aws

개발자 가이드

Amazon Data Firehose

Amazon Data Firehose: 개발자 가이드

Table of Contents

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.

Amazon Data Firehose란?

Amazon Data Firehose란 Amazon Simple Storage Service(Amazon S3), Amazon Redshift, Amazon OpenSearch Service, Amazon OpenSearch Serverless, Splunk, Apache Iceberg 테이블 및 사용자 지정 HTTP 엔드포인트 등의 대상 또는 지원되는 타사 서비스 공급자가 소유한 HTTP 엔드포인트(Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Coralogix, Elastic 등)에 실시간 스트리밍 데이터를 전송하기 위한 완전 관리형 서비스입니다. Amazon Data Firehose를 사용하면 애플리케이션을 쓰거나 리소스를 관리할 필요가 없습니다. 데이터 생산자를 Amazon Data Firehose로 데이터를 보내도록 구성하면 지정한 대상으로 데이터가 자동으로 전송됩니다. 전송 전에 데이터를 변환하도록 Amazon Data Firehose를 구성할 수도 있습니다.

AWS 빅 데이터 솔루션에 대한 자세한 내용은 의 빅 데이터를 AWS 참조하세요. AWS 스트리밍 데이터 솔루션에 대한 자세한 내용은 스트리밍 데이터란 무엇입니까?를 참조하세요.

주요 개념 알아보기

Amazon Data Firehose를 처음 사용할 때 다음 개념을 이해하고 있으면 도움이 될 수 있습니다.

Firehose 스트림

Amazon Data Firehose의 기본 엔터티입니다. Firehose 스트림을 만든 다음 데이터를 Firehose 스트림으로 전송하여 Amazon Data Firehose를 사용합니다. 자세한 내용은 자습서: 콘솔에서 Firehose 스트림 생성 및 Firehose 스트림으로 데이터 전송 섹션을 참조하세요.

레코드

데이터 생산자가 Firehose 스트림으로 보내는 관심 있는 데이터입니다. 레코드는 최대 1000KB가 될 수 있습니다.

데이터 생산자

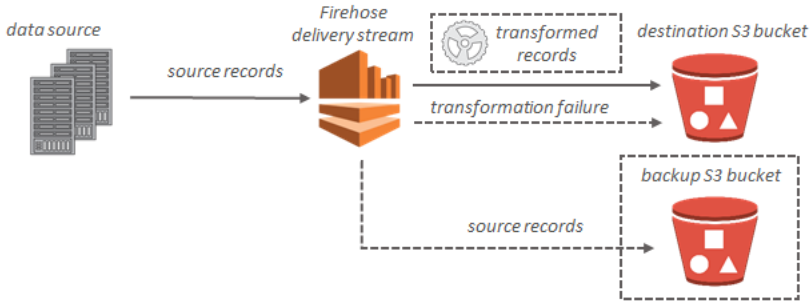
생산자는 Firehose 스트림에 레코드를 전송합니다. 예를 들어, Firehose 스트림에 로그 데이터를 보내는 웹 서버가 데이터 생산자입니다. 기존 Kinesis 데이터 스트림에서 데이터를 자동으로 읽어서 대상에 로드하도록 Firehose 스트림을 구성할 수도 있습니다. 자세한 내용은 Firehose 스트림으로 데이터 전송 섹션을 참조하세요.

버퍼 크기와 버퍼 간격

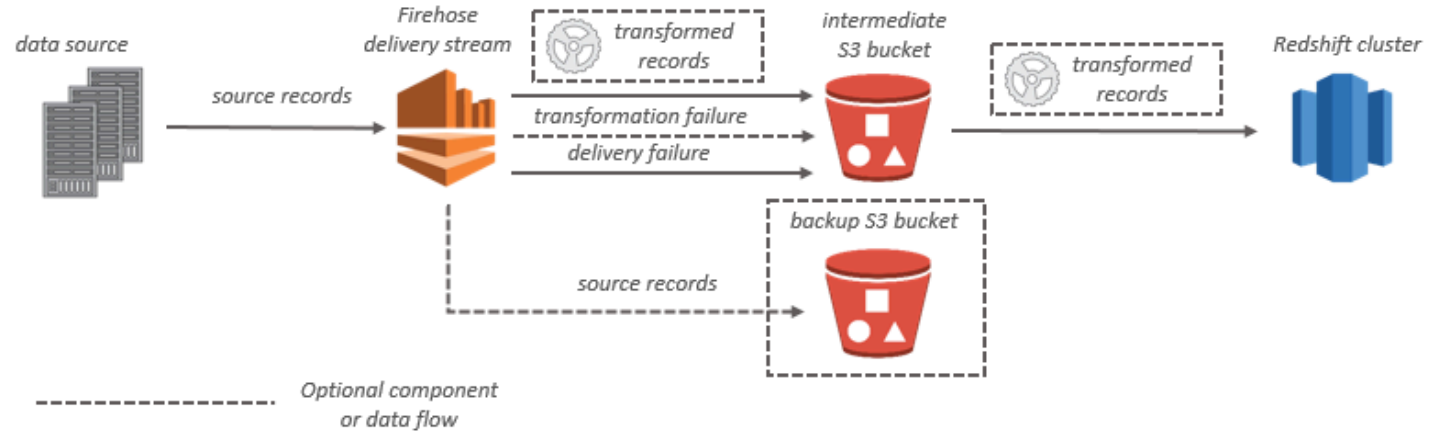
Amazon Data Firehose는 수신되는 스트리밍 데이터를 대상으로 전송하기 전에 특정 기간 또는 특정 크기로 버퍼링합니다. Buffer Size의 단위는 ‘MB’이고 Buffer Interval의 단위는 '초'입니다.

Amazon Data Firehose의 데이터 흐름 이해

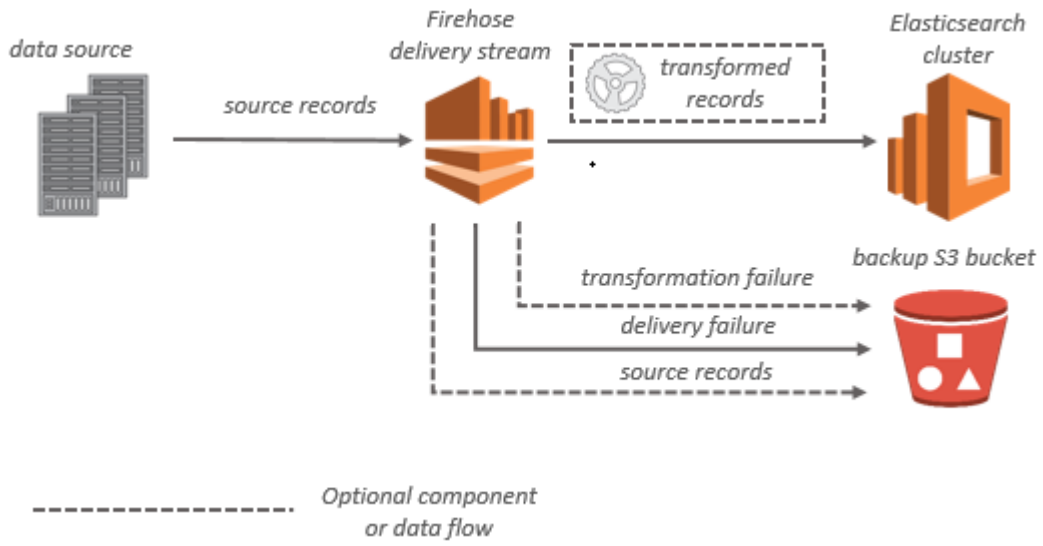
Amazon S3 대상인 경우, 스트리밍 데이터가 S3 버킷으로 전송됩니다. 데이터 변환이 활성화된 경우, 선택적으로 소스 데이터를 다른 Amazon S3 버킷으로 백업할 수 있습니다.



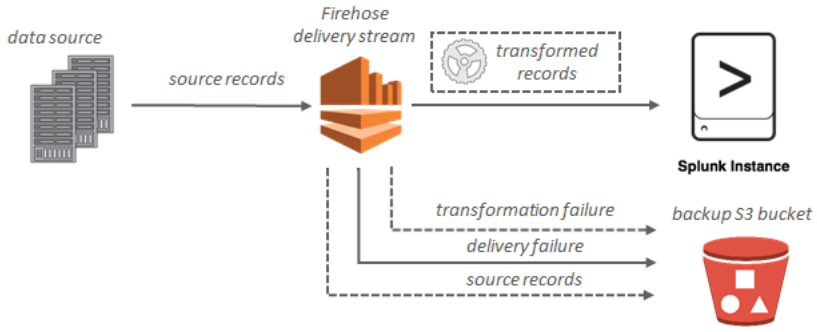
Amazon Redshift 대상인 경우, 스트리밍 데이터가 먼저 S3 버킷으로 전송됩니다. 그러면 Amazon Data Firehose는 Amazon Redshift COPY 명령을 실행하여 S3 버킷의 데이터를 Amazon Redshift 클러스터로 로드합니다. 데이터 변환이 활성화된 경우, 선택적으로 소스 데이터를 다른 Amazon S3 버킷으로 백업할 수 있습니다.



OpenSearch Service 대상인 경우 스트리밍 데이터가 OpenSearch Service 클러스터로 전송되며, 동시에 선택적으로 S3 버킷에 백업할 수 있습니다.



Splunk 대상인 경우 스트리밍 데이터가 Splunk 클러스터로 전송되며, 동시에 선택적으로 S3 버킷에 백업할 수 있습니다.



AWS SDK에서 Firehose 사용

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 여러 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예제 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제

SDK 설명서	코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예제
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

예제 가용성

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

Amazon Data Firehose를 설정하기 위한 사전 조건 완료

Amazon Data Firehose를 처음 사용한다면 먼저 다음 작업을 완료해야 합니다.

작업

- 에 가입 AWS
- (선택 사항) 라이브러리 및 도구 다운로드

에 가입 AWS

Amazon Web Services(AWS)에 가입하면 Amazon Data Firehose를 AWS포함한의 모든 서비스에 AWS 계정이 자동으로 등록됩니다. 사용자에게는 사용한 서비스에 대해서만 요금이 청구됩니다.

AWS 계정이 이미 있는 경우 다음 작업으로 건너뜁니다. AWS 계정이 없는 경우 다음 절차에 따라 계정을 생성합니다.

AWS 계정에 가입하려면

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

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

(선택 사항) 라이브러리 및 도구 다운로드

다음 라이브러리와 도구는 프로그래밍 방식 및 명령줄로 Amazon Data Firehose 작업을 하는 데 도움을 줍니다.

- Firehose API Operations는 Amazon Data Firehose가 지원하는 기본적인 작업 세트입니다.
- Go, Java, .NET, Node.js, Python 및 Ruby용 AWS SDKs에는 Amazon Data Firehose 지원 및 샘플이 포함되어 있습니다.

버전에 Amazon Data Firehose용 샘플 AWS SDK for Java 이 포함되어 있지 않은 경우 GitHub에서 최신 AWS SDK를 다운로드할 수도 있습니다.

- AWS Command Line Interface는 Amazon Data Firehose를 지원합니다. 를 AWS CLI 사용하면 명령 줄에서 여러 AWS 서비스를 제어하고 스크립트를 통해 자동화할 수 있습니다.

자습서: 콘솔에서 Firehose 스트림 생성

AWS Management Console 또는 AWS SDK를 사용하여 선택한 대상으로 Firehose 스트림을 생성할 수 있습니다.

Firehose 스트림을 만든 후 언제든지 Amazon Data Firehose 콘솔이나 UpdateDestination을 사용하여 구성을 업데이트할 수 있습니다. 구성이 업데이트되는 동안 Firehose 스트림은 `Active` 상태를 유지하며 사용자는 계속 데이터를 전송할 수 있습니다. 업데이트된 구성은 일반적으로 몇 분 내에 적용됩니다. 구성을 업데이트할 때마다 Firehose 스트림의 버전 번호가 1씩 증가합니다. 이는 전송된 Amazon S3 객체 이름에 반영됩니다. 자세한 내용은 Amazon S3 객체 이름 형식 구성 섹션을 참조하세요.

다음 주제에서 설명된 단계를 수행하여 Firehose 스트림을 생성할 수 있습니다.

주제

Firehose 스트림의 소스 및 대상 선택

1. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
2. Firehose 스트림 생성을 선택합니다.
3. Firehose 스트림 생성 페이지의 다음 옵션 중 하나에서 Firehose 스트림의 소스를 선택합니다.

    - Direct PUT - 생산자 애플리케이션이 직접 쓰는 대상 Firehose 스트림을 생성하려면 이 옵션을 선택합니다. 다음은 Amazon Data Firehose에서 Direct PUT와 통합되는 AWS 서비스 및 에이전트와 오픈 소스 서비스 목록입니다. 이 목록은 전체 목록이 아니며 Firehose로 직접 데이터를 전송하는 데 사용할 수 있는 추가 서비스가 있을 수 있습니다.
        - AWS SDK
        - AWS Lambda
        - AWS CloudWatch Logs

- AWS CloudWatch Events
- AWS Cloud Metric Streams
- AWS IoT
- AWS Eventbridge
- Amazon Simple Email Service
- Amazon SNS
- AWS WAF 웹 ACL 로그
- Amazon API Gateway - 액세스 로그
- Amazon Pinpoint
- Amazon MSK 브로커 로그
- Amazon Route 53 Resolver 쿼리 로그
- AWS Network Firewall Alerts Logs
- AWS Network Firewall Flow Logs
- Amazon Elasticache Redis SLOWLOG
- Kinesis Agent(Linux)
- Kinesis Tap(Windows)
- Fluentbit
- Fluentd
- Apache Nifi
- Snowflake

- Amazon Kinesis Data Streams - Kinesis 데이터 스트림을 데이터 소스로 사용하는 Firehose 스트림을 구성하려면 이 옵션을 선택합니다. 그러면 Firehose를 사용하여 기존 Kinesis 데이터 스트림에서 간편하게 데이터를 읽고 대상에 로드할 수 있습니다. Kinesis Data Streams를 데이터 소스로 사용하는 방법에 대한 자세한 내용은 Kinesis Data Streams를 사용하여 Firehose 스트림에 데이터 보내기를 참조하세요.
- Amazon MSK - Amazon MSK를 데이터 소스로 사용하는 Firehose 스트림을 구성하려면 이 옵션을 선택합니다. 그러면 Firehose를 사용하여 기존 Amazon MSK 클러스터에서 간편하게 데이터를 읽고 이를 지정된 S3 버킷으로 로드할 수 있습니다. 자세한 내용은 Amazon MSK를 사용해 Firehose 스트림으로 데이터 전송을 참조하세요.

4. Firehose가 지원하는 다음 대상 중 하나에서 Firehose 스트림의 대상을 선택합니다.

- Amazon OpenSearch Service

- Amazon OpenSearch Serverless
- Amazon Redshift
- Amazon S3
- Apache Iceberg 테이블
- Coralogix
- Datadog
- Dynatrace
- 탄력적
- HTTP 엔드포인트
- Honeycomb
- Logic Monitor
- Logz.io
- MongoDB Cloud
- New Relic
- Splunk
- Splunk Observability Cloud
- Sumo Logic
- Snowflake

5. Firehose 스트림 이름 의 경우 콘솔에서 생성하는 이름을 사용하거나 원하는 Firehose 스트림을 추가할 수 있습니다.

소스 설정 구성

콘솔에서 Firehose 스트림으로 정보를 전송하도록 선택한 소스를 기반으로 소스 설정을 구성할 수 있습니다. Amazon MSK 및 Amazon Kinesis Data Streams에 대한 소스 설정을 소스로 구성할 수 있습니다. Direct PUT에 소스로 사용할 수 있는 소스 설정이 없습니다.

Amazon MSK에 대한 소스 설정 구성

Amazon MSK를 선택하여 Firehose 스트림으로 정보를 전송하는 경우 MSK 프로비저닝 클러스터와 MSK 서버리스 클러스터 중에서 선택할 수 있습니다. 그런 다음 Firehose를 사용하여 특정 Amazon MSK 클러스터 및 주제에서 데이터를 쉽게 읽어 지정된 S3 대상으로 로드할 수 있습니다.

페이지의 소스 설정 섹션에서 다음 필드에 대한 값을 입력하세요.

Amazon MSK 클러스터 연결

클러스터 구성에 따라 프라이빗 부트스트랩 브로커(권장) 또는 퍼블릭 부트스트랩 브로커 옵션을 선택합니다. 부트스트랩 브로커는 Apache Kafka 클라이언트가 클러스터에 연결하기 위한 시작 지점으로 사용하는 브로커입니다. 퍼블릭 부트스트랩 브로커는 AWS 외부에서 퍼블릭 액세스하기 위한 용도이고, 프라이빗 부트스트랩 브로커는 AWS 내부에서 액세스하기 위한 용도입니다. Amazon MSK에 대한 자세한 내용은 Amazon Managed Streaming for Apache Kafka를 참조하세요.

프라이빗 부트스트랩 브로커를 통해 프로비저닝된 Amazon MSK 클러스터나 서버리스 Amazon MSK 클러스터에 연결하려면 클러스터가 다음 요구 사항을 모두 충족해야 합니다.

- 클러스터는 활성 상태여야 합니다.
- 클러스터의 액세스 제어 방법 중에 IAM이 있어야 합니다.
- IAM 액세스 제어 방법을 위해 다중 VPC 프라이빗 연결이 활성화되어 있어야 합니다.
- Firehose 서비스 주체에게 Amazon MSK `CreateVpcConnection` API 작업을 호출할 권한을 부여하는 리소스 기반 정책을 이 클러스터에 추가해야 합니다.

퍼블릭 부트스트랩 브로커를 통해 프로비저닝된 Amazon MSK 클러스터에 연결하려면 클러스터가 다음 요구 사항을 모두 충족해야 합니다.

- 클러스터는 활성 상태여야 합니다.
- 클러스터의 액세스 제어 방법 중에 IAM이 있어야 합니다.
- 클러스터는 공개적으로 액세스할 수 있어야 합니다.

MSK 클러스터 계정

Amazon MSK 클러스터가 있는 계정을 선택할 수 있습니다. 다음 중 하나가 될 수 있습니다.

- 현재 계정 - 현재 AWS 계정의 MSK 클러스터에서 데이터를 수집할 수 있습니다. 이를 위해 Firehose 스트림이 데이터를 읽을 Amazon MSK 클러스터의 ARN을 지정해야 합니다.
- 교차 계정 - 다른 AWS 계정의 MSK 클러스터에서 데이터를 수집할 수 있습니다. 자세한 내용은 Amazon MSK에서 계정 간 전송 섹션을 참조하세요.

주제

Firehose 스트림이 데이터를 수집할 때 사용할 Apache Kafka 주제를 지정하세요. Firehose 스트림 생성이 완료된 후에는 이 주제를 업데이트할 수 없습니다.

> **Note**
>
> Firehose는 Apache Kafka 메시지를 자동으로 압축 해제합니다.

Amazon Kinesis Data Streams의 소스 설정 구성

다음과 같이 Firehose 스트림으로 정보를 전송하도록 Amazon Kinesis Data Streams의 소스 설정을 구성합니다.

> **Important**
>
> Kinesis Producer Library(KPL)를 사용하여 Kinesis 데이터 스트림에 데이터를 쓰는 경우, 집계를 사용하여 해당 Kinesis 데이터 스트림에 쓰는 레코드를 결합할 수 있습니다. 그런 다음 해당 데이터 스트림을 Firehose 스트림의 원본으로 사용하면 Amazon Data Firehose가 레코드를 분해한 후 대상으로 전송합니다. 데이터를 변환하도록 Firehose 스트림을 구성한 경우, Amazon Data Firehose Firehose는 레코드를 분해한 후 AWS Lambda로 전송합니다. 자세한 내용은 Kinesis Producer Library를 사용하여 Amazon Kinesis Data Streams 생산자 개발 및 집계 섹션을 참조하세요.

소스 설정에서 Kinesis 데이터 스트림 목록에서 기존 스트림을 선택하거나 `arn:aws:kinesis:[Region]:[AccountId]:stream/[StreamName]` 형식으로 데이터 스트림 ARN을 입력합니다.

기존 데이터 스트림이 없는 경우 생성을 선택하여 Amazon Kinesis 콘솔에서 새 스트림을 생성합니다. Kinesis 스트림에 필요한 권한이 있는 IAM 역할이 필요할 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요. 새로운 스트림을 만든 후 새로 고침 아이콘을 선택하여 Kinesis 스트림 목록을 업데이트합니다. 스트림 개수가 많을 경우, [Filter by name]을 사용해 목록을 필터링합니다.

> **Note**
>
> Kinesis 데이터 스트림을 Firehose 스트림의 소스로 구성하는 경우, Amazon Data Firehose `PutRecord` 및 `PutRecordBatch` 작업이 비활성화됩니다. 이 경우 Firehose 스트림에 데이터를 추가하려면 Kinesis Data Streams `PutRecord` 및 `PutRecords` 작업을 사용합니다.

Amazon Data Firehose는 Kinesis 스트림의 `LATEST` 위치에서 데이터를 읽기 시작합니다. Kinesis Data Streams 위치에 대한 자세한 내용은 GetSharditerator를 참조하세요.

Amazon Data Firehose는 각 샤드에 대해 1초당 한 번씩 Kinesis Data Streams GetRecords 작업을 호출합니다. 그러나 전체 백업이 활성화되면 Firehose는 각 샤드에 대해 초당 두 번 Kinesis Data Streams `GetRecords` 작업을 호출합니다. 하나는 기본 전송 대상용이고 다른 하나는 전체 백업용입니다.

한 개 이상의 Firehose 스트림이 동일한 Kinesis 스트림에서 읽을 수 있습니다. 다른 Kinesis 애플리케이션(소비자)도 동일한 스트림에서 읽을 수 있습니다. Firehose 스트림이나 다른 소비자 애플리케이션의 각 호출 수는 샤드에 대한 전체 조절 한도를 기준으로 계산됩니다. 조절되지 않도록 하려면 애플리케이션을 신중하게 계획하세요. Kinesis Data Streams 제한에 대한 자세한 내용은 Amazon Kinesis Streams 제한을 참조하세요.

다음 단계로 진행하여 레코드 변환 및 형식 변환을 구성합니다.

(선택 사항) 레코드 변환 및 형식 변환 구성

Amazon Data Firehose가 레코드 데이터를 전환 및 변환하도록 구성합니다.

Firehose 스트림의 소스로 Amazon MSK를 선택하는 경우.

AWS Lambda를 사용하여 소스 레코드 변환 섹션에서 다음 필드에 값을 입력합니다.

1. 데이터 변환

   수신 데이터를 변환하지 않는 Firehose 스트림을 생성하려면 데이터 변환 활성화 확인란에 체크하지 마세요.

   Firehose가 수신 데이터를 전송하기 전 변환하기 위해 호출하여 사용할 Lambda 함수를 지정하려면 데이터 변환 활성화 확인란을 체크합니다. Lambda 블루프린트 중 하나를 사용하여 새 Lambda 함수를 구성하거나 기존 Lambda 함수를 선택할 수 있습니다. Lambda 함수에는 Firehose가 요구하는 상태 모델이 포함되어야 합니다. 자세한 내용은 Amazon Data Firehose에서 소스 데이터 변환 섹션을 참조하세요.
2. Convert record format(레코드 형식 변환) 섹션에서 다음 필드에 값을 입력합니다.

   Record format conversion(레코드 형식 변환)

   수신 데이터 레코드 형식을 변환하지 않는 Firehose 스트림을 생성하려면 비활성을 선택합니다.

수신 레코드 형식을 변환하려면 활성을 선택한 다음 원하는 출력 형식을 지정합니다. Firehose가 레코드 형식을 변환하는 데 사용할 스키마를 포함하는 AWS Glue 테이블을 지정해야 합니다. 자세한 내용은 입력 데이터 형식 변환 단원을 참조하십시오.

를 사용하여 레코드 형식 변환을 설정하는 방법의 예는 AWS::KinesisFirehose::DeliveryStream을 CloudFormation참조하세요.

Firehose 스트림의 소스로 Amazon Kinesis Data Streams 또는 Direct PUT를 선택하는 경우

소스 설정 섹션에서 다음 필드를 제공하세요.

1. 레코드 변환에서 다음 중 하나를 선택합니다.

   a. 대상이 Amazon S3 또는 Splunk인 경우 압축 해제 소스 레코드 Amazon CloudWatch Logs 섹션에서 압축 해제 켜기를 선택합니다.

   b. AWS Lambda를 사용하여 소스 레코드 변환 섹션에서 다음 필드에 값을 제공합니다.

      데이터 변환

      수신 데이터를 변환하지 않는 Firehose 스트림을 생성하려면 데이터 변환 활성화 확인란에 체크하지 마세요.

      Amazon Data Firehose가 수신 데이터를 전송하기 전 변환하기 위해 호출하여 사용할 Lambda 함수를 지정하려면, 데이터 변환 활성화 확인란을 체크합니다. Lambda 블루프린트 중 하나를 사용하여 새 Lambda 함수를 구성하거나 기존 Lambda 함수를 선택할 수 있습니다. Lambda 함수에는 Amazon Data Firehose가 요구하는 상태 모델이 포함되어야 합니다. 자세한 내용은 Amazon Data Firehose에서 소스 데이터 변환 섹션을 참조하세요.

2. Convert record format(레코드 형식 변환) 섹션에서 다음 필드에 값을 입력합니다.

   Record format conversion(레코드 형식 변환)

   수신 데이터 레코드 형식을 변환하지 않는 Firehose 스트림을 생성하려면 비활성을 선택합니다.

   수신 레코드 형식을 변환하려면 활성을 선택한 다음 원하는 출력 형식을 지정합니다. Amazon Data Firehose가 레코드 형식을 변환하는 데 사용할 스키마를 포함하는 AWS Glue 테이블을 지정해야 합니다. 자세한 내용은 입력 데이터 형식 변환 단원을 참조하십시오.

를 사용하여 레코드 형식 변환을 설정하는 방법의 예는 AWS::KinesisFirehose::DeliveryStream을 CloudFormation참조하세요.

대상 설정 구성

이 섹션에서는 선택한 대상에 따라 Firehose 스트림에 대해 구성해야 하는 설정을 설명합니다.

주제

Amazon S3의 대상 설정 구성

Firehose 스트림의 대상으로 Amazon S3을 사용하려면 다음 설정을 지정해야 합니다.

- 다음 필드에 값을 입력합니다.

  S3 버킷

  스트리밍 데이터가 전송되어야 하는 고유한 S3 버킷을 선택합니다. 새 S3 버킷을 생성하거나 기존 버킷을 선택할 수 있습니다.

  새 줄 구분 기호

  Amazon S3로 전송되는 객체의 레코드 사이에 새로운 줄 구분 기호를 추가하도록 Firehose 스트림을 구성할 수 있습니다. 이를 위해 Enabled(활성화)를 선택합니다. Amazon S3로 전달되는 객체의 레코드 사이에 새 줄 구분 기호를 추가하지 않으려면 Disabled(비활성화)를 선택합니다. Athena를 사용하여 집계된 레코드가 있는 S3 객체를 쿼리하려는 경우 이 옵션을 활성화합니다.

  동적 파티셔닝

  동적 파티셔닝을 활성화하고 구성하려면 Enabled(활성화)를 선택합니다.

  다중 레코드 분해

  Firehose 스트림의 레코드를 구문 분석하고 유효한 JSON 또는 지정된 새 줄 구분 기호를 기준으로 레코드를 분리하는 프로세스입니다.

  여러 개의 이벤트, 로그 또는 레코드를 단일 PutRecord 및 PutRecordBatch API 호출로 집계하는 경우라도 동적 파티셔닝을 활성화하고 구성할 수 있습니다. 집계된 데이터와 함께 동적 파티셔닝을 활성화하면 Amazon Data Firehose는 레코드를 구문 분석하여 각 API 호출 내에서 여러 개의 유효한 JSON 객체를 찾습니다. Kinesis Data Stream을 소스로 사용하여 Firehose 스트림을 구성할 때 Kinesis Producer Library(KPL)에 내장된 집계를 사용할 수도 있습니다. 데이터 파티션 기능은 데이터가 분해된 후에 실행됩니다. 따라서 각 API 호출의 각 레코드를 서로 다른 Amazon S3 접두사로 전송할 수 있습니다. 또한 Lambda 함수 통합을 활용하여 데이터 파티셔닝 기능 전에 기타 집계 해제나 기타 변환을 수행할 수 있습니다.

  > **Important**
  >
  > 데이터가 집계된 경우, 먼저 데이터를 분해한 경우에만 동적 파티셔닝을 적용할 수 있습니다. 따라서 집계된 데이터에 대해 동적 파티셔닝을 활성화하려면 Enabled(활성화)를 선택하여 다중 레코드 분해를 활성화해야 합니다.

Firehose 스트림은 KPL(protobuf) 분해, JSON 또는 구분 기호 분해, Lambda 처리, 데이터 파티셔닝, 데이터 형식 변환, Amazon S3 전송의 순서로 처리 과정을 수행합니다.

다중 레코드 분해 유형

다중 레코드 분해를 활성화한 경우 Firehose의 데이터 분해 방법을 지정해야 합니다. 드롭다운 메뉴를 사용하여 JSON 또는 구분 기호를 선택합니다.

인라인 구문 분석

이는 Amazon S3에 바인딩된 데이터를 동적으로 파티셔닝하기 위해 지원되는 메커니즘 중 하나입니다. 데이터의 동적 파티셔닝에 인라인 구문 분석을 사용하려면, 파티션 키로 사용할 데이터 레코드 파라미터를 지정하고 지정된 각 파티션 키의 값을 입력해야 합니다. 인라인 구문 분석을 활성화하고 구성하려면 Enabled(활성화)를 선택합니다.

> **Important**
>
> 소스 레코드를 변환하기 위해 위 단계에서 AWS Lambda 함수를 지정한 경우이 함수를 사용하여 S3에 바인딩된 데이터를 동적으로 분할할 수 있으며 인라인 구문 분석으로 파티셔닝 키를 생성할 수 있습니다. 동적 파티셔닝을 사용하면 인라인 구문 분석 또는 AWS Lambda 함수를 사용하여 파티셔닝 키를 생성할 수 있습니다. 또는 인라인 구문 분석과 AWS Lambda 함수를 동시에 사용하여 파티셔닝 키를 생성할 수 있습니다.

동적 파티션 키

Key 및 Value 필드를 사용하여 동적 파티션 키로 사용할 데이터 레코드 파라미터 및 동적 파티션 키 값을 생성하기 위한 jq 쿼리를 지정할 수 있습니다. Firehose에서는 jq 1.6만 지원됩니다. 동적 파티션 키는 최대 50개까지 지정할 수 있습니다. Firehose 스트림의 동적 파티셔닝을 성공적으로 구성하려면 동적 파티션 키 값에 유효한 jq 표현식을 입력해야 합니다.

S3 버킷 접두사

동적 파티셔닝을 활성화하고 구성할 때는 Amazon Data Firehose가 파티셔닝된 데이터를 전송할 S3 버킷 접두사를 반드시 지정해야 합니다.

동적 파티셔닝을 올바르게 구성하려면 S3 버킷 접두사의 수가 지정된 파티션 키의 수와 동일해야 합니다.

인라인 구문 분석 또는 지정된 AWS Lambda 함수를 사용하여 소스 데이터를 분할할 수 있습니다. 소스 데이터에 대한 파티션 키를 생성하도록 AWS Lambda 함수를 지정하는 경우, “PartitionKeyFromLambda:KeyID” 형식을 사용하여 S3 버킷 접두사 값을 수동으로 입력해야 합니다. 인라인 구문 분석을 사용하여 소스 데이터의 파티션 키를 지정하는 경우, “PartitionKeyFromQuery:KeyID” 형식을 사용하여 S3 버킷 미리 보기 값을 수동으로 입력하거나 동적 파티션 키 적용 버튼을 선택하고 동적 파티션 키/값 쌍을 사용하여 S3 버킷 접두사를 자동 생성할 수 있습니다. 인라인 구문 분석 또는 AWS Lambda로 데이터를 분할하는 동안 S3 버킷 접두사 !{namespace:value}에서 다음 표현식 양식을 사용할 수도 있습니다. 여기서 네임스페이스는 partitionKeyFromQuery 또는 partitionKeyFromLambda일 수 있습니다.

S3 버킷 및 S3 오류 출력 접두사 시간대

Amazon S3 객체의 사용자 지정 접두사에서 날짜 및 시간에 사용할 시간대를 선택합니다. 기본적으로 Firehose는 UTC에 시간 접두사를 추가합니다. 다른 시간대를 사용하려면 S3 접두사에 사용되는 시간대를 변경할 수 있습니다.

버퍼링 힌트

Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

S3 압축

GZIP, Snappy, Zip 또는 Hadoop-Compatible Snappy 데이터 압축 또는 데이터 압축 없음을 선택합니다. Amazon Redshift를 대상으로 하는 Firehose 스트림에는 Snappy, Zip, Hadoop 호환 Snappy 압축을 사용할 수 없습니다.

S3 파일 확장 형식(선택 사항)

Amazon S3 대상 버킷으로 전달되는 객체의 파일 확장 형식을 지정합니다. 이 기능을 활성화하면 지정된 파일 확장자가 데이터 형식 변환 또는 S3 압축 기능에 추가된 기본 파일 확장자(예: .parquet 또는 .gz)를 재정의합니다. 이 기능을 데이터 형식 변환 또는 S3 압축과 함께 사용할 때 올바른 파일 확장자를 구성했는지 확인하세요. 파일 확장자는 마침표(.)로 시작해야 하며 문자 0~9a~z!-_.*‘()를 포함할 수 있습니다. 파일 확장자는 128자를 초과할 수 없습니다.

S3 암호화

Firehose는 Amazon S3에서 전송된 데이터를 암호화하기 위해 AWS Key Management Service (SSE-KMS)를 사용한 Amazon S3 서버 측 암호화를 지원합니다. 대상 S3 버킷에 지정된 기본 암호화 유형을 사용하거나 소유한 키 목록의 AWS KMS 키로 암호화하도록 선택할 수 있습니다. AWS KMS 키로 데이터를 암호화하는 경우 기본 AWS 관리형 키(aws/s3) 또는 고객

관리형 키를 사용할 수 있습니다. 자세한 내용은 AWS KMS 관리형 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 데이터 보호를 참조하세요.

Apache Iceberg 테이블의 대상 설정 구성

Firehose는 중국 리전, 아시아 태평양(타이베이), AWS GovCloud (US) Regions아시아 태평양(말레이시아), 아시아 태평양(뉴질랜드), 멕시코(중부)를 AWS 리전 제외한 모든에서 Apache Iceberg 테이블을 대상으로 지원합니다.

대상인 Apache Iceberg 테이블에 대한 자세한 내용은 Amazon Data Firehose를 사용하여 Apache Iceberg 테이블에 데이터 전송을 참조하세요.

Amazon Redshift의 대상 설정 구성

이 섹션에서는 Amazon Redshift를 Firehose 스트림 대상으로 사용하기 위한 설정을 설명합니다.

Amazon Redshift 프로비저닝된 클러스터 및 Amazon Redshift Serverless 작업 그룹 중 어느 것을 사용하는지 여부에 따라 다음 절차 중 하나를 선택합니다.

- Amazon Redshift 프로비저닝된 클러스터
- Amazon Redshift Serverless 작업 그룹에 대한 대상 설정 구성

> **Note**
>
> Firehose는 향상된 VPC 라우팅을 사용하는 Amazon Redshift 클러스터에 쓸 수 없습니다.

Amazon Redshift 프로비저닝된 클러스터

이 섹션에서는 Amazon Redshift 프로비저닝된 클러스터를 Firehose 스트림 대상으로 사용하기 위한 설정을 설명합니다.

- 다음 필드에 값을 입력합니다.

  클러스터

  S3 버킷 데이터가 복사되는 Amazon Redshift 클러스터. Amazon Redshift 클러스터에 공개 액세스할 수 있도록 구성하고 Amazon Data Firehose IP 주소 차단을 해제합니다. 자세한 내용은 Firehose에 Amazon Redshift 대상에 대한 액세스 권한 부여 섹션을 참조하세요.

Authentication

사용자 이름/암호를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Amazon Redshift 클러스터에 액세스할 수 있습니다.

- 사용자 이름

  Amazon Redshift 클러스터에 대한 액세스 권한을 가진 Amazon Redshift 사용자를 특정합니다. 이 사용자에게는 S3 버킷에서 Amazon Redshift 클러스터로 데이터를 복사할 수 있는 Amazon Redshift `INSERT` 권한이 있어야 합니다.

- 암호

  클러스터에 액세스할 권한이 있는 사용자의 암호를 특정합니다.

- Secret

  Amazon Redshift 클러스터의 자격 증명 AWS Secrets Manager 이 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않는 경우 Amazon Redshift 보안 인증에 대해 AWS Secrets Manager 에서 보안 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 섹션을 참조하세요.

데이터베이스

데이터가 복사되는 Amazon Redshift 데이터베이스.

표

데이터가 복사되는 Amazon Redshift 테이블.

열

(선택 사항) 데이터가 복사되는 테이블의 특정 열입니다. Amazon S3 객체에 정의된 열의 수가 Amazon Redshift 테이블 내 열의 수보다 작은 경우 이 옵션을 사용합니다.

중간 S3 대상

Firehose는 먼저 S3 버킷으로 데이터를 전송한 다음 Amazon Redshift COPY 명령을 실행하여 Amazon Redshift 클러스터로 데이터를 로드합니다. 스트리밍 데이터가 전송되어야 하는 고유한 S3 버킷을 지정합니다. 새 S3 버킷을 생성하거나 기존에 가지고 있는 버킷을 선택합니다.

Firehose는 Amazon Redshift 클러스터에 S3 버킷의 데이터를 로드한 후 삭제하지 않습니다. 수명 주기 구성을 이용해 S3 버킷에서 데이터를 관리할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 수명 주기 관리를 참조하세요.

중간 S3 접두사

(옵션) Amazon S3 객체에 기본 접두사를 사용하려면 이 옵션을 비워 두세요. Firehose는 전송된 Amazon S3 객체에 대해 “YYYY/MM/dd/HH” UTC 시간 형식의 접두사를 자동으로 사용합니다. 이 접두사 시작 부분에 추가할 수 있습니다. 자세한 내용은 Amazon S3 객체 이름 형식 구성 섹션을 참조하세요.

COPY 옵션

Amazon Redshift COPY 명령에서 지정할 수 있는 파라미터. 이 파라미터는 구성에 필요할 수 있습니다. 예를 들어 Amazon S3 데이터 압축이 활성화된 경우 GZIP""이 필요합니다. S3 버킷이 Amazon Redshift 클러스터와 동일한 AWS 리전에 있지 않은 경우 "REGION"이 필요합니다. 자세한 내용을 알아보려면 Amazon Redshift 데이터베이스 개발자 안내서의 COPY를 참조하세요.

COPY 명령

Amazon Redshift COPY 명령. 자세한 내용을 알아보려면 Amazon Redshift 데이터베이스 개발자 안내서의 COPY를 참조하세요.

재시도 기간

Amazon Redshift 클러스터로의 데이터 COPY가 에 장애가 발생할 경우 Firehose가 재시도하는 데 걸리는 시간 (0~7,200초) 입니다. Firehose는 재시도 기간이 만료될 때까지 5분마다 재시도합니다. 재시도 기간을 0초로 설정한 경우, COPY 명령 실패 후 Firehose는 재시도를 수행하지 않습니다.

버퍼링 힌트

Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

S3 압축

GZIP, Snappy, Zip 또는 Hadoop-Compatible Snappy 데이터 압축 또는 데이터 압축 없음을 선택합니다. Amazon Redshift를 대상으로 하는 Firehose 스트림에는 Snappy, Zip, Hadoop 호환 Snappy 압축을 사용할 수 없습니다.

S3 파일 확장 형식(선택 사항)

S3 파일 확장자 형식(선택 사항) - Amazon S3 대상 버킷에 전달되는 객체에 대한 파일 확장자 형식을 지정합니다. 이 기능을 활성화하면 지정된 파일 확장자가 데이터 형식 변환 또는 S3 압축 기능에 추가된 기본 파일 확장자(예: .parquet 또는 .gz)를 재정의합니다. 이 기능을 데이터

형식 변환 또는 S3 압축과 함께 사용할 때 올바른 파일 확장자를 구성했는지 확인하세요. 파일 확장자는 마침표(.)로 시작해야 하며 문자 0~9a~z!-_.*'()를 포함할 수 있습니다. 파일 확장자는 128자를 초과할 수 없습니다.

S3 암호화

Firehose는 Amazon S3에서 전송된 데이터를 암호화하기 위해 AWS Key Management Service (SSE-KMS)를 사용한 Amazon S3 서버 측 암호화를 지원합니다. 대상 S3 버킷에 지정된 기본 암호화 유형을 사용하거나 소유한 키 목록의 AWS KMS 키로 암호화하도록 선택할 수 있습니다. AWS KMS 키로 데이터를 암호화하는 경우 기본 AWS 관리형 키(aws/s3) 또는 고객 관리형 키를 사용할 수 있습니다. 자세한 내용은 AWS KMS 관리형 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 데이터 보호를 참조하세요.

Amazon Redshift Serverless 작업 그룹에 대한 대상 설정 구성

이 섹션에서는 Amazon Redshift Serverless 작업 그룹을 Firehose 스트림 대상으로 사용하기 위한 설정을 설명합니다.

- 다음 필드에 값을 입력합니다.

  작업 그룹 이름

  S3 버킷 데이터가 복사되는 Amazon Redshift Serverless 작업 그룹. Amazon Redshift Serverless 작업 그룹에 공개 액세스할 수 있도록 구성하고 Firehose IP 주소 차단을 해제합니다. 자세한 내용은 Amazon Redshift Serverless에 연결의 공개 액세스가 가능한 Amazon Redshift Serverless 인스턴스에 연결 섹션 및 Firehose에 Amazon Redshift 대상에 대한 액세스 권한 부여를 참조하세요.

  Authentication

  사용자 이름/암호를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Amazon Redshift Serverless 작업 그룹에 액세스하도록 선택할 수 있습니다.

  - 사용자 이름

    Amazon Redshift Serverless 작업 그룹에 대한 액세스 권한을 가진 Amazon Redshift 사용자를 특정합니다. 이 사용자에게는 S3 버킷에서 Amazon Redshift Serverless 작업 그룹으로 데이터를 복사할 수 있는 Amazon Redshift `INSERT` 권한이 있어야 합니다.

  - 암호

Amazon Redshift Serverless 작업 그룹에 액세스할 권한이 있는 사용자의 비밀번호를 특정합니다.

- Secret

  Amazon Redshift Serverless 작업 그룹의 자격 증명 AWS Secrets Manager 이 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않는 경우 Amazon Redshift 보안 인증에 대해 AWS Secrets Manager 에서 보안 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 섹션을 참조하세요.

데이터베이스

데이터가 복사되는 Amazon Redshift 데이터베이스.

표

데이터가 복사되는 Amazon Redshift 테이블.

열

(선택 사항) 데이터가 복사되는 테이블의 특정 열입니다. Amazon S3 객체에 정의된 열의 수가 Amazon Redshift 테이블 내 열의 수보다 작은 경우 이 옵션을 사용합니다.

중간 S3 대상

Amazon Data Firehose는 먼저 S3 버킷으로 데이터를 전송한 다음 Amazon Redshift COPY 명령을 실행하여 Amazon Redshift Serverless 작업 그룹으로 데이터를 로드합니다. 스트리밍 데이터가 전송되어야 하는 고유한 S3 버킷을 지정합니다. 새 S3 버킷을 생성하거나 기존에 가지고 있는 버킷을 선택합니다.

Firehose는 Amazon Redshift Serverless 작업 그룹에 S3 버킷의 데이터를 로드한 후 삭제하지 않습니다. 수명 주기 구성을 이용해 S3 버킷에서 데이터를 관리할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 객체 수명 주기 관리를 참조하세요.

중간 S3 접두사

(옵션) Amazon S3 객체에 기본 접두사를 사용하려면 이 옵션을 비워 두세요. Firehose는 전송된 Amazon S3 객체에 대해 “YYYY/MM/dd/HH” UTC 시간 형식의 접두사를 자동으로 사용합니다. 이 접두사 시작 부분에 추가할 수 있습니다. 자세한 내용은 Amazon S3 객체 이름 형식 구성 섹션을 참조하세요.

COPY 옵션

Amazon Redshift COPY 명령에서 지정할 수 있는 파라미터. 이 파라미터는 구성에 필요할 수 있습니다. 예를 들어 Amazon S3 데이터 압축이 활성화된 경우 `GZIP""`이 필요합니다. S3 버킷이 Amazon Redshift Serverless 작업 그룹과 동일한 AWS 리전에 있지 않은 경우 "`REGION`"이 필요합니다. 자세한 내용을 알아보려면 Amazon Redshift 데이터베이스 개발자 안내서의 COPY를 참조하세요.

COPY 명령

Amazon Redshift COPY 명령. 자세한 내용을 알아보려면 Amazon Redshift 데이터베이스 개발자 안내서의 COPY를 참조하세요.

재시도 기간

Amazon Redshift Serverless 작업 그룹에 대한 데이터 COPY가 실패할 경우 Firehose가 재시도하는 데 걸리는 시간(0~7200초). Firehose는 재시도 기간이 만료될 때까지 5분마다 재시도합니다. 재시도 기간을 0초로 설정한 경우, COPY 명령 실패 후 Firehose는 재시도를 수행하지 않습니다.

버퍼링 힌트

Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

S3 압축

GZIP, Snappy, Zip 또는 Hadoop-Compatible Snappy 데이터 압축 또는 데이터 압축 없음을 선택합니다. Amazon Redshift를 대상으로 하는 Firehose 스트림에는 Snappy, Zip, Hadoop 호환 Snappy 압축을 사용할 수 없습니다.

S3 파일 확장 형식(선택 사항)

S3 파일 확장자 형식(선택 사항) - Amazon S3 대상 버킷에 전달되는 객체에 대한 파일 확장자 형식을 지정합니다. 이 기능을 활성화하면 지정된 파일 확장자가 데이터 형식 변환 또는 S3 압축 기능에 추가된 기본 파일 확장자(예: .parquet 또는 .gz)를 재정의합니다. 이 기능을 데이터 형식 변환 또는 S3 압축과 함께 사용할 때 올바른 파일 확장자를 구성했는지 확인하세요. 파일 확장자는 마침표(.)로 시작해야 하며 문자 0~9a~z!-_.*'()를 포함할 수 있습니다. 파일 확장자는 128자를 초과할 수 없습니다.

S3 암호화

Firehose는 Amazon S3에서 전송된 데이터를 암호화하기 위해 AWS Key Management Service (SSE-KMS)를 사용한 Amazon S3 서버 측 암호화를 지원합니다. 대상 S3 버킷에 지정된 기본 암호화 유형을 사용하거나 소유한 키 목록의 AWS KMS 키로 암호화하도록 선택할 수 있습니다. AWS KMS 키로 데이터를 암호화하는 경우 기본 AWS 관리형 키(aws/s3) 또는 고객 관리형 키를 사용할 수 있습니다. 자세한 내용은 AWS KMS 관리형 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 데이터 보호를 참조하세요.

OpenSearch Service의 대상 설정 구성

Firehose는 Elasticsearch 버전 - 1.5, 2.3, 5.1, 5.3, 5.5, 5.6 버전 뿐만 아니라 모든 6.*, 7.* 및 8.* 버전을 지원합니다. Firehose는 Amazon OpenSearch Service 2.x 및 3.x를 지원합니다.

이 섹션에서는 OpenSearch Service를 대상으로 사용하는 옵션에 대해 설명합니다.

- 다음 필드에 값을 입력합니다.

  OpenSearch Service 도메인

  데이터가 전송되는 OpenSearch Service 도메인.

  인덱스

  OpenSearch Service 클러스터에 데이터를 인덱싱할 때 사용되는 OpenSearch Service 인덱스 이름.

  인덱스 로테이션

  OpenSearch Service 인덱스의 로테이션 여부와 주기를 선택합니다. 인덱스 로테이션이 활성화되면 Amazon Data Firehose는 지정된 인덱스 이름에 해당 타임스탬프를 추가하여 교체합니다. 자세한 내용은 OpenSearch Service 인덱스 교체 구성 섹션을 참조하세요.

  유형

  OpenSearch Service 클러스터에 데이터를 인덱싱할 때 사용되는 OpenSearch Service 유형의 이름. Elasticsearch 7.x 및 OpenSearch 1.x의 경우 인덱스별 유형은 하나뿐입니다. 다른 유형을 가진 기존 인덱스에 새 유형을 지정하면 런타임 동안 Firehose가 오류를 반환합니다.

  Elasticsearch 7.x에서는 이 필드는 비워둡니다.

재시도 기간

OpenSearch에 대한 인덱스 요청이 실패한 경우 Firehose가 재시도하는 데 걸리는 시간 재시도 기간 동안 0~7,200초 사이의 값을 설정할 수 있습니다. 기본 재시도 기간은 300초입니다. Firehose는 재시도 기간이 만료될 때까지 지수 백오프를 통해 여러 번 재시도합니다.

재시도 기간이 만료되면 Firehose는 구성된 S3 오류 버킷인 DLQ(Dead Letter Queue)에 데이터를 전송합니다. DLQ로 전송되는 데이터의 경우 구성된 S3 오류 버킷에서 OpenSearch 대상으로 데이터를 다시 구동해야 합니다.

OpenSearch 클러스터의 가동 중지 또는 유지 관리로 인해 Firehose 스트림이 DLQ로 데이터를 전송하지 못하도록 하려면 재시도 기간을 초 단위로 더 높은 값으로 구성할 수 있습니다. AWS 지원 에 문의하여 재시도 지속 시간 값을 7,200초 이상으로 늘릴 수 있습니다.

DocumentID 유형

문서 ID를 설정하는 방법을 표시합니다. 지원되는 방법은 Firehose가 생성한 문서 ID 및 OpenSearch Service가 생성한 문서 ID입니다. 문서 ID 값이 설정되지 않은 경우 Firehose에서 생성된 문서 ID가 기본 옵션입니다. OpenSearch Service에서 생성된 문서 ID는 로그 분석 및 관찰성을 포함하여 쓰기가 많은 작업을 지원하고 OpenSearch Service 도메인에서 CPU 리소스를 적게 사용하여 성능이 향상되므로 권장되는 옵션입니다.

대상 VPC 연결

OpenSearch Service 도메인이 프라이빗 VPC에 있는 경우 이 섹션을 사용하여 해당 VPC를 지정합니다. 또한 Amazon Data Firehose가 OpenSearch Service 도메인으로 데이터를 전송할 때 사용할 서브넷 및 하위 그룹도 지정합니다. OpenSearch Service 도메인에서 사용하는 것과 동일한 보안 그룹을 사용할 수 있습니다. 다른 보안 그룹을 지정하는 경우 해당 보안 그룹이 OpenSearch Service 도메인의 보안 그룹에 대한 아웃바운드 HTTPS 트래픽을 허용하는지 확인합니다. 또한 OpenSearch Service 도메인의 보안 그룹이 Firehose 스트림을 구성할 때 지정한 보안 그룹의 HTTPS 트래픽을 허용해야 합니다. Firehose 스트림과 OpenSearch Service 도메인 모두에 동일한 보안 그룹을 사용하는 경우 보안 그룹의 인바운드 규칙이 HTTPS 트래픽을 허용하는지 확인합니다. 보안 그룹 규칙에 관한 자세한 정보는 Amazon VPC 설명서의 보안 그룹 규칙을 참조하세요.

> **Important**
>
> 프라이빗 VPC의 대상으로 데이터를 전송하기 위해 서브넷을 지정할 때는 선택한 서브넷에 사용 가능한 IP 주소가 충분히 있는지 확인합니다. 지정된 서브넷에 사용 가능한

IP 주소가 없는 경우 Firehose는 프라이빗 VPC에서 데이터 전송을 위한 ENI를 생성하거나 추가할 수 없으며 전송이 저하되거나 실패합니다.

버퍼 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

OpenSearch Serverless의 대상 설정 구성

이 섹션에서는 OpenSearch Serviceless를 대상으로 사용하는 옵션에 대해 설명합니다.

- 다음 필드에 값을 입력합니다.

  OpenSearch Serverless 컬렉션

  사용자의 데이터가 전달되는 OpenSearch Serverless 인덱스 그룹의 엔드포인트.

  인덱스

  OpenSearch Serverless 컬렉션에 데이터를 인덱싱할 때 사용되는 OpenSearch Serviceless 인덱스 이름.

  대상 VPC 연결

  OpenSearch Serverless 컬렉션이 프라이빗 VPC에 있는 경우 이 섹션을 사용하여 해당 VPC를 지정합니다. 또한 Amazon Data Firehose가 OpenSearch Serverless 컬렉션으로 데이터를 전송할 때 사용할 서브넷 및 하위 그룹도 지정합니다.

  Important

  프라이빗 VPC의 대상으로 데이터를 전송하기 위해 서브넷을 지정할 때는 선택한 서브넷에 사용 가능한 IP 주소가 충분히 있는지 확인합니다. 지정된 서브넷에 사용 가능한 IP 주소가 없는 경우 Firehose는 프라이빗 VPC에서 데이터 전송을 위한 ENI를 생성하거나 추가할 수 없으며 전송이 저하되거나 실패합니다.

재시도 기간

OpenSearch Serverless에 대한 인덱스 요청이 실패한 경우 Firehose가 재시도하는 데 걸리는 시간 재시도 기간 동안 0~7,200초 사이의 값을 설정할 수 있습니다. 기본 재시도 기간은 300초입니다. Firehose는 재시도 기간이 만료될 때까지 지수 백오프를 통해 여러 번 재시도합니다.

재시도 기간이 만료되면 Firehose는 구성된 S3 오류 버킷인 DLQ(Dead Letter Queue)에 데이터를 전송합니다. DLQ로 전송되는 데이터의 경우 구성된 S3 오류 버킷에서 OpenSearch Serverless 대상으로 데이터를 다시 구동해야 합니다.

OpenSearch Serverless 클러스터의 가동 중지 또는 유지 관리로 인해 Firehose 스트림이 DLQ로 데이터를 전송하지 못하도록 하려면 재시도 기간을 초 단위로 더 높은 값으로 구성할 수 있습니다. AWS 지원 에 문의하여 재시도 지속 시간 값을 7,200초 이상으로 늘릴 수 있습니다.

버퍼 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

HTTP 엔드포인트의 대상 설정 구성

이 섹션에서는 HTTP 엔드포인트를 대상으로 사용하는 옵션에 대해 설명합니다.

> **Important**
>
> HTTP 엔드포인트를 대상으로 선택한 경우 HTTP 엔드포인트 전송 요청 및 응답 사양에 대한 이해의 지침을 검토하고 따르세요.

- 다음 필드에 값을 입력하세요.

  HTTP 엔드포인트 이름 - 옵션

  HTTP 엔드포인트에 친숙한 이름을 지정합니다. 예를 들어 `My HTTP Endpoint Destination`입니다.

HTTP 엔드포인트 URL

HTTP 엔드포인트의 URL은 다음과 같은 형식으로 지정합니다: `https://xyz.httpendpoint.com`. URL은 HTTPS URL이어야 합니다.

Authentication

액세스 키를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 HTTP 엔드포인트에 액세스하도록 선택할 수 있습니다.

- (선택 사항) 액세스 키

  필요한 경우 엔드포인트 소유자에게 문의하여 Firehose에서 해당 엔드포인트로 데이터를 전송할 수 있도록 액세스 키를 받으세요(필요한 경우).

- Secret

  HTTP 엔드포인트에 대한 액세스 키 AWS Secrets Manager 가 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 액세스 키에 AWS Secrets Manager 대해에서 보안 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

콘텐츠 인코딩

Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 요청의 콘텐츠 인코딩을 활성화/비활성화하려면 GZIP 또는 비활성화를 선택합니다.

재시도 기간

Amazon Data Firehose가 선택된 HTTP 엔드포인트로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

> **Important**
>
> HTTP 엔드포인트 대상의 경우, CloudWatch Logs에 대상 엔드포인트로부터 413개의 응답 코드가 표시되는 경우 Firehose 스트림의 버퍼링 힌트 크기를 줄이고 다시 시도하세요.

Datadog의 대상 설정 구성

이 섹션에서는 Datadog를 대상으로 사용하는 옵션에 대해 설명합니다. Datadog에 대한 자세한 내용은 https://docs.datadoghq.com/integrations/amazon_web_services/를 참조하세요.

- 다음 필드에 값을 입력하세요.

  HTTP 엔드포인트 URL

  드롭다운 메뉴의 다음 옵션 중 하나에서 데이터를 전송할 위치를 선택합니다.

  - Datadog 로그 - US1
  - Datadog 로그 - US3
  - Datadog 로그 - US5

- Datadog 로그 - AP1
- Datadog 로그 - EU
- Datadog 로그 - GOV
- Datadog 지표 - US
- Datadog 지표 - US5
- Datadog 지표 - AP1
- Datadog 지표 - EU
- Datadog 구성 - US1
- Datadog 구성 - US3
- Datadog 구성 - US5
- Datadog 구성 - AP1
- Datadog 구성 - EU
- Datadog 구성 - US GOV

Authentication

API 키를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Datadog에 액세스하도록 선택할 수 있습니다.

- API 키

  Datadog에 문의하여 Firehose에서 이 엔드포인트로 데이터를 전송하는 데 필요한 API 키를 받으세요.

- Secret

  Datadog용 API 키 AWS Secrets Manager 가 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

콘텐츠 인코딩

Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 요청의 콘텐츠 인코딩을 활성화/비활성화하려면 GZIP 또는 비활성화를 선택합니다.

재시도 기간

Amazon Data Firehose가 선택된 HTTP 엔드포인트로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

Honeycomb의 대상 설정 구성

이 섹션에서는 Honeycomb을 대상으로 사용하는 옵션에 대해 설명합니다. Honeycomb에 대한 자세한 내용은 https://docs.honeycomb.io/getting-data-in/metrics/aws-cloudwatch-metrics/를 참조하세요.

- 다음 필드에 값을 입력하세요.

Honeycomb Kinesis 엔드포인트

HTTP 엔드포인트의 URL을 다음 형식으로 지정하세요: https://api.honeycomb.io/1/kinesis_events/{{dataset}}

Authentication

API 키를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Honeycomb에 액세스하도록 선택할 수 있습니다.

- API 키

  Honeycomb에 문의하여 Firehose에서 이 엔드포인트로 데이터를 전송하는 데 필요한 API 키를 받으세요.

- Secret

  Honeycomb용 API 키 AWS Secrets Manager 가 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

콘텐츠 인코딩

Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 사용자 요청의 콘텐츠 인코딩을 활성화하려면 GZIP을 선택합니다. 이 옵션은 Honeycomb 대상인 경우에 권장됩니다.

재시도 기간

Amazon Data Firehose가 선택된 HTTP 엔드포인트로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

Coralogix의 대상 설정 구성

이 섹션에서는 Coralogix를 대상으로 사용하는 옵션에 대해 설명합니다. Coralogix에 대한 자세한 내용은 Coralogix 시작하기를 참조하세요.

- 다음 필드에 값을 입력하세요.

  HTTP 엔드포인트 URL

  드롭다운 메뉴의 다음 옵션에서 HTTP 엔드포인트 URL을 선택합니다.

  - Coralogix - US
  - Coralogix - 싱가포르
  - Coralogix - 아일랜드
  - 코Coralogix - 인도
  - Coralogix - 스톡홀름

  Authentication

  프라이빗 키를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Coralogix에 액세스하도록 선택할 수 있습니다.

- 프라이빗 키

  Coralogix에 문의하여 Firehose에서 이 엔드포인트로 데이터를 전송하는 데 필요한 프라이빗 키를 받으세요.

- Secret

  Coralogix의 프라이빗 키 AWS Secrets Manager 가 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

콘텐츠 인코딩

Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 사용자 요청의 콘텐츠 인코딩을 활성화하려면 GZIP을 선택합니다. 이 옵션은 Coralogix 대상인 경우에 권장됩니다.

재시도 기간

Amazon Data Firehose가 선택된 HTTP 엔드포인트로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

- applicationName: Data Firehose를 실행하는 환경
- subsystemName: Data Firehose 통합의 이름
- computerName: 사용 중인 Firehose 스트림의 이름

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상에 대한 권장 버퍼 크기는 서비스 제공업체에 따라 다릅니다.

Dynatrace의 대상 설정 구성

이 섹션에서는 Dynatrace를 대상으로 사용하는 옵션에 대해 설명합니다. 자세한 내용은 https://www.dynatrace.com/support/help/technology-support/cloud-platforms/amazon-web-services/integrations/cloudwatch-metric-streams/을 참조하세요.

- Dynatrace를 Firehose 스트림의 대상으로 사용할 옵션을 선택합니다.

  수집 유형

  추가 분석 및 처리를 위해 Dynatrace에서 지표 또는 로그(기본값) 중 어떤 것을 제공할지를 선택합니다.

  HTTP 엔드포인트 URL

  드롭다운 메뉴에서 HTTP 엔드포인트 URL(Dynatrace US, Dynatrace EU 또는 Dynatrace 글로벌)을 선택합니다.

  Authentication

  API 토큰을 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Dynatrace에 액세스하도록 선택할 수 있습니다.

  - API 토큰

    Firehose에서 이 엔드포인트로 데이터 전송을 활성화하는 데 필요한 Dynatrace API 토큰을 생성합니다. 자세한 내용은 Dynatrace API - 토큰 및 인증을 참조하세요.

- Secret

  Dynatrace용 API 토큰 AWS Secrets Manager 이 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

API URL

Dynatrace 환경의 API URL을 입력하세요.

콘텐츠 인코딩

콘텐츠 인코딩을 활성화하여 요청 본문을 압축할지 여부를 선택합니다. Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 활성화되면 압축된 콘텐츠가 GZIP 형식으로 표시됩니다.

재시도 기간

Firehose가 선택된 HTTP 엔드포인트로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Firehose는 재시도 지속시간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다 최초 시도이든 재시도 이후이든 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 지속시간이 만료된 경우에도 Firehose는 확인을 수신하거나 확인 시간 제한 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과하는 경우 Firehose는 재시도 카운터에 시간이 남아 있는지 알아봅니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 버퍼 힌트에는 스트림의 버퍼 크기와 간격이 포함됩니다. 대상에 대한 권장 버퍼 크기는 서비스 제공업체에 따라 다릅니다.

LogicMonitor의 대상 설정 구성

이 섹션에서는 LogicMonitor를 대상으로 사용하는 옵션에 대해 설명합니다. 자세한 내용은 https://www.logicmonitor.com 참조하십시오.

- 다음 필드에 값을 입력하세요.

  HTTP 엔드포인트 URL

  HTTP 엔드포인트의 URL은 다음과 같은 형식으로 지정합니다.

  ```
  https://ACCOUNT.logicmonitor.com
  ```

  Authentication

  API 키를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 LogicMonitor에 액세스하도록 선택할 수 있습니다.

  - API 키

    LogicMonitor에 문의하여 Firehose에서 이 엔드포인트로 데이터를 전송하는 데 필요한 API 키를 받으세요.

  - Secret

    에서 LogicMonitor용 API 키 AWS Secrets Manager 가 포함된 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

  콘텐츠 인코딩

  Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 요청의 콘텐츠 인코딩을 활성화/비활성화하려면 GZIP 또는 비활성화를 선택합니다.

재시도 기간

Amazon Data Firehose가 선택된 HTTP 엔드포인트로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

Logz.io 대상 설정 구성

이 섹션에서는 Logz.io를 대상으로 사용하는 옵션에 대해 설명합니다. 자세한 내용은 https://logz.io/를 참조하세요.

> **Note**
>
> 유럽(밀라노) 리전에서는 Logz.io가 Amazon Data Firehose 대상으로 지원되지 않습니다.

- 다음 필드에 값을 입력하세요.

  HTTP 엔드포인트 URL

  HTTP 엔드포인트의 URL은 다음과 같은 형식으로 지정합니다. URL은 HTTPS URL이어야 합니다.

  ```
  https://listener-aws-metrics-stream-<region>.logz.io/
  ```

  예제

  ```
  https://listener-aws-metrics-stream-us.logz.io/
  ```

  Authentication

  배송 토큰을 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Logz.io 액세스할 수 있습니다.

  - 배송 토큰

    Firehose에서 이 엔드포인트로 데이터 전송을 활성화하는 데 필요한 배송 토큰을 얻으려면 Logz.io에 문의하세요.

  - Secret

    Logz.io 배송 토큰 AWS Secrets Manager 이 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

  재시도 기간

  Amazon Data Firehose가 Logz.io로 데이터 전송을 재시도하는 시간을 지정합니다.

  데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

MongoDB Atlas의 대상 설정 구성

이 섹션에서는 MongoDB Atlas를 대상으로 사용하는 옵션에 대해 설명합니다. 자세한 내용은 Amazon Web Services의 MongoDB Atlas를 참조하세요.

- 다음 필드에 값을 입력하세요.

  API Gateway CLI

  HTTP 엔드포인트의 URL은 다음과 같은 형식으로 지정합니다.

  ```
  https://xxxxx.execute-api.region.amazonaws.com/stage
  ```

  URL은 HTTPS URL이어야 합니다.

  Authentication

  API 키를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 MongoDB Atlas에 액세스할 수 있습니다.

- API 키

  Amazon Web Services의 MongoDB Atlas에 있는 지침에 따라 Firehose에서 이 엔드포인트로 데이터를 전송하는 데 필요한 `APIKeyValue`을(를) 가져옵니다.

- Secret

  MongoDB Atlas와 상호 작용하는 Lambda에서 지원하는 API Gateway용 API 키가 AWS Secrets Manager 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

콘텐츠 인코딩

Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 요청의 콘텐츠 인코딩을 활성화/비활성화하려면 GZIP 또는 비활성화를 선택합니다.

재시도 기간

Amazon Data Firehose가 선택된 타사 공급자로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

새 복제본의 대상 설정 구성

이 섹션에서는 New Relic을 대상으로 사용하는 옵션에 대해 설명합니다. 자세한 내용은 https://newrelic.com 참조하십시오.

- 다음 필드에 값을 입력하세요.

  HTTP 엔드포인트 URL

  드롭다운 목록의 다음 옵션에서 HTTP 엔드포인트 URL을 선택합니다.

  - New Relic 로그 - US
  - New Relic 지표 - US
  - New Relic 지표 - EU

  Authentication

  API 키를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 New Relic에 액세스할 수 있습니다.

  - API 키

    New Relic One Account 설정에서 40자 16진수 문자열의 라이선스 키를 입력합니다. Firehose에서 이 엔드포인트로 데이터를 전송하려면 이 API 키가 필요합니다.

  - Secret

    New Relic에 대한 API 키 AWS Secrets Manager 가 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

콘텐츠 인코딩

Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 요청의 콘텐츠 인코딩을 활성화/비활성화하려면 GZIP 또는 비활성화를 선택합니다.

재시도 기간

Amazon Data Firehose가 New Relic HTTP 엔드포인트로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

Snowflake의 대상 설정 구성

이 섹션에서는 Snowflake를 대상으로 사용하는 옵션에 대해 설명합니다.

> **Note**
>
> Snowflake와의 Firehose 통합은 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드), 미국 동부(오하이오), 아시아 태평양(도쿄), 유럽(프랑크푸르트), 아시아 태평양(싱가포르), 아시아 태평양(서울), 아시아 태평양(시드니), 아시아 태평양(뭄바이), 유럽(런던), 남아메리카(상파울루), 캐나다(중부), 유럽(파리), 아시아 태평양(오사카), 유럽(스톡홀름), 아시아 태평양(자카르타)에서 사용할 수 있습니다 AWS 리전.

연결 설정

- 다음 필드에 값을 입력하세요.

  Snowflake 계정 URL

  Snowflake 계정 URL을 지정합니다. 예를 들어 `xy12345.us-east-1.aws.snowflakecomputing.com`입니다. 계정 URL을 확인하는 방법은 Snowflake 설명서를 참조하세요. 포트 번호를 지정해서는 안 되지만 프로토콜(https://)은 선택 사항입니다.

  Authentication

  사용자 로그인, 프라이빗 키 및 암호를 수동으로 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Snowflake에 액세스할 수 있습니다.

  - 사용자 로그인

    데이터 로드에 사용할 Snowflake 사용자를 지정합니다. 사용자가 Snowflake 테이블에 데이터를 삽입할 수 있는 액세스 권한이 있는지 확인합니다.

  - 프라이빗 키

    Snowflake를 사용하여 인증할 프라이빗 키를 PKCS8 형식으로 지정합니다. 또한 프라이빗 키의 일부로 PEM 헤더와 바닥글을 포함하지 마세요. 키가 여러 줄로 분할된 경우 줄 바꿈을 제거합니다. 다음은 프라이빗 키의 예입니다.

    ```
    -----BEGIN PRIVATE KEY-----
    KEY_CONTENT
    -----END PRIVATE KEY-----
    ```

KEY_CONTENT에서 공백을 제거하고 Firehose에 제공합니다. 헤더/바닥글 또는 새 줄 문자는 필요하지 않습니다.

- 암호

  암호화된 프라이빗 키를 복호화할 암호를 지정합니다. 프라이빗 키가 암호화되지 않은 경우 이 필드를 비워 둘 수 있습니다. 자세한 내용은 키 페어 인증 및 키 교체 사용을 참조하세요.

- Secret

  Snowflake의 자격 증명 AWS Secrets Manager 이 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

역할 구성

기본 Snowflake 역할 사용 - 이 옵션을 선택하면 Firehose는 Snowflake에 역할을 전달하지 않습니다. 기본 역할은 데이터를 로드하는 것으로 간주됩니다. 기본 역할에 Snowflake 테이블에 데이터를 삽입할 수 있는 권한이 있는지 확인하세요.

사용자 지정 Snowflake 역할 사용 - Snowflake 테이블에 데이터를 로드할 때 Firehose가 수임할 기본이 아닌 Snowflake 역할을 입력합니다.

Snowflake 연결성

옵션은 프라이빗 또는 퍼블릭입니다.

프라이빗 VPCE ID(선택 사항)

Firehose가 Snowflake와 비공개로 연결할 수 있는 VPCE ID입니다. ID 형식은 com.amazonaws.vpce.[region].vpce-svc-*[id]*입니다. 자세한 내용은 AWS PrivateLink & Snowflake을 참조하세요.

> **Note**
>
> Snowflake 클러스터가 프라이빗 링크가 활성화된 경우 `AwsVpceIds` 기반 네트워크 정책을 사용하여 Amazon Data Firehose 데이터를 허용합니다. Firehose는 Snowflake 계정에서 IP 기반 네트워크 정책을 구성할 필요가 없습니다. IP 기반 네트워크 정책이 활성화되면 Firehose 연결이 중단될 수 있습니다. IP 기반 정책이 필요한 엣지 사례가

있는 경우 지원 티켓을 제출하여 Firehose 팀에 문의하세요. 사용할 수 있는 VPCE IDs 목록은 VPC에서 Snowflake에 액세스을 참조하세요.

데이터베이스 구성

- Firehose 스트림의 대상으로 Snowflake를 사용하려면 다음 설정을 지정해야 합니다.
  - Snowflake 데이터베이스 - Snowflake의 모든 데이터는 데이터베이스에 유지됩니다.
  - Snowflake 스키마 - 각 데이터베이스는 테이블 및 뷰와 같은 데이터베이스 객체의 논리적 그룹화인 하나 이상의 스키마로 구성됩니다.
  - Snowflake 테이블 - Snowflake의 모든 데이터는 데이터베이스 테이블에 저장되며, 열 및 행 모음으로 논리적으로 구성됩니다.

Snowflake 테이블의 데이터 로드 옵션

- JSON 키를 열 이름으로 사용
- VARIANT 열 사용
  - 콘텐츠 열 이름 - 원시 데이터를 로드해야 하는 테이블의 열 이름을 지정합니다.
  - 메타데이터 열 이름(선택 사항) - 메타데이터 정보를 로드해야 하는 테이블에서 열 이름을 지정합니다. 이 필드를 활성화하면 소스 유형에 따라 Snowflake 테이블에 다음 열이 표시됩니다.

    소스로 직접 PUT의 경우

    ```
    {
    "firehoseDeliveryStreamName" : "streamname",
    "IngestionTime" : "timestamp"
    }
    ```

    소스로 Kinesis Data Stream의 경우

    ```
    {
    "kinesisStreamName" : "streamname",
    "kinesisShardId" : "Id",
    "kinesisPartitionKey" : "key",
    "kinesisSequenceNumber" : "1234",
    "subsequenceNumber" : "2334",
    "IngestionTime" : "timestamp"
    ```

```
}
```

재시도 기간

Snowflake 서비스 문제로 인해 채널 열기 또는 Snowflake로의 전송에 실패하는 경우 Firehose가 재시도하는 시간(0~7,200초). 재시도 기간이 끝날 때까지 지수 백오프를 사용하여 Firehose 재시도합니다. 재시도 기간을 0(영)초로 설정하면 Firehose는 Snowflake 실패 시 재시도하지 않고 데이터를 Amazon S3 오류 버킷으로 라우팅합니다.

버퍼 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다. 자세한 내용은 버퍼링 힌트 구성 섹션을 참조하세요.

Splunk의 대상 설정 구성

이 섹션에서는 Splunk를 대상으로 사용하는 옵션에 대해 설명합니다.

> **Note**
>
> Firehose는 Classic Load Balancer 또는 Application Load Balancer 로 구성된 Splunk 클러스터에 데이터를 전송합니다.

- 다음 필드에 값을 입력하세요.

  Splunk 클러스터 엔드포인트

  엔드포인트를 확인하려면 Splunk 설명서에서 데이터를 Splunk 플랫폼으로 전송하도록 Amazon Data Firehose 구성을 참조하세요.

  Splunk 엔드포인트 유형

  대부분의 경우 `Raw endpoint`를 선택합니다. 를 사용하여 데이터를 사전 처리 AWS Lambda 하여 이벤트 유형별로 다른 인덱스로 데이터를 전송했는지 `Event endpoint` 선택합니다. 사용할 엔드포인트에 대한 세부 정보는 Splunk 설명서에서 데이터를 Splunk 플랫폼으로 전송하도록 Amazon Data Firehose 구성 섹션을 참조하세요.

Authentication

인증 토큰을 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Splunk에 액세스할 수 있습니다.

- 인증 토큰

  Amazon Data Firehose에서 데이터를 수신할 수 있는 Splunk 엔드포인트를 설정하려면 Splunk 설명서에서 Splunk Add-on for Amazon Data Firehose 설치 및 구성 개요를 참조하세요. 이번 Firehose 스트림에 대한 엔드포인트를 설정할 때 Splunk로부터 받은 토큰을 저장한 다음 여기에 추가합니다.

- Secret

  Splunk의 인증 토큰이 AWS Secrets Manager 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

HEC 확인 제한 시간

Amazon Data Firehose가 Splunk에서 보내는 인덱스 확인을 기다리는 시간을 지정합니다. 제한 시간에 도달하기 전에 Splunk가 확인을 보내지 않으면 Amazon Data Firehose는 이를 데이터 전송 실패로 간주합니다. 그러면 Amazon Data Firehose는 설정한 재시도 기간 값에 따라 재시도하거나 Amazon S3 버킷에 데이터를 백업합니다.

재시도 기간

Amazon Data Firehose가 Splunk로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 Splunk의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 Splunk로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 Splunk의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시

도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상에 대한 권장 버퍼 크기는 서비스 제공업체에 따라 다릅니다.

Splunk Observability Cloud의 대상 설정 구성

이 섹션에서는 Splunk Observability Cloud를 대상으로 사용하는 옵션에 대해 설명합니다. 자세한 내용은 https://docs.splunk.com/observability/en/gdi/get-data-in/connect/aws/aws-apiconfig.html#connect-to-aws-using-the-splunk-observability-cloud-api를 참조하세요.

- 다음 필드에 값을 입력하세요.

  Cloud 수집 엔드포인트 URL

  Splunk Observability Cloud의 실시간 데이터 수집 URL은 Splunk Observability 콘솔의 프로필 > 조직 > 실시간 데이터 수집 엔드포인트에서 확인할 수 있습니다.

  Authentication

  액세스 토큰을 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Splunk Observability Cloud에 액세스할 수 있습니다.

  - 액세스 토큰

    Splunk Observability 콘솔의 설정 내 액세스 토큰에서 INGEST 인증 범위와 함께 Splunk Observability 액세스 토큰을 복사합니다.

  - Secret

    Splunk Observability Cloud의 액세스 토큰 AWS Secrets Manager 이 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로인증 단원을 참조하십시오.

콘텐츠 인코딩

Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 요청의 콘텐츠 인코딩을 활성화/비활성화하려면 GZIP 또는 비활성화를 선택합니다.

재시도 기간

Amazon Data Firehose가 선택된 HTTP 엔드포인트로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

Sumo Logic의 대상 설정 구성

이 섹션에서는 Sumo Logic을 대상으로 사용하는 옵션에 대해 설명합니다. 자세한 내용은 https://www.sumologic.com 참조하십시오.

- 다음 필드에 값을 입력하세요.

  HTTP 엔드포인트 URL

  HTTP 엔드포인트의 URL은 다음과 같은 형식으로 지정합니다: `https://deployment name.sumologic.net/receiver/v1/kinesis/dataType/access token`. URL은 HTTPS URL이어야 합니다.

  콘텐츠 인코딩

  Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 요청의 콘텐츠 인코딩을 활성화/비활성화하려면 GZIP 또는 비활성화를 선택합니다.

  재시도 기간

  Amazon Data Firehose가 Sumo Logic으로 데이터 전송을 재시도하는 시간을 지정합니다.

  데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

  Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

  재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

  Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. Elastic 대상의 권장 버퍼 크기는 서비스 공급자마다 다릅니다.

Elastic의 대상 설정 구성

이 섹션에서는 Elastic을 대상으로 사용하는 옵션에 대해 설명합니다.

- 다음 필드에 값을 입력하세요.

  Elastic 엔드포인트 URL

  HTTP 엔드포인트의 URL은 다음과 같은 형식으로 지정합니다: `https://<cluster-id>.es.<region>.aws.elastic-cloud.com`. URL은 HTTPS URL이어야 합니다.

  Authentication

  API 키를 직접 입력하거나에서 보안 암호를 검색 AWS Secrets Manager 하여 Elastic에 액세스할 수 있습니다.

  - API 키

    Elastic 서비스에 문의하여 Firehose에서 해당 서비스로 데이터를 전송하는 데 필요한 API 키를 받으세요.

  - Secret

    Elastic에 대한 API 키 AWS Secrets Manager 가 포함된에서 보안 암호를 선택합니다. 드롭다운 목록에 보안 암호가 표시되지 않으면 AWS Secrets Manager에서 암호를 생성합니다. 자세한 내용은 Amazon Data Firehose AWS Secrets Manager 에서 로 인증 단원을 참조하십시오.

콘텐츠 인코딩

Amazon Data Firehose는 요청을 대상으로 전송하기 전에 콘텐츠 인코딩을 사용하여 요청 본문을 압축합니다. 요청의 콘텐츠 인코딩을 활성화/비활성화하려면 GZIP(기본값으로 선택되어 있음) 또는 비활성화를 선택합니다.

재시도 기간

Amazon Data Firehose가 Elastic으로 데이터 전송을 재시도하는 시간을 지정합니다.

데이터를 전송한 후, Amazon Data Firehose는 먼저 HTTP 엔드포인트의 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트로 데이터를 보낼 때마다(최초 시도이든 재시도이든) 확인 제한 시간 카운터를 시작하고 HTTP 엔드포인트의 확인 메시지를 기다립니다.

재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간 기간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

Amazon Data Firehose가 데이터 전송을 재시도하지 않도록 하려면 이 값을 0으로 설정합니다.

파라미터 - 옵션

Amazon Data Firehose는 각 HTTP 호출에 이러한 키-값 쌍을 포함시킵니다. 이 파라미터를 사용하면 대상을 식별하고 구성하는 데 도움이 됩니다.

버퍼링 힌트

Amazon Data Firehose는 수신 데이터를 지정된 대상으로 전송하기 전에 이를 버퍼링합니다. Elastic 대상의 권장 버퍼 크기는 1MiB입니다.

백업 설정 구성

Amazon Data Firehose는 Amazon S3를 사용하여 선택한 대상으로 전송하려고 시도하는 모든 데이터 또는 실패한 데이터만 백업합니다.

> **Important**
>
> - 백업 설정은 Firehose 스트림의 소스가 Direct PUT 또는 Kinesis Data Streams인 경우에만 지원됩니다.
> - 무버퍼링 기능은 애플리케이션 대상에만 사용할 수 있으며 Amazon S3 백업 대상에서는 사용할 수 없습니다.

다음 중 하나를 선택하면 Firehose 스트림에 대한 S3 백업 설정을 지정할 수 있습니다.

- Amazon S3를 Firehose 스트림의 대상으로 설정하고 AWS Lambda 함수를 지정하여 데이터 레코드를 변환하거나 Firehose 스트림의 데이터 레코드 형식을 변환하도록 선택한 경우.
- Amazon Redshift를 Firehose 스트림의 대상으로 설정하고 AWS Lambda 함수를 지정하여 데이터 레코드를 변환하도록 선택한 경우.
- Amazon OpenSearch Service, Datadog, Dynatrace, HTTP Endpoint, LogicMonitor, MongoDB Cloud, New Relic, Splunk, Sumo Logic, Snowflake, Apache Iceberg 테이블 중 한 서비스를 Firehose 스트림의 대상으로 설정한 경우입니다.

Firehose 스트림의 백업 설정은 다음과 같습니다.

- Amazon S3의 소스 레코드 백업 - S3 또는 Amazon Redshift를 대상으로 선택한 경우 이 설정은 소스 데이터 백업을 활성화할지 또는 비활성화된 상태로 유지할지를 나타냅니다. (S3 또는 Amazon Redshift 이외에) 지원되는 다른 서비스를 대상으로 선택한 경우 이 설정은 모든 소스 데이터를 백업할지 또는 실패한 데이터만 백업할지를 나타냅니다.
- S3 백업 버킷 - Amazon Data Firehose가 데이터를 백업하는 S3 버킷입니다.
- S3 백업 버킷 접두사 - Amazon Data Firehose가 사용자 데이터를 백업하는 접두사입니다.
- S3 백업 버킷 오류 출력 접두사 - 실패한 데이터는 모두 이 S3 버킷 오류 출력 접두사에 백업됩니다.
- 백업에 대한 버퍼링 힌트, 압축, 암호화 - Amazon Data Firehose는 Amazon S3를 사용하여 선택한 대상에 전송하려고 시도한 모든 데이터 또는 실패한 데이터만 백업합니다. Amazon Data Firehose는 수신 데이터를 Amazon S3에 전송(백업)하기 전에 이 데이터를 버퍼링합니다. 버퍼 크기 1~128MIB 및 버퍼 간격 60~900초에서 선택할 수 있습니다. 먼저 만족되는 조건에 의해 Amazon S3로의 데이터 전송이 트리거됩니다. 데이터 변환을 활성화하면 Amazon Data Firehose가 변환된 데이터를 수신한 시점부터 Amazon S3로 데이터가 전송될 때까지 버퍼 간격이 적용됩니다. 대상으로 데이터가 전송되는 속도가 Firehose 스트림에 데이터가 기록되는 속도보다 뒤처지는 경우 Amazon Data

Firehose가 속도를 따라잡기 위해 버퍼 크기를 동적으로 확장합니다. 이 작업을 통해 모든 데이터가 대상까지 잘 전송될 수 있습니다.

- S3 압축 - GZIP, Snappy, Zip 또는 Hadoop 호환 Snappy 데이터 압축 또는 데이터 압축 안 함을 선택합니다. Amazon Redshift를 대상으로 하는 Firehose 스트림에는 Snappy, Zip, Hadoop 호환 Snappy 압축을 사용할 수 없습니다.
- S3 파일 확장자 형식(선택 사항) - Amazon S3 대상 버킷에 전달되는 객체에 대한 파일 확장자 형식을 지정합니다. 이 기능을 활성화하면 지정된 파일 확장자가 데이터 형식 변환 또는 S3 압축 기능에 추가된 기본 파일 확장자(예: .parquet 또는 .gz)를 재정의합니다. 이 기능을 데이터 형식 변환 또는 S3 압축과 함께 사용할 때 올바른 파일 확장자를 구성했는지 확인하세요. 파일 확장자는 마침표(.)로 시작해야 하며 문자 0~9a~z!-_.*‘()를 포함할 수 있습니다. 파일 확장자는 128자를 초과할 수 없습니다.
- Firehose는 Amazon S3에서 전송된 데이터를 암호화하기 위해 AWS Key Management Service (SSE-KMS)를 사용한 Amazon S3 서버 측 암호화를 지원합니다. 대상 S3 버킷에 지정된 기본 암호화 유형을 사용하거나 소유한 키 목록의 AWS KMS 키로 암호화하도록 선택할 수 있습니다. AWS KMS 키로 데이터를 암호화하는 경우 기본 AWS 관리형 키(aws/s3) 또는 고객 관리형 키를 사용할 수 있습니다. 자세한 내용은 AWS KMS 관리형 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 데이터 보호를 참조하세요.

버퍼링 힌트 구성

Amazon Data Firehose는 수신되는 스트리밍 데이터를 메모리에 일정 크기(버퍼링 크기)로 일정 시간(버퍼링 간격) 동안 버퍼링한 후 지정된 대상에 전송합니다. 최적 크기의 파일을 Amazon S3에 전달하여 데이터 처리 애플리케이션에의 성능을 향상시키려 하거나 대상 속도에 맞게 Firehose 전송 속도를 조정하려는 경우 버퍼링 힌트를 사용할 수 있습니다.

새 Firehose 스트림을 생성하면서 버퍼링 크기와 버퍼링 간격을 구성하거나, 기존 Firehose 스트림의 버퍼링 크기 및 버퍼링 간격을 업데이트할 수 있습니다. 버퍼링 크기는 MB 단위로 측정되며, 버퍼링 간격은 초 단위로 측정됩니다. 하지만 이들 파라미터 중 하나에 값을 지정할 경우 다른 파라미터에도 값을 제공해야 합니다. 첫 번째 버퍼 조건이 충족되면 Firehose가 데이터를 전송하도록 트리거됩니다. 버퍼링 값을 구성하지 않으면 기본값이 사용됩니다.

AWS Management Console AWS Command Line Interface또는 AWS SDKs. 기존 스트림의 경우 콘솔의 편집 옵션 또는 UpdateDestination API를 사용하여 사용 사례에 적절한 값으로 버퍼링 힌트를 재구성할 수 있습니다. 새 스트림의 경우 콘솔을 사용하거나 CreateDeliveryStream API를 사용하여 새 스트림 생성의 일부로 버퍼링 힌트를 구성할 수 있습니다. 버퍼링 크기를 조정하려면 CreateDeliveryStream 또는 UpdateDestination API의 대상별 `DestinationConfiguration` 파라미터에서 `SizeInMBs` 및 `IntervalInSeconds`를 설정합니다.

> **Note**
>
> - 버퍼 힌트는 샤드 또는 파티션 수준에 적용되는 반면 동적 파티셔닝 버퍼 힌트는 스트림이나 주제 수준에 적용됩니다.
> - 실시간 사용 사례의 짧은 지연 시간을 달성하려면 무버퍼링 간격 힌트를 사용할 수 있습니다. 버퍼링 간격을 0초로 구성하면 Firehose는 데이터를 버퍼링하지 않고 몇 초 이내에 데이터를 전송합니다. 버퍼링 힌트를 더 낮은 값으로 변경하기 전에 공급업체에 문의하여 대상에 대해 권장되는 Firehose 버퍼링 힌트를 확인하세요.
> - 무버퍼링 기능은 애플리케이션 대상에만 사용할 수 있으며 Amazon S3 백업 대상에서는 사용할 수 없습니다.
> - 동적 파티셔닝에는 무버퍼링 기능을 사용할 수 없습니다.
> - Firehose는 버퍼 시간 간격을 60초 미만으로 구성하면 더 짧은 지연 시간을 제공하기 위해 S3 대상에 대해 멀티파트 업로드를 사용합니다. S3 대상에 대한 멀티파트 업로드로 인해 버퍼 시간 간격을 60초 미만으로 선택하면 S3 PUT API 비용이 약간 증가합니다.

대상별 버퍼링 힌트의 범위 및 기본값은 다음 표를 참조하세요.

Destination	버퍼링 크기(MB, 괄호 안은 기본값)	버퍼링 간격(초, 괄호 안은 기본값)
Amazon S3	1~128(5)	0~900(300)
Apache Iceberg 테이블	1~128(5)	0~900(300)
Amazon Redshift	1~128(5)	0~900(300)
OpenSearch Serverless	1~100(5)	0~900(300)
OpenSearch	1~100(5)	0~900(300)
Splunk	1~5(5)	0~60(60)
Datadog	1~4(4)	0~900(60)

Destination	버퍼링 크기(MB, 괄호 안은 기본값)	버퍼링 간격(초, 괄호 안은 기본값)
Coralogix	1~64(6)	0~900(60)
Dynatrace	1~64(5)	0~900(60)
탄력적	1	0~900(60)
Honeycomb	1~64(15)	0~900(60)
HTTP 엔드포인트	1~64(5)	0~900(60)
LogicMonitor	1~64(5)	0~900(60)
Logzio	1~64(5)	0~900(60)
mongoDB	1~16(5)	0~900(60)
newRelic	1~64(5)	0~900(60)
sumoLogic	1~64(1)	0~900(60)
Splunk Observability Cloud	1~64(1)	0~900(60)
Snowflake	1~128(1)	0~900(0)

고급 설정 구성

다음 섹션에는 Firehose 스트림 고급 설정에 대한 자세한 정보가 포함되어 있습니다.

- 서버 측 암호화 - Amazon Data Firehose는 Amazon S3에서 전송된 데이터를 암호화하기 위해 AWS Key Management Service(AWS KMS)를 사용한 Amazon S3 서버 측 암호화를 지원합니다. 자세한 내용은 AWS KMS 관리형 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 데이터 보호를 참조하세요.
- 오류 로깅 - Amazon Data Firehose는 처리 및 전송과 관련된 오류를 로그에 기록합니다. 또한 데이터 변환이 활성화되면 Lambda 호출을 기록하고 데이터 전송 오류를 CloudWatch Logs에 보낼 수 있

습니다. 자세한 내용은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 단원을 참조하십시오.

> **Important**
>
> 선택적으로, Firehose 스트림 생성 중에 Amazon Data Firehose 오류 로깅을 활성화하는 것이 좋습니다. 이렇게 하면 레코드 처리 또는 전송이 실패할 경우 오류 세부 정보에 액세스할 수 있습니다.

- 권한 - Amazon Data Firehose는 Firehose 스트림에 필요한 모든 권한에 대해 이 IAM 역할을 사용합니다. 필요한 권한이 자동으로 할당되는 새 역할을 만들거나, Amazon Data Firehose에 대해 생성된 기존 역할을 선택할 수 있습니다. 역할은 Firehose에 S3 버킷, AWS KMS 키(데이터 암호화가 활성화된 경우) 및 Lambda 함수(데이터 변환이 활성화된 경우)를 비롯한 다양한 서비스에 대한 액세스 권한을 부여하는 데 사용됩니다. 콘솔이 자리 표시자를 이용해 역할을 생성할 수 있습니다. 자세한 내용은 IAM이란?을 참조하세요.

  > **Note**
  >
  > IAM 역할(자리표시자 포함)은 Firehose 스트림을 만들 때 선택한 구성을 기반으로 생성됩니다. Firehose 스트림 소스나 대상을 변경하는 경우 IAM 역할을 수동으로 업데이트해야 합니다.

- 태그 - 태그를 추가하여 AWS 리소스를 구성하고, 비용을 추적하고, 액세스를 제어할 수 있습니다.

  `CreateDeliveryStream` 작업에서 태그를 지정하는 경우 Amazon Data Firehose는 `firehose:TagDeliveryStream` 작업에서 추가적인 권한 인증을 수행하여 사용자에게 태그를 생성할 권한이 있는지 확인합니다. 이 권한을 제공하지 않으면 IAM 리소스 태그가 있는 새 Firehose 스트림을 생성하려는 요청이 다음과 같이 `AccessDeniedException` 오류와 함께 실패합니다.

  ```
  AccessDeniedException
  User: arn:aws:sts::x:assumed-role/x/x is not authorized to perform:
   firehose:TagDeliveryStream on resource: arn:aws:firehose:us-east-1:x:deliverystream/
  x with an explicit deny in an identity-based policy.
  ```

  다음 예제는 사용자가 Firehose 스트림을 생성하고 태그를 적용할 수 있도록 허용하는 정책을 보여줍니다.

백업 및 고급 설정을 선택하고 선택 사항을 검토한 다음 Firehose 스트림 생성을 선택합니다.

새 Firehose 스트림을 사용할 수 있게 되기까지는 생성 중 상태에서 잠시 시간이 걸립니다. Firehose 스트림이 활성 상태가 되면 생산자로부터 데이터 전송을 시작할 수 있습니다.

샘플 데이터를 사용하여 Firehose 스트림 테스트

AWS Management Console 를 사용하여 시뮬레이션된 주식 티커 데이터를 수집할 수 있습니다. 콘솔은 브라우저에서 스크립트를 실행해 Firehose 스트림에 샘플 레코드를 넣습니다. 그러면 사용자는 테스트 데이터를 직접 생성할 필요 없이 Firehose 스트림의 구성을 테스트할 수 있습니다.

다음은 시뮬레이션한 데이터의 예입니다.

```
{"TICKER_SYMBOL":"QXZ","SECTOR":"HEALTHCARE","CHANGE":-0.05,"PRICE":84.51}
```

Firehose 스트림이 데이터를 전송하면 표준 Amazon Data Firehose 요금이 적용되지만 데이터가 생성될 때는 요금이 부과되지 않습니다. 이러한 요금이 발생하지 않도록 하기 위해 언제든 콘솔에서 샘플 스트림을 중단할 수 있습니다.

사전 조건

시작하기 전에 Firehose 스트림을 생성합니다. 자세한 내용은 자습서: 콘솔에서 Firehose 스트림 생성 섹션을 참조하세요.

Amazon S3 테스트

다음 절차에 따라 Amazon Simple Storage Service(S3)로 Firehose 스트림을 테스트합니다.

Amazon S3을 사용하여 Firehose 스트림 테스트

1. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
2. 활성 Firehose 스트림을 선택합니다. 데이터 전송을 시작하려면 Firehose 스트림이 활성 상태여야 합니다.
3. [Test with demo data]에서 [Start sending demo data]를 선택해 샘플 재고 티커 데이터를 생성합니다.
4. 화면의 지침을 따라 데이터가 S3 버킷으로 전송되고 있는지 확인합니다. 버킷의 버퍼링 구성에 따라 새 객체가 버킷에 표시되기까지 몇 분이 걸릴 수 있습니다.
5. 테스트가 완료되면 [Stop sending demo data]를 선택해 사용 요금이 발생하지 않도록 합니다.

Amazon Redshift로 테스트

다음 절차에 따라 Amazon Redshift로 Firehose 스트림을 테스트합니다.

Amazon Redshift를 사용하여 Firehose 스트림 테스트

1. Firehose 스트림은 Amazon Redshift 클러스터에 테이블이 표시되기를 기다립니다. SQL 인터페이스를 통해 Amazon Redshift에 연결하고 다음 문을 실행해 샘플 데이터를 수락하는 테이블을 만듭니다.

   ```
   create table firehose_test_table
   (
    TICKER_SYMBOL varchar(4),
    SECTOR varchar(16),
    CHANGE float,
    PRICE float
   );
   ```

2. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
3. 활성 Firehose 스트림을 선택합니다. 데이터 전송을 시작하려면 Firehose 스트림이 활성 상태여야 합니다.
4. 새로 생성된 `firehose_test_table` 테이블을 가리키도록 Firehose 스트림의 대상 세부 정보를 편집합니다.
5. [Test with demo data]에서 [Start sending demo data]를 선택해 샘플 재고 티커 데이터를 생성합니다.
6. 화면의 지침을 따라 데이터가 테이블로 전송되고 있는지 확인합니다. 버퍼링 구성에 따라 새 행이 테이블에 표시되기까지 몇 분이 걸릴 수 있습니다.
7. 테스트가 완료되면 [Stop sending demo data]를 선택해 사용 요금이 발생하지 않도록 합니다.
8. 다른 테이블을 가리키도록 Firehose 스트림의 대상 세부 정보를 편집합니다.
9. (선택 사항) `firehose_test_table` 테이블을 삭제합니다.

OpenSearch Service로 테스트

다음 절차에 따라 Amazon OpenSearch Service를 대상으로 사용하여 Firehose 스트림을 테스트합니다.

OpenSearch 서비스를 사용하여 Firehose 스트림 테스트

1. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
2. 활성 Firehose 스트림을 선택합니다. 데이터 전송을 시작하려면 Firehose 스트림이 활성 상태여야 합니다.
3. [Test with demo data]에서 [Start sending demo data]를 선택해 샘플 재고 티커 데이터를 생성합니다.
4. 화면의 지침을 따라 데이터가 OpenSearch Service 도메인으로 전송되고 있는지 확인합니다. 자세한 내용은 Amazon OpenSearch Service 개발자 안내서의 OpenSearch Service 도메인에서 문서 검색을 참조하세요.
5. 테스트가 완료되면 [Stop sending demo data]를 선택해 사용 요금이 발생하지 않도록 합니다.

Splunk로 테스트

다음 절차에 따라 Splunk를 대상으로 사용하여 Firehose 스트림을 테스트합니다.

Splunk를 사용하여 Firehose 스트림 테스트

1. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
2. 활성 Firehose 스트림을 선택합니다. 데이터 전송을 시작하려면 Firehose 스트림이 활성 상태여야 합니다.
3. [Test with demo data]에서 [Start sending demo data]를 선택해 샘플 재고 티커 데이터를 생성합니다.
4. Splunk 인덱스로 데이터가 전송되고 있는지 확인하세요. Splunk의 검색어 예시로는 `sourcetype="aws:firehose:json"` 및 `index="name-of-your-splunk-index"`가 있습니다. Splunk의 이벤트를 검색하는 방법에 대한 자세한 내용은 Splunk 설명서의 검색 매뉴얼을 참조하세요.

   테스트 데이터가 Splunk 인덱스에 표시되지 않는 경우 Amazon S3 버킷에 실패한 이벤트가 있는지 확인하세요. Splunk로 데이터가 전송되지 않음 또한 참조하세요.
5. 테스트를 완료하면 [Stop sending demo data]를 선택해 사용 요금이 발생하지 않도록 합니다.

Apache Iceberg 테이블로 테스트

다음 절차에 따라 Apache Iceberg 테이블을 대상으로 사용하여 Firehose 스트림을 테스트합니다.

Apache Iceberg 테이블을 사용하여 Firehose 스트림 테스트

1. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
2. 활성 Firehose 스트림을 선택합니다. 데이터 전송을 시작하려면 Firehose 스트림이 활성 상태여야 합니다.
3. [Test with demo data]에서 [Start sending demo data]를 선택해 샘플 재고 티커 데이터를 생성합니다.
4. 화면의 지침을 따라 데이터가 Apache Iceberg 테이블로 전송되고 있는지 확인합니다. 버퍼링 구성에 따라 새 객체가 버킷에 표시되기까지 몇 분이 걸릴 수 있습니다.
5. 테스트 데이터가 Apache Iceberg 테이블에 표시되지 않는 경우 Amazon S3 버킷에 실패한 이벤트가 있는지 확인하세요.
6. 테스트를 완료하면 [Stop sending demo data]를 선택해 사용 요금이 발생하지 않도록 합니다.

Firehose 스트림으로 데이터 전송

이 섹션은 다양한 데이터 소스를 사용하여 Firehose 스트림으로 데이터를 전송하는 방법을 설명합니다. Amazon Data Firehose를 처음 사용하는 경우, 잠시 시간을 내어 Amazon Data Firehose란? 섹션에 설명된 개념과 용어를 익히는 것이 좋습니다.

> **Note**
>
> 일부 AWS 서비스는 동일한 리전에 있는 Firehose 스트림에만 메시지와 이벤트를 보낼 수 있습니다. Amazon CloudWatch Logs, CloudWatch Events에 대한 대상을 구성할 때 Firehose 스트림이 옵션으로 표시되지 않는 경우 Firehose 스트림이 다른 서비스와 동일한 리전에 있는지 AWS IoT확인합니다. 각 리전의 서비스 엔드포인트에 대한 자세한 정보는 Amazon Data Firehose endpoints(Amazon Data Firehose 엔드포인트)를 참조하세요.

다음 데이터 소스에서 Firehose 스트림으로 데이터를 보낼 수 있습니다.

주제

- 데이터를 전송하도록 Kinesis 에이전트 구성
- AWS SDK를 사용하여 데이터 전송
- Firehose에 CloudWatch Logs 전송
- Firehose에 CloudWatch Events 전송
- Firehose AWS IoT 로 데이터를 전송하도록 구성

데이터를 전송하도록 Kinesis 에이전트 구성

Amazon Kinesis 에이전트는 데이터를 수집하고 Firehose로 전송하는 방법을 보여주는 참조 구현 역할을 하는 독립형 Java 소프트웨어 애플리케이션입니다. 에이전트가 파일 세트를 지속적으로 모니터링하고 새로운 데이터를 Firehose 스트림에 보냅니다. 에이전트는 파일 로테이션, 체크포인팅을 수행하는 방법을 보여주고, 실패할 경우 다시 시도합니다. 적시에 안정적이고 단순한 방식으로 모든 데이터를 전달하는 방법을 보여줍니다. 또한 효과적으로 모니터링하고 스트리밍 프로세스 문제를 해결하도록 Amazon CloudWatch 지표를 내보내는 방법을 보여줍니다. 자세한 내용은 awslabs/amazon-kinesis-agent를 참조하세요.

기본적으로 줄 바꿈 문자(`'\n'`)를 기반으로 각 파일에서 레코드가 구문 분석됩니다. 그러나 여러 줄 레코드를 구문 분석하도록 에이전트를 구성할 수도 있습니다(에이전트 구성 설정 지정 참조).

웹 서버, 로그 서버, 데이터베이스 서버 등 Linux 기반 서버 환경에 에이전트를 설치할 수 있습니다. 에이전트를 설치한 후 모니터링할 파일과 데이터의 Firehose 스트림을 지정하여 에이전트를 구성합니다. 에이전트가 구성되면 파일에서 일관되게 데이터를 수집하고, 안정적으로 Firehose 스트림에 전송합니다.

사전 조건

Kinesis 에이전트 사용을 시작하기 전에 다음 사전 조건을 충족하는지 확인해야 합니다.

- 사용자 운영 체제가 Amazon Linux 또는 Red Hat Enterprise Linux 버전 7 이상이어야 합니다.
- 에이전트 버전 2.0.0 이상은 JRE 버전 1.8 이상을 사용해 실행됩니다. 에이전트 버전 1.1.x는 JRE 버전 1.7 이상을 사용해 실행됩니다.
- Amazon EC2를 사용하여 에이전트를 실행하는 경우 EC2 인스턴스를 시작합니다.
- 지정한 IAM 역할 또는 AWS 자격 증명에는 에이전트가 Firehose 스트림으로 데이터를 전송하기 위해 Amazon Data Firehose PutRecordBatch 작업을 수행할 수 있는 권한이 있어야 합니다. 에이전트에 CloudWatch 모니터링을 활성화하는 경우 CloudWatch PutMetricData 작업을 수행할 권한도 필요합니다. 자세한 내용은 Amazon Data Firehose를 통한 액세스 제어, Kinesis 에이전트 상태 모니터링, Amazon CloudWatch에 대한 인증 및 액세스 제어를 참조하세요.

AWS 자격 증명 관리

다음 방법 중 하나를 사용하여 AWS 자격 증명을 관리합니다.

- 사용자 지정 자격 증명 공급자를 생성합니다. 자세한 내용은 the section called “사용자 지정 자격 증명 공급자 생성”을 참조하세요.
- EC2 인스턴스를 시작할 때 IAM 역할을 지정합니다.
- 에이전트를 구성할 때 AWS 자격 증명을 지정합니다(의 구성 표`awsSecretAccessKey`에서 `awsAccessKeyId` 및 항목 참조the section called “에이전트 구성 설정 지정”).
- 를 편집`/etc/sysconfig/aws-kinesis-agent`하여 AWS 리전 및 AWS 액세스 키를 지정합니다.
- EC2 인스턴스가 다른 AWS 계정에 있는 경우 IAM 역할을 생성하여 Amazon Data Firehose 서비스에 대한 액세스를 제공합니다. 에이전트를 구성할 때 해당 역할을 지정합니다(assumeRoleARN 및 assumeRoleExternalId 참조). 이전 방법 중 하나를 사용하여이 역할을 수임할 권한이 있는 다른 계정의 사용자의 AWS 자격 증명을 지정합니다.

사용자 지정 자격 증명 공급자 생성

사용자 지정 자격 증명 공급자를 생성하고 `userDefinedCredentialsProvider.classname` 및 `userDefinedCredentialsProvider.location` 구성 설정에서 클래스 이름과 Kinesis 에이전트까지의 jar 경로를 지정할 수 있습니다. 이러한 두 가지 구성 설정에 대한 설명은 the section called “에이전트 구성 설정 지정” 섹션을 참조하세요.

사용자 지정 자격 증명 공급자를 생성하려면 다음 예와 같이 `AWS CredentialsProvider` 인터페이스를 구현하는 클래스를 정의합니다.

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;

public class YourClassName implements AWSCredentialsProvider {
    public YourClassName() {
    }

    public AWSCredentials getCredentials() {
        return new BasicAWSCredentials("key1", "key2");
    }

    public void refresh() {
    }
}
```

클래스에는 인수를 취하지 않는 생성자가 있어야 합니다.

AWS 는 새로 고침 방법을 주기적으로 호출하여 업데이트된 자격 증명을 가져옵니다. 자격 증명 공급자가 수명 주기 동안 계속 다른 자격 증명을 제공하도록 하려면 이 메서드에 자격 증명을 새로 고치는 코드를 포함합니다. 또는 자격 증명 공급자가 정적(변경되지 않는) 자격 증명을 제공하기를 원할 경우 이 메서드를 비워 둘 수 있습니다.

에이전트 다운로드 및 설치

우선 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하세요. 연결 문제가 있는 경우 Amazon EC2 사용 설명서의 인스턴스 연결 문제 해결을 참조하세요.

다음으로, 다음 중 한 가지 방법을 사용하여 인스턴스를 설치합니다.

- Amazon Linux 리포지토리에서 에이전트 설정하는 방법

이 방법은 Amazon Linux 인스턴스에만 해당됩니다. 다음 명령을 사용합니다.

```
sudo yum install -y aws-kinesis-agent
```

에이전트 v 2.0.0 이상은 운영 체제가 Amazon Linux 2(AL2)인 컴퓨터에 설치됩니다. 이 에이전트 버전에는 Java 1.8 이상이 필요합니다. 필요한 Java 버전이 아직 없는 경우 에이전트 설치 프로세스에서 해당 버전이 설치됩니다. Amazon Linux 2에 대한 자세한 내용은 https://aws.amazon.com/amazon-linux-2/를 참조하세요.

- Amazon S3 리포지토리에서 에이전트 설정하는 방법

  이 방법은 공개적으로 사용 가능한 리포지토리에서 에이전트를 설치하기 때문에, Red Hat Enterprise Linux 및 Amazon Linux 2 인스턴스에도 적용됩니다. 다음 명령을 사용하여 에이전트 버전 2.x.x의 최신 버전을 다운로드하고 설치합니다.

  ```
  sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-latest.amzn2.noarch.rpm
  ```

  특정 버전의 에이전트를 설치하려면 명령에서 버전 번호를 지정합니다. 예를 들어 다음 명령은 에이전트 버전 2.0.1을 설치합니다.

  ```
  sudo yum install -y https://streaming-data-agent.s3.amazonaws.com/aws-kinesis-agent-2.0.1-1.amzn1.noarch.rpm
  ```

  현재 Java 1.7을 사용하고 있고 업그레이드를 원하지 않는 경우 Java 1.7과 호환되는 에이전트 버전 1.xx를 다운로드할 수 있습니다. 예를 들어 다음 명령을 사용하여 에이전트 버전 1.1.6을 다운로드할 수 있습니다.

  ```
  sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-1.1.6-1.amzn1.noarch.rpm
  ```

다음 명령을 사용하여 최신 에이전트를 다운로드할 수 있습니다.

```
sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-latest.amzn2.noarch.rpm
```

- GitHub 리포지토리에서 에이전트 설정하는 방법

  1. 먼저 에이전트 버전에 따라 필요한 Java 버전이 설치되어 있는지 확인합니다.
  2. awslabs/amazon-kinesis-agent GitHub 리포지토리에서 에이전트를 다운로드합니다.
  3. 다운로드 디렉터리로 이동하고 다음 명령을 실행해 에이전트를 설치합니다.

     ```
     sudo ./setup --install
     ```

- Docker 컨테이너에서 에이전트 설정

  amazonlinux 컨테이너 베이스를 통해서도 컨테이너에서 Kinesis 에이전트를 실행할 수 있습니다. 다음 Dockerfile을 사용하여 `docker build`를 실행합니다.

```
FROM amazonlinux

RUN yum install -y aws-kinesis-agent which findutils
COPY agent.json /etc/aws-kinesis/agent.json

CMD ["start-aws-kinesis-agent"]
```

에이전트 구성 및 시작

에이전트를 구성하고 시작하려면

1. 구성 파일(`/etc/aws-kinesis/agent.json`)을 열고 편집합니다(기본 파일 액세스 권한을 사용하는 경우 수퍼유저로).

   이 구성 파일에서, 에이전트가 데이터를 수집하는 파일(`"filePattern"`)과 에이전트가 데이터를 보내는 대상 Firehose 스트림의 이름(`"deliveryStream"`)을 지정합니다. 파일 이름은 패턴이며, 에이전트가 파일 로테이션을 인식합니다. 파일을 로테이션하거나 초당 1회 이하 새 파일을 생성할 수 있습니다. 에이전트는 파일 생성 타임스탬프를 사용하여 추적할 파일을 결정하고 Firehose 스

트림으로 추적합니다. 초당 1회보다 자주 새 파일을 생성하거나 파일을 로테이션하면 에이전트가 제대로 구별되지 않습니다.

```
{
   "flows": [
        {
            "filePattern": "/tmp/app.log*",
            "deliveryStream": "yourdeliverystream"
        }
   ]
}
```

기본 AWS 리전은 입니다us-east-1. 다른 리전을 사용 중이라면 리전에 엔드포인트를 지정해 구성 파일에 firehose.endpoint 설정을 추가합니다. 자세한 내용은 에이전트 구성 설정 지정 섹션을 참조하세요.

2. 에이전트를 수동으로 시작합니다.

```
sudo service aws-kinesis-agent start
```

3. (선택 사항) 시스템 시작 시 에이전트가 시작되도록 구성합니다.

```
sudo chkconfig aws-kinesis-agent on
```

현재 이 에이전트는 배경에서 시스템 서비스로 실행 중입니다. 지정된 파일을 지속적으로 모니터링하고 지정된 Firehose 스트림으로 데이터를 보냅니다. 에이전트 활동은 /var/log/aws-kinesis-agent/aws-kinesis-agent.log에 기록됩니다.

에이전트 구성 설정 지정

이 에이전트는 두 가지 의무 구성 설정인 filePattern, deliveryStream과 추가 기능을 제공하는 선택적 구성 설정을 지원합니다. /etc/aws-kinesis/agent.json에서 의무 및 선택적 구성 설정을 지정할 수 있습니다.

구성 파일을 변경할 때마다 다음 명령을 이용해 에이전트를 중지했다 시작해야 합니다.

```
sudo service aws-kinesis-agent stop
sudo service aws-kinesis-agent start
```

또는 다음 명령을 사용할 수 있습니다.

```
sudo service aws-kinesis-agent restart
```

다음은 일반적인 구성 설정입니다.

구성 설정	설명
assumeRoleARN	사용자가 맡을 역할의 Amazon 리소스 이름(ARN). 자세한 내용은 IAM 사용 설명서의 IAM 역할을 사용하여 AWS 계정 간 액세스 권한 위임을 참조하세요.
assumeRoleExternalId	역할을 맡을 사람을 결정하는 선택적 식별자입니다. 자세한 내용은 IAM 사용 설명서의 외부 ID 사용 방법을 참조하세요.
awsAccessKeyId	AWS 기본 자격 증명을 재정의하는 액세스 키 ID입니다. 이 설정은 다른 모든 자격 증명 공급자보다 우선 적용됩니다.
awsSecretAccessKey	AWS 기본 자격 증명을 재정의하는 보안 암호 키입니다. 이 설정은 다른 모든 자격 증명 공급자보다 우선 적용됩니다.
cloudwatch.emitMetrics	(true)로 설정하면 에이전트가 CloudWatch로 지표를 내보낼 수 있습니다. 기본값: true
cloudwatch.endpoint	CloudWatch에 대한 리전 엔드포인트. 기본값: monitoring.us-east-1.amazonaws.com
firehose.endpoint	Amazon Data Firehose의 리전 엔드포인트입니다. 기본값: firehose.us-east-1.amazonaws.com
sts.endpoint	AWS Security Token Service의 리전 엔드포인트입니다. 기본값: https://sts.amazonaws.com
userDefinedCredentialsProvider.classname	사용자 지정 자격 증명 공급자를 정의하는 경우 이 설정을 사용하여 정규화된 클래스 이름을 지정합니다. 클래스 이름 끝에 .class를 포함하지 마세요.

구성 설정	설명
userDefinedCredentialsProvider.location	사용자 지정 자격 증명 공급자를 정의하는 경우 이 설정을 사용하여 사용자 지정 자격 증명 공급자를 포함하는 jar의 절대 경로를 지정합니다. 또한 에이전트는 /usr/share/aws-kinesis-agent/lib/ 위치에서 jar 파일을 찾습니다.

다음은 흐름 구성 설정입니다.

구성 설정	설명
aggregatedRecordSizeBytes	에이전트가 레코드를 집계한 다음, 한 번의 작업으로 Firehose 스트림에 입력하도록 하려면 이 설정을 지정합니다. 에이전트가 Firehose 스트림에 넣기 전에 집계 레코드에 대해 원하는 크기로 설정합니다. 기본값: 0(집계 없음)
dataProcessingOptions	Firehose 스트림으로 전송되기 전에 구문 분석된 각 레코드에 적용되는 처리 옵션 목록입니다. 처리 옵션은 지정된 순서로 진행됩니다. 자세한 내용은 에이전트를 사용한 데이터 사전 처리 섹션을 참조하세요.
deliveryStream	[필수] Firehose 스트림 이름입니다.
filePattern	[필수] 에이전트가 모니터링해야 하는 파일에 대한 glob입니다. 이 패턴과 일치하는 파일을 에이전트가 자동으로 선별하여 모니터링합니다. 이 패턴과 일치하는 모든 파일에 대한 읽기 권한을 aws-kinesis-agent-user 에 부여해야 합니다. 파일이 포함된 디렉터리에 대한 읽기 및 실행 권한을 aws-kinesis-agent-user 에 부여해야 합니다. **Important** 에이전트는 이 패턴과 일치하는 파일을 선택합니다. 에이전트가 의도하지 않은 레코드를 선택하지 않도록 하려면 이 패턴을 신중하게 선택합니다.
initialPosition	파일 구문 분석이 처음 시작된 위치입니다. 유효 값은 START_OF_FILE 및 END_OF_FILE 입니다.

구성 설정	설명
	기본값: END_OF_FILE
maxBuffer AgeMillis	Firehose 스트림으로 보내기 전 에이전트가 데이터를 버퍼링하는 최대 시간(밀리초)입니다. 값 범위: 1,000~900,000(1초 ~ 15분) 기본값: 60,000(1분)
maxBuffer SizeBytes	Firehose 스트림으로 보내기 전 에이전트가 데이터를 버퍼링하는 최대 크기(바이트)입니다. 값 범위: 1~4,194,304(4MB) 기본값: 4,194,304(4MB)
maxBuffer SizeRecords	Firehose 스트림으로 보내기 전 에이전트가 데이터를 버퍼링하는 최대 레코드 수입니다. 값 범위: 1~500 기본값: 500
minTimeBe tweenFile PollsMillis	에이전트가 새로운 데이터에 대해 모니터링한 파일을 폴링하고 구문 분석하는 시간 간격(밀리초)입니다. 값 범위: 1 이상 기본값: 100
multiLine StartPattern	레코드의 시작을 식별하기 위한 패턴입니다. 레코드는 패턴과 일치하는 줄 1개 및 패턴과 일치하지 않는 나머지 줄로 이루어져 있습니다. 유효한 값은 정규식입니다. 기본적으로 로그 파일에서 각각의 줄 바꿈은 하나의 레코드로 구문 분석됩니다.

구성 설정	설명
skipHeaderLines	모니터링한 파일을 시작할 때 에이전트가 구문 분석을 건너뛰는 줄의 개수입니다. 값 범위: 0 이상 기본값: 0(영)
truncatedRecordTerminator	레코드 크기가 Amazon Data Firehose 레코드 크기 제한을 초과할 때 에이전트가 구문 분석된 레코드를 자르는 데 사용하는 문자열입니다. (1,000KB) 기본값: '\n'(줄 바꿈)

여러 파일 디렉터리 및 스트림 구성

여러 개의 흐름 구성 설정을 지정하여, 에이전트가 여러 파일 디렉터리를 모니터링하고 여러 스트림으로 데이터를 보내도록 구성할 수 있습니다. 다음 구성 예제에서 에이전트는 두 개의 파일 디렉터리를 모니터링하고 각각 Kinesis 데이터 스트림 및 Firehose 스트림으로 데이터를 보냅니다. 데이터 스트림과 Firehose 스트림이 같은 리전에 있을 필요가 없도록 Kinesis Data Streams 및 Amazon Data Firehose에 서로 다른 엔드포인트를 지정할 수 있습니다.

```
{
    "cloudwatch.emitMetrics": true,
    "kinesis.endpoint": "https://your/kinesis/endpoint",
    "firehose.endpoint": "https://your/firehose/endpoint",
    "flows": [
        {
            "filePattern": "/tmp/app1.log*",
            "kinesisStream": "yourkinesisstream"
        },
        {
            "filePattern": "/tmp/app2.log*",
            "deliveryStream": "yourfirehosedeliverystream"
        }
    ]
}
```

Amazon Kinesis Data Streams에서 에이전트를 사용하는 방법에 대한 자세한 내용은 Kinesis Agent를 사용하여 Amazon Kinesis Data Streams에 쓰기를 참조하세요.

에이전트를 사용한 데이터 사전 처리

모니터링한 파일에서 구문 분석한 레코드를 Firehose 스트림으로 보내기 전에 에이전트가 사전 처리할 수 있습니다. 파일 흐름에 `dataProcessingOptions` 구성 설정을 추가하여 이 기능을 활성화할 수 있습니다. 하나 이상의 처리 옵션을 추가할 수 있으며, 추가된 옵션은 지정된 순서로 수행됩니다.

에이전트는 다음 처리 옵션을 지원합니다. 에이전트는 오픈 소스이므로, 처리 옵션을 더 개발하고 확장할 수 있습니다. 에이전트는 Kinesis Agent에서 다운로드할 수 있습니다.

처리 옵션

`SINGLELINE`

줄 바꿈 문자, 선행 공백과 후행 공백을 삭제해 여러 줄 레코드를 한 줄 레코드로 변환합니다.

```
{
    "optionName": "SINGLELINE"
}
```

`CSVTOJSON`

구분 기호로 구분된 형식에서 JSON 형식으로 레코드를 변환합니다.

```
{
    "optionName": "CSVTOJSON",
    "customFieldNames": [ "field1", "field2", ... ],
    "delimiter": "yourdelimiter"
}
```

`customFieldNames`

[필수] 각각의 JSON 키 값 쌍에서 키로 사용되는 필드 이름입니다. 예를 들어 `["f1", "f2"]`를 지정하면 레코드 "v1, v2"가 `{"f1":"v1","f2":"v2"}`로 변환됩니다.

`delimiter`

레코드에서 구분 기호로 사용되는 문자열입니다. 기본값은 쉼표(,)입니다.

LOGTOJSON

로그 형식에서 JSON 형식으로 레코드를 변환합니다. 지원되는 로그 형식은 Apache Common Log, Apache Combined Log, Apache Error Log 및 RFC3164 Syslog입니다.

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "logformat",
    "matchPattern": "yourregexpattern",
    "customFieldNames": [ "field1", "field2", … ]
}
```

logFormat

[필수] 로그 항목 형식입니다. 유효한 값은 다음과 같습니다.

- COMMONAPACHELOG - Apache Common Log 형식입니다. 각 로그 항목에는 기본적으로 "%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes}" 패턴이 있습니다.
- COMBINEDAPACHELOG - Apache Combined Log 형식입니다. 각 로그 항목에는 기본적으로 "%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes} %{referrer} %{agent}" 패턴이 있습니다.
- APACHEERRORLOG - Apache Error Log 형식입니다. 각 로그 항목에는 기본적으로 "[%{timestamp}] [%{module}:%{severity}] [pid %{processid}:tid %{threadid}] [client: %{client}] %{message}" 패턴이 있습니다.
- SYSLOG - RFC3164 Syslog 형식입니다. 각 로그 항목에는 기본적으로 "%{timestamp} %{hostname} %{program}[%{processid}]: %{message}" 패턴이 있습니다.

matchPattern

지정된 로그 형식에 대한 기본 패턴을 재정의합니다. 사용자 지정 형식을 사용하는 경우 이 설정을 이용해 로그 항목에서 값을 추출합니다. matchPattern을 지정하면 customFieldNames도 함께 지정해야 합니다.

customFieldNames

각각의 JSON 키 값 쌍에서 키로 사용되는 사용자 지정 필드 이름입니다. 이 설정을 사용하여 matchPattern에서 추출한 값에 필드 이름을 정의하거나 사전 정의된 로그 형식의 기본 필드 이름을 재정의합니다.

Example: LOGTOJSON 구성

다음은 Apache Common Log 항목을 JSON 형식으로 변환하는 LOGTOJSON 구성의 예제입니다.

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "COMMONAPACHELOG"
}
```

변환 전:

```
64.242.88.10 - - [07/Mar/2004:16:10:02 -0800] "GET /mailman/listinfo/hsdivision
 HTTP/1.1" 200 6291
```

변환 후:

```
{"host":"64.242.88.10","ident":null,"authuser":null,"datetime":"07/
Mar/2004:16:10:02 -0800","request":"GET /mailman/listinfo/hsdivision
 HTTP/1.1","response":"200","bytes":"6291"}
```

Example: 사용자 지정 필드가 있는 LOGTOJSON 구성

다음은 LOGTOJSON 구성의 또 다른 예제입니다.

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "COMMONAPACHELOG",
    "customFieldNames": ["f1", "f2", "f3", "f4", "f5", "f6", "f7"]
}
```

이 구성 설정을 사용하면 이전 예제의 동일한 Apache Common Log 항목이 다음과 같이 JSON 형식으로 변환됩니다.

```
{"f1":"64.242.88.10","f2":null,"f3":null,"f4":"07/Mar/2004:16:10:02 -0800","f5":"GET /
mailman/listinfo/hsdivision HTTP/1.1","f6":"200","f7":"6291"}
```

Example: Apache Common Log 항목 변환

다음 흐름 구성은 Apache Common Log 항목을 JSON 형식의 한 줄 레코드로 변환합니다.

```
{
    "flows": [
        {
            "filePattern": "/tmp/app.log*",
            "deliveryStream": "my-delivery-stream",
            "dataProcessingOptions": [
                {
                    "optionName": "LOGTOJSON",
                    "logFormat": "COMMONAPACHELOG"
                }
            ]
        }
    ]
}
```

Example: 여러 줄 레코드 변환

다음 흐름 구성은 첫 줄이 "[SEQUENCE="로 시작하는 여러 줄 레코드를 구문 분석합니다. 먼저 각각의 레코드가 한 줄 레코드로 변환됩니다. 그런 다음 탭 구분 기호를 기반으로 레코드에서 값이 추출됩니다. 추출된 값은 지정된 customFieldNames 값에 매핑되어 JSON 형식의 한 줄 레코드를 형성합니다.

```
{
    "flows": [
        {
            "filePattern": "/tmp/app.log*",
            "deliveryStream": "my-delivery-stream",
            "multiLineStartPattern": "\\[SEQUENCE=",
            "dataProcessingOptions": [
                {
                    "optionName": "SINGLELINE"
                },
                {
                    "optionName": "CSVTOJSON",
                    "customFieldNames": [ "field1", "field2", "field3" ],
                    "delimiter": "\\t"
                }
            ]
        }
    ]
}
```

Example: 일치 패턴이 있는 LOGTOJSON 구성

다음은 마지막 필드(바이트)가 생략되어 있으며 Apache Common Log 항목을 JSON 형식으로 변환하는 LOGTOJSON 구성의 예제입니다.

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "COMMONAPACHELOG",
    "matchPattern": "^([\\d.]+) (\\S+) (\\S+) \\[([\\w:/]+\\s[+\\-]\\d{4})\\] \"(.
+?)\" (\\d{3})",
    "customFieldNames": ["host", "ident", "authuser", "datetime", "request",
 "response"]
}
```

변환 전:

```
123.45.67.89 - - [27/Oct/2000:09:27:09 -0400] "GET /java/javaResources.html HTTP/1.0"
 200
```

변환 후:

```
{"host":"123.45.67.89","ident":null,"authuser":null,"datetime":"27/Oct/2000:09:27:09
 -0400","request":"GET /java/javaResources.html HTTP/1.0","response":"200"}
```

일반적인 에이전트 CLI 명령 사용

다음 표에는 AWS Kinesis 에이전트 작업을 위한 일련의 일반적인 사용 사례와 해당 명령이 나와 있습니다.

사용 사례:	명령
시스템 시작 시 에이전트가 자동으로 시작됩니다.	`sudo chkconfig aws-kinesis-agent on`
에이전트의 상태를 확인합니다.	`sudo service aws-kinesis-agent status`
에이전트를 중지합니다.	`sudo service aws-kinesis-agent stop`

사용 사례:	명령
이 위치에서 에이전트의 로그 파일을 읽습니다.	`/var/log/aws-kinesis-agent/aws-kinesis-agent.log`
에이전트를 제거합니다.	`sudo yum remove aws-kinesis-agent`

Kinesis Agent에서 전송할 때 문제 해결

이 표는 Amazon Kinesis Agent를 사용할 때 발생하는 일반적인 문제에 대한 문제 해결 정보와 솔루션을 제공합니다.

문제	Solution
Kinesis Agent가 Windows에서 작동하지 않는 이유는 무엇인가요?	Windows용 Kinesis 에이전트는 Linux 플랫폼용 Kinesis 에이전트와 다른 소프트웨어입니다.
왜 Kinesis 에이전트가 느려지거나 `RecordSendErrors` 가 증가하나요?	이는 대개 Kinesis에서 제한하기 때문입니다. Kinesis Data Streams의 `WriteProvisionedThroughputExceeded` 지표나 Firehose 스트림의 `ThrottledRecords` 지표를 확인하세요. 이 지표 중 0에서 증가한 수치가 있으면 스트림 제한을 늘려야 한다는 의미입니다. 자세한 내용은 Kinesis Data Stream limits 및 Firehose 스트림을 참조하세요. 제한을 확인한 후에는 Kinesis 에이전트가 대량의 작은 파일을 테일링하도록 구성되어 있는지 확인하세요. Kinesis 에이전트가 새 파일을 테일링할 때 지연이 발생하므로 Kinesis 에이전트는 소량의 대용량 파일을 추적합니다. 로그 파일을 더 큰 파일로 통합해 보세요.
`java.lang.OutOfMemoryError` 예외를 해결하려면 어떻게 해야 하나요?	Kinesis 에이전트에는 현재 워크로드를 처리할 메모리가 충분하지 않을 때 발생합니다. `/usr/bin/start-aws-kinesis-agent` 의 `JAVA_START_HEAP` 및

문제	Solution
	JAVA_MAX_HEAP 을 늘리고 에이전트를 다시 시작해 보세요.
IllegalStateException : connection pool shut down 예외를 해결하려면 어떻게 해야 하나요?	Kinesis 에이전트에는 현재 워크로드를 처리할 연결이 충분하지 않습니다. /etc/aws-kinesis/agent.json 에서 일반 에이전트 구성 설정의 maxConnections 및 maxSendingThreads 를 늘려 보세요. 이 필드의 기본값은 제공되는 런타임 프로세서의 12배입니다. 고급 에이전트 구성 설정에 대한 자세한 내용은 AgentConfiguration.java를 참조하세요.
Kinesis 에이전트의 다른 문제는 어떻게 디버그할 수 있나요?	/etc/aws-kinesis/log4j.xml 에서 DEBUG 레벨 로그를 활성화할 수 있습니다.
Kinesis 에이전트는 어떻게 구성하나요?	maxBufferSizeBytes 의 크기가 작을수록 Kinesis 에이전트는 더 자주 데이터를 전송합니다. 이렇게 하면 레코드의 전송 시간이 줄어 유용하지만, Kinesis에 대한 초당 요청 수가 증가합니다.
왜 Kinesis 에이전트가 중복 레코드를 보내나요?	이 문제는 파일 테일링이 잘못 구성되어 발생합니다. fileFlow's filePattern 마다 매칭되는 파일은 하나뿐이어야 합니다. 사용 중인 logrotate 모드가 copytruncate 모드인 경우에도 이 문제가 발생할 수 있습니다. 모드를 기본 모드 또는 생성 모드로 변경하여 중복을 피하세요. 중복 레코드 처리에 대한 자세한 내용은 중복 레코드 처리를 참조하세요.

AWS SDK를 사용하여 데이터 전송

Amazon Data Firehose API를 사용하면 Java용AWS SDK, .NET, Node.js, Python 또는 Ruby를 사용하여 Firehose 스트림에 데이터를 전송할 수 있습니다. Amazon Data Firehose를 처음 사용하는 경우, 잠시 시간을 내어 Amazon Data Firehose란? 섹션에 설명된 개념과 용어를 익히는 것이 좋습니다. 자세한 내용은 Amazon Web Services로 개발 시작을 참조하세요.

이 예제는 가능한 모든 예외를 확인하지 않거나 가능한 모든 보안 및 성능 고려 사항을 감안하지 않는다는 점에서 프로덕션 지원 코드가 아닙니다.

Amazon Data Firehose API는 Firehose 스트림에 데이터를 전송하기 위한 두 가지 작업, PutRecord와 PutRecordBatch를 제공합니다. PutRecord()는 한 호출 내에 하나의 데이터 레코드를 보내고 PutRecordBatch()는 한 호출 내에 여러 데이터 레코드를 보낼 수 있습니다.

PutRecord를 이용한 단일 쓰기 작업

데이터 저장에는 Firehose 스트림 이름과 바이트 버퍼(<=1000KB)만 있으면 됩니다. Amazon Data Firehose는 파일을 Amazon S3에 로드하기 전에 여러 레코드를 일괄 처리하기 때문에 레코드 구분 기호를 추가해야 할 수 있습니다. Firehose 스트림에 한 번에 한 레코드씩 데이터를 제공하려면 다음 코드를 사용합니다.

```
PutRecordRequest putRecordRequest = new PutRecordRequest();
putRecordRequest.setDeliveryStreamName(deliveryStreamName);

String data = line + "\n";

Record record = new Record().withData(ByteBuffer.wrap(data.getBytes()));
putRecordRequest.setRecord(record);

// Put record into the DeliveryStream
firehoseClient.putRecord(putRecordRequest);
```

자세한 코드 컨텍스트는 AWS SDK에 포함된 샘플 코드를 참조하세요. 요청 및 응답 구문에 대한 자세한 내용은 Firehose API 작업의 관련 주제를 참조하세요.

PutRecordBatch를 이용한 일괄 쓰기 작업

데이터 저장에는 Firehose 스트림 이름과 레코드 목록만 있으면 됩니다. Amazon Data Firehose는 파일을 Amazon S3에 로드하기 전에 여러 레코드를 일괄 처리하기 때문에 레코드 구분 기호를 추가해야 할 수 있습니다. Firehose 스트림에 레코드 배치로 데이터를 제공하려면 다음 코드를 사용합니다.

```
PutRecordBatchRequest putRecordBatchRequest = new PutRecordBatchRequest();
putRecordBatchRequest.setDeliveryStreamName(deliveryStreamName);
putRecordBatchRequest.setRecords(recordList);

// Put Record Batch records. Max No.Of Records we can put in a
// single put record batch request is 500
firehoseClient.putRecordBatch(putRecordBatchRequest);

recordList.clear();
```

자세한 코드 컨텍스트는 AWS SDK에 포함된 샘플 코드를 참조하세요. 요청 및 응답 구문에 대한 자세한 내용은 Firehose API 작업의 관련 주제를 참조하세요.

Firehose에 CloudWatch Logs 전송

CloudWatch Logs 이벤트는 CloudWatch 구독 필터를 사용하여 Firehose로 전송할 수 있습니다. 자세한 내용은 Amazon Data Firehose를 사용한 구독 필터를 참조하세요.

CloudWatch Logs 이벤트는 압축된 gzip 형식으로 Firehose로 전송됩니다. 압축 해제된 로그 이벤트를 Firehose 대상에 전송하려면 Firehose의 압축 해제 기능을 사용하여 CloudWatch Logs의 압축을 자동으로 해제할 수 있습니다.

Important

Firehose는 현재 Amazon OpenSearch Service 대상으로 CloudWatch Logs를 전송하는 것을 지원하지 않습니다. Amazon CloudWatch는 여러 로그 이벤트를 하나의 Firehose 레코드로 결합하고 Amazon OpenSearch Service는 하나의 레코드로 된 여러 로그 이벤트를 받아들일 수 없기 때문입니다. 대신 CloudWatch Logs에서 Amazon OpenSearch Service에 대한 구독 필터 사용을 고려할 수 있습니다.

CloudWatch Logs 압축 해제

Firehose를 사용하여 CloudWatch Logs를 전송하고 압축 해제된 데이터를 Firehose 스트림 대상으로 전송하려는 경우 Firehose 데이터 형식 변환(Parquet, ORC) 또는 동적 파티셔닝을 사용합니다. Firehose 스트림에 대해 압축 해제를 활성화해야 합니다.

AWS Management Console AWS Command Line Interface 또는 AWS SDKs.

Note

스트림에서 압축 해제 기능을 활성화하는 경우, 해당 스트림은 Vended Logs가 아닌 CloudWatch Logs 구독 필터에만 사용합니다. CloudWatch Logs와 Vended Logs를 모두 수집하는 데 사용되는 스트림에서 압축 해제 기능을 활성화하면 Firehose로의 Vended Logs 수집이 실패합니다. 이 압축 해제 기능은 CloudWatch Logs에만 적용됩니다.

CloudWatch Logs 압축 해제 후 메시지 추출

압축 해제를 활성화하면 메시지 추출도 활성화할 수 있습니다. 메시지 추출을 사용할 때 Firehose는 압축 해제된 CloudWatch Logs 레코드에서 소유자, 로그 그룹, 로그 스트림 등의 모든 메타데이터를 필터링하고 메시지 필드 내의 콘텐츠만 전송합니다. Splunk 대상으로 데이터를 전송하는 경우 Splunk에서 데이터를 구문 분석하기 위해 메시지 추출을 활성화해야 합니다. 다음은 메시지 추출을 사용하거나 사용하지 않은 압축 해제 후의 샘플 출력입니다.

그림 1: 메시지 추출 없이 압축 해제 후 샘플 출력:

```
{
 "owner": "111111111111",
 "logGroup": "CloudTrail/logs",
 "logStream": "111111111111_CloudTrail/logs_us-east-1",
 "subscriptionFilters": [
 "Destination"
 ],
 "messageType": "DATA_MESSAGE",
 "logEvents": [
 {
 "id": "31953106606966983378809025079804211143289615424298221568",
 "timestamp": 1432826855000,
 "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root1\"}"
 },
 {
 "id": "31953106606966983378809025079804211143289615424298221569",
 "timestamp": 1432826855000,
 "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root2\"}"
 },
 {
 "id": "31953106606966983378809025079804211143289615424298221570",
 "timestamp": 1432826855000,
 "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root3\"}"
 }
 ]
}
```

그림 2: 메시지 추출을 사용한 압축 해제 후 샘플 출력:

```
{"eventVersion":"1.03","userIdentity":{"type":"Root1"}
{"eventVersion":"1.03","userIdentity":{"type":"Root2"}
{"eventVersion":"1.03","userIdentity":{"type":"Root3"}
```

콘솔에서 새 Firehose 스트림에 대한 압축 해제 활성화

를 사용하여 새 Firehose 스트림에서 압축 해제를 활성화하려면 AWS Management Console

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Amazon Data Firehose를 선택합니다.
3. Firehose 스트림 생성을 선택합니다.
4. 소스 및 대상 선택에서

   소스

   Firehose 스트림의 소스 다음 중 하나의 소스를 선택합니다.

   - Direct PUT - 생산자 애플리케이션이 직접 쓰는 대상 Firehose 스트림을 생성하려면 이 옵션을 선택합니다. Firehose의 Direct PUT와 통합된 AWS 서비스 및 에이전트와 오픈 소스 서비스 목록은 이 섹션을 참조하세요.
   - Kinesis 스트림: Kinesis 데이터 스트림을 데이터 소스로 사용하는 Firehose 스트림을 구성하려면 이 옵션을 선택합니다. 그러면 Firehose를 사용하여 기존 Kinesis 데이터 스트림에서 간편하게 데이터를 읽고 대상에 로드할 수 있습니다. 자세한 내용은 Writing to Firehose Using Kinesis Data Streams를 참조하세요.

   Destination

   Firehose 스트림의 대상. 다음 중 하나를 선택합니다.

   - Amazon S3
   - Splunk
5. Firehose 스트림 이름에 스트림 이름을 입력합니다.
6. (선택 사항) 레코드 변환에서:

   - Amazon CloudWatch Logs의 소스 레코드 압축 해제 섹션에서 압축 해제 켜기를 선택합니다.
   - 압축 해제 후 메시지 추출을 사용하려면 메시지 추출 켜기를 선택합니다.

기존 Firehose 스트림에서 압축 해제 활성화

이 섹션에서는 기존 Firehose 스트림에서 압축 해제를 활성화하는 지침을 제공합니다. 여기에는 Lambda 처리가 비활성화된 스트림과 Lambda 처리가 이미 활성화된 스트림이라는 두 가지 시나리오가 포함됩니다. 다음 섹션에서는 Lambda 함수 생성 또는 수정, Firehose 설정 업데이트, 내장 Firehose

압축 해제 기능의 성공적인 구현을 보장하기 위한 CloudWatch 지표 모니터링 등 각 사례에 대한 단계별 절차를 간략하게 설명합니다.

Lambda 처리가 비활성화된 경우 압축 해제 활성화

Lambda 처리가 비활성화된 기존 Firehose 스트림에서 압축 해제를 활성화하려면 먼저 Lambda 처리를 활성화해야 합니다. 이 조건은 기존 스트림에만 유효합니다. 다음 단계에서는 Lambda 처리가 활성화되지 않은 기존 스트림에서 압축 해제를 활성화하는 방법을 보여줍니다.

1. Lambda 함수를 생성합니다. 더미 레코드 패스스루를 생성하거나 이 블루프린트를 사용하여 새 Lambda 함수를 생성할 수 있습니다.
2. 현재 Firehose 스트림을 업데이트하여 Lambda 처리를 활성화하고 처리를 위해 생성한 Lambda 함수를 사용합니다.
3. 새 Lambda 함수로 스트림을 업데이트한 후 Firehose 콘솔로 돌아가 압축 해제를 활성화합니다.
4. 1단계에서 활성화한 Lambda 처리를 비활성화합니다. 이제 1단계에서 생성한 기능을 삭제할 수 있습니다.

Lambda 처리가 활성화된 경우 압축 해제 활성화

압축 해제를 수행하기 위해 Lambda 함수가 있는 Firehose 스트림이 이미 있는 경우 Firehose 압축 해제 기능으로 바꿀 수 있습니다. 계속하기 전에 Lambda 함수 코드를 검토하여 압축 해제 또는 메시지 추출만 수행하는지 확인합니다. Lambda 함수의 출력은 그림 1 또는 그림 2에 표시된 예제와 비슷해야 합니다. 출력이 비슷한 경우 다음 단계를 사용하여 Lambda 함수를 교체할 수 있습니다.

1. 현재 Lambda 함수를 이 블루프린트로 바꿉니다. 새로운 블루프린트 Lambda 함수는 수신 데이터가 압축되었는지 압축 해제되었는지 자동으로 감지합니다. 입력 데이터가 압축된 경우에만 압축 해제를 수행합니다.
2. 압축 해제를 위해 내장 Firehose 옵션을 사용하여 압축 해제를 켭니다.
3. Firehose 스트림이 아직 활성화되지 않은 경우 CloudWatch 지표를 활성화합니다. 지표 `CloudWatchProcessorLambda_IncomingCompressedData`를 모니터링하고이 지표가 0으로 변경될 때까지 기다립니다. 이렇게 하면 Lambda 함수로 전송된 모든 입력 데이터가 압축 해제되고 Lambda 함수가 더 이상 필요하지 않습니다.
4. Lambda 데이터 변환은 더 이상 스트림 압축을 풀 필요가 없으므로 제거합니다.

Firehose 스트림에서 압축 해제 비활성화

를 사용하여 데이터 스트림에서 압축 해제를 비활성화하려면 AWS Management Console

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/kinesis Kinesis 콘솔을 엽니다.
2. 탐색 창에서 Amazon Data Firehose를 선택합니다.
3. 편집하려는 Firehose 스트림을 선택합니다.
4. Firehose 스트림 세부 정보 페이지에서 구성 탭을 선택합니다.
5. 레코드 변환 및 변환 섹션에서 편집을 선택합니다.
6. Amazon CloudWatch Logs에서 소스 레코드 압축 해제에서 압축 해제 켜기를 해제한 다음 변경 사항 저장을 선택합니다.

Firehose의 압축 해제 문제 해결

다음 표는 Firehose가 데이터 압축 해제 및 처리 중에 오류 S3 버킷으로 레코드 전송, 오류 로깅, 지표 전송을 포함하여 오류를 처리하는 방법을 보여줍니다. 또한 승인되지 않은 데이터 풋 작업에 대해 반환된 오류 메시지도 설명합니다.

문제	Solution
압축 해제 중에 오류가 발생할 경우 소스 데이터는 어떻게 되나요?	Amazon Data Firehose가 레코드의 압축을 풀 수 없는 경우 레코드는 Firehose 스트림 생성 시간 동안 지정한 오류 S3 버킷에 있는 그대로(압축된 형식) 전달됩니다. 레코드와 함께 전달된 객체에는 오류 코드와 오류 메시지도 포함되며 이러한 객체는 `decompression-failed` 라는 S3 버킷 접두사로 전달됩니다. Firehose는 레코드의 압축 해제 실패 후에도 다른 레코드를 계속 처리합니다.
압축 해제 성공 후 처리 파이프라인에 오류가 발생할 경우 소스 데이터는 어떻게 됩니까?	동적 분할 및 데이터 형식 변환과 같은 압축 해제 후 처리 단계에서 Amazon Data Firehose 오류가 발생하는 경우 레코드는 Firehose 스트림 생성 시간 동안 지정한 오류 S3 버킷으로 압축된 형식으로 전송됩니다. 전송된 객체에는 레코드와 함께 오류 코드 및 오류 메시지도 포함됩니다.

문제	Solution
오류 또는 예외가 발생할 경우 어떻게 알 수 있나요?	압축 해제 중에 오류 또는 예외가 발생하는 경우 CloudWatch Logs를 구성하면 Firehose는 오류 메시지를 CloudWatch Logs에 기록합니다. 또한 Firehose는 모니터링할 수 있는 CloudWatch 지표로 지표를 전송합니다. Firehose에서 내보낸 지표를 기반으로 경보를 선택적으로 생성할 수도 있습니다.
put 작업이 CloudWatch Logs에서 오지 않으면 어떻게 되나요?	puts 고객이 CloudWatch Logs에서 오지 않으면 다음 오류 메시지가 반환됩니다. `Put to Firehose failed for AccountId: <accountID>, FirehoseName: <firehosename> because the request is not originating from allowed source types.`
Firehose는 압축 해제 기능을 위해 어떤 지표를 방출합니까?	Firehose는 모든 레코드의 압축 해제에 대한 지표를 내보냅니다. 기간(1분), 통계(합계), 날짜 범위를 선택하여 DecompressedRecords 실패 또는 성공 또는 DecompressedBytes 실패 또는 성공의 수를 구해야 합니다. 자세한 내용은 CloudWatch Logs 압축 해제 지표 섹션을 참조하세요.

Firehose에 CloudWatch Events 전송

CloudWatch Events 규칙에 대상을 추가하여 Firehose 스트림에 이벤트를 보내도록 Amazon CloudWatch를 구성할 수 있습니다.

기존 Firehose 스트림에 이벤트를 보내는 CloudWatch Events 규칙의 대상 만드는 방법

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 규칙 생성을 선택합니다.
3. 1단계: 규칙 생성 페이지에서 대상에 대해 대상 추가를 선택한 후 Firehose 스트림을 선택합니다.
4. 기존 Firehose 스트림을 선택합니다.

CloudWatch Events 규칙 생성에 대한 자세한 내용은 Amazon CloudWatch Events 시작하기를 참조하세요.

Firehose AWS IoT 로 데이터를 전송하도록 구성

작업을 추가하여 Firehose 스트림에 정보를 보내 AWS IoT 도록를 구성할 수 있습니다.

기존 Firehose 스트림으로 이벤트를 보내는 작업을 생성하려면

1. AWS IoT 콘솔에서 규칙을 생성할 때 규칙 생성 페이지의 하나 이상의 작업 설정에서 작업 추가를 선택합니다.
2. Amazon Kinesis Firehose 스트림으로 메시지 전송을 선택합니다.
3. 작업 구성을 선택합니다.
4. 스트림 이름에 대해 기존 Firehose 스트림을 선택합니다.
5. [Separator]에 대해 레코드 사이에 삽입할 구분자 문자를 선택합니다.
6. IAM 역할 이름에 대해서는 기존 IAM 역할을 선택하거나 새 역할 생성을 선택합니다.
7. 작업 추가를 선택합니다.

AWS IoT 규칙 생성에 대한 자세한 내용은 AWS IoT 규칙 자습서를 참조하세요.

Amazon Data Firehose에서 소스 데이터 변환

Amazon Data Firehose는 Lambda 함수를 호출해 수신되는 소스 데이터를 변환하고, 변환된 데이터를 대상으로 전송할 수 있습니다. Firehose 스트림을 생성할 때 Amazon Data Firehose 데이터 변환을 활성화할 수 있습니다.

데이터 변환 흐름 이해

Firehose 데이터 변환을 활성화하면 Firehose는 수신 데이터를 버퍼링합니다. 버퍼링 사이즈 힌트 범위는 0.2MB~3MB입니다. 기본 Lambda 버퍼링 크기 힌트는 Splunk 및 Snowflake를 제외한 모든 대상에 대해 1MB입니다. Splunk 및 Snowflake의 경우 기본 버퍼링 힌트는 256KB입니다. Lambda 버퍼링 간격 힌트 범위는 0~900초입니다. 기본 Lambda 버퍼링 간격 힌트는 Snowflake를 제외한 모든 대상에 대해 60초입니다. Snowflake의 경우 기본 버퍼링 힌트 간격은 30초입니다. 버퍼링 크기를 조정하려면 CreateDeliveryStream 또는 UpdateDestination API의 ProcessingConfiguration 파라미터를 `BufferSizeInMBs` 및 `IntervalInSeconds`라는 ProcessorParameter로 설정합니다. 그런 다음 Firehose는 AWS Lambda 동기식 호출 모드를 사용하여 버퍼링된 각 배치와 동기식으로 지정된 Lambda 함수를 호출합니다. 변환된 데이터는 Lambda로부터 Firehose로 전송됩니다. 이후 지정된 대상 버퍼링 크기 또는 버퍼링 간격 중 먼저 발생하는 값에 도달하면, Firehose가 이를 대상으로 전송합니다.

> **Important**
>
> Lambda 동기식 호출 모드는 요청 및 응답 모두 페이로드 크기 제한이 6MB입니다. 함수로 요청을 전송하기 위한 버퍼링 크기가 6MB 이하인지 확인해야 합니다. 또한 함수에서 반환하는 응답이 6MB를 초과하지 않는지 확인합니다.

Lambda 간접 호출 기간

Amazon Data Firehose는 Lambda 호출 시간을 최대 5분까지 지원합니다. Lambda 함수를 완료하는 데 5분 이상 걸리는 경우 다음 오류가 발생합니다. Firehose가 AWS Lambda를 호출할 때 시간 초과 오류가 발생했습니다. 지원되는 함수 제한 시간은 최대 5분입니다.

이러한 오류가 발생하는 경우 Amazon Data Firehose의 작동에 대한 자세한 내용은 the section called “데이터 변환 실패 처리” 섹션을 참조하세요.

데이터 변환에 필요한 파라미터

Lambda의 모든 변환된 레코드에는 다음 파라미터가 포함되어 있어야 합니다. 그렇지 않으면 Amazon Data Firehose가 이를 거부하고 데이터 변환 실패로 간주합니다.

For Kinesis Data Streams and Direct PUT

Lambda에서 변환된 모든 레코드에는 다음 파라미터가 필요합니다.

- `recordId` – 레코드 ID는 호출 중에 Amazon Data Firehose에서 Lambda로 전달됩니다. 변환된 레코드에는 동일한 레코드 ID가 포함되어야 합니다. 원래 레코드의 ID와 변환된 레코드의 ID 간 불일치는 데이터 변환 실패로 간주됩니다.
- `result` – 레코드의 데이터 변환 상태입니다. 가능한 값은 `Ok`(레코드가 성공적으로 변환되었음), `Dropped`(처리 로직에 의해 의도적으로 레코드가 삭제됨), `ProcessingFailed`(레코드를 변환하지 못함)입니다. 레코드의 상태가 `Ok` 또는 `Dropped`이면, Amazon Data Firehose는 성공적으로 처리된 것으로 간주합니다. 그렇지 않으면 Amazon Data Firehose는 이를 처리 실패로 간주합니다.
- `data` – base64 인코딩 후 변환된 데이터 페이로드입니다.

  다음은 Lambda 결과 출력 예입니다.

```
{
    "recordId": "<recordId from the Lambda input>",
    "result": "Ok",
    "data": "<Base64 encoded Transformed data>"
}
```

For Amazon MSK

Lambda에서 변환된 모든 레코드에는 다음 파라미터가 필요합니다.

- `recordId` – 레코드 ID는 호출 중에 Firehose에서 Lambda로 전달됩니다. 변환된 레코드에는 동일한 레코드 ID가 포함되어야 합니다. 원래 레코드의 ID와 변환된 레코드의 ID 간 불일치는 데이터 변환 실패로 간주됩니다.
- `result` – 레코드의 데이터 변환 상태입니다. 가능한 값은 `Ok`(레코드가 성공적으로 변환되었음), `Dropped`(처리 로직에 의해 의도적으로 레코드가 삭제됨), `ProcessingFailed`(레코드를 변환하지 못함)입니다. 레코드 상태가 `Ok` 또는 `Dropped`인 경우 Firehose는 성공적으로 처리된 것으로 간주합니다. 그렇지 않으면 Firehose는 이를 처리 실패로 간주합니다.

- `KafkaRecordValue` – base64 인코딩 후 변환된 데이터 페이로드입니다.

  다음은 Lambda 결과 출력 예입니다.

```
{
    "recordId": "<recordId from the Lambda input>",
    "result": "Ok",
    "kafkaRecordValue": "<Base64 encoded Transformed data>"
}
```

지원되는 Lambda 청사진

이러한 블루프린트는 AWS Lambda 함수를 생성하고 사용하여 Amazon Data Firehose 데이터 스트림의 데이터를 변환하는 방법을 보여줍니다.

AWS Lambda 콘솔에서 사용할 수 있는 블루프린트를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. 함수 생성을 선택한 다음 Use a blueprint(블루프린트 사용)를 선택합니다.
3. 블루프린트 필드에서 키워드 `firehose`를 검색하여 Amazon Data Firehose Lambda 블루프린트를 찾습니다.

블루프린트 목록:

- Amazon Data Firehose 스트림으로 전송된 레코드 처리(Node.js, Python)

  이 블루프린트는 AWS Lambda를 사용하여 Firehose 데이터 스트림의 데이터를 처리하는 방법의 기본 예제를 보여줍니다.

  최신 릴리스 날짜: 2016년 11월.

  릴리스 노트: 없음.

- Firehose로 전송된 CloudWatch 로그 처리

  이 청사진은 더 이상 사용되지 않습니다. 이 블루프린트를 사용하지 마세요. 압축 해제된 CloudWatch Logs 데이터가 6MB(Lambda 제한)를 초과하는 경우 요금이 많이 발생할 수 있습니다.

Firehose로 전송된 CloudWatch Logs 처리에 대한 자세한 내용은 CloudWatch Logs를 사용하여 Firehose에 쓰기를 참조하세요.

- Syslog 형식의 Amazon Data Firehose 스트림 레코드를 JSON(Node.js)으로 변환

  이 블루프린트는 RFC3164 Syslog 형식의 입력 레코드를 JSON 형식으로 변환하는 방법을 보여줍니다.

  최신 릴리스 날짜: 2016년 11월.

  릴리스 노트: 없음.

에서 사용할 수 있는 블루프린트를 보려면 AWS Serverless Application Repository

1. AWS Serverless Application Repository로 이동합니다.
2. 모든 애플리케이션 검색을 선택하세요.
3. 애플리케이션 필드에서 키워드 `firehose`를 검색합니다.

블루프린트를 사용하지 않고 Lambda 함수를 만들 수도 있습니다. AWS Lambda 시작하기를 참조하세요.

데이터 변환 실패 처리

네트워크 제한 시간이나 Lambda 호출 제한에 도달해 Lambda 함수 호출에 실패하는 경우, Amazon Data Firehose는 기본값인 3회의 호출을 재시도합니다. 호출에 성공하지 못하면 Amazon Data Firehose가 해당 레코드 배치를 건너뜁니다. 건너뛴 레코드는 제대로 처리되지 않은 레코드로 간주됩니다. CreateDeliveryStream 또는 `UpdateDestination` API를 사용하여 다시 시도 옵션을 정의하거나 재정의할 수 있습니다. 이러한 유형의 실패에 대해 Amazon CloudWatch Logs에 호출 오류를 기록할 수 있습니다. 자세한 내용은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.

레코드의 데이터 변환 상태가 `ProcessingFailed`인 경우, Amazon Data Firehose는 레코드를 제대로 처리되지 않은 것으로 간주합니다. 이러한 유형의 실패에 대해 Lambda 함수에서 Amazon CloudWatch Logs에 오류 로그를 출력할 수 있습니다. 더 자세한 내용은 AWS Lambda 개발자 안내서의 AWS Lambda용 Accessing Amazon CloudWatch Logs를 참조하세요.

데이터 변환에 실패하면 제대로 처리되지 않은 레코드는 S3 버킷의 `processing-failed` 폴더로 전송됩니다. 이 레코드는 다음 형식을 취합니다.

```
{
    "attemptsMade": "count",
    "arrivalTimestamp": "timestamp",
    "errorCode": "code",
    "errorMessage": "message",
    "attemptEndingTimestamp": "timestamp",
    "rawData": "data",
    "lambdaArn": "arn"
}
```

`attemptsMade`

시도한 호출 요청 횟수입니다.

`arrivalTimestamp`

Amazon Data Firehose가 레코드를 수신한 시간입니다.

`errorCode`

Lambda가 반환한 HTTP 오류 코드.

`errorMessage`

Lambda가 반환한 오류 메시지.

`attemptEndingTimestamp`

Amazon Data Firehose가 Lambda 호출 시도를 중단한 시간입니다.

`rawData`

base64 인코딩된 레코드 데이터입니다.

`lambdaArn`

Lambda 함수의 Amazon 리소스 이름(ARN)입니다.

소스 레코드 백업

Amazon Data Firehose는 변환된 레코드를 대상으로 전송하는 동시에, 변환되지 않은 모든 레코드를 S3 버킷에 백업할 수 있습니다. Firehose 스트림을 생성하거나 업데이트할 때 소스 레코드 백업을 활성화할 수 있습니다. 소스 레코드 백업을 활성화한 후 비활성화할 수 없습니다.

Amazon Data Firehose의 파티션 스트리밍 데이터

동적 파티셔닝을 사용하면 데이터 내의 키(예: `customer_id` 또는 `transaction_id`)를 사용하여 Firehose의 스트리밍 데이터를 지속적으로 분할한 다음 이 키에 의해 그룹화된 데이터를 해당되는 Amazon Simple Storage Service(Amazon S3) 접두사로 전달할 수 있습니다. 따라서 Amazon Athena, Amazon EMR, Amazon Redshift Spectrum, Amazon QuickSight와 같은 다양한 서비스를 사용하여 Amazon S3의 스트리밍 데이터에 대한 고성능 비용 효율적인 분석을 더 쉽게 실행할 수 있습니다. 또한 추가 처리가 필요한 사용 사례에서 동적으로 분할된 스트리밍 데이터가 Amazon S3로 전송된 후 AWS Glue는 보다 정교한 추출, 변환 및 로드(ETL) 작업을 수행할 수 있습니다.

데이터 파티셔닝을 통해 스캔되는 데이터 양을 최소화하고 성능을 최적화하며 Amazon S3의 분석 쿼리 비용을 절감할 수 있습니다. 또한 데이터에 대한 세분화된 액세스를 향상시킵니다. Firehose 스트림은 일반적으로 데이터를 캡처하여 Amazon S3로 로드하기 위해 사용됩니다. Amazon S3 기반 분석을 위해 스트리밍 데이터 세트를 분할하려면, 분석에 데이터를 제공하기에 앞서 Amazon S3 버킷 간에 파티셔닝 애플리케이션을 실행해야 하는데, 이는 복잡하거나 비용이 많이 들 수 있습니다.

Firehose는 동적 파티셔닝을 통해 동적 또는 정적으로 정의된 데이터 키를 사용하여 전송 중인 데이터를 지속적으로 그룹화하고, 해당 데이터를 키별 Amazon S3 접두사에 전달합니다. 이렇게 하면 인사이트 획득 시간을 몇 분 또는 몇 시간 단축할 수 있습니다. 또한 비용을 절감하고 아키텍처를 단순화할 수 있습니다.

주제

Amazon Data Firehose에서 동적 파티셔닝 활성화

Amazon Data Firehose Management Console, CLI 또는 API를 통해 Firehose 스트림의 동적 파티셔닝을 구성할 수 있습니다.

> **Important**
>
> 동적 파티셔닝은 새 Firehose 스트림을 생성할 때만 활성화할 수 있습니다. 동적 파티셔닝이 미리 활성화되지 않은 기존 Firehose 스트림에 대해서는 동적 파티셔닝을 활성화할 수 없습니다.

새 Firehose 스트림 생성 시 Firehose 관리 콘솔을 통해 동적 파티셔닝을 활성화하고 구성하는 방법에 대한 자세한 과정은 Amazon Firehose 스트림 생성을 참조하세요. Firehose 스트림의 대상을 지정하는 작업을 수행할 때는 대상 설정 구성 섹션의 단계를 따르세요. 현재 동적 파티셔닝은 Amazon S3를 대상으로 사용하는 Firehose 스트림에만 지원되기 때문입니다.

활성 Firehose 스트림에 대한 동적 파티셔닝이 활성화되면, 새 파티션 키와 S3 접두사 표현식을 추가하거나 또는 기존 파티션 키 및 S3 접두사 표현식을 제거 또는 업데이트하여 구성을 업데이트할 수 있습니다. 업데이트되고 나면 Firehose가 새 키 및 새 S3 접두사 표현식을 사용하기 시작합니다.

> **Important**
>
> Firehose 스트림에 동적 파티셔닝이 활성화되고 나면 이 Firehose 스트림에서 동적 파티셔닝을 비활성화할 수는 없습니다.

파티셔닝 키 이해

동적 파티셔닝을 사용하여, 파티션 키를 기반으로 데이터를 분할하고 스트리밍 S3 데이터에서 대상 데이터 세트를 생성합니다. 파티션 키를 사용하면 특정 값에 기반하여 스트리밍 데이터를 필터링할 수 있습니다. 예를 들어, 고객 ID 및 국가를 기준으로 데이터를 필터링해야 하는 경우 `customer_id`의 데이터 필드를 하나의 파티션 키로 지정하고 `country`의 데이터 필드는 또 다른 파티션 키로 지정할 수 있습니다. 그런 다음 표현식을 (지원되는 형식을 사용해) 지정하여 동적으로 파티셔닝된 데이터 레코드를 전송할 S3 버킷 접두사를 정의합니다.

다음 방법을 사용하여 파티셔닝 키를 생성할 수 있습니다.

- 인라인 구문 분석 - 이 방법은 Firehose의 내장 지원 메커니즘인 jq 구문 분석기를 사용하여 JSON 형식의 데이터 레코드에서 파티션 키를 추출합니다. 현재는 `jq 1.6` 버전만 지원합니다.
- AWS Lambda 함수 -이 메서드는 지정된 AWS Lambda 함수를 사용하여 파티셔닝에 필요한 데이터 필드를 추출하고 반환합니다.

> **Important**
>
> 동적 파티셔닝을 사용할 경우, 이러한 방법 중 하나 이상을 구성하여 데이터를 분할하도록 해야 합니다. 두 방법 중 하나를 구성하여 파티션 키를 지정하거나 두 방법을 동시에 지정할 수 있습니다.

인라인 구문 분석 방법으로 파티션 키 만들기

인라인 구문 분석 방법으로 스트리밍 데이터의 동적 파티셔닝을 구성하려면, 파티션 키로 사용할 데이터 레코드 파라미터를 선택하고 지정된 각 파티션 키의 값을 입력해야 합니다.

다음 샘플 데이터 레코드는 인라인 구문 분석을 사용하여 레코드에 대한 파티션 키를 정의하는 방법을 보여줍니다. 데이터는 Base64 형식으로 인코딩되어야 합니다. CLI 예제를 참조할 수도 있습니다.

```
{
   "type": {
    "device": "mobile",
    "event": "user_clicked_submit_button"
  },
  "customer_id": "1234567890",
  "event_timestamp": 1565382027,   #epoch timestamp
  "region": "sample_region"
}
```

예를 들면 `customer_id` 파라미터 또는 `event_timestamp` 파라미터를 기반으로 데이터를 분할하도록 선택할 수 있습니다. 즉, 각 레코드의 `customer_id` 파라미터 또는 `event_timestamp` 파라미터의 값을 사용하여 레코드가 전송될 S3 접두사를 결정하는 것입니다. `.type.device` 표현식이 있는 `device`와(과) 같이 중첩된 파라미터를 선택할 수도 있습니다. 동적 파티셔닝 로직은 여러 가지 파라미터에 따라 달라질 수 있습니다.

파티션 키의 데이터 파라미터를 선택한 다음 각 파라미터를 유효한 jq 표현식으로 매핑합니다. 다음 표에는 파라미터를 jq 표현식으로 매핑한 내용이 나와 있습니다.

파라미터	jq 표현식
`customer_id`	.customer_id
`device`	.type.device

파라미터	jq 표현식
year	.event_timestamp| strftime("%Y")
month	.event_timestamp| strftime("%m")
day	.event_timestamp| strftime("%d")
hour	.event_timestamp| strftime("%H")

런타임에서, Firehose는 위의 오른쪽 열을 사용하여 각 레코드의 데이터를 기준으로 파라미터를 평가합니다.

AWS Lambda 함수를 사용하여 파티셔닝 키 생성

압축 또는 암호화된 데이터 레코드 또는 JSON 이외의 파일 형식의 데이터의 경우 통합 AWS Lambda 함수를 사용자 지정 코드와 함께 사용하여 레코드를 압축 해제, 복호화 또는 변환하여 파티셔닝에 필요한 데이터 필드를 추출하고 반환할 수 있습니다. 이는 현재 Firehose와 함께 사용 가능한 기존 변환 Lambda 함수의 확장 기능입니다. 해당 데이터 필드를 변환, 구문 분석, 반환한 다음 동일한 Lambda 함수를 사용하여 동적 파티셔닝에 사용할 수 있습니다.

다음은 Python으로 된 Firehose 스트림 처리 Lambda 함수의 예시로, 입력에서 출력까지 모든 읽기 레코드를 재생하고 레코드에서 파티셔닝 키를 추출합니다.

```
from __future__ import print_function
import base64
import json
import datetime

# Signature for all Lambda functions that user must implement
def lambda_handler(firehose_records_input, context):
    print("Received records for processing from DeliveryStream: " +
 firehose_records_input['deliveryStreamArn']
          + ", Region: " + firehose_records_input['region']
          + ", and InvocationId: " + firehose_records_input['invocationId'])

    # Create return value.
    firehose_records_output = {'records': []}

    # Create result object.
```

```
    # Go through records and process them

    for firehose_record_input in firehose_records_input['records']:
        # Get user payload
        payload = base64.b64decode(firehose_record_input['data'])
        json_value = json.loads(payload)

        print("Record that was received")
        print(json_value)
        print("\n")
        # Create output Firehose record and add modified payload and record ID to it.
        firehose_record_output = {}
        event_timestamp = datetime.datetime.fromtimestamp(json_value['eventTimestamp'])
        partition_keys = {"customerId": json_value['customerId'],
                          "year": event_timestamp.strftime('%Y'),
                          "month": event_timestamp.strftime('%m'),
                          "day": event_timestamp.strftime('%d'),
                          "hour": event_timestamp.strftime('%H'),
                          "minute": event_timestamp.strftime('%M')
                          }

        # Create output Firehose record and add modified payload and record ID to it.
        firehose_record_output = {'recordId': firehose_record_input['recordId'],
                                  'data': firehose_record_input['data'],
                                  'result': 'Ok',
                                  'metadata': { 'partitionKeys': partition_keys }}

        # Must set proper record ID
        # Add the record to the list of output records.

        firehose_records_output['records'].append(firehose_record_output)

    # At the end return processed records
    return firehose_records_output
```

다음은 Go로 된 Firehose 스트림 처리 Lambda 함수의 예시로, 입력에서 출력까지 모든 읽기 레코드를 재생하고 레코드에서 파티셔닝 키를 추출합니다.

```
package main

import (
 "fmt"
```

```
 "encoding/json"
 "time"
 "strconv"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type DataFirehoseEventRecordData struct {
 CustomerId string `json:"customerId"`
}

func handleRequest(evnt events.DataFirehoseEvent) (events.DataFirehoseResponse, error)
 {

 fmt.Printf("InvocationID: %s\n", evnt.InvocationID)
 fmt.Printf("DeliveryStreamArn: %s\n", evnt.DeliveryStreamArn)
 fmt.Printf("Region: %s\n", evnt.Region)

 var response events.DataFirehoseResponse

 for _, record := range evnt.Records {
  fmt.Printf("RecordID: %s\n", record.RecordID)
  fmt.Printf("ApproximateArrivalTimestamp: %s\n", record.ApproximateArrivalTimestamp)

  var transformedRecord events.DataFirehoseResponseRecord
  transformedRecord.RecordID = record.RecordID
  transformedRecord.Result = events.DataFirehoseTransformedStateOk
  transformedRecord.Data = record.Data

  var metaData events.DataFirehoseResponseRecordMetadata
  var recordData DataFirehoseEventRecordData
  partitionKeys := make(map[string]string)

  currentTime := time.Now()
  json.Unmarshal(record.Data, &recordData)
  partitionKeys["customerId"] = recordData.CustomerId
  partitionKeys["year"] = strconv.Itoa(currentTime.Year())
  partitionKeys["month"] = strconv.Itoa(int(currentTime.Month()))
  partitionKeys["date"] = strconv.Itoa(currentTime.Day())
  partitionKeys["hour"] = strconv.Itoa(currentTime.Hour())
  partitionKeys["minute"] = strconv.Itoa(currentTime.Minute())
  metaData.PartitionKeys = partitionKeys
  transformedRecord.Metadata = metaData
```

```
  response.Records = append(response.Records, transformedRecord)
 }

 return response, nil
}

func main() {
 lambda.Start(handleRequest)
}
```

Amazon S3 버킷 접두사를 사용하여 데이터 전송

Amazon S3를 대상으로 사용하는 Firehose 스트림을 만들 때는 Firehose가 데이터를 전송할 Amazon S3 버킷을 지정해야 합니다. Amazon S3 버킷 접두사를 사용하여 S3 버킷에 저장할 데이터를 구성할 수 있습니다. Amazon S3 버킷 접두사는 유사한 객체를 함께 그룹화하는 데 사용하는 디렉터리와 유사합니다.

동적 파티셔닝을 통해 파티셔닝된 데이터는 지정된 Amazon S3 접두사로 전달됩니다. 동적 파티셔닝을 활성화하지 않는 경우 Firehose 스트림에 대한 S3 버킷 접두사를 지정하는 것은 선택 사항입니다. 단, 동적 파티셔닝을 활성화한다면 Firehose가 파티셔닝된 데이터를 전송할 S3 버킷 접두사를 반드시 지정해야 합니다.

동적 파티셔닝을 활성화하는 모든 Firehose 스트림에서 S3 버킷 접두사 값은 해당 Firehose 스트림에 대해 지정된 파티션 키에 기반한 표현식으로 구성됩니다. 위의 데이터 레코드 예시를 다시 사용하여, 다음과 같이 위에서 정의된 파티션 키에 기반한 표현식으로 구성되는 S3 접두사 값을 만들 수 있습니다.

```
"ExtendedS3DestinationConfiguration": {
"BucketARN": "arn:aws:s3:::my-logs-prod",
"Prefix": "customer_id=!{partitionKeyFromQuery:customer_id}/
    device=!{partitionKeyFromQuery:device}/
    year=!{partitionKeyFromQuery:year}/
    month=!{partitionKeyFromQuery:month}/
    day=!{partitionKeyFromQuery:day}/
    hour=!{partitionKeyFromQuery:hour}/"
}
```

Firehose는 런타임 시 위의 표현식을 평가합니다. 동일하게 평가된 S3 접두사 표현식과 일치하는 레코드를 단일 데이터 세트로 그룹화합니다. 이후 Firehose는 각 데이터 세트를 평가된 S3 접두사에 전달합니다. S3로 데이터 세트를 전송하는 주기는 Firehose 스트림 버퍼 설정에 따라 결정됩니다. 따라서 이 예시의 레코드는 다음 S3 객체 키로 전달됩니다.

```
s3://my-logs-prod/customer_id=1234567890/device=mobile/year=2019/month=08/day=09/
hour=20/my-delivery-stream-2019-08-09-23-55-09-a9fa96af-e4e4-409f-bac3-1f804714faaa
```

동적 파티셔닝의 경우, S3 버킷 접두사에 다음 표현식 형식을 사용해야 합니다: `!{namespace:value}`, 여기서 네임스페이스는 `partitionKeyFromQuery` 또는 `partitionKeyFromLambda`이거나, 둘 다일 수 있습니다. 인라인 구문 분석을 사용하여 소스 데이터에 대한 파티션 키를 생성하는 경우 다음 형식으로 지정된 표현식으로 구성되는 S3 버킷 접두사 값을 지정해야 합니다: `"partitionKeyFromQuery:keyID"`. AWS Lambda 함수를 사용하여 소스 데이터에 대한 파티션 키를 생성하는 경우 다음 형식으로 지정된 표현식으로 구성되는 S3 버킷 접두사 값을 지정해야 합니다: `"partitionKeyFromLambda:keyID"`.

> **Note**
>
> Hive 스타일 형식을 사용하여 S3 버킷 접두사 값을 지정할 수도 있습니다(예: customer_id=!{PartitionKeyFromQuery:Customer_ID}.

자세한 내용은 Amazon Firehose 스트림 생성 및 Amazon S3 객체의 사용자 지정 접두사의 “대상에 대한 Amazon S3 선택”을 참조하세요.

Amazon S3에 데이터 전송 시 새 줄 구분 기호 추가

새 줄 구분 기호를 활성화하여 Amazon S3에 전달되는 객체의 레코드 사이에 새 줄 구분 기호를 추가할 수 있습니다. Amazon S3의 객체를 구문 분석하는 데 유용합니다. 특히 집계된 데이터에 동적 파티셔닝을 적용할 때도 유용합니다. 멀티레코드 분해(동적 파티셔닝 수행 전에 집계 데이터에 적용해야 함) 시 구문 분석 프로세스 중에 레코드에서 새로운 줄이 제거되기 때문입니다.

집계 데이터에 동적 파티셔닝 추가

집계된 데이터(예: 단일 `PutRecord` 및 `PutRecordBatch` API 호출로 집계된 여러 이벤트, 로그 또는 레코드)에 동적 파티셔닝을 적용할 수 있으며, 단 이러한 데이터는 우선 분해되어야 합니다. 다중 레코

드 분해(Firehose 스트림의 레코드를 구문 분석하고 분리하는 프로세스)를 활성화하여 데이터를 분해할 수 있습니다.

다중 레코드 분해는 JSON 유형일 수 있습니다. 즉, 일련의 JSON을 개체를 기반으로 레코드가 분리됩니다. 분해는 또한 Delimited 유형일 수도 있습니다. 즉, 지정된 사용자 지정 구분 기호를 기반으로 레코드 분리가 수행됩니다. 이러한 사용자 지정 구분 기호는 Base-64로 인코딩된 문자열이어야 합니다. 예를 들어 다음 문자열을 사용자 지정 구분 기호 ####으로 사용하려면 이 문자열을 IyMjIw==로 변환하는 Base-64 인코딩 형식으로 지정해야 합니다. JSON 또는 구분 기호별 레코드 집계 해제는 레코드당 500개로 제한됩니다.

> **Note**
>
> JSON 레코드의 집계를 해제할 때는 입력이 지원되는 JSON 형식으로 제공되어야 합니다. JSON 객체는 구분 기호가 없거나 신규 줄로 구분(JSONL)되지 않은 단일 줄에 있어야 합니다. JSON 객체 배열은 유효한 입력이 아닙니다.
> 올바른 입력의 예: `{"a":1}{"a":2} and {"a":1}\n{"a":2}`
> 잘못된 입력의 예: `[{"a":1}, {"a":2}]`

집계된 데이터를 사용하여 동적 파티셔닝을 활성화하면 Firehose가 레코드를 구문 분석하여 지정된 다중 레코드 분해 유형에 따라 각 API 호출 내에서 유효한 JSON 객체 또는 구분된 레코드를 찾습니다.

> **Important**
>
> 데이터가 집계된 경우, 먼저 데이터가 분해된 경우에만 동적 파티셔닝을 적용할 수 있습니다.

> **Important**
>
> Firehose에서 Data Transformation 기능을 사용하는 경우 Data Transformation 전에 분해가 적용됩니다. Firehose로 들어오는 데이터는 Deaggregation(분해) → Lambda를 통한 Data Transformation(데이터 변환) → Partitioning Keys(파티션 키)의 순서로 처리됩니다.

동적 파티셔닝 오류 문제 해결

Amazon Data Firehose가 Firehose 스트림의 데이터 레코드를 구문 분석할 수 없거나 지정된 파티션 키를 추출하지 못하거나 S3 접두사 값에 포함된 표현식을 평가하지 못하는 경우, 이 데이터 레코드는

동적 파티셔닝이 활성화되는 전송 스트림을 생성할 때 지정해야 하는 S3 오류 버킷 접두사로 전송됩니다. S3 오류 버킷 접두사에는 Firehose가 지정된 S3 대상으로 전송할 수 없는 모든 레코드가 포함됩니다. 이러한 레코드는 오류 유형에 따라 정리됩니다. 전송된 객체에는 해당 레코드와 함께, 오류를 파악하고 해결하는 데 도움이 되는 오류 관련 정보도 포함됩니다.

이 Firehose 스트림에 대해 동적 파티셔닝을 활성화하려면 Firehose 스트림에 대해 S3 오류 버킷 접두사를 반드시 지정해야 합니다. Firehose 스트림에 대해 동적 파티셔닝을 활성화하지 않는 경우 S3 오류 버킷 접두사를 지정하는 것은 선택 사항입니다.

동적 파티셔닝을 위한 버퍼 데이터

Amazon Data Firehose는 수신되는 스트리밍 데이터를 지정된 대상으로 전송하기 전에 수신 스트리밍 데이터를 특정 크기로 특정 기간 동안 버퍼링합니다. 새 Firehose 스트림을 생성하면서 버퍼 크기와 버퍼 간격을 구성하거나, 기존 Firehose 스트림의 버퍼 크기 및 버퍼 간격을 업데이트할 수 있습니다. 버퍼 크기는 MB 단위로 측정하며, 버퍼 간격은 초 단위로 측정합니다.

> **Note**
>
> 동적 파티셔닝에는 무버퍼링 기능을 사용할 수 없습니다.

동적 파티셔닝이 활성화되면 Firehose는 구성된 버퍼링 힌트(크기 및 시간)에 따라 지정된 파티션에 속하는 레코드를 내부적으로 버퍼링한 다음 이 레코드를 Amazon S3 버킷으로 전송합니다. Firehose는 최대 크기의 객체를 전송하기 위해 내부에서 다단계 버퍼링을 사용합니다. 그에 따라, 레코드 배치의 종단 간 지연은 구성된 버퍼링 힌트 시간의 1.5배가 될 수 있습니다. 이는 Firehose 스트림의 데이터 최신성에 영향을 미칩니다.

활성 파티션 수는 전송 버퍼 내에 있는 총 활성 파티션 개수입니다. 예를 들어 동적 파티셔닝 쿼리가 초당 3개의 파티션을 구성하고 60초마다 전송을 트리거하도록 버퍼 힌트가 구성된 경우, 활성 파티션은 평균 180개가 됩니다. Firehose가 파티션의 데이터를 대상으로 전송할 수 없는 경우 이 파티션은 전송이 가능해질 때까지 전송 버퍼에서 활성 상태로 간주됩니다.

레코드 데이터 필드 및 S3 접두사 표현식에 따라 S3 접두사를 새 값으로 평가하면 새 파티션이 생성됩니다. 각각의 활성 파티션에 대해 새 버퍼가 생성됩니다. 동일하게 평가된 S3 접두사를 가진 모든 후속 레코드는 해당 버퍼에 전송됩니다.

버퍼가 버퍼 크기 제한 또는 버퍼 시간 간격에 도달하면, Firehose는 버퍼 데이터가 포함된 객체를 생성하여 이를 지정된 Amazon S3 접두사에 전송합니다. 객체가 전송된 후 해당 파티션의 버퍼와 파티션 자체가 삭제되고 활성 파티션 카운트에서 제거됩니다.

Firehose는 각 파티션별로 버퍼 크기나 간격이 충족되면 각 버퍼 데이터를 단일 객체로 전송합니다. 활성 파티션의 수가 Firehose 스트림당 한도인 500개에 도달하면 Firehose 스트림의 나머지 레코드는 지정된 S3 오류 버킷 접두사(activePartitionExceeded)로 전송됩니다. Amazon Data Firehose Limit 양식을 사용하여 지정된 Firehose 스트림당 최대 5000개의 활성 파티션까지 이 할당량을 늘리도록 요청할 수 있습니다. 파티션이 더 필요하다면 더 많은 Firehose 스트림을 생성하여 그 전송 스트림에 걸쳐 활성 파티션을 분산시킬 수 있습니다.

Amazon Data Firehose에서 입력 데이터 형식 변환

Amazon Data Firehose는 Amazon S3에 데이터를 저장하기 전에 입력 데이터의 형식을 JSON에서 Apache Parquet 또는 Apache ORC로 변환할 수 있습니다. Parquet 및 ORC는 공간을 절약하고 JSON 같은 행 기준 형식과 비교할 때 쿼리 속도가 더 빠른 열 방식 데이터 형식입니다. 쉼표로 구분된 값(CSV) 또는 구조화된 텍스트와 같은 JSON 이외의 입력 형식을 변환하려는 경우 AWS Lambda 를 사용하여 먼저 JSON으로 변환할 수 있습니다. 자세한 내용은 소스 데이터 변환 단원을 참조하십시오.

Amazon Data Firehose로 레코드를 전송하기 전에 레코드를 집계하는 경우에도 데이터 형식을 변환할 수 있습니다.

Amazon Data Firehose가 레코드 데이터 형식을 변환하려면 다음 3가지 요소가 필요합니다.

Deserializer

Amazon Data Firehose는 입력 데이터의 JSON을 읽으려면 해제자(Deserializer)가 필요합니다. 다음 해제자 두 유형 중 하나를 선택할 수 있습니다.

여러 JSON 문서를 같은 레코드로 결합하는 경우 지원되는 JSON 형식에서 입력이 여전히 표시되는지 확인하세요. JSON 문서 배열은 유효한 입력이 아닙니다.

예를 들어 올바른 입력은 `{"a": 1}{"b": 1}`이고 잘못된 입력은 `[{"a":1}, {"a":2}]`입니다.

- Apache Hive JSON SerDe
- OpenX JSON SerDe

JSON 해제자 선택

입력 JSON에 다음과 같은 형식의 타임스탬프가 포함된 경우 OpenX JSON SerDe를 선택하세요.

- yyyy-MM-dd'T'HH:mm:ss[.S]'Z', 이 부분은 최대 9자리로 구성될 수 있음 - 예: `2017-02-07T15:13:01.39256Z`.
- yyyy-[M]M-[d]d HH:mm:ss[.S], 이 부분은 최대 9자리로 구성될 수 있음 - 예: `2017-02-07 15:13:01.14`.
- Epoch 초 – 예: `1518033528`.
- Epoch 밀리초 – 예: `1518033528123`.
- 부동 소수점 epoch 초 – 예: `1518033528.123`.

OpenX JSON SerDe는 마침표(.)를 밑줄(_)로 변환할 수 있습니다. 또한 JSON 키를 deserializing하기 전에 소문자로 변환할 수 있습니다. Amazon Data Firehose에서 이 해제자(Deserialzer)에 사용할 수 있는 옵션에 대한 자세한 내용은 OpenXJsonSerDe를 참조하세요.

어떤 해제자를 선택해야 할지 모르겠으면 OpenX JSON SerDe를 사용하세요. 단, 이 해제자가 지원하지 않는 타임스탬프가 없어야 합니다.

위에 나열한 형식 외의 타임스탬프가 있는 경우 Apache Hive JSON SerDe를 사용하세요. 이 deserializer를 선택하면, 사용할 타임스탬프 형식을 지정할 수 있습니다. Joda-Time `DateTimeFormat` 형식 문자열의 패턴 구문에 따라 타임스탬프 형식을 지정하세요. 자세한 내용은 DateTimeFormat 클래스를 참조하세요.

특수 값 `millis`를 사용하여 epoch 밀리초 단위의 타임스탬프를 구문분석할 수 있습니다. 형식을 지정하지 않으면 Amazon Data Firehose는 기본적으로 `java.sql.Timestamp::valueOf`를 사용합니다.

Hive JSON SerDe는 다음을 허용하지 않습니다.

- 열 이름의 마침표(.).
- `uniontype` 형식의 필드.
- 스키마에 숫자 형식이 있지만 문자열은 JSON인 필드. 예를 들어 스키마가 (an int)인 경우 JSON이 `{"a":"123"}`이면 Hive SerDe에서 오류가 발생합니다.

Hive SerDe는 중첩된 JSON을 문자열로 변환하지 않습니다. 예를 들어 `{"a":{"inner":1}}`가 있으면 `{"inner":1}`을 문자열로 처리하지 않습니다.

스키마

Amazon Data Firehose에는 해당 데이터를 해석하는 방법을 결정하는 스키마가 필요합니다. AWS Glue를 사용하여 AWS Glue Data Catalog에서 스키마를 생성합니다. 그러면 Amazon Data Firehose는 이 스키마를 참조한 다음 이를 사용하여 입력 데이터를 해석합니다. 동일한 스키마를 사용하여 Amazon Data Firehose와 분석 소프트웨어를 모두 구성할 수 있습니다. 자세한 내용은 AWS Glue 개발자 안내서의 AWS Glue 데이터 카탈로그 채우기를 참조하세요.

> **Note**
>
> AWS Glue 데이터 카탈로그에서 생성된 스키마는 입력 데이터 구조와 일치해야 합니다. 그렇지 않으면 변환된 데이터가 스키마에 지정되지 않은 속성을 포함할 수 없습니다. 중첩된 JSON

을 사용하는 경우 JSON 데이터의 구조를 반영하는 스키마의 STRUCT 유형을 사용하세요. STRUCT 유형의 중첩된 JSON을 처리하는 방법은 이 예시를 참조하세요.

Important

크기 제한이 지정되지 않은 데이터 유형의 경우 한 행의 모든 데이터에 대한 실제 제한은 32 MB입니다.
CHAR 또는 VARCHAR의 길이를 지정하면 Firehose는 입력 데이터를 읽을 때 지정된 길이로 문자열을 잘라냅니다. 기본 데이터 문자열이 더 긴 경우 변경되지 않습니다.

Serializer

Firehose에는 데이터를 대상 열 방식 저장 형식(Parquet 또는 ORC)으로 변환하는 시리얼 설정자 필요 - 다음 두 유형 중 하나를 선택할 수 있습니다.

- ORC SerDe
- Parquet SerDe

시리얼 설정자 선택

어떤 serializer를 선택하느냐는 비즈니스 요구 사항에 달라집니다. 두 가지 serializer 옵션에 대한 자세한 내용은 ORC SerDe와 Parquet SerDe를 참조하세요.

레코드 형식 변환 활성화

레코드 형식 변환을 활성화하면 Amazon Data Firehose 대상을 Amazon OpenSearch Service, Amazon Redshift 또는 Splunk로 설정할 수 없습니다. 형식 변환이 활성화되면 Firehose 스트림의 대상으로 Amazon S3만 사용할 수 있습니다. 다음 섹션에서는 콘솔 및 Firehose API 작업에서 레코드 형식 변환을 활성화하는 방법을 보여줍니다. 를 사용하여 레코드 형식 변환을 설정하는 방법의 예는 AWS::DataFirehose::DeliveryStream을 CloudFormation참조하세요.

콘솔에서 레코드 형식 변환 활성화

Firehose 전송 스트림을 생성하거나 업데이트할 때 콘솔에서 데이터 형식 변환을 활성화할 수 있습니다. 데이터 형식 변환이 활성화되면 Firehose 스트림의 대상으로 Amazon S3만 구성할 수 있습니다. 또한 형식 변환을 활성화하면 Amazon S3 압축은 비활성화됩니다. 그러나 Snappy 압축은 변환 프로세스

의 일부로 자동으로 이루어집니다. Amazon Data Firehose가 이 경우에 사용하는 Snappy의 프레이밍 형식은 Hadoop과 호환됩니다. 즉, Snappy 압축 결과를 사용하고 Athena에서 이 데이터에 대한 쿼리를 실행할 수 있습니다. Hadoop이 사용하는 Snappy 프레이밍 포맷은 BlockCompressorStream.java를 참조하세요.

데이터 Firehose 스트림의 데이터 형식 변환 활성화

1. 에 로그인 AWS Management Console하고 https://console.aws.amazon.com/firehose/ Amazon Data Firehose 콘솔을 엽니다.
2. 업데이트할 Firehose 스트림을 선택하거나 자습서: 콘솔에서 Firehose 스트림 생성의 절차에 따라 Firehose 스트림을 새로 만듭니다.
3. Convert record format(레코드 형식 변환) 아래에서 Record format conversion(레코드 형식 변환)을 Enabled(사용)로 설정합니다.
4. 원하는 출력 형식을 선택합니다. 두 옵션에 대한 자세한 내용은 Apache Parquet 및 Apache ORC를 참조하세요.
5. AWS Glue 테이블을 선택하여 소스 레코드에 대한 스키마를 지정합니다. 리전, 데이터베이스, 테이블 및 테이블 버전을 설정합니다.

Firehose API에서 레코드 형식 변환 관리

Amazon Data Firehose가 입력 데이터의 형식을 JSON에서 Parquet 또는 ORC로 변환하도록 하려면 ExtendedS3DestinationConfiguration 또는 ExtendedS3DestinationUpdate의 DataFormatConversionConfiguration을 지정하세요. DataFormatConversionConfiguration를 지정하는 경우 다음 제한이 적용됩니다.

- 레코드 형식 변환을 사용하는 경우, BufferingHints에서 `SizeInMBs`를 64보다 낮은 값으로 설정할 수 없습니다. 형식 변환을 활성화하지 않는 경우 기본값은 5입니다. 형식 변환을 활성화하면 값이 128이 됩니다.
- ExtendedS3DestinationConfiguration 또는 ExtendedS3DestinationUpdate의 `CompressionFormat`을 `UNCOMPRESSED`로 설정해야 합니다. `CompressionFormat`의 기본값은 `UNCOMPRESSED`입니다. 따라서 ExtendedS3DestinationConfiguration에서 지정하지 않은 상태로 두어도 됩니다. 지정하지 않아도 데이터는 기본적으로 Snappy 압축을 사용하여 serialization 프로세스 중에 압축됩니다. Amazon Data Firehose가 이 경우에 사용하는 Snappy의 프레이밍 형식은 Hadoop과 호환됩니다. 즉, Snappy 압축 결과를 사용하고 Athena에서 이 데이터에 대한 쿼리를 실행할 수 있습니다. Hadoop이 사용하는 Snappy 프레이밍 포맷은 BlockCompressorStream.java를 참조하세요. serializer를 구성할 때 다른 압축 유형을 선택할 수 있습니다.

데이터 형식 변환 오류 처리

Amazon Data Firehose는 레코드의 구문 분석 또는 deserialize가 불가능한 경우(예를 들어 데이터가 스키마와 일치하지 않는 경우), Amazon S3에 오류 접두사를 붙여서 씁니다. 이 쓰기가 실패하는 경우, Amazon Data Firehose는 영구적 재시도를 수행하여 이후 전송이 차단됩니다. 각각의 실패한 레코드에 대해 Amazon Data Firehose는 다음 스키마로 JSON 문서를 씁니다.

```
{
  "attemptsMade": long,
  "arrivalTimestamp": long,
  "ErrorCode": string,
  "ErrorMessage": string,
  "attemptEndingTimestamp": long,
  "rawData": string,
  "sequenceNumber": string,
  "subSequenceNumber": long,
  "dataCatalogTable": {
    "catalogId": string,
    "databaseName": string,
    "tableName": string,
    "region": string,
    "versionId": string,
    "catalogArn": string
  }
}
```

Amazon Data Firehose의 데이터 전송 이해

Firehose 스트림으로 데이터를 전송하면 선택한 대상에 자동으로 데이터가 전송됩니다. 다음 표는 다양한 대상으로의 데이터 전송을 설명합니다.

Destination	세부 정보
Amazon S3	Amazon S3로 데이터를 전송할 때 Firehose는 Firehose 스트림의 버퍼링 구성에 따라 여러 수신 레코드를 연결합니다. 그런 다음 Amazon S3 객체로 레코드를 Amazon S3에 전송합니다. 기본적으로 Firehose는 구분 기호 없이 데이터를 연결합니다. 레코드 사이에 새 줄 구분 기호를 사용하려면 Firehose 콘솔 구성 또는 API 파라미터에서 기능을 활성화하여 새 줄 구분 기호를 추가할 수 있습니다. Firehose와 Amazon S3 대상 사이의 데이터 전송은 TLS(HTTPS)로 암호화됩니다.
Amazon Redshift	Amazon Redshift로 데이터를 전송하는 경우 Firehose는 먼저 수신 데이터를 앞서 설명한 형식으로 S3 버킷에 전송합니다. 그러면 Firehose는 Amazon Redshift COPY 명령을 실행하여 S3 버킷의 데이터를 Amazon Redshift 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹으로 로드합니다. Amazon Data Firehose가 여러 수신 레코드를 Amazon S3 객체로 연결한 후 Amazon S3 객체를 Amazon Redshift 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹에 복사할 수 있는지 확인하세요. 자세한 내용은 Amazon Redshift COPY 명령 데이터 형식 파라미터를 참조하세요.
OpenSearch Service 및 OpenSearch Serverless	OpenSearch Service 및 OpenSearch Serverless로 데이터를 전송하기 위해 Amazon Data Firehose는 Firehose 스트림의 버퍼링 구성에 따라 수신 레코드를 버퍼링합니다. 그런 다음 OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션에 여러 레코드를 인덱싱하기 위한 OpenSearch Service 또는 OpenSearch Serverless 대량 요청을 생성합니다. 레코드를 Amazon Data Firehose로 보내기 전에 레코드가 UTF-8로 인코딩되어 단일 줄 JSON 객체에 결합되었는지 확인해야 합니다. 또한 레코드 단위로 설정되는 명시적 인덱스를 이용해 대량 요청을 하기 위해서는 OpenSearch Service 클러스터에 대한 `rest.action.multi.allow_explicit_index` 옵션이 true(기본값)로

Destination	세부 정보
	설정되어야 합니다. 자세한 정보는 Amazon OpenSearch Service 개발자 안내서의 OpenSearch Service Configure 고급 옵션을 참조하세요.
Splunk	Splunk로 데이터를 전송하기 위해 Amazon Data Firehose는 전송하는 바이트를 연결합니다. 줄 바꿈 문자와 같은 데이터의 구분 기호를 원하는 경우 이를 직접 삽입해야 합니다. Splunk가 모든 구분 기호를 구문 분석하도록 구성되어야 합니다. S3 오류 버킷(S3 백업)에 전송된 데이터를 다시 Splunk로 리드라이브하려면 Splunk 설명서에 설명된 단계를 따릅니다.
HTTP 엔드포인트	지원되는 타사 서비스 공급자가 소유한 HTTP 엔드포인트로 데이터를 전송하는 경우, 통합된 Amazon Lambda 서비스를 사용하여 수신 레코드를 서비스 공급자의 통합에서 예상하는 형식과 일치하는 형식으로 변환하는 함수를 생성할 수 있습니다. 허용되는 레코드 형식에 대한 자세한 내용은 대상으로 선택한 HTTP 엔드포인트의 타사 서비스 공급자에게 문의하세요.
Snowflake	Snowflake로 데이터를 전송하기 위해 Amazon Data Firehose는 내부적으로 1초 동안 데이터를 버퍼링하고 Snowflake 스트리밍 API 작업을 사용하여 Snowflake에 데이터를 삽입합니다. 기본적으로 삽입하는 레코드는 매초마다 플러시되고 Snowflake 테이블에 커밋됩니다. 삽입 호출을 수행하면 Firehose는 데이터가 Snowflake에 커밋되는 데 걸린 시간을 측정하는 CloudWatch 지표를 내보냅니다. 현재 Firehose는 단일 JSON 항목만 레코드 페이로드로 지원하며 JSON 배열은 지원하지 않습니다. 입력 페이로드가 유효한 JSON 객체인지 확인하고 불필요한 큰따옴표나 작은따옴표 또는 이스케이프 문자 없이 잘 구성되었는지 확인합니다.

각 Firehose 대상에는 고유한 데이터 전송 빈도가 있습니다. 자세한 내용은 버퍼링 힌트 구성 섹션을 참조하세요.

중복 레코드

Amazon Data Firehose는 데이터 전송에 최소 한 번(at-least-once) 시맨틱을 사용합니다. 데이터 전송 시간이 초과되는 등의 일부 상황에서, 원본 데이터 전송 요청이 진행될 경우 Amazon Data Firehose에

서 전송을 재시도하면 중복이 발생할 수 있습니다. 이는 Amazon S3 대상, Apache Iceberg 테이블 및 Snowflake 대상을 제외한 Amazon Data Firehose가 지원하는 모든 대상 유형에 적용됩니다.

주제

AWS 계정 및 리전 간 전송 이해

Amazon Data Firehose는 AWS 계정 간 HTTP 엔드포인트 대상으로의 데이터 전송을 지원합니다. 대상으로 선택한 Firehose 스트림과 HTTP 엔드포인트는 서로 다른 AWS 계정에 속할 수 있습니다.

Amazon Data Firehose는 AWS 리전 간 HTTP 엔드포인트 대상으로의 데이터 전송도 지원합니다. 한 AWS 리전의 Firehose 스트림에서 다른 리전의 HTTP 엔드포인트로 데이터를 전송할 수 AWS 있습니다. 또한 Firehose 스트림의 데이터를 AWS 리전 외부의 HTTP 엔드포인트 대상으로 전송할 수 있습니다. 예를 들어 HTTP 엔드포인트 URL을 원하는 대상으로 설정하여 온프레미스 서버로 전송할 수 있습니다. 이러한 시나리오의 경우 전송 비용에 추가 데이터 전송 요금이 더해질 수 있습니다. 자세한 내용은 “On-Demand Pricing”(온디맨드 요금) 페이지의 데이터 전송을 참조하세요.

HTTP 엔드포인트 전송 요청 및 응답 사양에 대한 이해

Amazon Data Firehose가 사용자 지정 HTTP 엔드포인트에 데이터를 성공적으로 전송하기 위해서는 이러한 엔드포인트가 특정 Amazon Data Firehose 요청 및 응답 형식을 사용하여 요청을 수락하고 응답을 보내야 합니다. 이 섹션에서는 Amazon Data Firehose 서비스가 사용자 지정 HTTP 엔드포인트에 보내는 HTTP 요청의 형식 사양과 Amazon Data Firehose 서비스가 예상하는 HTTP 응답의 형식 사양에 대해 설명합니다. Amazon Data Firehose에서 요청 제한 시간을 초과하기 전까지 HTTP 엔드포인트는 3분 이내에 요청에 응답해야 합니다. Amazon Data Firehose는 알맞은 형식을 준수하지 않는 응답을 전송 실패로 간주합니다.

요청 형식

경로 및 URL 파라미터

이는 단일 URL 필드의 일부로 사용자가 직접 구성합니다. Amazon Data Firehose는 수정하지 않고 구성된 대로 이를 전송합니다. https 대상만 지원합니다. 전송 스트림 구성 중에 URL 제한이 적용됩니다.

> **Note**
>
> HTTP 엔드포인트 데이터 전송에 대해서는 현재 포트 443만 지원됩니다.

HTTP 헤더 - X-Amz-Firehose-Protocol-Version

이 헤더를 사용하여 요청/응답 형식의 버전을 표시합니다. 현재 1.0이 유일한 버전입니다.

HTTP 헤더 - X-Amz-Firehose-Request-Id

이 헤더의 값은 디버깅 및 중복 제거 목적으로 사용할 수 있는 불분명한 GUID입니다. 엔드포인트 구현은 이 헤더의 값을, 가능하면 성공 및 실패한 요청 모두에 대해 기록해야 합니다. 요청 ID는 같은 요청을 여러 번 시도해도 동일하게 유지됩니다.

HTTP 헤더 - Content-Type

Content-Type 헤더의 값은 항상 `application/json`입니다.

HTTP 헤더 - Content-Encoding

요청을 전송할 때 GZIP을 사용하여 본문을 압축하도록 Firehose 스트림을 구성할 수 있습니다. 이 압축이 활성화되면, 표준 관행에 따라 Content-Encoding 헤더의 값은 gzip으로 설정됩니다. 압축이 활성화되지 않으면, Content-Encoding 헤더 자체가 없습니다.

HTTP 헤더 - Content-Length

이 헤더는 표준 방식으로 사용됩니다.

HTTP 헤더 - X-Amz-Firehose-Source-Arn:

ASCII 문자열 형식으로 표현된 Firehose 스트림의 ARN입니다. ARN은 리전, AWS 계정 ID 및 스트림 이름을 인코딩합니다. 예를 들어 `arn:aws:firehose:us-east-1:123456789:deliverystream/testStream`입니다.

HTTP 헤더 - X-Amz-Firehose-Access-Key

이 헤더에는 API 키 또는 다른 자격 증명이 포함됩니다. 전송 스트림을 만들거나 업데이트할 때 API 키(인증 토큰)를 만들거나 업데이트할 수 있습니다. Amazon Data Firehose는 액세스 키의 크기를 4,096바이트로 제한합니다. Amazon Data Firehose는 어떤 방식으로도 이 키를 해석하려는 시도를 하지 않습니다. 구성된 키는 그대로 이 헤더 값에 복사됩니다. 그러나 Secrets Manager를 사용하여 키를 구성하는 경우 보안 암호는 특정 JSON 객체 형식인 `{"api_key": "..."}`을(를) 따라야 합니다.

그 내용은 임의적이며 JWT 토큰 또는 ACCESS_KEY를 나타낼 가능성이 있습니다. 엔드포인트에 다중 필드 자격 증명(예: 사용자 이름 및 암호) 이 필요한 경우, 모든 필드의 값을 엔드포인트가 인식하는 형식(JSON 또는 CSV)으로 단일 액세스 키 내에 함께 저장해야 합니다. 원본 콘텐츠가 바이너리인 경우, 이 필드는 Base-64로 인코딩될 수 있습니다. Amazon Data Firehose는 구성된 값을 수정 및/또는 인코딩하지 않고 해당 콘텐츠를 그대로 사용합니다.

HTTP 헤더 - X-Amz-Firehose-Common-Attributes

이 헤더에는 전체 요청 및/또는 요청 내의 모든 레코드와 관련된 공통 속성(메타데이터)이 포함됩니다. 이는 Firehose 스트림을 만들 때 사용자가 직접 구성합니다. 이러한 속성 값은 다음 스키마를 사용하여 JSON 객체로 인코딩됩니다.

```
"$schema": http://json-schema.org/draft-07/schema#

properties:
  commonAttributes:
    type: object
    minProperties: 0
    maxProperties: 50
    patternProperties:
      "^.{1,256}$":
        type: string
        minLength: 0
        maxLength: 1024
```

다음은 그 예입니다.

```
"commonAttributes": {
    "deployment -context": "pre-prod-gamma",
    "device-types": ""
  }
```

본문 - 최대 크기

최대 본문 크기는 사용자가 구성하며 압축 전 최대 64MiB까지 가능합니다.

본문 - 스키마

본문에는 다음과 같은 JSON Schema(YAML로 작성)를 사용한 단일 JSON 문서가 포함됩니다.

```
"$schema": http://json-schema.org/draft-07/schema#

title: FirehoseCustomHttpsEndpointRequest
description: >
  The request body that the Firehose service sends to
  custom HTTPS endpoints.
type: object
properties:
  requestId:
    description: >
      Same as the value in the X-Amz-Firehose-Request-Id header,
      duplicated here for convenience.
    type: string
  timestamp:
    description: >
      The timestamp (milliseconds since epoch) at which the Firehose
      server generated this request.
    type: integer
  records:
    description: >
      The actual records of the Firehose stream, carrying
      the customer data.
    type: array
    minItems: 1
    maxItems: 10000
    items:
      type: object
      properties:
        data:
          description: >
            The data of this record, in Base64. Note that empty
            records are permitted in Firehose. The maximum allowed
            size of the data, before Base64 encoding, is 1024000
```

```
            bytes; the maximum length of this field is therefore
            1365336 chars.
          type: string
          minLength: 0
          maxLength: 1365336

required:
  - requestId
  - records
```

다음은 그 예입니다.

```
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090901599
  "records": [
    {
      "data": "aGVsbG8="
    },
    {
      "data": "aGVsbG8gd29ybGQ="
    }
  ]
}
```

응답 형식

오류 발생 시 기본 동작

응답이 아래 요구 사항을 준수하지 못하는 경우, Firehose 서버는 이를 본문 없이 500 상태 코드가 있는 것처럼 처리합니다.

상태 코드

HTTP 상태 코드는 반드시 2XX, 4XX 또는 5XX 범위에 있어야 합니다.

Amazon Data Firehose 서버는 리디렉션(3XX 상태 코드)을 따르지 않습니다. 레코드를 HTTP/EP에 성공적으로 전송한 것으로 간주되는 것은 응답 코드 200뿐입니다. 응답 코드 413(크기 초과)은 영구 실패로 간주되며, 이 코드가 구성되면 레코드 배치가 오류 버킷으로 전송되지 않습니다. 다른 모

든 응답 코드는 재시도 가능한 오류로 간주되며, 이후 설명할 백오프 재시도 알고리즘이 적용됩니다.

헤더 – 콘텐츠 유형

허용되는 유일한 콘텐츠 유형은 애플리케이션/json입니다.

HTTP 헤더 - Content-Encoding

콘텐츠 인코딩은 사용하지 않아야 합니다. 본문은 반드시 압축을 풀어야 합니다.

HTTP 헤더 - Content-Length

응답에 본문이 있는 경우 반드시 Content-Length 헤더가 있어야 합니다.

본문 - 최대 크기

응답 본문의 크기는 1MiB 이하여야 합니다.

```
"$schema": http://json-schema.org/draft-07/schema#

title: FirehoseCustomHttpsEndpointResponse

description: >
  The response body that the Firehose service sends to
  custom HTTPS endpoints.
type: object
properties:
  requestId:
    description: >
      Must match the requestId in the request.
    type: string

  timestamp:
    description: >
      The timestamp (milliseconds since epoch) at which the
      server processed this request.
    type: integer

  errorMessage:
    description: >
      For failed requests, a message explaining the failure.
      If a request fails after exhausting all retries, the last
      Instance of the error message is copied to error output
      S3 bucket if configured.
```

```
    type: string
    minLength: 0
    maxLength: 8192
required:
  - requestId
  - timestamp
```

다음은 그 예입니다.

```
Failure Case (HTTP Response Code 4xx or 5xx)
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": "1578090903599",
  "errorMessage": "Unable to deliver records due to unknown error."
}
Success case (HTTP Response Code 200)
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090903599
}
```

오류 응답 처리

모든 오류 사례에서 Amazon Data Firehose 서버는 지수 백오프 알고리즘을 사용하여 동일한 레코드 배치의 전송을 다시 시도합니다. 지터 계수 (15%)인 초기 백오프 시간(1초)을 사용하여 재시도가 백오프되고, 이후 재시도할 때마다 지터가 추가된 공식(초기 백오프 시간* (승수(2) ^ retry_count))을 사용하여 백오프됩니다. 백오프 시간은 최대 2분 간격으로 제한됩니다. 예를 들어, 'n번째 재시도' 시 백오프 시간은 최대(120, 2^n) * 무작위(0.85, 1,15)입니다.

이전 방정식에서 지정된 파라미터는 변경될 수 있습니다. 지수 백오프 알고리즘에 사용되는 정확한 초기 백오프 시간, 최대 백오프 시간, 승수 및 지터 백분율은 AWS Firehose 설명서를 참조하세요.

이후에 다시 시도할 때마다 레코드가 전송되는 액세스 키 및/또는 대상은 업데이트된 Firehose 스트림 구성에 따라 변경될 수 있습니다. Amazon Data Firehose 서비스는 최선의 방법으로 여러 번의 재시도에 걸쳐 동일한 요청 ID(request-id)를 사용합니다. 이 최신 기능은 HTTP 엔드포인트 서버에서 중복 제거 목적으로 사용할 수 있습니다. Firehose 스트림 구성에 따라 허용된 최대 시간이 지난 후에도 요청이 전달되지 않는 경우, 스트림 구성에 따라 레코드 배치를 오류 버킷에 선택적으로 전송할 수 있습니다.

예제

CWLog 소싱 요청의 예:

```
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090901599,
  "records": [
   {
    "data": {
      "messageType": "DATA_MESSAGE",
      "owner": "123456789012",
      "logGroup": "log_group_name",
      "logStream": "log_stream_name",
      "subscriptionFilters": [
        "subscription_filter_name"
      ],
      "logEvents": [
        {
          "id": "01234567890123456789012345678901234567890123456789012345",
          "timestamp": 1510109208016,
          "message": "log message 1"
        },
        {
          "id": "01234567890123456789012345678901234567890123456789012345",
          "timestamp": 1510109208017,
          "message": "log message 2"
        }
      ]
    }
   }
  ]
}
```

데이터 전송 실패 처리

각 Amazon Data Firehose 대상마다 고유한 데이터 전송 실패 처리 방식이 있습니다.

OpenSearch, Splunk, HTTP 엔드포인트 등 다양한 대상에 대해 Firehose 스트림을 설정할 때, 전송에 실패한 데이터를 백업할 수 있는 S3 버킷도 설정합니다. 전송 실패 시 Firehose가 데이터를 백업하는

방법에 대한 자세한 내용은 이 페이지의 관련 대상 섹션을 참조하세요. 전송되지 못한 데이터를 백업할 수 있는 S3 버킷에 대한 액세스 권한 부여 방법에 대해서는 Firehose에 Amazon S3 대상 액세스 권한 부여를 참조하세요. Firehose가 (a) 스트림 대상으로 데이터를 전송하지 못하고 (b) 전송 실패에 따라 백업 S3 버킷에 데이터 쓰기에 실패한 경우, 데이터가 대상으로 전달되거나 백업 S3 위치에 기록될 때까지 스트림 전송을 사실상 일시 중지합니다.

Amazon S3

S3 버킷에 대한 데이터 전송은 여러 가지 이유로 실패할 수 있습니다. 예를 들어 버킷이 더 이상 존재하지 않거나, Amazon Data Firehose가 맡은 IAM 역할에 버킷 액세스 권한이 없거나, 네트워크 장애 또는 이와 유사한 이벤트가 발생하는 경우입니다. 이러한 조건에서 Amazon Data Firehose는 전송에 성공할 때까지 최대 24시간 동안 계속 재시도를 합니다. Amazon Data Firehose의 최대 데이터 스토리지 시간은 24시간입니다. 24시간 넘게 데이터 전송에 실패할 경우 데이터를 잃게 됩니다.

S3 버킷에 대한 데이터 전송은 여러 가지 이유로 실패할 수 있습니다.

- 버킷이 더 이상 존재하지 않습니다.
- Amazon Data Firehose에서 수임하는 IAM 역할에는 버킷에 대한 액세스 권한이 없습니다.
- 네트워크 문제가 발생했습니다.
- HTTP 500 또는 기타 API 실패와 같은 S3 오류가 발생했습니다.

이러한 경우 Amazon Data Firehose는 전송을 재시도합니다.

- DirectPut 소스: 재시도는 최대 24시간 동안 계속됩니다.
- Kinesis Data Streams 또는 Amazon MSK 소스: 재시도는 스트림에 정의된 보존 정책까지 무기한 계속됩니다.

Amazon Data Firehose는 Lambda 처리 또는 파켓 변환이 실패할 때만 실패한 레코드를 S3 오류 버킷으로 전송합니다. 다른 장애 시나리오에서는 보존 기간에 도달할 때까지 S3를 계속 재시도합니다. Firehose가 S3에 레코드를 성공적으로 전송하면 S3 객체 파일이 생성되고 부분 레코드 실패 시 자동으로 전송을 재시도하고 성공적으로 처리된 레코드로 동일한 S3 객체 파일을 업데이트합니다.

Amazon Redshift

Amazon Redshift 대상에 대해 Firehose 스트림 생성 시 재시도 기간(0~7200초)을 지정할 수 있습니다.

여러 가지 이유로 인해 Amazon Redshift 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹으로의 데이터 전송이 실패할 수 있습니다. 예를 들어 Firehose 스트림의 잘못된 클러스터 구

성, 클러스터 또는 작업 그룹의 유지 관리 진행, 네트워크 장애 등이 이유가 될 수 있습니다. 이러한 조건에서 Amazon Data Firehose는 지정된 시간 동안 재시도를 수행하고 이 Amazon S3 객체의 특정 배치를 건너뜁니다. 건너뛴 객체의 정보는 `errors/` 폴더의 매니페스트 파일로 S3 버킷에 전송되며, 이를 수동 채우기에 사용할 수 있습니다. 매니페스트 파일로 데이터를 수동으로 COPY하는 방법에 대한 내용은 매니페스트를 사용한 데이터 파일 지정을 참조하세요.

Amazon OpenSearch Service 및 OpenSearch Serverless

OpenSearch Service 및 OpenSearch Serverless 대상의 경우 Firehose 스트림 생성 시 재시도 기간(0~7200초)을 지정할 수 있습니다.

여러 가지 이유로 OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션으로의 데이터 전송이 실패할 수 있습니다. 예를 들어 Firehose 스트림의 잘못된 OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션 구성, OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션의 유지 관리 진행, 네트워크 장애 또는 이와 유사한 이벤트 발생 등이 이유가 될 수 있습니다. 이러한 조건에서 Amazon Data Firehose는 지정된 시간 동안 재시도를 수행하고 특정 인덱스 요청을 건너뜁니다. 건너뛴 문서는 `AmazonOpenSearchService_failed/` 폴더를 통해 S3 버킷에 전송되며, 이를 수동 채우기에 사용할 수 있습니다.

OpenSearch Service의 경우 각 문서의 JSON 형식은 다음과 같습니다.

```
{
    "attemptsMade": "(number of index requests attempted)",
    "arrivalTimestamp": "(the time when the document was received by Firehose)",
    "errorCode": "(http error code returned by OpenSearch Service)",
    "errorMessage": "(error message returned by OpenSearch Service)",
    "attemptEndingTimestamp": "(the time when Firehose stopped attempting index
 request)",
    "esDocumentId": "(intended OpenSearch Service document ID)",
    "esIndexName": "(intended OpenSearch Service index name)",
    "esTypeName": "(intended OpenSearch Service type name)",
    "rawData": "(base64-encoded document data)"
}
```

OpenSearch Serverless의 경우 각 문서의 JSON 형식은 다음과 같습니다.

```
{
    "attemptsMade": "(number of index requests attempted)",
    "arrivalTimestamp": "(the time when the document was received by Firehose)",
```

```
    "errorCode": "(http error code returned by OpenSearch Serverless)",
    "errorMessage": "(error message returned by OpenSearch Serverless)",
    "attemptEndingTimestamp": "(the time when Firehose stopped attempting index
 request)",
    "osDocumentId": "(intended OpenSearch Serverless document ID)",
    "osIndexName": "(intended OpenSearch Serverless index name)",
    "rawData": "(base64-encoded document data)"
}
```

Splunk

Amazon Data Firehose는 Splunk로 데이터를 전송할 때 Splunk로부터 확인을 기다립니다. 오류가 발생하거나 확인 제한 시간 내에 확인 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 Splunk로 데이터를 전송할 때마다(최초 시도이든 재시도이든 상관없이) 확인 제한 시간 카운터를 다시 시작합니다. 그런 다음 Splunk로부터 수신 확인을 기다립니다. 재시도 기간이 만료되어도, Amazon Data Firehose는 확인을 수신하거나 확인 제한 시간에 도달할 때까지 확인을 기다립니다. 확인 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 확인을 수신하거나 재시도 시간이 만료되었는지 확인할 때까지 논리를 반복합니다.

확인 수신에 실패하는 것이 발생할 수 있는 유일한 데이터 전송 오류는 아닙니다. 데이터 전송 오류의 다른 유형에 대한 내용은 Splunk 데이터 전송 오류를 참조하세요. 모든 데이터 전송 오류는 재시도 지속시간이 0보다 클 경우 재시도 로직을 트리거합니다.

다음은 오류 레코드의 예입니다.

```
{
  "attemptsMade": 0,
  "arrivalTimestamp": 1506035354675,
  "errorCode": "Splunk.AckTimeout",
  "errorMessage": "Did not receive an acknowledgement from HEC before the HEC
 acknowledgement timeout expired. Despite the acknowledgement timeout, it's possible
 the data was indexed successfully in Splunk. Amazon Data Firehose backs up in Amazon
 S3 data for which the acknowledgement timeout expired.",
  "attemptEndingTimestamp": 13626284715507,
  "rawData":
 "MiAyNTE2MjAyNzIyMDkgZW5pLTA1ZjMyMmQ1IDIxOC45Mi4xODguMjE0IDE3Mi4xNi4xLjE2NyAyNTIzMyAxNDMzIDYgM
```

```
  "EventId": "4957719392811414733960077847108249239316413987720003584 2.0"
}
```

HTTP 엔드포인트 대상

Amazon Data Firehose는 HTTP 엔드포인트 대상으로 데이터를 전송하고 이 대상의 응답을 기다립니다. 오류가 발생하거나 응답 제한 시간 내에 응답 메시지가 도착하지 않을 경우, Amazon Data Firehose는 재시도 기간 카운터를 시작합니다. 재시도 지속시간이 만료될 때까지 재시도합니다. 그 다음 Amazon Data Firehose는 이를 데이터 전송 실패로 간주하고 데이터를 Amazon S3 버킷에 백업합니다.

Amazon Data Firehose는 HTTP 엔드포인트 대상으로 데이터를 전송할 때마다(최초 시도이든 재시도이든 상관없이) 확인 제한 시간 카운터를 다시 시작합니다. 그런 다음 HTTP 엔드포인트 대상에서 응답이 도착하기를 기다립니다. 재시도 기간이 만료되어도, Amazon Data Firehose는 응답을 수신하거나 응답 제한 시간에 도달할 때까지 응답을 기다립니다. 응답 시간이 초과한 경우 Amazon Data Firehose는 재시도 카운터에 시간이 남아 있는지 확인합니다. 시간이 남은 경우 다시 시도하여 응답을 수신하거나 재시도 시간 만료가 확인될 때까지 논리를 반복합니다.

응답 수신에 실패하는 것이 발생할 수 있는 유일한 데이터 전송 오류는 아닙니다. 데이터 전송 오류의 다른 유형에 대한 내용은 HTTP 엔드포인트 데이터 전송 오류를 참조하세요.

다음은 오류 레코드의 예입니다.

```
{
 "attemptsMade":5,
 "arrivalTimestamp":1594265943615,
 "errorCode":"HttpEndpoint.DestinationException",
 "errorMessage":"Received the following response from the endpoint destination.
 {"requestId": "109777ac-8f9b-4082-8e8d-b4f12b5fc17b", "timestamp": 1594266081268,
 "errorMessage": "Unauthorized"}",
 "attemptEndingTimestamp":1594266081318,
 "rawData":"c2FtcGxlIHJhdyBkYXRh",
 "subsequenceNumber":0,
 "dataId":"49607357361271740811418664280693044274821622880012337186.0"
}
```

Snowflake

Snowflake 대상의 경우 Firehose 스트림을 생성할 때 선택 사항으로 재시도 기간(0~7,200초)을 지정할 수 있습니다. 재시도 기간의 기본값은 60초입니다.

Snowflake 테이블로 데이터 전송 시 잘못된 Snowflake 대상 구성, Snowflake 중단, 네트워크 장애 등의 몇 가지 이유로 실패할 수 있습니다. 재시도할 수 없는 오류에는 재시도 정책이 적용되지 않습니다. 예를 들어 테이블에 누락된 추가 열이 있어서 Snowflake가 JSON 페이로드를 거부하는 경우, Firehose는 전송을 재시도하지 않습니다. 대신 JSON 페이로드 문제로 인한 모든 삽입 실패 관련 백업을 S3 오류 버킷에 생성합니다.

마찬가지로 잘못된 역할, 테이블 또는 데이터베이스로 인해 전송이 실패하는 경우 Firehose는 재시도하지 않고 S3 버킷에 데이터를 기록합니다. 재시도 기간은 Snowflake 서비스 문제, 일시적인 네트워크 결함 등으로 인한 오류에만 적용됩니다. 이러한 조건에서 Firehose는 지정된 기간 동안 재시도한 후 S3에 데이터를 전송합니다. 실패한 레코드는 snowflake-failed/ 폴더에 전송되어 수동 채우기가 가능합니다.

다음은 S3에 전송하는 각 레코드에 대한 JSON 예제입니다.

```
{
    "attemptsMade": 3,
    "arrivalTimestamp": 1594265943615,
    "errorCode": "Snowflake.InvalidColumns",
    "errorMessage": "Snowpipe Streaming does not support columns of type AUTOINCREMENT,
 IDENTITY, GEO, or columns with a default value or collation",
    "attemptEndingTimestamp": 1712937865543,
    "rawData": "c2FtcGxlIHJhdyBkYXRh"
}
```

Amazon S3 객체 이름 형식 구성

Firehose가 Amazon S3에 데이터를 전송하면 S3 객체 키 이름은 <evaluated prefix><suffix> 형식을 따르며, 접미사는 <Firehose stream name>-<Firehose stream version>-<year>-<month>-<day>-<hour>-<minute>-<second>-<uuid><file extension> <Firehose stream version> 형식을 따릅니다. 이 형식은 1로 시작되며 Firehose 스트림의 구성이 변경될 때마다 1씩 증가합니다. Firehose 스트림 구성(예: S3 버킷 이름, 버퍼링 힌트, 압축, 암호화)을 변경할 수 있습니다. 이 작업은 Firehose 콘솔 또는 UpdateDestination API 작업을 사용하여 수행할 수 있습니다.

<평가된 접두사>의 경우 Firehose는 YYYY/MM/dd/HH 형식에 기본 시간 접두사를 추가합니다. 이 접두사는 버킷에 논리적 계층 구조를 생성하는데, 계층 구조 내에서 슬래시(/) 하나당 한 계층을 생성합니다. 런타임 시 평가되는 표현식을 포함하는 사용자 지정 접두사를 지정하면 이 구조를 수정할 수 있습니다. 사용자 지정 접두사를 지정하는 방법에 대한 자세한 정보는 Custom Prefixes for Amazon Simple Storage Service Objects(Amazon Simple Storage Service 객체의 사용자 지정 접두사)를 참조하세요.

기본적으로 시간 접두사 및 접미사에 사용되는 시간대는 UTC이지만 원하는 시간대로 변경할 수 있습니다. 예를 들어 UTC 대신 일본 표준시를 사용하려면 AWS Management Console 또는 API 파라미터 설정(CustomTimeZone)에서 시간대를 아시아/도쿄로 구성할 수 있습니다. 다음 목록에는 S3 접두사 구성 시 Firehose가 지원하는 시간대가 포함되어 있습니다.

지원되는 시간대

다음은 Firehose가 S3 접두사 구성 시에 지원하는 시간대 목록입니다.

Africa

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmera
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
```

```
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
```

America

```
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Aruba
America/Asuncion
America/Barbados
America/Belize
America/Bogota
America/Buenos_Aires
America/Caracas
America/Cayenne
America/Cayman
America/Chicago
America/Costa_Rica
America/Cuiaba
America/Curacao
America/Dawson_Creek
America/Denver
America/Dominica
America/Edmonton
America/El_Salvador
America/Fortaleza
America/Godthab
```

```
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Indianapolis
America/Jamaica
America/La_Paz
America/Lima
America/Los_Angeles
America/Managua
America/Manaus
America/Martinique
America/Mazatlan
America/Mexico_City
America/Miquelon
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Noronha
America/Panama
America/Paramaribo
America/Phoenix
America/Port_of_Spain
America/Port-au-Prince
America/Porto_Acre
America/Puerto_Rico
America/Regina
America/Rio_Branco
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Tegucigalpa
```

```
America/Thule
America/Tijuana
America/Tortola
America/Vancouver
America/Winnipeg
```

Antarctica

```
Antarctica/Casey
Antarctica/DumontDUrville
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
```

Asia

```
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dubai
Asia/Dushanbe
Asia/Hong_Kong
Asia/Irkutsk
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
```

```
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Katmandu
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuwait
Asia/Macao
Asia/Magadan
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novosibirsk
Asia/Phnom_Penh
Asia/Pyongyang
Asia/Qatar
Asia/Rangoon
Asia/Riyadh
Asia/Saigon
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan
```

Atlantic

```
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
```

```
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Jan_Mayen
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
```

Australia

```
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Darwin
Australia/Hobart
Australia/Lord_Howe
Australia/Perth
Australia/Sydney
```

Europe

```
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belgrade
Europe/Berlin
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Helsinki
Europe/Istanbul
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Minsk
```

```
Europe/Monaco
Europe/Moscow
Europe/Oslo
Europe/Paris
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/Simferopol
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Vaduz
Europe/Vienna
Europe/Vilnius
Europe/Warsaw
Europe/Zurich
```

Indian

```
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
```

Pacific

```
Pacific/Apia
Pacific/Auckland
Pacific/Chatham
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
Pacific/Fiji
```

```
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Majuro
Pacific/Marquesas
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
```

<file extension>을 제외한 접미사 필드는 변경할 수 없습니다. 데이터 형식 변환 또는 압축을 사용하면 Firehose는 구성을 기반으로 파일 확장자를 추가합니다. 다음 표는 Firehose가 추가하는 기본 파일 확장명을 설명합니다.

구성	파일 확장명
데이터 형식 변환: Parquet	.parquet
데이터 형식 변환: ORC	.orc
압축: Gzip	.gz

구성	파일 확장명
압축: Zip	.zip
압축: Snappy	.snappy
압축: Hadoop-Snappy	.hsnappy

Firehose 콘솔이나 API에서 원하는 파일 확장자를 지정할 수도 있습니다. 파일 확장자는 마침표(.)로 시작해야 하며 문자 0~9a~z!-_.*'()를 포함할 수 있습니다. 파일 확장자는 128자를 초과할 수 없습니다.

> **Note**
>
> 파일 확장자를 지정하면 데이터 형식 변환 또는 압축을 사용할 때 Firehose가 추가하는 기본 파일 확장자를 재정의합니다.

Amazon S3 객체의 사용자 지정 접두사 이해

Amazon S3에 전달되는 객체는 <평가된 접두사><접미사>의 이름 형식을 따릅니다. 런타임에 평가되는 표현식을 포함하는 사용자 지정 접두사를 지정할 수 있습니다. 지정한 사용자 지정 접두사는 기본 접두사인 `yyyy/MM/dd/HH`를 재정의합니다.

사용자 지정 접두사에 `!{namespace:`*`value`*`}` 형식의 표현식을 사용할 수 있으며, 여기서 `namespace`는 다음 섹션에서 설명하듯이 다음 중 하나가 될 수 있습니다.

- `firehose`
- `timestamp`
- `partitionKeyFromQuery`
- `partitionKeyFromLambda`

슬래시로 끝나는 접두사는 Amazon S3 버킷에서 자리 표시자로 나타납니다. 자세한 내용은 Amazon Data FirehoseDeveloper 안내서의 Amazon S3 객체 이름 형식을 참조하세요.

`timestamp` 네임스페이스

이 네임스페이스에 유효한 값은 유효한 Java DateTimeFormatter 문자열입니다. 예를 들어 2018년에 `!{timestamp:yyyy}` 표현식은 `2018`로 평가됩니다.

타임스탬프 평가 시, Firehose는 기록되는 Amazon S3 객체에 포함된 가장 오래된 레코드의 근사 도착 타임스탬프를 사용합니다.

기본적으로 타임스탬프는 UTC입니다. 하지만 원하는 시간대를 지정할 수 있습니다. 예를 들어 UTC 대신 일본 표준시를 사용하려는 경우 AWS Management Console 또는 API 파라미터 설정 (CustomTimeZone)에서 시간대를 아시아/도쿄로 구성할 수 있습니다. 지원되는 시간대 목록을 보려면 Amazon S3 객체 이름 형식을 참조하세요.

동일한 접두사 표현식에 `timestamp` 네임스페이스를 한 번 넘게 사용할 경우 각 인스턴스는 동일한 시간으로 평가됩니다.

`firehose` 네임스페이스

이 네임스페이스에는 `error-output-type` 값과 `random-string` 값을 사용할 수 있습니다. 다음 표에는 사용 방법이 나와 있습니다.

`firehose` 네임스페이스 값

변환	설명	입력 예	출력 예시	참고
`error-output-type`	Firehose 스트림 구성 및 실패 이유에 따라 {processing-failed, AmazonOpenSearchService-failed, splunk-failed, format-conversion-failed, http-endpoint-failed} 문자열 중 하나로 평가됩니다. 동일한 접두사 표현식에 한 번 넘게 사용할 경우 각 인스턴스는 동	`myPrefix/result=!{firehose:error-output-type}/!{timestamp:yyyy/MM/dd}`	`myPrefix/result=processing-failed/2018/08/03`	error-output-type 값은 ErrorOutputPrefix 필드에만 사용할 수 있습니다.

변환	설명	입력 예	출력 예시	참고
	일한 오류 문자열로 평가됩니다.			
`random-string`	11자의 무작위 문자열로 평가됩니다. 동일한 접두사 표현식에 한 번 넘게 사용할 경우 각 인스턴스는 새로운 무작위 문자열로 평가됩니다.	`myPrefix/!{firehose:random-string}/`	`myPrefix/046b6c7f-0b/`	두 접두사 유형 모두에 사용할 수 있습니다. 형식 문자열 선두에 이를 배치하여 무작위의 접두사를 얻을 수 있으며, 경우에 따라 이 접두사는 Amazon S3를 통해 최고의 처리량을 얻기 위해 필요한 경우가 있습니다.

`partitionKeyFromLambda` 및 `partitionKeyFromQuery` 네임스페이스

동적 파티셔닝의 경우, S3 버킷 접두사에 다음 표현식 형식을 사용해야 합니다: `!{namespace:value}`, 여기서 네임스페이스는 `partitionKeyFromQuery` 또는 `partitionKeyFromLambda`이거나, 둘 다일 수 있습니다. 인라인 구문 분석을 사용하여 소스 데이터에 대한 파티션 키를 생성하는 경우 다음 형식으로 지정된 표현식으로 구성되는 S3 버킷 접두사 값을 지정해야 합니다: `"partitionKeyFromQuery:keyID"`. AWS Lambda 함수를 사용하여 소스 데이터에 대한 파티션 키를 생성하는 경우 다음 형식으로 지정된 표현식으로 구성되는 S3 버킷 접두사 값을 지정해야 합니다: `"partitionKeyFromLambda:keyID"`. 자세한 내용은 Amazon Firehose 스트림 생성의 “Amazon S3를 대상으로 선택”을 참조하세요.

의미 체계 규칙

`Prefix` 및 `ErrorOutputPrefix` 표현식에 적용되는 규칙은 다음과 같습니다.

- `timestamp` 네임스페이스의 경우 작은따옴표로 묶이지 않은 문자가 평가됩니다. 즉, 값 필드에서 작은따옴표로 이스케이프 처리된 문자열은 문자로 처리됩니다.

- 타임스탬프 네임스페이스 표현식을 포함하지 않는 접두사를 지정할 경우, Firehose는 `Prefix` 필드의 값에 `!{timestamp:yyyy/MM/dd/HH/}` 표현식을 추가합니다.
- `!{` 시퀀스는 `!{namespace:value}` 표현식에만 나타날 수 있습니다.
- `Prefix`에 표현식이 없을 경우에만 `ErrorOutputPrefix`가 null이 될 수 있습니다. 이 경우 `Prefix`는 `<specified-prefix>yyyy/MM/DDD/HH/`로 평가되고 `ErrorOutputPrefix`는 `<specified-prefix><error-output-type>yyyy/MM/DDD/HH/`로 평가됩니다. `DDD`는 해당 연도의 날짜를 나타냅니다.
- `ErrorOutputPrefix`에 표현식을 지정할 경우, 최소 한 개의 `!{firehose:error-output-type}` 인스턴스를 포함시켜야 합니다.
- `Prefix`에는 `!{firehose:error-output-type}`을 포함할 수 없습니다.
- `Prefix` 또는 `ErrorOutputPrefix`는 평가 후 512자를 넘을 수 없습니다.
- 대상이 Amazon Redshift인 경우, `Prefix`에 표현식이 포함되어서는 안 되며, `ErrorOutputPrefix`는 null이어야 합니다.
- 대상이 Amazon OpenSearch Service 또는 Splunk이고 지정된 `ErrorOutputPrefix`가 없는 경우 Firehose는 실패한 레코드에 `Prefix` 필드를 사용합니다.
- 대상이 Amazon S3인 경우 Amazon S3 대상 구성의 `Prefix` 및 `ErrorOutputPrefix`를 각각 성공 레코드 및 실패 레코드에 사용합니다. AWS CLI 또는 API를 사용하는 경우 `ExtendedS3DestinationConfiguration`을 사용하여 자체 `Prefix`와 `ErrorOutputPrefix`로 Amazon S3 백업 구성을 지정할 수 있습니다.
- 를 사용하고 대상을 Amazon S3로 AWS Management Console 설정하면 Firehose는 각각 성공적인 레코드`Prefix`와 실패한 레코드에 대해 대상 구성`ErrorOutputPrefix`에서 및를 사용합니다. 표현식을 사용하여 접두사를 지정하는 경우 `!{firehose:error-output-type}`를 포함한 오류 접두사를 지정해야 합니다.
- 를 AWS CLI, API와 `ExtendedS3DestinationConfiguration` 함께 사용하거나를 지정하는 CloudFormation경우 `S3BackupConfiguration`Firehose는 기본를 제공하지 않습니다`ErrorOutputPrefix`.
- ErrorOutputPrefix 표현식을 생성할 때 `partitionKeyFromLambda` 및 `partitionKeyFromQuery` 네임스페이스를 사용할 수 없습니다.

접두사의 예

Prefix 및 **ErrorOutputPrefix**의 예

Input	평가된 접두사(2018년 8월 27일 오전 10:30 UTC)
Prefix: 지정 안 함 ErrorOutputPrefix : myFirehoseFailures/!{firehose:error-output-type}/	Prefix: 2018/08/27/10 ErrorOutputPrefix : myFirehoseFailures/processing-failed/
Prefix: !{timestamp:yyyy/MM/dd} ErrorOutputPrefix : 지정 안 함	잘못된 입력: 접두사에 표현식이 포함된 경우 ErrorOutputPrefix 는 null이 될 수 없음
Prefix: myFirehose/DeliveredYear=!{timestamp:yyyy}/anyMonth/rand=!{firehose:random-string} ErrorOutputPrefix : myFirehoseFailures/!{firehose:error-output-type}/!{timestamp:yyyy}/anyMonth/!{timestamp:dd}	Prefix: myFirehose/DeliveredYear=2018/anyMonth/rand=5abf82daaa5 ErrorOutputPrefix : myFirehoseFailures/processing-failed/2018/anyMonth/10
Prefix: myPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/ ErrorOutputPrefix : myErrorPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/!{firehose:error-output-type}	Prefix: myPrefix/year=2018/month=07/day=06/hour=23/ ErrorOutputPrefix : myErrorPrefix/year=2018/month=07/day=06/hour=23/processing-failed
Prefix: myFirehosePrefix/ ErrorOutputPrefix : 지정 안 함	Prefix: myFirehosePrefix/2018/08/27/

Input	평가된 접두사(2018년 8월 27일 오전 10:30 UTC)
	`ErrorOutputPrefix : myFirehosePrefix/processing-failed/2018/08/27/`

OpenSearch Service 인덱스 교체 구성

OpenSearch Service 대상의 경우 NoRotation, OneHour, OneDay, OneWeek, OneMonth 등 다섯 가지 옵션 중 하나로 시간 기반 인덱스 로테이션 옵션을 지정할 수 있습니다.

선택한 교체 옵션에 따라 Amazon Data Firehose가 UTC 도착 타임스탬프의 일부를 지정된 인덱스 이름에 추가합니다. 그리고 추가된 타임스탬프를 회전시킵니다. 다음 예는 각 인덱스 로테이션 옵션에 대한 OpenSearch Service의 결과 인덱스 이름을 보여주며, 여기서 지정된 인덱스 이름은 myindex이고 도착 타임스탬프는 `2016-02-25T13:00:00Z`입니다.

RotationPeriod	IndexName
`NoRotation`	`myindex`
`OneHour`	`myindex-2016-02-25-13`
`OneDay`	`myindex-2016-02-25`
`OneWeek`	`myindex-2016-w08`
`OneMonth`	`myindex-2016-02`

> **Note**
>
> `OneWeek` 옵션에서, Data Firehose는 <연도>-w<주 번호>(예:`2020-w33`) 형식을 사용하여 인덱스를 자동 생성하며, 여기서 주 번호는 UTC 시간 및 다음 미국 규칙에 따라 계산됩니다.
>
> - 한 주는 일요일에 시작함
> - 한 해의 첫 주는 그 해에 토요일이 포함된 첫 번째 주가 됨

데이터 전송 일시 중지 및 재개

Firehose 스트림을 설정한 후 스트림 소스에서 사용 가능한 데이터는 지속적으로 대상에 전송됩니다. 스트림 대상을 일시적으로 사용할 수 없는 상황이 발생하는 경우(예: 계획된 유지 관리 작업 진행), 데이터 전송을 일시적으로 중지하고 대상이 다시 사용 가능해지면 재개하는 것이 좋습니다.

> **Important**
>
> 아래 설명된 접근 방식을 사용하여 스트림을 일시 중지 및 재개하면 스트림을 재개한 후 Amazon S3의 오류 버킷으로 전송되는 레코드가 거의 없는 반면 나머지 스트림은 계속 대상으로 전송되는 것을 확인할 수 있습니다. 이는 해당 접근 방식의 알려진 제한 사항으로, 여러 번의 재시도 후에도 대상으로 전송되지 못한 소수의 레코드가 실패한 것으로 처리되기 때문입니다.

Firehose 스트림 일시 중지

Firehose의 스트림 전송을 일시 중지하려면 먼저 전송 실패에 대해 Firehose가 S3 백업 위치에 쓸 수 있는 권한을 제거합니다. 예를 들어 OpenSearch 대상과의 Firehose 스트림을 일시 중지할 경우 권한을 업데이트하여 이 작업을 수행할 수 있습니다. 자세한 정보는 Grant Firehose Access to a Public OpenSearch Service Destination(Firehose에 퍼블릭 OpenSearch Service 대상에 대한 액세스 권한 부여)을 참조하세요.

작업 `s3:PutObject`에 대한 `"Effect": "Allow"` 권한을 제거하고, 실패한 전송의 백업에 사용되는 S3 버킷에 대해 작업 `s3:PutObject`에 대한 `Effect": "Deny"` 권한을 적용하는 명령문을 명시적으로 추가합니다. 그런 다음 스트림 대상을 끄거나(예: 대상 OpenSearch 도메인 끄기) Firehose가 대상에 쓸 수 있는 권한을 제거합니다. 다른 대상에 대한 권한을 업데이트하려면 Controlling Access with Amazon Data Firehose(Amazon Data Firehose를 사용한 액세스 제어)에서 해당 대상에 대한 섹션을 확인하세요. 이 두 작업이 완료되면 Firehose의 스트림 전송이 중단되며 Firehose용 CloudWatch 지표를 통해 이를 모니터링할 수 있습니다.

> **Important**
>
> Firehose의 스트림 전송을 일시 중지할 때는, 스트림 전송이 재개되고 데이터가 대상으로 전달될 때까지 스트림 소스(예: Kinesis Data Streams 또는 Managed Service for Kafka)가 데이터를 보존하도록 구성되어야 합니다. 소스가 DirectPUT인 경우 Firehose는 24시간 동안 데이터를 보존합니다. 데이터 보존 기간이 만료되기 전에 스트림을 재개하여 데이터를 전송하지 않으면 데이터가 손실될 수 있습니다.

Firehose 스트림 재개

전송을 재개하려면, 먼저 대상을 켜고 Firehose에 스트림을 대상으로 전송할 권한이 있는지 확인하여 스트림 대상에 대한 이전의 변경 사항을 되돌립니다. 그런 다음, 실패한 전송을 백업하기 위해 S3 버킷에 적용되는 권한에 대한 이전의 변경 사항을 되돌립니다. 즉, 작업 `s3:PutObject`에 대한 `"Effect": "Allow"` 권한을 제거하고, 실패한 전송의 백업에 사용되는 S3 버킷에 대해 작업 `s3:PutObject`에 대한 `"Effect": "Deny"` 권한을 제거합니다. 마지막으로 Firehose용 CloudWatch 지표를 통해 모니터링하여 스트림이 대상으로 전송되고 있는지 확인합니다. 오류를 확인하고 문제를 해결하려면 Firehose에 대한 Amazon CloudWatch Logs 모니터링을 사용하세요.

Amazon Data Firehose를 사용하여 Apache Iceberg 테이블에 데이터 전송

Apache Iceberg는 빅 데이터 분석을 수행하기 위한 고성능 오픈 소스 테이블 형식입니다. Apache Iceberg는 Amazon S3 데이터 레이크에 SQL 테이블의 신뢰성과 단순성을 제공하며 Spark, Flink, Trino, Hive, Impala 등의 오픈 소스 분석 엔진이 동일한 데이터를 이용해 동시에 작업할 수 있도록 합니다. 자세한 내용은 Apache Iceberg 및 고려 사항 및 제한 사항 페이지를 참조하세요.

Firehose를 사용하여 Amazon S3의 Apache Iceberg 테이블로 스트리밍 데이터를 전송할 수 있습니다. Apache Iceberg 테이블은 Amazon S3에서 자체 관리형이거나 Amazon S3 테이블에서 호스팅될 수 있습니다. 자체 관리형 Iceberg 테이블에서는 압축 및 스냅샷 만료와 같은 모든 테이블 최적화를 관리합니다. Amazon S3 Tables은 쿼리 성능을 지속적으로 개선하고 테이블 형식 데이터의 스토리지 비용을 절감하도록 설계된 기능이 내장되어 있으며 대규모 분석 워크로드에 최적화된 S3 스토리지를 제공합니다. Amazon S3 Tables에 대한 자세한 내용은 Amazon S3 Tables를 참조하세요.

이 기능을 사용하면 단일 스트림의 레코드를 다른 Apache Iceberg 테이블로 라우팅할 수 있습니다. 이러한 테이블의 레코드에 삽입, 업데이트 및 삭제 작업을 자동으로 적용할 수 있습니다. 또한를 사용하여 Amazon S3의 Apache Iceberg 테이블에서 세분화된 데이터 액세스 제어를 지원합니다 AWS Lake Formation. 에서 중앙에서 액세스 제어를 지정 AWS Lake Formation 하고 Firehose에 대한 보다 세분화된 테이블 수준 및 열 수준 권한을 제공할 수 있습니다.

고려 사항 및 제한 사항

> **Note**
>
> Firehose는 중국 리전, AWS GovCloud (US) Regions아시아 태평양(타이베이), 아시아 태평양(말레이시아), 아시아 태평양(뉴질랜드), 멕시코(중부)를 AWS 리전 제외한 모든에서 Apache Iceberg 테이블을 대상으로 지원합니다.

Firehose의 Iceberg 테이블 지원에는 다음과 같은 고려 사항 및 제한 사항이 있습니다.

- 처리량 - Direct PUT를 소스로 사용하여 Apache Iceberg 테이블에 데이터를 전송하는 경우 스트림당 최대 처리량은 미국 동부(버지니아 북부), 미국 서부(오리건), 유럽(아일랜드) 리전에서는 초당 5MiB이고, 모든 다른 AWS 리전에서는 초당 1MiB입니다. 업데이트 및 삭제 없이 Iceberg 테이블에 데이터를 삽입하고 스트림의 처리량을 늘리려면 Firehose 한도 양식을 사용하여 처리량 한도 증가를 요청할 수 있습니다.

데이터만 삽입하고 업데이트 및 삭제를 수행하지 않기 위해 AppendOnly 플래그를 True(으)로 설정할 수도 있습니다. AppendOnly 플래그를 True(으)로 설정하면 Firehose는 처리량에 맞게 자동으로 규모가 조정됩니다. 현재 이 플래그는 CreateDeliveryStream API 작업을 통해서만 설정할 수 있습니다.

Firehose 스트림의 처리량 용량을 초과하는 더 높은 데이터 수집 볼륨으로 인해 Direct PUT 스트림에 스로틀링이 발생하는 경우 Firehose는 스로틀링이 포함될 때까지 스트림의 처리량 제한을 자동으로 증가시킵니다. 증가한 처리량 및 스로틀링에 따라 Firehose가 스트림의 처리량을 원하는 수준으로 늘리는 데 더 오래 걸릴 수 있습니다. 따라서 실패한 데이터 수집 레코드를 계속 재시도합니다. 데이터 볼륨이 갑자기 큰 버스트에서 증가할 것으로 예상되거나 새 스트림에 기본 처리량 제한보다 높은 처리량이 필요한 경우에 처리량 제한 증가를 요청합니다.

- 처리량 및 파티션 크기 조정 -이 서비스는 많은 수의 Iceberg 파티션 또는 매우 높은 수집 처리량을 지원하도록 최적화되어 있습니다. 수집 처리량이 증가하면 적극적으로 쓸 수 있는 파티션 수가 감소합니다.

  다음은 지원되는 수집 처리량 및 최대 활성 파티션에 대한 제한입니다.

수집 처리량	지원되는 최대 활성 파티션
≤ 20MB/s	최대 ~3,000
20~40MB/s	1000
40~400MB/s	100
400~750MB/s	50
750MB/s~1.5GB/s	1

- S3 초당 트랜잭션(TPS) - S3 성능을 최적화하려면 Kinesis Data Streams 또는 Amazon MSK를 소스로 사용하는 경우 적절한 파티션 키를 사용하여 소스 레코드를 분할하는 것이 좋습니다. 이렇게 하면 동일한 Iceberg 테이블로 라우팅되는 데이터 레코드가 샤드라고 하는 하나 또는 몇 개의 소스 파티션에 매핑됩니다. 가능하면 소스 주제/스트림의 모든 파티션/샤드에서 사용 가능한 모든 집계 처리량을 사용할 수 있도록 서로 다른 대상 Iceberg 테이블에 속하는 데이터 레코드를 서로 다른 파티션/샤드로 분산합니다.
- 열 - 열 이름 및 값의 경우 Firehose는 다중 레벨 중첩 JSON에서 첫 번째 레벨의 노드만 사용합니다. 예를 들어 Firehose는 position 필드를 포함한 첫 번째 레벨에서 사용할 수 있는 노드를 선택합니다. 소스 데이터의 열 이름과 데이터 유형이 대상 테이블의 열 이름 및 데이터 유형과 정확히 일치해야

Firehose가 성공적으로 전송할 수 있습니다. 이 경우 Firehose는 Iceberg 테이블에 position 필드와 일치하는 구조 또는 맵 데이터 유형 열이 있을 것으로 예상합니다. Firehose는 16개의 레벨 중첩을 지원합니다. 다음은 중첩된 JSON의 예입니다.

```
{
   "version":"2016-04-01",
   "deviceId":"<solution_unique_device_id>",
   "sensorId":"<device_sensor_id>",
   "timestamp":"2024-01-11T20:42:45.000Z",
   "value":"<actual_value>",
   "position":{
      "x":143.595901,
      "y":476.399628,
      "z":0.24234876
   }
}
```

열 이름 또는 데이터 유형이 일치하지 않는 경우 Firehose에 오류가 발생하고 S3 오류 버킷에 데이터를 전송합니다. Apache Iceberg 테이블의 모든 열 이름과 데이터 유형이 일치하지만 소스 레코드에 추가 필드가 있는 경우 Firehose는 새 필드를 건너뜁니다.

- 레코드당 하나의 JSON 객체 - 하나의 Firehose 레코드에서 하나의 JSON 객체만 전송할 수 있습니다. 한 레코드 내에서 여러 JSON 객체를 집계하여 전송하는 경우 Firehose에 오류가 발생하고 S3 오류 버킷에 데이터를 전송합니다. KPL을 사용하여 레코드를 집계하고 Amazon Kinesis Data Streams를 소스로 사용하여 Firehose에 데이터를 수집하는 경우 Firehose는 자동으로 집계를 해제하고 레코드당 하나의 JSON 객체를 사용합니다.
- 압축 및 스토리지 최적화 - Firehose를 사용하여 Iceberg Tables에 기록하면 매번 스냅샷, 작은 데이터 파일, 삭제 파일을 커밋하고 생성합니다. 많은 데이터 파일을 보유하고 있으면 메타데이터 오버헤드가 증가하고 읽기 성능에 영향을 미칩니다. 효율적인 쿼리 성능을 얻으려면 주기적으로 작은 데이터 파일을 가져와 적은 수의 큰 데이터 파일로 다시 쓰는 솔루션을 고려해 볼 수 있습니다. 이 프로세스를 compaction이라고 합니다.는 Apache Iceberg 테이블의 자동 압축을 AWS Glue Data Catalog 지원합니다. 자세한 내용은 AWS Glue 사용 설명서의 Compaction management(압축 관리)를 참조하세요. 추가적인 내용은 Apache Iceberg 테이블의 자동 압축을 참조하세요. 또는 Athena Optimize 명령을 실행하여 수동으로 압축을 수행할 수 있습니다. 최적화 명령에 대한 자세한 내용은 Athena Optimize를 참조하세요.

  데이터 파일을 압축하는 방법 외에도, 스냅샷 만료 및 연결 없는 파일 제거 등 Apache Iceberg 테이블에서 테이블 유지 관리를 수행하는 VACUUM 문을 사용하여 스토리지 소비를 최적화할 수도 수도 있습니다. 또는 데이터 파일, 분리된 파일을 자동으로 제거하고 더 이상 필요하지 않은 스냅샷을 만

료시켜 Apache Iceberg 테이블의 관리형 테이블 최적화 AWS Glue Data Catalog 를 지원하는를 사용할 수 있습니다. 자세한 내용은 Apache Iceberg 테이블의 스토리지 최적화에 대한 이 블로그 게시물을 참조하세요.

- Apache Iceberg 테이블에 대한 Amazon MSK Serverless 소스는 대상으로 지원하지 않습니다.
- 업데이트 작업의 경우 Firehose는 삭제 파일과 삽입 작업을 차례로 넣습니다. 삭제 파일을 입력하면 Amazon S3 풋 요금이 발생합니다.
- Firehose는 여러 Firehose 스트림을 사용하여 동일한 Apache Iceberg 테이블에 데이터를 쓰는 것을 권장하지 않습니다. 이는 Apache Iceberg가 낙관적 동시성 제어(OCC)를 사용하기 때문입니다. 여러 Firehose 스트림이 동시에 단일 Iceberg 테이블에 쓰기를 시도하면 한 번에 하나의 스트림만 데이터를 커밋하는 데 성공합니다. 백오프 커밋에 실패한 다른 스트림은 구성된 재시도 기간이 만료될 때까지 커밋 작업을 재시도합니다. 재시도 기간이 소진되면 데이터 및 삭제 파일 키(Amazon S3 경로)가 구성된 Amazon S3 오류 접두사로 전송됩니다.
- Firehose가 지원하는 현재 Iceberg 라이브러리 버전은 버전 1.5.2입니다.
- 암호화된 데이터를 Amazon S3 Tables로 전송하려면 Firehose 구성이 아닌 Amazon S3 Tables에서 AWS Key Management Service 파라미터를 구성해야 합니다. 암호화된 데이터를 Amazon S3 Tables로 전송하기 위해 Firehose에서 AWS Key Management Service 파라미터를 구성하면 Firehose는 해당 파라미터를 사용하여 암호화할 수 없습니다. 자세한 내용은 AWS KMS 키를 사용한 서버 측 암호화 사용을 참조하세요.
- Firehose 스트림은 Iceberg의 GlueCatalog API를 통해 생성된 데이터베이스 및 테이블로의 전송만 지원합니다. Glue SDK를 통해 생성된 데이터베이스 및 테이블로의 전송은 지원되지 않습니다. 하이픈(-)은 데이터베이스 및 Iceberg 라이브러리의 테이블 이름에 대해 지원되는 문자가 아닙니다. 자세한 내용은 Iceberg 라이브러리에서 지원하는 Glue Database Regex 및 Glue Table Regex를 참조하세요.
- Firehose에서 작성한 모든 파일은 레코드에 있는 파티션을 사용하여 계산됩니다. 이는 삭제된 파일에도 적용됩니다. 분할된 테이블에 대해 분할되지 않은 삭제 파일을 작성하는 것과 같은 전역 삭제는 지원되지 않습니다.
- Firehose는 현재 Apache Iceberg 테이블로 데이터를 전송할 때 블룸 필터 속성을 지원하지 않습니다. Iceberg 테이블에 블룸 필터 속성이 구성된 경우 Firehose는 데이터 전송 작업 중에 이러한 속성을 무시합니다.

Apache Iceberg 테이블을 대상으로 사용하기 위한 사전 조건

다음 옵션 중에서 선택하여 필수 사전 조건을 완료합니다.

주제

Amazon S3의 Iceberg 테이블로 전송하기 위한 사전 조건

시작하기 전에 다음 사전 조건을 완료합니다.

- Amazon S3 버킷 생성 - 테이블 생성 중에 메타데이터 파일 경로를 추가하려면 Amazon S3 버킷을 생성해야 합니다. 자세한 내용은 S3 버킷 생성을 참조하세요.
- 필요한 권한을 가진 IAM 역할 생성 - Firehose가 AWS Glue 테이블에 액세스하고 Amazon S3에 데이터를 쓰기 위해서는 특정 권한을 가진 IAM 역할이 필요합니다. 동일한 역할을 사용하여 Amazon S3 버킷에 대한 AWS Glue 액세스 권한을 부여합니다. 해당 IAM 역할은 Iceberg 테이블과 Firehose 스트림을 생성할 때 필요합니다. 자세한 내용은 Firehose에 Amazon S3 Tables에 대한 액세스 권한 부여 단원을 참조하십시오.
- Apache Iceberg 테이블 생성 - 업데이트 및 삭제를 위해 Firehose 스트림에서 고유 키를 구성하는 경우 Firehose는 테이블과 고유 키가 스트림 생성의 일부로 존재하는지 확인합니다. 이 시나리오에서는 Firehose 스트림을 생성하기 전에 테이블을 생성해야 합니다. AWS Glue 를 사용하여 Apache Iceberg 테이블을 생성할 수 있습니다. 자세한 내용은 Creating Apache Iceberg 테이블(Apache Iceberg 테이블 생성)을 참조하세요. Firehose 스트림에서 고유 키를 구성하지 않는 경우 Firehose 스트림을 생성하기 전에 Iceberg 테이블을 생성할 필요가 없습니다.

> **Note**
>
> Firehose는 Apache Iceberg 테이블에 대해 다음의 테이블 버전과 형식을 지원합니다.
>
> - 테이블 형식 버전 - Firehose는 V2 테이블 형식만 지원합니다. V1 형식으로 테이블을 생성하지 마세요. 그렇지 않으면 오류가 발생하고 데이터가 S3 오류 버킷으로 전송됩니다.
> - 데이터 스토리지 형식 - Firehose는 Apache Iceberg 테이블에 데이터를 Parquet 형식으로 씁니다.
> - 행 수준 작업 - Firehose는 Apache Iceberg 테이블에 데이터를 쓰는 MOR(Merge-on-Read) 모드를 지원합니다.

Amazon S3 Tables로 전송하기 위한 사전 조건

Amazon S3 Tables 버킷에 데이터를 전송하려면 다음 사전 조건을 완료합니다.

- S3 테이블 버킷, 네임스페이스, 테이블 버킷의 테이블 및 Amazon S3 Tables 시작하기에 설명된 기타 통합 단계를 생성합니다. S3 테이블 카탈로그 통합 제한에 지정된 대로 S3 테이블 카탈로그 통합에 적용되는 제한으로 인해 열 이름은 소문자여야 합니다.
- 필요한 권한을 가진 IAM 역할 생성 - Firehose가 AWS AWS Glue 테이블에 액세스하고 Amazon S3 Tables 버킷의 테이블에 데이터를 쓰기 위해서는 특정 권한을 가진 IAM 역할이 필요합니다. Amazon S3 테이블 버킷의 테이블에 쓰려면 IAM 역할에 필요한 권한도 제공해야 합니다. Amazon S3 Tables 카탈로그에 필요한 권한은 사용하는 액세스 제어 모드에 따라 다릅니다.
  - IAM 액세스 제어 - Firehose 전송 역할에는 Amazon S3 Tables 리소스에 대한 직접 IAM 권한이 필요합니다.
  - Lake Formation 액세스 제어 - Firehose 전송 역할에는 테이블 리소스에 대한 액세스를 관리하기 위한 AWS AWS Lake Formation 권한이 필요합니다.는 데이터 카탈로그 리소스에 대한 세분화된 액세스 제어를 지원하는 자체 권한 모델을 AWS Lake Formation 사용합니다.

  Firehose 스트림을 생성할 때 이 IAM 역할을 구성합니다. 자세한 내용은 Firehose에 Amazon S3 Tables에 대한 액세스 권한 부여를 참조하세요.

단계별 통합에 대해서는 Amazon Amazon S3 Tables 및 Amazon Data Firehose를 사용한 데이터 스트리밍을 위한 데이터 레이크 구축 블로그를 참조하세요. 자세한 내용은 AWS 분석 서비스와 함께 Amazon S3 Tables 사용을 참조하세요.

Firehose 스트림 설정

Apache Iceberg 테이블을 대상으로 하는 Firehose 스트림을 생성하려면 다음을 구성해야 합니다.

> **Note**
>
> S3 테이블 버킷의 테이블로 전송하기 위한 Firehose 스트림 설정은 Amazon S3의 Apache Iceberg 테이블과 동일합니다.

소스 및 대상 구성

Apache Iceberg 테이블에 데이터를 전송하려면 스트림의 소스를 선택합니다.

스트림에 대한 소스를 구성하려면 소스 설정 구성을 참조하세요.

그런 다음 대상으로 Apache Iceberg 테이블을 선택하고 Firehose 스트림 이름을 입력합니다.

데이터 변환 구성

수신 스트림에서 레코드를 추가하거나 수정하는 등 데이터에 대한 사용자 지정 변환을 수행하려면 Firehose 스트림에 Lambda 함수를 추가할 수 있습니다. Firehose 스트림에서 Lambda를 사용한 데이터 변환에 대한 자세한 내용은 Amazon Data Firehose에서 소스 데이터 변환 섹션을 참조하세요.

Apache Iceberg 테이블의 경우 수신 레코드를 다른 대상 테이블로 라우팅하는 방식과 수행할 작업을 지정해야 합니다. Firehose에 필요한 라우팅 정보를 제공하는 방법 중 하나는 Lambda 함수를 사용하는 것입니다.

자세한 내용은 레코드를 다른 Iceberg 테이블로 라우팅을 참조하세요.

데이터 카탈로그 연결

Apache Iceberg는 Apache Iceberg 테이블에 데이터를 쓰려면 데이터 카탈로그가 필요합니다. Firehose는 Apache Iceberg 테이블 AWS Glue Data Catalog 용와 통합됩니다.

Firehose 스트림과 AWS Glue Data Catalog 동일한 계정 또는 교차 계정 및 Firehose 스트림과 동일한 리전(기본값) 또는 다른 리전에서를 사용할 수 있습니다.

Amazon S3 Table로 전송하고 콘솔을 사용하여 Firehose 스트림을 설정하는 경우 Amazon S3 Table 카탈로그에 해당하는 카탈로그를 선택합니다. CLI를 사용하여 Firehose 스트림을 설정하는 경우 `CatalogConfiguration` 입력에서 `CatalogARN`를 `arn:aws:glue:<region>:<account-id>:catalog/s3tablescatalog/<s3 table bucket name>` 형식과 함께 사용합니다. 자세한 내용은 Amazon S3 tables에 대해 Firehose 스트림 설정을 참조하세요.

> **Note**
>
> Firehose는 Iceberg 테이블에 대해 삽입, 업데이트 및 삭제라는 세 가지 작업을 지원합니다. 지정된 작업이 없으면 Firehose는 기본적으로 삽입하여 각 수신 레코드를 새 행으로 추가하고 중복을 보존합니다. 대신 기존 레코드를 수정하려면 프라이머리 키를 사용하여 기존 행을 찾고 변경하는 "업데이트" 작업을 지정합니다.
> 예제:
>
> - 기본값(삽입): 여러 개의 동일한 고객 레코드가 중복 행을 생성합니다.
> - 지정된 업데이트: 새 고객 주소가 기존 레코드를 업데이트합니다.

JQ 표현식 구성

Apache Iceberg 테이블의 경우 수신 레코드를 다른 대상 테이블로 라우팅하는 방식과 수행할 삽입, 업데이트, 삭제 등의 작업을 지정해야 합니다. 이 작업은 Firehose가 필요한 정보를 구문 분석하고 가져오도록 JQ 표현식을 구성하여 수행할 수 있습니다. 자세한 내용은 ??? 단원을 참조하십시오.

고유 키 구성

두 개 이상의 테이블에 대한 업데이트 및 삭제 - 고유 키는 소스 레코드에서 Apache Iceberg 테이블의 행을 고유하게 식별하는 하나 이상의 필드입니다. 두 개 이상의 테이블에서 삽입 작업만 수행하는 시나리오의 경우 고유 키를 구성할 필요가 없습니다. 특정 테이블에 업데이트나 삭제 작업을 수행하려면 필요한 테이블에 대한 고유 키를 구성해야 합니다. 업데이트 작업 시 테이블의 행이 누락된 경우 자동으로 행을 삽입합니다. 테이블이 하나뿐인 경우 고유 키를 구성할 수 있습니다. 업데이트 작업의 경우 Firehose는 삭제 파일과 삽입을 차례로 넣습니다.

Firehose 스트림 생성의 일부로 테이블당 고유 키를 구성하거나, Iceberg에서 자체적으로 create table 또는 alter table 작업 중에 identifier-field-ids를 설정할 수 있습니다. 스트림 생성 중에 테이블당 고유 키를 구성하는 작업은 선택 사항입니다. 스트림 생성 중에 테이블당 고유 키를 구성하지 않으면 Firehose는 필요한 테이블의 `identifier-field-ids`를 확인하고 이를 고유 키로 사용합니다. 둘 다 구성되지 않은 경우 업데이트 및 삭제 작업이 포함된 데이터 전송이 실패합니다.

이 섹션을 구성하려면 데이터를 업데이트하거나 삭제할 테이블의 데이터베이스 이름, 테이블 이름, 고유 키를 제공합니다. 구성에서는 각 테이블에 대해 하나의 항목만 가질 수 있습니다. 추가 전용 시나리오에 대해 이 섹션을 구성할 필요가 없습니다. 선택 사항으로 다음 예시와 같이 테이블의 데이터가 전송되지 않는 경우 오류 버킷 접두사를 지정할 수도 있습니다.

```
[
  {
    "DestinationDatabaseName": "MySampleDatabase",
    "DestinationTableName": "MySampleTable",
    "UniqueKeys": [
      "COLUMN_PLACEHOLDER"
    ],
    "S3ErrorOutputPrefix": "OPTIONAL_PREFIX_PLACEHOLDER"
  }
]
```

Firehose는 제공된 열 이름이 전체 테이블에서 고유한 경우 고유한 키의 구성을 지원합니다. 그러나 정규화된 열 이름은 고유한 키로 지원되지 않습니다. 예를 들어 열 이름 `_id`이 최상위에도 있는 경우

top._id(이)라는 키는 고유한 키로 간주되지 않습니다. _id이(가) 전체 테이블에서 고유한 경우 테이블 구조 내의 위치에 관계없이 사용됩니다ㅌ. 최상위 열이든 중첩 열이든 마찬가지입니다. 다음 예제에서는 열 이름이 스키마 전체에서 고유하기 때문에 _id은(는) 스키마에 유효한 고유 키입니다.

```
[
 "schema": {
  "type": "struct",
  "fields": [
    {
      "name": "top",
      "type": {
        "type": "struct",
        "fields": [
          { "name": "_id", "type": "string" },
          { "name": "name", "type": "string" }
        ]
      }
    },
    { "name": "user", "type": "string" }
  ]
}
]
```

다음 예제에서는 최상위 열과 중첩된 구조체 모두에서 사용되기 때문에 _id은(는) 스키마에 유효한 고유 키가 아닙니다.

```
[
"schema": {
  "type": "struct",
  "fields": [
    {
      "name": "top",
      "type": {
        "type": "struct",
        "fields": [
          { "name": "_id", "type": "string" },
          { "name": "name", "type": "string" }
        ]
      }
    },
    { "name": "_id", "type": "string" }
  ]
```

```
}
]
```

재시도 기간 지정

Amazon S3에서 Firehose가 Apache Iceberg 테이블에서 데이터 쓰기 작업에 실패하는 경우 이 구성을 사용하여 재시도하는 기간을 초 단위로 지정할 수 있습니다. 재시도를 수행하는 시간의 값은 0초에서 7,200초까지 설정할 수 있습니다. Firehose는 기본값으로 300초동안 재시도합니다.

전송 또는 처리 실패 조치

재시도 기간이 만료된 후에도 스트림 처리 또는 전송에 실패하는 경우 S3 백업 버킷에 레코드를 전송하도록 Firehose를 구성해야 합니다. 이 작업을 수행하려면 콘솔의 백업 설정에서 S3 백업 버킷 및 S3 백업 버킷 오류 출력 접두사를 구성합니다.

오류 처리

Firehose는 모든 전송 오류를 CloudWatch Logs 및 Amazon S3 오류 버킷으로 전송합니다.

오류 목록:

오류 메시지	설명
`Iceberg.NoSuchTable`	Firehose가 존재하지 않는 테이블에 쓰거나 테이블이 V2 형식이 아닙니다. Firehose는 V1 형식의 테이블을 지원하지 않습니다.
`Iceberg.InvalidTableName`	null 또는 빈 테이블 이름이 전달되거나 테이블이 V2 형식이 아닙니다. Firehose는 V1 형식의 테이블을 지원하지 않습니다.
`S3.AccessDenied`	사전 조건 단계에서 생성된 IAM 역할에 필요한 권한과 신뢰 정책이 있는지 확인합니다.
`Glue.AccessDenied`	사전 조건 단계에서 생성된 IAM 역할에 필요한 권한과 신뢰 정책이 있는지 확인합니다.

버퍼 힌트 구성

Firehose는 수신되는 스트리밍 데이터를 메모리에 일정 크기(버퍼 크기)로 일정 시간(버퍼 간격) 동안 버퍼링한 후 Apache Iceberg 테이블로 전송합니다. 버퍼 크기는 1MiB~128MiB 사이, 버퍼 간격은 0초~900초 사이로 선택할 수 있습니다. 버퍼 힌트 값이 높을수록 S3 쓰기 횟수가 줄어들고, 데이터 파일의 크기가 커져 압축 비용이 줄어들고, 쿼리 런타임이 빨라지지만 지연 시간이 길어집니다. 버퍼 힌트 값이 낮으면 데이터 전송 시 지연 시간이 짧아집니다.

고급 설정 구성

Apache Iceberg 테이블에 대해 서버 측 암호화, 오류 로깅, 권한 및 태그를 구성할 수 있습니다. 자세한 내용은 고급 설정 구성 단원을 참조하십시오. ???의 일부로 생성한 IAM 역할을 추가해야 합니다. Firehose는 AWS Glue 테이블에 액세스하고 Amazon S3 버킷에 대한 쓰기 역할을 맡게 됩니다.

Firehose 스트림 생성을 완료하는 데 몇 분 정도 걸릴 수 있습니다. Firehose 스트림을 성공적으로 생성한 후에는 데이터 수집을 시작하고 Apache Iceberg 테이블에서 데이터를 볼 수 있습니다.

수신 레코드를 단일 Iceberg 테이블로 라우팅

Firehose가 단일 Iceberg 테이블에 데이터를 삽입하도록 하려면 다음 예시 JSON과 같이 스트림 구성에서 단일 데이터베이스와 테이블을 구성하기만 하면 됩니다. 단일 테이블의 경우 Firehose에 라우팅 정보를 제공하기 위해 JQ 표현식 및 Lambda 함수가 필요하지 않습니다. 이러한 필드를 JQ 또는 Lambda와 함께 제공하는 경우 Firehose는 JQ 또는 Lambda에서 입력을 받습니다.

```
[
  {
    "DestinationDatabaseName": "UserEvents",
    "DestinationTableName": "customer_id",
    "UniqueKeys": [
      "COLUMN_PLACEHOLDER"
    ],
    "S3ErrorOutputPrefix": "OPTIONAL_PREFIX_PLACEHOLDER"
  }
]
```

이 예시에서는 Firehose가 모든 입력 레코드를 `UserEvents` 데이터베이스의 `customer_id` 테이블에 라우팅합니다. 단일 테이블에서 업데이트나 삭제 작업을 수행하려면 JSONQuery 메서드 또는 Lambda 메서드를 사용하여 각 수신 레코드에 대한 작업을 Firehose에 제공해야 합니다.

수신 레코드를 다양한 Iceberg 테이블에 라우팅

Amazon Data Firehose는 레코드의 콘텐츠에 따라 스트림의 수신 레코드를 다양한 Iceberg 테이블로 라우팅할 수 있습니다. Amazon Data Firehose에서 전송된 레코드는 순서대로 유지되지 않습니다. 다음과 같은 샘플 입력 레코드를 생각해 보겠습니다.

```
{
  "deviceId": "Device1234",
  "timestamp": "2024-11-28T11:30:00Z",
  "data": {
    "temperature": 21.5,
    "location": {
      "latitude": 37.3324,
      "longitude": -122.0311
    }
  },
  "powerlevel": 84,
  "status": "online"
}
```

```
{
  "deviceId": "Device4567",
  "timestamp": "2023-11-28T10:40:00Z",
  "data": {
    "pressure": 1012.4,
    "location": {
      "zipcode": 24567
    }
  },
  "powerlevel": 82,
  "status": "online"
}
```

이 예제에서 **deviceId** 필드에는 Device1234 및 Device4567라는 두 개의 가능한 값이 있습니다. 수신 레코드의 **deviceId** 필드가 Device1234인 경우, 이 레코드를 Device1234이라는 이름의 Iceberg 테이블에 기록하고, 수신 레코드의 **deviceId** 필드가 Device4567인 경우, 이 레코드를 Device4567(이)라는 이름의 테이블에 기록하고자 합니다.

Device1234 및 Device4567가 있는 레코드에는 해당 Iceberg 테이블의 다른 열에 매핑되는 다른 필드 세트가 있을 수 있습니다. 수신 레코드에는 중첩된 JSON 구조를 가질 수 있으며, 이 JSON 레코드

내부에 **deviceId**이 중첩될 수 있습니다. 다음 섹션에서는 이러한 시나리오에서 적절한 라우팅 정보를 Firehose에 제공하여 레코드를 다른 테이블로 라우팅하는 방법을 설명합니다.

JSONQuery 표현식을 사용하여 Firehose에 라우팅 정보 제공

Firehose에 레코드 라우팅 정보를 제공하는 가장 간단하고 비용 효율적인 방법은 JSONQuery 표현식을 사용하는 것입니다. 이 접근 방식을 사용하면 `Database Name`, `Table Name` 파라미터, 그리고 선택 사항인 `Operation` 파라미터에 대한 JSONQuery 표현식을 제공합니다. Firehose는 사용자가 제공한 표현식을 사용하여 수신 스트림 레코드에서 정보를 추출하여 레코드를 라우팅합니다.

`Database Name` 파라미터는 대상 데이터베이스의 이름을 지정합니다. `Table Name` 파라미터는 대상 테이블의 이름을 지정하고, `Operation` 파라미터는 수신 스트림 레코드를 대상 테이블에 새 레코드로 삽입할지 아니면 대상 테이블의 기존 레코드를 수정 또는 삭제할지 여부를 나타내는 선택적 파라미터입니다. Operation 필드에는 `insert`, `update` 또는 `delete` 중 하나의 값이 있어야 합니다.

이러한 세 가지 파라미터 각각에 대해 정적 값을 지정하거나, 수신 레코드에서 값을 가져오는 동적 표현식을 지정할 수 있습니다. 예를 들어 모든 수신 스트림 레코드를 `IoTevents`라는 단일 데이터베이스로 전송하려는 경우 데이터베이스 이름은 정적 값인 "`IoTevents`"가 될 수 있습니다. 대상 테이블 이름을 수신 레코드의 필드에서 가져와야 하는 경우, 테이블 이름은 동적 표현식이 될 수 있으며, 수신 레코드에서 대상 테이블 이름을 가져올 특정 필드를 지정할 수 있습니다.

다음 예제에서는 Database Name에 정적 값, Table Name에 동적 값, Operation에 정적 값을 사용합니다. Operation 값을 지정하는 것은 선택 사항입니다. 이 값이 지정되지 않은 경우 Firehose는 기본값으로 수신 레코드를 대상 테이블에 새 레코드로 삽입합니다.

```
Database Name : "IoTevents"
Table Name : .deviceId
Operation : "insert"
```

`deviceId` 필드는 JSON 레코드 내에 중첩되어 있는 경우 Table Name을 중첩된 필드 정보가 포함된 `.event.deviceId`(으)로 지정합니다.

> **Note**
>
> - 작업을 `update` 또는 `delete`로 지정하는 경우 Firehose 스트림을 설정할 때 대상 테이블에 고유 키를 지정하거나, Iceberg에서 create table 또는 alter table 작업을 실행할 때 identifier-field-ids를 설정해야 합니다. 이를 지정하지 않으면 Firehose에 오류가 발생하고 S3 오류 버킷에 데이터를 전송합니다.

- `Database Name` 및 `Table Name` 값은 대상 데이터베이스 이름 및 테이블 이름과 정확히 일치해야 합니다. 일치하지 않는 경우 Firehose에 오류가 발생하고 S3 오류 버킷에 데이터를 전송합니다.

AWS Lambda 함수를 사용하여 라우팅 정보 제공

수신 레코드를 대상 테이블로 라우팅하는 방식을 결정하는 복잡한 규칙을 가진 시나리오가 있을 수 있습니다. 예를 들어 필드에 A, B 또는 F라는 값이 포함된 경우 TableX라는 대상 테이블로 라우팅하도록 규칙을 정의하거나, 추가적인 속성을 더해 수신되는 스트림 레코드를 보강할 수 있습니다. 예를 들어, 레코드의 `device_id` 필드 값이 1인 경우, 값이 “modem”인 다른 필드 `device_type`을(를) 추가하여 대상 테이블 열에 추가적인 필드를 작성할 수 있습니다. 이러한 경우 Firehose의 AWS Lambda 함수를 사용하여 소스 스트림을 변환하고 Lambda 변환 함수 출력의 일부로 라우팅 정보를 제공할 수 있습니다. Firehose에서 AWS Lambda 함수를 사용하여 소스 스트림을 변환하는 방법을 이해하려면 Amazon Data Firehose에서 소스 데이터 변환을 참조하세요.

Firehose에서 소스 스트림 변환을 위해 Lambda를 사용하는 경우 출력에 `recordId`, `result` 및 `data` 또는 `KafkaRecordValue` 파라미터가 포함되어야 합니다. `recordId` 파라미터에는 입력 스트림 레코드가 포함되어 있고, `result`는 변환 성공 여부를 나타내며, `data`는 Lambda 함수의 변환된 출력이 Base64로 인코딩되어 포함되어 있습니다. 자세한 내용은 ??? 단원을 참조하십시오.

```
{
  "recordId": "49655962066601463032522589543535113056108699331451682818000000",
  "result": "Ok",
  "data": "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1"
}
```

Lambda 함수의 일부로 스트림 레코드를 대상 테이블로 라우팅하는 방법에 대한 라우팅 정보를 Firehose에 지정하려면 Lambda 함수의 출력에 `metadata`에 대한 추가 섹션이 포함되어야 합니다. 다음 예시는 Kinesis Data Streams를 데이터 소스로 사용하는 Firehose 스트림의 Lambda 출력에 메타데이터 섹션을 추가하여 `IoTevents` 데이터베이스의 `Device1234`이라는 테이블에 새 레코드를 삽입하도록 Firehose에 지시하는 방법을 보여줍니다.

```
 {
"recordId": "49655962066601463032522589543535113056108699331451682818000000",
    "result": "Ok",
    "data":
 "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1",
```

```
    "metadata":{
"otfMetadata":{
            "destinationTableName":"Device1234",
            "destinationDatabaseName":"IoTevents",
            "operation":"insert"
        }
    }
  }
```

마찬가지로 다음 예시는 Amazon Managed Streaming for Apache Kafka를 데이터 소스로 사용하는 Firehose의 Lambda 출력에 메타데이터 섹션을 추가하여 `IoTevents` 데이터베이스의 `Device1234`이라는 테이블에 새 레코드를 삽입하도록 Firehose에 지시하는 방법을 보여줍니다.

```
{
"recordId": "496559620666014630325225895435351130561086993314516828180000000",
    "result": "Ok",
    "kafkaRecordValue":
 "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1",

    "metadata":{
"otfMetadata":{
            "destinationTableName":"Device1234",
            "destinationDatabaseName":"IoTevents",
            "operation":"insert"
        }
    }
  }
```

이 예시에 대한 설명은 다음과 같습니다.

- `destinationDatabaseName`은 대상 데이터베이스의 이름을 나타내며 필수 필드입니다.
- `destinationTableName`은 대상 테이블의 이름을 나타내며 필수 필드입니다.
- `operation`은 가능한 값이 `insert`, `update`, `delete`인 선택 필드입니다. 값을 지정하지 않으면 기본값은 `insert`입니다.

> **Note**
>
> - 작업을 `update` 또는 `delete`로 지정하는 경우 Firehose 스트림을 설정할 때 대상 테이블에 고유 키를 지정하거나, Iceberg에서 create table 또는 alter table 작업을 실행할 때 identifier-

field-ids를 설정해야 합니다. 이를 지정하지 않으면 Firehose에 오류가 발생하고 S3 오류 버킷에 데이터를 전송합니다.

- `Database Name` 및 `Table Name` 값은 대상 데이터베이스 이름 및 테이블 이름과 정확히 일치해야 합니다. 일치하지 않는 경우 Firehose에 오류가 발생하고 S3 오류 버킷에 데이터를 전송합니다.
- Firehose 스트림에 Lambda 변환 함수와 JSONQuery 표현식이 모두 있는 경우, Firehose는 먼저 Lambda 출력의 메타데이터 필드를 확인하여 레코드를 적절한 대상 테이블로 어떻게 라우팅할지 결정한 다음 JSONQuery 표현식의 출력에서 누락된 필드가 있는지 확인합니다.

  Lambda 또는 JSONQuery 표현식이 필요한 라우팅 정보를 제공하지 않는 경우 Firehose는 이를 단일 테이블 시나리오로 가정하고 고유 키 구성에서 단일 테이블 정보를 찾습니다.

  자세한 정보는 수신 레코드를 단일 Iceberg 테이블로 라우팅을 참조하세요. Firehose가 라우팅 정보를 결정하지 못하고 레코드를 지정된 대상 테이블에 일치시키지 못하면 지정된 S3 오류 버킷으로 데이터를 전송합니다.

샘플 Lambda 함수

이 Lambda 함수는 수신 스트림 레코드를 구문 분석하고 필수 필드를 추가하여 특정 테이블에 데이터를 기록하는 방법을 지정하는 샘플 Python 코드입니다. 아래 샘플 코드를 사용하여 라우팅 정보에 대한 메타데이터 섹션을 추가할 수 있습니다.

```
import json
import base64


def lambda_handler(firehose_records_input, context):
    print("Received records for processing from DeliveryStream: " +
 firehose_records_input['deliveryStreamArn'])

    firehose_records_output = {}
    firehose_records_output['records'] = []


    for firehose_record_input in firehose_records_input['records']:

        # Get payload from Lambda input, it could be different with different sources
        if 'kafkaRecordValue' in firehose_record_input:
```

```
            payload_bytes =
base64.b64decode(firehose_record_input['kafkaRecordValue']).decode('utf-8')
        else
            payload_bytes =
base64.b64decode(firehose_record_input['data']).decode('utf-8')

        # perform data processing on customer payload bytes here

        # Create output with proper record ID, output data (may be different with
different sources), result, and metadata
        firehose_record_output = {}

        if 'kafkaRecordValue' in firehose_record_input:
            firehose_record_output['kafkaRecordValue'] =
base64.b64encode(payload_bytes.encode('utf-8'))
        else
            firehose_record_output['data'] =
base64.b64encode(payload_bytes.encode('utf-8'))

        firehose_record_output['recordId'] = firehose_record_input['recordId']
        firehose_record_output['result'] =  'Ok'
        firehose_record_output['metadata'] = {
            'otfMetadata': {
                'destinationDatabaseName': 'your_destination_database',
                'destinationTableName': 'your_destination_table',
                'operation': 'insert'
                }
            }
        firehose_records_output['records'].append(firehose_record_output)
    return firehose_records_output
```

지표 모니터링

Apache Iceberg 테이블로 데이터를 전송할 때 Firehose는 스트림 수준에서 다음과 같은 CloudWatch 지표를 전송합니다.

지표	설명
DeliveryToIceberg.Bytes	지정된 시간 동안 Apache Iceberg 테이블로 전송된 데이터의 바이트 수입니다.

지표	설명
	단위: 바이트
DeliveryToIceberg.IncomingRowCount	Firehose가 Apache Iceberg 테이블에 전송하려고 시도한 레코드 수입니다. 단위: 개
DeliveryToIceberg.SuccessfulRowCount	Apache Iceberg 테이블에 성공적으로 전송된 행 수입니다. 단위: 개
DeliveryToIceberg.FailedRowCount	전송에 실패하여 S3 백업 버킷에 전송된 행 수입니다. 단위: 개
DeliveryToIceberg.DataFreshness	Firehose에서 가장 최신 레코드의 경과 시간(Firehose에 입력된 시점부터 현재까지)입니다. 이 경과 시간보다 최신인 레코드는 모두 Apache Iceberg에 전송되었습니다. 단위: 초
DeliveryToIceberg.Success	Apache Iceberg 테이블에 성공적으로 커밋된 총 횟수입니다.
JQProcessing.Duration	JQ 표현식을 실행하는 데 걸린 시간입니다. 단위: 밀리초

지원되는 데이터 유형 이해

Firehose는 Apache Iceberg가 지원하는 모든 기본 데이터 유형 및 복합 데이터 유형을 지원합니다. 자세한 정보는 스키마 및 데이터 유형을 참조하세요. 이진 데이터를 문자열로 전송할 때는 Firehose에서 지원하는 인코딩 유형(기본 Base64, MIME Base64, URL 및 파일 이름에 안전한 Base64, Hex)을 사용해야 합니다. Timestamp 데이터 유형의 경우 항상 마이크로초 단위로 전송해야 합니다.

데이터 유형 예제

다음 섹션에서는 다양한 데이터 유형 예제를 보여줍니다.

MapType


```
{
    "destination_column_0":
 {"WP5o0JOkuIQcDPcsvpJJygF1xzaOSq0wUlgTwuIeCEzgVneGxA":"PO3ReF3auyDqbfonx9Cd8NTmcQnqnw7JuZOCWwI
    "destination_column_1": "{\"{\\\"destination_nested_column_0\\\": \
\\"18:56:14.974\\\", \\\"destination_nested_column_1\\\": 241.86246}\":
\"M07kAvYdHvBh61F7RzfxtEd39YQI33LnM2NbGS67DOFFsRUyUUujKT5VnK7Wtfz1mHNeIix6FAY9cYpwTdedgr9XnFwG0
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 562.56384}\":
\"9GlxhDCt95LxBo51HybBZihqOqf6EU8jrDu7NMpxtGB2dY6q6kXpvxIrFuMdqHCJKIZIcDikwggLniUm8kgE4d
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 496.03268}\":
\"keTJZYLNvLRB50DMKzEI6M0AM4mueyNnA1m2YVnYdDwyxUpPqkb72Q6LiX0B9s8gCjZ6trW6C1PFk9KNBIpxYsj5Tc5Xs
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\\\", \\
\"destination_nested_column_1\\\": 559.0878}\":
\"mG0ZET84BUF28E312UCIWgmypyQFSUODH9NAMAnF3LJEutbooZWcBt97PP5AHaopNvC8pQZ4mGXB9hmVmjUNmuj5Qanyx
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 106.845245}\":
\"aidoVYrzu8gcLRkVVUyTKCN9gqTUFYi8uJQsrXEFEYl1f9ool7JhAtg9QKG5BBu67Ngb95ENsNKQyCHNImSu5x4hMnmHU
\"}"
}
```


DecimalType


```
{
    "destination_column_0": 9455262425851.1342772,
    "destination_column_1": "9455262425851.1342772",
    "destination_column_2": 9455262425852
}
```


BinaryType(base64-default, base64-mime, base64-url-safe, hex)


```
{
    "destination_column_0": "AsYhnHD\/Ra54hITl1daNV9glOjtWPEfopH
+PjgUKHYB6K7UcYi4K19b80wD4J\/93x5tyh+0y
+k5cMljVRlmfIkIuLxl9ERBiPPLhf4+yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA
+CBAWXfIuiG420JSQP5iAx5xFG\/
mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6RzdJlHhUDjoAXg
+4cvly27F82XpuGMNwpUj98AOrgbh2MoU9yvsM9ZrjD0eGVgOZP8Ky7Za4oE\/oK8j
+qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno+LYF5ZsySs2rB5AbVM73RfOPqdS\/c\/
r3MEqoEqt+nPx6eGam4WSA+0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY
\/8Bvy+4byoPlmr4Z5yhN1z3ZTOkx7eDR6xMv+vDVsDBtItVazDwHgDy41r
```

```
\/hQNeNedPKrozc8TY9k7wZre\/6V2lCa3BmT8Uu9b9ydjR9z+fCSdG
+VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZtyiZOaTGIj9rO0xX\/
W5dGe9\/4YChs6LbD584kxLTxvHgSl4vadaTGNKci3SvNmZNsz8ducxtNXF\/Tv2DUub465hzgpaLPur3+MB
+kfdN2YXUfqb
+xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvPRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7+yJwCB8qxhTTryxo
+bjtai4ndRCGcuCaxT8KkOcXsS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8+EYmcUIVB9XcNdke4zoUfhVQoruwi
\/kFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDSu\/r3wUqROa2tGK5\/
pQY24v+Jq0U\/jQ99GShlU283nZ85ot2ocbtMAgD\/WsrSEh6lNt9RaI3HfA7\/HcH\/
fgr9jsTtxDgZhabTBwwDwX0zjWGx1bCuTLKBN7byxg9ZvAVgqwPS4HERLer5T5UkKf74zn9Eq3HYH1Q5JpyDUx
+im7mte1sprf1+A24kksVU\/MD9aP9N8\/QDsQ13gkhOn5KwFMz3BC2Vw5gL
+gGNHFKDRL6wGIfhuYcx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm
+NO5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89+Rw==",
    "destination_column_1": "AsYhnHD\/Ra54hITl1daNV9glOjtWPEfopH
+PjgUKHYB6K7UcYi4K19b80wD4J\/93x5tyh+0y+k5c\r
\nMljVRlmfIkIuLxl9ERBiPPLhf4+yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA+CBAWXfI\r
\nuiG420JSQP5iAx5xFG\/mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6R\r
\nzdJlHhUDjoAXg+4cvly27F82XpuGMNwpUj98AOrgbh2MoU9yvsM9ZrjD0eGVgOZP8Ky7Za4oE\/oK\r
\n8j+qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno+LYF5ZsySs2rB5AbVM73RfOPqdS\/c\/r3MEqoEqt+
\r\nnPx6eGam4WSA+0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY\/8Bvy+4byoPlmr4Z5yhN1z
\r\n3ZTOkx7eDR6xMv+vDVsDBtItVazDwHgDy41r\/hQNeNedPKrozc8TY9k7wZre\/6V2lCa3BmT8Uu9b
\r\n9ydjR9z+fCSdG+VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZ
\r\ntyiZOaTGIj9rO0xX\/W5dGe9\/4YChs6LbD584kxLTxvHgSl4vadaTGNKci3SvNmZNsz8ducxtNXF
\/\r\nTv2DUub465hzgpaLPur3+MB+kfdN2YXUfqb+xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvP
\r\nRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7+yJwCB8qxhTTryxo+bjtai4ndRCGcuCaxT8KkOcXs\r
\nS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8+EYmcUIVB9XcNdke4zoUfhVQoruwidzDU\/k\r
\nFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDSu\/r3wUqROa2tGK5\r
\n\/pQY24v+Jq0U\/jQ99GShlU283nZ85ot2ocbtMAgD\/WsrSEh6lNt9RaI3HfA7\/HcH\/fgr9jsTtxDg
\r\nZhabTBwwDwX0zjWGx1bCuTLKBN7byxg9ZvAVgqwPS4HERLer5T5UkKf74zn9Eq3HYH1Q5JpyDUx+\r
\nim7mte1sprf1+A24kksVU\/MD9aP9N8\/QDsQ13gkhOn5KwFMz3BC2Vw5gL+gGNHFKDRL6wGIfhuYc
\r\nx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm+NO\r
\n5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89+Rw==",
    "destination_column_2": "AsYhnHD_Ra54hITl1daNV9glOjtWPEfopH-
PjgUKHYB6K7UcYi4K19b80wD4J_93x5tyh-0y-k5cMljVRlmfIkIuLxl9ERBiPPLhf4-
yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA-
CBAWXfIuiG420JSQP5iAx5xFG_mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6RzdJlHhUDjoAX
qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno-LYF5ZsySs2rB5AbVM73RfOPqdS_c_r3MEqoEqt-
nPx6eGam4WSA-0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY_8Bvy-4byoPlmr4Z5yhN1z3ZTOkx7eDR6xMv-
vDVsDBtItVazDwHgDy41r_hQNeNedPKrozc8TY9k7wZre_6V2lCa3BmT8Uu9b9ydjR9z-fCSdG-
VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZtyiZOaTGIj9rO0xX_W5dGe9_4YChs6LbD
MB-kfdN2YXUfqb-
xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvPRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7-
yJwCB8qxhTTryxo-bjtai4ndRCGcuCaxT8KkOcXsS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8-
EYmcUIVB9XcNdke4zoUfhVQoruwidzDU_kFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDS
Jq0U_jQ99GShlU283nZ85ot2ocbtMAgD_WsrSEh6lNt9RaI3HfA7_HcH_fgr9jsTtxDgZhabTBwwDwX0zjWGx1bCuTLKBN7
```

```
im7mte1sprf1-A24kksVU_MD9aP9N8_QDsQ13gkhOn5KwFMz3BC2Vw5gL-
gGNHFKDRL6wGIfhuYcx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm-
NO5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89-Rw==",
    "destination_column_3":
 "02c6219c70ff45ae788484e5d5d68d57d8253a3b563c47e8a47f8f8e050a1d807a2bb51c622e0ad7d6fcd300f827f
}
```

TimeType(마이크로초 단위 Epoch, LocalTime Java 객체)

```
{
    "destination_column_0": 68175096000,
    "destination_column_1": "18:56:15.096"
}
```

TimestampType.withZone(마이크로초 단위 Epoch, OffsetDateTime Java 객체, LocalDateTime Java 객체)

```
{
    "destination_column_0": 1725476175099000,
    "destination_column_1": "2024-09-04T18:56:15.099Z",
    "destination_column_2": "2024-09-04T18:56:15.099"
}
```

DoubleType

```
{
    "destination_column_0": 9.18477568715142,
    "destination_column_1": "9.18477568715142"
}
```

BooleanType

```
{
    "destination_column_0": true,
    "destination_column_1": "false",
    "destination_column_2": 1,
    "destination_column_3": 0
}
```

FloatType

```
{
    "destination_column_0": 0.6242226,
    "destination_column_1": "0.6242226"
}
```

IntegerType

```
{
    "destination_column_0": 7,
    "destination_column_1": "7"
}
```

TimestampType.withoutZone(마이크로초 단위 Epoch, LocalDateTime Java 객체, OffsetDateTime Java 객체, ZonedDateTime Java 객체)

```
{
    "destination_column_0": 1725476175114000,
    "destination_column_1": "2024-09-04T18:56:15.114",
    "destination_column_2": "2024-09-04T18:56:15.114Z",
    "destination_column_3": "2024-09-04T18:56:15.114-07:00"
}
```

DateType

```
{
    "destination_column_0": 19970,
    "destination_column_1": "2024-09-04"
}
```

LongType

```
{
    "destination_column_0": 8,
    "destination_column_1": "8"
}
```

UUIDType(UUID Java 객체)

```
{
    "destination_column_0": "21c5521c-a6d4-48d4-b2c8-7f6d842f72c3"
```

```
}
```

ListType

```
{
    "destination_column_0":
 ["s1FSrgb0lGDxfn2iYT0EtlP47aHSjwmLZgrdr1JqRs0dmbeCcQoaLr4Xhi2KIVvmus9ppFdpWIcOHnJ0omhAPhXH0yns
    "destination_column_1": "[{\"destination_nested_column_0\":\"bb00f8e6-
db82-4241-a5c5-0d9c0d2f71a4\",\"destination_nested_column_1\":907.35345},
{\"destination_nested_column_0\":\"2c77b702-d405-4fe1-beee-fb541d7ab833\",
\"destination_nested_column_1\":544.0026},{\"destination_nested_column_0\":
\"68389200-d6b1-413d-bcd9-fdb931708395\",\"destination_nested_column_1\":153.683},
{\"destination_nested_column_0\":\"bc31cbaa-39cd-4e2f-b357-9ea9ce75532b\",
\"destination_nested_column_1\":977.5165},{\"destination_nested_column_0\":
\"b7d627f9-0d5b-41b7-903a-525488259fba\",\"destination_nested_column_1\":434.17215},
{\"destination_nested_column_0\":\"06b6ec1e-1952-4582-b285-46aaf40064b8\",
\"destination_nested_column_1\":580.33124},{\"destination_nested_column_0\":
\"f04b3bbf-61ad-4c5c-8740-6f666f57c431\",\"destination_nested_column_1\":550.75793}]"
}
```

리소스

다음 리소스에서 자세히 알아보세요.

- Amazon Data Firehose를 사용하여 Amazon S3의 Apache Iceberg 테이블로 실시간 데이터 스트리밍
- Apache Iceberg 및 Amazon Data Firehose를 사용한 AWS WAF 로그 분석 간소화
- Amazon S3 Tables 및 Amazon Data Firehose를 사용하여 데이터 스트리밍을 위한 데이터 레이크 구축

Firehose 스트림에 태그 지정

Amazon Data Firehose에서 생성한 Firehose 스트림에 고유한 메타데이터를 태그 형태로 할당할 수 있습니다. 태그는 스트림에 대해 정의된 키-값 페어입니다. 태그를 사용하는 것은 AWS 리소스를 관리하고 결제 데이터를 포함한 데이터를 구성하는 간단하지만 강력한 방법입니다.

CreateDeliveryStream을 호출하여 새 Firehose 스트림을 생성할 때 태그를 지정할 수 있습니다. 기존 Firehose 스트림의 경우 다음 세 가지 작업을 사용하여 태그를 추가, 나열 및 제거할 수 있습니다.

- TagDeliveryStream
- ListTagsForDeliveryStream
- UntagDeliveryStream

태그의 기본 사항 이해

Amazon Data Firehose API 작업을 사용하여 다음 작업을 수행할 수 있습니다.

- Firehose 스트림에 태그를 추가합니다.
- Firehose 스트림에 대한 태그를 나열합니다.
- Firehose 스트림에서 태그를 제거합니다.

태그를 사용하여 Firehose 스트림을 분류할 수 있습니다. 예를 들어, 용도, 소유자 또는 환경별로 Firehose 스트림을 분류할 수 있습니다. 각 태그에 대해 키와 값이 정의되기 때문에 특정 요구를 충족하는 사용자 지정 카테고리 세트를 생성할 수 있습니다. 예를 들어, 태그 세트를 정의하여 소유자 및 연관된 애플리케이션에 따라 Firehose 스트림을 추적할 수 있습니다.

다음은 몇 가지 태그의 예입니다.

- `Project: Project name`
- `Owner: Name`
- `Purpose: Load testing`
- `Application: Application name`
- `Environment: Production`

`CreateDeliveryStream` 작업에서 태그를 지정하는 경우 Amazon Data Firehose는 `firehose:TagDeliveryStream` 작업에서 추가적인 권한 인증을 수행하여 사용자에게 태그를 생성할 권한이 있는지 확인합니다. 이 권한을 제공하지 않으면 IAM 리소스 태그가 있는 새 Firehose 스트림을 생성하려는 요청이 다음과 같이 `AccessDeniedException` 오류와 함께 실패합니다.

```
AccessDeniedException
User: arn:aws:sts::x:assumed-role/x/x is not authorized to perform:
 firehose:TagDeliveryStream on resource: arn:aws:firehose:us-east-1:x:deliverystream/x
 with an explicit deny in an identity-based policy.
```

다음 예제는 사용자가 Firehose 스트림을 생성하고 태그를 적용할 수 있도록 허용하는 정책을 보여줍니다.

태그 지정으로 비용 추적

태그를 사용하여 AWS 비용을 분류하고 추적할 수 있습니다. Firehose 스트림을 포함한 리소스에 AWS 태그를 적용하면 AWS 비용 할당 보고서에는 태그별로 집계된 사용량 및 비용이 포함됩니다. 비즈니스 범주를 나타내는 태그(예: 비용 센터, 애플리케이션 이름 또는 소유자)를 적용하여 여러 서비스에 대한 비용을 정리할 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서의 사용자 지정 결제 보고서에 비용 할당 태그 사용 섹션을 참조하세요.

태그 제한 파악

Amazon Data Firehose의 태그에는 다음과 같은 제한 사항이 적용됩니다.

기본 제한 사항

- 리소스(스트림)당 최대 태그 수는 50개입니다.
- 태그 키와 값은 대/소문자를 구분합니다.
- 삭제된 스트림에 대해 태그를 변경하거나 편집할 수 없습니다.

태그 키 제한 사항

- 각 태그 키는 고유해야 합니다. 이미 사용 중인 키를 가진 태그를 추가하면 기존 키-값 쌍에 새 태그가 덮어쓰기 됩니다.
- `aws:`를 사용하여 태그 키를 시작할 수 없습니다. 이 접두사는 AWS용으로 예약되어 있기 때문입니다. AWS 는 이 접두사로 시작되는 태그를 생성하지만, 사용자는 이를 편집하거나 삭제할 수 없습니다.

- 태그 키의 길이는 유니코드 1~128자여야 합니다.
- 태그 키의 문자로는 유니코드 문자, 숫자, 공백 그리고 _ . / = + - @ 같은 특수 문자가 허용됩니다.

태그 값 제한 사항

- 태그 값의 길이는 유니코드 0~255자여야 합니다.
- 태그 값은 공백 상태로 둘 수 있습니다. 아니면 유니코드 문자, 숫자, 공백 그리고 _ . / = + - @ 같은 특수 문자를 사용할 수 있습니다.

Amazon Data Firehose의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이를 클라우드의 보안과 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사자는 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Data Firehose에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내의AWS 서비스를 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 데이터의 민감도, 조직의 요건 및 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Data Firehose 사용 시 책임 분담 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 항목에서는 보안 및 규정 준수 목표를 충족하도록 Data Firehose를 구성하는 방법을 보여줍니다. 또한 Data Firehose 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

- Amazon Data Firehose의 데이터 보호
- Amazon Data Firehose를 통한 액세스 제어
- Amazon Data Firehose AWS Secrets Manager 에서 로 인증
- Amazon Data Firehose 콘솔을 통해 IAM 역할 관리
- Amazon Data Firehose의 규정 준수 이해
- Amazon Data Firehose의 복원력
- Amazon Data Firehose의 인프라 보안 이해
- Amazon Data Firehose의 보안 모범 사례 구현

Amazon Data Firehose의 데이터 보호

Amazon Data Firehose는 TLS 프로토콜을 사용하여 전송 중인 모든 데이터를 암호화합니다. 또한 처리 중에 임시 스토리지에 저장된 데이터에 대해 Amazon Data Firehose는 AWS Key Management Service를 사용하여 데이터를 암호화하고 체크섬(checksum) 검증을 사용하여 데이터 무결성을 확인합니다.

민감한 데이터가 있는 경우, Amazon Data Firehose를 사용할 때 서버 측 데이터 암호화를 활성화할 수 있습니다. 이 작업은 데이터 소스에 따라 다릅니다.

> **Note**
>
> 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-2 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하세요.

Kinesis Data Streams를 사용한 서버 측 암호화

데이터 생산자에서 데이터 스트림으로 데이터를 전송하면 Kinesis Data Streams는 저장 데이터를 저장하기 전에 AWS Key Management Service (AWS KMS) 키를 사용하여 데이터를 암호화합니다. Firehose 스트림이 사용자 데이터 스트림의 데이터를 읽으면 Kinesis Data Streams은 먼저 데이터를 복호화한 다음 Amazon Data Firehose로 전송합니다. Amazon Data Firehose는 지정된 버퍼링 힌트에 따라 메모리의 데이터를 버퍼링합니다. 그런 다음 암호화되지 않은 데이터를 저장하지 않고 대상으로 전송합니다.

Kinesis Data Streams의 서버 측 암호화 활성화 방법에 대한 자세한 내용은 Amazon Kinesis Data Streams 개발자 안내서의 서버 측 암호화 사용을 참조하세요.

Direct PUT 또는 다른 데이터 원본을 사용한 서버 측 암호화

PutRecord 또는 PutRecordBatch를 사용하여 Firehose 스트림으로 데이터를 보내거나 AWS IoT Amazon CloudWatch Logs 또는 CloudWatch Events를 사용하여 데이터를 보내는 경우 StartDeliveryStreamEncryption 작업을 사용하여 서버 측 암호화를 켤 수 있습니다.

서버 측 암호화를 중지하려면 StopDeliveryStreamEncryption 작업을 사용합니다.

또한 Firehose 스트림을 생성할 때 SSE를 활성화할 수도 있습니다. 이를 위해, CreateDeliveryStream을 호출할 때 DeliveryStreamEncryptionConfigurationInput을 지정합니다.

를 성공적으로 사용하려면 호출자의 IAM 정책과 KMS 키 정책 CUSTOMER_MANAGED_CMK모두에서 kms:GenerateDataKey 및 kms:Decrypt 작업을 허용해야 합니다. Firehose는 CUSTOMER_MANAGED_CMK 암호화를 사용하여 PutRecord 또는 PutRecordBatch를 호출할 때 이러한 권한을 검증합니다. 또한 CUSTOMER_MANAGED_CMK 암호화를 사용하여 CreateDeliveryStream 또는 StartDeliveryStreamEncryption을 호출할 때 kms:CreateGrant 권한이 필요합니다.

CUSTOMER_MANAGED_CMK의 유형이 CMK이고 KMSNotFoundException, KMSInvalidStateException, KMSDisabledException 또는 KMSAccessDeniedException으로 인해 Amazon Data Firehose 서비스가 레코드를 암호 해독할 수 없는 경우 이 서비스는 사용자가 해당 문제를 해결할 때까지 최대 24시간(보존 기간) 동안 대기합니다. 보존 기간을 초과하여 문제가 지속되는 경우 서비스에서는 보존 기간이 경과했으며 암호 해독할 수 없는 레코드를 건너뛴 다음 데이터를 삭제합니다. Amazon Data Firehose는 네 가지 AWS KMS 예외를 추적하는 데 사용할 수 있는 다음 네 가지 CloudWatch 지표를 제공합니다.

- KMSKeyAccessDenied
- KMSKeyDisabled
- KMSKeyInvalidState
- KMSKeyNotFound

이러한 4가지 지표에 대한 자세한 내용은 the section called “CloudWatch 지표를 사용한 모니터링” 섹션을 참조하세요.

> **Important**
>
> Firehose 스트림을 암호화하려면 대칭 CMK를 사용합니다. Amazon Data Firehose는 비대칭 CMK를 지원하지 않습니다. 대칭 및 비대칭 CMKs에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 대칭 및 비대칭 CMKs 정보를 참조하세요.

> **Note**
>
> 고객 관리형 키(CUSTOMER_MANAGED_CMK)를 사용하여 Firehose 스트림에 대한 서버 측 암호화(SSE)를 사용하도록 설정하면 Firehose 서비스는 키를 사용할 때마다 암호화 컨텍스트를 설정합니다. 이 암호화 컨텍스트는 AWS 계정이 소유한 키가 사용된 경우를 나타내므로 AWS 계정에 대한 AWS CloudTrail 이벤트 로그의 일부로 로깅됩니다. 이 암호화 컨텍스트는

Firehose 서비스에 의해 시스템에서 생성됩니다. 애플리케이션은 Firehose 서비스가 설정한 암호화 컨텍스트의 형식이나 내용에 대해 어떠한 가정도 해서는 안 됩니다.

Amazon Data Firehose를 통한 액세스 제어

다음 섹션에서는 Amazon Data Firehose 리소스와의 상호 액세스를 제어하는 방법을 다룹니다. 이들 섹션에서는 Firehose 스트림으로 데이터를 보낼 수 있도록 애플리케이션 액세스 권한을 부여하는 방법에 대한 정보도 다룹니다. 또한 Amazon Data Firehose에 Amazon Simple Storage Service(Amazon S3) 버킷, Amazon Redshift 클러스터 또는 Amazon OpenSearch Service 클러스터에 대한 액세스 권한을 부여하는 방법과 Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Splunk 또는 Sumo Logic을 대상으로 사용하는 경우 필요한 액세스 권한을 부여하는 방법도 설명합니다. 마지막으로, 이 항목에서는 다른 AWS 계정에 속한 대상으로 데이터를 전송할 수 있도록 Amazon Data Firehose를 구성하는 방법에 관한 지침도 제공합니다. 이러한 모든 형태의 액세스를 관리하는 기술은 AWS Identity and Access Management (IAM)입니다. IAM에 대한 자세한 내용은 IAM이란? 섹션을 참조하세요.

내용

- Firehose에 Snowflake 대상에 대한 액세스 권한 부여
- VPC에서 Snowflake에 액세스
- Firehose에 HTTP 엔드포인트 대상에 대한 액세스 권한 부여
- Amazon MSK에서 계정 간 전송
- Amazon S3 대상으로 교차 계정 전송
- OpenSearch Service 대상으로 교차 계정 전송
- 태그를 사용하여 액세스 제어

Firehose 리소스에 대한 액세스 권한 부여

Firehose 스트림에 애플리케이션 액세스 권한을 부여하려면 다음 예와 비슷한 정책을 사용하세요. `Action` 섹션을 수정하여 액세스 권한을 부여할 개별 API 작업을 조정하거나 `"firehose:*"`를 이용해 모든 작업에 대한 액세스를 허용할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DeleteDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:UpdateDestination"
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/delivery-stream-name"
            ]
        }
    ]
}
```

Firehose에 프라이빗 Amazon MSK 클러스터 액세스 권한 부여

Firehose 스트림의 소스가 프라이빗 Amazon MSK 클러스터인 경우 이 예시와 유사한 정책을 사용합니다.

클러스터의 리소스 기반 정책에 이와 같은 정책을 추가하여 Firehose 서비스 주체에 Amazon MSK `CreateVpcConnection` API 작업을 호출할 수 있는 권한을 부여해야 합니다.

Firehose의 IAM 역할 수임 허용

이 섹션에서는 소스에서 대상으로 데이터를 수집, 처리, 전송할 수 있는 액세스 권한을 Amazon Data Firehose에 부여하는 권한 및 정책을 설명합니다.

> **Note**
>
> 콘솔을 사용하여 Firehose 스트림을 생성하고 새 역할을 생성하는 옵션을 선택하면가 필요한 신뢰 정책을 역할에 AWS 연결합니다. Amazon Data Firehose가 기존 IAM 역할을 사용하도록 하거나 사용자가 직접 역할을 생성하는 경우, 다음 신뢰 정책을 해당 역할에 연결하여 Amazon Data Firehose가 해당 역할을 맡을 수 있도록 하세요. 정책을 편집하여 *account-id*를 AWS 계정 ID로 바꿉니다. 역할의 신뢰 관계를 수정하는 방법에 대한 자세한 내용은 역할 수정을 참조하세요.

Amazon Data Firehose는 Firehose 스트림의 데이터 처리 및 전송에 필요한 모든 권한에 대해 이 IAM 역할을 사용합니다. 다음과 같은 신뢰 정책이 해당 역할에 연결되어 있어야만 Amazon Data Firehose가 해당 역할을 사용할 수 있습니다.

Amazon MSK를 Firehose 스트림의 소스로 선택하는 경우, 지정된 Amazon MSK 클러스터에서 소스 데이터를 수집할 수 있는 권한을 Amazon Data Firehose에 부여하는 또 다른 IAM 역할을 지정해야 합니다. 다음과 같은 신뢰 정책이 해당 역할에 연결되어 있어야만 Amazon Data Firehose가 해당 역할을 사용할 수 있습니다.

JSON

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

```
      "Principal": {
        "Service": [
          "firehose.amazonaws.com"
        ]
      },
      "Effect": "Allow",
      "Action": "sts:AssumeRole"
    }
  ]
}
```

지정된 Amazon MSK 클러스터에서 소스 데이터를 수집할 수 있는 권한을 Amazon Data Firehose에 부여하는 이 역할은 다음과 같은 권한을 부여해야 합니다.

데이터 형식 변환을 AWS Glue 위해 Firehose에에 대한 액세스 권한 부여

Firehose 스트림이 데이터 형식 변환을 수행하는 경우 Amazon Data Firehose는 AWS Glue에 저장된 테이블 정의를 참조합니다. Amazon Data Firehose에 필요한 액세스 권한을 부여하려면 정책에 다음 문을 AWS Glue추가합니다. 테이블의 ARN을 찾는 방법에 대한 자세한 내용은 AWS Glue 리소스 ARNs.

```
{
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "glue:GetTable",
        "glue:GetTableVersion",
        "glue:GetTableVersions"
    ],
    "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/b",
        "arn:aws:glue:us-east-1:123456789012:table/b/easd"
    ]
},
{
      actions: ['glue:GetSchemaVersion'],
      grantee: options.role,
      resourceArns: ['*'],
}
```

스키마를 스키마 레지스트리에서 가져오는 데 권장되는 정책에는 리소스 제한이 없습니다. 자세한 내용은 AWS Glue 개발자 안내서의 역직렬화기에 대한 IAM 예제를 참조하세요.

Firehose에 Amazon S3 대상에 대한 액세스 권한 부여

Amazon S3 대상을 사용하는 경우 Amazon Data Firehose는 S3 버킷에 데이터를 전송하고 선택적으로 데이터 암호화에 소유한 AWS KMS 키를 사용할 수 있습니다. 오류 로깅이 활성화되면 Amazon Data Firehose는 CloudWatch 로그 그룹 및 스트림으로 데이터 전송 오류도 보냅니다. Firehose 스트림을 생성할 때 IAM 역할을 보유하고 있어야 합니다. Amazon Data Firehose는 이 IAM 역할을 사용하여 지정된 버킷, 키, CloudWatch 로그 그룹 및 스트림에 대한 액세스 권한을 얻습니다.

다음 액세스 정책을 이용해 Amazon Data Firehose이 S3 버킷 및 AWS KMS 키에 액세스할 수 있도록 합니다. S3 버킷을 소유하지 않은 경우 Amazon S3 작업 목록에 `s3:PutObjectAcl`을 추가합니다. 그렇게 하면 Amazon Data Firehose에서 전송한 객체에 대한 모든 액세스 권한이 버킷 소유자에게 부여됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
```

```
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:123456789012:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
```

```
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
}
```

위의 정책에는 Amazon Kinesis Data Streams에 대한 액세스를 허용하는 명령문도 있습니다. Kinesis Data Streams를 데이터 소스로 사용하지 않는 경우, 그 명령문을 제거할 수 있습니다. Amazon MSK를 소스로 사용하는 경우 해당 문을 다음과 같이 대체할 수 있습니다.

```
{
   "Sid":"",
   "Effect":"Allow",
   "Action":[
      "kafka:GetBootstrapBrokers",
      "kafka:DescribeCluster",
      "kafka:DescribeClusterV2",
      "kafka-cluster:Connect"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:cluster/
{{mskClusterName}}/{{clusterUUID}}"
},
{
   "Sid":"",
   "Effect":"Allow",
   "Action":[
      "kafka-cluster:DescribeTopic",
      "kafka-cluster:DescribeTopicDynamicConfiguration",
      "kafka-cluster:ReadData"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:topic/
{{mskClusterName}}/{{clusterUUID}}/{{mskTopicName}}"
},
{
   "Sid":"",
   "Effect":"Allow",
   "Action":[
      "kafka-cluster:DescribeGroup"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:group/
{{mskClusterName}}/{{clusterUUID}}/*"
```

```
}
```

다른 AWS 서비스가 리소스에 액세스하도록 허용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요 AWS .

다른 계정의 Amazon S3 대상에 대한 액세스 권한을 Amazon Data Firehose에 부여하는 방법은 the section called “Amazon S3 대상으로 교차 계정 전송” 섹션을 참조하세요.

Firehose에 Amazon S3 Tables에 대한 액세스 권한 부여

Firehose에는 테이블에 액세스하고 Amazon S3 AWS AWS Glue 테이블 버킷의 테이블에 데이터를 쓸 수 있는 특정 권한이 있는 IAM 역할이 필요합니다. Amazon S3 테이블 버킷의 테이블에 쓰려면 IAM 역할에 필요한 권한도 제공해야 합니다. Amazon S3 Tables 카탈로그에 필요한 권한은 사용하는 액세스 제어 모드에 따라 다릅니다.

- IAM 액세스 제어 - Firehose 전송 역할에는 Amazon S3 Tables 리소스에 대한 직접 IAM 권한이 필요합니다.
- Lake Formation 액세스 제어 - Firehose 전송 역할에는 테이블 리소스에 대한 액세스를 관리하기 위한 AWS AWS Lake Formation 권한이 필요합니다.는 데이터 카탈로그 리소스에 대한 세분화된 액세스 제어를 지원하는 자체 권한 모델을 AWS Lake Formation 사용합니다.

Firehose 스트림을 생성할 때 이 IAM 역할을 구성합니다. 액세스 제어 모드에 해당하는 탭을 선택합니다.

IAM 액세스 제어

IAM 액세스 제어 모드(없음 AWS Lake Formation)를 사용하는 경우 Firehose 전송 역할에는 Amazon S3 Tables 리소스 및 AWS Glue 데이터 카탈로그 객체에 대한 직접 IAM 권한이 필요합니다.

에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.

정책을 생성한 다음 정책 편집기에서 JSON을 선택합니다. 필요한 권한을 부여하는 다음 인라인 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3TablesAccessPermission",
            "Effect": "Allow",
```

```
            "Action": [
                "s3tables:GetTable",
                "s3tables:GetTableData",
                "s3tables:GetTableMetadataLocation",
                "s3tables:UpdateTableMetadataLocation"
            ],
            "Resource": [
                "arn:aws:s3tables:region:account-id:bucket/*",
                "arn:aws:s3tables:region:account-id:bucket/*/table/*"
            ]
        },
        {
            "Sid": "S3TableBucketAccessPermission",
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTableBucket"
            ],
            "Resource": "arn:aws:s3tables:region:account-id:bucket/*"
        },
        {
            "Sid": "GlueCatalogAccessPermission",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetTable",
                "glue:GetTables",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:region:account-id:catalog",
                "arn:aws:glue:region:account-id:catalog/s3tablescatalog",
                "arn:aws:glue:region:account-id:catalog/s3tablescatalog/*",
                "arn:aws:glue:region:account-id:database/*",
                "arn:aws:glue:region:account-id:table/*/*"
            ]
        },
        {
            "Sid": "S3DeliveryErrorBucketPermission",
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
```

```
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::error-delivery-bucket",
                "arn:aws:s3:::error-delivery-bucket/*"
            ]
        },
        {
            "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
            "Sid": "KMSPermissionForS3TablesEncryption",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:region:account-id:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn":
"arn:aws:s3tables:region:account-id:bucket/*/table/*"
                }
            }
        },
        {
            "Sid": "RequiredWhenUsingLambdaForDataTransformation",
            "Effect": "Allow",
            "Action": [
```

```
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": "arn:aws:lambda:region:account-id:function:function-
name:function-version"
        },
        {
            "Sid": "CloudWatchLogsPermission",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:region:account-id:log-group:log-group-name:log-
stream:log-stream-name"
        }
    ]
}
```

정책에는 Amazon Kinesis Data Streams에 대한 액세스, Lambda 함수 호출 및 AWS KMS 키에 대한 액세스를 허용하는 문이 있습니다. 이러한 리소스를 사용하지 않는 경우, 해당 문을 제거할 수 있습니다. 오류 로깅이 활성화되면 Amazon Data Firehose는 CloudWatch 로그 그룹 및 스트림으로 데이터 전송 오류도 보냅니다. 이 옵션을 사용하려면 로그 그룹 및 로그 스트림 이름을 구성해야 합니다. 로그 그룹 및 로그 스트림 이름은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.

인라인 정책에서 자리 표시자 값을 실제 리소스 이름, AWS 계정 번호 및 리전으로 바꿉니다.

정책을 생성한 후 IAM 콘솔을 https://console.aws.amazon.com/iam/에서 열고 AWS 서비스을(를) 신뢰할 수 있는 엔터티 유형으로 설정하여 IAM 역할을 생성합니다.

서비스 또는 사용 사례에서 Kinesis를 선택합니다. 사용 사례로 Kinesis Firehose를 선택합니다.

다음 페이지에서는 이전 단계에서 생성한 정책을 선택하여 이 역할에 연결합니다. 검토 페이지에서 Firehose 서비스가 해당 역할을 수임할 수 있도록 권한을 부여하는 신뢰 정책이 이미 이 역할에 연결되어 있음을 확인할 수 있습니다. 역할을 생성할 때 Amazon Data Firehose는 역할을 수임하여 AWS Glue 및 Amazon S3 Tables에서 필요한 작업을 수행할 수 있습니다. Firehose 서비스 보안 주체를 생성된 역할의 신뢰 정책에 추가합니다. 자세한 내용은 Firehose의 IAM 역할 수임 허용 단원을 참조하십시오.

Lake Formation 액세스 제어

AWS Lake Formation 액세스 제어 모드를 사용하는 경우 Firehose 전송 역할에는 IAM 정책 외에도 AWS Lake Formation 자격 증명 벤딩 권한이 필요합니다.

에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.

정책을 생성한 다음 정책 편집기에서 JSON을 선택합니다. 읽기/쓰기 권한, 데이터 카탈로그의 테이블 업데이트 권한 등의 Amazon S3 권한을 부여하는 다음 인라인 정책을 추가합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "S3TableAccessViaGlueFederation",
      "Effect": "Allow",
      "Action": [
        "glue:GetTable",
        "glue:GetDatabase",
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog/*",
        "arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog",
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/*",
        "arn:aws:glue:us-east-1:123456789012:table/*/*"
      ]
    },
    {
      "Sid": "S3DeliveryErrorBucketPermission",
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::<error delivery bucket>",
        "arn:aws:s3:::<error delivery bucket>/*"
      ]
    },
    {
      "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStream",
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:ListShards"
      ],
      "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/<stream-name>"
    },
    {
      "Sid":
"RequiredWhenDoingMetadataReadsANDDataAndMetadataWriteViaLakeformation",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RequiredWhenUsingKMSEncryptionForS3ErrorBucketDelivery",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:123456789012:key/<KMS-key-id>"
      ],
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.us-east-1.amazonaws.com"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::<error delivery
bucket>/prefix*"
        }
      }
    },
```

```
    {
      "Sid": "LoggingInCloudWatch",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name>:log-
stream:<log-stream-name>"
      ]
    },
    {
      "Sid": "RequiredWhenAttachingLambdaToFirehose",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:GetFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:us-east-1:123456789012:function:<function-
name>:<function-version>"
      ]
    }
  ]
}
```

정책에는 Amazon Kinesis Data Streams에 대한 액세스, Lambda 함수 호출 및 AWS KMS 키에 대한 액세스를 허용하는 문이 있습니다. 이러한 리소스를 사용하지 않는 경우, 해당 문을 제거할 수 있습니다. 오류 로깅이 활성화되면 Amazon Data Firehose는 CloudWatch 로그 그룹 및 스트림으로 데이터 전송 오류도 보냅니다. 이 옵션을 사용하려면 로그 그룹 및 로그 스트림 이름을 구성해야 합니다. 로그 그룹 및 로그 스트림 이름은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.

인라인 정책에서 `<error delivery bucket>`를 Amazon S3 버킷 이름으로 바꾸`aws-account-id`고 리전을 리소스의 유효한 AWS 계정 번호 및 리전으로 바꿉니다.

IAM 정책 외에도 Firehose 전송 역할에 필요한 권한을 부여해야 합니다 AWS Lake Formation. 자세한 내용은 테이블에 대한 권한 부여를 참조하세요.

정책을 생성한 후 IAM 콘솔을 https://console.aws.amazon.com/iam/에서 열고 AWS 서비스을(를) 신뢰할 수 있는 엔터티 유형으로 설정하여 IAM 역할을 생성합니다.

서비스 또는 사용 사례에서 Kinesis를 선택합니다. 사용 사례로 Kinesis Firehose를 선택합니다.

다음 페이지에서는 이전 단계에서 생성한 정책을 선택하여 이 역할에 연결합니다. 검토 페이지에서 Firehose 서비스가 해당 역할을 수임할 수 있도록 권한을 부여하는 신뢰 정책이 이미 이 역할에 연결되어 있음을 확인할 수 있습니다. 역할을 생성할 때 Amazon Data Firehose는 역할을 수임하여 AWS Glue 및 S3 버킷에서 필요한 작업을 수행할 수 있습니다. Firehose 서비스 보안 주체를 생성된 역할의 신뢰 정책에 추가합니다. 자세한 내용은 Firehose의 IAM 역할 수임 허용 단원을 참조하십시오.

Firehose에 Apache Iceberg 테이블 대상에 대한 액세스 권한 부여

AWS Glue를 사용하여 Firehose 스트림 및 Apache Iceberg 테이블을 생성하려면 먼저 IAM 역할이 있어야 합니다. 다음 단계를 따라 정책 및 IAM 역할을 생성합니다. Firehose는 이 IAM 역할을 수임하고 필요한 작업을 수행합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 정책을 생성한 다음 정책 편집기에서 JSON을 선택합니다.
3. 읽기/쓰기 권한, 데이터 카탈로그의 테이블 업데이트 권한 등의 Amazon S3 권한을 부여하는 다음 인라인 정책을 추가합니다.

   이 정책에는 Amazon Kinesis Data Streams에 대한 액세스, Lambda 함수 호출, KMS 키에 대한 액세스를 허용하는 문이 있습니다. 이러한 리소스를 사용하지 않는 경우, 해당 문을 제거할 수 있습니다.

   오류 로깅이 활성화되면 Firehose는 데이터 전송 오류도 CloudWatch 로그 그룹 및 스트림으로 보냅니다. 이를 위해서는 로그 그룹 및 로그 스트림 이름을 구성해야 합니다. 로그 그룹 및 로그 스트림 이름은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
4. 인라인 정책에서 *amzn-s3-demo-bucket*을 Amazon S3 버킷 이름으로 바꾸고, aws-account-id 및 리전을 리소스의 유효한 AWS 계정 번호 및 리전으로 바꿉니다.

   > **Note**
   >
   > 이 역할은 데이터 카탈로그의 모든 데이터베이스 및 테이블에 대한 권한을 부여합니다. 원하는 경우 특정 테이블 및 데이터베이스에 대해서만 권한을 부여할 수 있습니다.
5. 정책을 생성한 후 IAM 콘솔을 열고 AWS 서비스를 신뢰할 수 있는 엔터티 유형으로 설정하여 IAM 역할을 생성합니다.
6. 서비스 또는 사용 사례에서 Kinesis를 선택합니다. 사용 사례로 Kinesis Firehose를 선택합니다.

7. 다음 페이지에서는 이전 단계에서 생성한 정책을 선택하여 이 역할에 연결합니다. 검토 페이지에서 Firehose 서비스가 해당 역할을 수임할 수 있도록 권한을 부여하는 신뢰 정책이 이미 이 역할에 연결되어 있음을 확인할 수 있습니다. 역할을 생성하면 Amazon Data Firehose가 이 역할을 맡아 AWS Glue 및 S3 버킷에서 필요한 작업을 수행할 수 있습니다.

Firehose에 Amazon Redshift 대상에 대한 액세스 권한 부여

Amazon Redshift 대상을 사용하는 경우 Amazon Data Firehose에 액세스 권한을 부여할 때 다음 설명을 참조하세요.

주제

- IAM 역할 및 액세스 정책
- Amazon Redshift 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹에 대한 VPC 액세스

IAM 역할 및 액세스 정책

Amazon Redshift 대상을 사용 중일 때 Amazon Data Firehose는 중간 위치인 S3 버킷으로 데이터를 전송합니다. 선택적으로 데이터 암호화에 소유한 AWS KMS 키를 사용할 수 있습니다. 그러면 Amazon Data Firehose는 S3 버킷의 데이터를 Amazon Redshift 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹으로 로드합니다. 오류 로깅이 활성화되면 Amazon Data Firehose는 CloudWatch 로그 그룹 및 스트림으로 데이터 전송 오류도 보냅니다. Amazon Data Firehose는 지정된 Amazon Redshift 사용자 이름 및 암호를 사용하여 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹에 액세스하고, IAM 역할을 사용하여 지정된 버킷, 키, CloudWatch 로그 그룹 및 스트림에 액세스합니다. Firehose 스트림을 생성할 때 IAM 역할을 보유하고 있어야 합니다.

다음 액세스 정책을 이용해 Amazon Data Firehose이 S3 버킷 및 AWS KMS 키에 액세스할 수 있도록 합니다. S3 버킷을 소유하지 않은 경우, Amazon S3 작업 목록에 `s3:PutObjectAcl`을 추가하여, Amazon Data Firehose에서 전송한 객체에 대한 모든 액세스 권한을 버킷 소유자에게 부여합니다. 이 정책에는 Amazon Kinesis Data Streams에 대한 액세스를 허용하는 명령문도 있습니다. Kinesis Data Streams를 데이터 소스로 사용하지 않는 경우, 그 명령문을 제거할 수 있습니다.

JSON

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

```
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
        },
        {
           "Effect": "Allow",
           "Action": [
               "kinesis:DescribeStream",
               "kinesis:GetShardIterator",
               "kinesis:GetRecords",
               "kinesis:ListShards"
           ],
```

```
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
}
```

다른 AWS 서비스가 리소스에 액세스하도록 허용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요 AWS .

Amazon Redshift 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹에 대한 VPC 액세스

Amazon Redshift 프로비저닝 클러스터 또는 Amazon Redshift Serverless 작업 그룹이 Virtual Private Cloud(VPC) 내에 있으면 퍼블릭 IP 주소를 통해 공개적으로 액세스할 수 있어야 합니다. 또한 Amazon Data Firehose IP 주소를 차단 해제하여 Amazon Redshift 프로비저닝 클러스터 또는 Amazon Redshift Serverless 작업 그룹에 대한 액세스 권한을 Amazon Data Firehose에 부여하세요. Amazon Data Firehose는 현재 각 가용 리전마다 하나의 CIDR 블록을 사용합니다.

리전	CIDR 블록
미국 동부(오하이오)	13.58.135.96/27
미국 동부(버지니아 북부)	52.70.63.192/27
미국 서부(캘리포니아 북부)	13.57.135.192/27
미국 서부(오리건)	52.89.255.224/27
AWS GovCloud(미국 동부)	18.253.138.96/27
AWS GovCloud(미국 서부)	52.61.204.160/27
캐나다(중부)	35.183.92.128/27
캐나다 서부(캘거리)	40.176.98.192/27
아시아 태평양(홍콩)	18.162.221.32/27
아시아 태평양(뭄바이)	13.232.67.32/27
아시아 태평양(하이데라바드)	18.60.192.128/27
아시아 태평양(서울)	13.209.1.64/27
아시아 태평양(싱가포르)	13.228.64.192/27
아시아 태평양(시드니)	13.210.67.224/27
아시아 태평양(자카르타)	108.136.221.64/27
아시아 태평양(도쿄)	13.113.196.224/27
아시아 태평양(오사카)	13.208.177.192/27
아시아 태평양(태국)	43.208.112.96/27

리전	CIDR 블록
아시아 태평양(타이베이)	43.212.53.160/27
중국(베이징)	52.81.151.32/27
중국(닝샤)	161.189.23.64/27
유럽(취리히)	16.62.183.32/27
유럽(프랑크푸르트)	35.158.127.160/27
유럽(아일랜드)	52.19.239.192/27
유럽(런던)	18.130.1.96/27
유럽(파리)	35.180.1.96/27
유럽(스톡홀름)	13.53.63.224/27
유럽(스페인)	18.100.71.96/27
Middle East (Bahrain)	15.185.91.0/27
멕시코(중부)	78.12.207.32/27
남아메리카(상파울루)	18.228.1.128/27
유럽(밀라노)	15.161.135.128/27
아프리카(케이프타운)	13.244.121.224/27
중동(UAE)	3.28.159.32/27
이스라엘(텔아비브)	51.16.102.0/27
아시아 태평양(멜버른)	16.50.161.128/27
아시아 태평양(말레이시아)	43.216.58.0/27

IP 주소 차단 해제에 대한 자세한 내용은 Amazon Redshift 시작 가이드의 클러스터에 대한 액세스 권한 부여 단계를 참조하세요.

Firehose에 퍼블릭 OpenSearch Service 대상에 대한 액세스 권한 부여

OpenSearch Service 대상을 사용 중인 경우 Amazon Data Firehose는 OpenSearch Service 클러스터로 데이터를 전송하는 동시에 실패한 문서나 모든 문서를 S3 버킷에 백업합니다. 오류 로깅이 활성화되면 Amazon Data Firehose는 CloudWatch 로그 그룹 및 스트림으로 데이터 전송 오류도 보냅니다. Amazon Data Firehose는 IAM 역할을 사용하여 지정된 OpenSearch Service 도메인, S3 버킷, AWS KMS 키, CloudWatch 로그 그룹 및 스트림에 액세스합니다. Firehose 스트림을 생성할 때 IAM 역할을 보유하고 있어야 합니다.

다음 액세스 정책을 사용하여 Amazon Data Firehose가 S3 버킷, OpenSearch Service 도메인 및 AWS KMS 키에 액세스할 수 있도록 합니다. S3 버킷을 소유하지 않은 경우, Amazon S3 작업 목록에 `s3:PutObjectAcl`을 추가합니다. 이렇게 하면 Amazon Data Firehose에서 전송한 객체에 대한 모든 액세스 권한을 버킷 소유자에게 부여합니다. 이 정책에는 Amazon Kinesis Data Streams에 대한 액세스를 허용하는 명령문도 있습니다. Kinesis Data Streams를 데이터 소스로 사용하지 않는 경우, 그 명령문을 제거할 수 있습니다.

다른 AWS 서비스가 리소스에 액세스하도록 허용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요 AWS .

다른 계정의 OpenSearch Service 클러스터에 대한 액세스 권한을 Amazon Data Firehose에 부여하는 방법은 the section called “OpenSearch Service 대상으로 교차 계정 전송” 섹션을 참조하세요.

Firehose에 VPC의 OpenSearch Service 대상에 대한 액세스 권한 부여

OpenSearch Service 도메인이 VPC에 있는 경우 이전 섹션에서 설명한 권한을 Amazon Data Firehose에 부여해야 합니다. 또한 OpenSearch Service 도메인의 VPC에 액세스할 수 있도록 하려면 Amazon Data Firehose에 다음 권한을 부여해야 합니다.

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`

- `ec2:DeleteNetworkInterface`

> **Important**
>
> Firehose 스트림을 생성한 후에는 해당 권한을 취소하지 마세요. 이러한 권한을 취소하면 서비스가 ENIs를 쿼리하거나 업데이트하려고 할 때마다 Firehose 스트림의 성능이 저하되거나 OpenSearch 서비스 도메인으로의 데이터 전송이 중지됩니다.

> **Important**
>
> 프라이빗 VPC의 대상으로 데이터를 전송하기 위해 서브넷을 지정할 때는 선택한 서브넷에 사용 가능한 IP 주소가 충분히 있는지 확인합니다. 지정된 서브넷에 사용 가능한 IP 주소가 없는 경우 Firehose는 프라이빗 VPC에서 데이터 전송을 위한 ENI를 생성하거나 추가할 수 없으며 전송이 저하되거나 실패합니다.

Firehose 스트림을 생성하거나 업데이트하는 경우 OpenSearch Service 도메인으로 데이터를 전송할 때 Firehose가 사용할 보안 그룹을 지정합니다. OpenSearch Service 도메인에서 사용하는 것과 동일한 보안 그룹을 사용하거나 다른 보안 그룹을 사용할 수 있습니다. 다른 보안 그룹을 지정하는 경우 해당 보안 그룹이 OpenSearch Service 도메인의 보안 그룹에 대한 아웃바운드 HTTPS 트래픽을 허용하는지 확인합니다. 또한 OpenSearch Service 도메인의 보안 그룹이 Firehose 스트림을 구성할 때 지정한 보안 그룹의 HTTPS 트래픽을 허용해야 합니다. Firehose 스트림과 OpenSearch Service 도메인 모두에 동일한 보안 그룹을 사용하는 경우 보안 그룹 인바운드 규칙이 HTTPS 트래픽을 허용하는지 확인합니다. 보안 그룹 규칙에 관한 자세한 정보는 Amazon VPC 설명서의 보안 그룹 규칙을 참조하세요.

Firehose에 퍼블릭 OpenSearch Serverless 대상에 대한 액세스 권한 부여

OpenSearch Serverless 대상을 사용 중인 경우 Amazon Data Firehose는 OpenSearch Serverless 컬렉션으로 데이터를 전송하는 동시에 실패한 문서나 모든 문서를 S3 버킷에 백업합니다. 오류 로깅이 활성화되면 Amazon Data Firehose는 CloudWatch 로그 그룹 및 스트림으로 데이터 전송 오류도 보냅니다. Amazon Data Firehose는 IAM 역할을 사용하여 지정된 OpenSearch Serverless 컬렉션, S3 버킷, AWS KMS 키, CloudWatch 로그 그룹 및 스트림에 액세스합니다. Firehose 스트림을 생성할 때 IAM 역할을 보유하고 있어야 합니다.

다음 액세스 정책을 사용하여 Amazon Data Firehose가 S3 버킷, OpenSearch Serverless 도메인 및 AWS KMS 키에 액세스할 수 있도록 합니다. S3 버킷을 소유하지 않은 경우, Amazon S3 작업 목록에

`s3:PutObjectAcl`을 추가합니다. 이렇게 하면 Amazon Data Firehose에서 전송한 객체에 대한 모든 액세스 권한을 버킷 소유자에게 부여합니다. 이 정책에는 Amazon Kinesis Data Streams에 대한 액세스를 허용하는 명령문도 있습니다. Kinesis Data Streams를 데이터 소스로 사용하지 않는 경우, 그 명령문을 제거할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
```

```
            }
         },
        {
           "Effect": "Allow",
           "Action": [
               "kinesis:DescribeStream",
               "kinesis:GetShardIterator",
               "kinesis:GetRecords",
               "kinesis:ListShards"
           ],
           "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-name"
        },
        {
           "Effect": "Allow",
           "Action": [
               "logs:PutLogEvents"
           ],
           "Resource": [
               "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
           ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "lambda:InvokeFunction",
               "lambda:GetFunctionConfiguration"
           ],
           "Resource": [
               "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
           ]
        },
        {
         "Effect": "Allow",
         "Action": "aoss:APIAccessAll",
         "Resource": "arn:aws:aoss:us-east-1:123456789012:collection/collection-
id"
      }
    ]
}
```

위의 정책과 함께, 다음과 같은 최소한의 권한이 데이터 액세스 정책에 할당되도록 Amazon Data Firehose를 구성해야 합니다.

```
[
   {
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/target-collection/target-index"
            ],
            "Permission":[
               "aoss:WriteDocument",
               "aoss:UpdateIndex",
               "aoss:CreateIndex"
            ]
         }
      ],
      "Principal":[
         "arn:aws:sts::123456789012:assumed-role/firehose-delivery-role-name/*"
      ]
   }
]
```

다른 AWS 서비스가 리소스에 액세스하도록 허용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요 AWS .

Firehose에 VPC의 OpenSearch Serverless 대상에 대한 액세스 권한 부여

OpenSearch Serverless 컬렉션이 VPC에 있는 경우 Amazon Data Firehose에 이전 섹션에서 설명한 권한을 부여해야 합니다. 또한 OpenSearch Serverless 컬렉션의 VPC에 액세스할 수 있도록 하려면 Amazon Data Firehose에 다음 권한을 합니다.

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`

- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

> **Important**
>
> Firehose 스트림을 생성한 후에는 해당 권한을 취소하지 마세요. 이러한 권한을 취소하면 서비스가 ENIs를 쿼리하거나 업데이트하려고 할 때마다 Firehose 스트림의 성능이 저하되거나 OpenSearch 서비스 도메인으로의 데이터 전송이 중지됩니다.

> **Important**
>
> 프라이빗 VPC의 대상으로 데이터를 전송하기 위해 서브넷을 지정할 때는 선택한 서브넷에 사용 가능한 IP 주소가 충분히 있는지 확인합니다. 지정된 서브넷에 사용 가능한 IP 주소가 없는 경우 Firehose는 프라이빗 VPC에서 데이터 전송을 위한 ENI를 생성하거나 추가할 수 없으며 전송이 저하되거나 실패합니다.

Firehose 스트림을 생성하거나 업데이트하는 경우 OpenSearch Serverless 컬렉션으로 데이터를 전송할 때 Firehose가 사용할 보안 그룹을 지정합니다. OpenSearch Serverless 컬렉션에서 사용하는 것과 동일한 보안 그룹을 사용하거나 다른 보안 그룹을 사용할 수 있습니다. 다른 보안 그룹을 지정하는 경우 해당 보안 그룹이 OpenSearch Serverless 컬렉션의 보안 그룹에 대한 아웃바운드 HTTPS 트래픽을 허용하는지 확인합니다. 또한 OpenSearch Serverless 컬렉션의 보안 그룹이 Firehose 스트림을 구성할 때 지정한 보안 그룹의 HTTPS 트래픽을 허용해야 합니다. Firehose 스트림과 OpenSearch Serverless 컬렉션 모두에 동일한 보안 그룹을 사용하는 경우 보안 그룹 인바운드 규칙이 HTTPS 트래픽을 허용하는지 확인합니다. 보안 그룹 규칙에 관한 자세한 정보는 Amazon VPC 설명서의 보안 그룹 규칙을 참조하세요.

Firehose에 Splunk 대상에 대한 액세스 권한 부여

Splunk 대상을 사용 중일 때 Amazon Data Firehose는 Splunk HEC(HTTP Event Collector) 엔드포인트로 데이터를 전송합니다. 또한 지정한 Amazon S3 버킷에 해당 데이터를 백업하며, 선택적으로 Amazon S3 서버 측 암호화에 소유한 AWS KMS 키를 사용할 수 있습니다. 오류 로깅이 활성화되면 Firehose는 CloudWatch 로그 스트림으로 데이터 전송 오류를 보냅니다. 데이터 변환 AWS Lambda 에를 사용할 수도 있습니다.

AWS 로드 밸런서를 사용하는 경우 Classic Load Balancer 또는 Application Load Balancer인지 확인합니다. 또한 Classic Load Balancer의 경우 쿠키 만료를 비활성화한 기간 기반 스티키 세션을 사용 설정하고, Application Load Balancer의 경우 만료 기간을 최댓값(7일)으로 설정해야 합니다. 이 작업을 수행하는 방법에 대한 자세한 정보는 Classic Load Balancer 또는 Application Load Balancer의 기간 기반 세션 고정을 참조하세요.

Firehose 스트림을 생성할 때는 IAM 역할이 있어야 합니다. Firehose는 이 IAM 역할을 사용하여 지정된 버킷, 키, CloudWatch 로그 그룹 및 스트림에 대한 액세스 권한을 얻습니다.

다음 액세스 정책을 이용해 Amazon Data Firehose가 S3 버킷에 액세스할 수 있도록 합니다. S3 버킷을 소유하지 않은 경우, Amazon S3 작업 목록에 `s3:PutObjectAcl`을 추가하여, Amazon Data Firehose에서 전송한 객체에 대한 모든 액세스 권한을 버킷 소유자에게 부여합니다. 또한이 정책은 오류 로깅을 위해 CloudWatch에 대한 액세스 권한을 Amazon Data Firehose에 부여하고 데이터 변환을 위해 AWS Lambda 에 대한 액세스 권한을 부여합니다. 정책에는 Amazon Kinesis Data Streams에 대한 액세스를 허용하는 명령문도 있습니다. Kinesis Data Streams를 데이터 소스로 사용하지 않는 경우, 그 명령문을 제거할 수 있습니다. Amazon Data Firehose는 Splunk에 액세스하기 위해 IAM을 사용하지 않습니다. Splunk 액세스를 위해 HEC 토큰을 사용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
```

```
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:123456789012:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
```

```
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
}
```

다른 AWS 서비스가 리소스에 액세스하도록 허용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요 AWS .

VPC에서 Splunk에 액세스

Splunk 플랫폼이 VPC 내에 있으면 퍼블릭 IP 주소를 통해 공개적으로 액세스할 수 있어야 합니다. 또한 Amazon Data Firehose IP 주소를 차단 해제하여 Splunk 플랫폼에 대한 액세스 권한을 Amazon Data Firehose에 부여하세요. Amazon Data Firehose는 현재 다음 CIDR 블록을 사용합니다.

리전	CIDR 블록
미국 동부(오하이오)	18.216.68.160/27, 18.216.170.64/27, 18.216.170.96/27 \
미국 동부(버지니아 북부)	34.238.188.128/26, 34.238.188.192/26, 34.238.195.0/26
미국 서부(캘리포니아 북부)	13.57.180.0/26
미국 서부(오리건)	34.216.24.32/27, 34.216.24.192/27, 34.216.24.224/27
AWS GovCloud(미국 동부)	18.253.138.192/26
AWS GovCloud(미국 서부)	52.61.204.192/26
아시아 태평양(홍콩)	18.162.221.64/26

리전	CIDR 블록
아시아 태평양(타이베이)	43.212.53.192/26
아시아 태평양(뭄바이)	13.232.67.64/26
아시아 태평양(서울)	13.209.71.0/26
아시아 태평양(싱가포르)	13.229.187.128/26
아시아 태평양(시드니)	13.211.12.0/26
아시아 태평양(태국)	43.208.112.128/26
아시아 태평양(도쿄)	13.230.21.0/27, 13.230.21.32/27
캐나다(중부)	35.183.92.64/26
캐나다 서부(캘거리)	40.176.98.128/26
유럽(프랑크푸르트)	18.194.95.192/27, 18.194.95.224/27, 18.195.48.0/27
유럽(아일랜드)	34.241.197.32/27, 34.241.197.64/27, 34.241.197.96/27
유럽(런던)	18.130.91.0/26
유럽(파리)	35.180.112.0/26
유럽(스페인)	18.100.194.0/26
유럽(스톡홀름)	13.53.191.0/26
Middle East (Bahrain)	15.185.91.64/26
멕시코(중부)	78.12.207.64/26
남아메리카(상파울루)	18.228.1.192/26
유럽(밀라노)	15.161.135.192/26

리전	CIDR 블록
아프리카(케이프타운)	13.244.165.128/26
아시아 태평양(오사카)	13.208.217.0/26
중국(베이징)	52.81.151.64/26
중국(닝샤)	161.189.23.128/26
아시아 태평양(자카르타)	108.136.221.128/26
중동(UAE)	3.28.159.64/26
이스라엘(텔아비브)	51.16.102.64/26
유럽(취리히)	16.62.183.64/26
아시아 태평양(하이데라바드)	18.60.192.192/26
아시아 태평양(멜버른)	16.50.161.192/26
아시아 태평양(말레이시아)	43.216.44.192/26
아시아 태평양(뉴질랜드)	3.102.119.128/26

Amazon Data Firehose를 사용하여 Splunk로 VPC 흐름 로그 수집

VPC 흐름 로그 구독을 생성하고, Firehose에 게시하고, 지원되는 대상으로 VPC 흐름 로그를 전송하는 방법에 대해 자세히 알아보려면 Ingest VPC flow logs into Splunk using Amazon Data Firehose(Amazon Data Firehose를 사용하여 Splunk로 VPC 흐름 로그 수집)를 참조하세요.

Snowflake 또는 HTTP 엔드포인트 액세스

대상이 HTTP 엔드포인트 또는 Snowflake 퍼블릭 클러스터인 경우, Amazon Data Firehose에만 해당하는 특정한 AWS IP 주소 범위의 서브세트는 없습니다.

퍼블릭 Snowflake 클러스터의 허용 목록에 Firehose를 추가하거나, 퍼블릭 HTTP 또는 HTTPS 엔드포인트에 Firehose를 추가하려면 수신 규칙에 현재 AWS IP 주소 범위를 모두 추가하세요.

> **Note**
>
> 알림이 항상 연결된 주제와 동일한 AWS 리전의 IP 주소에서 소싱되는 것은 아닙니다. 모든 리전의 AWS IP 주소 범위를 포함해야 합니다.

Firehose에 Snowflake 대상에 대한 액세스 권한 부여

Snowflake를 대상으로 사용하는 경우 Firehose는 Snowflake 계정 URL을 사용하여 Snowflake 계정으로 데이터를 전송합니다. 또한 사용자가 지정한 Amazon Simple Storage Service 버킷에 오류 데이터를 백업하고 Amazon S3 서버 측 암호화에 소유한 AWS Key Management Service 키를 선택적으로 사용할 수 있습니다. 오류 로깅이 활성화되면 Firehose는 CloudWatch 로그 스트림으로 데이터 전송 오류를 보냅니다.

Firehose 스트림을 생성하려면 먼저 IAM 역할이 있어야 합니다. Firehose는 이 IAM 역할을 사용하여 지정된 버킷, 키, CloudWatch 로그 그룹 및 스트림에 대한 액세스 권한을 얻습니다. 다음 액세스 정책을 이용해 Firehose가 S3 버킷에 액세스하도록 합니다. S3 버킷을 소유하지 않은 경우, Amazon Simple Storage Service 작업 목록에 `s3:PutObjectAcl`을 추가하여 Firehose가 전송한 객체에 대한 모든 액세스 권한을 버킷 소유자에게 부여하세요. 이 정책은 오류 로깅을 위해 CloudWatch에 대한 액세스 권한을 Firehose에 부여합니다. 정책에는 Amazon Kinesis Data Streams에 대한 액세스를 허용하는 명령문도 있습니다. Kinesis Data Streams를 데이터 소스로 사용하지 않는 경우, 그 명령문을 제거할 수 있습니다. Firehose는 IAM을 사용하여 Snowflake에 액세스하지 않습니다. 프라이빗 클러스터의 경우 Snowflake에 액세스하기 위해 Snowflake 계정 URL 및 PrivateLink Vpce ID를 사용합니다.

JSON

```
{
"Version":"2012-10-17",
    "Statement":
    [
        {
"Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
```

```
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
"Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:123456789012:key/key-id"
            ],
            "Condition": {
"StringEquals": {
"kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
"kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket/prefix*"
                }
            }
        },
        {
"Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
"Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
```

```
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:*"
            ]
        }
    ]
}
```

다른 AWS 서비스가 AWS 리소스에 액세스하도록 허용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

VPC에서 Snowflake에 액세스

Snowflake 클러스터에 프라이빗 링크를 사용하도록 설정된 경우, Firehose는 프라이빗 링크를 생성할 때 다음 VPC 엔드포인트 중 하나를 사용하여 퍼블릭 인터넷을 거치지 않고 프라이빗 클러스터에 데이터를 전송합니다. 이를 위해 AWS 리전 클러스터에 `AwsVpceIds` 대해 다음의 수신을 허용하는 Snowflake 네트워크 규칙을 생성합니다. 자세한 내용은 Snowflake 사용 설명서의 네트워크 규칙 생성을 참조하세요.

클러스터가 있는 리전에 따라 사용할 VPC 엔드포인트 ID

AWS 리전	**VPCE IDs**
미국 동부(오하이오)	vpce-0d96cafcd96a50aeb vpce-0cec34343d48f537b
미국 동부(버지니아 북부)	vpce-0b4d7e8478e141ba8 vpce-0b75cd681fb507352 vpce-01c03e63820ec00d8 vpce-0c2cfc51dc2882422 vpce-06ca862f019e4e056 vpce-020cda0cfa63f8d1c vpce-0b80504a1a783cd70 vpce-0289b9ff0b5259a96

AWS 리전	**VPCE IDs**
	vpce-0d7add8628bd69a12
	vpce-02bfb5966cc59b2af
	vpce-09e707674af878bf2
	vpce-049b52e96cc1a2165
	vpce-0bb6c7b7a8a86cdbb
	vpce-03b22d599f51e80f3
	vpce-01d60dc60fc106fe1
	vpce-0186d20a4b24ecbef
	vpce-0533906401a36e416
	vpce-05111fb13d396710e
	vpce-0694613f4fbd6f514
	vpce-09b21cb25fe4cc4f4
	vpce-06029c3550e4d2399
	vpce-00961862a21b033da
	vpce-01620b9ae33273587
	vpce-078cf4ec226880ac9
	vpce-0d711bf076ce56381
	vpce-066b7e13cbfca6f6e
	vpce-0674541252d9ccc26
	vpce-03540b88dedb4b000
	vpce-0b1828e79ad394b95

AWS 리전	**VPCE IDs**
	vpce-0dc0e6f001fb1a60d vpce-0d8f82e71a244098a vpce-00e374d9e3f1af5ce vpce-0c1e3d6631ddb442f
미국 서부(오리건)	vpce-0f60f72da4cd1e4e7 vpce-0c60d21eb8b1669fd vpce-01c4e3e29afdafbef vpce-0cc6bf2a88da139de vpce-0797e08e169e50662 vpce-033cbe480381b5c0e vpce-00debbdd8f9eb10a5 vpce-08ec2f386c809e889 vpce-0856d14310857b545
유럽(프랑크푸르트)	vpce-068dbb7d71c9460fb vpce-0a7a7f095942d4ec9
유럽(아일랜드)	vpce-06857e59c005a6276 vpce-04390f4f8778b75f2 vpce-011fd2b1f0aa172fd
아시아 태평양(도쿄)	vpce-06369e5258144e68a vpce-0f2363cdb8926fbe8

AWS 리전	**VPCE IDs**
아시아 태평양(싱가포르)	vpce-049cd46cce7a12d52 vpce-0e8965a1a4bdb8941
아시아 태평양(서울)	vpce-0aa444d9001e1faa1 vpce-04a49d4dcfd02b884
아시아 태평양(시드니)	vpce-048a60a182c52be63 vpce-03c19949787fd1859
아시아 태평양(뭄바이)	vpce-0d68cb822f6f0db68 vpce-0517d32692ffcbde2
유럽(런던)	vpce-0fd1874a0ba3b9374 vpce-08091b1a85e206029
남아메리카(상파울루)	vpce-065169b8144e4f12e vpce-0493699f0e5762d63
캐나다(중부)	vpce-07e6ed81689d5271f vpce-0f532397305413 94c
유럽(파리)	vpce-09419680077e6488a vpce-0ea81ba2c08140c14
아시아 태평양(오사카)	vpce-0a9f003e6a7e38c05 vpce-02886510b897b1c5a
유럽(스톡홀름)	vpce-0d96410833219025a vpce-060a32f9a75ba969f

AWS 리전	VPCE IDs
아시아 태평양(자카르타)	vpce-00add4b9a25e5c649 vpce-004ae2de34338a856

Firehose에 HTTP 엔드포인트 대상에 대한 액세스 권한 부여

Amazon Data Firehose를 사용하여 모든 HTTP 엔드포인트 대상에 데이터를 보낼 수 있습니다. 또한 Amazon Data Firehose는 해당 데이터를 사용자가 지정한 Amazon Simple Storage Service 버킷에 백업하며, 원하는 경우 Amazon S3 서버 측 암호화를 위해 사용자가 소유한 AWS KMS 키를 사용할 수 있습니다. 오류 로깅이 활성화되면 Amazon Data Firehose는 CloudWatch 로그 스트림으로 데이터 전송 오류를 보냅니다. 데이터 변환 AWS Lambda 에를 사용할 수도 있습니다.

Firehose 스트림을 생성할 때 IAM 역할을 보유하고 있어야 합니다. Amazon Data Firehose는 이 IAM 역할을 사용하여 지정된 버킷, 키, CloudWatch 로그 그룹 및 스트림에 대한 액세스 권한을 얻습니다.

다음 액세스 정책을 이용해 Amazon Data Firehose이 데이터 백업용으로 지정된 S3 버킷에 액세스할 수 있도록 합니다. S3 버킷을 소유하지 않은 경우, Amazon S3 작업 목록에 `s3:PutObjectAcl`을 추가하여, Amazon Data Firehose에서 전송한 객체에 대한 모든 액세스 권한을 버킷 소유자에게 부여합니다. 또한이 정책은 오류 로깅을 위해 CloudWatch에 대한 액세스 권한을 Amazon Data Firehose에 부여하고 데이터 변환을 위해 AWS Lambda 에 대한 액세스 권한을 부여합니다. 정책에는 Amazon Kinesis Data Streams에 대한 액세스를 허용하는 명령문도 있습니다. Kinesis Data Streams를 데이터 소스로 사용하지 않는 경우, 그 명령문을 제거할 수 있습니다.

> **Important**
>
> Amazon Data Firehose는 지원되는 타사 서비스 제공업체(Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Splunk 또는 Sumo Logic 등)가 소유한 HTTP 엔드포인트 대상에 액세스하기 위해 IAM을 사용하지 않습니다. 지원되는 타사 서비스 공급자가 소유한 지정된 HTTP 엔드포인트 대상에 액세스하려면 해당 서비스 공급자에게 문의하여 Amazon Data Firehose에서 해당 서비스로 데이터를 전송하는 데 필요한 API 키 또는 액세스 키를 받으세요.

다른 AWS 서비스가 리소스에 액세스하도록 허용하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요 AWS .

> **Important**
>
> 현재 Amazon Data Firehose는 VPC의 HTTP 엔드포인트에 대한 데이터 전송을 지원하지 않습니다.

Amazon MSK에서 계정 간 전송

Firehose 계정(예: 계정 B)에서 Firehose 스트림을 생성하고 소스가 다른 AWS 계정(계정 A)의 MSK 클러스터인 경우 다음 구성이 있어야 합니다.

계정 A:

1. Amazon MSK 콘솔에서 프로비저닝된 클러스터를 선택한 다음 속성을 선택하세요.
2. 네트워크 설정에서 편집을 선택하여 다중 VPC 연결을 활성화하세요.
3. 보안 설정에서 클러스터 정책 편집을 선택합니다.
   a. 클러스터에 아직 정책이 구성되어 있지 않은 경우 Firehose 서비스 보안 주체 포함 및 Firehose 교차 계정 S3 전송 활성화를 선택합니다. AWS Management Console 는 적절한 권한이 있는 정책을 자동으로 생성합니다.
   b. 클러스터에 이미 구성된 정책이 있는 경우 기존 정책에 다음 권한을 추가하세요.

```
{
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
      ],
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/DO-NOT-TOUCH-
mskaas-provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20"  // ARN
 of the cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
      ],
      "Resource": "arn:aws:kafka:us-east-1:arn:topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*"//topic of the
 cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": "kafka-cluster:DescribeGroup",
      "Resource": "arn:aws:kafka:us-east-1:arn:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of
 the cluster
    },
 }
```

4. AWS 보안 주체에 계정 B의 보안 주체 ID를 입력하세요.
5. 주제에 Firehose 스트림이 데이터를 수집할 때 사용할 Apache Kafka 주제를 지정하세요. Firehose 스트림이 생성되고 나면 이 주제를 업데이트할 수 없습니다.
6. 변경 사항 저장을 선택합니다

계정 B:

1. Firehose 콘솔에서 계정 B를 사용하여 Firehose 스트림 생성을 선택하세요.
2. 소스에서 Amazon Managed Streaming for Apache Kafka를 선택하세요.
3. 소스 설정에서 Amazon Managed Streaming for Apache Kafka 클러스터에 계정 A의 Amazon MSK ARN을 입력하세요.
4. 주제에 Firehose 스트림이 데이터를 수집할 때 사용할 Apache Kafka 주제를 지정하세요. Firehose 스트림이 생성되고 나면 이 주제를 업데이트할 수 없습니다.
5. 전송 스트림 이름에 Firehose 스트림의 이름을 입력하세요.

Firehose 스트림을 생성할 때 계정 B에는 구성된 주제의 교차 계정 Amazon MSK 클러스터에 대한 '읽기' 액세스 권한을 Firehose 스트림에 부여하는 IAM 역할(사용 시 기본적으로 생성됨 AWS Management Console)이 있어야 합니다.

다음은 AWS Management Console에 의해 구성되는 내용입니다.

```
{
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
        ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:cluster/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the
 cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/mskaas_test_topic" //
topic of the cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeGroup"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the
 cluster
    },
 }
```

그 다음으로, 레코드 변환 및 레코드 형식 변환을 구성하는 옵션 단계를 완료할 수 있습니다. 자세한 내용은 (선택 사항) 레코드 변환 및 형식 변환 구성 섹션을 참조하세요.

Amazon S3 대상으로 교차 계정 전송

AWS CLI 또는 Amazon Data Firehose APIs를 사용하여 다른 계정의 Amazon S3 대상이 있는 한 AWS 계정에서 Firehose 스트림을 생성할 수 있습니다. 다음 절차는 계정 A가 소유한 Firehose 스트림에서 계정 B가 소유한 Amazon S3 버킷으로 데이터를 전송하도록 구성하는 예를 보여줍니다.

1. Grant Firehose Access to an Amazon S3 Destination(Firehose에 Amazon S3 대상에 대한 액세스 권한 부여)에 설명된 단계를 사용하여 계정 A에서 IAM 역할을 생성합니다.

   > **Note**
   >
   > 이 경우 액세스 정책에 지정된 Amazon S3 버킷은 계정 B가 소유합니다. 액세스 정책에서 Amazon S3 작업 목록에 `s3:PutObjectAcl`을 추가하여, Amazon Data Firehose에서 전송한 객체에 대한 모든 액세스 권한을 계정 B에 부여합니다. 교차 계정 전송 시 이 권한이 필요합니다. Amazon Data Firehose는 요청 시 “x-amz-acl” 헤더를 “bucket-owner-full-control”로 설정합니다.

2. 이전에 생성한 IAM 역할에서 액세스하도록 허용하려면 계정 B 하에서 S3 버킷 정책을 생성합니다. 다음 코드는 버킷 정책의 예입니다. 자세한 내용은 버킷 정책 및 사용자 정책 사용을 참조하세요.

   JSON

```
{

    "Version":"2012-10-17",
    "Id": "PolicyID",
    "Statement": [
        {
            "Sid": "StmtID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/iam-role-name"
            },
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
```

```
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

3. 1단계에서 생성한 IAM 역할을 이용하여 계정 A에서 Firehose 스트림을 생성합니다.

OpenSearch Service 대상으로 교차 계정 전송

AWS CLI 또는 Amazon Data Firehose APIs를 사용하여 다른 계정의 OpenSearch Service 대상이 있는 한 AWS 계정에서 Firehose 스트림을 생성할 수 있습니다. 다음 절차는 계정 A에서 Firehose 스트림을 만들고 계정 B가 소유한 OpenSearch Service 대상에 데이터를 전송하도록 구성하는 방법에 대한 예제를 보여줍니다.

1. the section called “Firehose에 퍼블릭 OpenSearch Service 대상에 대한 액세스 권한 부여”에 설명된 단계를 사용하여 계정 A에서 IAM 역할을 생성합니다.
2. 이전 단계에서 만든 IAM 역할의 액세스를 허용하려면 계정 B에서 OpenSearch Service 정책을 만듭니다. 다음은 JSON 예시입니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/firehose_delivery_role "
      },
      "Action": "es:ESHttpGet",
```

```
      "Resource": [
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/_all/
_settings",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_cluster/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
roletest*/_mapping/roletest",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes/*/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
roletest*/_stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/"
      ]
    }
  ]
}
```

3. 1단계에서 생성한 IAM 역할을 이용하여 계정 A에서 Firehose 스트림을 생성합니다. Firehose 스트림을 생성할 때 AWS CLI 또는 Amazon Data Firehose APIs를 사용하고 `DomainARN` OpenSearch Service 대신 `ClusterEndpoint` 필드를 지정합니다.

> **Note**
>
> 다른 AWS 계정의 OpenSearch Service 대상이 있는 한 계정에서 Firehose 스트림을 생성하려면 AWS CLI 또는 Amazon Data Firehose APIs를 사용해야 합니다. AWS Management Console 를 사용하여 이러한 종류의 교차 계정 구성을 생성할 수 없습니다.

태그를 사용하여 액세스 제어

IAM 정책의 선택적 `Condition` 요소(또는 `Condition` 블록)를 사용하여 태그 키와 값을 기반으로 Amazon Data Firehose 작업에 대한 액세스를 상세 조정할 수 있습니다. 다음 섹션에서는 다양한 Amazon Data Firehose 작업에 대해 이를 수행하는 방법을 설명합니다. `Condition` 요소의 사용 방법과 그 요소에 사용할 수 있는 작업을 알아보려면 IAM JSON 정책 요소: 조건을 참조하세요.

CreateDeliveryStream

`CreateDeliveryStream` 작업에는 `aws:RequestTag` 조건 키를 사용하세요. 다음 예에서 `MyKey`와 `MyValue`는 태그에 대한 키와 그 값을 나타냅니다. 자세한 내용은 태그의 기본 사항 이해 섹션을 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "firehose:CreateDeliveryStream",
            "firehose:TagDeliveryStream"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/MyKey": "MyValue"
            }
        }
    }]
}
```

TagDeliveryStream

`TagDeliveryStream` 작업에는 `aws:TagKeys` 조건 키를 사용하세요. 다음 예에서 `MyKey`은 태그 키의 예입니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:TagDeliveryStream",
            "Resource": "*",
            "Condition": {
```

```
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                }
            }
        }
    ]
}
```

UntagDeliveryStream

`UntagDeliveryStream` 작업에는 `aws:TagKeys` 조건 키를 사용하세요. 다음 예에서 `MyKey`은 태그 키의 예입니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:UntagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                }
            }
        }
    ]
}
```

ListDeliveryStreams

`ListDeliveryStreams`에는 태그 기반 액세스를 사용할 수 없습니다.

기타 작업

`CreateDeliveryStream`, `TagDeliveryStream`, `UntagDeliveryStream`, `ListDeliveryStreams` 이외의 모든 Firehose 작업에는 `aws:RequestTag` 조건 키를 사용합니다. 다음 예에서 `MyKey`와 `MyValue`는 태그에 대한 키와 그 값을 나타냅니다.

ListDeliveryStreams에서는 firehose:ResourceTag 조건 키를 사용하여 해당 Firehose 스트림의 태그를 기반으로 액세스를 제어합니다.

다음 예에서 MyKey와 MyValue는 태그에 대한 키와 그 값을 나타냅니다. 이 정책은 이름이 MyKey이고 값이 MyValue인 태그를 가진 Data Firehose 스트림에만 적용됩니다. 리소스 태그를 기반으로 액세스를 제어하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 태그를 사용하여 AWS 리소스에 대한 액세스 제어를 참조하세요.

Amazon Data Firehose AWS Secrets Manager 에서 로 인증

Amazon Data Firehose는와 통합되어 보안 암호 AWS Secrets Manager 에 대한 보안 액세스를 제공하고 자격 증명 교체를 자동화합니다. 이 통합을 통해 Firehose는 런타임 시 Secrets Manager에서 보안 암호를 검색하여 이전에 언급한 스트리밍 대상에 연결하고 데이터 스트림을 전송할 수 있습니다. 이렇게 하면 AWS Management Console 또는 API 파라미터에서 스트림 생성 워크플로 중에 보안 암호가 일반 텍스트로 표시되지 않습니다. 따라서 보안 암호를 안전하게 관리할 수 있고, 암호 교체 관리를 위해 Lambda 함수를 사용자 지정하여 설정하는 일과 같은 복잡한 자격 증명 관리 작업에서 벗어날 수 있습니다.

자세한 내용은 AWS Secrets Manager 사용 설명서를 참조하십시오.

주제

- 보안 암호 이해
- 보안 암호 생성
- 보안 암호 사용
- 보안 암호 교체

보안 암호 이해

보안 암호는 암호, 사용자 이름 및 암호와 같은 자격 증명 집합, OAuth 토큰 또는 Secrets Manager에 암호화된 형식으로 저장하는 기타 비밀 정보일 수 있습니다.

다음 섹션과 같이 각 대상에 대하여 보안 암호 키-값 페어를 올바른 JSON 형식으로 지정해야 합니다. 보안 암호가 대상에 맞는 올바른 JSON 형식이 아닌 경우 Amazon Data Firehose가 대상에 연결하지 못합니다.

MySQL 및 PostgreSQL과 같은 데이터베이스의 보안 암호 형식

```
{
    "username":  "<username>",
    "password":  "<password>"
}
```

Amazon Redshift 프로비저닝 클러스터 및 Amazon Redshift Serverless 작업 그룹의 보안 암호 형식

```
{
    "username":  "<username>",
    "password":  "<password>"
}
```

Splunk의 보안 암호 형식

```
{
    "hec_token":  "<hec token>"
}
```

Snowflake의 보안 암호 형식

```
{
    "user":  "<snowflake-username>",
    "private_key":  "<snowflake-private-key>", // without the beginning and ending
 private key, remove all spaces and newlines
    "key_passphrase":  "<snowflake-private-key-passphrase>" // optional
}
```

HTTP 엔드포인트, Coralogix, Datadog, Dynatrace, Elastic, Honeycomb, LogicMonitor, Logz.io, MongoDB Cloud, New Relic의 보안 암호 형식

```
{
    "api_key":  "<apikey>"
}
```

보안 암호 생성

보안 암호를 생성하려면 AWS Secrets Manager 사용 설명서의 AWS Secrets Manager 보안 암호 생성 단계를 따르세요.

보안 암호 사용

AWS Secrets Manager 를 사용하여 자격 증명 또는 키를 저장하여 Amazon Redshift, HTTP 엔드포인트, Snowflake, Splunk, Coralogix, Datadog, Dynatrace, Elastic, Honeycomb, LogicMonitor, Logz.io, MongoDB Cloud, New Relic과 같은 스트리밍 대상에 연결하는 것이 좋습니다.

Firehose 스트림 생성 시 이러한 대상에 대해 AWS 관리 콘솔을 통해 Secrets Manager로 인증을 구성할 수 있습니다. 자세한 내용은 대상 설정 구성 섹션을 참조하세요. 또는 CreateDeliveryStream 및 UpdateDestination API 작업을 사용하여 Secrets Manager로 인증을 구성할 수도 있습니다.

Firehose는 암호화를 사용하여 보안 암호를 캐시하고 대상에 연결할 때마다 이를 사용합니다. 그리고 최신 자격 증명이 사용되도록 10분마다 캐시를 새로 고칩니다.

스트림의 수명 주기 동안에는 언제든지 Secrets Manager에서 보안 암호 검색 기능을 끄도록 선택할 수 있습니다. Secrets Manager를 사용하지 않고 보안 암호를 검색하고 싶다면 사용자 이름/암호 또는 API 키를 사용할 수 있습니다.

> **Note**
>
> Firehose에서 이 기능에 대한 추가 비용은 없지만 Secrets Manager의 액세스 및 유지 관리에 대한 비용은 청구됩니다. 자세한 정보는 AWS Secrets Manager 요금 페이지를 참조하세요.

보안 암호 검색을 위해 Firehose에 액세스 권한 부여

Firehose가 보안 암호를 검색하려면 보안 암호에 액세스하는 데 필요한 권한과 보안 암호를 암호화하는 키를 Firehose에 제공해야 AWS Secrets Manager합니다.

AWS Secrets Manager 를 사용하여 보안 암호를 저장하고 검색할 때 보안 암호가 저장되는 위치와 암호화되는 방법에 따라 몇 가지 구성 옵션이 있습니다.

- 보안 암호가 IAM 역할과 동일한 AWS 계정에 저장되고 기본 AWS 관리형 키(`aws/secretsmanager`)로 암호화된 경우 Firehose가 수임하는 IAM 역할에는 보안 암호에 대한 `secretsmanager:GetSecretValue` 권한만 있으면 됩니다.

```
// secret role policy
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "Secret ARN"
        }
    ]
}
```

IAM 정책에 대한 자세한 정보는 AWS Secrets Manager에 대한 권한 정책 예를 참조하세요.

- 보안 암호가 역할과 동일한 계정에 저장되어 있지만 고객 관리형 키(CMK)로 암호화된 경우, 역할에는 `secretsmanager:GetSecretValue` 및 `kms:Decrypt` 권한이 모두 필요합니다. 또한 CMK 정책은 IAM 역할이 `kms:Decrypt`를 수행할 수 있도록 허용해야 합니다.
- 보안 암호가 역할과 다른 AWS 계정에 저장되고 기본 AWS 관리형 키로 암호화된 경우 보안 암호가 관리형 키로 암호화될 때 Secrets Manager가 교차 계정 액세스를 허용하지 않으므로이 구성이 불가능합니다 AWS .
- 보안 암호가 다른 계정에 저장되어 있고 CMK로 암호화된 경우 IAM 역할에는 보안 암호에 대한 `secretsmanager:GetSecretValue` 권한과 CMK에 대한 `kms:Decrypt` 권한이 필요합니다. 보안 암호의 리소스 정책 및 다른 계정의 CMK 정책에서도 IAM 역할에 필요한 권한을 허용해야 합니다. 자세한 정보는 교차 계정 액세스를 참조하세요.

보안 암호 교체

교체는 보안 암호를 주기적으로 업데이트하는 것입니다. 지정한 일정에 따라 보안 암호를 자동으로 교체 AWS Secrets Manager 하도록를 구성할 수 있습니다. 그러면 장기 보안 암호를 단기 보안 암호로 교체할 수 있습니다. 이를 통해 침해 위험을 줄일 수 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 AWS Secrets Manager 보안 암호 교체를 참조하세요.

Amazon Data Firehose 콘솔을 통해 IAM 역할 관리

Amazon Data Firehose는 대상에 실시간 스트리밍 데이터를 전송하는 완전관리형 서비스입니다. 또한 전송 전에 데이터를 변환하거나 형식을 변환하도록 Firehose를 구성할 수도 있습니다. 이러한 기능을 사용하려면, 먼저 Firehose 스트림을 생성하거나 편집할 때 Firehose에 권한을 부여하는 IAM 역할을 제공해야 합니다. Firehose는 Firehose 스트림에 필요한 모든 권한에 대해 이 IAM 역할을 사용합니다.

예를 들어 Amazon S3에 데이터를 전송하는 Firehose 스트림을 생성하고이 Firehose 스트림에 AWS Lambda 기능이 활성화된 변환 소스 레코드가 있는 시나리오를 가정해 보겠습니다. 이 경우 다음과 같이 S3 버킷에 액세스하고 Lambda 함수를 호출할 수 있는 권한을 Firehose에 부여하는 IAM 역할을 제공해야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "lambdaProcessing",
        "Effect": "Allow",
        "Action": ["lambda:InvokeFunction", "lambda:GetFunctionConfiguration"],
        "Resource": "arn:aws:lambda:us-east-1:123456789012:function:<lambda
 function name>:<lambda function version>"
    }, {
        "Sid": "s3Permissions",
        "Effect": "Allow",
        "Action": ["s3:AbortMultipartUpload", "s3:GetBucketLocation",
 "s3:GetObject", "s3:ListBucket", "s3:ListBucketMultipartUploads",
 "s3:PutObject"],
        "Resource": ["arn:aws:s3:::<bucket name>", "arn:aws:s3:::<bucket name>/
*"]
    }]
}
```

Firehose 콘솔을 사용하면 이러한 역할을 제공하는 방법을 선택할 수 있습니다. 다음 옵션 중 하나를 선택할 수 있습니다.

- 기존 IAM 역할 선택
- 콘솔에서 새 IAM 역할 생성

기존 IAM 역할 선택

기존 IAM 역할 중에서 선택할 수 있습니다. 이 옵션을 선택하는 경우, 선택한 IAM 역할에 소스 및 대상에 필요한 적절한 신뢰 정책 및 권한이 있는지 확인합니다. 자세한 내용은 Amazon Data Firehose를 통한 액세스 제어 섹션을 참조하세요.

콘솔에서 새 IAM 역할 생성

또는 Firehose 콘솔을 통해 사용자를 대신하여 새 역할을 생성할 수도 있습니다.

Firehose가 사용자를 대신하여 IAM 역할을 생성하면 이 역할에는 Firehose 스트림 구성에 따라 필요한 권한을 부여하는 모든 권한 및 신뢰 정책이 자동으로 포함됩니다.

예를 들어 AWS Lambda를 사용하여 소스 레코드 변환 기능을 사용하도록 설정하지 않은 경우 콘솔은 권한 정책에 다음 문을 생성합니다.

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
   "Action": [
     "lambda:InvokeFunction",
     "lambda:GetFunctionConfiguration"
   ],
   "Resource": "arn:aws:lambda:us-east-1123456789012:function:
%FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%"
}
```

> **Note**
>
> %FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%가 포함된 정책 문은 리소스에 대한 권한을 부여하지 않으므로 무시해도 안전합니다.

콘솔은 Firehose 스트림 워크플로를 생성 및 편집하고 신뢰 정책 또한 생성하여 IAM 역할에 연결합니다. 이 신뢰 정책은 Firehose가 IAM 역할을 맡을 수 있도록 허용합니다. 다음은 신뢰 정책의 예입니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "firehoseAssume",
        "Effect": "Allow",
        "Principal": {
            "Service": "firehose.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

Important

- 여러 Firehose 스트림에 동일한 콘솔 관리형 IAM 역할을 사용해서는 안 됩니다. 그렇지 않으면 IAM 역할의 허용 범위가 지나치게 넓어지거나 오류가 발생할 수 있습니다.
- 콘솔 관리형 IAM 역할의 권한 정책 내에서 여러 정책 문을 사용하려면 자체 IAM 역할을 생성하고 정책 문을 새 역할에 연결된 권한 정책에 복사하면 됩니다. Firehose 스트림에 역할을 연결하려면 서비스 액세스 에서 기존 IAM 역할 선택 옵션을 선택합니다.
- 콘솔은 ARN에 service-role 문자열이 포함된 모든 IAM 역할을 관리합니다. 기존 IAM 역할 옵션을 선택할 때는 콘솔이 해당 역할을 변경하지 않도록 ARN에 service-role 문자열이 포함되지 않은 IAM 역할을 선택해야 합니다.

콘솔에서 IAM 역할을 생성하는 방법

1. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
2. Firehose 스트림 생성을 선택합니다.
3. 소스 및 대상을 선택합니다. 자세한 내용은 자습서: 콘솔에서 Firehose 스트림 생성 섹션을 참조하세요.
4. 대상 설정을 선택합니다. 자세한 내용은 대상 설정 구성 섹션을 참조하세요.
5. 고급 설정의 서비스 액세스에서 IAM 역할 생성 또는 업데이트를 선택합니다.

   Note

   이는 기본 옵션입니다. 기존 역할을 사용하려면 기존 IAM 역할 선택 옵션을 선택합니다. Firehose 콘솔은 자신의 역할을 스스로 변경하지 않습니다.

6. Firehose 스트림 생성을 선택합니다.

콘솔에서 IAM 역할 편집

Firehose 스트림을 편집할 때 Firehose는 구성 및 권한 변경 사항을 반영하여 그에 따라 해당 권한 정책을 업데이트합니다.

예를 들어 Firehose 스트림을 편집하고 최신 버전의 Lambda 함수를 `exampleLambdaFunction`으로 사용하여 AWS Lambda를 사용하여 소스 레코드 변환 기능을 사용하도록 설정하면 권한 정책에 다음과 같은 정책 문이 표시됩니다.

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
  "Action": [
    "lambda:InvokeFunction",
    "lambda:GetFunctionConfiguration"
  ],
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:exampleLambdaFunction:
$LATEST"
}
```

Important

콘솔 관리형 IAM 역할은 자율적으로 동작하도록 설계되었습니다. 콘솔 외부에서 권한 정책이나 신뢰 정책을 수정하지 않는 것이 좋습니다.

콘솔에서 IAM 역할을 편집하는 방법

1. https://console.aws.amazon.com/firehose/에서 Firehose 콘솔을 엽니다.
2. Firehose 스트림을 선택하고 업데이트하려는 Firehose 스트림의 이름을 선택합니다.
3. 구성 탭의 서버 액세스 섹션에서 편집을 선택합니다.
4. IAM 역할 옵션을 업데이트합니다.

   Note

   기본적으로 콘솔은 ARN에 service-role 패턴이 포함된 IAM 역할을 항상 업데이트합니다. 기존 IAM 역할 옵션을 선택할 때는 콘솔이 해당 역할을 변경하지 않도록 ARN에 service-role 문자열이 포함되지 않은 IAM 역할을 선택해야 합니다.
5. 변경 사항 저장을 선택합니다.

Amazon Data Firehose의 규정 준수 이해

타사 감사자는 여러 규정 준수 프로그램의 일환으로 Amazon Data Firehose의 보안 및 AWS 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램 범위의 AWS 서비스 목록은 AWS 규정 준수 프로그램 제공 범위 내 서비스를 참조하세요. 일반 정보는 AWS 규정 준수 프로그램을 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 AWS 아티팩트에서 보고서 다운로드를 참조하세요.

Data Firehose 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률 및 규정에 따라 결정됩니다. Data Firehose를 사용할 때 HIPAA, PCI 또는 FedRAMP와 같은 표준을 준수해야 하는 경우는 도움이 되는 리소스를 AWS 제공합니다.

- 보안 및 규정 준수 빠른 시작 가이드 -이 배포 가이드에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수 중심 기준 환경을 배포하기 위한 단계를 제공합니다 AWS.
- HIPAA 보안 및 규정 준수를 위한 설계 백서 -이 백서에서는 기업이 AWS 를 사용하여 HIPAA 준수 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 업계 및 위치에 적용될 수 있습니다.
- AWS Config -이 AWS 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub CSPM -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

Amazon Data Firehose의 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS 글로벌 인프라 외에도 Data Firehose는 데이터 복원성과 백업 요구 사항을 지원하는 몇 가지 기능을 제공합니다.

재해 복구

Amazon Data Firehose는 서버리스 모드에서 실행되며, 자동 마이그레이션을 수행하여 호스트 저하, 가용 영역 가용성 및 기타 인프라 관련 문제를 처리합니다. 이 경우 Amazon Data Firehose는 Firehose 스트림이 데이터 손실 없이 마이그레이션되도록 보장합니다.

Amazon Data Firehose의 인프라 보안 이해

관리형 서비스인 Amazon Data Firehose는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 Firehose에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

> **Note**
>
> 발신 HTTPS 요청의 경우 Amazon Data Firehose는 HTTP 라이브러리를 사용하여 대상 측에서 지원되는 가장 높은 TLS 프로토콜 버전을 자동 선택합니다.

AWS PrivateLink에서 Amazon Data Firehose 사용

인터페이스 VPC 엔드포인트(AWS PrivateLink)를 사용하여 인터넷 게이트웨이 또는 NAT 게이트웨이 없이 VPC에서 Amazon Data Firehose에 액세스할 수 있습니다. 인터페이스 VPC 엔드포인트에는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 Direct Connect 연결이 필요하지 않습니다. 인터페이스 VPC 엔드포인트는 Amazon VPC의 프라이빗 IPs와 함께 탄력적 네트워크 인터페이스를 사용하여 AWS 서비스 간의 프라이빗 통신을 지원하는 AWS 기술인 AWS PrivateLink로 구동됩니다. 자세한 내용은 Amazon Virtual Private Cloud(VPC)를 참조하세요.

Firehose에 인터페이스 VPC 엔드포인트(AWS PrivateLink) 사용

시작하려면 Amazon VPC 리소스의 Amazon Data Firehose 트래픽이 인터페이스 VPC 엔드포인트를 통해 흐름을 시작하도록 인터페이스 VPC 엔드포인트를 생성합니다. 엔드포인트를 생성하면 Amazon Data Firehose에 대한 액세스를 제어하기 위한 엔드포인트 정책을 연결할 수 있습니다. 정책을 사용하여 VPC 엔드포인트에서 Amazon Data Firehose로의 액세스를 제어하는 방법에 대한 자세한 내용은 VPC 엔드포인트로 서비스 액세스 제어를 참조하세요.

다음 예제에서는 VPC에서 AWS Lambda 함수를 설정하고 함수가 Amazon Data Firehose 서비스와 안전하게 통신할 수 있도록 VPC 엔드포인트를 생성하는 방법을 보여줍니다. 이 예에서는 Lambda 함수가 현재 리전에 Firehose 스트림을 나열할 수 있지만 Firehose 스트림을 설명할 수는 없는 정책을 사용합니다.

VPC 엔드포인트 생성

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/vpc/ Amazon VPC 콘솔을 엽니다.
2. VPC 대시보드에서 엔드포인트를 선택합니다.
3. 엔드포인트 생성을 선택합니다.
4. 서비스 이름 목록에서 `com.amazonaws.`*`your_region`*`.kinesis-firehose`를 선택합니다.
5. 엔드포인트를 생성할 VPC와 서브넷(한 개 이상)을 선택합니다.
6. 보안 그룹을 한 개 이상 선택하여 엔드포인트와 연결합니다.
7. 정책에서 사용자 지정을 선택하고 다음 정책을 붙여 넣습니다.

```
{
    "Statement": [
        {
            "Sid": "Allow-only-specific-PrivateAPIs",
            "Principal": "*",
            "Action": [
                "firehose:ListDeliveryStreams"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Allow-only-specific-PrivateAPIs",
            "Principal": "*",
            "Action": [
                "firehose:DescribeDeliveryStream"
            ],
            "Effect": "Deny",
            "Resource": [
                "*"
            ]
        }
```

```
    ]
}
```

8. 엔드포인트 생성을 선택합니다.

Lambda 함수에 사용할 IAM 역할 만들기

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 왼쪽 탐색 창에서 Roles(역할)을 선택한 다음 Create role(역할 생성)을 선택합니다.
3. 신뢰할 수 있는 유형의 엔터티 선택에서 기본 선택인 AWS 서비스를 그대로 둡니다.
4. Choose the service that will use this role(이 역할을 사용할 서비스 선택) 아래에서 Lambda를 선택합니다.
5. Next: Permissions(다음: 권한)를 선택합니다.
6. 정책 목록에서 `AWS LambdaVPCAccessExecutionRole` 정책과 `AmazonDataFirehoseReadOnlyAccess` 정책을 검색하여 추가합니다.

> **Important**
> 이 정책은 예제이며, 프로덕션 환경의 경우 더 엄격한 정책이 필요할 수도 있습니다.

7. 다음: 태그를 선택합니다. 이 연습에서는 태그를 추가할 필요가 없습니다. 다음: 검토를 선택합니다.
8. 역할 이름을 입력한 다음 Create role(역할 생성)을 선택합니다.

VPC 내 Lambda 함수 생성

1. https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. 함수 생성을 선택합니다.
3. 새로 작성을 선택합니다.
4. 함수 이름을 입력한 후 런타임을 `Python 3.9` 이상으로 설정합니다.
5. 권한에서 실행 역할 선택 또는 생성을 확장합니다.
6. 실행 역할 목록에서 기존 역할 사용을 선택합니다.
7. 기존 역할 목록에서 앞서 만든 역할을 선택합니다.
8. 함수 생성을 선택합니다.
9. 함수 코드 아래에 다음 코드를 붙여 넣습니다.

```
   import json
   import boto3
   import os
   from botocore.exceptions import ClientError

   def lambda_handler(event, context):
       REGION = os.environ['AWS_REGION']
       client = boto3.client(
           'firehose',
           REGION

       )
       print("Calling list_delivery_streams with ListDeliveryStreams allowed
policy.")
       delivery_stream_request = client.list_delivery_streams()
       print("Successfully returned list_delivery_streams request %s." % (
           delivery_stream_request
       ))
       describe_access_denied = False
       try:
           print("Calling describe_delivery_stream with DescribeDeliveryStream
denied policy.")
           delivery_stream_info =
client.describe_delivery_stream(DeliveryStreamName='test-describe-denied')
       except ClientError as e:
           error_code = e.response['Error']['Code']
           print ("Caught %s." % (error_code))
           if error_code == 'AccessDeniedException':
               describe_access_denied = True

       if not describe_access_denied:
           raise
       else:
           print("Access denied test succeeded.")
```

10. 기본 설정에서 제한 시간을 1분으로 설정합니다.
11. 네트워크에서 앞서 엔드포인트를 생성한 VPC를 선택한 후, 엔드포인트를 생성할 때 엔드포인트와 연결한 서브넷 및 보안 그룹을 선택합니다.
12. 페이지 상단에서 Save(저장)를 선택합니다.
13. 테스트를 선택합니다.
14. 이벤트 이름을 입력한 다음 Create(생성)를 선택합니다.

15. 테스트를 다시 선택합니다. 그러면 함수가 실행됩니다. 실행 결과가 나타나면 세부 정보를 확장하고 로그 출력과 함수 코드를 비교합니다. 성공적인 결과에는 리전의 Firehose 스트림의 목록과 다음 출력이 표시됩니다.

```
Calling describe_delivery_stream.

AccessDeniedException

Access denied test succeeded.
```

지원됨 AWS 리전

인터페이스 VPC 엔드포인트는 현재 다음 리전 내에서 지원됩니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(태국)
- 아시아 태평양(도쿄)
- 아시아 태평양(홍콩)
- 캐나다(중부)
- 캐나다 서부(캘거리)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 멕시코(중부)

- 남아메리카(상파울루)
- AWS GovCloud(미국 동부)
- AWS GovCloud(미국 서부)
- 유럽(스페인)
- 중동(UAE)
- 아시아 태평양(자카르타)
- 아시아 태평양(오사카)
- 이스라엘(텔아비브)
- 아시아 태평양(말레이시아)

Amazon Data Firehose의 보안 모범 사례 구현

Amazon Data Firehose는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주세요.

최소 권한 액세스 구현

권한을 부여할 때 누가 어떤 Amazon Data Firehose 리소스에 대해 어떤 권한을 갖는지 결정합니다. 해당 리소스에서 허용할 작업을 사용 설정합니다. 따라서 작업을 수행하는 데 필요한 권한만 부여해야 합니다. 최소 권한 액세스를 구현하는 것이 오류 또는 악의적인 의도로 인해 발생할 수 있는 보안 위험과 영향을 최소화할 수 있는 근본적인 방법입니다.

IAM 역할 사용

생산자 및 클라이언트 애플리케이션은 Firehose 스트림에 액세스하려면 유효한 자격 증명이 있어야 하며, Firehose 스트림은 대상에 액세스하려면 유효한 자격 증명이 있어야 합니다. 클라이언트 애플리케이션 또는 Amazon S3 버킷에 직접 AWS 자격 증명을 저장해서는 안 됩니다. 이러한 보안 인증은 자동으로 교체되지 않으며 손상된 경우 비즈니스에 큰 영향을 줄 수 있는 장기 보안 인증입니다.

대신 IAM 역할을 사용하여 생산자 및 클라이언트 애플리케이션이 Firehose 스트림에 액세스하기 위한 임시 보안 인증을 관리해야 합니다. 역할을 사용하면 장기 자격 증명(예: 사용자 이름과 암호 또는 액세스 키)을 사용하여 다른 리소스에 액세스할 필요가 없습니다.

자세한 설명은 IAM 사용자 가이드에서 다음 주제를 참조하세요:

- IAM 역할

- 역할에 대한 일반적인 시나리오: 사용자, 애플리케이션 및 서비스

종속 리소스에서 서버 측 암호화 구현

Amazon Data Firehose에서 유휴 데이터와 전송 중인 데이터를 암호화할 수 있습니다. 자세한 내용은 Amazon Data Firehose의 데이터 보호 단원을 참조하십시오.

CloudTrail을 사용하여 API 직접 호출 모니터링

Amazon Data Firehose는 Amazon Data Firehose에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다.

CloudTrail에서 수집한 정보를 사용하여 Amazon Data Firehose에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

자세한 내용은 the section called “Firehose API 호출 로깅” 섹션을 참조하세요.

Amazon Data Firehose 모니터링

다음 기능을 사용하여 Amazon Data Firehose를 모니터링할 수 있습니다.

주제

CloudWatch 경보 모범 사례 실행

다음 지표가 버퍼링 제한(최대 15분)을 초과할 때 CloudWatch 알람을 추가합니다.

- `DeliveryToS3.DataFreshness`
- `DeliveryToIceberg.DataFreshness`
- `DeliveryToSplunk.DataFreshness`
- `DeliveryToAmazonOpenSearchService.DataFreshness`
- `DeliveryToAmazonOpenSearchServerless.DataFreshness`
- `DeliveryToHttpEndpoint.DataFreshness`

또한 다음 지표 수식 표현식을 기반으로 경보를 생성합니다.

- `IncomingBytes (Sum per 5 Minutes) / 300`은 `BytesPerSecondLimit`의 백분율에 가까워집니다.
- `IncomingRecords (Sum per 5 Minutes) / 300`은 `RecordsPerSecondLimit`의 백분율에 가까워집니다.
- `IncomingPutRequests (Sum per 5 Minutes) / 300`은 `PutRequestsPerSecondLimit`의 백분율에 가까워집니다.

경보를 권장하는 또 다른 지표는 ThrottledRecords입니다.

경보가 ALARM 상태가 될 경우 문제 해결 방법에 대한 자세한 내용은 오류 해결 섹션을 참조하세요.

CloudWatch 지표를 사용하여 Amazon Data Firehose 모니터링

> **Important**
>
> 오류를 빠르게 식별하려면 대상에 속하는 모든 CloudWatch 지표에 대한 알람을 활성화해야 합니다.

Amazon Data Firehose는 Amazon CloudWatch 지표와 통합되므로 Firehose 스트림에 대한 CloudWatch 지표를 수집, 확인, 분석할 수 있습니다. 예를 들어 IncomingBytes 및 IncomingRecords 지표를 모니터링하여 데이터 생산자로부터 Amazon Data Firehose로 수집된 데이터를 추적할 수 있습니다.

Amazon Data Firehose는 1분마다 CloudWatch 지표를 수집하여 게시합니다. 단, 몇 초 동안만이라도 수신 데이터 장애가 발생하면 1분 지표에 완전히 캡처또는 표시되지 않을 수 있습니다. 그 이유는 CloudWatch 지표가 Amazon Data Firehose에서 1분 간격으로 집계되기 때문입니다.

Firehose 스트림에 대해 수집된 지표는 무료로 제공됩니다. Kinesis 에이전트 지표에 대한 정보는 Kinesis 에이전트 상태 모니터링을 참조하세요.

주제

- 동적 파티셔닝용 CloudWatch 지표
- 데이터 전송에 대한 CloudWatch 지표
- 데이터 수집 지표
- API 수준 CloudWatch 지표
- 데이터 변환 CloudWatch 지표
- CloudWatch Logs 압축 해제 지표
- 형식 변환 CloudWatch 지표
- 서버 측 암호화(SSE) CloudWatch 지표
- Amazon Data Firehose의 측정기준
- Amazon Data Firehose 사용 지표

동적 파티셔닝용 CloudWatch 지표

동적 파티셔닝이 활성화된 경우 AWS/Firehose 네임스페이스에는 다음 지표가 포함됩니다.

지표	설명
ActivePartitionsLimit	Firehose 스트림이 오류 버킷으로 데이터를 전송하기 전에 처리하는 최대 활성 파티션 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
PartitionCount	처리 중인 파티션 수, 즉 활성 파티션의 수. 이 수는 1부터 파티션 카운트 한도인 500(기본값)까지 다양합니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
PartitionCountExceeded	이 지표는 파티션 카운트 한도를 초과하는지 여부를 나타냅니다. 한도 위반 여부에 따라 1 또는 0이 출력됩니다.
JQProcessing.Duration	JQ Lambda 함수에서 JQ 표현식을 실행하는 데 걸린 시간의 양을 반환합니다. 단위: 밀리초
PerPartitionThroughput	파티션별로 처리되는 처리량을 표시합니다. 이 지표를 사용하여 파티션별 처리량을 모니터링할 수 있습니다. 단위: StandardUnit.BytesSecond
DeliveryToS3.ObjectCount	S3 버킷으로 전송되는 객체의 수를 나타냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

데이터 전송에 대한 CloudWatch 지표

AWS/Firehose 네임스페이스에는 다음과 같은 서비스 수준 지표가 포함되어 있습니다. BackupToS3.Success, DeliveryToS3.Success, DeliveryToSplunk.Success, DeliveryToAmazonOpenSearchService.Success 또는 DeliveryToRedshift.Success의 평균이 약간 떨어진다고 해서 데이터 손실이 발생했다는 의미는 아닙니다. Amazon Data Firehose는 전송 오류를 재시도하고 레코드가 구성된 대상 또는 백업 S3 버킷으로 전송이 이루어질 때까지 작업을 진행하지 않습니다.

주제

OpenSearch Service로 전송

지표	설명
DeliveryToAmazonOpenSearchService.Bytes	지정된 기간 동안 OpenSearch Service로 인덱싱된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToAmazonOpenSearchService.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 OpenSearch Service에 전송되었습니다. 통계: Minimum, Maximum, Average, Sum, Samples

지표	설명
	단위: 초
DeliveryToAmazonOpenSearchService.Records	지정된 기간 동안 OpenSearch Service로 인덱싱된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToAmazonOpenSearchService.Success	성공적으로 인덱싱된 레코드의 합계입니다.
DeliveryToS3.Bytes	지정한 시간 동안 Amazon S3로 전송된 바이트 수. Amazon Data Firehose는 모든 문서에 대한 백업을 활성화한 경우에만 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToS3.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 S3 버킷으로 전송되었습니다. Amazon Data Firehose는 모든 문서에 대한 백업을 활성화한 경우에만 이 지표를 내보냅니다. 단위: 초
DeliveryToS3.Records	지정한 시간 동안 Amazon S3로 전송된 레코드 수. Amazon Data Firehose는 모든 문서에 대한 백업을 활성화한 경우에만 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

지표	설명
DeliveryToS3.Success	성공적인 Amazon S3 풋 명령의 합계입니다. Amazon Data Firehose는 백업이 실패한 문서에만 사용되는지 아니면 모든 문서에 사용되는지에 관계없이 항상 이 지표를 내보냅니다.
DeliveryToAmazonOpenSearchService.AuthFailure	인증 및 권한 부여 오류. OS/ES 클러스터 정책 및 역할 권한을 확인합니다. 0은 문제 없음을 나타내며 1은 인증 실패를 나타냅니다.
DeliveryToAmazonOpenSearchService.DeliveryRejected	전송 거부 오류. OS/ES 클러스터 정책 및 역할 권한을 확인합니다. 0은 문제 없음을 나타내며 1은 전송 실패를 나타냅니다.

OpenSearch Serverless로 전송

지표	설명
DeliveryToAmazonOpenSearchServerless.Bytes	지정된 기간 동안 OpenSearch Serverless로 인덱싱된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToAmazonOpenSearchServerless.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 OpenSearch Serverless에 전송되었습니다. 단위: 초
DeliveryToAmazonOpenSearchServerless.Records	지정된 기간 동안 OpenSearch Serverless로 인덱싱된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples

지표	설명
	단위: 개
DeliveryToAmazonOpenSearchServerless.Success	성공적으로 인덱싱된 레코드의 합계입니다.
DeliveryToS3.Bytes	지정한 시간 동안 Amazon S3로 전송된 바이트 수. Amazon Data Firehose는 모든 문서에 대한 백업을 활성화한 경우에만 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToS3.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 S3 버킷으로 전송되었습니다. Amazon Data Firehose는 모든 문서에 대한 백업을 활성화한 경우에만 이 지표를 내보냅니다. 단위: 초
DeliveryToS3.Records	지정한 시간 동안 Amazon S3로 전송된 레코드 수. Amazon Data Firehose는 모든 문서에 대한 백업을 활성화한 경우에만 이 지표를 내보냅니다. 단위: 개
DeliveryToS3.Success	성공적인 Amazon S3 풋 명령의 합계입니다. Amazon Data Firehose는 백업이 실패한 문서에만 사용되는지 아니면 모든 문서에 사용되는지에 관계없이 항상 이 지표를 내보냅니다.
DeliveryToAmazonOpenSearchServerless.AuthFailure	인증 및 권한 부여 오류. OS/ES 클러스터 정책 및 역할 권한을 확인합니다. 0은 문제 없음을 나타내며 1은 인증 실패를 나타냅니다.

지표	설명
DeliveryToAmazonOpenSearchServerless.DeliveryRejected	전송 거부 오류. OS/ES 클러스터 정책 및 역할 권한을 확인합니다. 0은 문제 없음을 나타내며 1은 전송 실패를 나타냅니다.

Amazon Redshift로 전송

지표	설명
DeliveryToRedshift.Bytes	지정한 시간 동안 Amazon Redshift으로 복사된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToRedshift.Records	지정한 시간 동안 Amazon Redshift으로 복사된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToRedshift.Success	성공한 Amazon Redshift COPY 명령의 합계입니다.
DeliveryToS3.Bytes	지정한 시간 동안 Amazon S3로 전송된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToS3.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 S3 버킷으로 전송됩니다. 단위: 초
DeliveryToS3.Records	지정한 시간 동안 Amazon S3로 전송된 레코드 수.

지표	설명
	통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToS3.Success	성공적인 Amazon S3 풋 명령의 합계입니다.
DeliveryToRedshift.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 기간보다 오래된 모든 레코드는 Amazon Redshift 클러스터로 전달됩니다.
BackupToS3.Bytes	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 바이트 수. Amazon S3에 대한 백업이 활성화된 경우 Amazon Data Firehose는 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
BackupToS3.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 백업을 위해 Amazon S3 버킷으로 전송되었습니다. Amazon S3에 대한 백업이 활성화된 경우 Amazon Data Firehose는 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
BackupToS3.Records	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 레코드 수. Amazon S3에 대한 백업이 활성화된 경우 Amazon Data Firehose는 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

지표	설명
BackupToS3.Success	백업을 위한 성공적인 Amazon S3 풋 명령의 합계입니다. Amazon S3에 대한 백업이 활성화된 경우 Amazon Data Firehose는 이 지표를 내보냅니다.

Amazon S3으로 전송

다음 표의 지표는 Firehose 스트림의 주요 대상인 Amazon S3로의 전송과 관련이 있습니다.

지표	설명
DeliveryToS3.Bytes	지정한 시간 동안 Amazon S3로 전송된 바이트 수. 데이터 변환이 활성화되면이 지표는 변환 전 사전 처리된 바이트 크기를 반영합니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToS3.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식 (Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 S3 버킷으로 전송되었습니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToS3.Records	지정한 시간 동안 Amazon S3로 전송된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToS3.Success	성공적인 Amazon S3 풋 명령의 합계입니다.
BackupToS3.Bytes	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 바이트 수. Amazon Data Firehose는 백업이 활성화된 경우 이

지표	설명
	지표를 생성합니다(데이터 변환까지 활성화된 경우에만 가능). 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
BackupToS3.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 백업을 위해 Amazon S3 버킷으로 전송되었습니다. Amazon Data Firehose는 백업이 활성화된 경우 이 지표를 생성합니다(데이터 변환까지 활성화된 경우에만 가능). 통계: Minimum, Maximum, Average, Samples 단위: 초
BackupToS3.Records	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 레코드 수. Amazon Data Firehose는 백업이 활성화된 경우 이 지표를 생성합니다(데이터 변환까지 활성화된 경우에만 가능). 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
BackupToS3.Success	백업을 위한 성공적인 Amazon S3 풋 명령의 합계입니다. Amazon Data Firehose는 백업이 활성화된 경우 이 지표를 생성합니다(데이터 변환까지 활성화된 경우에만 가능).

Snowflake로 전송

지표	설명
DeliveryToSnowflake.Bytes	지정한 시간 동안 Snowflake로 전송된 바이트 수

지표	설명
	통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToSnowflake.DataFreshness	Firehose에서 가장 오래된 레코드의 경과 시간(Firehose에 입력된 시점부터 현재까지)입니다. 이 경과 시간보다 오래된 레코드는 모두 Snowflake에 전송되었습니다. Firehose 삽입 호출이 성공한 후 Snowflake에 데이터를 커밋하는 데 몇 초가 걸릴 수 있습니다. Snowflake에 데이터를 커밋하는 데 걸리는 시간은 DeliveryToSnowflake.DataCommitLatency 지표를 참조하세요. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToSnowflake.DataCommitLatency	Firehose가 레코드를 성공적으로 삽입한 후 데이터가 Snowflake에 커밋되는 데 걸리는 시간입니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToSnowflake.Records	지정한 시간 동안 Snowflake로 전송된 레코드 수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToSnowflake.Success	Snowflake에 수행된 성공적인 삽입 호출의 합계입니다.

지표	설명
DeliveryToS3.Bytes	지정한 시간 동안 Amazon S3로 전송된 바이트 수. 이 지표는 Snowflake로의 전송이 실패하고 Firehose가 실패한 데이터를 S3로 백업하려고 시도하는 경우에만 사용할 수 있습니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToS3.Records	지정한 시간 동안 Amazon S3로 전송된 레코드 수. 이 지표는 Snowflake로의 전송이 실패하고 Firehose가 실패한 데이터를 S3로 백업하려고 시도하는 경우에만 사용할 수 있습니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToS3.Success	성공적인 Amazon S3 풋 명령의 합계입니다. 이 지표는 Snowflake로의 전송이 실패하고 Firehose가 실패한 데이터를 S3로 백업하려고 시도하는 경우에만 사용할 수 있습니다.
BackupToS3.DataFreshness	Firehose에서 가장 오래된 레코드의 경과 시간(Firehose에 입력된 시점부터 현재까지)입니다. 이 시간보다 오래된 레코드는 Amazon S3 버킷에 백업됩니다. 이 지표는 Firehose 스트림이 모든 데이터를 백업하도록 구성된 경우 사용할 수 있습니다. 통계: Minimum, Maximum, Average, Samples 단위: 초

지표	설명
BackupToS3.Records	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 레코드 수. 이 지표는 Firehose 스트림이 모든 데이터를 백업하도록 구성된 경우 사용할 수 있습니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위:개수
BackupToS3.Bytes	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 바이트 수. 이 지표는 Firehose 스트림이 모든 데이터를 백업하도록 구성된 경우 사용할 수 있습니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위:개수
BackupToS3.Success	백업을 위한 성공적인 Amazon S3 풋 명령의 합계입니다. Firehose 스트림이 모든 데이터를 백업하도록 구성될 때 Firehose는 이 지표를 생략합니다.

Splunk에 전송

지표	설명
DeliveryToSplunk.Bytes	지정한 시간 동안 Splunk로 전송된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToSplunk.DataAckLatency	Amazon Data Firehose에서 데이터를 전송한 후 Splunk로부터 승인을 수신하기까지 걸리는 대략적인 시간입니다. 이 측정치에 대한 증가 또는 감소 추세가 절대 근사치보다 더 유용합니다. 증가 추세는 Splunk 인덱서로부터 더 느린 인덱싱 및 승인 비율을 나타낼 수 있습니다. 통계: Minimum, Maximum, Average, Samples

지표	설명
	단위: 초
DeliveryToSplunk.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 Splunk로 전송되었습니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToSplunk.Records	지정한 시간 동안 Splunk로 전송된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToSplunk.Success	성공적으로 인덱싱된 레코드의 합계입니다.
DeliveryToS3.Success	성공적인 Amazon S3 풋 명령의 합계입니다. 이 지표는 Amazon S3으로의 백업이 활성화된 경우 내보내집니다.
BackupToS3.Bytes	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 바이트 수. Firehose 스트림이 모든 문서를 백업하도록 구성된 경우 Amazon Data Firehose는 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

지표	설명
BackupToS3.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 백업을 위해 Amazon S3 버킷으로 전송되었습니다. Firehose 스트림이 모든 문서를 백업하도록 구성된 경우 Amazon Data Firehose는 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
BackupToS3.Records	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 레코드 수. Firehose 스트림이 모든 문서를 백업하도록 구성된 경우 Amazon Data Firehose는 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
BackupToS3.Success	백업을 위한 성공적인 Amazon S3 풋 명령의 합계입니다. Firehose 스트림이 모든 문서를 백업하도록 구성된 경우 Amazon Data Firehose는 이 지표를 내보냅니다.

HTTP 엔드포인트로 전송

지표	설명
DeliveryToHttpEndpoint.Bytes	HTTP 엔드포인트에 성공적으로 전송된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToHttpEndpoint.Records	HTTP 엔드포인트에 성공적으로 전송된 레코드 수. 이 지표는 전송 시도가 성공한 경우에만 내보내지며 전송 시도가 실패할 때는 내보내지지 않습니다.

지표	설명
	통계: Minimum, Maximum, Average, Sum, Samples 단위: 개수
DeliveryToHttpEndpoint.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToHttpEndpoint.Success	전송 시도당 HTTP 엔드포인트로 성공적으로 전송된 레코드 수입니다. 와 달리 DeliveryToHttpEndpoint.Records 이 지표는 모든 전송 시도에 대해 내보내집니다. 성공 시 값은 전송 시도의 레코드 수와 같습니다. 전송 시도에서 모든 레코드가 실패하면 값은 0입니다. 최소 통계를 사용하여 전송 실패를 모니터링합니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToHttpEndpoint.ProcessedBytes	재시도를 포함하여 처리 시도된 바이트 수.
DeliveryToHttpEndpoint.ProcessedRecords	재시도를 포함하여 시도된 레코드 수.

데이터 수집 지표

주제

- Kinesis Data Streams를 통한 데이터 통합
- Direct PUT을 통한 데이터 수집
- MSK에서 데이터 통합

Kinesis Data Streams를 통한 데이터 통합

지표	설명
DataReadFromKinesisStream.Bytes	데이터 원본이 Kinesis 데이터 스트림인 경우 이 지표는 해당 데이터 스트림에서 읽은 바이트 수를 나타냅니다. 이 수는 장애 조치로 인한 다시 읽기를 포함합니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DataReadFromKinesisStream.Records	데이터 원본이 Kinesis 데이터 스트림인 경우 이 지표는 해당 데이터 스트림에서 읽은 레코드 수를 나타냅니다. 이 수는 장애 조치로 인한 다시 읽기를 포함합니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
ThrottledDescribeStream	데이터 원본이 Kinesis 데이터 스트림일 때 DescribeStream 작업에서 병목 현상이 일어나는 총 횟수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
ThrottledGetRecords	데이터 원본이 Kinesis 데이터 스트림일 때 GetRecords 작업에서 병목 현상이 일어나는 총 횟수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
ThrottledGetShardIterator	데이터 원본이 Kinesis 데이터 스트림일 때 GetShardIterator 작업에서 병목 현상이 일어나는 총 횟수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

지표	설명
KinesisMillisBehindLatest	데이터 원본이 Kinesis 데이터 스트림일 때, 이 지표는 마지막으로 읽은 레코드가 Kinesis 데이터 스트림의 최신 레코드보다 뒤쳐진 밀리초 수를 의미합니다. 통계: Minimum, Maximum, Average, Samples 단위: 밀리초

Direct PUT을 통한 데이터 수집

지표	설명
BackupToS3.Bytes	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 바이트 수. Amazon Data Firehose는 Amazon S3 또는 Amazon Redshift 대상에 데이터 변환이 활성화된 경우 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
BackupToS3.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 백업을 위해 Amazon S3 버킷으로 전송되었습니다. Amazon Data Firehose는 Amazon S3 또는 Amazon Redshift 대상에 데이터 변환이 활성화된 경우 이 지표를 내보냅니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
BackupToS3.Records	지정한 시간 동안 백업을 위해 Amazon S3로 전송된 레코드 수. Amazon Data Firehose는 Amazon S3 또는 Amazon Redshift 대상에 데이터 변환이 활성화된 경우 이 지표를 내보냅니다.

지표	설명
	통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
BackupToS3.Success	백업을 위한 성공적인 Amazon S3 풋 명령의 합계입니다. Amazon Data Firehose는 Amazon S3 또는 Amazon Redshift 대상에 데이터 변환이 활성화된 경우 이 지표를 내보냅니다.
BytesPerSecondLimit	조절 전에 Firehose 스트림이 수집할 수 있는 현재 초당 최대 바이트 수입니다. 이 한도 증가를 요청하려면 AWS Support Center에서 사례 생성을 선택한 다음 서비스 한도 증가를 선택합니다.
DeliveryToAmazonOpenSearchService.Bytes	지정된 기간 동안 OpenSearch Service로 인덱싱된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToAmazonOpenSearchService.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 OpenSearch Service에 전송되었습니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToAmazonOpenSearchService.Records	지정된 기간 동안 OpenSearch Service로 인덱싱된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToAmazonOpenSearchService.Success	성공적으로 인덱싱된 레코드의 합계입니다.

지표	설명
DeliveryToRedshift.Bytes	지정한 시간 동안 Amazon Redshift으로 복사된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToRedshift .Records	지정한 시간 동안 Amazon Redshift으로 복사된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToRedshift .Success	성공한 Amazon Redshift COPY 명령의 합계입니다.
DeliveryToS3.Bytes	지정한 시간 동안 Amazon S3로 전송된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToS3.DataF reshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식 (Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 S3 버킷으로 전송되었습니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToS3.Records	지정한 시간 동안 Amazon S3로 전송된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToS3.Success	성공적인 Amazon S3 풋 명령의 합계입니다.

지표	설명
DeliveryToSplunk.Bytes	지정한 시간 동안 Splunk로 전송된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DeliveryToSplunk.DataAckLatency	Amazon Data Firehose에서 데이터를 전송한 후 Splunk로부터 승인을 수신하기까지 걸리는 대략적인 시간입니다. 이 측정치에 대한 증가 또는 감소 추세가 절대 근사치보다 더 유용합니다. 증가 추세는 Splunk 인덱서로부터 더 느린 인덱싱 및 승인 비율을 나타낼 수 있습니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToSplunk.DataFreshness	Amazon Data Firehose에서 가장 오래된 레코드의 연식(Amazon Data Firehose에 입력된 시점부터 현재까지). 이 경과 시간보다 오래된 레코드는 모두 Splunk로 전송되었습니다. 통계: Minimum, Maximum, Average, Samples 단위: 초
DeliveryToSplunk.Records	지정한 시간 동안 Splunk로 전송된 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DeliveryToSplunk.Success	성공적으로 인덱싱된 레코드의 합계입니다.

지표	설명
IncomingBytes	지정된 기간 동안 Firehose 스트림에 성공적으로 수집된 바이트 수. Firehose 스트림 한도 중 하나라도 초과하면 데이터 수집이 제한될 수 있습니다. 제한된 데이터는 IncomingBytes 에 계산되지 않습니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
IncomingPutRequests	지정된 기간 동안 성공한 PutRecord 및 PutRecordBatch 요청 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
IncomingRecords	지정된 기간 동안 Firehose 스트림에 성공적으로 수집된 레코드 수. Firehose 스트림 한도 중 하나라도 초과하면 데이터 수집이 제한될 수 있습니다. 제한된 데이터는 IncomingRecords 에 계산되지 않습니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
RecordsPerSecondLimit	조절 전에 Firehose 스트림이 수집할 수 있는 현재 초당 최대 레코드 수입니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
ThrottledRecords	데이터 수집이 Firehose 스트림 제한 중 하나를 초과했기 때문에 조절된 레코드 수입니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

MSK에서 데이터 통합

지표	설명
DataReadFromSource.Records	소스 Kafka 주제에서 읽은 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
DataReadFromSource.Bytes	소스 Kafka 주제에서 읽은 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
SourceThrottled.Delay	소스 Kafka 클러스터가 소스 Kafka 주제의 레코드를 반환하는 데 지연된 시간. 통계: Minimum, Maximum, Average, Samples 단위: 밀리초
BytesPerSecondLimit	Firehose가 소스 Kafka 주제의 각 파티션에서 읽을 수 있는 처리량의 현재 한도. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트/초
KafkaOffsetLag	Firehose가 소스 Kafka 주제에서 읽은 레코드의 최대 오프셋과 소스 Kafka 주제에서 사용할 수 있는 레코드의 최대 오프셋 간의 차이. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
FailedValidation.Records	레코드 검증에 실패한 레코드 수. 통계: Minimum, Maximum, Average, Sum, Samples

지표	설명
	단위: 개
FailedValidation.Bytes	레코드 검증에 실패한 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
DataReadFromSource.Backpressured	파티션당 BytesPersecdLimit가 초과되었거나 정상적인 전송 흐름의 느린 속도 또는 중지로 인해 Firehose 스트림이 소스 파티션에서 레코드를 읽는 데 지연이 발생했음을 나타냅니다. 단위: 부울

API 수준 CloudWatch 지표

AWS/Firehose 네임스페이스에는 다음과 같은 API 수준 지표가 포함되어 있습니다.

지표	설명
DescribeDeliveryStream.Latency	DescribeDeliveryStream 작업 1건당 지정한 시간 동안 측정된 소요 시간 통계: Minimum, Maximum, Average, Samples 단위: 밀리초
DescribeDeliveryStream.Requests	DescribeDeliveryStream 요청 총 수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
ListDeliveryStreams.Latency	ListDeliveryStream 작업 1건당 지정한 시간 동안 측정된 소요 시간 통계: Minimum, Maximum, Average, Samples

지표	설명
	단위: 밀리초
ListDeliveryStreams.Requests	ListFirehose 요청 총 수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
PutRecord.Bytes	지정한 시간 동안 PutRecord 를 사용하여 Firehose 스트림에 입력된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
PutRecord.Latency	PutRecord 작업 1건당 지정한 시간 동안 측정된 소요 시간 통계: Minimum, Maximum, Average, Samples 단위: 밀리초
PutRecord.Requests	PutRecord 요청 총 수로서 PutRecord 작업의 총 레코드 수와 동일함 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
PutRecordBatch.Bytes	지정한 시간 동안 PutRecordBatch 를 사용하여 Firehose 스트림에 입력된 바이트 수. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트

지표	설명
PutRecordBatch.Latency	PutRecordBatch 작업 1건당 지정한 시간 동안 측정된 소요 시간 통계: Minimum, Maximum, Average, Samples 단위: 밀리초
PutRecordBatch.Records	PutRecordBatch 작업의 총 레코드 수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
PutRecordBatch.Requests	PutRecordBatch 요청 총 수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
PutRequestsPerSecondLimit	조절 전에 Firehose 스트림이 처리할 수 있는 초당 최대 PUT 요청 수입니다. 여기에는 PutRecord 및 PutRecord Batch 요청이 포함됩니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
ThrottledDescribeStream	데이터 원본이 Kinesis 데이터 스트림일 때 DescribeStream 작업에서 병목 현상이 일어나는 총 횟수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
ThrottledGetRecords	데이터 원본이 Kinesis 데이터 스트림일 때 GetRecords 작업에서 병목 현상이 일어나는 총 횟수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

지표	설명
ThrottledGetShardIterator	데이터 원본이 Kinesis 데이터 스트림일 때 GetShardIterator 작업에서 병목 현상이 일어나는 총 횟수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
UpdateDeliveryStream.Latency	UpdateDeliveryStream 작업 1건당 지정한 시간 동안 측정된 소요 시간 통계: Minimum, Maximum, Average, Samples 단위: 밀리초
UpdateDeliveryStream.Requests	UpdateDeliveryStream 요청 총 수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

데이터 변환 CloudWatch 지표

Lambda를 이용한 데이터 변환이 활성화된 경우 AWS/Firehose 네임스페이스에 다음과 같은 지표가 포함됩니다.

지표	설명
ExecuteProcessing.Duration	Firehose가 각 Lambda 함수 호출을 수행하는 데 걸리는 시간. 단위: 밀리초
ExecuteProcessing.Success	전체 Lambda 함수 호출의 합계 대비 성공한 Lambda 함수 호출의 합계.

지표	설명
SucceedProcessing.Records	지정한 시간 동안 성공적으로 처리된 레코드 수 단위: 개
SucceedProcessing.Bytes	지정한 시간 동안 성공적으로 처리된 바이트 수 단위: 바이트

CloudWatch Logs 압축 해제 지표

CloudWatch Logs 전송에 압축 해제가 활성화된 경우 AWS/Firehose 네임스페이스에는 다음 지표가 포함됩니다.

지표	설명
OutputDecompressedBytes.Success	바이트 단위의 성공적인 압축 해제 데이터 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
OutputDecompressedBytes.Failed	바이트 단위의 압축 해제된 데이터 실패 통계: Minimum, Maximum, Average, Sum, Samples 단위: 바이트
OutputDecompressedRecords.Success	압축 해제된 레코드 성공 수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
OutputDecompressedRecords.Failed	실패한 압축 해제된 레코드 수 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

형식 변환 CloudWatch 지표

형식 변환이 활성화된 경우 AWS/Firehose 네임스페이스에 다음 지표가 포함됩니다.

지표	설명
SucceedConversion.Records	성공적으로 변환된 레코드 수. 단위: 개
SucceedConversion.Bytes	성공적으로 변환된 레코드의 크기. 단위: 바이트
FailedConversion.Records	변환하지 못한 레코드 수. 단위: 개
FailedConversion.Bytes	변환하지 못한 레코드의 크기. 단위: 바이트

서버 측 암호화(SSE) CloudWatch 지표

AWS/Firehose 네임스페이스에는 SSE와 관련된 다음 지표가 포함됩니다.

지표	설명
KMSKeyAccessDenied	Firehose 스트림에 대해 서비스에서 KMSAccessDeniedException 이 발생한 횟수입니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
KMSKeyDisabled	Firehose 스트림에 대해 서비스에서 KMSDisabledException 이 발생한 횟수입니다. 통계: Minimum, Maximum, Average, Sum, Samples

지표	설명
	단위: 개
KMSKeyInvalidState	Firehose 스트림에 대해 서비스에서 KMSInvalidStateException 이 발생한 횟수입니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개
KMSKeyNotFound	Firehose 스트림에 대해 서비스에서 KMSNotFoundException 이 발생한 횟수입니다. 통계: Minimum, Maximum, Average, Sum, Samples 단위: 개

Amazon Data Firehose의 측정기준

Firehose 스트림별로 측정치를 필터링하려면 DeliveryStreamName 측정기준을 사용합니다.

Amazon Data Firehose 사용 지표

CloudWatch 사용량 지표를 사용하여 계정의 리소스 사용량을 확인할 수 있습니다. 이러한 지표를 사용하여 CloudWatch 그래프 및 대시보드에서 현재 서비스 사용량을 시각화합니다.

서비스 할당량 사용량 지표는 AWS/사용 네임스페이스에 있으며 3분마다 수집됩니다.

현재 CloudWatch가 게시하는 이 네임스페이스의 유일한 지표 이름은 ResourceCount입니다. 이 지표는 Service, Class, Type 및 Resource 차원으로 게시됩니다.

지표	설명
ResourceCount	계정에서 실행 중인 지정된 리소스의 수입니다. 리소스는 지표와 연결된 차원에 의해 정의됩니다. 이 지표에 가장 유용한 통계는 3분 동안 사용된 최대 리소스 수를 나타내는 MAXIMUM입니다.

다음 측정기준은 Amazon Data Firehose에 의해 게시되는 사용량 지표를 구체화하는 데 사용됩니다.

차원	설명
`Service`	리소스가 포함된 AWS 서비스의 이름입니다. Amazon Data Firehose 사용량 지표의 경우 이 측정기준 값은 `Firehose`입니다.
`Class`	추적 중인 리소스의 클래스입니다. Amazon Data Firehose API 사용량 지표는 이 측정기준을 None 값과 함께 사용합니다.
`Type`	추적 중인 리소스의 유형입니다. 현재 서비스 차원이 `Firehose`인 경우 Type에 대한 유일한 유효한 값은 `Resource`입니다.
`Resource`	AWS 리소스의 이름입니다. 현재 서비스 차원이 `Firehose`인 경우 Resource에 대한 유일한 유효한 값은 `DeliveryStreams` 입니다.

Amazon Data Firehose의 CloudWatch 지표 액세스

CloudWatch 콘솔, 명령줄 또는 CloudWatch API를 사용하여 Amazon Data Firehose에 대한 지표를 모니터링할 수 있습니다. 다음의 절차는 이처럼 다양한 방법을 사용하여 측정치에 액세스하는 방법을 설명합니다.

CloudWatch 콘솔을 사용하여 지표에 액세스

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 모음에서 리전을 선택합니다.
3. 탐색 창에서 지표(Metrics)를 선택합니다.
4. [Firehose] 네임스페이스를 선택합니다.
5. [Firehose 스트림 지표] 또는 [Firehose Metrics]를 선택합니다.
6. 지표를 선택하여 그래프에 추가합니다.

를 사용하여 지표에 액세스하려면 AWS CLI

list-metrics 명령과 get-metric-statistics 명령을 사용합니다.

```
aws cloudwatch list-metrics --namespace "AWS/Firehose"
```

```
aws cloudwatch get-metric-statistics --namespace "AWS/Firehose" \
--metric-name DescribeDeliveryStream.Latency --statistics Average --period 3600 \
--start-time 2017-06-01T00:00:00Z --end-time 2017-06-30T00:00:00Z
```

CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링

데이터 변환 또는 데이터 전송을 위한 Lambda 호출에 실패할 경우 특정 오류 로그를 확인할 수 있도록 Amazon Data Firehose가 Amazon CloudWatch Logs와 통합됩니다. Firehose 스트림을 생성할 때 Amazon Data Firehose 오류 로깅을 활성화할 수 있습니다.

Amazon Data Firehose 콘솔에서 Amazon Data Firehose 오류 로깅을 활성화하면 Firehose 스트림에 대해 로그 그룹과 해당 로그 스트림이 자동으로 생성됩니다. 로그 그룹 이름의 형식은 `/aws/kinesisfirehose/`*`delivery-stream-name`*이며, 여기서 *`delivery-stream-name`*는 해당 Firehose 스트림의 이름입니다. `DestinationDelivery`는 기본 대상으로의 전송과 관련된 오류를 기록하기 위해 생성되어 사용되는 로그 스트림입니다. `BackupDelivery`(이)라는 또 다른 로그 스트림은 대상에 대해 S3 백업이 활성화된 경우에만 생성됩니다. `BackupDelivery` 로그 스트림은 S3 백업으로 전송하는 것과 관련된 오류를 기록하는 데 사용됩니다.

예를 들어 Amazon Redshift를 대상으로 Firehose 스트림 "MyStream"을 생성하고 Amazon Data Firehose 오류 로그를 활성화하면, `aws/kinesisfirehose/MyStream`이라는 로그 그룹과 `DestinationDelivery` 및 BackupDelivery라는 두 개의 로그 스트림이 자동으로 생성됩니다. 이 예시에서 `DestinationDelivery`은(는) Amazon Redshift 대상 및 중간 S3 대상으로의 전송과 관련된 오류를 기록하는 데 사용됩니다. `BackupDelivery`, S3 백업이 활성화된 경우 S3 백업 버킷으로의 전송과 관련된 오류를 기록하는 데 사용됩니다.

, API를 통해 또는 `CloudWatchLoggingOptions` 구성을 CloudFormation 사용하여 Amazon Data Firehose 오류 로깅 AWS CLI을 활성화할 수 있습니다. 이렇게 하려면 로그 그룹과 로그 스트림을 미리 생성합니다. Amazon Data Firehose 오류 로깅만을 위한 로그 그룹과 로그 스트림을 예약하는 것이 좋습니다. 또한 연결된 IAM 정책에 `"logs:putLogEvents"` 권한이 있는지도 확인합니다. 자세한 내용은 Amazon Data Firehose를 통한 액세스 제어 섹션을 참조하세요.

Amazon Data Firehose는 모든 전송 오류 로그가 CloudWatch Logs로 전송된다고 보장하지 않습니다. 전송 실패율이 높은 경우 Amazon Data Firehose가 전송 오류 로그를 CloudWatch Logs로 보내기 전에 샘플링합니다.

CloudWatch Logs로 전송되는 오류 로그에는 일반 요금이 부과됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

내용

데이터 전송 오류

다음은 데이터 전송 오류 코드 목록과 각 Amazon Data Firehose 대상에 대한 메시지입니다. 각 오류 메시지는 문제를 해결하기 취해야 하는 적절한 조치도 설명합니다.

오류

Amazon S3 데이터 전송 오류

Amazon Data Firehose는 다음과 같은 Amazon S3 관련 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
S3.KMS.NotFoundException	"제공된 AWS KMS 키를 찾을 수 없습니다. 올바른 역할을 가진 유효한 AWS KMS 키라고 생각되는 키를 사용하는 경우 AWS KMS 키가 연결된 계정에 문제가 있는지 확인합니다."
S3.KMS.RequestLimitExceeded	"S3 객체 암호화를 시도하는 동안 초당 KMS 요청 제한을 초과했습니다. 초당 제한을 늘리세요." 자세한 정보는 AWS Key Management Service 개발자 안내서의 제한을 참조하세요.
S3.AccessDenied	"액세스가 거부되었습니다. 입력한 IAM 역할에 대한 신뢰 정책에 따라 Amazon Data Firehose가 역할을 사용할 수 있고, 액세스 정책에 따라 S3 버킷에 액세스가 가능해야 합니다."
S3.AccountProblem	" AWS 계정에 작업이 성공적으로 완료되지 못하는 문제가 있습니다. AWS Support에 문의하세요."
S3.AllAccessDisabled	"입력하신 계정에 대한 액세스가 거부되었습니다. AWS Support에 문의하세요."
S3.InvalidPayer	"입력하신 계정에 대한 액세스가 거부되었습니다. AWS Support에 문의하세요."
S3.NotSignedUp	"계정이 Amazon S3에 가입되지 않았습니다. 계정에 가입하거나 다른 계정을 사용하세요."
S3.NoSuchBucket	"지정된 버킷이 존재하지 않습니다. 버킷을 새로 만들거나, 다른 기존 버킷을 사용하세요."
S3.MethodNotAllowed	"지정된 방법이 이 리소스에 허용되지 않습니다. 올바른 Amazon S3 작업 권한을 허용하도록 버킷 정책을 수정하세요."
InternalError	"데이터 전송을 시도하는 동안 내부 오류가 발생했습니다. 전송이 재시도됩니다. 오류가 지속되면 해결을 AWS 위해에 보고됩니다."
S3.KMS.KeyDisabled	"제공된 KMS 키가 비활성화되어 있습니다. 키를 활성화하거나 다른 키를 사용하세요."

오류 코드	오류 메시지와 정보
S3.KMS.InvalidStateException	“제공된 KMS 키가 유효하지 않은 상태입니다. 다른 키를 사용하세요.”
KMS.InvalidStateException	“제공된 KMS 키가 유효하지 않은 상태입니다. 다른 키를 사용하세요.”
KMS.DisabledException	“제공된 KMS 키가 비활성화되어 있습니다. 키를 수정하거나 다른 키를 사용하세요.”
S3.SlowDown	“지정된 버킷에 대한 put 요청 비율이 너무 높았습니다. Firehose 스트림 버퍼 크기를 늘리거나 다른 애플리케이션의 PUT 요청을 줄이세요.”
S3.SubscriptionRequired	“S3를 호출하는 중 액세스가 거부되었습니다. 전달된 IAM 역할 및 KMS 키(입력한 경우)가 Amazon S3을 구독하고 있는지 확인하세요.”
S3.InvalidToken	“입력된 토큰의 형식이 잘못되었거나 유효하지 않습니다. 입력된 자격 증명을 확인하세요.”
S3.KMS.KeyNotConfigured	“KMS 키가 구성되어 있지 않습니다. KMSmasterKeyID를 구성하거나 S3 버킷의 암호화를 비활성화하세요.”
S3.KMS.AsymmetricCMKNotSupported	“Amazon S3는 대칭 CMK만 지원합니다. 비대칭 CMK를 사용하여 Amazon S3에서 데이터를 암호화할 수 없습니다. CMK 유형을 확인하려면 KMS DescribeKey 작업을 사용하세요.”
S3.IllegalLocationConstraintException	“현재 Firehose는 구성된 s3 버킷으로 데이터를 전송하기 위해 s3 글로벌 엔드포인트를 사용합니다. 구성된 s3 버킷의 리전은 s3 글로벌 엔드포인트를 지원하지 않습니다. s3 버킷과 동일한 리전에 Firehose 스트림을 생성하거나 글로벌 엔드포인트를 지원하는 리전에서 s3 버킷을 사용하세요.”
S3.InvalidPrefixConfigurationException	“타임스탬프 평가에 사용된 사용자 지정 s3 접두사가 유효하지 않습니다. s3 접두사에 해당 연도의 현재 날짜 및 시간에 대한 올바른 표현식이 포함되어 있는지 확인하세요.”

오류 코드	오류 메시지와 정보
`DataFormatConversion.MalformedData`	“토큰 사이에 잘못된 문자가 있습니다.”

Apache Iceberg 테이블 데이터 전송 오류

Apache Iceberg 테이블 데이터 전송 오류에 대해서는, Apache Iceberg 테이블에 데이터 전송을 참조하세요.

Amazon Redshift 데이터 전송 오류

Amazon Data Firehose는 다음과 같은 Amazon Redshift 관련 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
`Redshift.TableNotFound`	"데이터를 로드할 테이블을 찾을 수 없습니다. 지정된 테이블이 있어야 합니다." S3로부터 데이터를 복사해야 하는 Amazon Redshift의 대상 테이블을 찾을 수 없습니다. Amazon Redshift 테이블이 없는 경우, Amazon Data Firehose는 테이블을 만들지 않습니다.
`Redshift.SyntaxError`	"COPY 명령에 구문 오류가 있습니다. 명령을 다시 시도하세요."
`Redshift.AuthenticationFailed`	"입력하신 사용자 이름과 암호가 인증에 실패했습니다. 유효한 사용자 이름 및 암호를 입력하세요."
`Redshift.AccessDenied`	"액세스가 거부되었습니다. 입력한 IAM 역할에 대한 신뢰 정책에 따라 Amazon Data Firehose가 역할을 사용할 수 있어야 합니다."

오류 코드	오류 메시지와 정보
`Redshift.S3BucketAccessDenied`	"COPY 명령으로 S3 버킷에 액세스할 수 없습니다. 입력한 IAM 역할에 대한 액세스 정책에 따라 S3 버킷에 액세스할 수 있어야 합니다."
`Redshift.DataLoadFailed`	"테이블로 데이터를 로드하지 못했습니다. 자세한 정보는 STL_LOAD_ERRORS 시스템 테이블을 확인하세요."
`Redshift.ColumnNotFound`	"COPY 명령의 열이 테이블에 없습니다. 유효한 열 이름을 지정하세요."
`Redshift.DatabaseNotFound`	"Amazon Redshift 대상 구성 또는 JDBC URL에 지정된 데이터베이스를 찾을 수 없습니다. 유효한 데이터베이스 이름을 지정하세요."
`Redshift.IncorrectCopyOptions`	"충돌하거나 중복되는 COPY 옵션을 입력했습니다. 일부 옵션은 특정 조합에서 호환되지 않습니다. 자세한 내용은 COPY 명령 레퍼런스를 확인하세요." 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 Amazon Redshift COPY 명령을 참조하세요.
`Redshift.MissingColumn`	"테이블 스키마에 DEFAULT 값 없이 NOT NULL로 정의되어 있고 열 목록에 포함되지 않은 열이 있습니다. 이 열을 제외하고, 로드한 데이터가 항상 이 열에 대한 값을 제공하는지 확인하거나 이 테이블에 대한 Amazon Redshift 스키마에 기본 값을 추가하세요."
`Redshift.ConnectionFailed`	"지정된 Amazon Redshift 클러스터에 연결하지 못했습니다. 보안 설정이 Amazon Data Firehose 연결을 허용하는지, Amazon Redshift 대상 구성 또는 JDBC URL에 지정된 클러스터나 데이터베이스가 올바른지, 클러스터를 사용할 수 있는지 확인하세요."
`Redshift.ColumnMismatch`	"COPY 명령의 jsonpaths 수와 대상 테이블의 열 수가 일치해야 합니다. 명령을 다시 시도하세요."

오류 코드	오류 메시지와 정보
Redshift.IncorrectOrMissingRegion	"Amazon Redshift가 S3 버킷에 액세스하기 위해 잘못된 리전 엔드포인트를 사용하려고 했습니다. COPY 명령 옵션에 올바른 리전 값을 지정하거나 S3 버킷이 Amazon Redshift 데이터베이스와 같은 리전에 있는지 확인하세요."
Redshift.IncorrectJsonPathsFile	"제공된 jsonpaths 파일이 지원되는 JSON 형식이 아닙니다. 명령을 다시 시도하세요."
Redshift.MissingS3File	"Amazon Redshift가 요구한 하나 이상의 S3 파일이 S3 버킷에서 제거되었습니다. S3 버킷 정책을 확인하여 S3 파일의 자동 삭제를 제거하세요."
Redshift.InsufficientPrivilege	"사용자는 데이터를 테이블에 로드할 권한이 없습니다. Amazon Redshift 사용자 권한에 INSERT 권한이 있는지 확인하세요."
Redshift.ReadOnlyCluster	"시스템이 크기 조정 모드에 있기 때문에 쿼리를 실행할 수 없습니다. 나중에 다시 쿼리를 시도하세요."
Redshift.DiskFull	"디스크가 가득 차 데이터를 로드할 수 없습니다. Amazon Redshift 클러스터의 용량을 늘리거나 사용하지 않는 데이터를 삭제해 디스크 공간을 확보하세요."
InternalError	"데이터 전송을 시도하는 동안 내부 오류가 발생했습니다. 전송이 재시도됩니다. 오류가 지속되면 해결을 AWS 위해에 보고됩니다."
Redshift.ArgumentNotSupported	"COPY 명령에 지원되지 않는 옵션이 포함되어 있습니다."
Redshift.AnalyzeTableAccessDenied	"액세스가 거부되었습니다. 테이블 또는 데이터베이스 소유자만 테이블 분석을 수행할 수 있기 때문에 S3에서 Redshift로의 복사가 실패합니다."

오류 코드	오류 메시지와 정보
Redshift.SchemaNotFound	"Amazon Redshift 대상 구성의 DataTableName에 지정된 스키마를 찾을 수 없습니다. 유효한 스키마 이름을 지정하세요.”
Redshift.ColumnSpecifiedMoreThanOnce	“열 목록에 두 번 이상 지정된 열이 있습니다. 중복된 열은 제거해야 합니다.”
Redshift.ColumnNotNullWithoutDefault	“열 목록에 포함되어 있지 않으며 DEFAULT가 없는, null이 아닌 열이 있습니다. 해당 열이 열 목록에 포함되어 있는지 확인하세요.”
Redshift.IncorrectBucketRegion	“Redshift가 클러스터와 다른 리전에 있는 버킷을 사용하려고 시도했습니다. 클러스터와 같은 리전 내의 버킷을 지정하세요.”
Redshift.S3SlowDown	“S3에 대한 높은 요청율. 전송률이 저하되지 않도록 속도를 낮추세요.”
Redshift.InvalidCopyOptionForJson	“json CopyOption에 대해 자동 또는 유효한 S3 경로를 사용하세요.”
Redshift.InvalidCopyOptionJSONPathFormat	“COPY 실패 오류 \”잘못된 JSONPath 형식입니다. 배열 인덱스가 범위를 벗어났습니다\”. JSONPath 표현식을 수정하세요.”
Redshift.InvalidCopyOptionRBACAclNotAllowed	“COPY 실패 오류 \”권한 전파가 활성화되지 않은 상태에서는 RBAC acl 프레임워크를 사용할 수 없습니다.\”

오류 코드	오류 메시지와 정보
Redshift.DiskSpaceQuotaExceeded	“디스크 공간 할당량 초과로 인해 트랜잭션이 중단되었습니다. 디스크 공간을 확보하거나 스키마의 할당량 증가를 요청하세요.”
Redshift.ConnectionsLimitExceeded	“사용자에 대한 연결 제한이 초과되었습니다.”
Redshift.SslNotSupported	“서버가 SSL을 지원하지 않기 때문에 지정된 Amazon Redshift 클러스터에 연결하지 못했습니다. 클러스터 설정을 확인하세요.”
Redshift.HoseNotFound	“호스가 삭제되었습니다. 호스의 상태를 확인하세요.”
Redshift.Delimiter	“CopyCommand의 copyOptions 구분 기호가 잘못되었습니다. 단일 문자여야 합니다.”
Redshift.QueryCancelled	“사용자가 COPY 작업을 취소했습니다.”
Redshift.CompressionMismatch	“호스는 UNCOMPRESSED로 구성되었지만 CopyOption에 압축 형식이 포함되어 있습니다.”
Redshift.EncryptionCredentials	“ENCRYPTED 옵션을 사용하려면 다음 형식의 자격 증명이 필요합니다: 'aws_iam_role=...;master_symmetric_key=...' 또는 'aws_access_key_id=...;aws_secret_access_key=...[;token=...];master_symmetric_key=...'”
Redshift.InvalidCopyOptions	“COPY 구성 옵션이 잘못되었습니다.”
Redshift.InvalidMessageFormat	“COPY 명령에 잘못된 문자가 있습니다.”

오류 코드	오류 메시지와 정보
`Redshift.TransactionIdLimitReached`	“거래 ID 한도에 도달했습니다.”
`Redshift.DestinationRemoved`	“redshift 대상이 존재하고 Firehose 구성에 올바르게 설정되어 있는지 확인하세요.”
`Redshift.OutOfMemory`	“Redshift 클러스터에 메모리가 부족합니다. 클러스터의 용량이 충분한지 확인하세요.”
`Redshift.CannotForkProcess`	“Redshift 클러스터에 메모리가 부족합니다. 클러스터의 용량이 충분한지 확인하세요.”
`Redshift.SslFailure`	“핸드셰이크 중에 SSL 연결이 종료되었습니다.”
`Redshift.Resize`	“Redshift 클러스터의 크기가 조정되고 있습니다. 클러스터 크기가 조정되는 동안에는 Firehose가 데이터를 전송할 수 없습니다.”
`Redshift.ImproperQualifiedName`	“인증된 이름이 적절하지 않습니다(점 표시 이름이 너무 많음).”
`Redshift.InvalidJsonPathFormat`	“잘못된 JSONPath 형식입니다.”
`Redshift.TooManyConnectionsException`	“Redshift에 대한 연결이 너무 많습니다.”
`Redshift.PSQLException`	“Redshift에서 PSQlException이 발견되었습니다.”

오류 코드	오류 메시지와 정보
`Redshift.DuplicateSecondsSpecification`	“날짜/시간 형식의 초 지정이 중복되었습니다.”
`Redshift.RelationCouldNotBeOpened`	“Redshift 오류가 발생했습니다. 관계를 열 수 없습니다. 지정된 DB에 대한 Redshift 로그를 확인하세요.”
`Redshift.TooManyClients`	“Redshift에서 너무 많은 클라이언트 예외가 발생했습니다. 여러 명의 생산자가 동시에 데이터베이스에 데이터를 쓰는 경우 데이터베이스에 대한 최대 연결 수를 다시 확인하세요.”

Snowflake 데이터 전송 오류

Firehose는 다음과 같은 Snowflake 관련 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
`Snowflake.InvalidUrl`	“Firehose가 Snowflake에 연결할 수 없습니다. Snowflake 대상 구성에서 계정 URL이 올바르게 지정되었는지 확인하세요.”
`Snowflake.InvalidUser`	“Firehose가 Snowflake에 연결할 수 없습니다. Snowflake 대상 구성에서 사용자가 올바르게 지정되었는지 확인하세요.”
`Snowflake.InvalidRole`	“지정된 snowflake 역할이 존재하지 않거나 권한이 없습니다. 지정된 사용자에게 역할이 부여되었는지 확인하세요.”
`Snowflake.InvalidTable`	“제공된 테이블이 없거나 권한이 없습니다.”
`Snowflake.InvalidSchema`	“제공된 스키마가 존재하지 않거나 권한이 없습니다.”

오류 코드	오류 메시지와 정보
`Snowflake.InvalidDatabase`	“제공된 데이터베이스가 존재하지 않거나 권한이 없습니다.”
`Snowflake.InvalidPrivateKeyOrPassphrase`	“지정된 프라이빗 키 또는 암호가 유효하지 않습니다. 제공된 프라이빗 키는 유효한 PEM RSA 프라이빗 키여야 합니다.”
`Snowflake.MissingColumns`	“입력 페이로드의 열이 누락되어 삽입 요청이 거부되었습니다. nullable이 아닌 모든 열에 대해 값이 지정되어 있는지 확인합니다.”
`Snowflake.ExtraColumns`	“추가 열로 인해 삽입 요청이 거부되었습니다. 테이블에 없는 열은 지정하면 안 됩니다.”
`Snowflake.InvalidInput`	“잘못된 입력 형식으로 인해 전송에 실패했습니다. 제공된 입력 페이로드가 허용되는 JSON 형식인지 확인합니다.”
`Snowflake.IncorrectValue`	“입력 페이로드의 잘못된 데이터 유형으로 인해 전송에 실패했습니다. 입력 페이로드에 지정된 JSON 값이 Snowflake 테이블 정의에 선언된 데이터 유형을 준수하는지 확인합니다.”

Splunk 데이터 전송 오류

Amazon Data Firehose는 다음과 같은 Splunk 관련 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
`Splunk.ProxyWithoutStickySessions`	"Amazon Data Firehose 노드와 HEC 노드 사이에 프록시(ELB 또는 기타)가 있는 경우, HEC ACK를 지원하려면 고정 세션을 활성화해야 합니다."
`Splunk.DisabledToken`	"HEC 토큰이 비활성화됩니다. Splunk로의 데이터 전송을 허용하려면 이 토큰을 활성화하세요."

오류 코드	오류 메시지와 정보
`Splunk.InvalidToken`	"HEC 토큰이 잘못되었습니다. 유효한 HEC 토큰으로 Amazon Data Firehose를 업데이트하세요.”
`Splunk.InvalidDataFormat`	"데이터의 형식이 올바르지 않습니다. 원시 또는 이벤트 HEC 엔드포인트에 대해 데이터 형식을 적절히 지정하는 방법은 Splunk Event Data를 참조하세요."
`Splunk.InvalidIndex`	"HEC 토큰 또는 입력이 잘못된 인덱스로 구성되어 있습니다. 인덱스 구성을 확인한 후 다시 시도하세요."
`Splunk.ServerError`	“HEC 노드의 서버 오류로 인해 Splunk로 데이터를 전송하지 못했습니다. Amazon Data Firehose의 재시도 기간이 0보다 큰 경우 Amazon Data Firehose는 데이터 전송을 재시도합니다. 재시도가 모두 실패하면 Amazon Data Firehose는 데이터를 Amazon S3에 백업합니다.”
`Splunk.DisabledAck`	"HEC 토큰에 대해 인덱서 확인이 비활성화됩니다. 인덱서 확인을 활성화하고 다시 시도하세요. 자세한 정보는 Enable indexer acknowledgement를 참조하세요."
`Splunk.AckTimeout`	"HEC 확인 제한 시간 만료 전에 HEC에서 확인을 받지 않았습니다. 확인 제한 시간에도 불구하고, Splunk에서 데이터를 올바로 인덱싱할 수 있었습니다. Amazon Data Firehose는 확인 제한 시간이 만료된 데이터를 Amazon S3에 백업합니다.”
`Splunk.MaxRetriesFailed`	"Splunk로의 데이터 전송 또는 확인 수신에 실패했습니다. HEC 상태를 확인한 후 다시 시도하세요."
`Splunk.ConnectionTimeout`	"Splunk 연결 시간을 초과했습니다. 이는 일시적인 오류일 수 있으며 요청이 재시도됩니다. 재시도가 모두 실패하면 Amazon Data Firehose는 데이터를 Amazon S3에 백업합니다.”
`Splunk.InvalidEndpoint`	"HFC 엔드포인트에 연결할 수 없습니다. HEC 엔드포인트 URL이 유효하고 Amazon Data Firehose에서 이 URL에 접근 가능한지 확인합니다."
`Splunk.ConnectionClosed`	"연결 실패로 인해 Splunk에 데이터를 전송할 수 없습니다. 일시적 오류일 수 있습니다. Amazon Data Firehose 구성에서 재시도 지속시간을 늘리면 이러한 일시적 오류를 방지할 수도 있습니다."

오류 코드	오류 메시지와 정보
`Splunk.SSLUnverified`	"HFC 엔드포인트에 연결할 수 없습니다. 호스트가 피어에 의해 제공된 인증서와 일치하지 않습니다. 인증서와 호스트가 유효한지 확인하세요."
`Splunk.SSLHandshake`	"HFC 엔드포인트에 연결할 수 없습니다. 인증서와 호스트가 유효한지 확인하세요."
`Splunk.URLNotFound`	“Splunk 서버에서 요청된 URL을 찾을 수 없습니다. Splunk 클러스터가 올바르게 구성되었는지 확인하세요.”
`Splunk.ServerError.ContentTooLarge`	“StatusCode: 413(메시지: 클라이언트가 보낸 요청이 너무 큼)과 함께 서버 오류로 인해 Splunk로 데이터를 전송하지 못했습니다. max_content_length 설정은 Splunk 문서를 참조하세요.”
`Splunk.IndexerBusy`	“HEC 노드의 서버 오류로 인해 Splunk로 데이터를 전송하지 못했습니다. HEC 엔드포인트 또는 Elastic Load Balancer가 정상 상태이며 연결 가능한지 확인하세요.”
`Splunk.ConnectionRecycled`	“Firehose와 Splunk의 연결이 다시 순환되었습니다. 전송을 다시 시도합니다.”
`Splunk.AcknowledgementsDisabled`	“POST에 대한 확인을 받을 수 없습니다. HEC 엔드포인트에 확인이 활성화되어 있는지 확인하세요.”
`Splunk.InvalidHecResponseCharacter`	“HEC 응답에 잘못된 문자가 있습니다. 서비스와 HEC 구성을 확인하세요.”

ElasticSearch 데이터 전송 오류

Amazon Data Firehose는 다음과 같은 ElasticSearch 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
ES.AccessDenied	"액세스가 거부되었습니다. Firehose와 관련하여 입력된 IAM 역할이 삭제되지 않았는지 확인하세요.”
ES.ResourceNotFound	"지정된 AWS Elasticsearch 도메인이 존재하지 않습니다."

HTTPS 엔드포인트 데이터 전송 오류

Amazon Data Firehose는 다음과 같은 HTTP 엔드포인트 관련 오류를 CloudWatch Logs로 전송할 수 있습니다. 이 오류 중에 현재 발생한 문제와 일치하는 오류가 없는 경우 기본 오류는 다음과 같습니다: “데이터 전송을 시도하는 동안 내부 오류가 발생했습니다. 전송이 재시도됩니다. 오류가 지속되면 해결을 AWS 위해에 보고됩니다.”

오류 코드	오류 메시지와 정보
HttpEndpoint.RequestTimeout	응답을 받기 전에 전송 시간이 초과되었으므로 다시 시도됩니다. 이 오류가 지속되면 AWS Firehose 서비스 팀에 문의하세요.
HttpEndpoint.ResponseTooLarge	“엔드포인트에서 수신한 응답이 너무 큽니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”
HttpEndpoint.InvalidResponseFromDestination	“지정된 엔드포인트에서 받은 응답이 유효하지 않습니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”
HttpEndpoint.DestinationException	“엔드포인트 대상으로부터 다음 응답을 수신했습니다.”

오류 코드	오류 메시지와 정보
HttpEndpoint.ConnectionFailed	“대상 엔드포인트에 연결할 수 없습니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”
HttpEndpoint.ConnectionReset	“엔드포인트와의 연결을 유지할 수 없습니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”
HttpEndpoint.ConnectionReset	“엔드포인트와의 연결을 유지하는 데 문제가 있습니다. 엔드드포인트 소유자에게 문의하세요.”
HttpEndpoint.ResponseReasonPhraseExceededLimit	“엔드포인트에서 수신한 응답 이유 문구가 구성된 제한인 64자를 초과했습니다.”
HttpEndpoint.InvalidResponseFromDestination	“엔드포인트에서 받은 응답이 유효하지 않습니다. 자세한 내용은 Firehose 설명서의 HTTP 엔드포인트 문제 해결을 참조하세요. 원인: “
HttpEndpoint.DestinationException	“엔드포인트로의 전송에 실패했습니다. 자세한 내용은 Firehose 설명서의 HTTP 엔드포인트 문제 해결을 참조하세요. 상태 코드와 함께 응답을 수신함 ”
HttpEndpoint.InvalidStatusCode	“잘못된 응답 상태 코드를 수신했습니다.”
HttpEndpoint.SSLHandshakeFailure	“엔드포인트를 통한 SSL 핸드셰이크를 완료할 수 없습니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”

오류 코드	오류 메시지와 정보
HttpEndpoint.SSLHandshakeFailure	“엔드포인트를 통한 SSL 핸드셰이크를 완료할 수 없습니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”
HttpEndpoint.SSLFailure	“엔드포인트를 통한 TLS 핸드셰이크를 완료할 수 없습니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”
HttpEndpoint.SSLHandshakeCertificatePathFailure	“잘못된 인증 경로로 인해 엔드포인트를 통한 SSL 핸드셰이크를 완료할 수 없습니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”
HttpEndpoint.SSLHandshakeCertificatePathValidationFailure	“인증 경로 검증 실패로 인해 엔드포인트를 통한 SSL 핸드셰이크를 완료할 수 없습니다. 엔드포인트 소유자에게 문의하여 이 문제를 해결하세요.”
HttpEndpoint.MakeRequestFailure.IllegalUriException	“잘못된 URI 입력으로 인해 HttpendPoint 요청이 실패했습니다. 입력한 URI의 모든 문자가 올바른지 확인하세요.”
HttpEndpoint.MakeRequestFailure.IllegalCharacterInHeaderValue	“잘못된 응답 오류로 인해 HttpendPoint 요청이 실패했습니다. 헤더 값에 잘못된 '\n' 문자가 있습니다.”

오류 코드	오류 메시지와 정보
HttpEndpoint.IllegalResponseFailure	“잘못된 응답 오류로 인해 HttpendPoint 요청이 실패했습니다. HTTP 메시지는 콘텐츠-유형 헤더를 두 개 이상 포함할 수 없습니다.”
HttpEndpoint.IllegalMessageStart	“잘못된 응답 오류로 인해 HttpendPoint 요청이 실패했습니다. HTTP 메시지 시작이 잘못되었습니다. 자세한 내용은 Firehose 설명서의 HTTP 엔드포인트 문제 해결을 참조하세요.”

Amazon OpenSearch Service 데이터 전송 오류

OpenSearch Service 대상의 경우 Amazon Data Firehose는 OpenSearch Service에서 오류가 반환되면 이를 CloudWatch Logs에 전송합니다.

OpenSearch 클러스터에서 반환될 수 있는 오류 외에도 다음 두 가지 오류가 발생할 수 있습니다.

- 대상 OpenSearch Service 클러스터에 데이터를 전달하려고 시도하는 동안 인증/권한 부여 오류가 발생합니다. 이는 권한 문제로 인해 발생하거나 Amazon Data Firehose의 대상 OpenSearch Service 도메인 구성이 수정될 때 간헐적으로 발생할 수 있습니다. 클러스터 정책 및 역할 권한을 확인하세요.
- 인증/권한 부여 실패로 인해 대상 OpenSearch 서비스 클러스터로 데이터를 전달할 수 없습니다. 이는 권한 문제로 인해 발생하거나 Amazon Data Firehose의 대상 OpenSearch Service 도메인 구성이 수정될 때 간헐적으로 발생할 수 있습니다. 클러스터 정책 및 역할 권한을 확인하세요.

오류 코드	오류 메시지와 정보
OS.AccessDenied	"액세스가 거부되었습니다. 입력한 IAM 역할에 대한 신뢰 정책에 따라 Firehose가 역할을 사용할 수 있고, 액세스 정책에 따라 Amazon OpenSearch Service API에 액세스가 가능해야 합니다."
OS.AccessDenied	"액세스가 거부되었습니다. 입력한 IAM 역할에 대한 신뢰 정책에 따라 Firehose가 역할을 사용할 수 있고, 액세스 정책에 따라 Amazon OpenSearch Service API에 액세스가 가능해야 합니다."

오류 코드	오류 메시지와 정보
OS.AccessDenied	"액세스가 거부되었습니다. Firehose와 관련하여 입력된 IAM 역할이 삭제되지 않았는지 확인하세요.”
OS.AccessDenied	"액세스가 거부되었습니다. Firehose와 관련하여 입력된 IAM 역할이 삭제되지 않았는지 확인하세요.”
OS.ResourceNotFound	“지정된 Amazon OpenSearch Service 도메인이 존재하지 않습니다.”
OS.ResourceNotFound	“지정된 Amazon OpenSearch Service 도메인이 존재하지 않습니다.”
OS.AccessDenied	"액세스가 거부되었습니다. 입력한 IAM 역할에 대한 신뢰 정책에 따라 Firehose가 역할을 사용할 수 있고, 액세스 정책에 따라 Amazon OpenSearch Service API에 액세스가 가능해야 합니다."
OS.RequestTimeout	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션에 대한 요청 시간이 초과되었습니다. 클러스터 또는 컬렉션에 현재 워크로드에 필요한 용량이 충분한지 확인하세요.”
OS.ClusterError	“Amazon OpenSearch Service 클러스터가 지정되지 않은 오류를 반환했습니다.”
OS.RequestTimeout	“Amazon OpenSearch Service 클러스터에 대한 요청 시간이 초과되었습니다. 클러스터에 현재 워크로드에 필요한 용량이 충분한지 확인하세요.”
OS.ConnectionFailed	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션에 연결하는 데 문제가 있습니다. 클러스터 또는 컬렉션이 정상 상태이고 연결 가능한지 확인하세요.”
OS.ConnectionReset	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션과의 연결을 유지할 수 없습니다. 클러스터 또는 컬렉션 소유자에게 문의하여 이 문제를 해결하세요.”
OS.ConnectionReset	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션과의 연결을 유지하는 데 문제가 있습니다. 클러스터 또는 컬렉션이 정상 상태이고 현재 워크로드에 필요한 용량이 충분한지 확인하세요.”

오류 코드	오류 메시지와 정보
OS.ConnectionReset	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션과의 연결을 유지하는 데 문제가 있습니다. 클러스터 또는 컬렉션이 정상 상태이고 현재 워크로드에 필요한 용량이 충분한지 확인하세요.”
OS.AccessDenied	"액세스가 거부되었습니다. Amazon OpenSearch Service 클러스터의 액세스 정책이 구성된 IAM 역할에 대한 액세스를 허용하는지 확인하세요.”
OS.ValidationException	“OpenSearch 클러스터가 ESServiceException을 반환했습니다. 한 가지 원인으로는, 클러스터가 OS 2.x 이상으로 업그레이드되었지만 호스에 여전히 TypeName 파라미터가 구성되어 있기 때문입니다. TypeName을 빈 문자열로 설정하여 호스 구성을 업데이트하거나 엔드포인트를 Type 파라미터를 지원하는 클러스터로 변경하세요.”
OS.ValidationException	“구성원은 다음 정규 표현식 패턴을 충족해야 합니다: [a-z][a-z0-9\\ -]+
OS.JsonParseException	“Amazon OpenSearch Service 클러스터가 JSONParseException을 반환했습니다. 입력되는 데이터가 유효한지 확인하세요.”
OS.AmazonOpenSearchServiceParseException	“Amazon OpenSearch Service 클러스터가 AmazonOpenSearchServiceParseException를 반환했습니다. 입력되는 데이터가 유효한지 확인하세요.”
OS.ExplicitIndexInBulkNotAllowed	“Amazon OpenSearch Service 클러러스터에 rest.action.multi.allow_explicit_index가 true로 설정되어 있는지 확인하세요.”
OS.ClusterError	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션이 지정되지 않은 오류를 반환했습니다.”
OS.ClusterBlockException	“클러스터가 ClusterBlockException를 반환했습니다. 과부하가 발생했을 수 있습니다.”
OS.InvalidARN	“입력된 Amazon OpenSearch Service ARN이 유효하지 않습니다. DeliveryStream 구성을 확인하세요.”

오류 코드	오류 메시지와 정보
OS.MalformedData	“하나 이상의 레코드 형식이 잘못되었습니다. 각 레코드는 하나의 유효한 JSON 객체여야 하며, 줄 바꿈이 포함되지 않아야 합니다.”
OS.InternalError	"데이터 전송을 시도하는 동안 내부 오류가 발생했습니다. 전송이 재시도됩니다. 오류가 지속되면 해결을 AWS 위해에 보고됩니다.”
OS.AliasWithMultipleIndicesNotAllowed	“별칭에 연결된 인덱스가 두 개 이상 있습니다. 별칭에 연결된 인덱스는 하나만 있어야 합니다.”
OS.UnsupportedVersion	“Amazon OpenSearch Service 6.0은 현재 Amazon Data Firehose에서 지원되지 않습니다. 자세한 내용은 AWS Support에 문의하세요.”
OS.CharConversionException	“하나 이상의 레코드에 잘못된 문자가 포함되어 있습니다.”
OS.InvalidDomainNameLength	“도메인 이름 길이가 유효한 OS 한도를 벗어났습니다.”
OS.VPCDomainNotSupported	“VPC 내의 Amazon OpenSearch Service 도메인은 현재 지원되지 않습니다.”
OS.ConnectionError	“http 서버가 예기치 않게 연결을 끊었습니다. Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션의 상태를 확인하세요.”
OS.LargeFieldData	“Amazon OpenSearch Service 클러스터에 허용된 것보다 큰 필드 데이터가 포함되어 있어 요청을 중단했습니다.”
OS.BadGateway	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션은 502 Bad Gateway 응답과 함께 요청을 중단했습니다.”

오류 코드	오류 메시지와 정보
`OS.ServiceException`	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션으로부터 오류가 수신되었습니다. 클러스터 또는 컬렉션이 VPC 뒤에 있는 경우 네트워크 구성이 연결을 허용하는지 확인하세요.”
`OS.GatewayTimeout`	“Amazon OpenSearch Service 클러스터 또는 OpenSearch Serverless 컬렉션에 연결하는 도중 Firehose에 시간 초과 오류가 발생했습니다.”
`OS.MalformedData`	“Amazon Data Firehose는 Firehose 레코드 내에 Amazon OpenSearch Service Bulk API 명령을 지원하지 않습니다.”
`OS.ResponseEntryCountMismatch`	“Bulk API의 응답에 전송된 레코드 수보다 많은 항목이 포함되었습니다. 각 레코드에 하나의 JSON 객체만 포함될 수 있으며 줄 바꿈이 없어야 합니다.”

Lambda 호출 오류

Amazon Data Firehose는 다음과 같은 Lambda 호출 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
`Lambda.AssumeRoleAccessDenied`	"액세스가 거부되었습니다. 입력한 IAM 역할에 대한 신뢰 정책에 따라 Amazon Data Firehose가 역할을 사용할 수 있어야 합니다."
`Lambda.InvokeAccessDenied`	"액세스가 거부되었습니다. 액세스 정책이 Lambda 함수에 대한 액세스를 허용해야 합니다."
`Lambda.JsonProcessingException`	"Lambda 함수에서 반환한 레코드를 구문 분석하는 중 오류가 발생했습니다. 반환된 레코드가 Amazon Data Firehose에서 요구하는 상태 모델을 따르는지 확인하세요." 자세한 내용은 데이터 변환에 필요한 파라미터 섹션을 참조하세요.

오류 코드	오류 메시지와 정보
`Lambda.InvokeLimitExceeded`	"Lambda 동시 실행 한도를 초과했습니다. 동시 실행 한도를 늘리세요." 자세한 정보는 AWS Lambda 개발자 안내서의 AWS Lambda 제한을 참조하세요.
`Lambda.DuplicatedRecordId`	"동일한 레코드 ID로 여러 레코드가 반환되었습니다. Lambda 함수가 각 레코드에 고유한 레코드 ID를 반환하는지 확인하세요." 자세한 내용은 데이터 변환에 필요한 파라미터 섹션을 참조하세요.
`Lambda.MissingRecordId`	"하나 이상의 레코드 ID가 반환되었습니다. Lambda 함수가 수신한 모든 레코드 ID를 반환하는지 확인하세요." 자세한 내용은 데이터 변환에 필요한 파라미터 섹션을 참조하세요.
`Lambda.ResourceNotFound`	"지정된 Lambda 함수가 존재하지 않습니다. 다른 기존 함수를 사용하세요."
`Lambda.InvalidSubnetIDException`	"Lambda 함수 VPC 구성에서 지정된 서브넷 ID가 유효하지 않습니다. 서브넷 ID가 유효한지 확인하세요."
`Lambda.InvalidSecurityGroupIDException`	"Lambda 함수 VPC 구성에서 지정된 보안 그룹 ID가 유효하지 않습니다. 보안 그룹 ID가 유효한지 확인하세요."
`Lambda.SubnetIPAddressLimitReachedException`	"하나 이상의 구성된 서브넷에 사용 가능한 IP 주소가 없기 때문에 Lambda 함수에 대한 VPC 액세스를AWS Lambda 설정할 수 없습니다. IP 주소 제한을 늘리세요." 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC 제한 - VPC 및 서브넷을 참조하세요.

오류 코드	오류 메시지와 정보
`Lambda.ENILimitReachedException`	“네트워크 인터페이스 한도에 도달했기 때문에 Lambda 함수 구성의 일부로 지정된 VPC에서 탄력적 네트워크 인터페이스(ENI)를 생성할 수AWS Lambda 없었습니다. 네트워크 인터페이스 제한을 늘리세요." 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC 제한 - 네트워크 인터페이스를 참조하세요.
`Lambda.FunctionTimedOut`	Lambda 함수가 시간을 초과했습니다. Lambda 함수의 Timeout 설정을 늘리세요. 자세한 내용은 함수 제한 시간을 참조하세요.
`Lambda.FunctionError`	이는 다음 오류 중 하나로 인해 발생할 수 있습니다. • 잘못된 출력 구조. 함수를 확인하고 출력이 필수 형식인지 확인하세요. 또한 처리된 레코드에 `Dropped`, `Ok` 또는 `ProcessingFailed` 의 유효한 결과 상태가 포함되어 있는지 확인하세요. • Lambda 함수가 성공적으로 호출되었지만 오류 결과를 반환했습니다. • KMS 액세스가 거부되었기 때문에 Lambda가 환경 변수를 복호화할 수 없습니다. 함수의 KMS 키 설정과 키 정책을 확인하세요. 자세한 내용은 키 액세스 문제 해결을 참조하세요.
`Lambda.FunctionRequestTimedOut`	Amazon Data Firehose에서 Lambda를 호출하는 중 Request did not complete before the request timeout configuration(요청 시간 초과 구성 전에 요청이 완료되지 않음) 오류가 발생했습니다. Lambda 코드를 다시 검토하여 Lambda 코드가 구성된 제한 시간을 초과하여 실행되어야 하는지 확인하세요. 그런 경우, 메모리, 제한 시간을 포함한 Lambda 구성 설정의 조정을 고려하세요. 자세한 내용은 Lambda 함수 옵션 구성을 참조하세요.
`Lambda.TargetServerFailedToRespond`	Amazon Data Firehose에 오류가 발생했습니다. AWS Lambda 서비스를 호출할 때 대상 서버가 오류에 응답하지 못했습니다.
`Lambda.InvalidZipFileException`	Amazon Data Firehose에서 Lambda 함수를 호출하는 중 InvalidZipFileException 오류가 발생했습니다. Lambda 함수 구성 설정 및 Lambda 코드 압축 파일을 확인하세요.

오류 코드	오류 메시지와 정보
`Lambda.InternalServerError`	“ AWS Lambda 서비스를 호출할 때 Amazon Data Firehose에서 InternalServerError가 발생했습니다. Amazon Data Firehose는 정해진 횟수만큼 데이터 전송을 재시도합니다. `CreateDeliveryStream` 또는 `UpdateDestination` API를 사용하여 재시도 옵션을 정의하거나 재정의할 수 있습니다. 오류가 지속되면 AWS Lambda 지원팀에 문의하십시오.
`Lambda.ServiceUnavailable`	Amazon Data Firehose에서 AWS Lambda 서비스를 호출할 때 ServiceUnavailableException이 발생했습니다. Amazon Data Firehose는 정해진 횟수만큼 데이터 전송을 재시도합니다. `CreateDeliveryStream` 또는 `UpdateDestination` API를 사용하여 재시도 옵션을 정의하거나 재정의할 수 있습니다. 오류가 지속되면 AWS Lambda 지원팀에 문의하십시오.
`Lambda.InvalidSecurityToken`	잘못된 보안 토큰으로 인해 Lambda 함수를 호출할 수 없습니다. 파티션 간 Lambda 호출은 지원되지 않습니다.
`Lambda.InvocationFailure`	이는 다음 오류 중 하나로 인해 발생할 수 있습니다. • AWS Lambda를 호출할 때 Amazon Data Firehose에서 오류가 발생했습니다. 작업이 다시 시도됩니다. 오류가 지속될 경우 해결을 위해 AWS에 보고됩니다." • Amazon Data Firehose에서 Lambda의 KMSInvalidStateException 오류가 발생했습니다. 오류: 사용된 KMS 키가 복호화에 대해 잘못된 상태에 있으므로 Lambda가 환경 변수를 복호화할 수 없습니다. Lambda 함수의 KMS 키를 확인하세요. • Amazon Data Firehose에서 Lambda의 AWS LambdaException 오류가 발생했습니다. Lambda가 제공된 컨테이너 이미지를 초기화하지 못했습니다. 이미지를 확인하세요. • Amazon Data Firehose에서 AWS Lambda를 호출할 때 제한 시간 오류가 발생했습니다. 지원되는 함수 제한 시간은 최대 5분입니다. 자세한 내용은 데이터 변환 실행 기간간을 참조하세요.

오류 코드	오류 메시지와 정보
`Lambda.JsonMappingException`	Lambda 함수에서 반환한 레코드를 구문 분석하는 중 오류가 발생했습니다. 데이터 필드가 Base-64로 인코딩되었는지 확인하세요.

Kinesis 호출 오류

Amazon Data Firehose는 다음과 같은 Kinesis 호출 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
`Kinesis.AccessDenied`	“Kinesis를 호출하는 동안 액세스가 거부되었습니다. 사용된 IAM 역할의 액세스 정책이 해당 Kinesis API에 대한 액세스를 허용하는지 확인하세요.”
`Kinesis.ResourceNotFound`	“Firehose가 스트림에서 데이터를 읽지 못했습니다. Firehose가 Kinesis Stream에 연결되어 있는 경우 스트림이 존재하지 않거나 샤드가 병합 또는 분할되었을 수 있습니다. Firehose가 DirectPut 유형인 경우 Firehose가 더 이상 존재하지 않을 수 있습니다.”
`Kinesis.SubscriptionRequired`	“Kinesis를 호출하는 동안 액세스가 거부되었습니다. Kinesis 스트림 액세스를 위해 전달된 IAM 역할에 AWS Kinesis 구독이 있는지 확인합니다.”
`Kinesis.Throttling`	“Kinesis를 호출하는 동안 제한(Throttling)오류가 발생했습니다. 이는 다른 애플리케이션이 Firehose 스트림과 동일한 API를 호출하거나, 사용자가 소스와 동일한 Kinesis 스트림을 사용하여 Firehose 스트림을 너무 많이 생성했기 때문일 수 있습니다.”
`Kinesis.Throttling`	“Kinesis를 호출하는 동안 제한(Throttling)오류가 발생했습니다. 이는 다른 애플리케이션이 Firehose 스트림과 동일한 API를 호출하거나, 사용자가 소스와 동일한 Kinesis 스트림을 사용하여 Firehose 스트림을 너무 많이 생성했기 때문일 수 있습니다.”

오류 코드	오류 메시지와 정보
`Kinesis.AccessDenied`	“Kinesis를 호출하는 동안 액세스가 거부되었습니다. 사용된 IAM 역할의 액세스 정책이 해당 Kinesis API에 대한 액세스를 허용하는지 확인하세요.”
`Kinesis.AccessDenied`	“기본 Kinesis Stream에서 API 작업을 호출하려고 시도하는 동안 액세스가 거부되었습니다. IAM 역할이 전파되어 있고 유효한지 확인하세요.”
`Kinesis.KMS.AccessDeniedException`	“Firehose가 Kinesis 스트림을 암호화/복호화하는 데 사용되는 KMS 키에 액세스할 수 없습니다. Firehose 전송 역할에 키에 대한 액세스 권한을 부여하세요.”
`Kinesis.KMS.KeyDisabled`	“암호화/복호화에 사용되는 KMS 키가 비활성화되어 있기 때문에 Firehose가 소스 Kinesis Stream을 읽을 수 없습니다. 읽기를 계속할 수 있도록 키를 활성화하세요.”
`Kinesis.KMS.InvalidStateException`	“암호화에 사용되는 KMS 키가 유효하지 않은 상태기 때문에 Firehose가 소스 Kinesis Stream을 읽을 수 없습니다.”
`Kinesis.KMS.NotFoundException`	“암호화에 사용되는 KMS 키를 찾을 수 없기 때문에 Firehose가 소스 Kinesis Stream을 읽을 수 없습니다.”

Kinesis DirectPut 호출 오류

Amazon Data Firehose는 다음과 같은 Kinesis DirectPut 호출 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
`Firehose.KMS.Acces`	“Firehose가 KMS 키에 액세스할 수 없습니다. 키 정책을 확인하세요.”

오류 코드	오류 메시지와 정보
sDeniedEx ception	
Firehose. KMS.Inval idStateEx ception	“암호화에 사용되는 KMS 키가 유효하지 않은 상태기 때문에 Firehose가 데이터를 복호화할 수 없습니다.”
Firehose. KMS.NotFo undException	“암호화에 사용되는 KMS 키를 찾을 수 없기 때문에 Firehose가 데이터를 복호화할 수 없습니다.”
Firehose. KMS.KeyDi sabled	“데이터 암호화에 사용되는 KMS 키가 비활성화되어 있기 때문에 Firehose가 데이터를 복호화할 수 없습니다. 데이터 전송을 계속할 수 있도록 키를 활성화하세요.”

AWS Glue 호출 오류

Amazon Data Firehose는 CloudWatch Logs에 다음과 같은 AWS Glue 호출 오류를 보낼 수 있습니다.

오류 코드	오류 메시지와 정보
DataForma tConversi on.Invali dSchema	“스키마가 유효하지 않습니다.”
DataForma tConversi on.Entity NotFound	“지정된 테이블 또는 데이터베이스를 찾을 수 없습니다. 테이블/데이터베이스가 존재하고 스키마 구성에 제공된 값이 특히 대/소문자와 관련하여 올바른지 확인하세요.”
DataForma tConversi on.Invali dInput	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 제공된 카탈로그 ID를 가진 지정된 데이터베이스가 존재하는지 확인하세요.”

오류 코드	오류 메시지와 정보
DataFormatConversion.InvalidInput	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 전달된 ARN이 올바른 형식인지 확인하세요.”
DataFormatConversion.InvalidInput	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 제공된 CataloGID가 유효한지 확인하세요.”
DataFormatConversion.InvalidVersionId	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 지정된 버전의 테이블이 존재하는지 확인하세요.”
DataFormatConversion.NonExistentColumns	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 테이블이 대상 열을 포함한, null이 아닌 스토리지 설명자로 구성되어 있는지 확인하세요.”
DataFormatConversion.AccessDenied	“역할 수임 중 액세스가 거부되었습니다. 데이터 형식 변환 구성에 지정된 역할이 Firehose 서비스에 역할을 사용할 수 있는 권한을 부여했는지 확인하세요.”
DataFormatConversion.ThrottledByGlue	“Glue를 호출하는 동안 제한(Throttling)오류가 발생했습니다. 요청 비율 제한을 늘리거나 다른 애플리케이션을 통해 Glue를 호출하는 현재 비율을 낮추세요.”
DataFormatConversion.AccessDenied	“Glue를 호출하는 동안 액세스가 거부되었습니다. 데이터 형식 변환 구성에 지정된 역할에 필수 권한이 있는지 확인하세요.”

오류 코드	오류 메시지와 정보
DataFormatConversion.InvalidGlueRole	“잘못된 역할입니다. 데이터 형식 변환 구성에 지정된 역할이 존재하는지 확인하세요.”
DataFormatConversion.InvalidGlueRole	“요청에 포함된 보안 토큰이 잘못되었습니다. Firehose와 관련하여 입력된 IAM 역할이 삭제되지 않았는지 확인하세요.”
DataFormatConversion.GlueNotAvailableInRegion	“지정한 리전에서 아직AWS Glue를 사용할 수 없습니다. 다른 리전을 지정하세요.”
DataFormatConversion.GlueEncryptionException	“마스터 키를 검색하는 중 오류가 발생했습니다. 키가 존재하고 올바른 액세스 권한이 있는지 확인하세요.”
DataFormatConversion.SchemaValidationTimeout	“Glue에서 테이블을 가져오는 중 시간이 초과되었습니다. Glue 테이블 버전이 많은 경우 'Glue:GettableVersion' 권한을 추가하거나 (권장) 사용하지 않는 테이블 버전을 삭제하세요. Glue에 테이블 수가 많지 않은 경우 AWS Support에 문의하세요.”
DataFirehose.InternalError	“Glue에서 테이블을 가져오는 중 시간이 초과되었습니다. Glue 테이블 버전이 많은 경우 'Glue:GettableVersion' 권한을 추가하거나 (권장) 사용하지 않는 테이블 버전을 삭제하세요. Glue에 테이블 수가 많지 않은 경우 AWS Support에 문의하세요.”

오류 코드	오류 메시지와 정보
DataFormatConversion.GlueEncryptionException	“마스터 키를 검색하는 중 오류가 발생했습니다. 키가 존재하고 상태가 정상인지 확인하세요.”

DataFormatConversion 호출 오류

Amazon Data Firehose는 다음과 같은 DataFormatConversion 호출 오류를 CloudWatch Logs로 전송할 수 있습니다.

오류 코드	오류 메시지와 정보
DataFormatConversion.InvalidSchema	“스키마가 유효하지 않습니다.”
DataFormatConversion.ValidationException	“열 이름과 유형은 비어 있지 않은 문자열이어야 합니다.”
DataFormatConversion.ParseError	“잘못된 JSON 형식입니다.”
DataFormatConversion.MalformedData	“데이터가 스키마와 일치하지 않습니다.”
DataFormatConversi	“json 키의 길이는 262144를 초과할 수 없습니다.”

오류 코드	오류 메시지와 정보
on.Malfor medData	
DataForma tConversi on.Malfor medData	“데이터를 UTF-8 형식으로 디코딩할 수 없습니다.”
DataForma tConversi on.Malfor medData	“토큰 사이에 잘못된 문자가 있습니다.”
DataForma tConversi on.Invali dTypeFormat	“잘못된 Type 형식입니다. Type 구문을 확인하세요.”
DataForma tConversi on.Invali dSchema	“잘못된 스키마입니다. 열 이름에 특수 문자나 공백이 있는지 확인하세요.”
DataForma tConversi on.Invali dRecord	“레코드가 스키마와 다릅니다. 맵<스트링,문자열>에 대해 하나 이상의 맵 키가 유효하지 않습니다.”
DataForma tConversi on.Malfor medData	“입력 JSON의 최상위 레벨에 프리미티브가 포함되어 있습니다. 최상위 레벨은 객체 또는 배열이어야 합니다.”

오류 코드	오류 메시지와 정보
DataFormatConversion.MalformedData	“입력 JSON의 최상위 레벨에 프리미티브가 포함되어 있습니다. 최상위 레벨은 객체 또는 배열이어야 합니다.”
DataFormatConversion.MalformedData	“레코드가 비어 있거나 공백만 포함되어 있습니다.”
DataFormatConversion.MalformedData	“잘못된 문자입니다.”
DataFormatConversion.MalformedData	“유효하지 않거나 지원되지 않는 타임스탬프 형식입니다. 지원되는 타임스탬프 형식은 Firehose 개발자 가이드를 참조하세요.”
DataFormatConversion.MalformedData	“데이터에서 스칼라 유형을 찾았지만 스키마에 복합 유형이 지정되었습니다.”
DataFormatConversion.MalformedData	“데이터가 스키마와 일치하지 않습니다.”
DataFormatConversion.MalformedData	“데이터에서 스칼라 유형을 찾았지만 스키마에 복합 유형이 지정되었습니다.”

오류 코드	오류 메시지와 정보
DataFormatConversion.ConversionFailureException	"ConversionFailureException"
DataFormatConversion.DataFormatConversionCustomerErrorException	"DataFormatConversionCustomerErrorException"
DataFormatConversion.DataFormatConversionCustomerErrorException	"DataFormatConversionCustomerErrorException"
DataFormatConversion.MalformedData	“데이터가 스키마와 일치하지 않습니다.”
DataFormatConversion.InvalidSchema	“스키마가 유효하지 않습니다.”

오류 코드	오류 메시지와 정보
DataFormatConversion.MalformedData	“데이터가 스키마와 일치하지 않습니다. 하나 이상의 날짜의 형식이 잘못되었습니다.”
DataFormatConversion.MalformedData	“데이터에 지원되지 않으며 고도로 중첩된 JSON 구조가 포함되어 있습니다.”
DataFormatConversion.EntityNotFound	“지정된 테이블 또는 데이터베이스를 찾을 수 없습니다. 테이블/데이터베이스가 존재하고 스키마 구성에 제공된 값이 특히 대/소문자와 관련하여 올바른지 확인하세요.”
DataFormatConversion.InvalidInput	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 제공된 카탈로그 ID를 가진 지정된 데이터베이스가 존재하는지 확인하세요.”
DataFormatConversion.InvalidInput	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 전달된 ARN이 올바른 형식인지 확인하세요.”
DataFormatConversion.InvalidInput	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 제공된 CataloGID가 유효한지 확인하세요.”
DataFormatConversion.InvalidVersionId	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 지정된 버전의 테이블이 존재하는지 확인하세요.”

오류 코드	오류 메시지와 정보
DataFormatConversion.NonExistentColumns	“Glue에서 일치하는 스키마를 찾을 수 없습니다. 테이블이 대상 열을 포함한, null이 아닌 스토리지 설명자로 구성되어 있는지 확인하세요.”
DataFormatConversion.AccessDenied	“역할 수임 중 액세스가 거부되었습니다. 데이터 형식 변환 구성에 지정된 역할이 Firehose 서비스에 역할을 사용할 수 있는 권한을 부여했는지 확인하세요.”
DataFormatConversion.ThrottledByGlue	“Glue를 호출하는 동안 제한(Throttling)오류가 발생했습니다. 요청 비율 제한을 늘리거나 다른 애플리케이션을 통해 Glue를 호출하는 현재 비율을 낮추세요.”
DataFormatConversion.AccessDenied	“Glue를 호출하는 동안 액세스가 거부되었습니다. 데이터 형식 변환 구성에 지정된 역할에 필수 권한이 있는지 확인하세요.”
DataFormatConversion.InvalidGlueRole	“잘못된 역할입니다. 데이터 형식 변환 구성에 지정된 역할이 존재하는지 확인하세요.”
DataFormatConversion.GlueNotAvailableInRegion	“지정한 리전에서 아직AWS Glue를 사용할 수 없습니다. 다른 리전을 지정하세요.”
DataFormatConversion.GlueEncryptionException	“마스터 키를 검색하는 중 오류가 발생했습니다. 키가 존재하고 올바른 액세스 권한이 있는지 확인하세요.”

오류 코드	오류 메시지와 정보
`DataFormatConversion.SchemaValidationTimeout`	“Glue에서 테이블을 가져오는 중 시간이 초과되었습니다. Glue 테이블 버전이 많은 경우 'Glue:GettableVersion' 권한을 추가하거나 (권장) 사용하지 않는 테이블 버전을 삭제하세요. Glue에 테이블 수가 많지 않은 경우 AWS Support에 문의하세요.”
`DataFirehose.InternalError`	“Glue에서 테이블을 가져오는 중 시간이 초과되었습니다. Glue 테이블 버전이 많은 경우 'Glue:GettableVersion' 권한을 추가하거나 (권장) 사용하지 않는 테이블 버전을 삭제하세요. Glue에 테이블 수가 많지 않은 경우 AWS Support에 문의하세요.”
`DataFormatConversion.MalformedData`	“하나 이상의 필드 형식이 잘못되었습니다.”

Amazon Data Firehose에 대한 CloudWatch 로그 액세스

Amazon Data Firehose 콘솔이나 CloudWatch 콘솔을 이용해 Amazon Data Firehose 데이터 전송 실패와 관련된 오류 로그를 볼 수 있습니다. 다음의 절차는 이 두 가지 방법을 사용하여 오류 로그에 액세스하는 방법을 설명합니다.

Amazon Data Firehose 콘솔을 사용하여 오류 로그에 액세스하는 방법

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/firehose Firehose 콘솔을 엽니다.
2. 탐색 모음에서 AWS 리전을 선택합니다.
3. Firehose 스트림 이름을 선택하여 Firehose 스트림 세부 정보 페이지로 이동합니다.
4. [Error Log]를 선택해 데이터 전송 실패와 관련된 오류 로그 목록을 봅니다.

CloudWatch 콘솔을 사용하여 오류 로그에 액세스하는 방법

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 모음에서 리전을 선택합니다.

3. 탐색 창에서 로그를 선택합니다.
4. 로그 그룹과 로그 스트림을 선택해 데이터 전송 실패와 관련된 오류 로그 목록을 봅니다.

Kinesis 에이전트 상태 모니터링

Kinesis Agent는AWS KinesisAgent라는 네임스페이스를 사용하여 사용자 지정 CloudWatch 지표를 게시합니다. 이를 통해 에이전트가 정상인지, 지정된 대로 Amazon Data Firehose에 데이터를 제출하고 있으며 데이터 생산자에게 적절한 용량의 CPU와 메모리 리소스를 소비하고 있는지를 평가할 수 있습니다.

전송된 레코드 수와 바이트 등의 측정치는 에이전트가 Firehose 스트림에 데이터를 제출하는 속도를 이해하는 데 유용합니다. 이러한 측정치가 예상 임계값에 다소 못 미치거나 0으로 떨어지는 경우, 구성 문제, 네트워크 오류 또는 에이전트 상태 문제를 나타낼 수 있습니다. 호스트상의 CPU 및 메모리 소비 등의 측정치와 에이전트 오류 카운터는 데이터 생산자의 리소스 사용량을 나타내며, 잠재적인 구성 또는 호스트 오류에 대한 통찰을 제공합니다. 마지막으로 에이전트는 서비스 예외도 로깅하여 에이전트 문제를 조사하는 데 도움을 줍니다.

에이전트 측정치는 에이전트 구성 설정 `cloudwatch.endpoint`에 지정된 리전에서 보고됩니다. 자세한 내용은 에이전트 구성 설정 지정 섹션을 참조하세요.

여러 Kinesis Agent에서 게시된 Cloudwatch 지표를 집계 또는 결합합니다.

Kinesis Agent에서 내보낸 지표에는 일반 요금이 부과되며, 이는 기본적으로 활성화되어 있습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

CloudWatch를 사용하여 모니터링

Kinesis Agent는 CloudWatch에 다음 지표를 전송합니다.

지표	설명
`BytesSent`	지정된 시간 동안 Firehose 스트림에 전송된 바이트 수입니다. 단위: 바이트
`RecordSendAttempts`	지정된 기간 동안 `PutRecordBatch` 에 대한 호출에서 시도한 레코드 수입니다(처음 또는 다시 시도). 단위: 개

지표	설명
RecordSendErrors	지정한 기간 동안 재시도를 포함해 PutRecordBatch 에 대한 호출에서 실패 상태를 반환한 레코드 수입니다. 단위: 개
ServiceErrors	지정된 기간 동안 서비스 오류(조절 오류 제외)를 일으킨 PutRecordBatch 에 대한 호출 수입니다. 단위: 개

를 사용하여 Amazon Data Firehose API 호출 로깅 AWS CloudTrail

Amazon Data Firehose는 Amazon Data Firehose에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 Amazon Data Firehose에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Amazon Data Firehose 콘솔로부터의 호출과 Amazon Data Firehose API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 Amazon Data Firehose 이벤트를 비롯하여 CloudTrail 이벤트를 Amazon S3 버킷으로 지속적으로 배포하도록 할 수 있습니다. 추적을 구성하지 않은 경우에도 이벤트 기록에서 CloudTrail 콘솔의 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon Data Firehose에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

구성 및 활성화 방법을 포함하여 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용자 안내서를 참조하세요.

CloudTrail의 Firehose 정보

AWS 계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. Amazon Data Firehose에서 지원되는 이벤트 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 설명은 CloudTrail 이벤트 기록으로 이벤트 보기를 참조하세요.

Amazon Data Firehose에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 기본적으로 콘솔에서 추적을 생성하면 추적이 모든 AWS 리전에 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한

CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

Amazon Data Firehose는 CloudTrail 로그 파일에 다음 작업을 이벤트로 로깅하도록 지원합니다.

- CreateDeliveryStream
- DeleteDeliveryStream
- DescribeDeliveryStream
- ListDeliveryStreams
- ListTagsForDeliveryStream
- TagDeliveryStream
- StartDeliveryStreamEncryption
- StopDeliveryStreamEncryption
- UntagDeliveryStream
- UpdateDestination

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

예시: Firehose 로그 파일 항목

트레일이란 지정한 S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `CreateDeliveryStream`, `DescribeDeliveryStream`, `ListDeliveryStreams`, `UpdateDestination`, `DeleteDeliveryStream` 작업을 보여 주는 CloudTrail 로그 항목이 나타낸 예입니다.

```
{
  "Records":[
        {
            "eventVersion":"1.02",
            "userIdentity":{
                "type":"IAMUser",
                "principalId":"AKIAIOSFODNN7EXAMPLE",
                "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
                "accountId":"111122223333",
                "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
                "userName":"CloudTrail_Test_User"
            },
            "eventTime":"2016-02-24T18:08:22Z",
            "eventSource":"firehose.amazonaws.com",
            "eventName":"CreateDeliveryStream",
            "awsRegion":"us-east-1",
            "sourceIPAddress":"127.0.0.1",
            "userAgent":"aws-internal/3",
            "requestParameters":{
                "deliveryStreamName":"TestRedshiftStream",
                "redshiftDestinationConfiguration":{
                "s3Configuration":{
                    "compressionFormat":"GZIP",
                    "prefix":"prefix",
                    "bucketARN":"arn:aws:s3:::amzn-s3-demo-bucket",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
                    "encryptionConfiguration":{
```

```
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    }
                },
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "password":"",
                "username":"",
                "roleARN":"arn:aws:iam::111122223333:role/Firehose"
            }
        },
        "responseElements":{
            "deliveryStreamARN":"arn:aws:firehose:us-
east-1:111122223333:deliverystream/TestRedshiftStream"
        },
        "requestID":"958abf6a-db21-11e5-bb88-91ae9617edf5",
        "eventID":"875d2d68-476c-4ad5-bbc6-d02872cfc884",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:08:54Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DescribeDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
```

```
        "responseElements":null,
        "requestID":"aa6ea5ed-db21-11e5-bb88-91ae9617edf5",
        "eventID":"d9b285d8-d690-4d5c-b9fe-d1ad5ab03f14",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:00Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"ListDeliveryStreams",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "limit":10
        },
        "responseElements":null,
        "requestID":"d1bf7f86-db21-11e5-bb88-91ae9617edf5",
        "eventID":"67f63c74-4335-48c0-9004-4ba35ce00128",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:09Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"UpdateDestination",
```

```
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "destinationId":"destinationId-000000000001",
            "deliveryStreamName":"TestRedshiftStream",
            "currentDeliveryStreamVersionId":"1",
            "redshiftDestinationUpdate":{
                "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "password":"",
                "username":"",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "s3Update":{
                    "bucketARN":"arn:aws:s3:::amzn-s3-demo-bucket-update",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "compressionFormat":"GZIP",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    },
                    "prefix":"arn:aws:s3:::amzn-s3-demo-bucket"
                }
            }
        },
        "responseElements":null,
        "requestID":"d549428d-db21-11e5-bb88-91ae9617edf5",
        "eventID":"1cb21e0b-416a-415d-bbf9-769b152a6585",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
```

```
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:12Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DeleteDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"d85968c1-db21-11e5-bb88-91ae9617edf5",
        "eventID":"dd46bb98-b4e9-42ff-a6af-32d57e636ad1",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    }
  ]
}
```

AWS SDKs를 사용한 Firehose 코드 예제

다음 코드 예제에서는 Firehose를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Firehose 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDKs를 사용한 Firehose의 기본 예제
  - AWS SDKs를 사용한 Firehose 작업
    - AWS SDK 또는 CLI와 PutRecord 함께 사용
    - AWS SDK 또는 CLI와 PutRecordBatch 함께 사용
- AWS SDKs를 사용한 Firehose 시나리오
  - Amazon Data Firehose를 사용하여 개별 레코드 및 배치 레코드 처리

AWS SDKs를 사용한 Firehose의 기본 예제

다음 코드 예시는 AWS SDK와 Amazon Data Firehose를 함께 사용하는 방법에 대한 기본적인 내용을 보여줍니다.

예제

- AWS SDKs를 사용한 Firehose 작업
  - AWS SDK 또는 CLI와 PutRecord 함께 사용
  - AWS SDK 또는 CLI와 PutRecordBatch 함께 사용

AWS SDKs를 사용한 Firehose 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 Firehose 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이들 발췌문은 Firehose API를 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDKs를 사용한 Firehose 시나리오에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon Data Firehose API Reference(Amazon Data Firehose API 참조)를 참조하세요.

예제

- AWS SDK 또는 CLI와 PutRecord 함께 사용
- AWS SDK 또는 CLI와 PutRecordBatch 함께 사용

AWS SDK 또는 CLI와 **PutRecord** 함께 사용

다음 코드 예시는 `PutRecord`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Firehose에 레코드 넣기

CLI

AWS CLI

스트림에 레코드를 쓰는 방법

다음 `put-record` 예제에서는 스트림에 데이터를 씁니다. 데이터는 Base64 형식으로 인코딩됩니다.

```
aws firehose put-record \
    --delivery-stream-name my-stream \
    --record '{"Data":"SGVsbG8gd29ybGQ="}'
```

출력:

```
{
    "RecordId": "RjB5K/nnoGFHqwTsZlNd/
TTqvjE8V5dsyXZTQn2JXrdpMTOwssyEb6nfC8fwf1whhwnItt4mvrn+gsqeK5jB7QjuLg283+Ps4Sz/
j1Xujv31iDhnPdaLw4BOyM9Amv7PcCuB2079RuM0NhoakbyUymlwY8yt20G8X2420wu1jlFafhci4erAt7QhDEvpw
    "Encrypted": false
}
```

자세한 내용은 Amazon Kinesis Data Firehose 개발자 안내서의 Amazon Kinesis Data Firehose 전송 스트림으로 데이터 전송을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutRecord를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
```

```
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutRecord를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
```

```
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 PutRecord를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    DATA(lo_record) = NEW /aws1/cl_frhrecord( iv_data = iv_data ).

    DATA(lo_result) = lo_frh->putrecord(
      iv_deliverystreamname = iv_deliv_stream_name
      io_record             = lo_record ).

    MESSAGE 'Record sent to Firehose delivery stream.' TYPE 'I'.
  CATCH /aws1/cx_frhresourcenotfoundex.
    MESSAGE 'Delivery stream not found.' TYPE 'E'.
  CATCH /aws1/cx_frhinvalidargumentex.
    MESSAGE 'Invalid argument provided.' TYPE 'E'.
  CATCH /aws1/cx_frhserviceunavailex.
    MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
ENDTRY.
```

- API에 대한 세부 정보는 AWS SDK for SAP ABAP API 참조의 PutRecord를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Firehose 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **PutRecordBatch** 함께 사용

다음 코드 예시는 PutRecordBatch의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- Firehose에 레코드 넣기

CLI

AWS CLI

스트림에 여러 레코드를 쓰는 방법

다음 put-record-batch 예시에서는 하나의 스트림에 3개의 레코드를 씁니다. 데이터는 Base64 형식으로 인코딩됩니다.

```
aws firehose put-record-batch \
    --delivery-stream-name my-stream \
    --records file://records.json
```

myfile.json의 콘텐츠:

```
[
    {"Data": "Rmlyc3QgdGhpbmc="},
    {"Data": "U2Vjb25kIHRoaW5n"},
    {"Data": "VGhpcmQgdGhpbmc="}
]
```

출력:

```
{
    "FailedPutCount": 0,
    "Encrypted": false,
    "RequestResponses": [
        {
            "RecordId": "9D2OJ6t2EqCTZTXwGzeSv/EVHxRoRCw89xd+o3+sXg8DhYOaWKPSmZy/
CGlRVEys1u1xbeKh6VofEYKkoeiDrcjrxhQp9iF7sUW7pujiMEQ5LzlrzCkGosxQn
+3boDnURDEaD42V7GiixpOyLJkYZcae1i7HzlCEoy9LJhMr8EjDSi4Om/9Vc2uhwwuAtGE0XKpxJ2WD7ZRWtAnYlK
        },
        {
            "RecordId": "jFirejqxCLlK5xjH/UNmlMVcjktEN76I7916X9PaZ
+PVaOSXDfU1WGOqEZhxq2js7xcZ552eoeDxsuTU1MSq9nZTbVfb6cQTIXnm/
GsuF37Uhg67GKmR5z90l6XKJ+/
```

```
+pDloFv7Hh9a3oUS6wYm3DcNRLTHHAimANp1PhkQvWpvLRfzbuCUkBphR2QVzhP9OiHLbzGwy8/
DfH8sqWEUYASNJKS8GXP5s"
        },
        {
            "RecordId":
 "oy0amQ40o5Y2YV4vxzufdcMOOw6n3EPr3tpPJGoYVNKH4APPVqNcbUgefo1stEFRg4hTLrf2k6eliHu/9+YJ5R3
DTBt3qBlmTj7Xq8SKVbO1S7YvMTpWkMKA86f8JfmT8BMKoMb4XZS/sOkQLe+qh0sYKXWl"
        }
    ]
}
```

자세한 내용은 Amazon Kinesis Data Firehose 개발자 안내서의 Amazon Kinesis Data Firehose 전송 스트림으로 데이터 전송을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutRecordBatch를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param records           a list of maps representing the records to be
sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
```

```
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord =
objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose
record", e);
                   }
               }).collect(Collectors.toList());

               PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                   .deliveryStreamName(deliveryStreamName)
                   .records(batchRecords)
                   .build();

               PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

               if (response.failedPutCount() > 0) {
                   response.requestResponses().stream()
                       .filter(r -> r.errorCode() != null)
                       .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
               }
               System.out.println("Batch sent with size: " +
batchRecords.size());
```

```
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
 e.getMessage(), e);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutRecordBatch를 참조하세요.

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
```

```
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 PutRecordBatch를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn put_record_batch(
    client: &Client,
    stream: &str,
    data: Vec<Record>,
) -> Result<PutRecordBatchOutput, SdkError<PutRecordBatchError>> {
    client
        .put_record_batch()
        .delivery_stream_name(stream)
        .set_records(Some(data))
        .send()
        .await
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 PutRecordBatch를 참조하세요.

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        DATA(lo_result) = lo_frh->putrecordbatch(
          iv_deliverystreamname = iv_deliv_stream_name
          it_records            = it_records ).
```

```
        DATA(lv_failed_count) = lo_result->get_failedputcount( ).

        IF lv_failed_count > 0.
          MESSAGE |{ lv_failed_count } records failed to send.| TYPE 'I'.
        ELSE.
          MESSAGE 'All records sent successfully to Firehose delivery stream.'
TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_frhresourcenotfoundex.
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
      CATCH /aws1/cx_frhinvalidargumentex.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
      CATCH /aws1/cx_frhserviceunavailex.
        MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 PutRecordBatch를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Firehose 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs를 사용한 Firehose 시나리오

다음 코드 예제에서는 Firehose AWS SDKs에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이 시나리오에서는 Firehose 내에서 여러 함수를 호출하거나 다른 AWS 서비스와 결합하여 특정 작업을 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정 및 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- Amazon Data Firehose를 사용하여 개별 레코드 및 배치 레코드 처리

Amazon Data Firehose를 사용하여 개별 레코드 및 배치 레코드 처리

다음 코드 예제에서는 Firehose를 사용하여 개별 레코드 및 배치 레코드를 처리하는 방법을 보여줍니다.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예제는 개별 레코드 및 배치 레코드를 Firehose에 보냅니다.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
```

```
            List<Map<String, Object>> sampleData =
objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
```

```
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
     * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
     * @throws RuntimeException if there is an error putting the record to the
delivery stream.
     */
    public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
        if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

```
   /**
    * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param records           a list of maps representing the records to be
sent
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord =
objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose
record", e);
                   }
               }).collect(Collectors.toList());
```

```
                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime,
Instant endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)

.dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).bui
                .startTime(startTime)
                .endTime(endTime)
```

```
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
 getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
 response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
 e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
 FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8))
 {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - PutRecord
  - PutRecordBatch

Python

SDK for Python(Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 스크립트는 개별 레코드 및 배치 레코드를 Firehose에 넣습니다.

```
import json
import logging
import random
from datetime import datetime, timedelta

import backoff
import boto3

from config import get_config


def load_sample_data(path: str) -> dict:
    """
    Load sample data from a JSON file.

    Args:
        path (str): The file path to the JSON file containing sample data.

    Returns:
        dict: The loaded sample data as a dictionary.
    """
    with open(path, "r") as f:
        return json.load(f)


# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class FirehoseClient:
```

```
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
```

```
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")


    def get_metric_statistics(
        self,
        metric_name: str,
        start_time: datetime,
        end_time: datetime,
        period: int,
```

```
        statistics: list = ["Sum"],
    ) -> list:
        """
        Retrieve metric statistics from CloudWatch.

        Args:
            metric_name (str): The name of the metric.
            start_time (datetime): The start time for the metric statistics.
            end_time (datetime): The end time for the metric statistics.
            period (int): The granularity, in seconds, of the returned data
points.
            statistics (list): A list of statistics to retrieve. Default is
['Sum'].

        Returns:
            list: List of datapoints containing the metric statistics.
        """
        response = self.cloudwatch.get_metric_statistics(
            Namespace="AWS/Firehose",
            MetricName=metric_name,
            Dimensions=[
                {"Name": "DeliveryStreamName", "Value":
self.delivery_stream_name},
            ],
            StartTime=start_time,
            EndTime=end_time,
            Period=period,
            Statistics=statistics,
        )
        return response["Datapoints"]

    def monitor_metrics(self):
        """
        Monitor Firehose metrics for the last 5 minutes.

        This method retrieves and logs the 'IncomingBytes', 'IncomingRecords',
and 'FailedPutCount' metrics
        from CloudWatch for the last 5 minutes.
        """
        end_time = datetime.utcnow()
        start_time = end_time - timedelta(minutes=10)
        period = int((end_time - start_time).total_seconds())

        metrics = {
```

```
            "IncomingBytes": self.get_metric_statistics(
                "IncomingBytes", start_time, end_time, period
            ),
            "IncomingRecords": self.get_metric_statistics(
                "IncomingRecords", start_time, end_time, period
            ),
            "FailedPutCount": self.get_metric_statistics(
                "FailedPutCount", start_time, end_time, period
            ),
        }

        for metric, datapoints in metrics.items():
            if datapoints:
                total_sum = sum(datapoint["Sum"] for datapoint in datapoints)
                if metric == "IncomingBytes":
                    logger.info(
                        f"{metric}: {round(total_sum)} ({total_sum / (1024 *
1024):.2f} MB)"
                    )
                else:
                    logger.info(f"{metric}: {round(total_sum)}")
            else:
                logger.info(f"No data found for {metric} over the last 5
minutes")


    def _create_record_entry(self, record: dict) -> dict:
        """
        Create a record entry for Firehose.

        Args:
            record (dict): The data record to be sent.

        Returns:
            dict: The record entry formatted for Firehose.

        Raises:
            Exception: If a simulated network error occurs.
        """
        if random.random() < 0.2:
            raise Exception("Simulated network error")
        elif random.random() < 0.1:
            return {"Data": '{"malformed": "data"'}
        else:
```

```
            return {"Data": json.dumps(record)}

    def _log_response(self, response: dict, entry: dict):
        """
        Log the response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record API call.
            entry (dict): The record entry that was sent.
        """
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            logger.info(f"Sent record: {entry}")
        else:
            logger.info(f"Fail record: {entry}")

    def _log_batch_response(self, response: dict, batch_size: int):
        """
        Log the batch response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record_batch API
 call.
            batch_size (int): The number of records in the batch.
        """
        if response.get("FailedPutCount", 0) > 0:
            logger.info(
                f'Failed to send {response["FailedPutCount"]} records in batch of
 {batch_size}'
            )
        else:
            logger.info(f"Successfully sent batch of {batch_size} records")


if __name__ == "__main__":
    config = get_config()
    data = load_sample_data(config.sample_data_file)
    client = FirehoseClient(config)

    # Process the first 100 sample network records
    for record in data[:100]:
        try:
            client.put_record(record)
        except Exception as e:
            logger.info(f"Put record failed after retries and backoff: {e}")
```

```
    client.monitor_metrics()

    # Process remaining records using the batch method
    try:
        client.put_record_batch(data[100:])
    except Exception as e:
        logger.info(f"Put record batch failed after retries and backoff: {e}")
    client.monitor_metrics()
```

이 파일에는 위 스크립트에 대한 구성이 포함되어 있습니다.

```
class Config:
    def __init__(self):
        self.delivery_stream_name = "ENTER YOUR DELIVERY STREAM NAME HERE"
        self.region = "us-east-1"
        self.sample_data_file = (
            "../../../../../scenarios/features/firehose/resources/
sample_records.json"
        )


def get_config():
    return Config()
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하세요.
  - PutRecord
  - PutRecordBatch

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서 Firehose 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Data Firehose 오류 해결

Firehose가 데이터를 전송하거나 처리하는 동안 오류가 발생하면 구성된 재시도 기간이 만료될 때까지 다시 시도합니다. 데이터가 성공적으로 전송되기 전에 재시도 기간이 끝나면 Firehose 는 데이터를 구성된 S3 백업 버킷에 백업합니다. 대상이 Amazon S3이고 전송에 실패하거나 백업 S3 버킷으로 전송하는 데 실패하는 경우, Firehose는 보존 기간이 끝날 때까지 계속 재시도합니다.

CloudWatch를 사용하여 전송 오류를 추적하는 방법에 대한 자세한 내용은 the section called "CloudWatch 로그를 사용한 모니터링"을 참조하세요.

Direct PUT

DirectPut Firehose 스트림의 경우 Firehose는 24시간 동안 레코드를 보존합니다. 데이터 소스가 Kinesis 데이터 스트림인 Firehose 스트림의 경우 데이터 보존 기간 변경에 설명된 대로 보존 기간을 변경할 수 있습니다. 이 경우 Firehose는 DescribeStream, GetRecords, GetShardIterator 작업을 무기한 재시도합니다.

Firehose 스트림에서 DirectPut을 사용하는 경우, IncomingBytes 및 IncomingRecords 지표를 보고 들어오는 트래픽이 있는지 확인합니다. PutRecord 또는 PutRecordBatch를 사용하는 경우, 예외를 포착하고 다시 시도하세요. 지수 백오프와 지터 및 여러 번의 재시도가 포함된 재시도 정책을 사용하는 것이 좋습니다. 또한 PutRecordBatch API를 사용하는 경우, API 호출이 성공하더라도 코드가 응답에서 FailedPutCount 값을 검사하도록 해야 합니다.

Kinesis Data Stream

Firehose 스트림이 Kinesis 데이터 스트림을 소스로 사용하는 경우, 소스 데이터 스트림에 대한 IncomingBytes 및 IncomingRecords 지표를 확인합니다. 또한 Firehose 스트림에 대해 DataReadFromKinesisStream.Bytes 및 DataReadFromKinesisStream.Records 지표가 전송되고 있는지도 확인합니다.

일반적인 문제

다음은 Firehose 스트림으로 작업하는 동안 발생할 수 있는 일반적인 문제를 해결하는 데 도움이 되는 문제 해결 팁입니다.

Firehose 스트림 사용 불가

일부 서비스는 동일한에 있는 Firehose 스트림에만 메시지와 이벤트를 보낼 수 있으므로 AWS Firehose 스트림을 CloudWatch Logs, CloudWatch Events 또는 AWS IoT 작업의 대상으로 사용할 수 없습니다 AWS 리전. Firehose 스트림이 다른 서비스와 동일한 리전에 있는지 확인하세요.

대상에 데이터 없음

데이터 수집 문제가 없고 Firehose 스트림에 대해 방출된 지표가 양호해 보이는데도 대상에 데이터가 표시되지 않는 경우, 독자 로직을 확인합니다. 독자가 모든 데이터를 올바르게 구문 분석하고 있는지 확인하세요.

데이터 신선도 지표 증가 또는 미방출

데이터 신선도란 Firehose 스트림 내 데이터의 최신성을 측정하는 것입니다. 이는 Firehose 스트림에서 가장 오래된 데이터 레코드의 수명으로, Firehose가 데이터를 수집한 시점부터 현재까지의 시간으로 측정됩니다. Firehose는 데이터 신선도를 모니터링하는 데 사용할 수 있는 지표를 제공합니다. 지정된 대상에 대한 데이터 신선도 지표를 확인하는 방법은 the section called “CloudWatch 지표를 사용한 모니터링” 섹션을 참조하세요.

모든 이벤트 또는 모든 문서에 대해 백업을 활성화한 경우, 개별 데이터 신선도 지표 두 가지를 모니터링해야 합니다. 하나는 기본 대상용이고, 다른 하나는 백업용입니다.

데이터 신선도 지표가 방출되지 않는 것은 해당 Firehose 스트림의 활성 전송이 없음을 의미합니다. 데이터 전송이 완전히 차단되거나 들어오는 데이터가 없을 때 이러한 현상이 발생합니다.

데이터 신선도 지표가 지속적으로 증가하는 것은 데이터 전송이 뒤쳐지고 있음을 의미합니다. 이것은 다음과 같은 이유 중 하나로 발생할 수 있습니다.

- 대상에서 그 전송 속도를 처리할 수 없습니다. 트래픽이 많아 Firehose 에 일시적인 오류가 발생하면 전송이 지연될 수 있습니다. 이 문제는 Amazon S3 이외의 대상에 발생할 수 있습니다(OpenSearch Service, Amazon Redshift 또는 Splunk에서 발생 가능). 대상의 용량이 들어오는 트래픽을 처리하기에 충분한지 확인합니다.
- 대상이 느립니다. Firehose의 지연 시간이 길면 데이터 전송이 지연될 수 있습니다. 대상의 대기 시간 지표를 모니터링합니다.
- Lambda 함수가 느립니다. 이로 인해 Firehose 스트림의 데이터 수집 속도보다 데이터 전송 속도가 낮아질 수 있습니다. 가능하면 Lambda 함수의 효율성을 높이세요. 예를 들어 함수가 네트워크 IO를 수행하는 경우, 다중 스레드 또는 비동기 IO를 사용하여 병렬 처리를 늘립니다. 또한 CPU 할당

량이 그에 따라 늘어날 수 있도록 Lambda 함수의 메모리 크기를 늘리는 것도 고려하세요. 그러면 Lambda 호출이 더 빨라질 수 있습니다. Lambda 함수 구성에 대한 자세한 내용은 AWS Lambda 함수 구성을 참조하세요.

- 데이터 전송 중에 오류가 발생했습니다. Amazon CloudWatch Logs를 사용하여 오류를 모니터링하는 방법에 대한 자세한 내용은 the section called “CloudWatch 로그를 사용한 모니터링”을 참조하세요.
- Firehose 스트림의 데이터 소스가 Kinesis 데이터 스트림인 경우 스로틀링이 발생할 수 있습니다. `ThrottledGetRecords`, `ThrottledGetShardIterator`, `ThrottledDescribeStream` 지표를 확인하세요. Kinesis 데이터 스트림에 연결된 소비자가 여럿인 경우 다음을 고려하세요.
  - `ThrottledGetRecords` 및 `ThrottledGetShardIterator` 지표가 높으면 데이터 스트림에 대해 프로비저닝된 샤드 수를 늘리는 것이 좋습니다.
  - `ThrottledDescribeStream`가 높으면 KinesisStreamSourceConfiguration에서 구성된 역할에 `kinesis:listshards` 권한을 추가하는 것이 좋습니다.
- 대상에 대한 낮은 버퍼링 힌트입니다. 이로 인해 Firehose가 대상까지 이동해야 하는 왕복 횟수가 증가하여 전송이 지연될 수 있습니다. 버퍼링 힌트의 값을 늘리는 것이 좋습니다. 자세한 내용은 BufferingHints를 참조하세요.
- 재시도 기간이 길면 오류가 자주 발생할 때 전송이 지연될 수 있습니다. 재시도 기간을 줄이는 것이 좋습니다. 이와 함께 오류를 모니터링하고 오류를 줄이세요. Amazon CloudWatch Logs를 사용하여 오류를 모니터링하는 방법에 대한 자세한 내용은 the section called “CloudWatch 로그를 사용한 모니터링”을 참조하세요.
- 대상이 Splunk이고 `DeliveryToSplunk.DataFreshness`가 높지만 `DeliveryToSplunk.Success`는 양호해 보이는 경우, Splunk 클러스터가 사용 중일 수 있습니다. 가능하면 Splunk 클러스터를 비우세요. 또는 AWS Support에 문의하여 Firehose가 Splunk 클러스터와 통신하는 데 사용하는 채널 수를 늘려 달라고 요청하세요.

Apache Parquet으로의 레코드 형식 변환 실패

이는 `Set` 유형이 포함된 DynamoDB 데이터를 가져와 Lambda를 통해 Firehose 스트림으로 스트리밍하고를 사용하여 레코드 형식을 Apache Parquet으로 변환 AWS Glue Data Catalog 하는 경우에 발생합니다.

AWS Glue 크롤러가 DynamoDB 세트 데이터 형식(`StringSet`, `NumberSet`및 `BinarySet`)을 인덱싱하면 데이터 카탈로그에 `SET<BINARY>`각각 `SET<STRING>`, `SET<BIGINT>`및 로 저장됩니다. 하지만 Firehose에서 데이터 레코드를 Apache Parquet 형식으로 변환하려면 Apache Hive 데이터 유형이 필요합니다. 이 세트 유형은 유효한 Apache Hive 데이터 유형이 아니기 때문에 변환이 실패합니다. 변

환이 수행되도록 하려면 데이터 카탈로그를 Apache Hive 데이터 유형으로 업데이트합니다. 데이터 카탈로그에서 `set`를 `array`로 변경하여 이 작업을 수행할 수 있습니다.

데이터 카탈로그**`setarray`**에서 하나 이상의 AWS Glue 데이터 형식을에서 로 변경하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/glue/ AWS Glue 콘솔을 엽니다.
2. 왼쪽 창에 있는 데이터 카탈로그 머리글에서 테이블을 선택합니다.
3. 테이블 목록에서 하나 이상의 데이터 유형을 수정해야 하는 테이블의 이름을 선택합니다. 그러면 해당 테이블에 대한 세부 정보 페이지로 이동합니다.
4. 세부 정보 페이지의 상단 오른쪽 모서리에 있는 Edit schema(스키마 편집) 버튼을 선택합니다.
5. 데이터 유형 열에서 첫 번째 `set` 데이터 유형을 선택합니다.
6. 열 유형 드롭다운 목록에서 유형을 `set`에서 `array`로 변경합니다.
7. ArraySchema 필드에서 사용자 시나리오에 적합한 데이터 유형에 따라 `array<string>`, `array<int>` 또는 `array<binary>`를 입력합니다.
8. 업데이트를 선택합니다.
9. 기타 `set` 유형을 `array` 유형으로 변환하려면 이전 단계를 반복합니다.
10. 저장을 선택합니다.

Lambda로 변환한 객체의 필드 누락

Lambda 데이터 변환을 사용하여 JSON 데이터를 Parquet 객체로 변경하는 경우, 변환한 후에 일부 필드가 누락될 수 있습니다. JSON 객체에 대문자가 있고 대/소문자 구분이 `false`로 설정되어 있는 경우 데이터 변환 후 JSON 키가 불일치하여 s3 버킷의 결과 Parquet 객체에 누락된 데이터가 발생할 수 있습니다.

이 문제를 해결하려면 호스 구성에서 `deserializationOption: case.insensitive`가 `true`로 설정되어 있는지 확인하여 변환 후 JSON 키가 일치하도록 합니다.

Amazon S3 문제 해결

Amazon Simple Storage Service(Amazon S3) 버킷으로 데이터가 전송되지 않는 경우 다음 사항을 확인하세요.

- Firehose `IncomingBytes` 및 `IncomingRecords` 지표를 확인하여 데이터가 Firehose 스트림으로 성공적으로 전송되는지 확인하세요. 자세한 내용은 CloudWatch 지표를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- Lambda를 이용한 데이터 변환 사용이 설정된 경우 Firehose `ExecuteProcessingSuccess` 지표를 확인하여 Firehose가 Lambda 함수 호출을 시도했는지 확인하세요. 자세한 내용은 CloudWatch 지표를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- Firehose `DeliveryToS3.Success` 지표를 확인하여 Firehose가 Amazon S3 버킷에 데이터를 넣으려고 시도했는지 확인하세요. 자세한 내용은 CloudWatch 지표를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- 오류 로깅이 아직 활성화되지 않은 경우 활성화하고, 오류 로그에서 전송 실패 여부를 확인합니다. 자세한 내용은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- 로그에 “Firehose가 Amazon S3 서비스를 직접 호출하는 중 InternalServerError가 발생했습니다. 작업을 재시도합니다. 오류가 지속되면 S3에 문의하여 해결하세요.”라는 오류 메시지가 표시되는 경우 S3의 단일 파티션에서 요청 속도가 크게 증가했기 때문일 수 있습니다. S3 접두사 설계 패턴을 최적화하여 이 문제를 완화할 수 있습니다. 자세한 내용은 모범 사례 설계 패턴: Amazon S3 성능 최적화를 참조하세요. 이렇게 해도 문제가 해결되지 않으면 AWS Support에 문의하여 추가 지원을 받으세요.
- Firehose 스트림에 지정된 Amazon S3 버킷이 아직 존재하는지 확인합니다.
- Lambda를 이용한 데이터 변환 사용이 설정된 경우 Firehose 스트림에 지정된 Lambda 함수가 아직 존재하는지 확인합니다.
- Firehose 스트림에 지정된 IAM 역할이 S3 버킷 및 Lambda 함수(데이터 변환 기능이 활성화된 경우)에 액세스할 수 있는지 확인합니다. 또한 오류 로그를 확인할 수 있는 CloudWatch 로그 그룹 및 로그 스트림에 대한 액세스 권한이 IAM 역할에 있는지 확인합니다. 자세한 내용은 Firehose에 Amazon S3 대상에 대한 액세스 권한 부여 섹션을 참조하세요.
- 데이터 변환을 사용하는 경우 Lambda 함수가 절대로 페이로드 크기가 6MB를 초과하는 응답을 반환하지 않게 해야 합니다. 자세한 정보는 Amazon Data Firehose 데이터 변환을 참조하세요.

Amazon Redshift 문제 해결

Amazon Redshift 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹에 데이터가 전송되지 않은 경우 다음 내용을 확인해야 합니다.

데이터는 Amazon Redshift로 로드되기 전에 S3 버킷으로 먼저 전송됩니다. 데이터가 S3 버킷으로 전송되지 않은 경우 Amazon S3 문제 해결 섹션을 참조하세요.

- Firehose `DeliveryToRedshift.Success` 지표를 확인하여 Firehose가 S3 버킷에서 Amazon Redshift 프로비저닝 클러스터 또는 Amazon Redshift Serverless 작업 그룹으로 데이터를 복사하려고 시도했는지 확인하세요. 자세한 내용은 CloudWatch 지표를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- 오류 로깅이 아직 활성화되지 않은 경우 활성화하고, 오류 로그에서 전송 실패 여부를 확인합니다. 자세한 내용은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- Amazon Redshift `STL_CONNECTION_LOG` 테이블을 확인하여 Firehose가 성공적으로 연결할 수 있는지 확인하세요. 이 테이블에서 사용자 이름을 이용해 연결과 연결 상태를 확인할 수 있어야 합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 `STL_CONNECTION_LOG`을 참조하세요.
- 앞서 확인한 결과 연결이 설정되고 있는 중이라면 Amazon Redshift `STL_LOAD_ERRORS` 테이블을 확인하여 COPY 명령이 실패한 이유를 살펴봅니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 `STL_LOAD_ERRORS`을 참조하세요.
- Firehose 스트림의 Amazon Redshift 구성이 정확하고 유효한지 확인합니다.
- Firehose 스트림에 지정된 IAM 역할이 Amazon Redshift가 데이터를 복사해 오는 S3 버킷과, 데이터 변환을 위한 Lambda 함수(데이터 변환이 활성화된 경우)에 액세스할 수 있는지 확인합니다. 또한 오류 로그를 확인할 수 있는 CloudWatch 로그 그룹 및 로그 스트림에 대한 액세스 권한이 IAM 역할에 있는지 확인합니다. 자세한 내용은 Firehose에 Amazon Redshift 대상에 대한 액세스 권한 부여 섹션을 참조하세요.
- Amazon Redshift 프로비저닝 클러스터 또는 Amazon Redshift Serverless 작업 그룹이 가상 사설 클라우드(VPC) 에 있는 경우 클러스터가 Firehose IP 주소에서의 액세스를 허용하는지 확인하세요. 자세한 내용은 Firehose에 Amazon Redshift 대상에 대한 액세스 권한 부여 섹션을 참조하세요.
- Amazon Redshift 프로비저닝된 클러스터 또는 Amazon Redshift Serverless 작업 그룹이 공개적으로 사용 가능한지 확인해야 합니다.
- 데이터 변환을 사용하는 경우 Lambda 함수가 절대로 페이로드 크기가 6MB를 초과하는 응답을 반환하지 않게 해야 합니다. 자세한 정보는 Amazon Data Firehose 데이터 변환을 참조하세요.

Amazon OpenSearch Service 문제 해결

OpenSearch Service 도메인으로 데이터가 전송되지 않는 경우 다음 사항을 확인하세요.

데이터는 동시에 Amazon S3 버킷으로 백업될 수 있습니다. 데이터가 S3 버킷으로 전송되지 않은 경우 Amazon S3 문제 해결 섹션을 참조하세요.

- Firehose `IncomingBytes` 및 `IncomingRecords` 지표를 확인하여 데이터가 Firehose 스트림으로 성공적으로 전송되는지 확인하세요. 자세한 내용은 CloudWatch 지표를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- Lambda를 이용한 데이터 변환 사용이 설정된 경우 Firehose `ExecuteProcessingSuccess` 지표를 확인하여 Firehose가 Lambda 함수 호출을 시도했는지 확인하세요. 자세한 내용은 CloudWatch 지표를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- Firehose `DeliveryToAmazonOpenSearchService.Success` 지표를 확인하여 Firehose가 OpenSearch 서비스 클러스터에 대한 데이터 인덱싱을 시도했는지 확인하세요. 자세한 내용은 CloudWatch 지표를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- 오류 로깅이 아직 활성화되지 않은 경우 활성화하고, 오류 로그에서 전송 실패 여부를 확인합니다. 자세한 내용은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.
- Firehose 스트림의 OpenSearch Service 구성이 정확하고 유효한지 확인합니다.
- Lambda를 이용한 데이터 변환 사용이 설정된 경우 Firehose 스트림에 지정된 Lambda 함수가 아직 존재하는지 확인합니다. 또한 오류 로그를 확인할 수 있는 CloudWatch 로그 그룹 및 로그 스트림에 대한 액세스 권한이 IAM 역할에 있는지 확인합니다. 자세한 정보는 Firehose에 퍼블릭 OpenSearch Service 대상에 대한 액세스 권한 부여를 참조하세요.
- Firehose 스트림에 지정된 IAM 역할이 OpenSearch Service 클러스터, S3 백업 버킷, Lambda 함수(데이터 변환이 활성화된 경우)에 액세스할 수 있는지 확인합니다. 또한 오류 로그를 확인할 수 있는 CloudWatch 로그 그룹 및 로그 스트림에 대한 액세스 권한이 IAM 역할에 있는지 확인합니다. 자세한 내용은 Firehose에 퍼블릭 OpenSearch Service 대상에 대한 액세스 권한 부여 단원을 참조하십시오.
- 데이터 변환을 사용하는 경우 Lambda 함수가 절대로 페이로드 크기가 6MB를 초과하는 응답을 반환하지 않게 해야 합니다. 자세한 정보는 Amazon Data Firehose 데이터 변환을 참조하세요.
- Amazon Data Firehose는 현재 Amazon OpenSearch Service 대상으로 CloudWatch Logs를 전송하는 것을 지원하지 않습니다. Amazon CloudWatch는 여러 로그 이벤트를 하나의 Firehose 레코드로 결합하고 Amazon OpenSearch Service는 하나의 레코드로 된 여러 로그 이벤트를 받아들일 수 없기 때문입니다. 대신 CloudWatch Logs에서 Amazon OpenSearch Service에 대한 구독 필터 사용을 고려할 수 있습니다.

Splunk 문제 해결

Splunk 엔드포인트로 데이터가 전송되지 않는 경우, 다음 사항을 확인하세요.

- Splunk 플랫폼이 VPC에 있는 경우 Firehose가 해당 플랫폼에 액세스할 수 있어야 합니다. 자세한 내용은 VPC에서 Splunk에 액세스 단원을 참조하십시오.

- AWS 로드 밸런서를 사용하는 경우 Classic Load Balancer 또는 Application Load Balancer인지 확인합니다. 또한 Classic Load Balancer의 경우 쿠키 만료를 비활성화한 기간 기반 스티키 세션을 사용 설정하고, Application Load Balancer의 경우 만료 기간을 최댓값(7일)으로 설정해야 합니다. 이 작업을 수행하는 방법에 대한 자세한 정보는 Classic Load Balancer 또는 Application Load Balancer의 기간 기반 세션 고정을 참조하세요.
- Splunk 플랫폼 요구사항을 검토합니다. Firehose용 Splunk 추가 기능을 사용하려면 Splunk 플랫폼 버전 6.6.X 이상이 필요합니다. 자세한 내용은 Amazon Kinesis Firehose용 Splunk 추가 기능을 참조하세요.
- Firehose 노드와 HEC(HTTP Event Collector) 노드 사이에 프록시(Elastic Load Balancing 또는 기타)가 있는 경우, HEC ACK(acknowledgement)를 지원하려면 고정 세션을 활성화해야 합니다.
- 사용 중인 HEC 토큰이 유효한지 확인합니다.
- HEC 토큰이 활성화되었는지 확인합니다.
- Splunk로 전송하는 데이터가 올바른 형식으로 지정되어 있는지 확인하세요. 자세한 내용은 HTTP Event Collector에 대한 이벤트 형식 지정을 참조하세요.
- HEC 토큰과 입력 이벤트가 유효한 인덱스로 구성되어 있는지 확인합니다.
- HEC 노드에서의 서버 오류 때문에 Splunk로의 업로드에 실패할 때는 요청을 자동으로 재시도하게 됩니다. 모든 재시도에 실패할 경우에는 데이터가 Amazon S3에 백업됩니다. Amazon S3에 데이터가 나타나는지 확인합니다. 이는 위와 같은 실패가 발생했음을 나타냅니다.
- HEC 토큰에 대해 인덱서 확인을 활성화했는지 확인합니다.
- Firehose 스트림의 Splunk 대상 구성에서 `HECAcknowledgmentTimeoutInSeconds`의 값을 증가시킵니다.
- Firehose 스트림의 Splunk 대상 구성에서 `RetryOptions`의 `DurationInSeconds` 값을 증가시킵니다.
- HEC 상태를 확인하세요. 상태 확인 활성화는 Splunk로 데이터를 전송하기 위한 사전 시퀀스입니다.
- 데이터 변환을 사용하는 경우 Lambda 함수가 절대로 페이로드 크기가 6MB를 초과하는 응답을 반환하지 않게 해야 합니다. 자세한 내용은 Amazon Data Firehose 데이터 변환을 참조하세요.
- Splunk 파라미터 `ackIdleCleanup`이 `true`로 설정되었는지 확인합니다. 이 파라미터의 기본값은 false입니다. 이 파라미터를 `true`로 설정하려면 다음을 수행합니다.
  - 관리형 Splunk Cloud 배포의 경우, Splunk 지원 포털을 사용하여 사례를 제출합니다. 사례에서, Splunk 지원 부서에 HTTP 이벤트 수집기를 활성화하고, `ackIdleCleanup`에서 `true`을 `inputs.conf`로 설정하고, 이 추가 기능에서 로드 밸런서를 사용하도록 수정을 요청합니다.
  - 배포형 Splunk Enterprise 배포의 경우 `ackIdleCleanup` 파라미터를 `inputs.conf` 파일에서 true로 설정합니다. *nix 사용자의 경우 이 파일은 `$SPLUNK_HOME/etc/apps/`

splunk_httpinput/local/에 있습니다. Windows 사용자의 경우에는 %SPLUNK_HOME%\etc\apps\splunk_httpinput\local\에 있습니다.

- 단일 인스턴스 Splunk Enterprise 배포의 경우 ackIdleCleanup 파라미터를 inputs.conf 파일에서 true로 설정합니다. *nix 사용자의 경우 이 파일은 $SPLUNK_HOME/etc/apps/splunk_httpinput/local/에 있습니다. Windows 사용자의 경우에는 %SPLUNK_HOME%\etc\apps\splunk_httpinput\local\에 있습니다.

- Firehose 스트림에 지정된 IAM 역할이 S3 백업 버킷과 데이터 변환을 위한 Lambda 함수(데이터 변환이 활성화된 경우)에 액세스할 수 있는지 확인합니다. 또한 오류 로그를 확인할 수 있는 CloudWatch 로그 그룹 및 로그 스트림에 대한 액세스 권한이 IAM 역할에 있는지 확인합니다. 자세한 정보는 Firehose에 Splunk 대상에 대한 액세스 권한 부여를 참조하세요.
- S3 오류 버킷(S3 백업)에 전송된 데이터를 다시 Splunk로 리드라이브하려면 Splunk 설명서에 설명된 단계를 따릅니다.
- Amazon Kinesis Firehose용 Splunk 추가 기능 문제 해결을 참조하세요.

Snowflake 문제 해결

이 섹션에서는 Snowflake를 대상으로 사용할 때의 일반적인 문제 해결 단계를 설명합니다.

Firehose 스트림 생성 실패

PrivateLink가 사용 설정된 Snowflake 클러스터로 데이터를 전송하는 스트림에 대해 Firehose 스트림 생성이 실패하는 경우 Firehose에서 VPCE-ID에 연결할 수 없음을 나타냅니다. 이유는 다음 중 하나 때문일 수 있습니다.

- VPCE-ID가 잘못되었을 수 있습니다. 오타가 없는지 확인하세요.
- Firehose는 미리 보기 상태에서 리전이 없는 Snowflake URL을 지원하지 않습니다. Snowflake Account Locator를 사용하여 URL을 입력하세요. 자세한 내용은 Snowflake 설명서를 참조하세요.
- Firehose 스트림이 Snowflake AWS 리전과 동일한 리전에 생성되었는지 확인합니다.
- 문제가 지속되면 AWS 지원팀에 문의하십시오.

전송 실패

Snowflake 테이블로 데이터가 전송되지 않는 경우 다음 사항을 확인하세요. Snowflake 전송에 실패한 데이터는 해당 페이로드의 오류 코드 및 오류 메시지와 함께 S3 오류 버킷으로 전송됩니다. 몇 가지 일반적인 오류 시나리오는 다음과 같습니다. 전체 오류 코드 목록은 Snowflake 데이터 전송 오류 섹션을 참조하세요.

- 오류 코드: Snowflake.DefaultRoleMissing: Firehose 스트림을 생성하는 동안 Snowflake 역할이 구성되지 않았음을 나타냅니다. Snowflake 역할이 구성되지 않은 경우 지정된 Snowflake 사용자에 기본 역할을 설정해야 합니다.
- 오류 코드: Snowflake.ExtraColumns: 입력 페이로드에 있는 추가 열로 인해 Snowflake에 삽입이 거부되었음을 나타냅니다. 테이블에 없는 열은 지정하면 안 됩니다. Snowflake 열의 이름은 대/소문자를 구분합니다. 테이블에 열이 있음에도 불구하고 이 오류로 전송이 실패하는 경우 입력 페이로드 열 이름의 대/소문자가 테이블 정의에 선언된 열 이름과 일치하는지 확인합니다.
- 오류 코드: Snowflake.MissingColumns: 입력 페이로드에 열이 누락되어 Snowflake에 삽입이 거부되었음을 나타냅니다. null 값을 가질 수 없는 모든 열에 대해 값이 지정되어 있는지 확인합니다.
- 오류 코드: Snowflake.InvalidInput: Firehose가 제공된 입력 페이로드를 유효한 JSON 형식으로 구문 분석하지 못했을 때 발생할 수 있습니다. json 페이로드가 제대로 구성되었는지, 추가 큰따옴표, 작은따옴표, 이스케이프 문자 등이 없는지 확인합니다. 현재 Firehose는 단일 JSON 항목만 레코드 페이로드로 지원하며 JSON 배열은 지원하지 않습니다.
- 오류 코드: Snowflake.InvalidValue: 입력 페이로드의 잘못된 데이터 유형으로 인해 전송이 실패했음을 나타냅니다. 입력 페이로드에 지정된 JSON 값이 Snowflake 테이블 정의에 선언된 데이터 유형을 준수하는지 확인합니다.
- 오류 코드: Snowflake.InvalidTableType: Firehose 스트림에 구성된 테이블 유형이 지원되지 않음을 나타냅니다. 지원되는 테이블, 열, 데이터 유형에 대해서는 Snowpipe 스트리밍의 제한 사항을 참조하세요.

> **Note**
>
> 어떤 이유로든 Firehose 스트림을 생성한 후 Snowflake 대상에서 테이블 정의 또는 역할 권한이 변경되면 Firehose가 이러한 변경 사항을 감지하는 데는 몇 분 정도 걸릴 수 있습니다. 이로 인해 전송 오류가 발생하는 경우 Firehose 스트림을 삭제한 후 다시 생성해 보세요.

Firehose 엔드포인트 연결 가능성 문제 해결

Firehose API에 시간 초과가 발생하면 다음 단계를 수행하여 엔드포인트 연결 가능성을 테스트합니다.

- API 요청이 VPC의 호스트에서 이루어지는지 확인합니다. VPC의 모든 트래픽은 Firehose VPC 엔드포인트를 설정해야 합니다. 자세한 내용은 Using Firehose with AWS PrivateLink를 참조하세요.

- 트래픽이 퍼블릭 네트워크 또는 특정 서브넷에 Firehose VPC 엔드포인트가 설정된 VPC에서 오는 경우 호스트에서 다음 명령을 실행하여 네트워크 연결을 확인합니다. Firehose 엔드포인트는 Firehose 엔드포인트 및 할당량에서 찾을 수 있습니다.
  - traceroute 또는 tcping과 같은 도구를 사용하여 네트워크 설정이 올바른지 확인합니다. 해결되지 않으면 네트워크 설정을 확인합니다.

    예제:

    ```
    traceroute firehose.us-east-2.amazonaws.com
    ```

    또는

    ```
    tcping firehose.us-east-2.amazonaws.com 443
    ```

  - 네트워크가 올바르게 설정되었지만 다음 명령이 실패하는 경우 Amazon CA(인증 기관)가 신뢰 체인에 있는지 확인합니다.

    예제:

    ```
    curl firehose.us-east-2.amazonaws.com
    ```

  위의 명령이 성공하면 API를 다시 시도하여 API에서 반환되는 응답이 있는지 확인합니다.

HTTP 엔드포인트 문제 해결

이 섹션에서는 Amazon Data Firehose가 일반 HTTP 엔드포인트 대상 및 Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Splunk 또는 Sumo Logic을 포함한 파트너 대상으로 데이터를 전송하는 문제를 처리할 때 발생하는 일반적인 문제 해결 단계를 설명합니다. 이 섹션에서는 해당하는 모든 대상을 HTTP 엔드포인트라고 합니다. Firehose 스트림에 지정된 IAM 역할이 S3 백업 버킷과 데이터 변환을 위한 Lambda 함수(데이터 변환이 활성화된 경우)에 액세스할 수 있는지 확인합니다. 또한 오류 로그를 확인할 수 있는 CloudWatch 로그 그룹 및 로그 스트림에 대한 액세스 권한이 IAM 역할에 있는지 확인합니다. 자세한 정보는 Firehose에 HTTP 엔드포인트 대상에 대한 액세스 권한 부여를 참조하세요.

> **Note**
>
> 이 섹션의 정보는 Splunk, OpenSearch Service, S3, Redshift와 같은 대상에는 해당되지 않습니다.

CloudWatch Logs

CloudWatch Logging for를 활성화하는 것이 좋습니다. 로그는 대상으로 전송하는 데 오류가 발생한 경우에만 게시됩니다.

대상 예외

ErrorCode: HttpEndpoint.DestinationException

```
{
    "deliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ronald-test",
    "destination": "custom.firehose.endpoint.com...",
    "deliveryStreamVersionId": 1,
    "message": "The following response was received from the endpoint destination.
 413: {\"requestId\": \"43b8e724-dbac-4510-adb7-ef211c6044b9\", \"timestamp\":
 1598556019164, \"errorMessage\": \"Payload too large\"}",
    "errorCode": "HttpEndpoint.DestinationException",
    "processor": "arn:aws:lambda:us-east-1:379522611494:function:httpLambdaProcessing"
}
```

대상 예외는 Firehose가 엔드포인트에 대한 연결을 설정하고 HTTP 요청을 할 수 있지만 200 응답 코드를 받지 못했음을 나타냅니다. 200이 아닌 2xx 응답도 대상 예외가 발생합니다. Amazon Data Firehose는 구성된 엔드포인트에서 수신된 응답 코드 및 잘린 응답 페이로드를 CloudWatch Logs에 기록합니다. Amazon Data Firehose는 수정 또는 해석 없이 응답 코드와 페이로드를 기록하므로 Amazon Data Firehose의 HTTP 전송 요청을 거부한 정확한 이유를 제공하는 것은 엔드포인트에 달려 있습니다. 이러한 예외 경우에 대한 가장 일반적으로 권장하는 문제 해결 방법은 다음과 같습니다.

- 400: 잘못된 Amazon Data Firehose 구성으로 인해 잘못된 요청을 보내고 있음을 나타냅니다. 대상의 URL, 공통 속성, 콘텐츠 인코딩, 액세스 키, 버퍼링 힌트가 올바른지 확인하세요. 필수 구성에 대한 대상별 설명서를 참조하세요.
- 401: Firehose 스트림에 대해 구성한 액세스 키가 잘못되었거나 누락되었음을 나타냅니다.

- 403: Firehose 스트림에 대해 구성한 액세스 키에 구성된 엔드포인트에 데이터를 전송할 권한이 없음을 나타냅니다.
- 413: Amazon Data Firehose가 엔드포인트로 전송하는 요청 페이로드가 엔드포인트에서 처리하기에 너무 크다는 것을 표시합니다. 버퍼링 힌트를 대상의 권장 크기로 낮춰 보세요.
- 429: Amazon Data Firehose가 대상이 처리 가능 속도보다 빠른 속도로 요청을 보내고 있음을 나타냅니다. 버퍼링 시간을 늘리거나 버퍼링 크기를 늘려 버퍼링 힌트를 미세 조정하세요 (단, 대상 한도 내에서).
- 5xx: 대상에 문제가 있음을 나타냅니다. Amazon Data Firehose 서비스는 여전히 제대로 작동합니다.

> **Important**
>
> 중요: 이는 일반적인 문제 해결 권장 사항이지만 엔드포인트별로 응답 코드를 제공하는 이유가 다를 수 있으므로 엔드포인트별 권장 사항을 먼저 따라야 합니다.

잘못된 응답

오류 코드: HttpEndpoint.InvalidResponseFromDestination

```
{
    "deliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ronald-test",
    "destination": "custom.firehose.endpoint.com...",
    "deliveryStreamVersionId": 1,
    "message": "The response received from the specified endpoint is invalid.
 Contact the owner of the endpoint to resolve the issue. Response for request
 2de9e8e9-7296-47b0-bea6-9f17b133d847 is not recognized as valid JSON or has unexpected
 fields. Raw response received: 200 {\"requestId\": null}",
    "errorCode": "HttpEndpoint.InvalidResponseFromDestination",
    "processor": "arn:aws:lambda:us-east-1:379522611494:function:httpLambdaProcessing"
}
```

잘못된 응답 예외는 Amazon Data Firehose가 엔드포인트 대상으로부터 잘못된 응답을 받았음을 나타냅니다. 응답은 응답 사양을 준수해야 합니다. 그렇지 않으면 Amazon Data Firehose는 전송 시도를 실패로 간주하고 구성된 재시도 기간이 초과될 때까지 동일한 데이터를 재전송합니다. Amazon Data Firehose는 응답의 상태가 200이더라도 응답 사양을 따르지 않는 응답은 실패로 간주합니다. Amazon

Data Firehose 호환 엔드포인트를 개발 중인 경우 데이터가 성공적으로 전달되도록 응답 사양을 따르세요.

다음은 잘못된 응답의 일반적인 유형 일부와 그 해결 방법입니다.

- 잘못된 JSON 또는 예상치 못한 필드: 응답을 JSON으로 적절하게 역직렬화할 수 없거나 예상치 못한 필드가 있음을 나타냅니다. 응답이 콘텐츠 인코딩되어 있는지 확인하세요.
- 요청 ID 누락: 응답에 요청 ID가 포함되어 있지 않음을 나타냅니다.
- 요청 ID 불일치: 응답의 RequestID가 발신 요청 ID와 일치하지 않음을 나타냅니다.
- 타임스탬프 누락: 응답에 타임스탬프 필드가 포함되어 있지 않음을 나타냅니다. 타임스탬프 필드는 문자열이 아니라 숫자여야 합니다.
- 콘텐츠 유형 헤더 누락: 응답에 “content-type: application/json” 헤더가 포함되어 있지 않음을 나타냅니다. 다른 콘텐츠 유형은 허용되지 않습니다.

> **Important**
>
> 중요: Amazon Data Firehose는 Firehose 요청 및 응답 사양을 따르는 엔드포인트에만 데이터를 전송할 수 있습니다. 타사 서비스로 대상을 구성하는 경우, 퍼블릭 수집 엔드포인트와는 다른 올바른 Amazon Data Firehose 호환 엔드포인트를 사용해야 합니다. 예를 들어 Datadog의 Amazon Data Firehose 엔드포인트는 `https://aws-kinesis-http-intake.logs.datadoghq.com/`이고 퍼블릭 엔드포인트는 `https://api.datadoghq.com/`입니다.

기타 일반 오류

추가 오류 코드 및 정의는 아래와 같습니다.

- 오류 코드: httpEndPoint.RequestTimeout - 엔드포인트가 응답하기까지 3분 이상 걸렸음을 나타냅니다. 대상 소유자인 경우 대상 엔드포인트의 응답 시간을 단축하세요. 대상 소유자가 아닌 경우 소유자에게 문의하여 응답 시간을 단축할 수 있는 방안(예: 요청당 처리할 데이터를 줄이기 위해 버퍼링 힌트를 줄임)이 있는지 확인하세요.
- 오류 코드: httpEndPoint.responseToolarge - 응답이 너무 크다는 것을 나타냅니다. 응답은 헤더를 포함하여 1MiB 미만이어야 합니다.

- 오류 코드: httpEndPoint.ConnectionFailed - 구성된 엔드포인트와 연결을 설정할 수 없음을 나타냅니다. 구성된 URL에 오타가 있거나, Amazon Data Firehose에서 엔드포인트에 액세스할 수 없거나, 엔드포인트가 연결 요청에 응답하는 데 시간이 너무 오래 걸리는 원인일 수 있습니다.
- 오류 코드: httpEndPoint.ConnectionReset - 연결이 설정되었지만 엔드포인트에 의해 재설정되었거나 조기에 종료되었음을 나타냅니다.
- 오류 코드: httpEndPoint.sslHandshakeFailure - 구성된 엔드포인트에서 SSL 핸드셰이크를 성공적으로 완료할 수 없음을 나타냅니다.

MSK As Source 문제 해결

이 섹션에서는 MSK As Source를 사용할 때의 일반적인 문제 해결 단계를 설명합니다.

> **Note**
>
> 처리, 변환 또는 S3 전송 문제를 해결하려면 이전 섹션을 참조하세요.

호스 생성 실패

MSK As A Source와의 호스가 생성되지 않는 경우 다음 사항을 확인하세요.

- 소스 MSK 클러스터가 활성 상태인지 확인하세요.
- 프라이빗 연결을 사용하는 경우 클러스터의 프라이빗 링크가 켜져 있는지 확인하세요.

  프라이빗 연결을 사용하는 경우 클러스터의 프라이빗 액세스가 켜져 있는지 확인하세요.
- 프라이빗 연결을 사용하는 경우 Firehose가 프라이빗 링크를 만들 수 있도록 허용하는 리소스 기반 정책을 추가해야 합니다. 또한 MSK 교차 계정 권한을 참조하세요.
- 소스 구성의 역할에 클러스터 주제의 데이터를 수집할 수 있는 권한이 있는지 확인하세요.
- VPC 보안 그룹이 클러스터의 부트스트랩 서버가 사용하는 포트로 들어오는 트래픽을 허용하는지 확인하세요.

호스 일시 중단

호스가 일시 중지 상태인 경우 다음 사항을 확인하세요.

- 소스 MSK 클러스터가 활성 상태인지 확인하세요.

- 소스 주제가 존재하는지 확인하세요. 주제가 삭제되었다가 다시 생성된 경우 Firehose 스트림까지 삭제하고 다시 생성해야 합니다.

백프레셔 호스

파티션당 초당 바이트 수 제한을 초과하거나 정상적인 전송 흐름이 느리거나 중단된 경우 DataReadFromSource.Backpressured 값은 1이 됩니다.

- BytesPerSecondLimit(초당 바이트 수 한도)에 도달한 경우 DataReadFromSource.Bytes 지표를 확인하고 한도 증가를 요청하세요.
- CloudWatch 로그, 대상 지표, 데이터 변환 지표 및 형식 변환 지표를 확인하여 병목 현상을 파악하세요.

잘못된 데이터 업데이트

데이터 새로 고침이 잘못된 것 같습니다.

- Firehose는 사용된 레코드의 타임스탬프를 기반으로 데이터 최신성을 계산합니다. 생산자 레코드가 Kafka의 브로커 로그에 유지될 때 이 타임스탬프가 올바르게 기록되도록 하려면, Kafka 주제 타임스탬프 유형 구성을 `message.timestamp.type=LogAppendTime`로 설정하세요.

MSK 클러스터 연결 문제

다음 절차에서는 MSK 클러스터에 대한 연결 상태를 검증하는 방법을 설명합니다. Amazon MSK 클라이언트에 대한 자세한 내용은 Amazon Managed Streaming for Apache Kafka 개발자 안내서의 Amazon MSK 사용 시작하기를 참조하세요.

MSK 클러스터에 대한 연결 검증 방법

1. Unix 기반(AL2 권장) Amazon EC2 인스턴스를 생성합니다. 클러스터에서 VPC 연결만 사용하도록 설정된 경우 EC2 인스턴스가 동일한 VPC에서 실행되는지 확인합니다. 인스턴스가 사용 가능하면 SSH로 접속합니다. 자세한 정보는 Amazon EC2 사용 설명서의 이 자습서를 참조하세요.
2. 다음 명령을 실행하여 Yum 패키지로 Java를 설치합니다. 자세한 정보는 Amazon Corretto 8 사용 설명서의 설치 지침을 참조하세요.

```
sudo yum install java-1.8.0
```

3. 다음 명령을 실행하여 AWS 클라이언트를 설치합니다.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

4. 다음 명령을 실행하여 Apache Kafka 클라이언트 2.6* 버전을 다운로드합니다.

```
wget https://archive.apache.org/dist/kafka/2.6.2/kafka_2.12-2.6.2.tgz
tar -xzf kafka_2.12-2.6.2.tgz
```

5. kafka_2.12-2.6.2/libs 디렉터리로 이동하고 다음 명령을 실행하여 Amazon MSK IAM JAR 파일을 다운로드합니다.

```
wget https://github.com/aws/aws-msk-iam-auth/releases/download/v1.1.3/aws-msk-iam-
auth-1.1.3-all.jar
```

6. Kafka bin 폴더에 client.properties 파일을 생성합니다.

7. awsRoleArn을 Firehose SourceConfiguration에서 사용한 역할 ARN으로 바꾸고 인증서 위치를 확인합니다. AWS 클라이언트 사용자가 역할을 수임하도록 허용합니다awsRoleArn. AWS 클라이언트 사용자는 여기에서 지정한 역할을 수임하려고 시도합니다.

```
[ec2-user@ip-xx-xx-xx-xx bin]$ cat client.properties
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required
 awsRoleArn="<role arn>" awsStsRegion="<region name>";
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
awsDebugCreds=true
ssl.truststore.location=/usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.342.b07-1.amzn2.0.1.x86_64/jre/lib/security/cacerts
ssl.truststore.password=changeit
```

8. 다음 Kafka 명령을 실행해 주제를 나열합니다. 연결이 퍼블릭인 경우, 퍼블릭 엔드포인트 Bootstrap 서버를 사용합니다. 연결이 프라이빗인 경우, 프라이빗 엔드포인트 Bootstrap 서버를 사용합니다.

```
bin/kafka-topics.sh --list --bootstrap-server <bootstrap servers> --command-config
 bin/client.properties
```

요청이 성공하면 다음 예제와 비슷하게 출력됩니다.

```
[ec2-user@ip-xx-xx-xx-xx kafka_2.12-2.6.2]$ bin/kafka-topics.sh --list --bootstrap-
server <bootstrap servers> --command-config bin/client.properties

[xxxx-xx-xx 05:49:50,877] WARN The configuration 'awsDebugCreds' was supplied but
 isn't a known config. (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'ssl.truststore.location' was
 supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'sasl.jaas.config' was supplied
 but isn't a known config. (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration
 'sasl.client.callback.handler.class' was supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'ssl.truststore.password' was
 supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:50:21,629] WARN [AdminClient clientId=adminclient-1] Connection to
 node...
__amazon_msk_canary
__consumer_offsets
```

9. 이전 스크립트를 실행하는 데 문제가 있는 경우 제공한 부트스트랩 서버가 지정된 포트에서 연결할 수 있는지 확인합니다. 이 작업을 수행하려면 다음 명령과 같이 telnet 또는 이와 유사한 유틸리티를 다운로드하여 사용할 수 있습니다.

```
sudo yum install telnet
telnet <bootstrap servers><port>
```

요청이 성공하면 다음 출력이 생성됩니다. 즉, 로컬 VPC 내에서 MSK 클러스터에 연결할 수 있고 부트스트랩 서버가 지정된 포트에서 정상적으로 작동한다는 뜻입니다.

```
Connected to ..
```

10. 요청이 실패하면 VPC 보안 그룹의 인바운드 규칙을 확인합니다. 예를 들어 인바운드 규칙에 다음과 같은 속성을 사용할 수 있습니다.

```
Type: All traffic
Port: Port used by the bootstrap server (e.g. 14001)
```

```
Source: 0.0.0.0/0
```

이전 단계에 표시된 대로 telnet 연결을 다시 시도합니다. 그래도 여전히 연결할 수 없거나 Firehose 연결이 계속 실패하는 경우 AWS Support에 문의하세요.

Amazon Data Firehose 할당량

이 단원에서는 Amazon Data Firehose 내의 현재 할당량(이전에는 제한이라고 함)에 대해 설명합니다. 각 할당량은 다르게 지정되지 않는 한 리전별로 적용됩니다.

Service Quotas 콘솔은 AWS 서비스에 대한 할당량을 보고 관리하고 사용하는 많은 리소스에 대한 할당량 증가를 요청할 수 있는 중앙 위치입니다. 제공하는 할당량 정보를 사용하여 AWS 인프라를 관리합니다. 실제로 필요할 시점보다 미리 할당량 증가를 요청하도록 계획하세요.

자세한 내용은 Amazon Web Services 일반 참조의 Amazon Data Firehose 엔드포인트 및 할당량을 참조하세요.

다음 섹션에서는 Amazon Data Firehose에 다음 할당량이 있음을 보여줍니다.

- Amazon MSK를 Firehose 스트림의 소스로 사용하는 경우 각 Firehose 스트림의 기본 할당량은 파티션당 읽기 처리량 10MB/초 및 최대 레코드 크기 10MB입니다.
- Amazon MSK를 Firehose 스트림의 소스로 사용하면 AWS Lambda가 활성화된 경우 최대 레코드 크기는 6MB이고 Lambda가 비활성화된 경우 최대 레코드 크기는 10MB입니다. AWS Lambda는 수신 레코드를 6MB로 제한하고 Amazon Data Firehose는 6Mb를 초과하는 레코드를 오류 S3 버킷에 전달합니다. Lambda가 비활성화된 경우 Firehose는 수신되는 레코드를 최대 10MB로 제한합니다. Amazon Data Firehose가 Amazon MSK로부터 10MB보다 큰 레코드 크기를 수신하는 경우, Amazon Data Firehose는 이 레코드를 S3 오류 버킷으로 전송하고 Cloudwatch 지표를 사용자 계정으로 보냅니다. AWS Lambda 한도에 대한 자세한 내용은 Lambda 할당량을 참조하세요.
- Firehose 스트림에 동적 파티셔닝이 활성화된 경우 해당 Firehose 스트림에 대해 생성할 수 있는 활성 파티션의 기본 할당량은 500개입니다. 활성 파티션 수는 전송 버퍼 내에 있는 총 활성 파티션 개수입니다. 예를 들어 동적 파티셔닝 쿼리가 초당 3개의 파티션을 구성하고 60초마다 전송을 트리거하도록 버퍼 힌트가 구성된 경우, 활성 파티션은 평균적으로 180개가 됩니다. 파티션에 데이터가 전달된 이후에는 이 파티션은 더 이상 활성화되지 않습니다. 파티션이 더 필요하다면 더 많은 Firehose 스트림을 생성하여 그 전송 스트림에 걸쳐 활성 파티션을 분산시킬 수 있습니다.
- Firehose 스트림에 동적 파티셔닝이 활성화되면, 각 활성 파티션에 초당 1GB의 최대 처리량이 지원됩니다.
- 각 계정마다 리전당 Firehose 스트림 수에 대한 할당량은 다음과 같습니다.
  - 미국 동부(버지니아 북부), 미국 서부(오레곤), 유럽(아일랜드), 아시아 태평양(도쿄): 5,000개의 Firehose 스트림

- 유럽(프랑크푸르트), 유럽(런던), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(서울), 아시아 태평양(뭄바이), AWS GovCloud(미국 서부), 캐나다(서부), 캐나다(중부): Firehose 스트림 2,000개
- 유럽(파리), 유럽(밀라노), 유럽(스톡홀름), 아시아 태평양(홍콩), 아시아 태평양(오사카), 남아메리카(상파울루), 중국(닝샤), 중국(베이징), 중동(바레인), AWS GovCloud(미국 동부), 아프리카(케이프타운): 500개의 Firehose 스트림
- 유럽(취리히), 유럽(스페인), 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태평양(멜버른), 중동(UAE), 이스라엘(텔아비브), 캐나다 서부(캘거리), 캐나다(중부), 아시아 태평양(말레이시아), 아시아 태평양(태국), 멕시코(중부): 100개의 Firehose 스트림
- 이 수를 초과한 경우 CreateDeliveryStream을 호출하면 `LimitExceededException` 예외가 발생합니다. 해당 리전에서 이 할당량을 사용할 수 있는 경우 이 할당량을 늘리려면 Service Quotas를 사용할 수 있습니다. Service Quotas 사용에 대한 자세한 내용은 할당량 증가 요청을 참조하세요.

- Direct PUT이 데이터 소스로 구성된 경우 각각의 Firehose 스트림은 PutRecord 및 PutRecordBatch 요청에 대해 다음과 같이 결합된 할당량을 제공합니다.
  - 미국 동부(버지니아 북부), 미국 서부(오레곤) 및 유럽(아일랜드)의 경우: 레코드 500,000개/초, 요청 2,000개/초, 5Mib/초.
  - 기타 AWS 리전: 100,000개 레코드/초, 1,000개 요청/초, 1MiB/초.

  Firehose 스트림의 처리량 용량을 초과하는 더 높은 데이터 수집 볼륨으로 인해 Direct PUT 스트림에 스로틀링이 발생하는 경우 Amazon Data Firehose는 스로틀링이 포함될 때까지 스트림의 처리량 제한을 자동으로 증가시킵니다. 처리량 증가 및 스로틀링에 따라 Firehose가 스트림의 처리량을 원하는 수준으로 늘리는 데 더 오래 걸릴 수 있습니다. 따라서 실패한 데이터 수집 레코드를 계속 재시도합니다. 데이터 볼륨이 갑자기 큰 버스트에서 증가할 것으로 예상되거나 새 스트림에 기본 처리량 제한보다 높은 처리량이 필요한 경우에 처리량 제한 증가를 요청합니다.

  할당량에 비례하는 세 가지 할당량 척도가 있습니다. 예를 들어, 미국 동부(버지니아 북부), 미국 서부(오레곤) 또는 유럽(아일랜드)의 처리량 할당량을 10MiB/초로 늘리면 나머지 두 할당량은 요청 4,000개/초 및 레코드 1,000,000개/초로 증가합니다.

  > **Note**
  > - 리소스 수준 제한 및 할당량을 서비스 사용을 제어하는 방법으로 사용하지 마세요.
  > - Kinesis Data Streams가 데이터 소스로 구성되어 있으면 이 할당량이 적용되지 않아 Amazon Data Firehose가 아무런 제한 없이 확장/축소됩니다.

- 증가한 할당량이 실행 중인 트래픽보다 훨씬 높을 경우, 대상으로 전송되는 배치가 작아집니다. 이는 비효율적이며 대상 서비스에서 비용이 더 높아질 수 있습니다. 현재 실행 중인 트래픽과 일치하는 할당량까지만 늘려야 하며, 트래픽이 증가하면 할당량을 더 늘려야 합니다.
- 데이터 레코드가 작을수록 비용이 증가할 수 있습니다. Firehose 통합 요금은 서비스에 전송하는 데이터 레코드의 수에 각 레코드의 크기를 가장 가까운 5KB(5120바이트) 단위로 반올림한 값을 곱한 값을 기준으로 합니다. 따라서 같은 양의 수신 데이터(바이트)에 대해 수신되는 레코드 수가 많을수록 비용이 더 올라갑니다. 예를 들어 총 수신 데이터 용량이 5MiB인 경우, 5,000개 이상의 레코드로 5MiB의 데이터를 보내는 것은 1,000개의 레코드를 사용하여 같은 양의 데이터를 전송하는 것에 비해 비용이 더 많이 듭니다. 자세한 내용은 AWS 계산기의 Amazon Data Firehose를 참조하세요.

- 각 Firehose 스트림은, 가용 전송 대상이 없고 소스가 DirectPut인 경우 최대 24시간 동안 데이터 레코드를 저장합니다. 소스가 Kinesis Data Streams(KDS) 이고 대상을 사용할 수 없는 경우 데이터는 KDS 구성에 따라 보관됩니다.
- Base64 인코딩 이전 Amazon Data Firehose로 전송되는 레코드의 최대 크기는 1,000KiB입니다.
- PutRecordBatch 작업은 호출당 최대 500개의 레코드 또는 호출당 4MiB 중 더 작은 크기를 지원합니다. 이 할당량은 변경할 수 없습니다.
- 다음 작업은 각각 초당 최대 5개의 호출을 제공할 수 있으며, 이는 엄격하게 제한됩니다.
  - `CreateDeliveryStream`
  - `DeleteDeliveryStream`
  - `DescribeDeliveryStream`
  - `ListDeliveryStreams`
  - `UpdateDestination`
  - `TagDeliveryStream`
  - `UntagDeliveryStream`
  - `ListTagsForDeliveryStream`
  - `StartDeliveryStreamEncryption`
  - `StopDeliveryStreamEncryption`
- 버퍼 간격 힌트의 범위는 60초 ~ 900초입니다.
- Amazon Data Firehose에서 Amazon Redshift로 전송되는 경우 공개 액세스가 가능한 Amazon Redshift 클러스터만 지원됩니다.
- 재시도 시간 범위는 Amazon Redshift 및 OpenSearch Service 전송 모두 0초 ~ 7,200초입니다.

- 대상이 Amazon S3, Amazon Redshift 또는 OpenSearch Service인 경우 Amazon Data Firehose는 샤드당 최대 5개의 미해결 Lambda 호출을 허용합니다. Splunk에 대한 할당량은 샤드당 10개의 미해결 Lambda 호출입니다.
- CUSTOMER_MANAGED_CMK 유형의 CMK를 사용하여 최대 500개의 Firehose 스트림을 암호화할 수 있습니다.

문서 기록

다음 표에서는 Amazon Data Firehose 설명서의 주요 변경 사항에 대해 설명합니다.

변경	설명	변경 날짜
소스인 데이터베이스 제거(공개 미리 보기)	이제 소스인 데이터베이스(공개 미리 보기)가 제거되었습니다.	2025년 9월 24일
Glue 다중 카탈로그 계층 구조에 대한 지원 추가	이렇게 하면 기본 데이터 카탈로그와 `S3TablesCatalog` 간에 리소스 링크를 사용할 필요 없이 Firehose와 Amazon S3 Tables의 통합이 간소화됩니다. Amazon S3 Tables로 Firehose 스트림 설정을 참조하세요.	2025년 5월 14일
데이터베이스를 소스로 추가(공개 미리 보기)	이제 Amazon S3의 Apache Iceberg 테이블에 데이터베이스 변경 사항을 복제할 수 있습니다.	2024년 11월 15일
대상으로 추가된 Apache Iceberg 테이블의 일반 가용성(GA) 릴리스	Apache Iceberg 테이블을 대상으로 사용하여 Firehose 스트림을 생성할 수 있습니다. Apache Iceberg 테이블에 데이터 전송을(를) 참조하세요.	2024년 9월 30일
데이터 유형 예제 추가됨	Apache Iceberg 테이블에 지원되는 데이터 유형의 예제가 추가되었습니다. 지원되는 데이터 유형 이해을(를) 참조하세요.	2024년 8월 22일
새로운 지역 출범	아시아 태평양(말레이시아)에서 Amazon Data Firehose를 사용할 수 있습니다. Amazon Data Firehose 할당량을(를) 참조하세요.	2024년 8월 22일
Apache Iceberg 테이블을 대상으로 추가(공개 미리 보기)	Apache Iceberg 테이블을 대상으로 사용하여 Firehose 스트림을 생성할 수 있습니다. Apache Iceberg 테이블에 데이터 전송을(를) 참조하세요.	2024년 7월 25일

변경	설명	변경 날짜
Snowflake에 대한 버퍼링 힌트	이제 Snowflake는 버퍼링 힌트를 지원합니다. the section called “Snowflake의 대상 설정 구성”을(를) 참조하세요.	2024년 7월 25일
새 리전의 대상으로서 Snowflake	이제 Snowflake를 아시아 태평양(싱가포르), 아시아 태평양(서울) 및 아시아 태평양(시드니)의 목적지로 사용할 수 있습니다. the section called “Snowflake의 대상 설정 구성”을(를) 참조하세요.	2024년 7월 25일
재구성된 사용 설명서 섹션	사용 설명서의 섹션에 대한 간단한 탐색. Firehose 스트림으로 데이터 전송 및 오류 해결 섹션을 참조하세요.	2024년 7월 5일
Amazon Data Firehose는와 통합됩니다. AWS Secrets Manager	이제 Secrets Manager를 사용하여 보안 암호에 액세스하고 보안 인증 정보 교체를 안전하게 자동화할 수 있습니다. the section called “ AWS Secrets Manager를 사용하여 인증”을(를) 참조하세요.	2024년 6월 6일
Dynatrace에 대한 로그 수집 지원 추가	이제 추가 분석을 위해 Dynatrace로 로그 및 이벤트를 보낼 수 있습니다. the section called “Dynatrace의 대상 설정 구성”을(를) 참조하세요.	2024년 4월 18일
Snowflake를 대상으로 하는 일반 가용성(GA) 릴리스	이제 Snowflake를 대상으로 사용할 수 있습니다. the section called “Snowflake의 대상 설정 구성”을(를) 참조하세요.	2024년 4월 17일
Amazon Kinesis Data Firehose는 이제 Amazon Data Firehose로 알려져 있습니다.	Amazon Kinesis Data Firehose가 Amazon Data Firehose로 브랜드가 변경되었습니다. Amazon Data Firehose란 섹션을 참조하세요	2024년 2월 9일
Snowflake를 대상으로 추가(공개 미리 보기)	Snowflake를 대상으로 Firehose 스트림을 생성할 수 있습니다. the section called “Snowflake의 대상 설정 구성”을(를) 참조하세요.	2024년 1월 19일

변경	설명	변경 날짜
CloudWatch Logs의 자동 압축 해제 추가	새 스트림 또는 기존 스트림에서 압축 해제를 활성화하여 압축 해제된 CloudWatch Logs 데이터를 Firehose 대상으로 전송할 수 있습니다. the section called “Firehose에 CloudWatch Logs 전송”을(를) 참조하세요.	2023년 12월 15일
Splunk Observability Cloud가 대상으로 추가됨	Splunk Observability Cloud를 대상으로 하여 Firehose 스트림을 만들 수 있습니다. the section called “Splunk Observability Cloud의 대상 설정 구성”을(를) 참조하세요.	2023년 10월 3일
Amazon Managed Streaming for Apache Kafka가 데이터 소스로 추가됨	이제 Firehose 스트림으로 정보를 보내도록 Amazon MSK를 구성할 수 있습니다. the section called “Amazon MSK에 대한 소스 설정 구성”을(를) 참조하세요.	2023년 9월 26일
OpenSearch Service 대상의 DocumentID 유형에 대한 지원 추가	OpenSearch Service가 Firehose 스트림의 대상인 경우 DocumenTID 유형은 문서 ID를 설정하는 방법을 나타냅니다. 지원되는 방법은 Firehose에서 생성한 문서 ID 및 OpenSearch Service에서 생성한 문서 ID입니다. the section called “대상 설정 구성”을(를) 참조하세요.	2023년 5월 10일
동적 파티셔닝에 대한 지원 추가	Amazon Data Firehose에서 스트리밍 데이터의 연속 동적 파티셔닝에 대한 지원을 추가했습니다. 파티션 스트리밍 데이터을(를) 참조하세요.	2021년 8월 31일
고객 접두사에 대한 내용을 추가했습니다.	Amazon S3로 전송되는 데이터의 사용자 지정 접두사를 만들 때 사용하는 표현식에 관한 항목을 추가했습니다. the section called “Amazon S3 객체의 사용자 지정 접두사 이해”을(를) 참조하세요.	2018년 12월 20일
새로운 Amazon Data Firehose 자습서 추가	Amazon Data Firehose를 통해 Splunk로 Amazon VPC 흐름 로그를 전송하는 방법을 설명하는 자습서를 추가했습니다. Amazon Data Firehose를 사용하여 Splunk로 VPC 흐름 로그 수집을(를) 참조하세요.	2018년 10월 30일

변경	설명	변경 날짜
Amazon Data Firehose 리전 네 곳이 새로 추가되었습니다.	파리, 뭄바이, 상파울루, 런던이 추가되었습니다. 자세한 내용은 Amazon Data Firehose 할당량 섹션을 참조하세요.	2018년 27월 6일
Amazon Data Firehose 리전 두 곳이 새로 추가되었습니다.	서울과 몬트리올이 추가되었습니다. 자세한 내용은 Amazon Data Firehose 할당량 섹션을 참조하세요.	2018년 13월 6일
새로운 기능 - 소스로서의 Kinesis 스트림	Firehose 스트림에서 Kinesis 스트림이 레코드에 대한 소스로 추가되었습니다. 자세한 내용은 Firehose 스트림의 소스 및 대상 선택 섹션을 참조하세요.	2017년 8월 18일
콘솔 설명서 업데이트	Firehose 스트림 생성 마법사가 업데이트되었습니다. 자세한 내용은 자습서: 콘솔에서 Firehose 스트림 생성 섹션을 참조하세요.	2017년 7월 19일
새로운 데이터 변환	데이터 전송 전에 데이터를 변환하도록 Amazon Data Firehose를 구성할 수 있습니다. 자세한 내용은 Amazon Data Firehose에서 소스 데이터 변환 섹션을 참조하세요.	2016년 19월 12일
새로운 Amazon Redshift COPY 재시도	Amazon Redshift 클러스터에 COPY 명령이 실패하면 다시 시도하도록 Amazon Data Firehose를 구성할 수 있습니다. 자세한 내용은 자습서: 콘솔에서 Firehose 스트림 생성, Amazon Data Firehose의 데이터 전송 이해, Amazon Data Firehose 할당량 섹션을 참조하세요.	2016년 5월 18일
새로운 Amazon Data Firehose 대상, Amazon OpenSearch Service	Amazon OpenSearch Service를 대상으로 하여 Firehose 스트림을 생성할 수 있습니다. 자세한 내용은 자습서: 콘솔에서 Firehose 스트림 생성, Amazon Data Firehose의 데이터 전송 이해, Firehose에 퍼블릭 OpenSearch Service 대상에 대한 액세스 권한 부여 섹션을 참조하세요.	2016년 4월 19일

변경	설명	변경 날짜
새롭게 향상된 CloudWatch 지표 및 문제 해결 기능	Amazon Data Firehose 모니터링 및 Amazon Data Firehose 오류 해결을 업데이트했습니다.	2016년 4월 19일
새롭게 향상된 Kinesis 에이전트	데이터를 전송하도록 Kinesis 에이전트 구성 업데이트됨	2016년 4월 11일
새로운 Kinesis 에이전트	데이터를 전송하도록 Kinesis 에이전트 구성 추가.	2015년 10월 2일
초기 릴리스	Amazon Data Firehose 개발자 안내서의 최초 릴리스입니다.	2015년 10월 4일