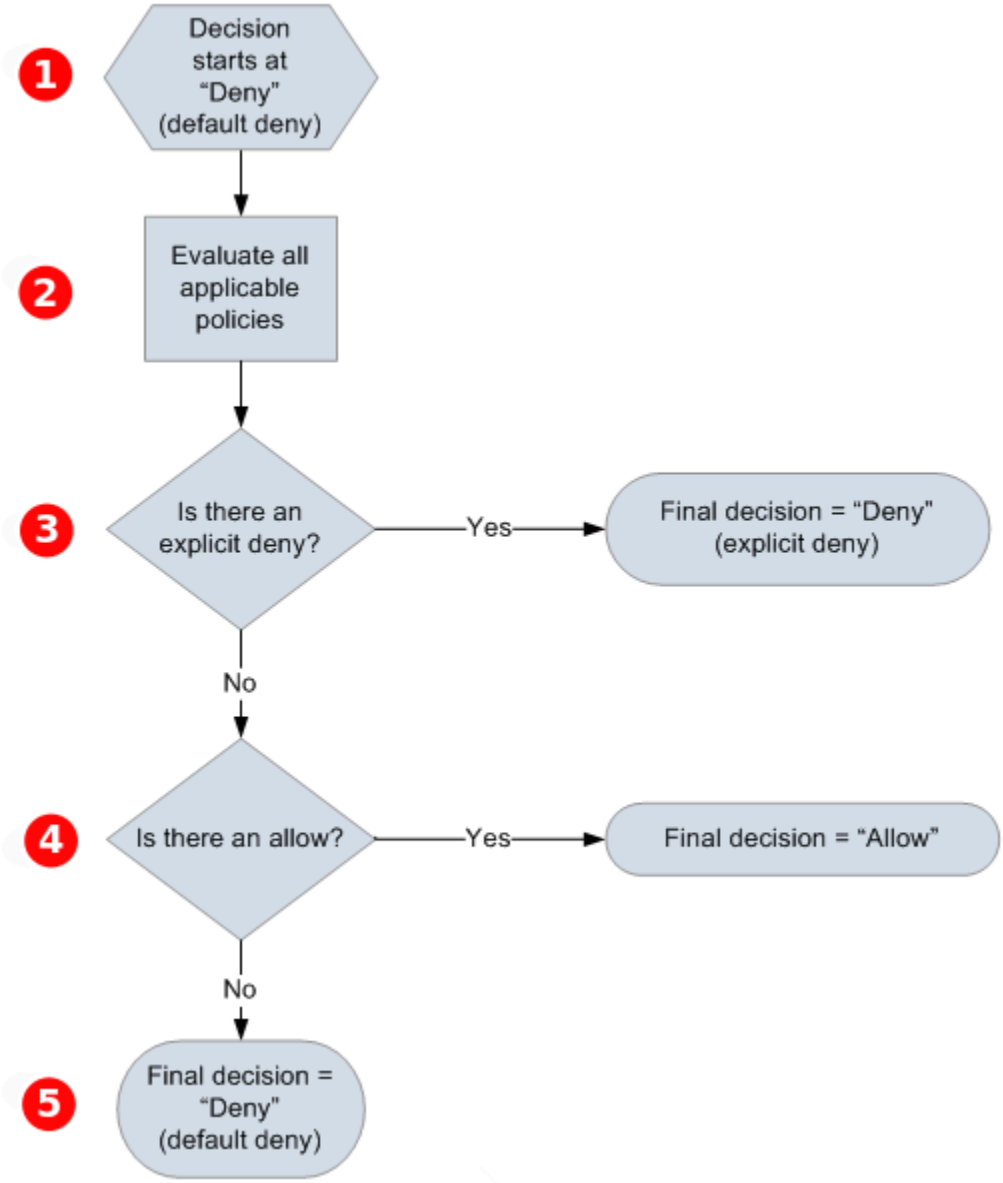
리소스에 액세스 요청을 전송하는 사용자입니다.

Amazon SQS 액세스 정책 언어 평가 로직

평가 시점에 Amazon SQS는 리소스 소유자 이외의 다른 사람의 요청을 허용할지 또는 거부할지 결정합니다. 평가 로직에서는 몇 가지 기본 규칙을 따릅니다.

- 기본적으로, 본인을 제외한 사람이 리소스를 사용하겠다고 요청하면 모두 거부됩니다.
- 허용는 모든 Default-deny를 재정의합니다.
- Explicit-deny는 모든 허용을 재정의합니다.
- 정책이 평가되는 순서는 중요하지 않습니다.

다음 그림에서는 Amazon SQS가 액세스 권한에 대한 결정을 평가하는 방식에 대해 자세히 설명합니다.



1

결정은 기본 거부에서 시작합니다.

2

적용 코드에서 (리소스, 보안 주체, 작업, 조건을 고려하여) 해당 요청에 적용 가능한 모든 정책을 평가합니다. 적용 코드에서 정책을 평가하는 순서는 중요하지 않습니다.

3

적용 코드는 해당 요청에 적용할 수 있는 명시적 거부 명령을 찾습니다. 하나라도 찾으면 이 적용 코드는 거부 결정을 반환하고 프로세스는 종료됩니다.

4

명시적 거부 명령이 없을 경우 적용 코드는 해당 요청에 적용할 수 있는 허용 명령을 찾습니다. 하나라도 찾으면 적용 코드는 허용 결정을 반환하고 프로세스가 완료됩니다(서비스에서는 계속 요청을 처리함).

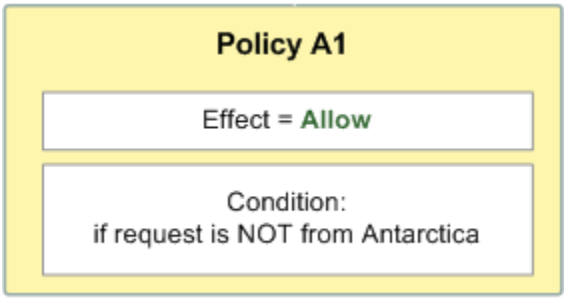
5

허용 명령이 없을 경우 최종 결정은 거부입니다. 명시적 거부 또는 허용이 없으므로 기본 거부로 간주됩니다.

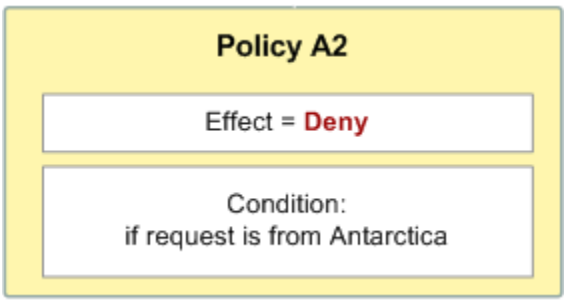
Amazon SQS 액세스 정책 언어의 명시적 거부와 기본 거부 간의 관계

Amazon SQS 정책이 요청에 직접 적용되지 않으면 요청 결과는 Default-deny입니다. 예를 들어, 한 사용자가 Amazon SQS 사용 권한을 요청했지만 해당 사용자에게 적용되는 유일한 정책에서 DynamoDB를 사용할 수 있으면, 요청 결과는 기본 거부가 됩니다.

설명의 조건이 충족되지 않을 경우에도 요청 결과는 기본 거부가 됩니다. 명령문의 모든 조건이 충족되면 요청은 정책의 효과 요소 값에 따라 허용 또는 Explicit-deny로 처리됩니다. 정책에서는 어떤 조건이 충족되지 않을 때 해야 할 일을 지정하지 않으므로, 이러한 경우 기본 결과는 기본 거부가 됩니다. 예를 들어 남극 대륙에서 오는 요청을 차단하고 싶다면 남극 대륙에서 오지 않은 요청만 허용하는 정책 A1을 작성하면 됩니다. 다음 다이어그램은 Amazon SQS 정책을 보여줍니다.

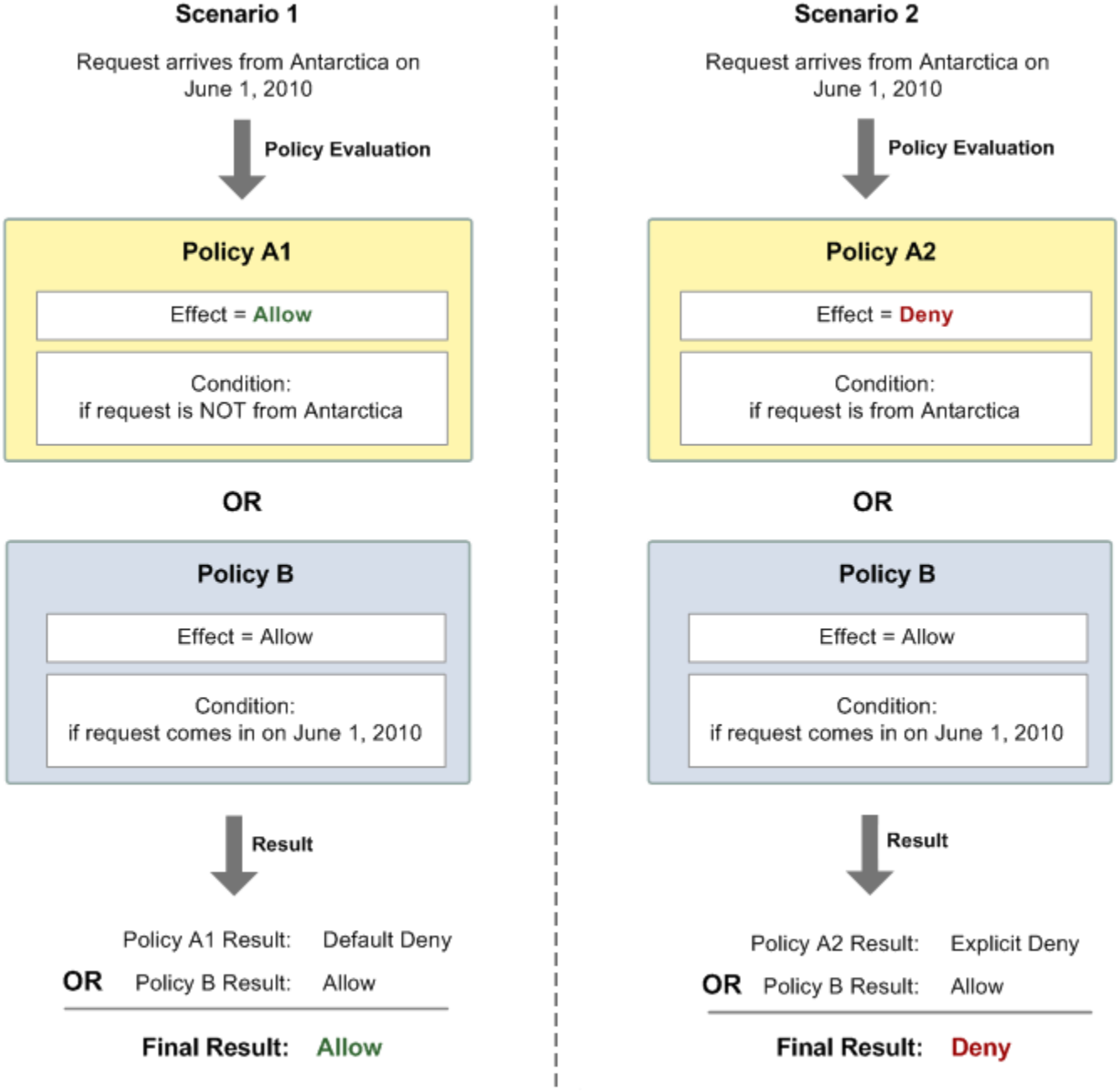


사용자가 미국에서 요청을 보낼 경우 (남극 대륙에서 온 요청이 아니므로) 조건을 충족하고, 요청 결과는 허용이 됩니다. 그러나 사용자가 남극 대륙에서 요청을 보낸다면 조건이 충족되지 않으므로 요청 결과는 기본 거부가 됩니다. 이 결과를 명시적 거부로 변경하려면 남극 대륙에서 온 요청을 명시적으로 거부하는 정책 A2를 작성합니다. 다음 다이어그램은 이 정책을 보여줍니다.



사용자가 남극 대륙에서 요청을 보낸다면 조건이 충족되므로 요청 결과는 명시적 거부가 됩니다.

허용은 기본 거부를 덮어쓸 수 있지만 명시적 거부는 덮어쓸 수 없기 때문에 전자와 후자를 구분하는 것이 중요합니다. 예를 들어, 요청이 2010년 6월 1일에 도착하면 정책 B는 요청을 허용합니다. 다이어그램은 이 정책을 정책 A1과 정책 A2 조합을 비교한 것입니다.



시나리오 1에서 이 정책은 2010년 6월 1일에 들어오는 요청을 허용하기 때문에 정책 A1의 결과는 기본 거부가 되고 정책 B의 결과는 허용이 됩니다. 정책 B의 허용은 정책 A1의 기본 거부보다 우선하므로 이 요청은 허용됩니다.

시나리오 2에서 정책 B2의 결과는 명시적 거부이고 정책 B의 결과는 허용입니다. 정책 A2의 명시적 거부는 정책 B의 허용보다 우선하고 이 요청은 거부됩니다.

Amazon SQS 사용자 지정 정책의 제한

크로스 계정 액세스

다음 작업에는 교차 계정 권한이 적용되지 않습니다.

- AddPermission
- CancelMessageMoveTask
- CreateQueue
- DeleteQueue
- ListMessageMoveTask
- ListQueues
- ListQueueTags
- RemovePermission
- SetQueueAttributes
- StartMessageMoveTask
- TagQueue
- UntagQueue

조건 키

현재 Amazon SQS는 다음과 같이 IAM에서 사용할 수 있는 조건 키의 제한된 하위 세트만 지원합니다. 자세한 정보는 Amazon SQS API 권한: 작업 및 리소스 참조을 참조하세요.

사용자 지정 Amazon SQS 액세스 정책 언어 예제

다음은 일반적인 Amazon SQS 액세스 정책의 예제입니다.

예제 1: 1개의 계정에 권한 부여

다음 Amazon SQS 정책 예제는 AWS 계정 4444555566666이 소유한 queue2에 대해 전송 및 수신할 수 있는 권한을 AWS 계정 111122223333에 부여합니다.

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

예제 2: 1개 이상의 계정에 권한 부여

다음 예제 Amazon SQS 정책은 특정 기간 동안 사용자 계정이 소유한 대기열에 대한 AWS 계정 액세스 권한을 한 번 이상 부여합니다. `AddPermission` 작업은 대기열에 대한 액세스 권한을 부여할 때 시간 제한 지정을 허용하지 않기 때문에 이 정책을 작성하고 `SetQueueAttributes` 작업을 사용하여 Amazon SQS에 업로드해야 합니다.

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

```
}
```

예제 3: Amazon EC2 인스턴스의 요청에 권한 부여

다음 예제 Amazon SQS 정책에서는 Amazon EC2 인스턴스에서 오는 요청에 대한 액세스를 부여합니다. 이 예제는 "예제 2: 1개 이상의 계정에 권한 부여" 예제를 보강한 것으로, 2009년 6월 30일 12시(정오)(UTC) 이전까지 액세스할 수 있고 액세스 IP 범위를 `203.0.113.0/24`로 제한합니다. `AddPermission` 작업은 대기열에 대한 액세스 권한을 부여할 때 IP 주소 제한 지정을 허용하지 않기 때문에 이 정책을 작성하고 `SetQueueAttributes` 작업을 사용하여 Amazon SQS에 업로드해야 합니다.

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

예제 4: 특정 계정에 대한 액세스 거부

다음 예제 Amazon SQS 정책은 대기열에 AWS 계정 대한 특정 액세스를 거부합니다. 이 예제는 "예제 1: 1개의 계정에 권한 부여" 예제를 기반으로 합니다. 이 예제는 지정된 항목에 대한 액세스를 거부

합니다. AWS 계정AddPermission 작업은 대기열에 대한 액세스 거부를 허용하지 않고 대기열에 대한 액세스 권한 부여만 허용하기 때문에 이 정책을 작성하고 SetQueueAttributes 작업을 사용하여 Amazon SQS에 업로드해야 합니다.

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

예제 5: VPC 엔드포인트에서 온 액세스가 아닌 경우 액세스 거부

다음 예제 Amazon SQS 정책은 queue1에 대한 액세스를 제한합니다. 111122223333은 VPC 엔드포인트 ID vpce-1a2b3c4d(aws:sourceVpce 조건을 사용하여 지정됨)에서만 SendMessage 및 ReceiveMessage 작업을 수행할 수 있습니다. 자세한 정보는 Amazon SQS용 Amazon Virtual Private Cloud 엔드포인트을 참조하세요.

> **Note**
>
> - aws:sourceVpce 조건은 VPC 엔드포인트 리소스의 ARN을 요구하지 않고 VPC 엔드포인트 ID만 요구합니다.
> - 두 번째 문의 모든 Amazon SQS 작업(sqs:*)을 거부하도록 다음 예제를 수정하여 특정 VPC 엔드포인트에 대한 모든 작업을 제한할 수 있습니다. 그러나 그러한 정책 문은 모든 작업(대기열 권한 수정을 위한 관리 작업 포함)을 정책에 정의된 특정 VPC 엔드포인트를 통해서만 해야 하도록 규정하게 될 것이므로 앞으로 사용자가 대기열 권한을 수정하지 못하게 될 가능성이 있습니다.

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

Amazon SQS에 대한 임시 보안 자격 증명 사용

자체 보안 자격 증명으로 사용자를 생성하는 것 외에도 IAM을 사용하면 모든 사용자에게 임시 보안 자격 증명을 부여하여 사용자가 AWS 서비스와 리소스에 액세스할 수 있도록 할 수 있습니다. AWS 계정이 있는 사용자를 관리할 수 있습니다. 또한 시스템에서 계정이 없는 사용자 AWS 계정 (페더레이션 사용자) 를 관리할 수 있습니다. 또한 AWS 리소스에 액세스하기 위해 만든 애플리케이션도 “사용자”로 간주될 수 있습니다.

Amazon SQS에 요청할 때 이 임시 보안 자격 증명을 사용할 수 있습니다. API 라이브러리는 요청 인증 시 이 자격 증명을 사용하여 필요한 서명 값을 계산합니다. 만료된 자격 증명으로 요청을 보내면 Amazon SQS는 요청을 거부합니다.

> **Note**
>
> 임시 자격 증명을 기반으로 정책을 설정할 수 없습니다.

사전 조건

1. IAM을 사용하여 임시 보안 자격 증명 생성:
   - 보안 토큰
   - 액세스 키 ID
   - 비밀 액세스 키
2. 문자열을 준비하여 임시 액세스 키 ID와 보안 토큰으로 서명합니다.
3. 자체 보안 액세스 키가 아닌 임시 보안 액세스 키를 사용하여 쿼리 API 요청에 서명합니다.

> **Note**
>
> 서명된 쿼리 API 요청을 제출할 때 자체 액세스 키 ID가 아닌 임시 액세스 키 ID를 사용하여 보안 토큰을 포함시킵니다. 임시 보안 자격 증명을 위한 IAM 지원에 대한 자세한 내용은 IAM 사용 설명서의 AWS 리소스에 대한 임시 액세스 권한 부여를 참조하십시오.

임시 보안 자격 증명을 사용하여 Amazon SQS 쿼리 API 작업을 호출하려면

1. 를 사용하여 임시 보안 토큰을 요청하세요. AWS Identity and Access Management자세한 내용은 IAM 사용 설명서의 IAM 사용자의 액세스를 활성화하기 위한 임시 보안 자격 증명 생성을 참조하세요.

   IAM은 보안 토큰, 액세스 키 ID 및 보안 액세스 키를 반환합니다.
2. 자체 액세스 키 ID가 아닌 임시 액세스 키 ID를 사용하여 쿼리를 준비하고 보안 토큰을 포함시킵니다. 자체 보안 액세스 키가 아닌 임시 보안 액세스 키를 사용하여 쿼리 요청에 서명합니다.
3. 임시 액세스 키 ID와 보안 토큰으로 서명된 쿼리 문자열을 제출합니다.

다음 예는 Amazon SQS 요청을 인증하기 위해 임시 보안 자격 증명을 사용하는 방법을 보여줍니다. *AUTHPARAMS*의 구조는 API 요청의 서명에 따라 달라집니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS API 요청 서명을 참조하십시오.

```
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
```

다음 예제에서는 임시 보안 자격 증명을 사용하여 SendMessageBatch 작업을 통해 두 개의 메시지를 전송합니다.

```
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
```

최소 권한 정책이 적용된 암호화된 Amazon SQS 대기열에 대한 액세스 관리

Amazon SQS를 사용하여 AWS Key Management Service (KMS)와 통합된 서버 측 암호화(SSE)를 사용하여 애플리케이션 간에 민감한 데이터를 교환할 수 있습니다. Amazon SQS 및 AWS KMS의 통합으로 Amazon SQS를 보호하는 키와 다른 리소스를 보호하는 키를 중앙에서 관리할 수 있습니다. AWS

여러 AWS 서비스가 Amazon SQS로 이벤트를 보내는 이벤트 소스 역할을 할 수 있습니다. 이벤트 소스가 암호화된 Amazon SQS 대기열에 액세스할 수 있게 하려면 고객 AWS KMS 관리 키로 대기열을

구성해야 합니다. 그런 다음 키 정책을 사용하여 서비스가 필요한 AWS KMS API 방법을 사용하도록 허용하십시오. 또한 이 서비스에는 대기열이 이벤트를 보낼 수 있도록 액세스를 인증할 수 있는 권한이 필요합니다. Amazon SQS 대기열 및 해당 데이터에 대한 액세스를 제어하는 데 사용할 수 있는 리소스 기반 정책인 Amazon SQS 정책을 사용하여 이를 달성할 수 있습니다.

다음 섹션에서는 Amazon SQS 정책 및 AWS KMS 키 정책을 통해 암호화된 Amazon SQS 대기열에 대한 액세스를 제어하는 방법에 대한 정보를 제공합니다. 이 가이드의 정책은 최소 권한을 획득하는 데 도움이 됩니다.

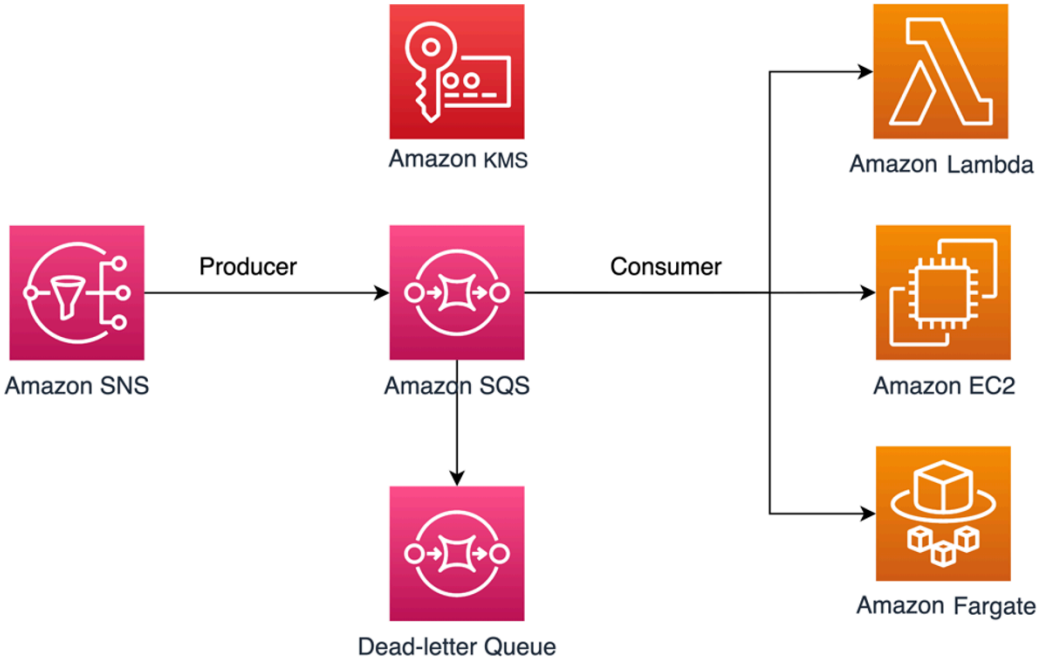
또한 이 가이드에서는 리소스 기반 정책이 `aws:SourceArn`, `aws:SourceAccount`, `aws:PrincipalOrgID` 글로벌 IAM 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 해결하는 방법도 설명합니다.

주제

- 개요
- Amazon SQS에 대한 최소 권한 키 정책
- DLQ(Dead Letter Queue)에 대한 Amazon SQS 정책 명령문
- 교차 서비스 혼동된 대리자 문제 방지
- IAM Access Analyzer를 사용하여 크로스 계정 액세스를 검토할 수 있습니다.

개요

이 주제에서는 일반적인 사용 사례를 통해 키 정책과 Amazon SQS 대기열 정책을 구축하는 방법을 설명합니다. 이 사용 사례는 다음 이미지에 나와 있습니다.



이 예제에서 메시지 생산자는 Amazon Simple Notification Service(SNS) 주제로, 메시지를 암호화된 Amazon SQS 대기열로 팬아웃하도록 구성되어 있습니다. 메시지 소비자는 AWS Lambda 함수, Amazon Elastic Compute Cloud(EC2) 인스턴스 또는 AWS Fargate 컨테이너와 같은 컴퓨팅 서비스입니다. 그런 다음 실패한 메시지를 DLQ(Dead Letter Queue)로 보내도록 Amazon SQS 대기열을 구성합니다. DLQ는 소비되지 않은 메시지를 구분하여 처리가 실패한 이유를 확인할 수 있으므로 이 방법은 애플리케이션 또는 메시징 시스템을 디버깅하는 데 유용합니다. 이 주제에 정의된 솔루션에서는 Lambda 함수와 같은 컴퓨팅 서비스를 사용하여 Amazon SQS 대기열에 저장된 메시지를 처리합니다. 메시지 소비자가 Virtual Private Cloud(VPC)에 있는 경우 이 안내서에 포함된 `DenyReceivingIfNotThroughVPCE` 정책 명령문을 통해 메시지 수신을 특정 VPC로 제한할 수 있습니다.

> **Note**
>
> 이 안내서에는 정책 명령문 형식의 필수 IAM 권한만 포함되어 있습니다. 정책을 구성하려면 Amazon SQS 정책 또는 AWS KMS 키 정책에 설명을 추가해야 합니다. 이 안내서에서는 Amazon SQS 대기열 또는 키를 생성하는 방법에 대한 지침을 제공하지 않습니다. AWS KMS 이러한 리소스를 생성하는 방법에 대한 지침은 Amazon SQS 대기열 생성 및 키 생성을 참조하세요.
>
> 이 안내서에 정의된 Amazon SQS 정책은 메시지를 동일하거나 다른 Amazon SQS 대기열로 직접 리드라이브하는 것을 지원하지 않습니다.

Amazon SQS에 대한 최소 권한 키 정책

이 섹션에서는 Amazon SQS 대기열을 암호화하는 데 사용하는 고객 관리 AWS KMS 키에 필요한 최소 권한 권한을 설명합니다. 이러한 권한을 사용하면 최소 권한을 구현하면서 의도한 엔터티로만 액세스를 제한할 수 있습니다. 키 정책은 다음과 같은 정책 명령문으로 구성되어야 하며, 이에 대해서는 아래에서 자세히 설명합니다.

- 키에 관리자 권한을 부여하십시오. AWS KMS
- 키 메타데이터에 대한 읽기 전용 액세스 권한 부여
- Amazon SNS에 대기열에 메시지를 게시할 수 있는 Amazon SNS KMS 권한 부여
- 소비자가 대기열에 있는 메시지를 해독할 수 있도록 허용

키에 관리자 권한을 부여하십시오. AWS KMS

키를 생성하려면 AWS KMS 키를 배포하는 데 사용하는 IAM 역할에 AWS KMS 관리자 권한을 제공해야 합니다 AWS KMS . 이러한 관리자 권한은 다음 `AllowKeyAdminPermissions` 정책 명령문에 정의되어 있습니다. 이 설명을 AWS KMS 키 정책에 추가할 때는 키 <admin-role ARN>배포나 AWS KMS 키 관리 또는 두 가지 모두에 사용되는 IAM 역할의 Amazon 리소스 이름 (ARN) 으로 바꿔야 합니다. AWS KMS 이는 배포 파이프라인의 IAM 역할이거나 AWS 조직에서 조직의 관리자 역할일 수 있습니다.

```
{
  "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**
>
> AWS KMS 키 정책에서는 Resource 요소의 값이 “이 AWS KMS 키”를 의미하는 값이어야 * 합니다. 별표 (*) 는 AWS KMS 키 정책이 연결된 키를 나타냅니다.

키 메타데이터에 대한 읽기 전용 액세스 권한 부여

다른 IAM 역할에 키 메타데이터에 대한 읽기 전용 액세스 권한을 부여하려면 키 정책에 AllowReadAccessToKeyMetaData 명령문을 추가합니다. 예를 들어, 다음 설명을 사용하면 감사 목적으로 계정의 모든 AWS KMS 키를 나열할 수 있습니다. 이 명령문은 AWS 루트 사용자에게 키 메타데이터에 대한 읽기 전용 액세스 권한을 부여합니다. 따라서 계정 내 모든 IAM 보안 주체는 자격 증명 기반 정책이 다음 명령문 kms:Describe*, kms:Get* 및 kms:List*에 나열된 권한을 가질 때 키 메타데이터에 액세스할 수 있습니다. *<account-ID>*를 자신의 정보로 바꿉니다.

```
{
  "Sid": "AllowReadAcesssToKeyMetaData",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::<accountID>:root"
    ]
  },
  "Action": [
    "kms:Describe*",
    "kms:Get*",
    "kms:List*"
  ],
  "Resource": "*"
}
```

Amazon SNS에 대기열에 메시지를 게시할 수 있는 Amazon SNS KMS 권한 부여

Amazon SNS 주제가 암호화된 Amazon SQS 대기열에 메시지를 게시하도록 허용하려면 키 정책에 `AllowSNSToSendToSQS` 정책 명령문을 추가합니다. 이 명령문은 Amazon SNS에 AWS KMS 키를 사용하여 Amazon SQS 대기열에 게시할 수 있는 권한을 부여합니다. *<account-ID>*를 자신의 정보로 바꿉니다.

> **Note**
>
> 이 `Condition` 설명서의 내용은 동일한 AWS 계정의 Amazon SNS 서비스에만 액세스를 제한합니다.

```
{
  "Sid": "AllowSNSToSendToSQS",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "sns.amazonaws.com"
    ]
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "<account-id>"
    }
  }
}
```

소비자가 대기열에 있는 메시지를 해독할 수 있도록 허용

다음 `AllowConsumersToReceiveFromTheQueue` 명령문은 암호화된 Amazon SQS 대기열에서 수신한 메시지를 해독하는 데 필요한 권한을 Amazon SQS 메시지 소비자에게 부여합니다. 정책 명령문을 첨부할 때 *<consumer's runtime role ARN>*을 메시지 소비자의 IAM 런타임 역할 ARN으로 대체하세요.

```
{
```

```
  "Sid": "AllowConsumersToReceiveFromTheQueue",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "<consumer's execution role ARN>"
    ]
  },
  "Action": [
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

Amazon SQS 정책에 대한 최소 권한

이 섹션에서는 이 안내서에서 다루는 사용 사례(예: Amazon SNS에서 Amazon SQS로)에 대한 최소 권한 Amazon SQS 대기열 정책을 안내합니다. 정의된 정책은 `Deny` 및 `Allow` 명령문을 모두 사용하여 의도하지 않은 액세스를 방지하도록 설계되었습니다. `Allow` 명령문은 의도한 엔터티 또는 엔터티에 대한 액세스 권한을 부여합니다. 이 `Deny` 명령문은 정책 조건 내에서 의도한 엔터티를 제외하면서 의도하지 않은 다른 엔터티가 Amazon SQS 대기열에 액세스하는 것을 방지합니다.

Amazon SQS 정책에는 다음과 같은 명령문이 포함되어 있으며, 이에 대해서는 아래에서 자세히 설명합니다.

- Amazon SQS 관리 권한 제한
- 지정된 조직의 Amazon SQS 대기열 작업을 제한합니다.
- 소비자에게 Amazon SQS 권한 부여
- 전송 중 암호화 적용
- 특정 Amazon SNS 주제로의 메시지 전송 제한
- (선택 사항) 메시지 수신을 특정 VPC 엔드포인트로 제한

Amazon SQS 관리 권한 제한

다음 `RestrictAdminQueueActions` 정책 명령문에서는 Amazon SQS 관리 권한을 대기열을 배포하거나 대기열을 관리하는 데 사용하는 IAM 역할 또는 두 가지 모두에 사용하는 역할로만 제한합니다. *<placeholder values>*을 자체 정보로 바꾸세요. Amazon SQS 대기열을 배포하는 데 사용되는 IAM 역할의 ARN과 Amazon SQS 관리 권한이 있어야 하는 관리자 역할의 ARN을 지정합니다.

```
{
```

```
  "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>"
      ]
    }
  }
}
```

지정된 조직의 Amazon SQS 대기열 작업을 제한합니다.

외부 액세스(AWS 조직 외부 엔터티의 액세스)로부터 Amazon SQS 리소스를 보호하려면 다음 명령문을 사용하세요. 이 명령문은 Amazon SQS 대기열 액세스를 `Condition`에 지정된 조직으로 제한합니다. *<SQS queue ARN>*을 Amazon SQS 대기열을 배포하는 데 사용된 IAM 역할의 ARN으로 바꾸고, *<org-id>*를 조직 ID로 바꾸어야 합니다.

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

소비자에게 Amazon SQS 권한 부여

Amazon SQS 대기열에서 메시지를 수신하려면 메시지 소비자에게 필요한 권한을 제공해야 합니다. 다음 정책 명령문은 지정한 소비자에게 Amazon SQS 대기열의 메시지를 소비하는 데 필요한 권한을 부여합니다. Amazon SQS 정책에 이 명령문을 추가할 때 *<consumer's IAM runtime role ARN>*을 소비자가 사용하는 IAM 런타임 역할의 ARN으로 바꾸고, *<SQS queue ARN>*을 Amazon SQS 대기열을 배포하는 데 사용되는 IAM 역할의 ARN으로 바꾸어야 합니다.

```
{
  "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>"
}
```

다른 엔터티가 Amazon SQS 대기열에서 메시지를 수신하지 못하도록 하려면 Amazon SQS 대기열 정책에 `DenyOtherConsumersFromReceiving` 명령문을 추가하세요. 이 명령문은 메시지 소비를 지정한 소비자로 제한하여 다른 소비자는 자격 증명 권한에 따라 액세스 권한이 있는 경우에도 액세스하지 못하도록 합니다. *<SQS queue ARN>*과 *<consumer's runtime role ARN>*을 자신의 정보로 바꾸어야 합니다.

```
{
  "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>"
    }
  }
}
```

전송 중 암호화 적용

다음 DenyUnsecureTransport 정책 명령문은 소비자와 생산자가 보안 채널(TLS 연결)을 사용하여 Amazon SQS 대기열에서 메시지를 보내고 받도록 강제합니다. *<SQS queue ARN>*을 Amazon SQS 대기열을 배포하는 데 사용된 IAM 역할의 ARN으로 바꾸어야 합니다.

```
{
  "Sid": "DenyUnsecureTransport",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:ReceiveMessage",
    "sqs:SendMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "Bool": {
      "aws:SecureTransport": "false"
    }
  }
}
```

특정 Amazon SNS 주제로의 메시지 전송 제한

다음 AllowSNSToSendToTheQueue 정책 명령문은 특정 Amazon SNS 주제가 Amazon SQS 대기열에 메시지를 전송하도록 허용합니다. *<SQS queue ARN>*을 Amazon SQS 대기열을 배포하는 데 사용된 IAM 역할의 ARN으로 바꾸고, *<SNS topic ARN>*을 Amazon SNS 주제 ARN으로 바꾸어야 합니다.

```
{
  "Sid": "AllowSNSToSendToTheQueue",
  "Effect": "Allow",
  "Principal": {
    "Service": "sns.amazonaws.com"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn": "<SNS topic ARN>"
    }
  }
}
```

다음 DenyAllProducersExceptSNSFromSending 정책 명령문은 다른 생산자가 대기열에 메시지를 보내는 것을 방지합니다. *<SQS queue ARN>* 및 *<SNS topic ARN>*을 자신의 정보로 바꿉니다.

```
{
  "Sid": "DenyAllProducersExceptSNSFromSending",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "ArnNotLike": {
      "aws:SourceArn": "<SNS topic ARN>"
    }
  }
```

```
}
```

(선택 사항) 메시지 수신을 특정 VPC 엔드포인트로 제한

메시지 수신을 특정 VPC 엔드포인트로만 제한하려면 Amazon SQS 대기열 정책에 다음 정책 명령문을 추가하세요. 이 명령문은 메시지가 원하는 VPC 엔드포인트에서 전송되지 않는 한 메시지 소비자가 대기열에서 메시지를 수신하지 못하도록 합니다. *<SQS queue ARN>*을 Amazon SQS 대기열을 배포하는 데 사용되는 IAM 역할의 ARN으로 바꾸고, *<vpce_id>*를 VPC 엔드포인트의 ID로 바꿉니다.

```
{
  "Sid": "DenyReceivingIfNotThroughVPCE",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "StringNotEquals": {
      "aws:sourceVpce": "<vpce id>"
    }
  }
}
```

DLQ(Dead Letter Queue)에 대한 Amazon SQS 정책 명령문

명령문 ID로 식별되는 다음 정책 명령문을 DLQ 액세스 정책에 추가합니다.

- `RestrictAdminQueueActions`
- `DenyQueueActionsOutsideOrg`
- `AllowConsumersToReceiveFromTheQueue`
- `DenyOtherConsumersFromReceiving`
- `DenyUnsecureTransport`

DLQ 액세스 정책에 이전 정책 명령문을 추가하는 것 외에도 다음 섹션에 설명된 대로 메시지 전송을 Amazon SQS 대기열로 제한하는 명령문도 추가해야 합니다.

Amazon SQS 대기열로 메시지 전송 제한

동일한 계정의 Amazon SQS 대기열로만 액세스를 제한하려면 DLQ 대기열 정책에 다음 `DenyAnyProducersExceptSQS` 정책 명령문을 추가합니다. 이 명령문은 기본 대기열을 생성하기 전에 DLQ를 배포해야 하므로 메시지 전송을 특정 대기열로 제한하지 않습니다. 따라서 DLQ를 생성할 때 Amazon SQS ARN을 알 수 없습니다. 하나의 Amazon SQS 대기열로만 액세스를 제한해야 하는 경우, Amazon SQS 소스 대기열의 ARN을 사용하여(아는 경우) `Condition`의 `aws:SourceArn`을 수정합니다.

```
{
  "Sid": "DenyAnyProducersExceptSQS",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS DLQ ARN>",
  "Condition": {
    "ArnNotLike": {
      "aws:SourceArn": "arn:aws:sqs:<region>:<account-id>:*"
    }
  }
}
```

Important

이 안내서에 정의된 Amazon SQS 대기열 정책은 `sqs:PurgeQueue` 작업을 특정 IAM 역할로 제한하지 않습니다. `sqs:PurgeQueue` 작업을 사용하면 Amazon SQS 대기열의 모든 메시지를 삭제할 수 있습니다. 또한 이 작업을 사용하여 Amazon SQS 대기열을 교체하지 않고도 메시지 형식을 변경할 수 있습니다. 애플리케이션을 디버깅할 때 Amazon SQS 대기열을 지워 오류가 있을 수 있는 메시지를 제거할 수 있습니다. 애플리케이션을 테스트할 때 Amazon SQS 대기열을 통해 대량의 메시지를 전송한 다음, 대기열을 제거하여 프로덕션에 들어가기 전에 새로 시작할 수 있습니다. 이 작업을 특정 역할로 제한하지 않는 이유는 Amazon SQS 대기열을 배포할 때 이 역할을 알지 못할 수 있기 때문입니다. 대기열을 제거하려면 이 권한을 역할의 ID 기반 정책에 추가해야 합니다.

교차 서비스 혼동된 대리자 문제 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 이를 방지하기 위해 계정 내 리소스에 대한 액세스 권한을 제3자(교차 계정이라고 함) 또는 기타 AWS 서비스 (크로스 서비스라고 함) 에 제공하는 경우 계정을 보호하는 데 도움이 되는 도구를 AWS 제공합니다. 이 섹션의 정책 명령문은 교차 서비스 혼동된 대리자 문제를 방지하는 데 도움이 될 수 있습니다.

교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 호출할 때 발생할 수 있습니다. 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이 문제를 방지하기 위해 이 게시물에 정의된 리소스 기반 정책은 `aws:SourceArn`, `aws:SourceAccount`, `aws:PrincipalOrgID` 글로벌 IAM 조건 컨텍스트 키를 사용합니다. 이렇게 하면 서비스가 가진 권한이 Organizations의 특정 리소스, 특정 계정 또는 특정 조직에 제한됩니다. AWS

IAM Access Analyzer를 사용하여 크로스 계정 액세스를 검토할 수 있습니다.

AWS IAM 액세스 분석기를 사용하여 Amazon SQS 대기열 정책 AWS KMS 및 키 정책을 검토하고 Amazon SQS 대기열 또는 AWS KMS 키가 외부 개체에 대한 액세스 권한을 부여하면 알림을 받을 수 있습니다. IAM Access Analyzer는 신뢰 영역 외부의 엔터티와 공유되는 조직 및 계정의 리소스를 식별하는 데 도움이 됩니다. 이 신뢰 영역은 IAM Access Analyzer를 활성화할 때 지정하는 Organizations 내의 AWS 계정 또는 AWS 조직일 수 있습니다.

IAM Access Analyzer는 로직 기반 추론을 사용하여 사용자 환경의 리소스 기반 정책을 분석함으로써 외부 보안 주체와 공유되는 리소스를 식별합니다. AWS 신뢰 영역 외부에서 공유되는 리소스의 각 인스턴스에 대해 Access Analyzer는 결과를 생성합니다. 결과에는 액세스에 대한 정보와 액세스 권한이 부여되는 외부 보안 주체에 대한 정보가 포함됩니다. 결과를 검토하여 액세스가 의도한 안전한 액세스인지 또는 액세스가 의도하지 않은 보안 위험인지 확인할 수 있습니다. 의도하지 않은 액세스가 발생한 경우 영향을 받는 정책을 검토하고 수정합니다. IAM Access Analyzer가 리소스에 대한 의도하지 않은 액세스를 식별하는 방법에 AWS 대한 자세한 내용은 이 블로그 게시물을 참조하십시오. AWS

IAM 액세스 분석기에 대한 자세한 내용은 AWS IAM 액세스 분석기 설명서를 참조하십시오.AWS

Amazon SQS API 권한: 작업 및 리소스 참조

액세스 제어를 설정하고 IAM 자격 증명에 연결할 수 있는 권한 정책을 작성할 때 다음 표를 참조로 사용할 수 있습니다. 목록에는 각 Amazon Simple Queue Service 작업, 작업을 수행할 권한을 부여할 수 있는 해당 작업, 권한을 부여할 수 있는 AWS 리소스가 포함되어 있습니다.

정책의 `Action` 필드에서 작업을 지정하고, 정책의 `Resource` 필드에서 리소스 값을 지정합니다. 작업을 지정하려면 `sqs:` 접두사 다음에 작업 이름을 사용합니다(예: `sqs:CreateQueue`).

현재 Amazon SQS는 IAM에서 사용할 수 있는 전역 조건 문맥 키를 지원합니다.

Amazon Simple Queue Service API 및 작업에 필요한 권한

AddPermission

작업: `sqs:AddPermission`

리소스: `arn:aws:sqs:region:account_id:queue_name`

ChangeMessage가시성

작업: `sqs:ChangeMessageVisibility`

리소스: `arn:aws:sqs:region:account_id:queue_name`

ChangeMessageVisibilityBatch

작업: `sqs:ChangeMessageVisibilityBatch`

리소스: `arn:aws:sqs:region:account_id:queue_name`

CreateQueue

작업: `sqs:CreateQueue`

리소스: `arn:aws:sqs:region:account_id:queue_name`

DeleteMessage

작업: `sqs:DeleteMessage`

리소스: `arn:aws:sqs:region:account_id:queue_name`

DeleteMessageBatch

작업: `sqs:DeleteMessageBatch`

리소스: `arn:aws:sqs:region:account_id:queue_name`

DeleteQueue

작업: `sqs:DeleteQueue`

리소스: `arn:aws:sqs:region:account_id:queue_name`

GetQueue속성

작업: `sqs:GetQueueAttributes`

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

GetQueueURL

작업: sqs:GetQueueUrl

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

ListDeadLetterSource대기열

작업: sqs:ListDeadLetterSourceQueues

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

ListQueues

작업: sqs:ListQueues

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

ListQueue태그

작업: sqs:ListQueueTags

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

PurgeQueue

작업: sqs:PurgeQueue

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

ReceiveMessage

작업: sqs:ReceiveMessage

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

RemovePermission

작업: sqs:RemovePermission

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

SendMessage및 SendMessageBatch

작업: sqs:SendMessage

리소스: arn:aws:sqs:*region*:*account_id*:*queue_name*

SetQueue속성

작업: `sqs:SetQueueAttributes`

리소스: `arn:aws:sqs:region:account_id:queue_name`

TagQueue

작업: `sqs:TagQueue`

리소스: `arn:aws:sqs:region:account_id:queue_name`

UntagQueue

작업: `sqs:UntagQueue`

리소스: `arn:aws:sqs:region:account_id:queue_name`

Amazon SQS의 로깅 및 모니터링

이 섹션에서는 API 호출을 캡처하는 데 사용하는 CloudTrail 방법, 대기열 활동 및 성능에 대한 통찰력을 얻기 위한 CloudWatch 지표 등 Amazon SQS의 로깅 및 모니터링 옵션에 대한 정보를 제공합니다.

주제

- AWS CloudTrail을 사용하여 Amazon SQS API 호출 로깅
- 를 사용하여 Amazon SQS 대기열을 모니터링합니다. CloudWatch

AWS CloudTrail을 사용하여 Amazon SQS API 호출 로깅

Amazon SQS는 사용자, 역할 또는 서비스로부터의 Amazon SQS 호출을 AWS CloudTrail 기록하기 위해 통합되어 있습니다. AWS CloudTrail Amazon SQS 표준 및 FIFO 대기열과 관련된 API 호출을 이벤트로 캡처합니다. 여기에는 Amazon SQS 콘솔 및 Amazon SQS API 호출을 통한 프로그래밍 방식으로 시작된 상호 작용이 포함됩니다.

주제

- 아마존 SQS 정보 입력 CloudTrail
- 의 관리 이벤트 CloudTrail
- 의 데이터 이벤트 CloudTrail
- 예: Amazon SQS의 CloudTrail 관리 이벤트

- 예: Amazon SQS용 CloudTrail 데이터 이벤트

아마존 SQS 정보 입력 CloudTrail

CloudTrail 계정을 생성할 때 기본적으로 활성화됩니다. AWS 지원되는 Amazon SQS 이벤트 활동이 발생하면 이벤트 기록의 다른 AWS 서비스 CloudTrail 이벤트와 함께 이벤트에 기록됩니다. AWS 계정의 최근 이벤트를 보고, 검색하고, 다운로드할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 기록과 함께 이벤트 보기를 참조하십시오.

대기열 관리 작업을 호출하는 Amazon SQS API는 관리 이벤트로 `AddPermission` 분류되며 기본적으로 로그인됩니다. CloudTrail Amazon SQS 대기열에서 수행되는 대용량 작업인 Amazon SQS API는 데이터 이벤트로 `SendMessage` 분류되며 옵트인 후 로깅됩니다. CloudTrail

CloudTrail 수집된 정보를 사용하여 Amazon SQS API에 대한 특정 요청, 요청자의 IP 주소 또는 ID, 요청 날짜 및 시간을 식별할 수 있습니다. CloudTrail 트레일을 구성하면 Amazon S3 버킷에 지속적으로 CloudTrail 이벤트를 전송하고 Amazon CloudWatch Logs 및 로 선택적으로 전송할 수 AWS EventBridge 있습니다. 트레일을 구성하지 않으면 CloudTrail 콘솔에서 이벤트의 관리 이벤트 기록만 볼 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서에서 추적 생성 개요를 참조하세요.

의 관리 이벤트 CloudTrail

Amazon SQS는 다음 API 작업을 관리 이벤트로 로깅합니다.

- `AddPermission`
- `CreateQueue`
- `CancelMessageMoveTask`
- `DeleteQueue`
- `ListMessageMoveTasks`
- `PurgeQueue`
- `RemovePermission`
- `SetQueueAttributes`
- `StartMessageMoveTask`
- `TagQueue`
- `UntagQueue`

다음 Amazon SQS API는 로깅이 지원되지 않습니다. CloudTrail

- GetQueueAttributes
- GetQueueUrl
- ListDeadLetterSourceQueues
- ListQueueTags
- ListQueues

의 데이터 이벤트 CloudTrail

데이터 이벤트는 Amazon SQS 대기열로 또는 Amazon SQS 대기열에서 Amazon SQS 메시지를 주고받는 등 리소스에서 또는 리소스에 대해 수행되는 리소스 작업에 대한 정보를 제공합니다. 데이터 이벤트는 기본적으로 기록되지 CloudTrail 않는 대용량 활동입니다. API를 사용하여 CloudTrail SQS 대기열에 대한 데이터 이벤트 API 작업 로깅을 활성화할 수 있습니다. 자세한 내용을 알아보려면 AWS CloudTrail 사용 설명서의 데이터 이벤트 로깅을 참조하십시오.

에서는 고급 이벤트 선택기를 사용하여 어떤 Amazon SQS API 활동을 기록하고 기록할지 결정할 수 있습니다. CloudTrail Amazon SQS 데이터 이벤트를 로깅하려면 리소스 유형 `AWS::SQS::Queue`를 포함해야 합니다. 이렇게 설정하면 `eventName` 필터를 사용하여 `SendMessage` 이벤트를 추적하는 등 기록할 특정 데이터 이벤트를 선택하여 로깅 기본 설정을 더 세분화할 수 있습니다. 자세한 내용을 알아보려면 AWS CloudTrail API 참조의 AdvancedEventSelector(을)를 참조하세요.

Amazon SQS 데이터 이벤트:

- SendMessage
- SendMessageBatch
- ReceiveMessage
- DeleteMessage
- DeleteMessageBatch
- ChangeMessageVisibility
- ChangeMessageVisibilityBatch

데이터 이벤트에는 추가 요금이 적용됩니다. 자세한 내용은 AWS CloudTrail 요금을 참조하십시오.

예: Amazon SQS의 CloudTrail 관리 이벤트

다음 예는 지원되는 API의 CloudTrail 로그 항목을 보여줍니다.

AddPermission

다음 예제는 AddPermission API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.


```
{
   "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

다음 예제는 CreateQueue API 호출의 CloudTrail 로그 항목을 보여줍니다.

```
{
   "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

다음 예제는 `DeleteQueue` API 호출의 CloudTrail 로그 항목을 보여줍니다.

```
{
   "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

다음 예제는 RemovePermission API 호출의 CloudTrail 로그 항목을 보여줍니다.

```
{
   "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

다음 예제는 에 대한 CloudTrail 로그 항목을 보여줍니다`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"
     }
```

```
    ]
  }
```

예: Amazon SQS용 CloudTrail 데이터 이벤트

다음은 Amazon SQS 데이터 CloudTrail 이벤트 API와 관련된 이벤트의 예입니다.

SendMessage

다음 예제는 에 대한 CloudTrail 데이터 이벤트를 보여줍니다. 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

다음 예제는 에 대한 CloudTrail 데이터 이벤트를 보여줍니다ReceiveMessage.


```
    {
  "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

다음 예제는 에 대한 CloudTrail 데이터 이벤트를 보여줍니다`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

다음 예제는 에 대한 CloudTrail 데이터 이벤트를 보여줍니다ChangeMessageVisibilityBatch.


```
    {
  "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"
  }
}
```

를 사용하여 Amazon SQS 대기열을 모니터링합니다. CloudWatch

Amazon SQS와 CloudWatch Amazon은 통합되어 있으므로 Amazon SQS 대기열에 대한 지표를 보고 분석하는 CloudWatch 데 사용할 수 있습니다. Amazon SQS 콘솔, 콘솔, CloudWatch 또는 API를 사용하여 AWS CLI대기열의 지표를 보고 분석할 수 있습니다. CloudWatch Amazon SQS 지표에 대한 CloudWatch 경보를 설정할 수도 있습니다.

CloudWatch Amazon SQS 대기열의 지표는 자동으로 수집되어 1분 간격으로 푸시됩니다. CloudWatch 이러한 지표는 활성 상태 가이드라인을 충족하는 모든 대기열에서 수집됩니다. CloudWatch CloudWatch대기열에 메시지가 포함되어 있거나 대기열에 액세스하는 작업이 있을 경우 대기열을 최대 6시간 동안 활성 상태로 간주합니다.

Amazon SQS 대기열이 6시간 이상 비활성 상태인 경우 Amazon SQS 서비스는 절전 모드로 간주되어 서비스에 대한 지표 전송을 중단합니다. CloudWatch 누락된 데이터 또는 0을 나타내는 데이터는 Amazon SQS 대기열이 비활성 상태였던 기간 동안 Amazon SQS CloudWatch 지표에서 시각화할 수 없습니다.

> **Note**
>
> - Amazon SQS 대기열은 사용자가 대기열에 대해 API를 호출하는 것이 승인되지 않아 요청이 실패할 때 활성화될 수 있습니다.
> - Amazon SQS 콘솔은 대기열 페이지가 열릴 때 `GetQueueAttributes` API 호출을 수행합니다. `GetQueueAttributes`API 요청은 대기열을 활성화합니다.
> - 대기열이 비활성 상태에서 활성화되면 CloudWatch 지표에서 최대 15분의 지연이 발생합니다.
> - 에서 보고된 Amazon SQS 지표에는 요금이 부과되지 않습니다. CloudWatch 이것은 Amazon SQS 서비스의 일부로 제공됩니다.
> - CloudWatch 지표는 표준 대기열과 FIFO 대기열 모두에서 지원됩니다.

주제

- Amazon SQS용 CloudWatch 지표에 액세스
- Amazon CloudWatch SQS 지표에 대한 경보 생성
- Amazon SQS에서 사용할 수 있는 CloudWatch 지표

Amazon SQS용 CloudWatch 지표에 액세스

Amazon SQS와 CloudWatch Amazon은 통합되어 있으므로 Amazon SQS 대기열에 대한 지표를 보고 분석하는 CloudWatch 데 사용할 수 있습니다. Amazon SQS 콘솔, 콘솔, CloudWatch 또는 API를 사용하여 AWS CLI대기열의 지표를 보고 분석할 수 있습니다. CloudWatch Amazon SQS 지표에 대한 CloudWatch 경보를 설정할 수도 있습니다.

Amazon SQS 콘솔

1. Amazon SQS 콘솔에 로그인합니다.
2. 대기열 목록에서 측정치에 액세스할 대기열에 해당하는 상자를 선택합니다. 최대 10개의 측정치를 표시할 수 있습니다.
3. 모니터링 탭을 선택합니다.

   [SQS metrics] 섹션에 다양한 그래프가 표시됩니다.
4. 특정 그래프가 나타내는 정보를 이해하려면, 원하는 그래프 옆에 있는

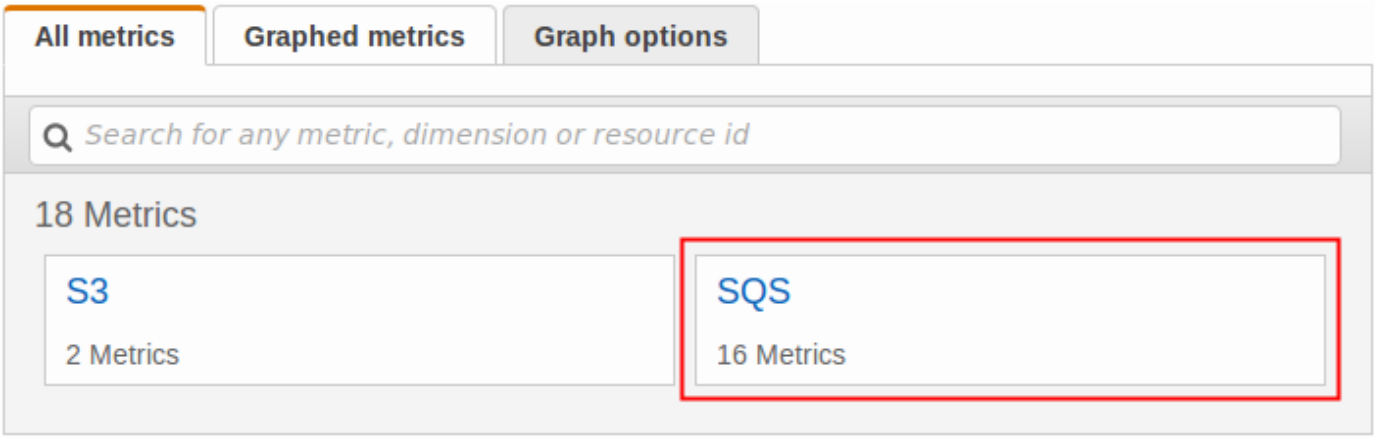
   

   위에 마우스 포인터를 놓거나 Amazon SQS에서 사용할 수 있는 CloudWatch 지표 단원을 참조하십시오.
5. 동시에 모든 그래프의 시간 범위를 변경하려면, [Time Range]에서 원하는 시간 범위(예: [Last Hour])를 선택합니다.
6. 개별 그래프의 추가 통계를 보려면 그래프를 선택합니다.
7. CloudWatch 모니터링 세부 정보 대화 상자에서 통계 (예: 합계) 를 선택합니다. 지원되는 통계 목록은 Amazon SQS에서 사용할 수 있는 CloudWatch 지표 단원을 참조하십시오.
8. 개별 그래프가 표시되는 시간 범위와 시간 간격을 변경하려면(예: 시간 범위를 지난 5분이 아닌 지난 24시간으로 표시), 그래프의 대화 상자를 계속 표시한 상대로 [Time Range]에서 원하는 시간 범위(예: [Last 24 Hours])를 선택합니다. [Period]에서 원하는 기간을 지정된 시간 범위 내로 선택합니다(예: [1 Hour]). 그래프 확인이 끝나면, [Close]를 선택합니다.
9. (선택 사항) 추가 CloudWatch 기능을 사용하려면 모니터링 탭에서 모든 CloudWatch 지표 보기를 선택한 다음 아마존 CloudWatch 콘솔 절차의 지침을 따르십시오.

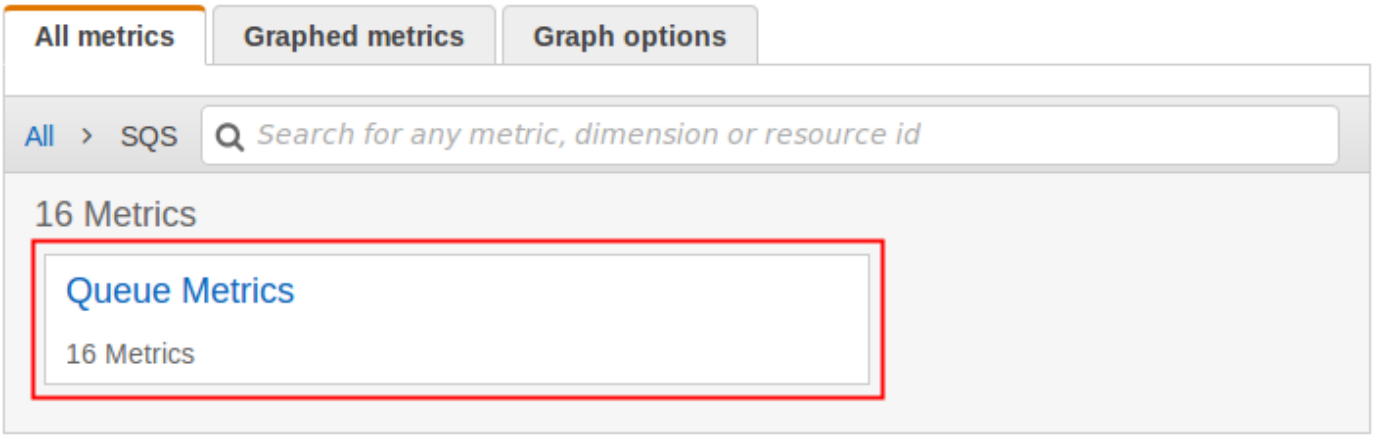
아마존 CloudWatch 콘솔

1. CloudWatch 콘솔에 로그인합니다.
2. 탐색 창에서 [Metrics]를 선택합니다.

3. [SQS] 측정치 네임스페이스를 선택합니다.

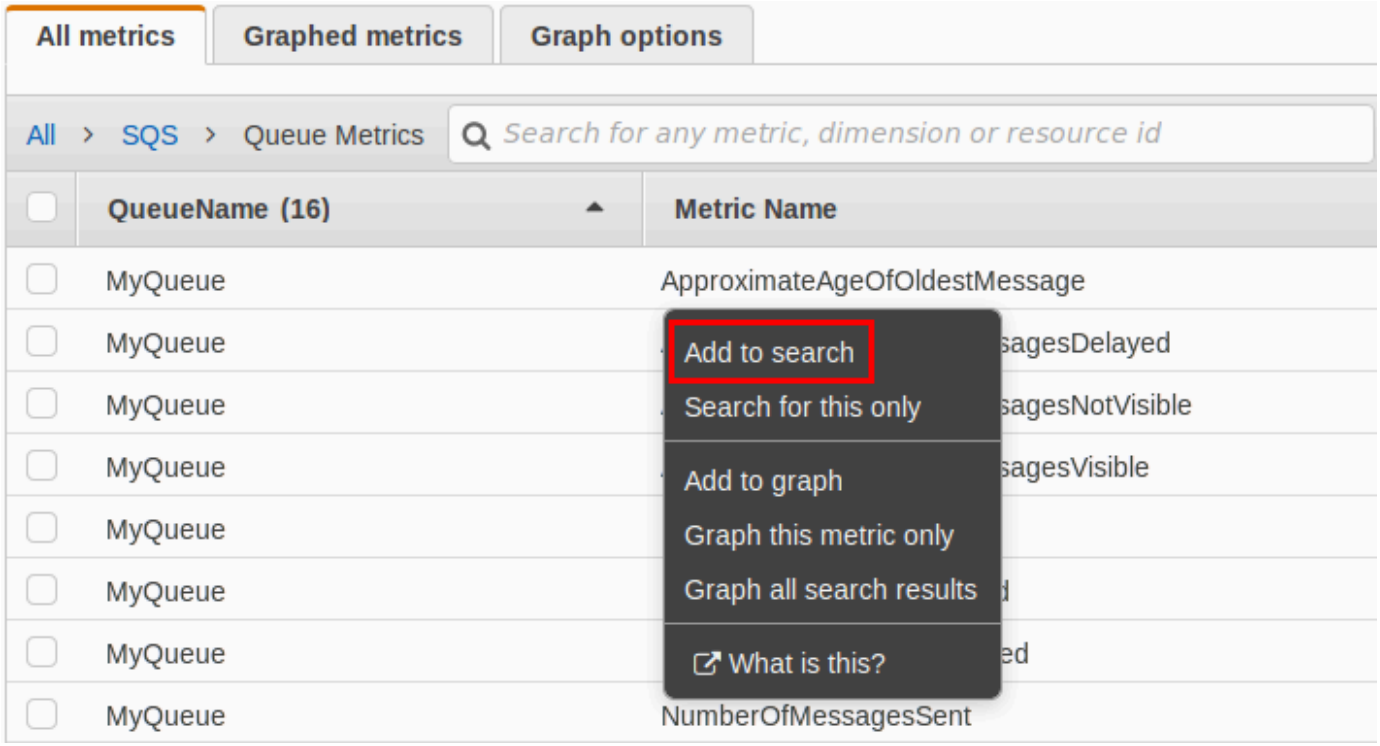


4. [Queue Metrics] 측정치 차원을 선택합니다.



5. 이제 Amazon SQS 지표를 검토할 수 있습니다.

   - 지표를 정렬하려면 열 머리글을 사용합니다.
   - 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다.
   - 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.



자세한 내용 및 추가 옵션은 Amazon 사용 설명서의 그래프 지표 및 Amazon CloudWatch 대시보드 사용을 참조하십시오. CloudWatch

AWS Command Line Interface

를 사용하여 Amazon SQS 지표에 AWS CLI액세스하려면 명령을 실행합니다. `get-metric-statistics`

자세한 내용은 Amazon CloudWatch 사용 설명서의 지표에 대한 통계 가져오기를 참조하십시오.

CloudWatch API

CloudWatch API를 사용하여 Amazon SQS 지표에 액세스하려면 작업을 사용하십시오. `GetMetricStatistics`

자세한 내용은 Amazon CloudWatch 사용 설명서의 지표에 대한 통계 가져오기를 참조하십시오.

Amazon CloudWatch SQS 지표에 대한 경보 생성

CloudWatch 지표 임계값을 기반으로 경보를 트리거할 수 있습니다. 예를 들어, `NumberOfMessagesSent` 지표에 대한 경보를 생성할 수 있습니다. 예를 들어 1시간 동안 `MyQueue`

대기열에 메시지 100개 이상이 전송된 경우 알림 이메일을 발송합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 생성을 참조하십시오.

1. https://console.aws.amazon.com/cloudwatch/ 에서 AWS Management Console 로그인하고 CloudWatch 콘솔을 엽니다.
2. 알람를 선택한 다음 알람 생성을 선택합니다.
3. 경보 생성 대화 상자의 지표 선택 섹션에서 지표 찾아보기, SQS를 선택합니다.
4. SQS > 대기열 지표의 QueueName경우 경보를 설정할 지표 이름 및 지표를 선택한 후 다음을 선택합니다. 사용 가능한 지표 목록은 Amazon SQS에서 사용할 수 있는 CloudWatch 지표 단원을 참조하십시오.

   다음 예에서는 `NumberOfMessagesSent` 대기열에 대한 `MyQueue` 지표 경보를 선택합니다. 전송된 메시지 수가 100개를 초과하면 경보가 트리거됩니다.
5. 경보 생성 대화 상자의 경보 정의 섹션에서 다음 작업을 수행합니다.

   a. 경보 임계값에서 경보의 이름 및 설명을 입력합니다.
   b. [is]를 [> 100]으로 설정합니다.
   c. 대상을 1 out of 1 datapoints(1 중 1 데이터 포인트)로 지정합니다.
   d. 경보 미리 보기에서 기간을 1시간으로 지정합니다.
   e. 통계를 표준, 합계로 지정합니다.
   f. 작업에서 이 경보가 발생할 경우 항상을 상태가 ALARM입니다.로 지정합니다.

   경보가 트리거될 때 알림을 CloudWatch 보내려면 기존 Amazon SNS 주제를 선택하거나 New list를 선택하고 이메일 주소를 쉼표로 구분하여 입력합니다.

   > **Note**
   >
   > 새 Amazon SNS 주제를 생성하는 경우 알림을 받기 전에 이메일 주소를 검증해야 합니다. 이메일 검증 전에 경보 상태가 변경되면 알림이 전송되지 않습니다.
6. 경보 생성을 선택합니다.

   경보가 생성됩니다.

Amazon SQS에서 사용할 수 있는 CloudWatch 지표

Amazon SQS는 에 다음과 같은 지표를 전송합니다. CloudWatch

> **Note**
>
> 표준 대기열의 경우, Amazon SQS의 분산 아키텍처로 인해 그 결과는 거의 비슷합니다. 대부분의 경우 그 수는 대기열에 있는 메시지의 실제 개수에 근접해야 합니다.
> FIFO 대기열의 경우, 그 결과는 정확합니다.

Amazon SQS 지표

`AWS/SQS` 네임스페이스에 포함된 지표는 다음과 같습니다.

지표	설명
`ApproximateAgeOfOldestMessage`	대기열에서 가장 오래된 비삭제 메시지의 대략적인 사용 기간. **Note** • 메시지를 3회 이상 수신했는데 처리하지 않았으면 메시지가 대기열의 맨 뒤로 이동되고 `ApproximateAgeOfOldestMessage` 지표는 3회 이하로 수신한 두 번째로 오래된 메시지를 가리킵니다. 이 작업은 대기열에 리드라이브 정책이 있는 경우에도 발생합니다. • 독약 메시지(여러 번 수신되었지만 삭제되지 않음)가 이 지표를 왜곡할 수 있으므로 독약 메시지가 성공적으로 소비될 때까지 독약 메시지의 사용 기간은 지표에 포함되지 않습니다. • 대기열에 리드라이브 정책이 있는 경우, 구성한 수신 최댓값이

지표	설명
	넘어가면 메시지가 데드-레터 대기열로 이동합니다. 메시지가 데드-레터 대기열로 이동하면 데드-레터 대기열의 `ApproximateAgeOfOldestMessage` 지표에 메시지를 보낸 시간이 아닌 데드-레터 대기열로 이동한 시간이 표시됩니다. • FIFO 대기열의 경우 FIFO 주문 보증이 위반되므로 메시지가 대기열 뒤쪽으로 이동하지 않습니다. DLQ가 구성되어 있는 경우 메시지가 대신 DLQ로 이동합니다. 그렇지 않으면 메시지 그룹이 성공적으로 삭제되거나 만료될 때까지 메시지 그룹이 차단됩니다. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 초 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨)

지표	설명
ApproximateNumberOfMessagesDelayed	지연되어 즉시 읽을 수 없는 대기열의 메시지 수. 이러한 경우는 대기열이 지연 대기열로 구성되거나 메시지가 지연 파라미터와 함께 전송되었을 때 발생할 수 있습니다. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 개 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨)
ApproximateNumberOfMessagesNotVisible	이동 중인 메시지의 수. 클라이언트에게 전송되었으나 아직 삭제되지 않았거나 가시성 창 말단에 이르지 않은 경우 이동 중인 것으로 간주됩니다. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 개 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨)

지표	설명
ApproximateNumberOfMessagesVisible	처리할 메시지 수. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 개 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨) 처리할 메시지 수에는 제한이 없지만 이 백로그에 보존 기간을 적용할 수 있습니다.
NumberOfEmptyReceives [1]	메시지를 반환하지 않은 ReceiveMessage API 호출의 수. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 개 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨)

지표	설명
NumberOfMessagesDeleted [1]	대기열에서 삭제된 메시지의 개수. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 개 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨) Amazon SQS는 중복 삭제를 포함하여 유효한 수신 핸들을 사용하는 모든 성공적인 삭제 작업에 대해 NumberOfMessagesDeleted 지표를 내보냅니다. 다음 시나리오로 인해 NumberOfMessagesDeleted 표지의 값이 예상보다 더 높아질 수 있습니다. • 동일한 메시지에 속한 서로 다른 수신 핸들에서 DeleteMessage 작업 호출: 해당 메시지가 제한 시간 초과 만료 전에 처리되지 않으면 이 메시지를 처리하고 다시 삭제할 수 있는 다른 소비자가 사용할 수 있게 되므로 NumberOfMessagesDeleted 지표의 값이 증가합니다. • 동일한 수신 핸들에서 DeleteMessage 작업 호출: 해당 메시지가 처리된 후 삭제되었으나 동일한 수신 핸들을 사용하여 DeleteMessage 작업을 다시 호출하면 성공 상태가 반환되어 NumberOfMessagesDeleted 지표의 값이 증가합니다.

지표	설명
NumberOfMessagesReceived [1]	ReceiveMessage 작업에 대한 호출로 반환된 메시지의 수. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 개 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨)
NumberOfMessagesSent [1]	대기열에 추가된 메시지의 수. 메시지를 배달 못한 편지 대기열에 수동으로 전송하는 경우, 이것은 NumberOfMessagesSent 측정치로 캡처됩니다. 그러나 처리 시도 실패로 인해 데드 레터 대기열로 메시지가 전송되는 경우 이 지표는 메시지를 캡처하지 않습니다. 따라서 NumberOfMessagesSent 및 NumberOfMessagesReceived 값이 다를 수 있습니다. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 개 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨)

지표	설명
SentMessageSize [1]	대기열에 추가된 메시지의 크기. 보고 기준: 대기열이 활성화된 경우 음수가 아닌 값이 보고됩니다. 단위: 바이트 유효한 통계: Average, Minimum, Maximum, Sum, Data Samples(Amazon SQS 콘솔에서는 Sample Count로 표시됨) **Note** SentMessageSize 하나 이상의 메시지가 해당 대기열로 전송될 때까지 CloudWatch 콘솔에 사용 가능한 지표로 표시되지 않습니다.

[1] 이러한 지표는 서비스 관점에서 계산되며 재시도를 포함할 수 있습니다. 이러한 지표의 절대값에 의존하거나 측정치를 사용하여 현재 대기열 상태를 추정하지 마세요.

Amazon SQS 지표 차원

Amazon SQS가 전송하는 CloudWatch 유일한 차원은 입니다. QueueName 이는 사용 가능한 모든 통계가 QueueName(대기열 이름)으로 필터링됨을 뜻합니다.

Amazon SQS에 대한 규정 준수 확인

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램별 범위 내 규정 준수AWS 서비스 프로그램별 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스AWS — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.
- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

Amazon SQS의 복원성

AWS 글로벌 인프라는 AWS 지역 및 가용 영역을 중심으로 구축됩니다. AWS 지역은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며, 이러한 가용 영역은 지연 시간이 짧고 처리량이 높으며 이중

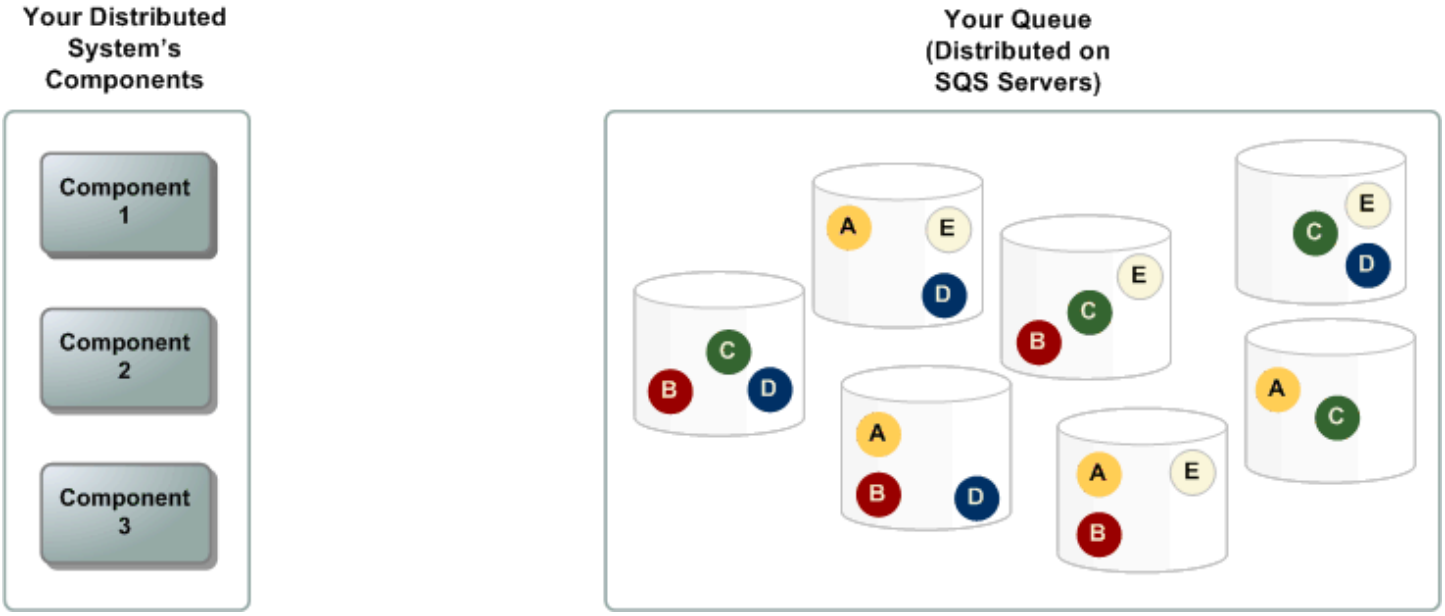
화가 심한 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다. AWS 지역 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하십시오.

Amazon SQS는 AWS 글로벌 인프라 외에도 분산 대기열을 제공합니다.

분산 대기열

분산 메시징 시스템에는 세 가지 주요 부분, 즉 분산 시스템의 구성 요소, 대기열(Amazon SQS 서버에 분산됨), 대기열의 메시지가 있습니다.

다음 시나리오에서 시스템에는 여러 생산자(메시지를 대기열로 전송하는 구성 요소) 및 소비자(대기열의 메시지를 수신하는 구성 요소)가 있습니다. 대기열(메시지 A~E 유지)은 여러 Amazon SQS 서버에서 메시지를 중복으로 저장합니다.



Amazon SQS의 인프라 보안

Amazon SQS는 관리형 서비스로서 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차에 따라 보호됩니다.

AWS 게시된 API 작업을 사용하여 네트워크를 통해 Amazon SQS에 액세스할 수 있습니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다.

IAM 주체와 연결되어 있는 액세스 키 ID 및 보안 액세스 키를 사용해 요청에 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용해 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

이러한 API 작업은 모든 네트워크 위치에서 호출할 수 있지만, Amazon SQS는 원본 IP 주소를 기반으로 하는 제한을 포함할 수 있는 리소스 기반 액세스 정책을 지원합니다. Amazon SQS 정책을 사용하여 특정 Amazon VPC 엔드포인트 또는 특정 VPC에서 액세스를 제어할 수도 있습니다. 이렇게 하면 지정된 Amazon SQS 대기열에 대한 네트워크 액세스를 네트워크 내의 특정 VPC에서만 효과적으로 분리할 수 있습니다. AWS 자세한 내용은 예제 5: VPC 엔드포인트에서 온 액세스가 아닌 경우 액세스 거부을(를) 참조하세요.

Amazon SQS 보안 모범 사례

AWS 는 Amazon SQS의 다양한 보안 기능을 제공하며, 사용자는 자체 보안 정책의 맥락에서 검토해야 합니다. 다음은 Amazon SQS에 대한 예방 보안 모범 사례입니다.

> **Note**
>
> 제공된 구체적인 구현 지침은 일반적인 사용 사례 및 구현을 위한 것입니다. 특정 사용 사례, 아키텍처 및 위협 모델의 컨텍스트에서 이러한 모범 사례를 살펴보는 것이 좋습니다.

주제

- 대기열에 대한 공개적 액세스 방지
- 최소 권한 액세스 구현
- Amazon SQS 액세스가 필요한 애플리케이션 및 AWS 서비스에 IAM 역할 사용
- 서버 측 암호화 구현
- 전송 중인 데이터의 암호화 강제 시행
- VPC 엔드포인트를 사용한 Amazon SQS 액세스 고려

대기열에 대한 공개적 액세스 방지

인터넷에 연결된 모든 사용자가 Amazon SQS 대기열을 읽거나 쓸 수 있도록 명시적으로 요구하는 경우가 아니라면 대기열에 공개적으로 액세스 (전 세계 모든 사용자 또는 인증된 사용자가 액세스 가능) 할 수 없도록 해야 합니다. AWS

- `Principal`이 `""`로 설정된 정책을 생성하지 마세요.

- 와일드카드(*)를 사용하지 마세요. 대신에 특정 사용자 또는 사용자의 이름을 지정합니다.

최소 권한 액세스 구현

권한을 부여할 때 권한을 받는 사용자, 권한이 있는 대기열 및 이러한 대기열에 허용하려는 특정 API 작업을 결정합니다. 최소 권한 구현은 보안 위험을 줄이고 오류 또는 악의적인 의도에 따른 영향을 줄이는 데 중요합니다.

최소 권한 부여에 대한 표준 보안 권고 사항을 따릅니다. 즉, 특정 작업을 수행하는 데 필요한 권한만 부여합니다. 보안 정책 조합을 사용하여 이를 구현할 수 있습니다.

Amazon SQS는 세 가지 유형의 사용자 계정 액세스를 필요로 하는 생산자-소비자 모델을 사용합니다.

- 관리자 - 대기열을 생성, 수정 및 삭제할 수 있습니다. 또한 관리자는 대기열 정책도 제어합니다.
- 생산자 - 대기열에 메시지를 보낼 수 있습니다.
- 소비자 - 대기열에서 보낸 메시지를 수신하고 삭제할 수 있습니다.

자세한 내용은 다음 단원을 참조하세요.

- Amazon SQS의 Identity and Access Management
- Amazon SQS API 권한: 작업 및 리소스 참조
- Amazon SQS 액세스 정책 언어와 함께 사용자 지정 정책 사용

Amazon SQS 액세스가 필요한 애플리케이션 및 AWS 서비스에 IAM 역할 사용

Amazon EC2와 같은 애플리케이션 또는 AWS 서비스가 Amazon SQS 대기열에 액세스하려면 API 요청에 AWS 유효한 자격 증명을 사용해야 합니다. AWS 이러한 자격 증명은 자동으로 교체되지 않으므로 애플리케이션 또는 EC2 인스턴스에 AWS 자격 증명을 직접 저장해서는 안 됩니다.

IAM 역할을 사용하여 Amazon SQS에 액세스해야 하는 애플리케이션이나 서비스의 임시 자격 증명을 관리해야 합니다. 역할을 사용하면 장기 자격 증명 (예: 사용자 이름, 암호, 액세스 키) 을 EC2 인스턴스 또는 AWS 서비스 (예:) 에 배포하지 않아도 됩니다. AWS Lambda대신 역할은 애플리케이션이 다른 AWS 리소스를 호출할 때 사용할 수 있는 임시 권한을 제공합니다.

자세한 정보는 IAM 사용 설명서의 IAM 역할 및 역할에 대한 일반 시나리오: 사용자, 애플리케이션 및 서비스를 참조하세요.

서버 측 암호화 구현

데이터 유출 문제를 완화하려면 메시지를 저장하는 위치와 다른 위치에 저장된 키를 통해 메시지를 암호화하는 유휴 시 데이터 암호화를 사용합니다. 서버 측 암호화(SSE)는 저장된 데이터 암호화 기능을 제공합니다. Amazon SQS는 데이터를 저장할 때 메시지 수준에서 데이터를 암호화하고 사용자가 액세스할 때 메시지를 복호화합니다. SSE는 관리되는 키를 사용합니다. AWS Key Management Service 요청을 인증하기만 하면 액세스 권한을 갖게 되며, 대기열의 암호화 여부와 관계없이 액세스 방식에는 차이가 없습니다.

자세한 정보는 Amazon SQS의 저장 중 암호화 및 아마존 SQS 키 관리에서 확인하세요.

전송 중인 데이터의 암호화 강제 시행

HTTPS (TLS) 가 없으면 네트워크 기반 공격자는 다음과 같은 공격을 사용하여 네트워크 트래픽을 도청하거나 조작할 수 있습니다. man-in-the-middle 요청에 SSL을 강제로 적용하도록 하려면 대기열 정책의 `aws:SecureTransport` 조건을 사용하여 HTTPS(TLS)를 통한 암호화된 연결만 허용합니다.

VPC 엔드포인트를 사용한 Amazon SQS 액세스 고려

상호 작용할 수 있어야 하지만 인터넷에 절대 노출되지 않아야 하는 대기열이 있는 경우, VPC 엔드포인트를 사용하여 특정 VPC 내의 호스트에 대한 액세스만 대기시킵니다. 대기열 정책을 사용하여 특정 엔드포인트 또는 특정 Amazon VPC에서 대기열에 대한 액세스를 제어할 수 있습니다.

Amazon SQS VPC 엔드포인트는 메시지에 대한 액세스를 제어하는 두 가지 방법을 제공합니다.

- 사용자는 특정 VPC 종단점을 통해 허용되는 요청, 사용자 또는 그룹을 제어할 수 있습니다.
- 대기열 정책을 사용하여 대기열에 액세스할 수 있는 VPC 또는 VPC 엔드포인트를 제어할 수 있습니다.

자세한 내용은 Amazon SQS용 Amazon Virtual Private Cloud 엔드포인트 및 Amazon SQS용 Amazon VPC 엔드포인트 정책 생성 단원을 참조하세요.

Amazon SQS 관련 리소스

다음 표에는 이 서비스를 사용할 때 유용하게 참조할 수 있는 관련 리소스가 나열되어 있습니다.

Resource	설명
Amazon Simple Queue Service API 참조	작업, 파라미터, 데이터 유형 등의 설명 및 반환하는 오류 목록.
AWS CLI 명령 참조의 Amazon SQS	대기열 작업에 사용할 수 있는 AWS CLI 명령에 대한 설명
리전 및 엔드포인트	Amazon SQS 리전 및 엔드포인트에 대한 정보
제품 페이지	Amazon SQS에 대한 정보를 제공하는 기본 웹 페이지입니다.
토론 포럼	Amazon SQS와 관련된 기술적 질문에 대해 토론할 수 있는 개발자를 위한 커뮤니티 기반 포럼입니다.
AWS 프리미엄 지원 정보	인프라 서비스에서 애플리케이션을 빌드하고 실행하는 데 도움이 되는 일대일 빠른 응답 지원 채널인 AWS Premium Support에 AWS 대한 정보를 제공하는 기본 웹 페이지입니다.

문서 기록

아래 표에는 2019년 1월 이후 Amazon Simple Queue Service 개발자 안내서의 중요한 변경 사항이 설명되어 있습니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

서비스 기능이 서비스를 사용할 수 있는 AWS 지역에 점진적으로 배포되는 경우가 있습니다. 이 문서는 첫 번째 릴리스에 대해서만 업데이트됩니다. 리전 가용성에 대한 정보를 제공하거나 후속 리전 롤아웃을 발표하지 않습니다. 서비스 기능의 지역 가용성에 대한 자세한 내용과 업데이트에 대한 알림을 구독하려면 무엇이 새로워졌나요? 를 AWS 참조하십시오. .

변경 사항	설명	날짜
AWS JSON 프로토콜	AWS JSON 프로토콜을 사용하여 API 요청을 생성합니다.	2023년 7월 27일
Amazon SQS의 AWS 관리형 정책 및 해당 정책의 업데이트를 설명하는 새 섹션	Amazon SQS는 특정 소스 대기열 아래에 가장 최근의 메시지 이동 작업(최대 10개)을 나열할 수 있는 새 작업을 추가했습니다. 이 작업은 `ListMessageMoveTasks` API 작업과 연결되어 있습니다.	2023년 6월 7일
API를 사용하여 배달 못한 편지 대기열 리드라이브	Amazon SQS API를 사용하여 배달 못한 편지 대기열 리드라이브를 구성합니다.	2023년 6월 7일
Amazon SQS에 대한 ABAC	유연하고 확장 가능한 액세스 권한을 위해 대기열 태그를 사용하는 ABAC(속성 기반 액세스 제어)입니다.	2022년 11월 10일
FIFO 높은 처리량 한도 증가	FIFO 높은 처리량 문서 최적화와 더불어 상용 리전의 FIFO 높은 처리량 모드에 대한 기본 할당량이 증가했습니다.	2022년 10월 20일

기본 서버 측 암호화(SSE)를 사용할 수 있음	기본적으로 SQS 소유 암호화(SSE-SQS)를 사용하는 서버 측 암호화(SSE)를 사용할 수 있습니다.	2022년 9월 26일
Amazon SQS 혼동된 대리인 보호 지원을 사용할 수 있음	혼동된 대리인 보호를 사용하면 요청에 새 헤더를 지정할 수 있으며, 이러한 헤더는 Amazon SQS 관리형 SSE를 사용할 때 KMS 정책의 조건과 비교하여 검토됩니다.	2021년 12월 29일
관리형 SSE를 사용할 수 있음	Amazon SQS 관리형 SSE(SSE-SQS)는 Amazon SQS 소유 암호화 키를 사용하여 메시지 대기열을 통해 전송되는 민감한 데이터를 보호하는 관리형 서버 측 암호화입니다.	2021년 11월 23일
배달 못한 편지 대기열 리드라이브를 사용할 수 있음	Amazon SQS는 표준 대기열에 대해 배달 못한 편지 대기열 리드라이브를 지원합니다.	2021년 11월 10일
FIFO 대기열의 메시지에 대해 높은 처리량을 사용할 수 있음	Amazon SQS FIFO 대기열의 높은 처리량은 FIFO 대기열의 메시지에 대해 더 많은 초당 트랜잭션(TPS) 수를 제공합니다. 처리량 할당량에 대한 자세한 내용은 메시지 관련 할당량을 참조하세요.	2021년 5월 27일

평가판 릴리스에서 FIFO 대기열의 메시지에 대해 높은 처리량을 사용할 수 있음	Amazon SQS FIFO 대기열의 높은 처리량은 평가판 릴리스에서 제공되며 변경될 수 있습니다. 이 기능은 FIFO 대기열에 있는 메시지에 대해 더 많은 초당 트랜잭션(TPS) 수를 제공합니다. 처리량 할당량에 대한 자세한 내용은 메시지 관련 할당량을 참조하세요.	2020년 12월 17일
새로운 Amazon SQS 콘솔 설계	개발 및 프로덕션 워크플로를 단순화하기 위해 Amazon SQS 콘솔에 새로운 사용자 경험이 추가되었습니다.	2020년 7월 8일
Amazon SQS는 목록 대기열에 대한 페이지 매김을 지원하고 listDeadLetter SourceQueues	ListQueues 또는 목록 요청에서 반환할 최대 결과 수를 지정할 수 있습니다. DeadLetterSourceQueues	2020년 6월 22일
Amazon SQS는 AWS GovCloud (미국) 지역을 제외한 모든 AWS 지역에서 1분 Amazon CloudWatch 메트릭을 지원합니다.	Amazon SQS의 1분 CloudWatch 지표는 지역을 제외한 AWS GovCloud (US) 모든 지역에서 사용할 수 있습니다.	2020년 1월 9일
Amazon SQS는 1분 지표를 지원합니다. CloudWatch	Amazon SQS의 1분 CloudWatch 지표는 현재 미국 동부 (오하이오), 유럽 (아일랜드), 유럽 (스톡홀름), 아시아 태평양 (도쿄) 지역에서만 사용할 수 있습니다.	2019년 11월 25일
AWS Lambda Amazon SQS FIFO 대기열에 대한 트리거를 사용할 수 있습니다.	FIFO 대기열에 도착하는 메시지를 Lambda 함수 트리거로 구성할 수 있습니다.	2019년 11월 25일

중국 리전에서 Amazon SQS에 대한 서버 측 암호화(SSE)를 사용할 수 있음	Amazon SQS에 대한 SSE를 중국 리전에서 사용할 수 있습니다.	2019년 11월 13일
중동(바레인) 리전에서 FIFO 대기열을 사용할 수 있음	중동(바레인) 리전에서 FIFO 대기열을 사용할 수 있습니다.	2019년 10월 10일
Amazon SQS용 Amazon VPC(가상 사설 클라우드) 엔드포인트는 AWS GovCloud (미국 동부) 및 (미국 서부) 지역에서 사용할 수 있습니다. AWS GovCloud	(미국 동부) 및 (미국 서부) 지역의 AWS GovCloud Amazon VPC에서 Amazon SQS 대기열로 메시지를 보낼 수 있습니다. AWS GovCloud	2019년 9월 5일
Amazon SQS에서는 메시지 시스템 속성을 사용하여 AWS X-Ray 대기열 문제를 해결할 수 있습니다.	X-Ray를 사용하여 Amazon SQS 대기열을 통해 전달되는 메시지의 문제를 해결할 수 있습니다. 이 릴리스는 `SendMessage` 및 `SendMessageBatch` API 작업에 대한 `MessageSystemAttribute` 요청 파라미터(이 파라미터를 사용하면 Amazon SQS를 통해 X-Ray 추적 헤더를 전송할 수 있음), `ReceiveMessage` API 작업에 대한 `AWSTraceHeader` 속성 및 `MessageSystemAttributeValue` 데이터 형식을 추가합니다.	2019년 8월 28일

생성 시 Amazon SQS 대기열에 태그를 지정할 수 있음	단일 Amazon SQS API 호출, AWS SDK 함수 또는 AWS Command Line Interface (AWS CLI) 명령을 사용하여 대기열을 생성하고 태그를 동시에 지정할 수 있습니다. 또한 Amazon SQS는 `aws:TagKeys` 및 `aws:RequestTag` AWS Identity and Access Management (IAM) 키를 지원합니다.	2019년 8월 22일
Amazon SQS에 대한 임시 대기열 클라이언트를 지금 사용할 수 있음	임시 대기열을 사용하면 request-response와 같은 공통 메시지 패턴을 사용할 때 개발 시간 및 배포 비용을 줄일 수 있습니다. 임시 대기열 클라이언트를 사용하여 대량고속처리가 가능하고 비용 효율적인 애플리케이션 관리 임시 대기열을 생성할 수 있습니다.	2019년 7월 25일
Amazon SQS용 SSE는 (미국 동부) 지역에서 사용할 수 있습니다 AWS GovCloud .	Amazon SQS용 서버 측 암호화 (SSE) 는 (미국 동부) 지역에서 사용할 수 있습니다. AWS GovCloud	2019년 6월 20일
FIFO 대기열은 아시아 태평양 (홍콩), 중국 (베이징), (미국 동부) 및 (미국 서부) 지역에서 사용할 수 있습니다. AWS GovCloud AWS GovCloud	FIFO 대기열은 아시아 태평양 (홍콩), 중국 (베이징), (미국 동부) 및 AWS GovCloud (미국 서부) 지역에서 사용할 수 있습니다. AWS GovCloud	2019년 5월 15일
Amazon SQS에 대해 Amazon VPC 엔드포인트 정책을 사용할 수 있음	Amazon SQS에 대한 Amazon VPC 엔드포인트 정책을 생성할 수 있습니다.	2019년 4월 4일

유럽(스톡홀름) 및 중국(닝샤) 리전에서 FIFO 대기열을 사용할 수 있음	유럽(스톡홀름) 및 중국(닝샤) 리전에서 FIFO 대기열을 사용할 수 있습니다.	2019년 3월 14일
Amazon SQS를 사용할 수 있는 모든 리전에서 FIFO 대기열을 사용할 수 있음	미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오레곤), 아시아 태평양(뭄바이), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 유럽(파리) 및 남아메리카(상파울루) 리전에서 FIFO 대기열을 사용할 수 있습니다.	2019년 2월 7일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.