개발자 가이드

Amazon Managed Streaming for Apache Kafka

Amazon Managed Streaming for Apache Kafka: 개발자 가이드

Table of Contents

Amazon MSK 개발자 안내서에 오신 것을 환영합니다.

Amazon Managed Streaming for Apache Kafka 개발자 안내서를 시작합니다. 다음 주제는 수행하려는 작업에 따라 이 설명서 사용을 시작하는 데 도움이 될 수 있습니다.

- Amazon MSK 사용 시작하기 자습서에 따라 MSK Provisioned 클러스터를 생성합니다.
- MSK Provisioned란 무엇인가요?에서 MSK Provisioned의 기능에 대해 자세히 알아보세요.
- MSK Serverless로 클러스터 용량을 관리하고 확장할 필요 없이 Apache Kafka를 실행합니다.
- MSK Connect를 사용하여 Apache Kafka 클러스터와 데이터를 주고받으며 스트리밍합니다.
- MSK Replicator를 사용하여 서로 다르거나 동일한 AWS 리전의 MSK Provisioned 클러스터 간에 데이터를 신뢰할 수 있는 방식으로 복제합니다.

주요 내용, 제품 세부 정보, 가격 책정은 Amazon MSK의 서비스 페이지를 참조하세요.

Amazon MSK란 무엇인가요?

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 Apache Kafka를 사용하여 스트리밍 데이터를 처리하는 애플리케이션의 구축 및 실행을 위해 사용할 수 있는 완전관리형 서비스입니다. Amazon MSK는 클러스터 생성, 업데이트, 삭제와 같은 컨트롤 플레인 작업을 제공합니다. 따라서 데이터 생성 및 소비와 같은 Apache Kafka 데이터 영역 작업을 사용할 수 있습니다. Apache Kafka의 오픈 소스 버전을 실행합니다. 즉, 파트너와 Apache Kafka 커뮤니티의 기존 애플리케이션, 도구 및 플러그인이 지원되므로 애플리케이션 코드를 변경할 필요가 없습니다. Amazon MSK를 사용하면 the section called “지원되는 Apache Kafka 버전” 아래 나열된 Apache Kafka 버전 중 하나를 사용하는 클러스터를 생성할 수 있습니다.

이러한 구성 요소는 Amazon MSK의 아키텍처를 설명합니다.

- 브로커 노드 - Amazon MSK 클러스터를 생성할 때 각 가용 영역에 생성할 Amazon MSK 브로커 노드 수를 지정합니다. 최소 값은 가용 영역당 브로커 1개입니다. 각 가용 영역에는 고유한 virtual private cloud(VPC) 서브넷이 있습니다.

  Amazon MSK Provisioned는 Amazon MSK Standard 브로커 및 Amazon MSK Express 브로커의 두 가지 브로커 유형을 제공합니다. MSK Serverless에서 MSK는 트래픽을 처리하는 데 사용되는 브로커 노드를 관리하고 클러스터 수준에서만 Kafka 서버 리소스를 프로비저닝합니다.
- ZooKeeper 노드 - Amazon MSK에서 Apache ZooKeeper 노드도 생성합니다. Apache ZooKeeper는 안정성이 뛰어난 분산 조정을 지원하는 오픈 소스 서버입니다.

- KRaft 컨트롤러 - Apache Kafka 커뮤니티는 Apache Kafka 클러스터의 메타데이터 관리를 위해 Apache ZooKeeper를 대체하기 위해 KRaft를 개발했습니다. KRaft 모드에서 클러스터 메타데이터는 ZooKeeper 노드 대신 Kafka 클러스터의 일부인 Kafka 컨트롤러 그룹 내에서 전파됩니다. KRaft 컨트롤러는 추가 비용 없이 포함되어 있으므로 추가 설정이나 관리가 필요하지 않습니다.
- 생산자, 소비자, 주제 생성자 - Amazon MSK를 사용하면 Apache Kafka 데이터 영역 작업을 통해 주제를 생성하고 데이터를 생산 및 소비할 수 있습니다.
- 클러스터 작업 AWS Management 콘솔, AWS Command Line Interface(AWS CLI) 또는 SDK의 API를 사용하여 컨트롤 플레인 작업을 수행할 수 있습니다. 예를 들어 Amazon MSK 클러스터를 생성하거나 삭제하고, 계정의 모든 클러스터를 나열하고, 클러스터의 속성을 보고, 클러스터에 있는 브로커의 수와 유형을 업데이트할 수 있습니다.

Amazon MSK는 클러스터에 대한 가장 일반적인 장애 시나리오를 감지하고 자동으로 복구하므로 생산자 및 소비자 애플리케이션이 최소한으로 영향을 받으면서 쓰기 및 읽기 작업을 계속할 수 있습니다. Amazon MSK는 브로커 장애가 감지되면 장애를 완화하거나 상태가 좋지 않거나 연결할 수 없는 브로커를 새 브로커로 변경합니다. 또한 가능한 경우 이전 브로커의 스토리지를 재사용하여 Apache Kafka가 복제해야 하는 데이터를 줄입니다. 가용성에 미치는 영향은 Amazon MSK가 탐지 및 복구를 완료하는 데 필요한 시간으로 제한됩니다. 복구 후 생산자와 소비자 앱은 결함 이전에 사용한 것과 동일한 브로커 IP 주소와 계속 통신할 수 있습니다.

Amazon MSK 설정

Amazon MSK를 처음 사용하기 전에 다음 태스크를 완료합니다.

Tasks

- AWS에 가입
- 라이브러리와 도구 다운로드

AWS에 가입

AWS에 가입하면 Amazon Web Services 계정이 Amazon MSK를 포함한 AWS의 모든 서비스에 자동으로 등록됩니다. 사용자에게는 사용한 서비스에 대해서만 요금이 청구됩니다.

이미 AWS 계정이 있다면 다음 태스크로 건너뜁니다. AWS 계정이 없는 경우 다음 절차에 따라 계정을 생성합니다.

Amazon Web Services 계정에 가입하려면 다음을 수행합니다.

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

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

   AWS 계정 루트 사용자에 가입하면 AWS 계정 루트 사용자가 만들어집니다. 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.

라이브러리와 도구 다운로드

다음 라이브러리와 도구는 Amazon MSK로 작업하는 데 도움이 될 수 있습니다.

- AWS Command Line Interface(AWS CLI)는 Amazon MSK를 지원합니다. AWS CLI를 사용하면 명령줄에서 여러 Amazon Web Services를 제어하고 스크립트를 통해 자동화할 수 있습니다. AWS CLI를 최신 버전으로 업그레이드하여 이 사용 설명서에 설명된 Amazon MSK 기능을 지원하도록 하세요. AWS CLI를 업그레이드하는 방법에 대한 자세한 지침은 AWS Command Line Interface 설치를

참조하세요. AWS CLI를 설치한 후에는 이를 구성해야 합니다. AWS CLI를 구성하는 방법에 대한 자세한 내용은 AWS 구성을 참조하세요.

- Amazon Managed Streaming for Kafka API 참조에는 Amazon MSK가 지원하는 API 작업이 문서화되어 있습니다.
- Go, Java, JavaScript, .NET, Node.js, PHP, Python, Ruby를 위한 Amazon Web Services SDK에는 Amazon MSK 지원 및 샘플이 포함되어 있습니다.

MSK Provisioned란 무엇인가요?

Amazon MSK Provisioned 클러스터는 클러스터의 성능을 최적화하고 스트리밍 요구 사항을 충족하는 데 도움이 되는 다양한 기능을 제공합니다. 아래 주제에서는 기능에 대해 자세히 설명합니다.

MSK Provisioned는 Apache Kafka 클러스터를 수동으로 구성하고 확장할 수 있는 MSK 클러스터 배포 옵션입니다. 이를 통해 Apache Kafka 환경을 지원하는 인프라를 다양한 수준으로 제어할 수 있습니다. MSK Provisioned를 사용하면 Kafka 클러스터를 구성하는 인스턴스 유형, 스토리지 볼륨(Standard 브로커) 및 브로커 노드 수를 선택할 수 있습니다. 데이터 처리 요구 사항이 발전함에 따라 브로커를 추가하거나 제거하여 클러스터를 스케일링할 수도 있습니다. 이러한 유연성을 통해 처리량, 보존 용량 또는 기타 성능 특성을 극대화하는지 여부와 관계없이 특정 워크로드 요구 사항에 맞게 클러스터를 최적화할 수 있습니다. MSK Provisioned는 인프라 구성 옵션 외에도 엔터프라이즈급 보안, 모니터링 및 운영 이점을 제공합니다. 여기에는 Apache Kafka 버전 업그레이드, 암호화 및 액세스 제어를 통한 기본 제공 보안, 모니터링을 위한 Amazon CloudWatch와 AWS 서비스같은 다른 와의 통합과 같은 기능이 포함됩니다. MSK Provisioned는 Standard와 Express라는 두 가지 주요 브로커 유형을 제공합니다.

MSK Provisioned API에 대한 자세한 내용은 [Amazon MSK API 참조]를 참조하세요.

Amazon MSK 사용 시작하기

이 자습서에서는 MSK 클러스터를 생성하고, 데이터를 생산 및 소비하고, 지표를 사용하여 클러스터의 상태를 모니터링하는 방법의 예를 보여줍니다. 이 예제에는 MSK 클러스터를 생성할 때 선택할 수 있는 옵션이 모두 나와 있지는 않습니다. 이 자습서의 다른 부분에서는 간단한 설명을 위해 기본 옵션을 선택합니다. 이 옵션이 MSK 클러스터 또는 클라이언트 인스턴스를 설정하는 데 사용되는 유일한 옵션이라는 의미는 아닙니다.

주제

1단계: MSK Provisioned 클러스터 생성

Amazon MSK 사용 시작하기 단계에서는 Amazon MSK Provisioned 클러스터를 생성합니다. 에서 빠른 생성 옵션을 사용하여이 클러스터AWS Management 콘솔를 생성합니다.

를 사용하여 Amazon MSK 클러스터를 생성하려면AWS Management 콘솔

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 클러스터 생성을 선택합니다.
3. 생성 방법에서는 빠른 생성 옵션을 선택한 상태로 둡니다. 빠른 생성 옵션을 사용하면 기본 설정으로 클러스터를 생성할 수 있습니다.
4. 클러스터 이름에 클러스터를 설명하는 이름을 입력합니다. 예를 들어 **`MSKTutorialCluster`**입니다.
5. 일반 클러스터 속성에서 다음을 수행합니다.

    a. 클러스터 유형으로는 프로비저닝을 선택합니다.

    b. 브로커에서 실행하려는 Apache Kafka 버전을 선택합니다. 버전 호환성 보기를 선택하여 비교 테이블을 확인합니다.

    c. 브로커 유형에서 Standard 또는 Express 브로커를 선택합니다.

    d. 브로커 크기를 선택합니다.
6. 이 자습서의 뒷부분에 필요하므로 모든 클러스터 설정 아래의 표에서 다음 설정의 값을 복사하고 저장합니다.

    - VPC
    - 서브넷
    - VPC와 연결된 보안 그룹
7. 클러스터 생성을 선택합니다.
8. 클러스터 요약 페이지에서 클러스터 상태를 확인합니다. Amazon MSK가 클러스터를 프로비저닝하면 상태가 생성 중에서 활성으로 변경됩니다. 상태가 활성이면 클러스터에 연결할 수 있습니다. 클러스터 상태에 대한 자세한 내용은 MSK 프로비저닝된 클러스터의 상태 이해 섹션을 참조하세요.

다음 단계

2단계: Amazon MSK 클러스터에서 주제를 생성할 수 있는 액세스 권한을 부여받는 IAM 역할 생성

2단계: Amazon MSK 클러스터에서 주제를 생성할 수 있는 액세스 권한을 부여받는 IAM 역할 생성

이 단계에서는 두 가지 태스크를 수행합니다. 첫 번째 태스크는 클러스터에서 주제를 생성하고 해당 주제에 데이터를 전송할 수 있는 액세스 권한을 부여하는 IAM 정책을 생성하는 것입니다. 두 번째 태스크는 IAM 역할을 생성하고 해당 정책을 여기에 연결하는 것입니다. 이후 단계에서 이 역할을 맡는 클라이언트 머신을 생성하고 이를 사용하여 클러스터에 주제를 만들고 해당 주제로 데이터를 전송합니다.

주제를 생성하고 주제에 글을 쓸 수 있는 IAM 정책을 만들려면 다음을 수행합니다.

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택합니다.
4. 정책 편집기에서 JSON을 선택한 후 편집기 창의 JSON을 다음 JSON으로 변경합니다.

   다음 예제에서 다음을 변경합니다.

   - 클러스터를 생성한 AWS 리전의 코드가 있는 *region*.
   - 계정 ID 예: *123456789012*, AWS 계정ID 포함.
   - *MSKTutorialCluster* 및 *MSKTutorialCluster*/*7d7131e1-25c5-4e9a-9ac5-ea85bee4da11-14*를 클러스터 이름 및 해당 ID로 변경합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:AlterCluster",
                "kafka-cluster:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:kafka:us-
east-1:123456789012:cluster/MSKTutorialCluster/7d7131e1-25c5-4e9a-9ac5-
ea85bee4da11-14"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:*Topic*",
                "kafka-cluster:WriteData",
                "kafka-cluster:ReadData"
            ],
            "Resource": [
            "arn:aws:kafka:us-east-1:123456789012:topic/MSKTutorialCluster/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
            "arn:aws:kafka:us-east-1:123456789012:group/MSKTutorialCluster/*"
            ]
        }
    ]
}
```

보안 정책을 작성하는 방법에 대한 지침은 [the section called “IAM 액세스 제어”]() 섹션을 참조하세요.

5. 다음을 선택합니다.

6. 검토 및 생성 페이지에서 다음을 수행합니다.

   a. 정책 이름에 설명이 포함된 이름을 입력합니다(예: **`msk-tutorial-policy`**).

   b. 이 정책에 정의된 권한에서 정책에 정의된 권한을 검토 및/또는 편집합니다.

   c. (선택 사항) 정책을 식별, 구성 또는 검색하는 데 도움이 되도록 새 태그 추가를 선택하여 태그를 키-값 페어로 추가합니다. 예를 들어 키-값 페어가 **`Environment`** 및 **`Test`**인 태그를 정책에 추가합니다.

      태그 사용에 대한 자세한 내용은 IAM 사용 설명서의 [AWS Identity and Access Management 리소스 태그를 참조하세요]().

7. 정책 생성을 선택합니다.

IAM 역할을 생성하여 여기에 정책을 연결하려면 다음을 수행합니다.

1. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
2. 신뢰할 수 있는 엔터티 선택 페이지에서 다음을 수행합니다.
    a. 신뢰할 수 있는 엔터티 유형에서 AWS 서비스를 선택합니다.
    b. 서비스 또는 사용 사례에서 EC2를 선택합니다.
    c. 사용 사례에서 EC2를 선택합니다.
3. 다음을 선택합니다.
4. 권한 추가 페이지에서 다음을 수행합니다.
    a. 권한 정책 검색 상자에 이 자습서를 위해 이전에 생성한 정책의 이름을 입력합니다. 그런 다음 정책 이름 왼쪽에서 확인란을 선택합니다.
    b. (선택 사항) 권한 경계를 선택합니다. 이는 서비스 역할에서 사용 가능한 고급 기능이며 서비스에 연결된 역할은 아닙니다. 권한 경계 설정에 대한 자세한 내용은 IAM 사용 설명서의 역할 생성 및 정책 연결(콘솔)을 참조하세요.
5. 다음을 선택합니다.
6. 이름, 검토 및 생성 페이지에서 다음을 수행합니다.
    a. 역할 이름에 설명이 포함된 이름을 입력합니다(예: **msk-tutorial-role**).

> **Important**
>
> 역할 이름을 지정할 때는 다음 사항에 유의하세요.
>
> - 역할 이름은 내에서 고유해야 하며 대/소문자를 구분할 AWS 계정수 없습니다.
>
>   예를 들어, 이름이 **PRODROLE**과 **prodrole**, 두 가지로 지정된 역할을 만들지 마세요. 역할 이름이 정책 또는 ARN의 일부로 사용되는 경우 역할 이름은 대소문자를 구분합니다. 그러나 로그인 프로세스와 같이 콘솔에서 역할 이름이 고객에게 표시되는 경우에는 역할 이름이 대소문자를 구분하지 않습니다.
> - 다른 엔터티가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.

    b. (선택 사항) 설명에 역할에 대한 설명을 입력합니다.

c. (선택 사항) 역할에 대한 사용 사례와 권한을 편집하려면 1단계: 신뢰할 수 있는 엔터티 선택 또는 2단계: 권한 추가 섹션에서 편집을 선택합니다.

d. (선택 사항) 역할을 식별, 구성 또는 검색하는 데 도움이 되도록 새 태그 추가를 선택하여 태그를 키-값 페어로 추가합니다. 예를 들어 **ProductManager** 및 **John**의 키-값 페어를 사용하여 역할에 태그를 추가합니다.

   태그 사용에 대한 자세한 내용은 IAM 사용 설명서의 AWS Identity and Access Management 리소스 태그를 참조하세요.

7. 역할을 검토한 다음 역할 생성을 선택합니다.

다음 단계

3단계: 클라이언트 머신 생성

3단계: 클라이언트 머신 생성

Amazon MSK 사용 시작하기 단계에서 클라이언트 머신을 생성합니다. 이 클라이언트 머신을 사용하여 데이터를 생산하고 소비하는 주제를 만듭니다. 간단하게 하기 위해 클라이언트가 클러스터에 간편하게 연결할 수 있도록 MSK 클러스터와 연결된 VPC에 이 클라이언트 머신을 생성합니다.

클라이언트 머신을 만들려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. Amazon EC2 콘솔 대시보드에서 인스턴스 시작을 선택합니다.
3. (선택 사항) 쉽게 추적할 수 있도록 이름 및 태그 아래의 이름에 클라이언트 머신에 대한 설명이 포함된 이름을 입력합니다. 예를 들어 **MSKTutorialClient**입니다.
4. 애플리케이션 및 OS 이미지(Amazon Machine Image)의 Amazon Machine Image(AMI)(AMI)에서 Amazon Linux 2 AMI(HVM) - 커널 5.10, SSD 볼륨 유형을 선택합니다.
5. 인스턴스 유형의 경우 기본 선택인 t2.micro를 그대로 유지합니다.
6. 키 페어(로그인)에서 기존 키 페어를 선택하거나 새 키 페어를 생성합니다. 인스턴스에 연결하는 데 키 페어가 필요하지 않은 경우 키 페어 없이 계속 진행(권장되지 않음)을 선택할 수 있습니다.

   새 키 페어를 생성하려면 다음 작업을 수행하십시오.

   a. 새 키 페어 생성을 선택합니다.
   b. 키 페어 이름(Key pair name)에 **MSKKeyPair**를 입력합니다.

c. 키 페어 유형 및 프라이빗 키 파일 형식의 경우 기본 선택을 유지합니다.

d. 키 페어 생성(Create key pair)를 선택합니다.

또는 기존 키 페어를 사용할 수 있습니다.

7. 페이지를 아래로 스크롤하여 고급 세부 정보 섹션을 확장한 후 다음을 수행합니다.

   - IAM 인스턴스 프로파일에서 클라이언트 머신이 수임할 IAM 역할을 선택합니다.

     IAM 역할이 없는 경우 다음을 수행합니다.

     i. 새 IAM 프로필 생성을 선택합니다.

     ii. 2단계: IAM 역할 생성에 언급된 단계를 수행합니다.

8. 인스턴스 시작을 선택합니다.

9. 인스턴스 보기를 선택합니다. 그런 다음 보안 그룹 열에서 새 인스턴스와 연결된 보안 그룹을 선택합니다. 보안 그룹의 ID를 복사하여 나중에 사용할 수 있도록 저장합니다.

10. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.

11. 탐색 창에서 Security Groups를 선택합니다. the section called “클러스터 생성”에서 ID를 저장한 보안 그룹을 찾습니다.

12. 인바운드 규칙 탭에서 인바운드 규칙 편집을 선택합니다.

13. 규칙 추가를 선택합니다.

14. 새 규칙의 유형 열에서 모든 트래픽을 선택합니다. 소스 열의 두 번째 필드에서 클라이언트 머신의 보안 그룹을 선택합니다. 이 그룹은 클라이언트 머신 인스턴스를 시작한 후 이름을 저장한 그룹입니다.

15. 규칙 저장을 선택합니다. 이제 클러스터의 보안 그룹은 클라이언트 컴퓨터의 보안 그룹에서 오는 트래픽을 허용할 수 있습니다.

다음 단계

4단계: Amazon MSK 클러스터에서 주제 생성

4단계: Amazon MSK 클러스터에서 주제 생성

Amazon MSK 사용 시작하기 단계에서는 클라이언트 머신에 Apache Kafka 클라이언트 라이브러리 및 도구를 설치한 다음 주제를 생성합니다.

> **Warning**
>
> 이 자습서에서 사용된 Apache Kafka 버전 번호는 예제일 뿐입니다. MSK 클러스터 버전과 동일한 버전의 클라이언트를 사용하는 것을 권장합니다. 이전 클라이언트 버전에는 특정 기능 및 중요한 버그 수정이 누락되었을 수 있습니다.

주제

MSK 클러스터 버전 확인

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 탐색 모음에서 MSK 클러스터를 생성한 리전을 선택합니다.
3. MSK 클러스터를 선택합니다.
4. 클러스터에서 사용되는 Apache Kafka 버전을 확인합니다.
5. 자습서에서 Amazon MSK 버전 번호의 인스턴스를 3단계에서 얻은 버전으로 변경합니다.

클라이언트 머신에 토픽 생성

1. 클라이언트 머신에 연결합니다.

   a. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

   b. 탐색 창에서 인스턴스를 선택합니다. 그런 다음 3단계: 클라이언트 머신 생성에서 생성한 클라이언트 머신 이름 옆의 확인란을 선택합니다.

   c. 작업을 선택하고 연결을 선택합니다. 콘솔의 지침을 따라 클라이언트 머신에 연결합니다.

2. Java를 설치하고 Kafka 버전 환경 변수를 설정합니다.

   a. 다음 명령을 실행하여 클라이언트 머신에 Java를 설치합니다.

   ```
   sudo yum -y install java-11
   ```

   b. 다음 명령에서와 같이 환경 변수 `KAFKA_VERSION`에 MSK 클러스터의 Kafka 버전을 저장합니다. 이 정보는 설정 전반에 걸쳐 필요합니다.

```
export KAFKA_VERSION={KAFKA VERSION}
```

예를 들어 버전 3.6.0을 사용 중인 경우 다음 명령을 사용합니다.

```
export KAFKA_VERSION=3.6.0
```

3. Apache Kafka를 다운로드하고 추출합니다.

   a. Apache Kafka를 다운로드하려면 다음 명령을 실행합니다.

   ```
   wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
   $KAFKA_VERSION.tgz
   ```

   > **Note**
   >
   > 다음 목록에는 문제가 발생할 경우 사용할 수 있는 몇 가지 대체 Kafka 다운로드 정보가 나와 있습니다.
   >
   > - 연결 문제가 발생하거나 미러 사이트를 사용하려는 경우 다음 명령에서와 같이 Apache 미러 선택기를 사용해 보세요.
   >
   >   ```
   >   wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
   >   kafka_2.13-$KAFKA_VERSION.tgz
   >   ```
   >
   > - Apache Kafka 웹 사이트에서 직접 적절한 버전을 다운로드합니다.

   b. 이전 단계에 TAR 파일을 다운로드한 디렉토리에서 다음 명령을 실행합니다.

   ```
   tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
   ```

   c. 새로 생성된 디렉터리의 전체 경로를 KAFKA_ROOT 환경 변수 내에 저장합니다.

   ```
   export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
   ```

4. MSK 클러스터에 대한 인증을 설정합니다.

   a. Amazon MSK IAM 클라이언트 라이브러리의 최신 버전을 찾습니다. 이 라이브러리를 사용하면 클라이언트 머신이 IAM 인증을 사용하여 MSK 클러스터에 액세스할 수 있습니다.

b. 다음 명령을 통해 `$KAFKA_ROOT/libs` 디렉터리로 이동하여 이전 단계에서 찾은 관련 Amazon MSK IAM JAR을 다운로드합니다. *{LATEST VERSION}*을 다운로드하는 실제 버전 번호로 바꿔야 합니다.

```
cd $KAFKA_ROOT/libs
```

```
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-auth-{LATEST VERSION}-all.jar
```

> **Note**
>
> MSK 클러스터와 상호 작용하는 Kafka 명령을 실행하기 전에 Java 클래스 경로에 Amazon MSK IAM JAR 파일을 추가해야 할 수 있습니다. 다음 예제에 표시된 대로 `CLASSPATH` 환경 변수를 설정합니다.
>
> ```
> export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-{LATEST VERSION}-all.jar
> ```
>
> 이를 통해 전체 세션에 대해 `CLASSPATH`가 설정되므로 모든 후속 Kafka 명령에서 JAR을 사용할 수 있습니다.

c. `$KAFKA_ROOT/config` 디렉터리로 이동하여 클라이언트 구성 파일을 생성합니다.

```
cd $KAFKA_ROOT/config
```

d. 다음 속성 설정을 복사하여 새 파일에 붙여넣습니다. 파일을 **`client.properties`**(으)로 저장합니다.

```
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandle
```

5. (선택 사항) Kafka 도구의 Java 힙 크기를 조정합니다.

메모리 관련 문제가 발생하거나 많은 수의 토픽 또는 파티션으로 작업하는 경우 Java 힙 크기를 조정할 수 있습니다. 이를 수행하려면 Kafka 명령을 실행하기 전에 `KAFKA_HEAP_OPTS` 환경 변수를 설정합니다.

다음 예제에서는 최대 힙 크기와 초기 힙 크기를 모두 512MB로 설정합니다. 특정 요구 사항 및 사용 가능한 시스템 리소스에 따라 이러한 값을 조정합니다.

```
export KAFKA_HEAP_OPTS="-Xmx512M -Xms512M"
```

6. 클러스터 연결 정보를 가져옵니다.

   a. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.

   b. 클러스터 상태가 활성이 될 때까지 기다립니다. 몇 분 정도 걸릴 수 있습니다. 상태가 활성이 되면 클러스터 이름을 선택합니다. 클러스터 요약이 포함된 페이지로 이동합니다.

   c. 클라이언트 정보 보기를 선택합니다.

   d. 프라이빗 엔드포인트에 대한 연결 문자열을 복사합니다.

      각 브로커에 대해 3개의 엔드포인트가 제공됩니다. 다음 명령에서와 같이 이러한 연결 문자열 중 하나를 BOOTSTRAP_SERVER 환경 변수에 저장합니다. *<bootstrap-server-string>*을 연결 문자열의 실제 값으로 바꿉니다.

      ```
      export BOOTSTRAP_SERVER=<bootstrap-server-string>
      ```

7. 다음 명령을 실행하여 토픽을 생성합니다.

```
$KAFKA_ROOT/bin/kafka-topics.sh --create --bootstrap-server $BOOTSTRAP_SERVER
 --command-config $KAFKA_ROOT/config/client.properties --replication-factor 3 --
partitions 1 --topic MSKTutorialTopic
```

`client.properties` 파일에 대해 NoSuchFileException을 가져오는 경우 이 파일이 Kafka bin 디렉터리 내의 현재 작업 디렉터리에 있는지 확인합니다.

> **Note**
>
> 전체 세션에 대해 CLASSPATH 환경 변수를 설정하지 않으려면 각 Kafka 명령에 CLASSPATH 변수 접두사를 붙일 수도 있습니다. 이 접근 방식은 해당 특정 명령에만 클래스 경로를 적용합니다.
>
> ```
> CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-{LATEST VERSION}-all.jar \
> $KAFKA_ROOT/bin/kafka-topics.sh --create \
> --bootstrap-server $BOOTSTRAP_SERVER \
> --command-config $KAFKA_ROOT/config/client.properties \
> ```

```
--replication-factor 3 \
--partitions 1 \
--topic MSKTutorialTopic
```

8. (선택 사항) 토픽이 성공적으로 생성되었는지 확인합니다.

   a. 명령이 성공하면 `Created topic MSKTutorialTopic.` 메시지가 표시됩니다.
   b. 모든 토픽을 나열하여 토픽이 존재하는지 확인합니다.

   ```
   $KAFKA_ROOT/bin/kafka-topics.sh --list --bootstrap-server $BOOTSTRAP_SERVER --
   command-config $KAFKA_ROOT/config/client.properties
   ```

   명령이 실패하거나 오류가 발생하면 Amazon MSK 클러스터 문제 해결 섹션에서 문제 해결 정보를 참조하세요.

9. (선택 사항) 이 자습서에서 사용한 환경 변수를 삭제합니다.

   이 자습서의 다음 단계를 위해 환경 변수를 유지하려면 이 단계를 건너뜁니다. 아니면 다음 예제에서와 같이 이러한 변수를 설정 해제할 수 있습니다.

   ```
   unset KAFKA_VERSION KAFKA_ROOT BOOTSTRAP_SERVER CLASSPATH KAFKA_HEAP_OPTS
   ```

다음 단계

5단계: 데이터 생산 및 소비

5단계: 데이터 생산 및 소비

Amazon MSK 사용 시작하기 단계에서 데이터를 생성하고 소비합니다.

메시지를 생산하고 소비하려면

1. 콘솔 생산자를 시작하려면 다음 명령을 실행합니다.

   ```
   $KAFKA_ROOT/bin/kafka-console-producer.sh --broker-list $BOOTSTRAP_SERVER --
   producer.config $KAFKA_ROOT/config/client.properties --topic MSKTutorialTopic
   ```

2. 원하는 메시지를 입력하고 Enter키를 누릅니다. 이 단계를 두 번 또는 세 번 반복하십시오. 한 줄을 입력하고 Enter를 누를 때마다, 그 줄은 Apache Kafka 클러스터에 별도의 메시지로 전송됩니다.

3. 클라이언트 머신에 대한 연결을 열어 둔 다음, 새 창에서 해당 머신에 대한 별도의 두 번째 연결을 엽니다. 이 세션은 새로운 세션이므로 KAFKA_ROOT 및 BOOTSTRAP_SERVER 환경 변수를 다시 설정합니다. 이러한 환경 변수를 설정하는 방법에 대한 자세한 내용은 클라이언트 머신에 토픽 생성 섹션을 참조하세요.
4. 클라이언트 머신에 대한 두 번째 연결 스트링을 통해 다음 명령을 실행하여 콘솔 컨슈머를 생성합니다.

```
$KAFKA_ROOT/bin/kafka-console-consumer.sh --bootstrap-server $BOOTSTRAP_SERVER --
consumer.config $KAFKA_ROOT/config/client.properties --topic MSKTutorialTopic --
from-beginning
```

   콘솔 프로듀서 명령을 사용할 때 앞에서 입력한 메시지가 표시됩니다.
5. 생산자 창에 메시지를 더 입력하고 소비자 창에 메시지가 표시되는지 확인합니다.

다음 단계

6단계: Amazon CloudWatch를 사용하여 Amazon MSK 지표 보기

6단계: Amazon CloudWatch를 사용하여 Amazon MSK 지표 보기

Amazon MSK 사용 시작하기 단계에서는 Amazon CloudWatch에서 Amazon MSK 지표를 살펴봅니다.

CloudWatch에서 Amazon MSK 지표를 보려면 다음을 수행합니다.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 모든 지표 탭을 선택한 다음 AWS/Kafka를 선택합니다.
4. 브로커 수준 지표를 보려면 Broker ID, Cluster Name(브로커 ID, 클러스터 이름)을 선택합니다. 클러스터 수준 지표의 경우 클러스터 이름을 선택합니다.
5. (선택 사항) 그래프 창에서 통계와 기간을 선택한 후 해당 설정을 사용하여 CloudWatch 경보를 생성합니다.

다음 단계

7단계:이 자습서를 위해 생성된 AWS리소스 삭제

7단계:이 자습서를 위해 생성된 AWS리소스 삭제

Amazon MSK 사용 시작하기의 마지막 단계에서는 이 자습서를 위해 생성한 MSK 클러스터와 클라이언트 머신을 삭제합니다.

를 사용하여 리소스를 삭제하려면AWS Management 콘솔

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 클러스터 이름을 선택합니다. MSKTutorialCluster를 예로 들 수 있습니다.
3. 실행을 선택하고 삭제를 선택합니다.
4. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
5. 클라이언트 머신용으로 생성한 인스턴스(예: **`MSKTutorialClient`**)를 선택합니다.
6. 인스턴스 상태를 선택한 다음 인스턴스 종료를 선택합니다.

IAM 정책 및 역할을 삭제하려면

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 검색 상자에 이 자습서를 위해 생성한 IAM 역할의 이름을 입력합니다.
4. 역할을 선택합니다. 그런 다음 역할 삭제를 선택하고 삭제를 확인합니다.
5. 탐색 창에서 정책을 선택합니다.
6. 검색 상자에 이 자습서를 위해 생성한 정책의 이름을 입력합니다.
7. 정책을 선택하면 해당 요약 페이지가 열립니다. 정책의 요약 페이지에서 정책 삭제를 선택합니다.
8. 삭제를 선택합니다.

Amazon MSK: 작동 방식

Amazon MSK는 Apache Kafka를 사용하여 스트리밍 데이터를 처리하는 애플리케이션을 쉽게 구축하고 실행하는 데 사용할 수 있는 완전관리형 Apache Kafka 서비스입니다. 이 가이드는 개발자가 Amazon MSK의 작동 방식과 애플리케이션에서 이를 효과적으로 사용하는 방법을 이해하는 데 도움이 되는 정보를 제공합니다.

상위 수준에서, Amazon MSK는 AWS에서 프로비저닝하고 운영하는 완전관리형 Apache Kafka 클러스터를 제공합니다. 즉, EC2 인스턴스 프로비저닝, 네트워크 설정 구성, Kafka 브로커 관리 또는 지속적인 유지 관리 작업 수행에 대해 걱정할 필요가 없습니다. 대신 애플리케이션 구축에 집중하고

Amazon MSK가 인프라를 처리하도록 할 수 있습니다. Amazon MSK는 필요한 컴퓨팅, 스토리지 및 네트워크 리소스를 자동으로 프로비저닝하고 자동 규모 조정, 고가용성 및 장애 조치와 같은 기능을 제공하여 Kafka 클러스터의 신뢰성과 가용성을 보장합니다. 이 가이드에서는 Amazon MSK의 주요 구성 요소와 이를 사용하여 스트리밍 데이터 애플리케이션을 구축하는 방법을 다룹니다.

프로비저닝된 클러스터 관리

Amazon MSK 클러스터는 계정에서 생성할 수 있는 기본 Amazon MSK 리소스입니다. 이 섹션의 항목에서는 일반적인 Amazon MSK 작업을 수행하는 방법에 대해 설명합니다. MSK 클러스터에서 수행할 수 있는 모든 작업 목록은 다음을 참조하세요.

- AWS Management 콘솔은
- Amazon MSK API 참조
- Amazon MSK CLI 명령 참조

주제

MSK 프로비저닝된 클러스터 생성

> **Important**
>
> 클러스터를 생성한 후에는 MSK 프로비저닝된 클러스터의 VPC를 변경할 수 없습니다.

MSK 프로비저닝된 클러스터를 생성하기 전에 Amazon Virtual Private Cloud(VPC)가 있고 해당 VPC 내에 서브넷을 설정해야 합니다.

미국 서부(캘리포니아 북부) 리전의 Standard 브로커의 경우, 서로 다른 두 가용 영역에 두 개의 서브넷이 필요합니다. Amazon MSK를 사용할 수 있는 다른 모든 리전에서는 2개 또는 3개의 서브넷을 지정할 수 있습니다. 모든 서브넷은 서로 다른 가용 영역에 있어야 합니다. Express 브로커의 경우 3개의 서로 다른 가용 영역에 3개의 서브넷이 필요합니다. MSK 프로비저닝된 클러스터를 생성하면 Amazon MSK에서 지정된 서브넷에 브로커 노드를 균등하게 분배합니다.

주제

를 사용하여 MSK 프로비저닝 클러스터 생성AWS Management 콘솔

이 주제의 프로세스에서는 AWS Management 콘솔에서 사용자 지정 생성 옵션을 사용하여 MSK 프로비저닝된 클러스터를 생성하는 일반적인 작업을 설명합니다. 에서 사용할 AWS Management 콘솔수 있는 다른 옵션을 사용하여 다음을 생성할 수도 있습니다.

- 서버리스 클러스터
- 빠른 생성 옵션을 사용하는 MSK 프로비저닝 클러스터

이 주제의 절차

- 4단계: 보안 설정 구성
- 5단계: 모니터링 옵션 구성
- 6단계: 클러스터 구성 검토

1단계: 초기 클러스터 설정 및 구성

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 클러스터 생성을 선택합니다.
3. 클러스터 생성 방법 에서 사용자 지정 생성을 선택합니다.
4. 클러스터 이름의 경우 64자 이하로 고유하게 지정합니다.
5. 클러스터 유형으로는 프로비저닝을 선택합니다.
6. Apache Kafka 버전에서 브로커에서 실행할 버전을 선택합니다. 각 Apache Kafka 버전에서 지원되는 Amazon MSK 기능의 비교를 보려면 버전 호환성 보기를 선택합니다.
7. 브로커 섹션에서 다음을 수행합니다.

    a. 브로커 유형에서 다음 옵션 중 하나를 선택합니다.

    - Express 브로커: 완전관리형 가상 스토리지를 갖춘 확장 가능한 고성능 브로커입니다. 까다로운 고처리량 애플리케이션에 대해 이 브로커 유형을 선택합니다.
    - Standard 브로커: 전체 구성 제어가 가능한 기존 Kafka 브로커. 처리량 요구 사항이 중간인 범용 워크로드의 경우 이 브로커 유형을 선택합니다.

    이러한 브로커 타입에 대한 자세한 설명은 Amazon MSK 브로커 유형 섹션을 참조하세요.

    b. 브로커 크기의 경우 클러스터의 컴퓨팅, 메모리 및 스토리지 요구 사항에 따라 클러스터에 사용할 크기를 택합니다.

    c. 영역의 수의 경우 브로커가 배포되는 AWS 가용 영역의 수를 선택합니다.

    고가용성을 위해 Express 브로커에는 3개의 가용 영역이 필요합니다.

    d. 영역별 브로커의 경우 Amazon MSK가 각 가용 영역에 생성할 브로커의 수를 지정합니다. 최소 값은 가용 영역당 브로커 1개이고 최대 값은 ZooKeeper 기반 클러스터의 경우 클러스터당 브로커 30개이고 KRaft 기반 클러스터의 경우 클러스터당 브로커 60개입니다.

2단계: 스토리지 및 클러스터 설정 구성

이 절차에서는 모든 브로커에서 데이터 스토리지 요구 사항을 구성하고 스토리지 모드를 지정하는 방법을 설명합니다. 이를 통해 워크로드 요구 사항에 따라 데이터 스토리지 요구 사항을 정의할 수 있습니다. 또한 이 절차에서는 브로커의 작동 방식을 제어하는 클러스터 구성 설정을 설명합니다. 이 설정에는 브로커 구성, 기본 주제 설정, 계층형 스토리지 정책 등이 포함됩니다.

1. 브로커 유형을 Standard로 선택한 경우 스토리지 섹션에서 다음을 수행합니다.

    a. 스토리지의 경우 클러스터에 포함할 초기 스토리지 양을 선택합니다. 클러스터를 생성한 후에는 스토리지 용량을 줄일 수 없습니다.

    b. (선택) 선택한 브로커 크기(인스턴스 크기)에 따라 브로커당 프로비저닝된 스토리지 처리량 또한 지정할 수 있습니다. 이 옵션을 사용하면 각 브로커의 Amazon EBS 볼륨에 전용 입력 및 출력(I/O) 성능을 할당할 수 있습니다.

       이 옵션을 활성화하려면 x86의 경우 브로커 크기(인스턴스 크기)로 kafka.m5.4xlarge 이상을 선택하고 Graviton 기반 인스턴스의 경우 kafka.m7g.2xlarge 이상을 선택합니다. 그런 다음 프로비저닝된 스토리지 처리량 활성화 확인란을 선택합니다. 이 확인란을 선택하면 초당 최소 250MiB의 처리량을 수동으로 설정할 수 있습니다. 이는 예측 가능한 고속 스토리지 성능이 필요한 I/O 집약적 워크로드 또는 애플리케이션에 유용합니다. 자세한 내용은 ??? 단원을 참조하십시오.

    c. 클러스터 스토리지 모드의 경우 클러스터 내에서 데이터를 저장하고 관리하는 방법을 지정합니다. 이 옵션은 브로커에 사용되는 스토리지의 유형과 구성을 결정합니다. 다음 옵션 중 하나를 선택하세요.

       - EBS 스토리지만 해당: 각 브로커에 연결된 Amazon Elastic Block Store(Amazon EBS) 볼륨에 모든 주제 데이터를 로컬로 저장합니다. 일관된 성능 요구 사항과 최신 메시지에 대한 빠른 액세스를 위해 이 모드를 선택합니다.
       - 계층형 스토리지 및 EBS 스토리지: 로컬 Amazon EBS 데이터를 Amazon S3의 대규모 데이터 세트를 위한 비용 효율적인 원격 스토리지와 결합합니다. 이 모드는 Amazon EBS 스토리지 비용을 줄이고, 데이터 보존을 더 오래 지원하며, 수동 개입 없이 스토리지를 자동으로 확장합니다. 더 낮은 비용으로 더 오랜 기간 동안 데이터를 보존하거나 스토리지가 크게 증가해야 하는 경우 이 모드를 선택합니다.

> **Note**
>
> Express 브로커의 스토리지는 관리할 필요가 없습니다.

2. 클러스터 구성에서 다음 옵션 중 하나를 지정하여 클러스터의 동작을 정의합니다.

   - Amazon MSK 기본 구성: 범용 사용 사례에 최적화된 사전 정의된 구성 세트를 포함합니다. 빠른 클러스터 설정 및 배포를 위해 이 옵션을 선택합니다. Amazon MSK 구성에 대한 자세한 내용은 Amazon MSK Provisioned 구성 섹션을 참조하세요.
   - 사용자 지정 구성: 자체 브로커 및 주제 설정을 지정할 수 있습니다. 목록에서 기존 사용자 지정 구성을 선택하거나 새 사용자 지정 구성을 생성할 수 있습니다. 특정 성능 튜닝, 보안 설정 작업 등 브로커를 미세 조정하여 제어하려면 이 옵션을 선택합니다.

3. 다음을 선택하여 계속 진행합니다.

3단계: 네트워크 설정 구성

네트워크 구성은 AWS인프라 내에 클러스터를 배포하는 방법을 정의합니다. 여기에는 네트워킹, 가용성 및 액세스를 제어하는 VPC, 가용 영역 및 서브넷과 보안 그룹이 포함됩니다.

1. 네트워킹의 경우 다음을 수행합니다.

   a. 클러스터에 사용하고 싶은 VPC를 선택합니다.

   b. 이전에 선택한 가용 영역 수에 따라 브로커가 배포할 가용 영역 및 서브넷을 지정합니다.

      미국 서부(캘리포니아 북부) 리전의 Standard 브로커의 경우, 서로 다른 두 가용 영역에 두 개의 서브넷이 필요합니다. Amazon MSK를 사용할 수 있는 다른 모든 리전에서는 2개 또는 3개의 서브넷을 지정할 수 있습니다. 모든 서브넷은 서로 다른 가용 영역에 있어야 합니다.

      Express 브로커의 경우 3개의 서로 다른 가용 영역에 3개의 서브넷이 필요합니다.

      프로비저닝된 MSK 클러스터를 생성하면 MSK에서 지정된 서브넷에 브로커 노드를 균등하게 분배합니다.

   c. Amazon EC2의 보안 그룹의 경우 클러스터에 대한 액세스 권한을 부여하려는 보안 그룹을 하나 이상 선택하거나 생성합니다. 이 Amazon EC2 보안 그룹은 브로커에 대한 인바운드 및 아웃바운드 트래픽을 제어합니다. 예를 들어 클라이언트 시스템의 보안 그룹입니다.

      사용자와 공유된 보안 그룹을 지정하는 경우 사용자에게 해당 보안 그룹에 대한 권한이 있어야 합니다. 특히, `ec2:DescribeSecurityGroups` 권한이 필요합니다. 자세한 내용은 MSK 클러스터에 연결 섹션을 참조하세요.

2. 다음을 선택하여 계속 진행합니다.

4단계: 보안 설정 구성

1. 보안 설정 섹션에서 다음을 수행합니다.

   - 다음 인증 및 권한 부여 방법 중 하나 이상을 선택하여 Kafka 클러스터에 대한 클라이언트 액세스를 제어합니다.

     - 인증되지 않은 액세스: 클라이언트가 인증 자격 증명을 제공하지 않고 클러스터에 액세스할 수 있도록 허용합니다. 이 방법은 보안 위험이며 보안 모범 사례 미준수에 해당할 수 있습니다. 자세한 내용은 msk-unrestricted-access-check를 참조하세요.
     - IAM 역할 기반 인증: AWSIAM 사용자/역할을 사용하여 클라이언트 인증 및 권한 부여를 활성화합니다. 이 방법을 사용하면 IAM 정책을 통해 클러스터 액세스를 세밀하게 제어할 수 있습니다. 이미 실행 중인 애플리케이션에는이 방법을 사용하는 것이 좋습니다AWS.
     - SASL/SCRAM 인증: 클라이언트가 AWS Secrets Manager인증을 위해에 저장된 사용자 이름과 암호 자격 증명을 제공해야 합니다. Amazon MSK는 Secrets Manager에서 이러한 자격 증명을 검색하고 사용자를 안전하게 인증합니다.

       클러스터 인증과 관련된 로그인 자격 증명을 설정하려면 먼저 Secrets Manager에서 보안 암호 리소스를 생성합니다. 그런 다음 로그인 자격 증명을 해당 보안 암호와 연결합니다. 이 액세스 제어 방법에 대한 자세한 내용은 Amazon MSK 클러스터를 위해 SASL/SCRAM 인증 설정 섹션을 참조하세요.
     - AWS Certificate Manager(ACM)을 통한 TLS 클라이언트 인증: 디지털 인증서를 사용하여 클라이언트와 브로커 간의 상호 인증을 활성화합니다. AWS 계정클러스터와 동일하거나 다른에서 AWS Private Certificate Authority(AWS Private CA)를 구성해야 합니다.

       mTLSAWS Private CA를 구현할 때는 각 MSK 클러스터에 대해 독립적인를 사용하는 것이 좋습니다. 이렇게 하면 PCA에서 서명한 TLS 인증서가 단일 MSK 클러스터로만 인증되므로 엄격한 액세스 제어를 유지할 수 있습니다.

2. 암호화에서 저장 데이터를 암호화하는 데 사용할 KMS 키의 종류를 선택합니다. 자세한 내용은 the section called “저장 시 Amazon MSK 암호화” 단원을 참조하십시오.

   저장 데이터 암호화는 저장된 데이터 무결성을 보호하고, 전송 중 데이터 암호화는 전송 중 네트워크 모니터링으로부터 데이터 기밀성을 보호합니다.

3. 다음을 선택하여 계속 진행합니다.

5단계: 모니터링 옵션 구성

이 절차에서는 브로커 지표를 설정하고 브로커 로그를 수집 및 전송하는 방법을 설명합니다. 이러한 설정을 사용하면 클러스터의 상태, 성능을 관찰 및 분석하고 문제를 해결할 수 있습니다. 자세한 내용은 the section called “클러스터 모니터링” 단원을 참조하십시오.

1. 이 클러스터에 대한 Amazon CloudWatch 지표를 설정하기 위해 다음 모니터링 수준 중 하나를 선택합니다. 각 모니터링 수준에서 수집된 지표는 시각화 및 알림을 위해 CloudWatch와 통합됩니다.

   a. 기본 모니터링: 추가 비용 없이 일련의 필수 클러스터 수준 지표를 제공합니다. 이 수준은 일반적인 모니터링이 필요한 대부분의 사용 사례에 적합합니다.

   b. 향상된 브로커 수준 모니터링: 추가 비용으로 자세한 브로커 지표를 제공합니다. 이 수준에는 기본 모니터링 지표와 더불어 다른 브로커로의 수신/송신 계층형 스토리지 바이트 수, 읽기/쓰기 작업에 소요되는 총 시간 등 더 세부적인 브로커 지표가 포함됩니다. 이 수준의 지표에는 해당 비용을 지불하지만, 기본 수준 지표는 계속 무료로 제공됩니다.

   c. 향상된 주제 수준 모니터링: 추가 비용으로 개별 주제에 대한 지표를 제공합니다. 브로커 전반의 주제 성능을 더 세부적으로 보려면 이 수준을 선택합니다. 이 수준에는 향상된 브로커 수준 모니터링과 더불어 주제 수준 지표(예: 지정된 주제 및 초당 수신 메시지 수 등 계층형 스토리지 지표)가 포함됩니다.

   d. 향상된 파티션 수준 모니터링: 추가 비용으로 파티션당 지표를 가장 세밀하게 확인할 수 있습니다. 브로커 전반에서 각 주제 내의 파티션별 지표를 캡처하여 가장 세부적인 모니터링 결과를 얻으려면 이 수준을 선택합니다. 이 수준에는 향상된 주제 수준 모니터링과 더불어 세분화된 파티션별 지표(예: 오프셋 지연 지표)가 포함됩니다.

   이러한 각 모니터링 수준에서 Standard 및 Express 브로커 유형에 사용할 수 있는 지표에 대한 자세한 내용은 Standard 브로커에 대한 CloudWatch 지표 및 Express 브로커에 대한 CloudWatch 지표 섹션을 참조하세요.

2. (선택 사항) JMX Exporter, Node Exporter 또는 둘 다를 사용하여 Prometheus 형식으로 지표를 내보내려면 Prometheus에서 오픈 모니터링 활성화를 선택합니다. 이 옵션에 대한 자세한 내용은 Prometheus를 사용하여 모니터링 섹션을 참조하세요.

3. (선택 사항) 문제 해결 및 감사를 위해 브로커 로그를 다양한 AWS 서비스에 전달하도록 MSK 클러스터를 구성하려면 다음 옵션 중 하나 이상을 선택합니다. Amazon MSK는 이러한 대상 리소스가 아직 존재하지 않는 경우 이를 생성하지 않습니다. 자세한 내용은 브로커 로그 단원을 참조하십시오.

- Amazon CloudWatch Logs로 전송: 클러스터링, 검색 및 시각화 기능과 함께 로그를 CloudWatch로 전송합니다. AWS Management 콘솔을 벗어나지 않고 로그를 쿼리하고 분석할 수 있습니다.
- Amazon S3로 전송: 장기 보관 및 분석을 위해 로그를 파일로 Amazon S3에 저장합니다.
- Amazon Data Firehose로 전송: 실시간 문제 해결을 위해 Amazon OpenSearch Service로 자동 전송되도록 Firehose에 로그를 전송합니다.

4. (선택 사항) 클러스터를 식별, 구성 또는 검색하는 데 도움이 되도록 새 태그 추가를 선택하여 태그를 키-값 페어로 추가합니다. 예를 들어 키-값 페어가 **`Load testing`** 및 **`Test`**인 태그를 클러스터에 추가합니다.

   클러스터에서 태그를 사용하는 법에 대한 자세한 내용을 알아보려면 Amazon MSK 클러스터에 태그 지정 섹션을 참조하세요.
5. 다음을 선택하여 계속 진행합니다.

6단계: 클러스터 구성 검토

1. 클러스터의 설정을 검토합니다.

   편집 또는 이전을 선택하여 이전에 지정한 설정을 변경하거나 이전 콘솔 화면으로 돌아갑니다.
2. 클러스터 생성을 선택합니다.
3. 클러스터 세부 정보 페이지의 클러스터 요약 섹션에서 이 클러스터의 상태를 확인합니다. Amazon MSK가 클러스터를 프로비저닝하면 상태가 생성 중에서 활성으로 변경됩니다. 상태가 활성이면 클러스터에 연결할 수 있습니다. 클러스터 상태에 대한 자세한 내용은 MSK 프로비저닝된 클러스터의 상태 이해 섹션을 참조하세요.

를 사용하여 프로비저닝된 Amazon MSK 클러스터 생성AWS CLI

1. 다음 JSON을 복사하여 파일에 저장합니다. 파일 이름을 `brokernodegroupinfo.json`으로 지정합니다. JSON의 서브넷 ID를 서브넷에 해당하는 값으로 바꿉니다. 이러한 서브넷은 서로 다른 가용 영역에 있어야 합니다. *`"Security-Group-ID"`*를 클라이언트 VPC의 하나 이상의 보안 그룹 ID로 바꿉니다. 이 보안 그룹과 연결된 클라이언트는 클러스터에 액세스할 수 있습니다. 사용자와 공유된 보안 그룹을 지정할 경우 사용자가 해당 보안 그룹에 대한 권한이 있어야 합니다. 특히, `ec2:DescribeSecurityGroups` 권한이 필요합니다. 예제는 Amazon EC2: 특정 VPC와 연결

된 Amazon EC2 보안 그룹을 콘솔에서 프로그래밍 방식으로 관리할 수 있도록 허용을 참조하세요. 마지막으로가 AWS CLI설치된 컴퓨터에 업데이트된 JSON 파일을 저장합니다.

```
{
  "InstanceType": "kafka.m5.large",
  "ClientSubnets": [
    "Subnet-1-ID",
    "Subnet-2-ID"
  ],
  "SecurityGroups": [
    "Security-Group-ID"
  ]
}
```

> **Important**
>
> Express 브로커의 경우 3개의 서로 다른 가용 영역에 3개의 서브넷이 필요합니다. 또한 스토리지 관련 속성을 정의할 필요가 없습니다.
> 미국 서부(캘리포니아 북부) 리전의 Standard 브로커의 경우, 서로 다른 두 가용 영역에 두 개의 서브넷이 필요합니다. Amazon MSK를 사용할 수 있는 다른 모든 리전에서는 2개 또는 3개의 서브넷을 지정할 수 있습니다. 모든 서브넷은 서로 다른 가용 영역에 있어야 합니다. 클러스터를 생성하면 Amazon MSK는 지정한 서브넷에 브로커 노드를 균등하게 분배합니다.

2. `brokernodegroupinfo.json` 파일을 저장한 디렉터리에서 다음 AWS CLI명령을 실행하여 *"Your-Cluster-Name"*을 원하는 이름으로 바꿉니다. *"Monitoring-Level"*에 대해 DEFAULT, PER_BROKER 또는 PER_TOPIC_PER_BROKER, 세 값 중 하나를 지정할 수 있습니다. 이러한 세 가지 모니터링 수준에 대한 자세한 내용은 ??? 단원을 참조하십시오. `enhanced-monitoring` 파라미터는 선택 항목입니다. `create-cluster` 명령에서 지정하지 않으면 DEFAULT 모니터링 수준이 적용됩니다.

```
aws kafka create-cluster --cluster-name "Your-Cluster-Name" --broker-node-group-info file://brokernodegroupinfo.json --kafka-version "2.8.1" --number-of-broker-nodes 3 --enhanced-monitoring "Monitoring-Level"
```

명령의 출력은 다음 JSON과 같습니다.

```
{
    "ClusterArn": "...",
```

```
    "ClusterName": "AWSKafkaTutorialCluster",
    "State": "CREATING"
}
```

> **Note**
>
> 이 `create-cluster` 명령은 하나 이상의 서브넷이 지원되지 않는 가용 영역에 속한다는 오류를 반환할 수 있습니다. 이 경우 오류는 어떤 가용 영역이 지원되지 않는지 표시합니다. 지원되지 않는 가용 영역을 사용하지 않는 서브넷을 만들고, `create-cluster` 명령을 다시 시도합니다.

3. 클러스터에서 다른 작업을 수행하는 데 필요하므로 `ClusterArn` 키 값을 저장합니다.

4. 다음 명령을 실행하여 `STATE` 클러스터를 확인합니다. `STATE` 값은 Amazon MSK가 클러스터를 프로비저닝함에 따라 `CREATING`에서 `ACTIVE`으로 변경됩니다. 상태가 `ACTIVE`이면 클러스터에 연결할 수 있습니다. 클러스터 상태에 대한 자세한 내용은 MSK 프로비저닝된 클러스터의 상태 이해 섹션을 참조하세요.

```
aws kafka describe-cluster --cluster-arn <your-cluster-ARN>
```

를 사용하여 사용자 지정 Amazon MSK 구성으로 MSK 프로비저닝 클러스터 생성AWS CLI

사용자 지정 Amazon MSK 구성 및 생성 방법에 대한 자세한 내용은 the section called “브로커 구성” 섹션을 참조하세요.

1. 다음 JSON을 파일에 저장하고 *`configuration-arn`*을 클러스터를 생성하는 데 사용할 구성의 ARN으로 바꿉니다.

```
{
    "Arn": configuration-arn,
    "Revision": 1
}
```

2. `create-cluster` 명령을 실행하고 `configuration-info` 옵션을 사용하여 이전 단계에 저장한 JSON 파일을 가리킵니다. 다음은 예입니다.

```
aws kafka create-cluster --cluster-name ExampleClusterName --broker-node-group-
info file://brokernodegroupinfo.json --kafka-version "2.8.1" --number-of-broker-
nodes 3 --enhanced-monitoring PER_TOPIC_PER_BROKER --configuration-info file://
configuration.json
```

다음은 이 명령을 실행한 후 성공적인 응답의 예입니다.

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:123456789012:cluster/
CustomConfigExampleCluster/abcd1234-abcd-dcba-4321-a1b2abcd9f9f-2",
    "ClusterName": "CustomConfigExampleCluster",
    "State": "CREATING"
}
```

Amazon MSK API를 사용하여 MSK 프로비저닝된 클러스터 생성

Amazon MSK API를 사용하면 자동화된 인프라 프로비저닝 또는 배포 스크립트의 일부로 MSK 프로비저닝된 클러스터를 프로그래밍 방식으로 생성하고 관리할 수 있습니다.

API를 사용하여 MSK 프로비저닝된 클러스터를 생성하려면 CreateCluster를 참조하세요.

Amazon MSK 클러스터 나열

Amazon MSK 클러스터에 대한 부트스트랩 브로커를 가져오려면 클러스터 Amazon 리소스 이름(ARN)이 필요합니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. the section called “부트스트랩 브로커 가져오기”을(를) 참조하세요.

주제

- 를 사용하여 클러스터 나열AWS Management 콘솔
- 를 사용하여 클러스터 나열AWS CLI
- API를 사용하여 클러스터 나열

를 사용하여 클러스터 나열AWS Management 콘솔

Amazon MSK 클러스터에 대한 부트스트랩 브로커를 가져오려면 클러스터 Amazon 리소스 이름(ARN)이 필요합니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. the section called “부트스트랩 브로커 가져오기”을(를) 참조하세요.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 표에 이 계정에 속한 현재 리전의 모든 클러스터가 나열됩니다. 세부 정보를 보려면 클러스터의 이름을 선택합니다.

를 사용하여 클러스터 나열AWS CLI

Amazon MSK 클러스터에 대한 부트스트랩 브로커를 가져오려면 클러스터 Amazon 리소스 이름(ARN)이 필요합니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. the section called "부트스트랩 브로커 가져오기"을(를) 참조하세요.

```
aws kafka list-clusters
```

API를 사용하여 클러스터 나열

Amazon MSK 클러스터에 대한 부트스트랩 브로커를 가져오려면 클러스터 Amazon 리소스 이름(ARN)이 필요합니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. the section called "부트스트랩 브로커 가져오기"을(를) 참조하세요.

API를 사용하여 클러스터를 나열하려면 ListClusters를 참조하십시오.

Amazon MSK 프로비저닝된 클러스터에 연결

기본적으로 클라이언트는 클러스터와 동일한 VPC에 있는 경우에만 MSK 프로비저닝된 클러스터에 액세스할 수 있습니다. Kafka 클라이언트와 MSK 프로비저닝된 클러스터 간의 모든 통신은 기본적으로 프라이빗이며 스트리밍 데이터는 절대로 인터넷을 통과하지 않습니다. 클러스터와 동일한 VPC에 있는 클라이언트에서 MSK 프로비저닝된 클러스터에 연결하려면 클러스터의 보안 그룹에 클라이언트의 보안 그룹으로부터 트래픽을 허용하는 인바운드 규칙이 있는 것을 확인해야 합니다. 이러한 규칙 설정에 대한 자세한 내용은 보안 그룹 규칙을 참조하십시오. 클러스터와 동일한 VPC에 있는 Amazon EC2 인스턴스에서 클러스터에 액세스하는 방법의 예제는 the section called "시작하기" 섹션을 참조하세요.

> **Note**
>
> KRaft 메타데이터 모드와 MSK Express 브로커는 공개 모니터링과 퍼블릭 액세스 둘 다를 활성화할 수 없습니다.

클러스터의 VPC 외부에 있는 클라이언트에서 MSK 프로비저닝된 클러스터에 연결하려면 클러스터의 VPC 내부 AWS및 외부에서 액세스를 참조하세요.

주제

- MSK 프로비저닝된 클러스터에 대한 퍼블릭 액세스 활성화
- 클러스터의 VPC 내부 AWS및 외부에서 액세스

MSK 프로비저닝된 클러스터에 대한 퍼블릭 액세스 활성화

Amazon MSK는 Apache Kafka 2.6.0 이상 버전을 실행하는 MSK 프로비저닝된 클러스터의 브로커에 대한 퍼블릭 액세스를 활성화할 수 있는 옵션을 제공합니다. 보안상의 이유로 MSK 클러스터를 생성하는 동안에는 퍼블릭 액세스를 활성화할 수 없습니다. 그러나 기존 클러스터를 업데이트하여 공개적으로 액세스할 수 있도록 할 수 있습니다. 새 클러스터를 만든 다음 업데이트하여 공개적으로 액세스할 수 있도록 할 수도 있습니다.

추가 비용 없이 MSK 클러스터에 대한 퍼블릭 액세스를 켤 수 있지만 클러스터 내부 및 외부AWS로의 데이터 전송에는 표준 데이터 전송 비용이 적용됩니다. 가격 책정에 대한 자세한 내용은 Amazon EC2 온디맨드 가격 책정을 참조하세요.

> **Note**
>
> SASL/SCRAM 또는 mTLS 액세스 제어 방법을 사용하는 경우 먼저 클러스터에 대해 Apache Kafka ACL을 설정해야 합니다. 그런 다음 클러스터의 구성을 업데이트하여 `allow.everyone.if.no.acl.found` 속성이 false로 설정되도록 합니다. 클러스터 구성을 업데이트하는 방법에 대한 자세한 내용은 the section called “브로커 구성 작업” 섹션을 참조하세요.

MSK 프로비저닝된 클러스터에 대한 퍼블릭 액세스를 활성화하려면 먼저 클러스터가 다음 조건을 모두 충족하도록 해야 합니다.

- 클러스터와 연결된 서브넷은 퍼블릭이어야 합니다. 각 퍼블릭 서브넷에는 연결된 퍼블릭 IPv4 주소가 있으며, 퍼블릭 IPv4 주소는 Amazon VPC 요금 페이지에 표시된 대로 가격이 책정됩니다. 즉, 서브넷에는 인터넷 게이트웨이가 연결된 관련 라우팅 테이블이 있어야 합니다. 인터넷 게이트웨이를 생성하고 연결하는 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터넷 게이트웨이를 사용하여 VPC 인터넷 액세스 활성화를 참조하세요.

- 인증되지 않은 액세스 제어는 비활성화되어 있어야 하며 SASL/IAM, SASL/SCRAM, mTLS 액세스 제어 방법 중 하나 이상이 활성화되어 있어야 합니다. 클러스터의 액세스 제어 방법을 업데이트하는 방법에 대한 자세한 내용은 the section called “클러스터 보안 업데이트” 섹션을 참조하세요.
- 클러스터 내 암호화가 활성화되어 있어야 합니다. 클러스터를 생성할 때 기본 설정은 켜짐입니다. 암호화를 비활성화한 상태로 생성한 클러스터에 대해서는 클러스터 내에서 암호화를 활성화할 수 없습니다. 따라서 클러스터 내에서 암호화를 비활성화한 상태로 생성된 클러스터에 대해서는 퍼블릭 액세스를 활성화할 수 없습니다.
- 브로커와 클라이언트 간의 일반 텍스트 트래픽은 비활성화되어 있어야 합니다. 활성화되어 있는 경우 비활성화하는 방법에 대한 자세한 내용은 the section called “클러스터 보안 업데이트” 섹션을 참조하세요.
- IAM 액세스 제어를 사용 중이고 권한 부여 정책을 적용하거나 업데이트하려는 경우 the section called “IAM 액세스 제어” 섹션을 참조하세요. Apache Kafka ACL에 대한 자세한 내용은 the section called “Apache Kafka ACL” 섹션을 참조하세요.

MSK 클러스터가 위에 나열된 조건을 충족하는지 확인한 후AWS CLI, 또는 Amazon MSK APIAWS Management 콘솔를 사용하여 퍼블릭 액세스를 켤 수 있습니다. 클러스터에 대해 퍼블릭 액세스를 활성화한 후에는 클러스터에 대한 퍼블릭 부트스트랩 브로커 문자열을 가져올 수 있습니다. 클러스터에 대한 부트스트랩 브로커를 가져오는 방법에 대한 자세한 내용은 the section called “부트스트랩 브로커 가져오기” 섹션을 참조하세요.

> **Important**
>
> 퍼블릭 액세스를 활성화하는 것 외에도 클러스터의 보안 그룹에 내 IP 주소에서 퍼블릭 액세스를 허용하는 인바운드 TCP 규칙이 있는지 확인합니다. 이러한 규칙은 가능한 한 제한적으로 설정하는 것을 권장합니다. 보안 그룹 및 인바운드 규칙에 대한 자세한 내용은 Amazon VPC 사용 설명서에서 VPC의 보안 그룹을 참조하세요. 포트 번호는 the section called “포트 정보” 섹션을 참조하세요. 클러스터의 보안 그룹을 변경하는 방법에 대한 지침은 the section called “보안 그룹 변경” 섹션을 참조하세요.

> **Note**
>
> 다음 지침에 따라 퍼블릭 액세스를 활성화한 후에도 여전히 클러스터에 액세스할 수 없는 경우 the section called “퍼블릭 액세스가 활성화된 클러스터에 액세스할 수 없음” 섹션을 참조하세요.

콘솔을 사용하여 퍼블릭 액세스를 활성화하려면 다음을 수행합니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 클러스터 목록에서 퍼블릭 액세스를 활성화하려는 클러스터를 선택합니다.
3. 속성 탭을 선택한 다음 네트워크 설정 섹션을 찾습니다.
4. 퍼블릭 액세스 편집을 선택합니다.

를 사용하여 퍼블릭 액세스 켜기AWS CLI

1. 다음 AWS CLI명령을 실행하여 *ClusterArn* 및 *Current-Cluster-Version*을 클러스터의 ARN 및 현재 버전으로 바꿉니다. 클러스터의 현재 버전을 찾으려면 DescribeCluster 작업 또는 describe-clusterAWS CLI 명령을 사용합니다. 버전의 예를 들면 `KTVPDKIKX0DER`입니다.

```
aws kafka update-connectivity --cluster-arn ClusterArn --current-
version Current-Cluster-Version --connectivity-info '{"PublicAccess": {"Type":
 "SERVICE_PROVIDED_EIPS"}}'
```

이 `update-connectivity` 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

> **Note**
>
> 퍼블릭 액세스를 끄려면 비슷한 AWS CLI명령을 사용하되 대신 다음 연결 정보와 함께 사용합니다.
>
> ```
> '{"PublicAccess": {"Type": "DISABLED"}}'
> ```

2. update-connectivity 작업 결과를 가져오려면 다음 명령을 실행하여 *ClusterOperationArn*을 update-connectivity 명령의 출력에서 가져온 ARN으로 바꿉니다.

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

이 describe-cluster-operation 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "982168a3-939f-11e9-8a62-538df00285db",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-06-20T21:08:57.735Z",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_COMPLETE",
        "OperationType": "UPDATE_CONNECTIVITY",
        "SourceClusterInfo": {
            "ConnectivityInfo": {
                "PublicAccess": {
                    "Type": "DISABLED"
                }
            }
        },
        "TargetClusterInfo": {
            "ConnectivityInfo": {
                "PublicAccess": {
                    "Type": "SERVICE_PROVIDED_EIPS"
                }
            }
        }
    }
}
```

OperationState 값이 UPDATE_IN_PROGRESS인 경우, 잠시 기다린 다음 describe-cluster-operation 명령을 다시 실행합니다.

Amazon MSK API를 사용하여 퍼블릭 액세스를 활성화하려면 다음을 수행합니다.

- API를 사용하여 클러스터에 대한 퍼블릭 액세스를 켜거나 끄려면 UpdateConnectivity를 참조하세요.

> **Note**
> 보안상의 이유로 Amazon MSK는 Apache ZooKeeper 또는 켜기 컨트롤러 노드에 대한 퍼블릭 액세스를 허용하지 않습니다.

클러스터의 VPC 내부 AWS및 외부에서 액세스

클러스터의 Amazon VPC 내부 AWS및 외부에서 MSK 클러스터에 연결하려면 다음 옵션이 있습니다.

Amazon VPC 피어링

클러스터의 VPC와 다른 VPC에서 MSK 클러스터에 연결하려면 두 VPC 간에 피어링 연결을 생성하면 됩니다. VPC 피어링에 대한 자세한 내용은 Amazon VPC 피어링 안내서를 참조하십시오.

Direct Connect

Direct Connect는 표준 1기가비트 또는 10기가비트 이더넷 광섬유 케이블을 AWS통해 온프레미스 네트워크를에 연결합니다. 케이블의 한쪽 끝은 라우터에 연결되고 다른 쪽 끝은 Direct Connect라우터에 연결됩니다. 이 연결을 사용하면 네트워크 경로에서 인터넷 서비스 공급자를 우회하여 AWS클라우드 및 Amazon VPC에 직접 가상 인터페이스를 생성할 수 있습니다. 자세한 내용은 Direct Connect 단원을 참조하십시오.

AWS Transit Gateway

AWS Transit Gateway는 VPCs와 온프레미스 네트워크를 단일 게이트웨이에 연결할 수 있는 서비스입니다. AWS Transit Gateway 사용 방법에 대한 자세한 내용은 AWS Transit Gateway 단원을 참조하십시오.

VPN 연결

VPN 연결 주제에 설명된 VPN 연결 옵션을 사용하여 MSK 클러스터의 VPC를 원격 네트워크 및 사용자에 연결할 수 있습니다.

REST 프록시

클러스터의 Amazon VPC 내에서 실행 중인 인스턴스에 REST 프록시를 설치할 수 있습니다. REST 프록시를 사용하면 생산자와 소비자가 HTTP API 요청을 통해 클러스터와 통신할 수 있습니다.

다중 리전 다중 VPC 연결

다중 리전 다중 VPC 연결 문서에서는 서로 다른 리전에 상주하는 다중 VPC에 대한 연결 옵션을 설명합니다.

단일 리전 다중 VPC 프라이빗 연결

Amazon Managed Streaming for Apache Kafka(Amazon MSK) 클러스터에 대한 다중 VPC 프라이빗 연결(AWSPrivateLink 제공)은 다양한 Virtual Private Cloud(VPCs) 및 AWS계정에서 호스팅되는 Kafka 클라이언트를 Amazon MSK 클러스터에 더 빠르게 연결할 수 있는 기능입니다.

크로스 계정 클라이언트를 위한 단일 리전 다중 VPC 연결을 참조하세요.

EC2-Classic 네트워킹이 사용 중지됨

Amazon MSK는 더 이상 Amazon EC2-Classic 네트워킹에서 실행되는 Amazon EC2 인스턴스를 지원하지 않습니다.

EC2-Classic 네트워킹은 사용 중지 중 - 준비 방법은 다음과 같습니다를 참조하세요.

단일 리전에서의 Amazon MSK 다중 VPC 프라이빗 연결

Amazon Managed Streaming for Apache Kafka(Amazon MSK) 클러스터에 대한 다중 VPC 프라이빗 연결(AWSPrivateLink 제공)은 다양한 Virtual Private Cloud(VPCs) 및 AWS계정에서 호스팅되는 Kafka 클라이언트를 Amazon MSK 클러스터에 더 빠르게 연결할 수 있는 기능입니다.

다중 VPC 프라이빗 연결은 다중 VPC 및 크로스 계정 연결을 위한 네트워킹 인프라를 간소화하는 관리형 솔루션입니다. 클라이언트는 AWS네트워크 내에서 모든 트래픽을 유지하면서 PrivateLink를 통해 Amazon MSK 클러스터에 연결할 수 있습니다. Amazon MSK 클러스터에 대한 다중 VPC 프라이빗 연결은 Amazon MSK를 사용할 수 있는 모든 AWS리전에서 사용할 수 있습니다.

주제

- 다중 VPC 프라이빗 연결이란 무엇인가요?
- 다중 VPC 프라이빗 연결의 이점
- 다중 VPC 프라이빗 연결에 대한 요구 사항과 제한
- 다중 VPC 프라이빗 연결을 사용하여 시작하기

- 클러스터에서 권한 부여 체계 업데이트
- Amazon MSK 클러스터에 대한 관리형 VPC 연결 거부
- Amazon MSK 클러스터에 대한 관리형 VPC 연결 삭제
- 다중 VPC 프라이빗 연결에 대한 권한

다중 VPC 프라이빗 연결이란 무엇인가요?

Amazon MSK에 대한 다중 VPC 프라이빗 연결은 다양한 Virtual Private Cloud(VPCs) 및 AWS계정에서 호스팅되는 Apache Kafka 클라이언트를 MSK 클러스터에 연결할 수 있는 연결 옵션입니다.

Amazon MSK는 클러스터 정책으로 크로스 계정 액세스를 간소화합니다. 이러한 정책을 통해 클러스터 소유자는 다른 AWS계정에 MSK 클러스터에 대한 프라이빗 연결을 설정할 수 있는 권한을 부여할 수 있습니다.

다중 VPC 프라이빗 연결의 이점

다중 VPC 프라이빗 연결은 다른 연결 솔루션에 비해 몇 가지 이점이 있습니다.

- 프라이빗 AWSPrivateLink 연결 솔루션의 운영 관리를 자동화합니다.
- 이 솔루션은 연결된 VPC 간 IP 중복을 허용하므로 다른 VPC 연결 솔루션과 관련된 중복되지 않는 IP, 복잡한 피어링, 라우팅 테이블을 유지할 필요가 없습니다.

MSK 클러스터에 대한 클러스터 정책을 사용하여 MSK 클러스터에 대한 교차 계정 프라이빗 연결을 설정할 권한이 있는 AWS계정을 정의합니다. 크로스 계정 관리자는 적절한 역할 또는 사용자에게 권한을 위임할 수 있습니다. IAM 클라이언트 인증과 함께 사용하는 경우 클러스터 정책을 사용하여 연결 클라이언트에 대해 세분화된 단위로 Kafka 데이터 영역 권한을 정의할 수도 있습니다.

다중 VPC 프라이빗 연결에 대한 요구 사항과 제한

다중 VPC 프라이빗 연결을 실행하려면 다음 MSK 클러스터 요구 사항을 참고하세요.

- 다중 VPC 프라이빗 연결은 Apache Kafka 2.7.1 이상에서만 지원됩니다. MSK 클러스터와 함께 사용하는 모든 클라이언트가 클러스터와 호환되는 Apache Kafka 버전을 실행하고 있는지 확인합니다.
- 다중 VPC 프라이빗 연결은 인증 유형 IAM, TLS, SASL/SCRAM을 지원합니다. 인증되지 않은 클러스터는 다중 VPC 프라이빗 연결을 사용할 수 없습니다.
- SASL/SCRAM 또는 mTLS 액세스 제어 방법을 사용하는 경우 클러스터에 대해 Apache Kafka ACL을 설정해야 합니다. 먼저 클러스터에 대한 Apache Kafka ACL을 설정합니다. 그런 다음 클러스터의

구성을 업데이트하여 클러스터에 대해 `allow.everyone.if.no.acl.found` 속성이 false로 설정되도록 합니다. 클러스터 구성을 업데이트하는 방법에 대한 자세한 내용은 the section called “브로커 구성 작업” 섹션을 참조하세요. IAM 액세스 제어를 사용 중이고 권한 부여 정책을 적용하거나 업데이트하려는 경우 the section called “IAM 액세스 제어” 섹션을 참조하세요. Apache Kafka ACL에 대한 자세한 내용은 the section called “Apache Kafka ACL” 섹션을 참조하세요.

- 다중 VPC 프라이빗 연결은 t3.small 인스턴스 유형을 지원하지 않습니다.
- 다중 VPC 프라이빗 연결은 AWS리전 간에 지원되지 않으며 동일한 리전 내의 AWS계정에서만 지원됩니다.
- 다중 VPC 프라이빗 연결을 설정하려면 클러스터 서브넷과 동일한 수의 클라이언트 서브넷이 있어야 합니다. 또한 클라이언트 서브넷과 클러스터 서브넷의 가용 영역 ID가 동일한지 확인해야 합니다.
- Amazon MSK는 Zookeeper 노드에 대한 다중 VPC 프라이빗 연결을 지원하지 않습니다.

다중 VPC 프라이빗 연결을 사용하여 시작하기

주제

이 자습서에서는 다중 VPC 연결을 사용하여 Apache Kafka 클라이언트를 클러스터의 내부 AWS및 VPC 외부에서 MSK 클러스터에 비공개로 연결하는 방법의 예로 일반적인 사용 사례를 사용합니다. 이 프로세스를 수행하려면 크로스 계정 사용자가 필요한 클라이언트 권한을 포함하여 각 클라이언트에 대한 MSK 관리형 VPC 연결 및 구성을 생성해야 합니다. 또한 이 프로세스에서는 MSK 클러스터 소유자가 MSK 클러스터에서 PrivateLink 연결을 활성화하고 클러스터에 대한 액세스를 제어하기 위해 인증 체계를 선택해야 합니다.

이 자습서의 다른 부분에서는 이 예제에 적용되는 옵션을 선택합니다. 이 옵션이 MSK 클러스터 또는 클라이언트 인스턴스를 설정하는 데 사용되는 유일한 옵션이라는 의미는 아닙니다.

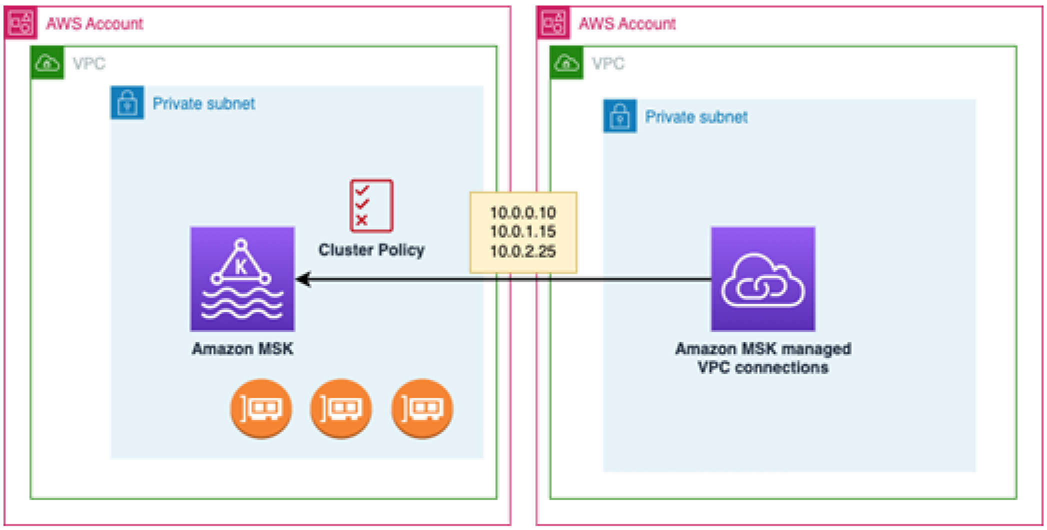
이 사용 사례의 네트워크 구성은 다음과 같습니다.

- 크로스 계정 사용자(Kafka 클라이언트)와 MSK 클러스터가 동일한 AWS 네트워크/리전에 있지만 서로 다른 계정에 있습니다.
  - 계정 A의 MSK 클러스터
  - 계정 B의 Kafka 클라이언트

- 크로스 계정 사용자는 IAM 인증 체계를 사용하여 MSK 클러스터에 비공개로 연결합니다.

이 자습서에서는 Apache Kafka 버전 2.7.1 이상으로 생성된 프로비저닝된 MSK 클러스터가 있다고 가정합니다. 구성 프로세스를 시작하기 전에 MSK 클러스터가 활성 상태여야 합니다. 잠재적인 데이터 손실이나 가동 중지를 방지하기 위해 다중 VPC 프라이빗 연결을 사용하여 클러스터에 연결할 클라이언트는 클러스터와 호환되는 Apache Kafka 버전을 사용해야 합니다.

다음 다이어그램은 다른 AWS계정의 클라이언트에 연결된 Amazon MSK 다중 VPC 연결의 아키텍처를 보여줍니다.



1단계: 계정 A의 MSK 클러스터에서 클러스터의 IAM 인증 체계를 위한 다중 VPC 연결을 활성화

MSK 클러스터 소유자는 클러스터가 생성된 후 활성 상태에서 MSK 클러스터에 대한 구성 설정을 해야 합니다.

클러스터 소유자는 클러스터에서 활성화될 모든 인증 체계에 대해 활성 클러스터에서 다중 VPC 프라이빗 연결을 활성화합니다. 이는 UpdateSecurity API 또는 MSK 콘솔을 사용하여 수행할 수 있습니다. 다중 VPC 프라이빗 연결은 여러 인증 체계 SASL/SCRAM, IAM, TLS를 지원합니다. 인증되지 않은 클러스터에는 다중 VPC 프라이빗 연결을 활성화할 수 없습니다.

이 사용 사례에서는 IAM 인증 체계를 사용하도록 클러스터를 구성합니다.

> **Note**
>
> SASL/SCRAM 인증 체계를 사용하도록 MSK 클러스터를 구성하는 경우 Apache Kafka ACL 속성 “`allow.everyone.if.no.acl.found=false`”는 필수입니다. Apache Kafka ACLs를 참조하세요.

다중 VPC 프라이빗 연결 설정을 업데이트하면 Amazon MSK는 브로커 구성을 업데이트하는 브로커 노드의 롤링 재부팅을 시작합니다. 이 작업을 완료하는 데 최대 30분 이상 소요될 수 있습니다. 연결이 업데이트되는 동안에는 클러스터에 다른 업데이트를 수행할 수 없습니다.

콘솔을 사용하여 계정 A의 클러스터에서 선택한 인증 체계에 대해 다중 VPC를 활성화합니다.

1. https://console.aws.amazon.com/msk/에서 클러스터가 있는 계정에 대해 Amazon MSK 콘솔을 엽니다.
2. 탐색 창의 MSK 클러스터에서 클러스터를 선택하여 계정의 클러스터 목록을 표시합니다.
3. 다중 VPC 프라이빗 연결을 구성할 클러스터를 선택합니다. 클러스터는 활성 상태여야 합니다.
4. 클러스터 속성 탭을 선택한 다음 네트워크 설정으로 이동합니다.
5. 편집 드롭다운 메뉴를 선택하고 다중 VPC 연결 켜기를 선택합니다.
6. 이 클러스터에 대해 활성화하려는 인증 유형을 하나 이상 선택합니다. 이 사용 사례에서는 IAM 역할 기반 인증을 선택합니다.
7. 변경 사항 저장을 선택합니다.

Example - 클러스터에서 다중 VPC 프라이빗 연결 인증 체계를 활성화하는 UpdateConnectivity API

MSK 콘솔 대신 UpdateConnectivity API를 사용하여 다중 VPC 프라이빗 연결을 활성화하고 활성 클러스터에서 인증 체계를 구성할 수 있습니다. 다음 예제는 클러스터에 대해 설정된 IAM 인증 체계를 보여 줍니다.

```
{
  "currentVersion": "K3T4TT2Z381HKD",
  "connectivityInfo": {
    "vpcConnectivity": {
      "clientAuthentication": {
        "sasl": {
          "iam": {
            "enabled": TRUE
            }
        }
      }
    }
  }
}
```

Amazon MSK는 프라이빗 연결에 필요한 네트워킹 인프라를 생성합니다. 또한 Amazon MSK는 프라이빗 연결이 필요한 각 인증 유형에 대해 새 부트스트랩 브로커 엔드포인트 세트를 생성합니다. 일반 텍스트 인증 체계는 다중 VPC 프라이빗 연결을 지원하지 않는다는 점에 유의하세요.

2단계: MSK 클러스터에 클러스터 정책 연결

클러스터 소유자는 클러스터 정책(리소스 기반 정책이라고도 함)을 MSK 클러스터에 연결하여 다중 VPC 프라이빗 연결을 활성화할 수 있습니다. 클러스터 정책은 클라이언트가 다른 계정에서 클러스터에 액세스할 수 있는 권한을 부여합니다. 클러스터 정책을 편집하기 전에 MSK 클러스터에 액세스할 수 있는 권한이 있어야 하는 계정의 계정 ID가 필요합니다. Amazon MSK가 IAM과 작동하는 방식을 참조하세요.

클러스터 소유자는 계정 B의 크로스 계정 사용자가 클러스터에 대한 부트스트랩 브로커를 가져오고 계정 A의 MSK 클러스터에서 다음 작업을 할 수 있도록 권한을 부여하는 클러스터 정책을 MSK 클러스터에 연결해야 합니다.

- CreateVpcConnection
- GetBootstrapBrokers
- DescribeCluster
- DescribeClusterV2

Example

참고로 다음은 MSK 콘솔 IAM 정책 편집기에 표시되는 기본 정책과 유사한 기본 클러스터 정책에 대한 JSON의 예제입니다. 다음 정책은 클러스터, 주제 및 그룹 수준 액세스에 대한 권한을 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012"
      },
      "Action": [
        "kafka:CreateVpcConnection",
        "kafka:GetBootstrapBrokers",
```

```
          "kafka:DescribeCluster",
          "kafka:DescribeClusterV2",
          "kafka-cluster:*"
        ],
        "Resource": "arn:aws:kafka:us-east-1:111122223333:cluster/testing/
de8982fa-8222-4e87-8b20-9bf3cdfa1521-2"
      },
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "123456789012"
        },
        "Action": "kafka-cluster:*",
        "Resource": "arn:aws:kafka:us-east-1:111122223333:topic/testing/*"
      },
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "123456789012"
        },
        "Action": "kafka-cluster:*",
        "Resource": "arn:aws:kafka:us-east-1:111122223333:group/testing/*"
      }
    ]
}
```

MSK 클러스터에 클러스터 정책 연결

1. Amazon MSK 콘솔의 MSK 클러스터에서 클러스터를 선택합니다.
2. 보안 설정까지 아래로 스크롤하여 클러스터 편집 정책을 선택합니다.
3. 콘솔의 클러스터 정책 편집 화면에서 다중 VPC 연결을 위한 기본 정책을 선택합니다.
4. 계정 ID 필드에 이 클러스터에 액세스할 수 있는 권한이 있어야 하는 각 계정의 계정 ID를 입력합니다. ID를 입력하면 ID가 표시된 정책 JSON 구문에 자동으로 복사됩니다. 예제 클러스터 정책에서 계정 ID는 *111122223333*입니다.
5. 변경 사항 저장을 선택합니다.

클러스터 정책 API에 대한 자세한 내용은 Amazon MSK 리소스 기반 정책을 참조하세요.

3단계: 크로스 계정 사용자 작업을 통해 클라이언트 관리형 VPC 연결 구성

MSK 클러스터와 다른 계정의 클라이언트 간에 다중 VPC 프라이빗 연결을 설정하려면 크로스 계정 사용자가 클라이언트에 대한 관리형 VPC 연결을 생성합니다. 이 절차를 반복하여 여러 클라이언트를 MSK 클러스터에 연결할 수 있습니다. 이 사용 사례에서는 하나의 클라이언트만 구성합니다.

클라이언트는 지원되는 인증 체계인 IAM, SASL/SCRAM 또는 TLS를 사용할 수 있습니다. 각 관리형 VPC 연결에 하나의 인증 체계만 연결할 수 있습니다. 클라이언트 인증 체계는 클라이언트가 연결할 MSK 클러스터에서 구성해야 합니다

이 사용 사례에서는 계정 B의 클라이언트가 IAM 인증 체계를 사용하도록 클라이언트 인증 체계를 구성합니다.

사전 조건

이 프로세스에는 다음 항목이 필요합니다.

- 이전에 만든 클러스터 정책으로, 계정 B의 클라이언트가 계정 A의 MSK 클러스터에서 작업을 수행할 수 있는 권한을 부여합니다.
- `kafka:CreateVpcConnection`, `ec2:CreateTags`, `ec2:CreateVPCEndpoint` 및 `ec2:DescribeVpcAttribute` 작업에 대한 권한을 부여하는 계정 B의 클라이언트에 연결된 자격 증명 정책입니다.

Example

참고로 다음은 기본 클라이언트 ID 정책에 대한 JSON의 예제입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kafka:CreateVpcConnection",
        "ec2:CreateTags",
        "ec2:CreateVPCEndpoint",
        "ec2:DescribeVpcAttribute"
      ],
```

```
            "Resource": "*"
        }
    ]
}
```

계정 B에서 클라이언트를 위한 관리형 VPC 연결을 만들려면 다음을 수행합니다.

1. 클러스터 관리자로부터 계정 B의 클라이언트가 연결할 계정 A에 있는 MSK 클러스터의 클러스터 ARN을 가져옵니다. 나중에 사용할 클러스터 ARN을 기록해 둡니다.
2. 클라이언트 계정 B의 MSK 콘솔에서 관리형 VPC 연결을 선택한 다음 연결 생성을 선택합니다.
3. 연결 설정 창에서 클러스터 ARN을 클러스터 ARN 텍스트 필드에 붙여넣은 다음 확인을 선택합니다.
4. 계정 B에서 클라이언트에 대한 인증 유형을 선택합니다. 이 사용 사례의 경우 클라이언트 VPC 연결을 생성할 때 IAM을 선택합니다.
5. 클라이언트에 대한 VPC를 선택합니다.
6. 최소 2개의 가용 영역과 관련 서브넷을 선택합니다. AWSManagement Console 클러스터 세부 정보에서 또는 DescribeCluster API 또는 describe-clusterAWS CLI 명령을 사용하여 가용 영역 IDs를 가져올 수 있습니다. 클라이언트 서브넷에 지정하는 영역 ID는 클러스터 서브넷의 영역 ID와 일치해야 합니다. 서브넷의 값이 누락된 경우 먼저 MSK 클러스터와 동일한 영역 ID를 가진 서브넷을 생성합니다.
7. 이 VPC 연결에 사용할 보안 그룹을 선택합니다. 기본 보안 그룹을 사용할 수 있습니다. 보안 그룹 구성에 대한 자세한 내용은 보안 그룹을 사용하여 리소스에 대한 트래픽 제어를 참조하세요.
8. 연결 생성을 선택합니다.
9. 크로스 계정 사용자의 MSK 콘솔(클러스터 세부 정보 > 관리형 VPC 연결)에서 새 부트스트랩 브로커 문자열 목록을 가져오려면 “클러스터 연결 문자열” 아래에 표시된 부트스트랩 브로커 문자열을 참조하세요. 클라이언트 계정 B에서 부트스트랩 브로커 목록은 GetBootstrapBrokers API를 호출하거나 콘솔 클러스터 세부 정보에서 부트스트랩 브로커 목록을 확인하여 볼 수 있습니다.
10. 다음과 같이 VPC 연결과 관련된 보안 그룹을 업데이트합니다.

    a. 계정 B 네트워크의 IP 범위에 대한 모든 트래픽을 허용하도록 PrivateLink VPC에 대한 인바운드 규칙을 설정합니다.

    b. [선택 사항] MSK 클러스터에 대한 아웃바운드 규칙 연결을 설정합니다. VPC 콘솔에서 보안 그룹을 선택하고 아웃바운드 규칙 편집을 선택한 다음 포트 범위 14001~14100에 대한 사용자 지정 TCP 트래픽에 대한 규칙을 추가합니다. 다중 VPC Network Load Balancer는 14001~14100 포트 범위에서 수신 대기합니다. Network Load Balancers를 참조하세요.

11. 다중 VPC 프라이빗 연결을 위한 새 부트스트랩 브로커를 사용하여 계정 A의 MSK 클러스터에 연결하도록 계정 B의 클라이언트를 구성합니다. [데이터 생성 및 소비]를 참조하세요.

인증이 완료되면 Amazon MSK는 지정된 각 VPC 및 인증 체계에 대해 관리형 VPC 연결을 생성합니다. 선택한 보안 그룹이 각 연결과 관련이 있습니다. 이 관리형 VPC 연결은 Amazon MSK에서 브로커에 비공개로 연결하도록 구성합니다. 새로운 부트스트랩 브로커 세트를 사용하여 Amazon MSK 클러스터에 비공개로 연결할 수 있습니다.

클러스터에서 권한 부여 체계 업데이트

다중 VPC 프라이빗 연결은 여러 인증 체계 SASL/SCRAM, IAM, TLS를 지원합니다. 클러스터 소유자는 하나 이상의 인증 체계에 대한 프라이빗 연결을 활성화/비활성화할 수 있습니다. 이 작업을 수행하려면 클러스터가 활성 상태여야 합니다.

Amazon MSK 콘솔을 사용하여 인증 체계를 사용 설정하려면 다음을 수행합니다.

1. 편집하려는 클러스터에 대해 [AWS Management 콘솔]에서 Amazon MSK 콘솔을 엽니다.
2. 탐색 창의 MSK 클러스터에서 클러스터를 선택하여 계정의 클러스터 목록을 표시합니다.
3. 편집하려는 클러스터를 선택합니다. 클러스터는 활성 상태여야 합니다.
4. 클러스터 속성 탭을 선택한 다음 네트워크 설정으로 이동합니다.
5. 새 인증 체계를 활성화하려면 편집 드롭다운 메뉴를 선택하고 다중 VPC 연결 켜기를 선택합니다.
6. 이 클러스터에 대해 활성화하려는 인증 유형을 하나 이상 선택합니다.
7. 선택 켜기를 선택합니다.

새 인증 체계를 사용 설정할 때 새 인증 체계에 대한 관리형 VPC 연결도 새로 만들고 클라이언트가 새 인증 체계에 특정한 부트스트랩 브로커를 사용하도록 업데이트해야 합니다.

Amazon MSK 콘솔을 사용하여 인증 체계를 비활성화하려면 다음을 수행합니다.

> **Note**
>
> 인증 체계에 대한 다중 VPC 프라이빗 연결을 비활성화하면 관리형 VPC 연결을 포함한 모든 연결 관련 인프라가 삭제됩니다.

인증 체계에 대한 다중 VPC 프라이빗 연결을 비활성화하면 클라이언트 측의 기존 VPC 연결이 비활성 상태로 변경되고 클러스터 측의 관리형 VPC 연결을 포함한 클러스터 측의 PrivateLink 인프라가 제거

됩니다. 크로스 계정 사용자는 비활성 VPC 연결만 삭제할 수 있습니다. 클러스터에서 프라이빗 연결이 다시 활성화된 경우 크로스 계정 사용자는 클러스터에 대한 새 연결을 생성해야 합니다.

1. AWS Management 콘솔에서 Amazon MSK 콘솔을 엽니다.
2. 탐색 창의 MSK 클러스터에서 클러스터를 선택하여 계정의 클러스터 목록을 표시합니다.
3. 편집하려는 클러스터를 선택합니다. 클러스터는 활성 상태여야 합니다.
4. 클러스터 속성 탭을 선택한 다음 네트워크 설정으로 이동합니다.
5. 인증 체계를 비활성화하려면 편집 드롭다운 메뉴를 선택하고 다중 VPC 연결 끄기를 선택합니다.
6. 이 클러스터에 대해 비활성화하려는 인증 유형을 하나 이상 선택합니다.
7. 선택 끄기를 선택합니다.

Example API를 사용하여 인증 체계를 켜거나 끄려면 다음을 수행합니다.

MSK 콘솔 대신 UpdateConnectivity API를 사용하여 다중 VPC 프라이빗 연결을 활성화하고 활성 클러스터에서 인증 체계를 구성할 수 있습니다. 다음 예제는 클러스터에 설정된 SASL/SCRAM 및 IAM 인증 체계를 보여줍니다.

새 인증 체계를 사용 설정할 때 새 인증 체계에 대한 관리형 VPC 연결도 새로 만들고 클라이언트가 새 인증 체계에 특정한 부트스트랩 브로커를 사용하도록 업데이트해야 합니다.

인증 체계에 대한 다중 VPC 프라이빗 연결을 비활성화하면 클라이언트 측의 기존 VPC 연결이 비활성 상태로 변경되고 관리형 VPC 연결을 포함하여 클러스터 측의 PrivateLink 인프라가 제거됩니다. 크로스 계정 사용자는 비활성 VPC 연결만 삭제할 수 있습니다. 클러스터에서 프라이빗 연결이 다시 활성화된 경우 크로스 계정 사용자는 클러스터에 대한 새 연결을 생성해야 합니다.

```
Request:
{
  "currentVersion": "string",
  "connnectivityInfo": {
    "publicAccess": {
      "type": "string"
    },
    "vpcConnectivity": {
      "clientAuthentication": {
        "sasl": {
          "scram": {
            "enabled": TRUE
          },
          "iam": {
```

```
              "enabled": TRUE
            }
          },
          "tls": {
            "enabled": FALSE
          }
        }
      }
    }
}

Response:
{
  "clusterArn": "string",
  "clusterOperationArn": "string"
}
```

Amazon MSK 클러스터에 대한 관리형 VPC 연결 거부

클러스터 관리자 계정의 Amazon MSK 콘솔에서 클라이언트 VPC 연결을 거부할 수 있습니다. 클라이언트 VPC 연결이 거부되려면 사용 가능 상태여야 합니다. 더 이상 클러스터에 연결할 수 있는 권한이 없는 클라이언트의 관리형 VPC 연결을 거부할 수 있습니다. 클라이언트에 연결되는 새 관리형 VPC 연결을 방지하려면 클러스터 정책에서 클라이언트에 대한 액세스를 거부합니다. 거부된 연결은 연결 소유자가 삭제할 때까지 비용이 계속 발생합니다. Amazon MSK 클러스터에 대한 관리형 VPC 연결 삭제를 참조하세요.

MSK 콘솔을 사용하여 클라이언트 VPC 연결을 거부하려면 다음을 수행합니다.

1. AWS Management 콘솔에서 Amazon MSK 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택하고 네트워크 설정 > 클라이언트 VPC 연결 목록으로 스크롤합니다.
3. 거부하려는 연결을 선택하고 클라이언트 VPC 연결 거부를 선택합니다.
4. 선택한 클라이언트 VPC 연결을 거부할지 확인합니다.

API를 사용하여 관리형 VPC 연결을 거부하려면 `RejectClientVpcConnection` API를 사용합니다.

Amazon MSK 클러스터에 대한 관리형 VPC 연결 삭제

크로스 계정 사용자는 클라이언트 계정 콘솔에서 MSK 클러스터에 대한 관리형 VPC 연결을 삭제할 수 있습니다. 클러스터를 소유한 사용자는 관리형 VPC 연결을 소유하지 않으므로 클러스터 관리자 계정에서 연결을 삭제할 수 없습니다. VPC 연결이 삭제되면 더 이상 비용이 발생하지 않습니다.

MSK 콘솔을 사용하여 관리형 VPC 연결을 삭제하려면 다음을 수행합니다.

1. 클라이언트 계정의 AWS Management 콘솔에서 Amazon MSK 콘솔을 엽니다.
2. 탐색 창에서 관리형 VPC 연결을 선택합니다.
3. 연결 목록에서 삭제하려는 연결을 선택합니다.
4. VPC 연결을 삭제할 것인지 확인합니다.

API를 사용해 관리형 VPC 연결을 삭제하려면 `DeleteVpcConnection` API를 사용합니다.

다중 VPC 프라이빗 연결에 대한 권한

이 섹션에서는 다중 VPC 프라이빗 연결 기능을 사용하는 클라이언트와 클러스터에 필요한 권한을 요약합니다. 다중 VPC 프라이빗 연결은 클라이언트 관리자가 MSK 클러스터에 대한 관리형 VPC 연결이 가능한 각 클라이언트에 대한 권한을 생성해야 합니다. 또한 MSK 클러스터 관리자는 MSK 클러스터에서 PrivateLink 연결을 활성화하고 인증 체계를 선택하여 클러스터에 대한 액세스를 제어해야 합니다.

클러스터 인증 유형 및 주제 액세스 권한

MSK 클러스터에서 활성화된 인증 체계에 대해 다중 VPC 프라이빗 연결 기능을 사용 설정합니다. 다중 VPC 프라이빗 연결에 대한 요구 사항과 제한을 참조하세요. SASL/SCRAM 인증 체계를 사용하도록 MSK 클러스터를 구성하는 경우 Apache Kafka ACL 속성 `allow.everyone.if.no.acl.found=false`는 필수입니다. 클러스터에 대해 Apache Kafka ACL을 설정한 후 클러스터의 구성을 업데이트하여 클러스터에 대해 `allow.everyone.if.no.acl.found` 속성이 false로 설정되도록 합니다. 클러스터 구성을 업데이트하는 방법에 대한 자세한 내용은 브로커 구성 작업 섹션을 참조하세요.

크로스 계정 클러스터 정책 권한

Kafka 클라이언트가 MSK 클러스터와 다른 AWS계정에 있는 경우 클라이언트 루트 사용자에게 교차 계정 연결을 승인하는 클러스터 기반 정책을 MSK 클러스터에 연결합니다. MSK 콘솔의 IAM 정책 편집기(클러스터 보안 설정 > 클러스터 정책 편집)를 사용하여 다중 VPC 클러스터 정책을 편집하거나 다음 API를 사용하여 클러스터 정책을 관리할 수 있습니다.

PutClusterPolicy

    클러스터 정책을 클러스터에 연결합니다. 이 API를 사용하여 지정된 MSK 클러스터 정책을 생성하거나 업데이트할 수 있습니다. 정책을 업데이트하는 경우 요청 페이로드에 currentVersion 필드가 필수입니다.

GetClusterPolicy

클러스터에 연결된 클러스터 정책 문서의 JSON 텍스트를 검색합니다.

DeleteClusterPolicy

클러스터 정책을 삭제합니다.

다음은 MSK 콘솔 IAM 정책 편집기에 표시된 것과 유사한 기본 클러스터 정책에 대한 JSON의 예제입니다. 다음 정책은 클러스터, 주제 및 그룹 수준 액세스에 대한 권한을 부여합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "AWS": [
                "123456789012"
            ]
        },
        "Action": [
            "kafka-cluster:*",
            "kafka:CreateVpcConnection",
            "kafka:GetBootstrapBrokers",
            "kafka:DescribeCluster",
            "kafka:DescribeClusterV2"
        ],
        "Resource": [
            "arn:aws:kafka:us-east-1:123456789012:cluster/testing/
de8982fa-8222-4e87-8b20-9bf3cdfa1521-2",
            "arn:aws:kafka:us-east-1:123456789012:topic/testing/*",
            "arn:aws:kafka:us-east-1:123456789012:group/testing/*"
        ]
    }]
}
```

MSK 클러스터에 대한 다중 VPC 프라이빗 연결을 위한 클라이언트 권한

Kafka 클라이언트와 MSK 클러스터 간에 다중 VPC 프라이빗 연결을 설정하려면 클라이언트에 대한 `kafka:CreateVpcConnection`, `ec2:CreateTags`, `ec2:CreateVPCEndpoint` 작업에 대한 권한을 부여하는 연결된 ID 정책이 필요합니다. 참고로 다음은 기본 클라이언트 ID 정책에 대한 JSON의 예제입니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kafka:CreateVpcConnection",
        "ec2:CreateTags",
        "ec2:CreateVPCEndpoint"
      ],
      "Resource": "*"
    }
  ]
}
```

포트 정보

Amazon MSK가 클라이언트 머신과 통신할 수 있도록 다음 포트 번호를 사용합니다.

- 브로커와 일반 텍스트로 통신하려면 포트 9092를 사용합니다.
- TLS 암호화를 사용하여 브로커와 통신하려면 내부에서 액세스하는 경우 포트 9094를 사용하고 퍼블릭 액세스의 경우 AWS포트 9194를 사용합니다.
- SASL/SCRAM을 사용하여 브로커와 통신하려면 포트 9096을 내부에서 액세스하고 AWS포트 9196을 퍼블릭 액세스에 사용합니다.
- 를 사용하도록 설정된 클러스터의 브로커와 통신하려면 내부 액세스에는 포트 9098the section called “IAM 액세스 제어”을 사용하고 퍼블릭 액세스에는 AWS포트 9198을 사용합니다.
- TLS 암호화를 사용하여 Apache ZooKeeper와 통신하려면 포트 2182를 사용합니다. Apache ZooKeeper 노드는 기본적으로 포트 2181을 사용합니다.

Amazon MSK 클러스터를 위한 부트스트랩 브로커 가져오기

부트스트랩 브로커라는 용어는 Apache Kafka 클라이언트가 Amazon MSK 클러스터에 연결하는 데 사용할 수 있는 브로커 목록을 의미합니다. 이 목록에 클러스터의 모든 브로커가 포함되지 않을 수 있습니다. AWS CLI, 또는 Amazon MSK API를 사용하여 부트스트랩 AWS Management 콘솔브로커를 가져올 수 있습니다.

주제

를 사용하여 부트스트랩 브로커 가져오기AWS Management 콘솔

이 프로세스는를 사용하여 클러스터의 부트스트랩 브로커를 가져오는 방법을 설명합니다AWS Management 콘솔. 부트스트랩 브로커라는 용어는 Apache Kafka 클라이언트가 클러스터에 연결하기 위한 시작점으로 사용할 수 있는 브로커 목록을 의미합니다. 이 목록에 클러스터의 모든 브로커가 포함되어 있는 것은 아닙니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 표에 이 계정에 속한 현재 리전의 모든 클러스터가 나열됩니다. 설명을 보려면 클러스터의 이름을 선택합니다.
3. 클러스터 요약 페이지에서 클라이언트 정보 보기를 선택합니다. 여기에는 부트스트랩 브로커와 Apache ZooKeeper 연결 문자열이 표시됩니다.

를 사용하여 부트스트랩 브로커 가져오기AWS CLI

다음 명령을 실행하여 *ClusterArn*을 클러스터 생성 후 받은 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.

```
aws kafka get-bootstrap-brokers --cluster-arn ClusterArn
```

the section called “IAM 액세스 제어”을 사용하는 MSK 클러스터의 경우 해당 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "BootstrapBrokerStringSaslIam": "b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9098,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9098"
}
```

다음 예제는 퍼블릭 액세스가 설정된 클러스터의 부트스트랩 브로커를 보여줍니다. 퍼블릭 액세스`BootstrapBrokerStringPublicSaslIam`에는를 사용하고 내부에서 액세스에는 `BootstrapBrokerStringSaslIam` 문자열을 사용합니다AWS.

```
{
    "BootstrapBrokerStringPublicSaslIam": "b-2-public.myTestCluster.v4ni96.c2.kafka-
beta.us-east-1.amazonaws.com:9198,b-1-public.myTestCluster.v4ni96.c2.kafka-
beta.us-east-1.amazonaws.com:9198,b-3-public.myTestCluster.v4ni96.c2.kafka-beta.us-
east-1.amazonaws.com:9198",
    "BootstrapBrokerStringSaslIam": "b-2.myTestCluster.v4ni96.c2.kafka-
beta.us-east-1.amazonaws.com:9098,b-1.myTestCluster.v4ni96.c2.kafka-beta.us-
east-1.amazonaws.com:9098,b-3.myTestCluster.v4ni96.c2.kafka-beta.us-
east-1.amazonaws.com:9098"
}
```

부트스트랩 브로커 문자열에는 MSK 클러스터가 배포된 가용 영역의 브로커 3개가 포함되어야 합니다(사용 가능한 브로커가 2개만 있는 경우 제외).

API를 사용하여 부트스트랩 브로커 가져오기

API를 사용하여 부트스트랩 브로커를 가져오려면 GetBootstrapBrokers를 참조하십시오.

Amazon MSK 프로비저닝된 클러스터 모니터링

Amazon MSK를 사용하여 Amazon MSK 프로비저닝된 클러스터의 상태를 모니터링하는 데 도움이 되는 몇 가지 방법이 있습니다.

- Amazon MSK는 Apache Kafka 지표를 수집하여 사용자가 볼 수 있는 Amazon CloudWatch로 전송합니다. Amazon MSK가 표시하는 지표를 포함하여 Apache Kafka 지표에 대한 자세한 내용은 Apache Kafka 설명서에서 모니터링을 참조하세요.
- 또한 오픈 소스 모니터링 애플리케이션인 Prometheus로 MSK 클러스터를 모니터링할 수도 있습니다. Prometheus에 대한 자세한 내용은 Prometheus 설명서의 Overview를 참조하십시오. Prometheus를 사용하여 MSK 프로비저닝된 클러스터를 모니터링하는 방법은 the section called “Prometheus를 사용하여 모니터링” 섹션을 참조하세요.

- (Standard 브로커 전용) Amazon MSK는 프로비저닝된 클러스터가 스토리지 용량 한도에 도달하려할 때 스토리지 용량 알림을 자동으로 전송하여 디스크 스토리지 용량을 모니터링할 수 있도록 도와줍니다. 또한 알림은 탐지된 문제를 해결할 수 있는 최선의 조치에 대한 권장 사항을 제공합니다. 이를 통해 디스크 용량 문제가 심각해지기 전에 이를 식별하여 신속하게 해결할 수 있습니다. Amazon MSK는 이러한 알림을 AWS계정의 Amazon MSK 콘솔Health Dashboard, Amazon EventBridge 및 이메일 연락처로 자동으로 전송합니다. 스토리지 용량 알림에 대한 자세한 내용은 Amazon MSK 스토리지 용량 알림 사용 섹션을 참조하세요.

주제

CloudWatch를 사용하여 Amazon MSK 지표 보기

CloudWatch 콘솔, 명령줄 또는 CloudWatch API를 사용하여 Amazon MSK에 대한 지표를 모니터링할 수 있습니다. 다음의 절차는 이처럼 다양한 방법을 사용하여 측정치에 액세스하는 방법을 설명합니다.

CloudWatch 콘솔을 사용하여 지표에 액세스

에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.

1. 탐색 창에서 지표(Metrics)를 선택합니다.
2. 모든 지표 탭을 선택한 다음 AWS/Kafka를 선택합니다.
3. 주제 수준 지표를 보려면 주제, 브로커 ID, 클러스터 이름을 선택하고 브로커 수준 지표의 경우 브로커 ID, 클러스터 이름을 선택하고 클러스터 수준 지표의 경우 클러스터 이름을 선택합니다.
4. (선택 사항) 그래프 창에서 통계와 기간을 선택한 후 해당 설정을 사용하여 CloudWatch 경보를 생성합니다.

를 사용하여 지표에 액세스하려면AWS CLI

list-metrics 명령과 get-metric-statistics 명령을 사용합니다.

CloudWatch 콘솔을 사용하여 지표에 액세스

mon-list-metrics 명령과 mon-get-stats 명령을 사용합니다.

CloudWatch API를 사용하여 지표에 액세스

ListMetrics 작업과 GetMetricStatistics 작업을 사용합니다.

CloudWatch를 이용한 Standard 브로커 모니터링을 위한 Amazon MSK 지표

Amazon MSK는 Amazon CloudWatch와 통합되므로 MSK Standard 브로커에 대한 CloudWatch 지표를 수집, 조회, 분석할 수 있습니다. MSK 프로비저닝된 클러스터에 대해 구성한 지표는 1분 간격으로 자동으로 수집되어 CloudWatch로 푸시됩니다. MSK 프로비저닝된 클러스터의 모니터링 수준을 DEFAULT, PER_BROKER, PER_TOPIC_PER_BROKER 또는 PER_TOPIC_PER_PARTITION 중 하나로 설정할 수 있습니다. 다음 섹션의 표에는 각 모니터링 수준부터 사용할 수 있는 모든 지표가 나와 있습니다.

> **Note**
>
> CloudWatch 모니터링을 위한 일부 Amazon MSK 지표의 이름이 버전 3.6.0 이상에서 변경되었습니다. 해당 지표를 모니터링할 때 새 이름을 사용하세요. 이름이 변경된 지표의 경우 아래 표에 버전 3.6.0 이상에서 사용된 이름과 버전 2.8.2.tired에서 사용된 이름이 차례로 나와 있습니다.

DEFAULT 수준 지표는 무료입니다. 다른 지표에 대한 요금은 Amazon CloudWatch 요금 페이지에 설명되어 있습니다.

DEFAULT 수준 모니터링

다음 표에 설명된 지표는 DEFAULT 모니터링 수준에서 사용할 수 있습니다. 무료로 제공됩니다.

이름	표시되는 경우	측정 기준	설명
ActiveControllerCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름	클러스터당 오직 하나의 컨트롤러만 지정된 시간에 활성화되어야 합니다.

이름	표시되는 경우	측정 기준	설명
`BurstBalance`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	클러스터의 EBS 볼륨에 대한 입력-출력 버스트 크레딧의 남은 잔액. 지연 시간 또는 처리량 감소를 조사하는 데 사용합니다. 볼륨의 기준 성능이 최대 버스트 성능보다 높은 경우 EBS 볼륨에 대해 `BurstBalance` 가 보고되지 않습니다. 자세한 내용은 I/O 크레딧 및 버스트 성능을 참조하세요.
`BytesInPerSec`	주제를 생성한 후.	클러스터 이름, 브로커 ID, 주제	클라이언트로부터 받은 초당 바이트 수입니다. 이 지표는 브로커별 및 주제별로 제공됩니다.
`BytesOutPerSec`	주제를 생성한 후.	클러스터 이름, 브로커 ID, 주제	클라이언트에 전송된 초당 바이트 수입니다. 이 지표는 브로커별 및 주제별로 제공됩니다.
`ClientConnectionCount`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID, 클라이언트 인증	인증된 활성 클라이언트 연결 수입니다.

이름	표시되는 경우	측정 기준	설명
`ConnectionCount`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	활성 인증, 미인증 및 브로커 간 연결 수입니다.
`CPUCreditBalance`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	노드 실행 이후 노드가 누적한 획득 CPU 크레딧 수입니다. 크레딧은 획득 이후에 크레딧 밸런스에 누적되고, 소비 시 크레딧 밸런스에서 소멸됩니다. CPU 크레딧 밸런스가 부족해져 클러스터 성능에 부정적인 영향을 미칠 수 있습니다. CPU 부하를 줄이기 위한 조치를 취할 수 있습니다. 예를 들어 클라이언트 요청 수를 줄이거나 브로커 유형을 M5 브로커 유형으로 업데이트할 수 있습니다.
`CpuIdle`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	CPU 유휴 시간의 백분율입니다.
`CpuIoWait`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	보류 중인 디스크 작업 중 CPU 유휴 시간의 백분율입니다.

이름	표시되는 경우	측정 기준	설명
CpuSystem	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	커널 공간에 있는 CPU의 백분율입니다.
CpuUser	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	사용자 공간에 있는 CPU의 백분율입니다.
GlobalPartitionCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름	클러스터의 모든 주제에서 복제본을 제외한 파티션 수입니다. GlobalPartitionCount 에는 복제본이 포함되지 않으므로 주제의 복제 인수가 1보다 큰 경우 PartitionCount 값의 합계가 GlobalPartitionCount보다 클 수 있습니다.
GlobalTopicCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름	클러스터의 모든 브로커에 있는 총 주제 수입니다.
EstimatedMaxTimeLag	컨슈머 그룹이 주제에서 컨슘한 후.	클러스터 이름, 컨슈머 그룹, 주제	MaxOffsetLag 를 배출하는 데 걸리는 예상 시간(초)입니다.

이름	표시되는 경우	측정 기준	설명
KafkaAppLogsDiskUsed	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	애플리케이션 로그에 사용된 디스크 공간의 백분율입니다.
KafkaDataLogsDiskUsed (Cluster Name, Broker ID 차원)	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	데이터 로그에 사용된 디스크 공간의 백분율입니다.
LeaderCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커당 총 파티션 리더 수(복제본 제외)입니다.
MaxOffsetLag	컨슈머 그룹이 주제에서 컨슘한 후.	클러스터 이름, 컨슈머 그룹, 주제	주제의 모든 파티션에 대한 최대 오프셋 지연.
MemoryBuffered	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대한 버퍼링된 메모리의 크기(바이트)입니다.

이름	표시되는 경우	측정 기준	설명
MemoryCached	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대한 캐시 메모리의 크기(바이트)입니다.
MemoryFree	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 사용할 수 있는 메모리의 크기(바이트)입니다.
HeapMemoryAfterGC	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	가비지 수집 이후 사용된 총 힙 메모리의 백분율입니다.
MemoryUsed	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에서 사용 중인 메모리의 크기(바이트)입니다.
MessagesInPerSec	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커의 초당 수신 메시지 수입니다.
NetworkRxDropped	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	삭제된 수신 패키지의 수입니다.

이름	표시되는 경우	측정 기준	설명
`NetworkRxErrors`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대한 네트워크 수신 오류 수입니다.
`NetworkRxPackets`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에서 수신된 패킷 수입니다.
`NetworkTxDropped`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	삭제된 전송 패키지의 수입니다.
`NetworkTxErrors`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커의 네트워크 전송 오류 수입니다.
`NetworkTxPackets`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커가 전송한 패킷 수입니다.
`OfflinePartitionsCount`	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름	클러스터에서 오프라인 상태인 총 파티션 수입니다.

이름	표시되는 경우	측정 기준	설명
PartitionCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	복제본을 포함하여 브로커당 주제 파티션의 총 수입니다.
ProduceTotalTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	밀리초 단위의 평균 프로덕션 시간.
RequestBytesMean	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대한 요청 바이트의 평균 수입니다.
RequestTime	요청 스로틀링이 적용된 후.	클러스터 이름, 브로커 ID	브로커 네트워크 및 I/O 스레드가 요청을 처리하는 데 걸린 평균 시간(밀리초)입니다.
RootDiskUsed	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커가 사용하는 루트 디스크의 백분율입니다.

이름	표시되는 경우	측정 기준	설명
SumOffsetLag	컨슈머 그룹이 주제에서 컨슘한 후.	클러스터 이름, 컨슈머 그룹, 주제	주제의 모든 파티션에 대한 집계된 오프셋 지연
SwapFree	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에서 사용할 수 있는 스왑 메모리의 크기(바이트)입니다.
SwapUsed	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에서 사용 중인 스왑 메모리의 크기(바이트)입니다.
TrafficShaping	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	네트워크 할당 초과로 인해 형성(삭제 또는 대기열에 추가)된 패킷 수를 나타내는 상위 수준 지표입니다. PER_BROKER 지표를 사용하면 더 자세한 내용을 확인할 수 있습니다.
UnderMinIsrPartitionCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커의 minIsr 파티션 수입니다.

이름	표시되는 경우	측정 기준	설명
UnderReplicatedPartitions	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대해 복제가 덜 진행된 파티션 수입니다.
UserPartitionExists	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 사용자 소유 파티션이 있음을 나타내는 부울 지표입니다. 값이 1이면 브로커에 파티션이 있음을 나타냅니다.
ZooKeeperRequestLatencyMsMean	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	ZooKeeper 기반 클러스터용. 브로커에서 Apache ZooKeeper 요청에 대한 평균 대기 시간(밀리초)입니다.
ZooKeeperSessionState	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	ZooKeeper 기반 클러스터용. 다음 중 하나일 수 있는 브로커의 ZooKeeper 세션의 연결 상태: NOT_CONNECTED: '0.0', ASSOCIATING: '0.1', CONNECTING: '0.5', CONNECTEDREADONLY: '0.8', CONNECTED: '1.0', CLOSED: '5.0', AUTH_FAILED: '10.0'.

PER_BROKER 수준 모니터링

모니터링 수준을 PER_BROKER로 설정하면 모든 DEFAULT 수준 지표에 추가로 다음 표에 설명된 지표가 표시됩니다. 다음 표의 지표에 해당하는 비용을 지불하지만 DEFAULT 수준 지표는 계속 무료로 제공됩니다. 이 표에 있는 지표의 측정 기준은 클러스터 이름, 브로커 ID입니다.

이름	표시되는 경우	설명
`BwInAllowanceExceeded`	클러스터가 ACTIVE 상태에 도달한 후.	인바운드 집계 대역폭이 브로커의 최대치를 초과하여 형성된 패킷 수입니다.
`BwOutAllowanceExceeded`	클러스터가 ACTIVE 상태에 도달한 후.	아웃바운드 집계 대역폭이 브로커의 최대치를 초과하여 형성된 패킷 수입니다.
`ConntrackAllowanceExceeded`	클러스터가 ACTIVE 상태에 도달한 후.	연결 추적이 브로커의 최대치를 초과하여 형성된 패킷 수입니다. 연결 추적은 설정된 각 연결을 추적하여 반환 패킷이 예상대로 전달되도록 하는 보안 그룹과 관련이 있습니다.
`ConnectionCloseRate`	클러스터가 ACTIVE 상태에 도달한 후.	리스너당 초당 닫힌 연결 수입니다. 이 숫자는 리스너별로 집계되며 클라이언트 리스너에 대해 필터링됩니다.
`ConnectionCreationRate`	클러스터가 ACTIVE 상태에 도달한 후.	리스너당 초당 설정된 새 연결 수입니다. 이 숫자는 리스너별로 집계되며 클라이언트 리스너에 대해 필터링됩니다.
`CpuCreditUsage`	클러스터가 ACTIVE 상태에 도달한 후.	브로커가 소비하는 CPU 크레딧 수입니다. CPU 크레딧 밸런스가 부족해져 클러스터 성능에 부정적인 영향을 미칠 수 있습니다. CPU 부하를 줄이기 위한 조치를 취할 수 있습니다. 예를 들어 클라이언트 요청 수를 줄이거나 브로커 유형을 M5 브로커 유형으로 업데이트할 수 있습니다.
`FetchConsumerLocalTimeMsMean`	프로듀서/컨슈머가 만들어진 이후.	컨슈머 요청이 리더에서 처리되는 데 걸리는 평균 시간(밀리초)입니다.
`FetchConsumerRequestQueueTimeMsMean`	프로듀서/컨슈머가 만들어진 이후.	컨슈머 요청이 요청 대기열에서 대기하는 평균 시간(밀리초)입니다.

이름	표시되는 경우	설명
FetchConsumerResponseQueueTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	컨슈머 요청이 응답 대기열에서 대기하는 평균 시간(밀리초)입니다.
FetchConsumerResponseSendTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	컨슈머가 응답을 보내는 평균 시간(밀리초)입니다.
FetchConsumerTotalTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	컨슈머가 브로커에서 데이터를 가져오는 데 걸리는 평균 총 시간(밀리초)입니다.
FetchFollowerLocalTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어 요청이 리더에서 처리되는 데 걸리는 평균 시간(밀리초)입니다.
FetchFollowerRequestQueueTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어 요청이 요청 대기열에서 대기하는 평균 시간(밀리초)입니다.
FetchFollowerResponseQueueTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어 요청이 응답 대기열에서 대기하는 평균 시간(밀리초)입니다.
FetchFollowerResponseSendTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어가 응답을 보내는 평균 시간(밀리초)입니다.
FetchFollowerTotalTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어가 브로커에서 데이터를 가져오는 데 걸리는 평균 총 시간(밀리초)입니다.
FetchMessageConversionsPerSec	주제를 생성한 후.	브로커의 초당 가져오기 메시지 변환 횟수입니다.
FetchThrottleByteRate	대역폭 스로틀링이 적용된 후.	초당 스로틀링된 바이트 수입니다.
FetchThrottleQueueSize	대역폭 스로틀링이 적용된 후.	스로틀 대기열에 있는 메시지 수입니다.
FetchThrottleTime	대역폭 스로틀링이 적용된 후.	평균 가져오기 스로틀링 시간(밀리초)입니다.

이름	표시되는 경우	설명
IAMNumberOfConnectionRequests	클러스터가 ACTIVE 상태에 도달한 후.	초당 IAM 인증 요청의 수입니다.
IAMTooManyConnections	클러스터가 ACTIVE 상태에 도달한 후.	100을 초과하여 시도된 연결 수입니다. 0은 연결 수가 한도 내에 있음을 의미합니다. 0보다 큰 경우 스로틀 한도가 초과되므로 연결 수를 줄여야 합니다.
NetworkProcessorAvgIdlePercent	클러스터가 ACTIVE 상태에 도달한 후.	네트워크 프로세서가 유휴 상태인 시간의 평균 백분율입니다.
PpsAllowanceExceeded	클러스터가 ACTIVE 상태에 도달한 후.	양방향 PPS가 브로커의 최대치를 초과하여 형성된 패킷 수입니다.
ProduceLocalTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	리더에서 요청이 처리되는 데 걸리는 평균 시간(밀리초)입니다.
ProduceMessageConversionsPerSec	주제를 생성한 후.	브로커의 초당 생산 메시지 변환 수입니다.
ProduceMessageConversionsTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	메시지 형식 변환에 소요된 평균 시간(밀리초)입니다.
ProduceRequestQueueTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	요청 메시지가 대기열에서 소비하는 평균 시간(밀리초)입니다.
ProduceResponseQueueTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	응답 메시지가 대기열에서 소비하는 평균 시간(밀리초)입니다.
ProduceResponseSendTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	응답 메시지를 보내는 데 걸린 평균 시간(밀리초)입니다.
ProduceThrottleByteRate	대역폭 스로틀링이 적용된 후.	초당 스로틀링된 바이트 수입니다.
ProduceThrottleQueueSize	대역폭 스로틀링이 적용된 후.	스로틀 대기열에 있는 메시지 수입니다.

이름	표시되는 경우	설명
ProduceThrottleTime	대역폭 스로틀링이 적용된 후.	평균 프로덕션 스로틀링 시간(밀리초)입니다.
ProduceTotalTimeMs Mean	클러스터가 ACTIVE 상태에 도달한 후.	밀리초 단위의 평균 프로덕션 시간.
RemoteFetchBytesPerSec (RemoteBytesInPerSec in v2.8.2.tiered)	프로듀서/컨슈머가 생성된 후.	소비자 가져오기에 대한 응답으로 계층형 스토리지에서 전송된 총 바이트 수입니다. 이 지표에는 다운스트림 데이터 전송 트래픽에 기여하는 모든 주제 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
RemoteCopyBytesPerSec (RemoteBytesOutPerSec in v2.8.2.tiered)	생산자/소비자가 생성된 후.	로그 세그먼트, 인덱스, 기타 보조 파일의 데이터를 포함하여 계층화된 스토리지로 전송된 총 바이트 수입니다. 이 지표에는 업스트림 데이터 전송 트래픽에 기여하는 모든 토픽 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
RemoteLogManagerTasksAvgIdlePercent	클러스터가 ACTIVE 상태에 도달한 후.	원격 로그 관리자가 유휴 상태로 보낸 평균 시간 비율입니다. 원격 로그 관리자는 브로커에서 계층화된 스토리지로 데이터를 전송합니다. 범주는 내부 활동입니다. 이는 KIP-405 지표입니다.
RemoteLogReaderAvgIdlePercent	클러스터가 ACTIVE 상태에 도달한 후.	원격 로그 리더가 유휴 상태로 보낸 평균 시간 비율입니다. 원격 로그 리더는 소비자의 가져오기에 대한 응답으로 원격 스토리지에서 브로커로 데이터를 전송합니다. 범주는 내부 활동입니다. 이는 KIP-405 지표입니다.

이름	표시되는 경우	설명
RemoteLogReaderTaskQueueSize	클러스터가 ACTIVE 상태에 도달한 후.	예약 대기 중인 계층형 스토리지에서 읽기를 담당하는 작업의 수입니다. 범주는 내부 활동입니다. 이는 KIP-405 지표입니다.
RemoteFetchErrorsPerSec (RemoteReadErrorPerSec in v2.8.2.tiered)	클러스터가 ACTIVE 상태에 도달한 후.	지정된 브로커가 소비자 가져오기에 대한 응답으로 데이터를 검색하기 위해 계층화된 스토리지로 전송한 읽기 요청에 대한 총 오류 발생율입니다. 이 지표에는 다운스트림 데이터 전송 트래픽에 영향을 미치는 모든 주제 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
RemoteFetchRequestsPerSec (RemoteReadRequestsPerSec in v2.8.2.tiered)	클러스터가 ACTIVE 상태에 도달한 후.	지정한 브로커가 소비자 가져오기에 대한 응답으로 데이터를 검색하기 위해 계층화된 스토리지로 전송한 총 읽기 요청 수입니다. 이 지표에는 다운스트림 데이터 전송 트래픽에 영향을 미치는 모든 주제 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
RemoteCopyErrorsPerSec (RemoteWriteErrorPerSec in v2.8.2.tiered)	클러스터가 ACTIVE 상태에 도달한 후.	지정된 브로커가 데이터를 업스트림으로 전송하기 위해 계층화된 스토리지로 전송한 쓰기 요청에 대한 응답으로 발생한 총 오류 발생율입니다. 이 지표에는 업스트림 데이터 전송 트래픽에 영향을 미치는 모든 주제 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.

이름	표시되는 경우	설명
`RemoteLogSizeBytes`	클러스터가 ACTIVE 상태에 도달한 후.	원격 계층에 저장된 바이트 수입니다. 이 지표는 Amazon MSK의 Apache Kafka 버전 3.7.x의 계층형 스토리지 클러스터에 사용할 수 있습니다.
`ReplicationBytesInPerSec`	주제를 생성한 후.	다른 브로커로부터 수신하는 초당 바이트 수입니다.
`ReplicationBytesOutPerSec`	주제를 생성한 후.	다른 브로커로 전송되는 초당 바이트 수입니다.
`RequestExemptFromThrottleTime`	요청 스로틀링이 적용된 후.	브로커 네트워크 및 I/O 스레드가 스로틀링에서 제외된 요청을 처리하는 데 걸린 평균 시간(밀리초)입니다.
`RequestHandlerAvgIdlePercent`	클러스터가 ACTIVE 상태에 도달한 후.	요청 핸들러 스레드가 유휴 상태인 시간의 평균 백분율입니다.
`RequestThrottleQueueSize`	요청 스로틀링이 적용된 후.	스로틀 대기열에 있는 메시지 수입니다.
`RequestThrottleTime`	요청 스로틀링이 적용된 후.	평균 요청 스로틀링 시간(밀리초)입니다.
`TcpConnections`	클러스터가 ACTIVE 상태에 도달한 후.	SYN 플래그가 설정된 수신 및 발신 TCP 세그먼트 수를 표시합니다.
`RemoteCopyLagBytes (TotalTierBytesLag in v2.8.2.tiered)`	주제를 생성한 후.	브로커에서 계층화할 수 있지만 아직 계층화된 스토리지로 전송되지 않은 데이터의 총 바이트 수입니다. 이 지표는 업스트림 데이터 전송의 효율성을 보여줍니다. 지연이 증가하면 계층형 스토리지에 유지되지 않는 데이터의 양이 증가합니다. 범주는 아카이브 지연입니다. 이것은 KIP-405 지표가 아닙니다.

이름	표시되는 경우	설명
`TrafficBytes`	클러스터가 ACTIVE 상태에 도달한 후.	클라이언트(생산자 및 소비자)와 브로커 간의 네트워크 트래픽을 전체 바이트 단위로 표시합니다. 브로커 사이의 트래픽은 보고되지 않습니다.
`VolumeQueueLength`	클러스터가 ACTIVE 상태에 도달한 후.	지정된 기간 동안 완료되기를 기다리는 읽기 및 쓰기 작업 요청의 수입니다.
`VolumeReadBytes`	클러스터가 ACTIVE 상태에 도달한 후.	지정된 기간 동안 읽은 바이트 수입니다.
`VolumeReadOps`	클러스터가 ACTIVE 상태에 도달한 후.	지정된 기간 동안의 읽기 작업 횟수입니다.
`VolumeTotalReadTime`	클러스터가 ACTIVE 상태에 도달한 후.	지정된 기간에 완료된 모든 읽기 작업에 소요된 총 시간(초)입니다.
`VolumeTotalWriteTime`	클러스터가 ACTIVE 상태에 도달한 후.	지정된 기간에 완료된 모든 쓰기 작업에 소요된 총 시간(초)입니다.
`VolumeWriteBytes`	클러스터가 ACTIVE 상태에 도달한 후.	지정된 기간 동안 기록된 바이트 수입니다.
`VolumeWriteOps`	클러스터가 ACTIVE 상태에 도달한 후.	지정된 기간 동안의 쓰기 작업 횟수입니다.

`PER_TOPIC_PER_BROKER` 수준 모니터링

모니터링 수준을 `PER_TOPIC_PER_BROKER`로 설정하면 모든 `PER_BROKER` 및 기본 수준 지표 외에 다음 표에 설명된 지표가 표시됩니다. `DEFAULT` 수준 지표만 무료입니다. 이 표에 있는 지표의 측정 기준은 클러스터 이름, 브로커 ID, 주제입니다.

> **Important**
>
> Apache Kafka 2.4.1 이상 버전을 사용하는 Amazon MSK 클러스터의 경우 다음 표의 지표는 해당 값이 처음으로 0이 아닌 값이 된 후에만 나타납니다. 예를 들어, `BytesInPerSec`를 보려면 하나 이상의 생산자가 먼저 클러스터로 데이터를 전송해야 합니다.

이름	표시되는 경우	설명
`FetchMessageConversionsPerSec`	주제를 생성한 후.	초당 가져와서 변환한 메시지 수입니다.
`MessagesInPerSec`	주제를 생성한 후.	초당 수신된 메시지 수입니다.
`ProduceMessageConversionsPerSec`	주제를 생성한 후.	생산된 메시지의 초당 변환 수입니다.
`RemoteFetchBytesPerSec (RemoteBytesInPerSec in v2.8.2.tiered)`	주제 생성 후 해당 주제가 프로듀싱/컨슈밍되고 있는 경우.	지정된 주제 및 브로커에 대한 소비자 가져오기에 대한 응답으로 계층형 스토리지에서 전송된 바이트 수입니다. 이 지표에는 지정된 브로커의 다운스트림 데이터 전송 트래픽에 기여하는 주제의 모든 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
`RemoteCopyBytesPerSec (RemoteBytesOutPerSec in v2.8.2.tiered)`	주제 생성 후 해당 주제가 프로듀싱/컨슈밍되고 있는 경우.	지정된 주제 및 브로커에 대해 계층형 스토리지로 전송된 바이트 수입니다. 이 지표에는 지정된 브로커의 업스트림 데이터 전송 트래픽에 기여하는 주제의 모든 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
`RemoteFetchErrorsPerSec (RemoteReadErrorPerSec in v2.8.2.tiered)`	주제 생성 후 해당 주제가 프로듀싱/컨슈밍되고 있는 경우.	지정된 주제에 대한 소비자 가져오기에 대한 응답으로 데이터를 검색하기 위해 지정된 브로커가 계층화된 스토리지로 보내는 읽기 요청에 대한 응답으로 발생하는 오류의 비율입니다. 이 지표에는 지정된 브로커의 다운스트림 데이터 전송

이름	표시되는 경우	설명
		트래픽에 기여하는 주제의 모든 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
RemoteFetchRequestsPerSec (RemoteReadRequestsPerSec in v2.8.2.tiered)	주제 생성 후 해당 주제가 프로듀싱/컨슈밍되고 있는 경우.	지정된 주제에 대한 소비자 가져오기에 대한 응답으로 데이터를 검색하기 위해 지정한 브로커가 계층화된 스토리지로 보내는 읽기 요청의 수입니다. 이 지표에는 지정된 브로커의 다운스트림 데이터 전송 트래픽에 기여하는 주제의 모든 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
RemoteCopyErrorsPerSec (RemoteWriteErrorPerSec in v2.8.2.tiered)	주제 생성 후 해당 주제가 프로듀싱/컨슈밍되고 있는 경우.	지정된 브로커가 데이터를 업스트림으로 전송하기 위해 계층형 스토리지로 보내는 쓰기 요청에 대한 응답으로 발생하는 오류의 비율입니다. 이 지표에는 지정된 브로커의 업스트림 데이터 전송 트래픽에 기여하는 주제의 모든 파티션이 포함되어 있습니다. 범주는 트래픽 및 오류 발생율입니다. 이는 KIP-405 지표입니다.
RemoteLogSizeBytes	주제를 생성한 후.	원격 계층에 저장된 바이트 수입니다. 이 지표는 Amazon MSK의 Apache Kafka 버전 3.7.x의 계층형 스토리지 클러스터에 사용할 수 있습니다.

PER_TOPIC_PER_PARTITION 수준 모니터링

모니터링 수준을 PER_TOPIC_PER_PARTITION로 설정하면 모든 PER_TOPIC_PER_BROKER, PER_BROKER, 기본 수준 지표 외에 다음 표에 설명된 지표가 표시됩니다. DEFAULT 수준 지표만 무료입니다. 이 표에 있는 지표의 측정 기준은 컨슈머 그룹, 주제, 파티션입니다.

이름	표시되는 경우	설명
`EstimatedTimeLag`	컨슈머 그룹이 주제에서 컨숨한 후.	파티션 오프셋 지연을 배출하는 데 걸리는 예상 시간(초)입니다.
`OffsetLag`	컨슈머 그룹이 주제에서 컨숨한 후.	파티션 수준 컨슈머 지연의 오프셋 수입니다.

MSK 프로비저닝된 클러스터의 상태 이해

다음 표에는 MSK 프로비저닝된 클러스터가 나타낼 수 있는 상태가 표시되어 있으며, 각 상태가 의미하는 바가 설명되어 있습니다. 달리 지정하지 않는 한 MSK 프로비저닝 클러스터 상태는 Standard 및 Express 브로커 유형 모두에 적용됩니다. 이 표는 또한 MSK 프로비저닝된 클러스터가 이러한 상태 중 하나일 때 수행할 수 있는 작업과 수행할 수 없는 작업에 대해서도 설명합니다. 클러스터의 상태를 확인하려면 AWS Management 콘솔을 방문합니다. describe-cluster-v2 명령 또는 DescribeClusterV2 작업을 사용하여 프로비저닝된 클러스터를 설명할 수도 있습니다. 클러스터의 설명에는 클러스터의 상태가 포함되어 있습니다.

MSK 프로비저닝된 클러스터 상태	의미 및 가능한 조치
ACTIVE	데이터를 생산하고 소비할 수 있습니다. 클러스터에서 Amazon MSK API 및 AWS CLI작업을 수행할 수도 있습니다.
CREATING	Amazon MSK가 프로비저닝된 클러스터를 설정하고 있습니다. 클러스터를 사용하여 데이터를 생성 또는 소비하거나 클러스터에서 Amazon MSK API 또는 AWS CLI작업을 수행하려면 클러스터가 ACTIVE 상태에 도달할 때까지 기다려야 합니다.
DELETING	프로비저닝된 클러스터를 삭제 중입니다. 데이터를 생산하거나 소비하는 데 사용할 수 없습니다. 또한 Amazon MSK API 또는 AWS CLI작업도 수행할 수 없습니다.

MSK 프로비저닝된 클러스터 상태	의미 및 가능한 조치
FAILED	프로비저닝된 클러스터 생성 또는 삭제 프로세스가 실패했습니다. 클러스터를 사용하여 데이터를 생산하거나 소비할 수 없습니다. 클러스터를 삭제할 수 있지만 클러스터에서 Amazon MSK API 또는 AWS CLI업데이트 작업을 수행할 수는 없습니다.
복구	Amazon MSK는 비정상 브로커를 교체하는 등의 내부 작업을 실행하고 있습니다. 예를 들어 브로커가 응답하지 않을 수 있습니다. 여전히 프로비저닝된 클러스터를 사용하여 데이터를 프로듀싱하고 컨슘할 수 있습니다. 그러나 클러스터가 ACTIVE 상태로 돌아올 때까지는 클러스터에서 Amazon MSK API 또는 AWS CLI업데이트 작업을 수행할 수 없습니다.
유지 관리	(Standard 브로커 전용) Amazon MSK는 클러스터에서 정기적인 유지 관리 작업을 수행하고 있습니다. 이러한 유지 관리 작업에는 보안 패치가 포함됩니다. 여전히 클러스터를 사용하여 데이터를 생성하고 소비할 수 있습니다. 그러나 클러스터가 ACTIVE 상태로 돌아갈 때까지 클러스터에서 Amazon MSK API 또는 AWSCLI 업데이트 작업을 수행할 수 없습니다. Express 브로커를 유지 관리하는 동안 클러스터 상태는 활성 상태로 유지됩니다. [MSK Provisioned 클러스터에 패치 적용](#)을 참조하세요.
REBOOTING_BROKER	Amazon MSK는 브로커를 재부팅하고 있습니다. 여전히 프로비저닝된 클러스터를 사용하여 데이터를 프로듀싱하고 컨슘할 수 있습니다. 그러나 클러스터가 ACTIVE 상태로 돌아올 때까지는 클러스터에서 Amazon MSK API 또는 AWS CLI업데이트 작업을 수행할 수 없습니다.

MSK 프로비저닝된 클러스터 상태	의미 및 가능한 조치
업데이트 중	사용자가 시작한 Amazon MSK API 또는 AWS CLI작업이 프로비저닝된 클러스터를 업데이트하는 중입니다. 여전히 프로비저닝된 클러스터를 사용하여 데이터를 프로듀싱하고 컨슘할 수 있습니다. 그러나 클러스터가 ACTIVE 상태로 돌아갈 때까지 클러스터에서 추가 Amazon MSK API 또는 AWS CLI업데이트 작업을 수행할 수 없습니다.

CloudWatch를 사용한 Express 브로커 모니터링에 대한 Amazon MSK 지표

Amazon MSK는 CloudWatch와 통합되므로 MSK Express 브로커에 대한 CloudWatch 지표를 수집, 확인, 분석할 수 있습니다. MSK 프로비저닝된 클러스터에 대해 구성한 지표는 1분 간격으로 자동으로 수집되어 CloudWatch로 푸시됩니다. MSK 프로비저닝된 클러스터의 모니터링 수준을 `DEFAULT`, `PER_BROKER`, `PER_TOPIC_PER_BROKER` 또는 `PER_TOPIC_PER_PARTITION` 중 하나로 설정할 수 있습니다. 다음 섹션의 표에는 각 모니터링 수준부터 사용할 수 있는 지표가 나와 있습니다.

`DEFAULT` 수준 지표는 무료입니다. 다른 지표에 대한 요금은 [Amazon CloudWatch 요금]() 페이지에 설명되어 있습니다.

Express 브로커에 대한 **`DEFAULT`** 레벨 모니터링

다음 표에 설명된 지표는 `DEFAULT` 모니터링 수준에서 무료로 사용할 수 있습니다.

이름	표시되는 경우	측정 기준	설명
ActiveControllerCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름	클러스터당 오직 하나의 컨트롤러만 지정된 시간에 활성화되어야 합니다.
BytesInPerSec	주제를 생성한 후.	클러스터 이름, 브로커 ID, 주제	클라이언트로부터 받은 초당 바이트 수입니다. 이 지표는 브로커별 및 주제별로 제공됩니다.

이름	표시되는 경우	측정 기준	설명
BytesOutPerSec	주제를 생성한 후.	클러스터 이름, 브로커 ID, 주제	클라이언트에 전송된 초당 바이트 수입니다. 이 지표는 브로커별 및 주제별로 제공됩니다.
ClientConnectionCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID, 클라이언트 인증	인증된 활성 클라이언트 연결 수입니다.
ConnectionCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	활성 인증, 미인증 및 브로커 간 연결 수입니다.
CpuIdle	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	CPU 유휴 시간의 백분율입니다.
CpuSystem	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	커널 공간에 있는 CPU의 백분율입니다.
CpuUser	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	사용자 공간에 있는 CPU의 백분율입니다.
GlobalPartitionCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름	클러스터의 모든 주제에서 복제본을 제외한 파티션 수입니다. `GlobalPartitionCount` 에는 복제본이 포함되지 않으므로 주제의 복제 인수가 1보다 큰 경우 `PartitionCount` 값의 합계가 `GlobalPartitionCount` 보다 클 수 있습니다.

이름	표시되는 경우	측정 기준	설명
GlobalTopicCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름	클러스터의 모든 브로커에 있는 총 주제 수입니다.
EstimatedMaxTimeLag	컨슈머 그룹이 주제에서 컨슘한 후.	소비자 그룹, 주제	`MaxOffsetLag` 를 배출하는 데 걸리는 예상 시간(초)입니다.
LeaderCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커당 총 파티션 리더 수(복제본 제외)입니다.
MaxOffsetLag	컨슈머 그룹이 주제에서 컨슘한 후.	소비자 그룹, 주제	주제의 모든 파티션에 대한 최대 오프셋 지연.
MemoryBuffered	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대한 버퍼링된 메모리의 크기(바이트)입니다.
MemoryCached	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대한 캐시 메모리의 크기(바이트)입니다.
MemoryFree	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 사용할 수 있는 메모리의 크기(바이트)입니다.
MemoryUsed	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에서 사용 중인 메모리의 크기(바이트)입니다.
MessagesInPerSec	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커의 초당 수신 메시지 수입니다.
NetworkRxDropped	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	삭제된 수신 패키지의 수입니다.

이름	표시되는 경우	측정 기준	설명
NetworkRxErrors	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대한 네트워크 수신 오류 수입니다.
NetworkRxPackets	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에서 수신된 패킷 수입니다.
NetworkTxDropped	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	삭제된 전송 패키지의 수입니다.
NetworkTxErrors	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커의 네트워크 전송 오류 수입니다.
NetworkTxPackets	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커가 전송한 패킷 수입니다.
PartitionCount	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	복제본을 포함하여 브로커당 주제 파티션의 총 수입니다.
ProduceTotalTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	밀리초 단위의 평균 프로덕션 시간.
RequestBytesMean	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 대한 요청 바이트의 평균 수입니다.
RequestTime	요청 스로틀링이 적용된 후.	클러스터 이름, 브로커 ID	브로커 네트워크 및 I/O 스레드가 요청을 처리하는 데 걸린 평균 시간(밀리초)입니다.
StorageUsed	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름	클러스터의 모든 파티션에서 복제본을 제외하고 사용한 총 스토리지 수입니다.

이름	표시되는 경우	측정 기준	설명
SumOffsetLag	컨슈머 그룹이 주제에서 컨슘한 후.	소비자 그룹, 주제	주제의 모든 파티션에 대한 집계된 오프셋 지연
UserPartitionExists	클러스터가 ACTIVE 상태에 도달한 후.	클러스터 이름, 브로커 ID	브로커에 사용자 소유 파티션이 있음을 나타내는 부울 지표입니다. 값이 1이면 브로커에 파티션이 있음을 나타냅니다.

Express 브로커에 대한 **PER_BROKER** 레벨 모니터링

모니터링 수준을 PER_BROKER로 설정하면 모든 DEFAULT 수준 지표에 추가로 다음 표에 설명된 지표가 표시됩니다. 다음 표의 지표에 대해서는 해당 비용을 지불하지만 DEFAULT 수준 지표는 계속 무료로 제공됩니다. 이 표에 있는 지표의 측정 기준은 클러스터 이름, 브로커 ID입니다.

이름	표시되는 경우	설명
ConnectionCloseRate	클러스터가 ACTIVE 상태에 도달한 후.	리스너당 초당 닫힌 연결 수입니다. 이 숫자는 리스너별로 집계되며 클라이언트 리스너에 대해 필터링됩니다.
ConnectionCreationRate	클러스터가 ACTIVE 상태에 도달한 후.	리스너당 초당 설정된 새 연결 수입니다. 이 숫자는 리스너별로 집계되며 클라이언트 리스너에 대해 필터링됩니다.
FetchConsumerLocalTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	컨슈머 요청이 리더에서 처리되는 데 걸리는 평균 시간(밀리초)입니다.

이름	표시되는 경우	설명
FetchConsumerRequestQueueTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	컨슈머 요청이 요청 대기열에서 대기하는 평균 시간(밀리초)입니다.
FetchConsumerResponseQueueTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	컨슈머 요청이 응답 대기열에서 대기하는 평균 시간(밀리초)입니다.
FetchConsumerResponseSendTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	컨슈머가 응답을 보내는 평균 시간(밀리초)입니다.
FetchConsumerTotalTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	컨슈머가 브로커에서 데이터를 가져오는 데 걸리는 평균 총 시간(밀리초)입니다.
FetchFollowerLocalTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어 요청이 리더에서 처리되는 데 걸리는 평균 시간(밀리초)입니다.
FetchFollowerRequestQueueTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어 요청이 요청 대기열에서 대기하는 평균 시간(밀리초)입니다.
FetchFollowerResponseQueueTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어 요청이 응답 대기열에서 대기하는 평균 시간(밀리초)입니다.
FetchFollowerResponseSendTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어가 응답을 보내는 평균 시간(밀리초)입니다.
FetchFollowerTotalTimeMsMean	프로듀서/컨슈머가 만들어진 이후.	팔로어가 브로커에서 데이터를 가져오는 데 걸리는 평균 총 시간(밀리초)입니다.
FetchThrottleByteRate	대역폭 스로틀링이 적용된 후.	초당 스로틀링된 바이트 수입니다.

이름	표시되는 경우	설명
FetchThrottleQueueSize	대역폭 스로틀링이 적용된 후.	스로틀 대기열에 있는 메시지 수입니다.
FetchThrottleTime	대역폭 스로틀링이 적용된 후.	평균 가져오기 스로틀링 시간 (밀리초)입니다.
IAMNumberOfConnectionRequests	클러스터가 ACTIVE 상태에 도달한 후.	초당 IAM 인증 요청의 수입니다.
IAMTooManyConnections	클러스터가 ACTIVE 상태에 도달한 후.	100을 초과하여 시도된 연결 수입니다. `0`은 연결 수가 한도 내에 있음을 의미합니다. `>0`인 경우 스로틀 한도가 초과되므로 연결 수를 줄여야 합니다.
NetworkProcessorAvgIdlePercent	클러스터가 ACTIVE 상태에 도달한 후.	네트워크 프로세서가 유휴 상태인 시간의 평균 백분율입니다.
ProduceLocalTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	리더에서 요청이 처리되는 데 걸리는 평균 시간(밀리초)입니다.
ProduceRequestQueueTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	요청 메시지가 대기열에서 소비하는 평균 시간(밀리초)입니다.
ProduceResponseQueueTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	응답 메시지가 대기열에서 소비하는 평균 시간(밀리초)입니다.
ProduceResponseSendTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	응답 메시지를 보내는 데 걸린 평균 시간(밀리초)입니다.
ProduceThrottleByteRate	대역폭 스로틀링이 적용된 후.	초당 스로틀링된 바이트 수입니다.

이름	표시되는 경우	설명
ProduceThrottleQueueSize	대역폭 스로틀링이 적용된 후.	스로틀 대기열에 있는 메시지 수입니다.
ProduceThrottleTime	대역폭 스로틀링이 적용된 후.	평균 프로덕션 스로틀링 시간(밀리초)입니다.
ProduceTotalTimeMsMean	클러스터가 ACTIVE 상태에 도달한 후.	밀리초 단위의 평균 프로덕션 시간.
ReplicationBytesInPerSec	주제를 생성한 후.	다른 브로커로부터 수신하는 초당 바이트 수입니다.
ReplicationBytesOutPerSec	주제를 생성한 후.	다른 브로커로 전송되는 초당 바이트 수입니다.
RequestExemptFromThrottleTime	요청 스로틀링이 적용된 후.	브로커 네트워크 및 I/O 스레드가 스로틀링에서 제외된 요청을 처리하는 데 걸린 평균 시간(밀리초)입니다.
RequestHandlerAvgIdlePercent	클러스터가 ACTIVE 상태에 도달한 후.	요청 핸들러 스레드가 유휴 상태인 시간의 평균 백분율입니다.
RequestThrottleQueueSize	요청 스로틀링이 적용된 후.	스로틀 대기열에 있는 메시지 수입니다.
RequestThrottleTime	요청 스로틀링이 적용된 후.	평균 요청 스로틀링 시간(밀리초)입니다.
TcpConnections	클러스터가 ACTIVE 상태에 도달한 후.	SYN 플래그가 설정된 수신 및 발신 TCP 세그먼트 수를 표시합니다.

이름	표시되는 경우	설명
TrafficBytes	클러스터가 ACTIVE 상태에 도달한 후.	클라이언트(생산자 및 소비자)와 브로커 간의 네트워크 트래픽을 전체 바이트 단위로 표시합니다. 브로커 사이의 트래픽은 보고되지 않습니다.

Express 브로커에 대한 **PER_TOPIC_PER_PARTITION** 레벨 모니터링

모니터링 수준을 PER_TOPIC_PER_PARTITION으로 설정하면 모든 PER_TOPIC_PER_BROKER, PER_BROKER, DEFAULT 수준 지표에 추가로 다음 표에 설명된 지표가 표시됩니다. DEFAULT 수준 지표만 무료입니다. 이 표에 있는 지표의 측정 기준은 컨슈머 그룹, 주제, 파티션입니다.

이름	표시되는 경우	설명
EstimatedTimeLag	컨슈머 그룹이 주제에서 컨슘한 후.	파티션 오프셋 지연을 배출하는 데 걸리는 예상 시간(초)입니다.
OffsetLag	컨슈머 그룹이 주제에서 컨슘한 후.	파티션 수준 컨슈머 지연의 오프셋 수입니다.

Express 브로커에 대한 **PER_TOPIC_PER_BROKER** 레벨 모니터링

모니터링 수준을 PER_TOPIC_PER_BROKER로 설정하면 모든 PER_BROKER 및 DEFAULT 기본 수준 지표에 추가로 다음 표에 설명된 지표가 표시됩니다. DEFAULT 수준 지표만 무료입니다. 이 표에 있는 지표의 측정 기준은 클러스터 이름, 브로커 ID, 주제입니다.

> **Important**
> 다음 표의 지표는 해당 값이 처음으로 0이 아닌 상태가 된 후에만 나타납니다. 예를 들어, BytesInPerSec를 보려면 하나 이상의 프로듀서가 먼저 클러스터로 데이터를 전송해야 합니다.

이름	표시되는 경우	설명
MessagesInPerSec	주제를 생성한 후.	초당 수신된 메시지 수입니다.

Prometheus를 사용하여 MSK 프로비저닝 클러스터 모니터링

시계열 지표 데이터를 위한 오픈 소스 모니터링 시스템인 Prometheus로 MSK 프로비저닝된 클러스터를 모니터링할 수 있습니다. Prometheus의 원격 쓰기 기능을 사용하여 이 데이터를 Prometheus용 Amazon 관리형 서비스에 게시할 수 있습니다. 또한 Prometheus 형식의 지표와 호환되는 도구 또는 Amazon MSK 오픈 모니터링과 통합되는 도구(Datadog, Lenses, New Relic, Sumo logic)를 사용할 수 있습니다. 오픈 모니터링은 무료로 제공되지만 가용 영역 간 데이터 전송에 대해서는 요금이 부과됩니다.

Prometheus에 대한 자세한 내용은 Prometheus 설명서를 참조하십시오.

Prometheus 사용에 대한 자세한 내용은 Prometheus용 Amazon Managed Service 및 Amazon Managed Grafana를 사용하여 Amazon MSK에 대한 운영 인사이트 향상을 참조하세요.

> **Note**
>
> KRaft 메타데이터 모드와 MSK Express 브로커는 공개 모니터링과 퍼블릭 액세스 둘 다를 활성화할 수 없습니다.

새로운 MSK 프로비저닝된 클러스터에서 오픈 모니터링 활성화

이 절차에서는AWS Management 콘솔, AWS CLI또는 Amazon MSK API를 사용하여 새 MSK 클러스터에서 공개 모니터링을 활성화하는 방법을 설명합니다.

사용AWS Management 콘솔

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 모니터링 섹션에서 Enable open monitoring with Prometheus(Prometheus를 사용하여 오픈 모니터링 활성화) 옆에 있는 확인란을 선택합니다.
3. 페이지의 모든 섹션에 필요한 정보를 제공하고 사용 가능한 모든 옵션을 검토합니다.
4. 클러스터 생성을 선택합니다.

사용AWS CLI

- [create-cluster](#) 명령을 호출하고 해당 `open-monitoring` 옵션을 지정합니다. `JmxExporter`, `NodeExporter` 또는 둘 다를 활성화합니다. `open-monitoring`을 지정하면 두 Exporter를 동시에 비활성화할 수 없습니다.

API 사용

- [CreateCluster](#) 작업을 호출하고 `OpenMonitoring`을 지정합니다. `jmxExporter`, `nodeExporter` 또는 둘 다를 활성화합니다. `OpenMonitoring`을 지정하면 두 Exporter를 동시에 비활성화할 수 없습니다.

기존 MSK 프로비저닝된 클러스터에서 오픈 모니터링 활성화

개방형 모니터링을 사용하려면 MSK 프로비저닝된 클러스터가 `ACTIVE` 상태인지 확인합니다.

사용AWS Management 콘솔

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 업데이트할 클러스터 이름을 선택합니다. 그러면 클러스터에 대한 세부 정보가 포함된 페이지로 이동합니다.
3. 속성 탭에서 아래로 스크롤하여 모니터링 섹션을 찾습니다.
4. 편집을 선택합니다.
5. Enable open monitoring with Prometheus(Prometheus를 사용하여 오픈 모니터링 활성화) 옆에 있는 확인란을 선택합니다.
6. 변경 사항 저장을 선택합니다.

사용AWS CLI

- [update-monitoring](#) 명령을 호출하고 해당 `open-monitoring` 옵션을 지정합니다. `JmxExporter`, `NodeExporter` 또는 둘 다를 활성화합니다. `open-monitoring`을 지정하면 두 Exporter를 동시에 비활성화할 수 없습니다.

API 사용

- UpdateMonitoring 작업을 호출하고 `OpenMonitoring`을 지정합니다. `jmxExporter`, `nodeExporter` 또는 둘 다를 활성화합니다. `OpenMonitoring`을 지정하면 두 Exporter를 동시에 비활성화할 수 없습니다.

Amazon EC2 인스턴스에서 Prometheus 호스트 설정

이 절차에서는 prometheus.yml 파일을 사용하여 Prometheus 호스트를 설정하는 방법을 설명합니다.

1. https://prometheus.io/download/#prometheus에서 Amazon EC2 인스턴스로 Prometheus 서버를 다운로드합니다.
2. 디렉터리에 다운로드한 파일의 압축을 풀고 해당 디렉터리로 이동합니다.
3. 다음 콘텐츠가 포함된 파일을 생성하고 이름을 `prometheus.yml`로 지정합니다.

```
# file: prometheus.yml
# my global config
global:
  scrape_interval:     60s

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped
 from this config.
  - job_name: 'prometheus'
    static_configs:
    # 9090 is the prometheus server port
    - targets: ['localhost:9090']
  - job_name: 'broker'
    file_sd_configs:
    - files:
      - 'targets.json'
```

4. ListNodes 작업을 사용하여 클러스터의 브로커 목록을 가져옵니다.
5. 다음 JSON을 통해 `targets.json` 파일을 생성합니다. *`broker_dns_1`*, *`broker_dns_2`* 및 나머지 브로커 DNS 이름을 이전 단계에서 브로커에 대해 얻은 DNS 이름으로 바꿉니다. 이전 단계에서 획득한 모든 브로커를 포함합니다. Amazon MSK는 JMX Exporter에 포트 11001을, Node Exporter에 포트 11002를 사용합니다.

ZooKeeper mode targets.json

```
[
  {
    "labels": {
      "job": "jmx"
    },
    "targets": [
      "broker_dns_1:11001",
      "broker_dns_2:11001",
      .
      .
      .
      "broker_dns_N:11001"
    ]
  },
  {
    "labels": {
      "job": "node"
    },
    "targets": [
      "broker_dns_1:11002",
      "broker_dns_2:11002",
      .
      .
      .
      "broker_dns_N:11002"
    ]
  }
]
```

KRaft mode targets.json

```
[
  {
    "labels": {
      "job": "jmx"
    },
    "targets": [
      "broker_dns_1:11001",
      "broker_dns_2:11001",
      .
```

```
      .
      .
      "broker_dns_N:11001",
      "controller_dns_1:11001",
      "controller_dns_2:11001",
      "controller_dns_3:11001"
    ]
  },
  {
    "labels": {
      "job": "node"
    },
    "targets": [
      "broker_dns_1:11002",
      "broker_dns_2:11002",
      .
      .
      .
      "broker_dns_N:11002"
    ]
  }
]
```

> **Note**
>
> KRaft 컨트롤러에서 JMX 지표를 스크레이핑하려면 JSON 파일에 컨트롤러 DNS 이름을 대상으로 추가합니다. 예를 들어, `controller_dns_1:11001`인 경우 `controller_dns_1`를 실제 컨트롤러 DNS 이름으로 바꿉니다.

6. Amazon EC2 인스턴스에서 Prometheus 서버를 시작하려면 Prometheus 파일을 추출하고 `prometheus.yml` 및 `targets.json`을 저장한 디렉터리에서 다음 명령을 실행합니다.

```
./prometheus
```

7. 이전 단계에서 Prometheus를 실행한 Amazon EC2 인스턴스의 IPv4 퍼블릭 IP 주소를 찾습니다. 다음 단계에서 이 퍼블릭 IP 주소가 필요합니다.

8. Prometheus 웹 UI에 액세스하려면 Amazon EC2 인스턴스에 액세스할 수 있는 브라우저를 열고 *`Prometheus-Instance-Public-IP`*`:9090`으로 이동합니다. 여기서 *`Prometheus-Instance-Public-IP`*는 이전 단계에서 얻은 퍼블릭 IP 주소입니다.

Prometheus 지표 사용

JMX로 Apache Kafka에 의해 방출된 모든 지표는 Prometheus와 오픈 모니터링을 사용하여 액세스할 수 있습니다. Apache Kafka 지표에 대한 자세한 내용은 Apache Kafka 설명서의 Monitoring을 참조하십시오. Apache Kafka 지표와 함께 소비자 지연 지표도 JMX MBean 이름 `kafka.consumer.group:type=ConsumerLagMetrics` 아래 포트 11001에서 사용할 수 있습니다. Prometheus 노드 내보내기를 사용하여 포트 11002에서 브로커에 대한 CPU 및 디스크 지표를 가져올 수도 있습니다.

Amazon Managed Service for Prometheus에 Prometheus 지표 저장

Amazon Managed Service for Prometheus는 Amazon MSK 클러스터 모니터링에 사용할 수 있는 Prometheus 호환 모니터링 및 알림 서비스입니다. 지표의 수집, 스토리지, 쿼리 및 알림을 자동으로 확장하는 완전관리형 서비스입니다. 또한 AWS보안 서비스와 통합되어 데이터에 빠르고 안전하게 액세스할 수 있습니다. 오픈 소스 PromQL 쿼리 언어를 사용하여 지표를 쿼리하고 지표에 대해 알릴 수 있습니다.

자세한 내용은 Amazon Managed Service for Prometheus 시작하기를 참조하세요.

소비자 지연 모니터링

소비자 지연을 모니터링하면 주제에서 사용 가능한 최신 데이터를 따라잡지 못하는 느리거나 멈춰 있는 소비자를 식별할 수 있습니다. 그런 다음 필요한 경우 해당 소비자의 규모를 조정하거나 재부팅하는 등의 수정 조치를 취할 수 있습니다. 소비자 지연을 모니터링하려면 Amazon CloudWatch를 사용하거나 Prometheus로 모니터링을 열 수 있습니다.

소비자 지연 지표는 주제에 기록된 최신 데이터와 애플리케이션에서 읽은 데이터 간의 차이를 정량화합니다. Amazon MSK는 Amazon CloudWatch를 통해 또는 Prometheus를 사용한 개방형 모니터링을 통해 얻을 수 있는 소비자 지연 지표(`EstimatedMaxTimeLag`, `EstimatedTimeLag`, `MaxOffsetLag`, `OffsetLag`, `SumOffsetLag`)를 제공합니다. 지표에 대한 자세한 내용은 the section called “Standard 브로커에 대한 CloudWatch 지표” 섹션을 참조하세요.

Amazon MSK는 Apache Kafka 2.2.1 이상 버전이 설치된 클러스터에 대한 소비자 지연 지표를 지원합니다. Kafka 및 CloudWatch 지표를 사용할 때는 다음 사항을 고려하세요.

- 컨슈머 지연 지표는 STABLE 또는 EMPTY 상태의 컨슈머 그룹에만 내보내집니다. 소비자 그룹은 재조정이 성공적으로 완료된 후 STABLE이므로 소비자 간에 파티션이 고르게 분산됩니다.
- 다음 시나리오에서는 컨슈머 지연 지표가 없습니다.
  - 컨슈머 그룹이 불안정한 경우.

  - 컨슈머 그룹의 이름에 콜론(:)이 포함되어 있습니다.
  - 컨슈머 그룹에 대한 컨슈머 오프셋을 설정하지 않았습니다.
- 컨슈머 그룹 이름이 CloudWatch에서 컨슈머 지연 지표의 측정 기준으로 사용됩니다. Kafka는 컨슈머 그룹 이름에서 UTF-8 문자를 지원하지만 CloudWatch는 측정 기준 값에 대해 ASCII 문자만 지원합니다. 컨슈머 그룹 이름에 ASCII가 아닌 문자를 사용하는 경우 CloudWatch는 컨슈머 지연 지표를 삭제합니다. CloudWatch에서 컨슈머 지연 지표를 올바르게 캡처하려면 컨슈머 그룹 이름에 ASCII 문자만 사용해야 합니다.

Amazon MSK 스토리지 용량 알림 사용

Amazon MSK 프로비저닝 클러스터에서 클러스터의 기본 스토리지 용량을 선택합니다. 프로비저닝된 클러스터에서 브로커의 스토리지 용량을 소진하면 브로커의 데이터 생성 및 소비 능력에 영향을 주어 비용이 많이 드는 가동 중지를 초래할 수 있습니다. Amazon MSK는 클러스터의 스토리지 용량을 모니터링하는 데 도움이 되는 CloudWatch 지표를 제공합니다. 하지만 스토리지 용량 문제를 더 쉽게 감지하고 해결할 수 있도록 Amazon MSK는 동적 클러스터 스토리지 용량 알림을 자동으로 전송합니다. 스토리지 용량 알림에는 클러스터의 스토리지 용량을 관리하기 위한 단기 및 장기적 단계의 권장 사항이 포함됩니다. Amazon MSK 콘솔에서 알림 내의 빠른 링크를 사용하여 즉시 권장 조치를 수행할 수 있습니다.

MSK 스토리지 용량 알림에는 사전 예방과 개선이라는 두 가지 유형이 있습니다.

- 사전 예방('조치 필요') 스토리지 용량 알림은 클러스터의 잠재적 스토리지 문제에 대해 경고합니다. MSK 클러스터의 브로커가 디스크 스토리지 용량의 60% 또는 80% 이상을 사용한 경우 영향을 받는 브로커에 대한 사전 알림을 받게 됩니다.
- 개선('중요 조치 필요') 스토리지 용량 알림은 사용자에게 MSK 클러스터의 브로커 중 하나에 디스크 스토리지 용량이 부족할 때 수정 조치를 취하여 중요한 클러스터 문제를 해결하도록 합니다.

Amazon MSK는 이러한 알림을 Amazon MSK 콘솔, AWS Health Dashboard, Amazon EventBridge 및 AWS계정의 이메일 연락처로 자동으로 전송합니다. 또한 Amazon EventBridge가 이러한 알림을 Slack이나 New Relic, Datadog과 같은 도구에 전달하도록 구성할 수 있습니다.

스토리지 용량 경고는 모든 MSK 프로비저닝된 클러스터에서 기본적으로 활성화되며 끌 수 없습니다. 이 기능은 MSK를 사용할 수 있는 모든 리전에서 사용할 수 있습니다.

스토리지 용량 알림 모니터링

여러 가지 방법으로 스토리지 용량 알림을 확인할 수 있습니다.

- Amazon MSK 콘솔로 이동합니다. 스토리지 용량 알림은 클러스터 알림 창에 90일 동안 표시됩니다. 알림에는 디스크 스토리지 용량 문제를 해결하기 위한 권장 사항 및 원클릭 링크 동작이 포함됩니다.
- ListClusters, ListClustersV2, DescribeCluster 또는 DescribeClusterV2 API를 사용하여 `CustomerActionStatus` 및 모든 클러스터 알림을 볼 수 있습니다.
- AWS Health Dashboard로 이동하여 MSK 및 기타 AWS 서비스의 알림을 확인합니다.
- AWS Health API와 Amazon EventBridge를 설정하여 알림을 Datadog, NewRelic, Slack과 같은 타사 플랫폼으로 라우팅할 수 있습니다.

Amazon MSK 클러스터의 보안 설정 업데이트

UpdateSecurity Amazon MSK 작업을 사용하여 MSK 클러스터의 인증 및 클라이언트-브로커 암호화 설정을 업데이트합니다. 또한 상호 TLS 인증을 위해 인증서에 서명하는 데 사용되는 사설 보안 기관을 업데이트할 수도 있습니다. 클러스터 내(브로커 간) 암호화 설정은 변경할 수 없습니다.

클러스터는 보안 설정을 업데이트할 수 있는 `ACTIVE` 상태여야 합니다.

IAM, SASL 또는 TLS를 사용하여 인증을 활성화하는 경우 클라이언트와 브로커 간의 암호화도 활성화해야 합니다. 다음 표에는 가능한 조합이 나와 있습니다.

Authentication	클라이언트-브로커 암호화 옵션	브로커-브로커 암호화
Unauthenticated	TLS, PLAINTEXT, TLS_PLAINTEXT	활성화되거나 비활성화된 상태일 수 있습니다.
mTLS	TLS, TLS_PLAINTEXT	활성화된 상태여야 합니다.
SASL/SCRAM	TLS	활성화된 상태여야 합니다.
SAL/IAM	TLS	활성화된 상태여야 합니다.

클라이언트-브로커 암호화가 `TLS_PLAINTEXT`로 설정되고 클라이언트 인증이 `mTLS`로 설정된 경우 Amazon MSK는 클라이언트가 연결할 수 있는 두 가지 유형의 리스너를 생성합니다. 하나는 클라이언트가 TLS 암호화가 포함된 mTLS 인증을 사용하여 연결할 수 있는 리스너이고 다른 하나는 클라이언트가 인증이나 암호화 없이 연결할 수 있는 리스너(일반 텍스트)입니다.

보안 설정에 대한 자세한 내용은 the section called “보안” 섹션을 참조하세요.

를 사용하여 Amazon MSK 클러스터 보안 설정 업데이트AWS Management 콘솔

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 업데이트하려는 MSK 클러스터를 선택합니다.
3. 보안 설정 섹션에서 편집을 선택합니다.
4. 클러스터에 사용할 인증 및 암호화 설정을 선택한 다음 변경 사항 저장을 선택합니다.

를 사용하여 Amazon MSK 클러스터 보안 설정 업데이트AWS CLI

1. 클러스터에 적용하려는 암호화 설정이 포함된 JSON 파일을 생성합니다. 다음은 예입니다.

   > **Note**
   > 클라이언트-브로커 암호화 설정만 업데이트할 수 있습니다. 클러스터 내(브로커 간) 암호화 설정은 업데이트할 수 없습니다.

   ```
   {"EncryptionInTransit":{"ClientBroker": "TLS"}}
   ```

2. 클러스터에 적용할 인증 설정이 포함된 JSON 파일을 생성합니다. 다음은 예입니다.

   ```
   {"Sasl":{"Scram":{"Enabled":true}}}
   ```

3. 다음 AWS CLI명령을 실행합니다.

   ```
   aws kafka update-security --cluster-arn ClusterArn --current-version Current-Cluster-Version --client-authentication file://Path-to-Authentication-Settings-JSON-File --encryption-info file://Path-to-Encryption-Settings-JSON-File
   ```

   이 `update-security` 작업의 출력은 다음 JSON과 같습니다.

   ```
   {
       "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
   ```

```
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

4. `update-security` 작업 상태를 보려면 다음 명령을 실행하여 *`ClusterOperationArn`*을 `update-security` 명령 출력에서 가져온 ARN으로 변경합니다.

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

이 `describe-cluster-operation` 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "c0b7af47-8591-45b5-9c0c-909a1a2c99ea",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2021-09-17T02:35:47.753000+00:00",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "PENDING",
        "OperationType": "UPDATE_SECURITY",
        "SourceClusterInfo": {},
        "TargetClusterInfo": {}
    }
}
```

`OperationState` 값이 `PENDING` 또는 `UPDATE_IN_PROGRESS`인 경우 잠시 기다린 다음 `describe-cluster-operation` 명령을 다시 실행합니다.

> **Note**
> 클러스터의 보안 설정을 업데이트하기 위한 AWS CLI 및 API 작업은 멱등성입니다. 즉, 보안 업데이트 작업을 호출하고 클러스터가 현재 가지고 있는 설정과 동일한 인증 또는 암호화 설정을 지정해도 해당 설정은 변경되지 않습니다.

API를 사용하여 클러스터의 보안 설정 업데이트

API를 사용하여 Amazon MSK 클러스터의 보안 설정을 업데이트하려면 UpdateSecurity를 참조하세요.

> **Note**
>
> MSK 클러스터의 보안 설정을 업데이트하기 위한 AWS CLI및 API 작업은 멱등성입니다. 즉, 보안 업데이트 작업을 호출하고 클러스터가 현재 가지고 있는 설정과 동일한 인증 또는 암호화 설정을 지정해도 해당 설정은 변경되지 않습니다.

Amazon MSK 클러스터의 브로커 수 확장

MSK 클러스터의 브로커 수를 늘리려는 경우 이 Amazon MSK 작업을 사용합니다. 클러스터를 확장하려면 해당 클러스터가 `ACTIVE` 상태인지 확인합니다.

> **Important**
>
> MSK 클러스터를 확장하려면 이 Amazon MSK 작업을 사용합니다. 이 작업을 사용하지 않고 클러스터에 브로커를 추가하지 마십시오.

클러스터에 브로커를 추가한 후 파티션을 재분배하는 방법은 the section called “파티션 재할당” 단원을 참조하십시오.

를 사용하여 Amazon MSK 클러스터 확장AWS Management 콘솔

이 프로세스는 AWS Management 콘솔를 사용하여 Amazon MSK 클러스터의 브로커 수를 늘리는 방법을 설명합니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 브로커 수를 늘리려는 MSK 클러스터를 선택합니다.
3. 작업 드롭다운에서 브로커 수 편집을 선택합니다.
4. 가용 영역당 클러스터가 보유할 브로커 수를 입력한 다음 변경 사항 저장을 선택합니다.

를 사용하여 Amazon MSK 클러스터 확장AWS CLI

이 프로세스는 AWS CLI를 사용하여 Amazon MSK 클러스터의 브로커 수를 늘리는 방법을 설명합니다.

1. 다음 명령을 실행하여 *ClusterArn*을 클러스터 생성 후 받은 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.

   *Current-Cluster-Version*을 클러스터의 현재 버전으로 바꿉니다.

   > **Important**
   >
   > 클러스터 버전은 단순한 정수가 아닙니다. 클러스터의 현재 버전을 찾으려면 DescribeCluster 작업 또는 describe-clusterAWS CLI 명령을 사용합니다. 버전의 예를 들면 `KTVPDKIKX0DER`입니다.

   Target-Number-of-Brokers 파라미터는 이 작업을 성공적으로 완료하여 클러스터에 포함하고자 하는 총 브로커 노드 수를 나타냅니다. *Target-Number-of-Brokers*에 지정하는 값은 클러스터의 현재 브로커 수보다 큰 정수여야 합니다. 또한 가용 영역 수의 배수여야 합니다.

   ```
   aws kafka update-broker-count --cluster-arn ClusterArn --current-version Current-Cluster-Version --target-number-of-broker-nodes Target-Number-of-Brokers
   ```

   이 `update-broker-count` 작업의 출력은 다음 JSON과 같습니다.

   ```
   {

       "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
       "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-abcd-4f7f-1234-9876543210ef"
   }
   ```

2. `update-broker-count` 작업 결과를 가져오려면 다음 명령을 실행하여 *ClusterOperationArn*을 `update-broker-count` 명령의 출력에서 가져온 ARN으로 바꿉니다.

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

이 describe-cluster-operation 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "c0b7af47-8591-45b5-9c0c-909a1a2c99ea",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-09-25T23:48:04.794Z",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_COMPLETE",
        "OperationType": "INCREASE_BROKER_COUNT",
        "SourceClusterInfo": {
            "NumberOfBrokerNodes": 9
        },
        "TargetClusterInfo": {
            "NumberOfBrokerNodes": 12
        }
    }
}
```

이 출력에서 OperationType은 INCREASE_BROKER_COUNT입니다. OperationState 값이 UPDATE_IN_PROGRESS인 경우, 잠시 기다린 다음 describe-cluster-operation 명령을 다시 실행합니다.

API를 사용하여 Amazon MSK 클러스터 확장

API를 사용하여 클러스터의 브로커 수를 늘리려면 UpdateBrokerCount를 참조하십시오.

Amazon MSK 클러스터에서 브로커 제거

Amazon Managed Streaming for Apache Kafka(MSK) 프로비저닝 클러스터에서 브로커를 제거하려할 때 이 Amazon MSK 작업을 사용합니다. 가용성 영향, 데이터 내구성 위험 또는 데이터 스트리밍 애플리케이션의 중단 없이 브로커 세트를 제거하여 클러스터의 스토리지 및 컴퓨팅 용량을 줄일 수 있습니다.

클러스터에 브로커를 더 추가하여 트래픽 증가를 처리하고 트래픽이 줄어들 때 브로커를 제거할 수 있습니다. 브로커 추가 및 제거 기능을 사용하여 클러스터 용량을 최대한 활용하고 MSK 인프라 비용을 최적화할 수 있습니다. 브로커를 제거하면 워크로드 요구 사항에 맞게 기존 클러스터 용량을 브로커 수준에서 제어하고 다른 클러스터로 마이그레이션하지 않아도 됩니다.

AWS콘솔, 명령줄 인터페이스(CLI), SDK 또는 CloudFormation를 사용하여 프로비저닝된 클러스터의 브로커 수를 줄입니다. MSK는 파티션이 없는 브로커(카나리 주제 제외)를 선택하고 애플리케이션이 해당 브로커에 데이터를 생성하지 못하도록 방지하는 동시에 클러스터에서 해당 브로커를 안전하게 제거합니다.

클러스터의 스토리지 및 컴퓨팅을 줄이려면 가용 영역당 1개의 브로커를 제거해야 합니다. 예를 들어 단일 브로커 제거 작업에서 2개의 가용 영역 클러스터에서 2개의 브로커를 제거하거나 3개의 가용 영역 클러스터에서 3개의 브로커를 제거할 수 있습니다.

클러스터에서 브로커를 제거한 후 파티션을 재조정하는 방법은 [the section called “파티션 재할당”](#) 단원을 참조하세요.

인스턴스 크기에 관계없이 모든 M5 및 M7g 기반 MSK 프로비저닝 클러스터에서 브로커를 제거할 수 있습니다.

브로커 제거는 KRaft 모드 클러스터를 비롯하여 Kafka 버전 2.8.1 이상에서 가능합니다.

주제

- 모든 파티션을 제거하여 브로커 제거 준비
- AWSManagement Console을 사용하여 브로커 제거
- AWSCLI를 사용하여 브로커 제거
- AWSAPI를 사용하여 브로커 제거

모든 파티션을 제거하여 브로커 제거 준비

브로커 제거 프로세스를 시작하기 전에 먼저 제거하려는 브로커의 주제 `__amazon_msk_canary` 및 `__amazon_msk_canary_state`에 대한 파티션을 제외한 모든 파티션을 이동합니다. 이는 Amazon MSK가 클러스터 상태 및 진단 지표에 대해 생성하는 내부 주제입니다.

Kafka 관리자 API 또는 Cruise Control을 사용하여 클러스터에 유지하려는 다른 브로커로 파티션을 이동할 수 있습니다. 파티션 재할당을 참조하세요.

파티션을 제거하는 예제 프로세스

이 섹션은 제거하려는 브로커에서 파티션을 제거하는 방법의 예제입니다. 각 AZ에 브로커 2개씩, 브로커 6개가 있는 클러스터가 있고 다음 네 가지 주제가 있다고 가정해 보겠습니다.

- `__amazon_msk_canary`
- `__consumer_offsets`
- `__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-c657f7e4ff32-2`
- `msk-brk-rmv`

1. [클라이언트 머신 생성]에 설명된 대로 클라이언트 머신을 생성합니다.
2. 클라이언트 머신을 구성한 후 다음 명령을 실행하여 클러스터에서 사용 가능한 모든 주제를 나열합니다.

   ```
   ./bin/kafka-topics.sh --bootstrap-server "CLUSTER_BOOTSTRAP_STRING" --list
   ```

   이 예제에서는 `__amazon_msk_canary`, `__consumer_offsets`, `__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-c657f7e4ff32-2`, `msk-brk-rmv`의 4개의 주제 이름을 볼 수 있습니다.
3. 클라이언트 머신에서 `topics.json`이라는 json 파일을 생성하고 다음 코드 예제와 같이 모든 사용자 주제 이름을 추가합니다. 이 `__amazon_msk_canary` 주제는 필요할 때 자동으로 이동되는 서비스 관리형 주제이므로 주제 이름을 포함할 필요가 없습니다.

   ```
   {
   "topics": [
   {"topic": "msk-brk-rmv"},
   {"topic": "__consumer_offsets"},
   {"topic": "__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-
   c657f7e4ff32-2"}
   ],
   "version":1
   }
   ```
4. 다음 명령을 실행하여 클러스터의 브로커 6개 중 브로커 3개의 브로커로만 파티션을 이동하는 제안을 생성합니다.

```
./bin/kafka-reassign-partitions.sh --bootstrap-server "CLUSTER_BOOTSTRAP_STRING" --
topics-to-move-json-file topics.json --broker-list 1,2,3 --generate
```

5. `reassignment-file.json`이라는 파일을 생성하고 위 명령에서 가져온 `proposed partition reassignment configuration`을 복사합니다.
6. 다음 명령을 실행하여 `reassignment-file.json`에서 지정한 파티션을 이동합니다.

   ```
   ./bin/kafka-reassign-partitions.sh --bootstrap-server "CLUSTER_BOOTSTRAP_STRING" --
   reassignment-json-file reassignment-file.json --execute
   ```

   출력 결과는 다음과 비슷합니다:

   ```
   Successfully started partition reassignments for morpheus-test-topic-1-0,test-
   topic-1-0
   ```

7. 다음 명령을 실행하여 모든 파티션이 이동했는지 확인합니다.

   ```
   ./bin/kafka-reassign-partitions.sh --bootstrap-server "CLUSTER_BOOTSTRAP_STRING" --
   reassignment-json-file reassignment-file.json --verify
   ```

   출력 결과는 다음과 비슷합니다. 요청한 주제의 모든 파티션이 성공적으로 재할당될 때까지 상태를 모니터링합니다.

   ```
   Status of partition reassignment:
   Reassignment of partition msk-brk-rmv-0 is completed.
   Reassignment of partition msk-brk-rmv-1 is completed.
   Reassignment of partition __consumer_offsets-0 is completed.
   Reassignment of partition __consumer_offsets-1 is completed.
   ```

8. 상태가 각 파티션에 대한 파티션 재할당이 완료되었음을 나타내면 5분 동안 `UserPartitionExists` 지표를 모니터링하여 파티션을 이동한 브로커에 대해 `0`이 표시되는지 확인합니다. 이 값을 확인한 후 클러스터에서 브로커를 제거할 수 있습니다.

AWSManagement Console을 사용하여 브로커 제거

AWSManagement Console을 사용하여 브로커를 제거하려면

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.

2. 제거하려는 브로커가 포함된 MSK 클러스터를 선택합니다.
3. 클러스터 세부 정보 페이지에서 작업 버튼을 선택하고 브로커 수 편집 옵션을 선택합니다.
4. 가용 영역당 클러스터에 보유할 브로커 수를 입력합니다. 콘솔에는 제거할 가용 영역의 브로커 수가 요약되어 있습니다. 이 값이 원하는 수치인지 확인합니다.
5. 변경 사항 저장을 선택합니다.

실수로 브로커가 제거되지 않도록 콘솔에서 브로커를 삭제할지 확인하도록 요청하는 메시지가 표시됩니다.

AWSCLI를 사용하여 브로커 제거

다음 명령을 실행하여 `ClusterArn`을 클러스터 생성 후 받은 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 [Amazon ECS 클러스터 나열]을 참조하세요. `Current-Cluster-Version`을 클러스터의 버전으로 바꿉니다.

> **Important**
>
> 클러스터 버전은 단순한 정수가 아닙니다. 클러스터의 현재 버전을 찾으려면 [DescribeCluster] 작업 또는 [describe-cluster]AWS CLI 명령을 사용합니다. 버전의 예를 들면 `KTVPDKIKX0DER`입니다.

`Target-Number-of-Brokers` 파라미터는 이 작업을 성공적으로 완료하여 클러스터에 포함하고자 하는 총 브로커 노드 수를 나타냅니다. *`Target-Number-of-Brokers`*에 지정하는 값은 클러스터의 현재 브로커 수보다 작은 정수여야 합니다. 또한 가용 영역 수의 배수여야 합니다.

```
aws kafka update-broker-count --cluster-arn ClusterArn --current-version Current-Cluster-Version --target-number-of-broker-nodes Target-Number-of-Brokers
```

이 `update-broker-count` 작업의 출력은 다음 JSON과 같습니다.


```
{
"ClusterOperationInfo": {
"ClientRequestId": "c0b7af47-8591-45b5-9c0c-909a1a2c99ea",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-09-25T23:48:04.794Z",
```

```
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_COMPLETE",
        "OperationType": "DECREASE_BROKER_COUNT",
        "SourceClusterInfo": {
"NumberOfBrokerNodes": 12
        },
        "TargetClusterInfo": {
"NumberOfBrokerNodes": 9
        }
    }
}
```

이 출력에서 `OperationType`은 `DECREASE_BROKER_COUNT`입니다. `OperationState` 값이 `UPDATE_IN_PROGRESS`인 경우, 잠시 기다린 다음 `describe-cluster-operation` 명령을 다시 실행합니다.

AWSAPI를 사용하여 브로커 제거

API를 사용하여 클러스터에서 브로커를 제거하려면 Amazon Managed Streaming for Apache Kafka API 참조의 [UpdateBrokerCount](#)를 참조하세요.

Amazon MSK 클러스터 브로커 크기 업데이트

Apache Kafka 파티션을 재할당하지 않고도 브로커의 크기를 변경하여 온디맨드 방식으로 MSK 클러스터의 규모를 조정할 수 있습니다. 브로커 크기를 변경하면 클러스터 I/O를 중단하지 않고도 워크로드 변화에 따라 MSK 클러스터의 컴퓨팅 용량을 유연하게 조정할 수 있습니다. Amazon MSK는 지정된 클러스터의 모든 브로커에 대해 동일한 브로커 크기를 사용합니다.

Standard 브로커의 경우 클러스터 브로커 크기를 M5 또는 T3에서 M7g로, T3에서 M5로 또는 M7g에서 M5로 업데이트할 수 있습니다.

> **Note**
>
> 더 큰 브로커 크기에서 더 작은 브로커 크기로 마이그레이션할 수 없습니다. 예: M7g.large에서 T3.small.

Express 브로커의 경우 M7g 브로커 크기만 사용할 수 있습니다.

이 주제에서는 MSK 클러스터의 브로커 크기를 업데이트하는 방법을 설명합니다.

더 작은 브로커 크기로 마이그레이션하면 성능이 저하되고 브로커당 달성 가능한 최대 처리량이 감소할 수 있습니다. 더 큰 브로커 크기로 마이그레이션하면 성능이 향상될 수 있지만 비용이 더 커질 수 있습니다.

브로커 크기 업데이트는 클러스터가 실행되는 동안 롤링 방식으로 이루어집니다. 즉, Amazon MSK는 브로커 크기 업데이트를 수행하기 위해 한 번에 하나의 브로커를 중단합니다. 브로커 크기 업데이트 중에 클러스터를 고가용성으로 만드는 방법에 대한 자세한 내용은 [the section called “고가용성 클러스터 빌드”] 단원을 참조하세요. 생산성에 미칠 수 있는 잠재적 영향을 더욱 줄이려면 트래픽이 적은 기간에 브로커 크기 업데이트를 수행하면 됩니다.

브로커 크기 업데이트 중 데이터를 계속 생성하고 사용할 수 있습니다. 그러나 업데이트가 완료될 때까지 기다려야 브로커를 재부팅하거나 Amazon MSK 작업 아래에 나열된 업데이트 작업을 호출할 수 있습니다.

클러스터를 더 작은 브로커 크기로 업데이트하려는 경우 먼저 테스트 클러스터에서 업데이트를 시도하여 시나리오에 어떤 영향을 미치는지 확인하는 것이 좋습니다.

> **Important**
>
> 브로커당 파티션 수가 the section called “ 클러스터 크기를 적절하게 조정: Standard 브로커당 파티션 수”에 지정된 최대 수를 초과하는 경우에는 클러스터를 더 작은 브로커 크기로 업데이트할 수 없습니다.

주제

- 를 사용하여 Amazon MSK 클러스터 브로커 크기 업데이트AWS Management 콘솔
- 를 사용하여 Amazon MSK 클러스터 브로커 크기 업데이트AWS CLI
- API를 사용하여 브로커 크기 업데이트

를 사용하여 Amazon MSK 클러스터 브로커 크기 업데이트AWS Management 콘솔

이 프로세스는를 사용하여 Amazon MSK 클러스터 브로커 크기를 업데이트하는 방법을 보여줍니다.AWS Management 콘솔

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 브로커 크기를 업데이트할 MSK 클러스터를 선택합니다.

3. 클러스터의 세부 정보 페이지에서 브로커 요약 섹션을 찾아 브로커 크기 편집을 선택합니다.
4. 목록에서 원하는 브로커 크기를 선택합니다.
5. 변경 내용을 저장합니다.

를 사용하여 Amazon MSK 클러스터 브로커 크기 업데이트AWS CLI

다음 명령을 실행하여 *ClusterArn*을 클러스터 생성 후 받은 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.

1. *Current-Cluster-Version*을 클러스터의 현재 버전으로 변경하고 *TargetType*을 브로커가 될 새 크기로 변경합니다. 브로커 크기에 대해 자세히 알아보려면 the section called “브로커 유형” 단원을 참조하세요.

```
aws kafka update-broker-type --cluster-arn ClusterArn --current-version Current-
Cluster-Version --target-instance-type TargetType
```

다음은 이 명령을 사용하는 방법을 보여주는 예제입니다.

```
aws kafka update-broker-type --cluster-arn "arn:aws:kafka:us-
east-1:0123456789012:cluster/exampleName/abcd1234-0123-abcd-5678-1234abcd-1" --
current-version "K1X5R6FKA87" --target-instance-type kafka.m5.large
```

이 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:0123456789012:cluster/exampleName/
abcd1234-0123-abcd-5678-1234abcd-1",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

2. update-broker-type 작업 결과를 가져오려면 다음 명령을 실행하여 *ClusterOperationArn*을 update-broker-type 명령의 출력에서 가져온 ARN으로 바꿉니다.

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

이 `describe-cluster-operation` 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
  "ClusterOperationInfo": {
    "ClientRequestId": "982168a3-939f-11e9-8a62-538df00285db",
    "ClusterArn": "arn:aws:kafka:us-east-1:0123456789012:cluster/exampleName/
abcd1234-0123-abcd-5678-1234abcd-1",
    "CreationTime": "2021-01-09T02:24:22.198000+00:00",
    "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-operation/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
    "OperationState": "UPDATE_COMPLETE",
    "OperationType": "UPDATE_BROKER_TYPE",
    "SourceClusterInfo": {
      "InstanceType": "t3.small"
    },
    "TargetClusterInfo": {
      "InstanceType": "m5.large"
    }
  }
}
```

`OperationState` 값이 `UPDATE_IN_PROGRESS`인 경우, 잠시 기다린 다음 `describe-cluster-operation` 명령을 다시 실행합니다.

API를 사용하여 브로커 크기 업데이트

API를 사용하여 브로커 크기를 업데이트하려면 UpdateBrokerType을 참조하세요.

`UpdateBrokerType`을 사용하여 클러스터 브로커 크기를 M5 또는 T3에서 M7g로 업데이트하거나 M7g에서 M5로 업데이트할 수 있습니다.

Amazon MSK에서 Apache Kafka용 LinkedIn의 Cruise Control 사용

LinkedIn의 크루즈 컨트롤을 사용하여 Amazon MSK 클러스터의 균형을 재조정하고, 이상 징후를 감지 및 수정하고, 클러스터의 상태를 모니터링할 수 있습니다.

> **Note**
>
> 새로 생성된 Express 기반 클러스터에 대해 지능형 리밸런싱이 켜져 있는 경우 파티션 리밸런싱에 Cruise Control과 같은 타사 도구를 사용할 수 없습니다. 이러한 타사 도구에서 제공하는 파티션 재할당 API를 사용하려면 먼저 지능형 재분배를 일시 중지해야 합니다.

Cruise Control을 다운로드하고 빌드하려면 다음을 수행합니다.

1. Amazon MSK 클러스터와 동일한 Amazon VPC에 Amazon EC2 인스턴스를 생성합니다.
2. 이전 단계에서 생성한 Amazon EC2 인스턴스에 Prometheus를 설치합니다. 프라이빗 IP와 포트를 기록해 둡니다. 기본 포트 번호는 9090입니다. 클러스터에 대한 지표를 집계하도록 Prometheus를 구성하는 방법에 대한 자세한 내용은 the section called “Prometheus를 사용하여 모니터링” 섹션을 참조하세요.
3. Amazon EC2 인스턴스에서 Cruise Control을 다운로드합니다. (또는 원하는 경우 Cruise Control에 별도의 Amazon EC2 인스턴스를 사용할 수 있습니다.) Apache Kafka 버전 2.4.*가 설치된 클러스터의 경우 최신 2.4.* Cruise Control 릴리스를 사용합니다. 클러스터에 2.4.*보다 오래된 Apache Kafka 버전이 있는 경우, 최신 2.0.* Cruise Control 릴리스를 사용합니다.
4. Cruise Control 파일의 압축을 푼 다음 압축을 푼 폴더로 이동합니다.
5. 다음 명령을 실행하여 git을 설치합니다.

   ```
   sudo yum -y install git
   ```

6. 다음 명령을 실행하여 로컬 리포지토리를 초기화합니다. *Your-Cruise-Control-Folder*를 현재 폴더의 이름(Cruise Control 다운로드의 압축을 풀었을 때 가져온 폴더)으로 변경합니다.

   ```
   git init && git add . && git commit -m "Init local repo." && git tag -a Your-
   Cruise-Control-Folder -m "Init local version."
   ```

7. 다음 명령을 실행하여 소스 코드를 빌드합니다.

   ```
   ./gradlew jar copyDependantLibs
   ```

Cruise Control을 구성하고 실행하려면 다음을 수행합니다.

1. `config/cruisecontrol.properties` 파일을 다음과 같이 업데이트합니다. 예제 부트스트랩 서버 및 부트스트랩 브로커 문자열을 클러스터에 대한 값으로 바꿉니다. 클러스터에

대한 이러한 문자열을 얻으려면 콘솔에서 클러스터 세부 정보를 확인할 수 있습니다. 또는 GetBootstrapBrokers 및 DescribeCluster API 작업 또는 이에 상응하는 CLI를 사용할 수 있습니다.

```
# If using TLS encryption, use 9094; use 9092 if using plaintext
bootstrap.servers=b-1.test-cluster.2skv42.c1.kafka.us-
east-1.amazonaws.com:9094,b-2.test-cluster.2skv42.c1.kafka.us-
east-1.amazonaws.com:9094,b-3.test-cluster.2skv42.c1.kafka.us-
east-1.amazonaws.com:9094

# SSL properties, needed if cluster is using TLS encryption
security.protocol=SSL
ssl.truststore.location=/home/ec2-user/kafka.client.truststore.jks

# Use the Prometheus Metric Sampler
metric.sampler.class=com.linkedin.kafka.cruisecontrol.monitor.sampling.prometheus.Prometheu

# Prometheus Metric Sampler specific configuration
prometheus.server.endpoint=1.2.3.4:9090 # Replace with your Prometheus IP and port

# Change the capacity config file and specify its path; details below
capacity.config.file=config/capacityCores.json
```

Express 브로커의 경우 분석기 구성에 구성된 목표에 `DiskCapacityGoal`을 사용하지 않는 것이 좋습니다.

2. `config/capacityCores.json` 파일을 편집하여 올바른 디스크 크기와 CPU 코어 및 네트워크 입/출력 제한을 지정합니다. Express 브로커의 경우 `DISK` 용량 항목은 Cruise Control을 설정하는 데만 필요합니다. MSK는 Express 브로커의 모든 스토리지를 관리하므로이 값을 `Integer.MAX_VALUE (2147483647)`와 같이 매우 높은 수로 설정해야 합니다. Standard 브로커의 경우 DescribeCluster API 작업(또는 describe-cluster CLI)을 사용하여 디스크 크기를 얻을 수 있습니다. CPU 코어 및 네트워크 입/출력 제한에 대해서는 Amazon EC2 인스턴스 유형을 참조하세요.

   Standard broker config/capacityCores.json

   ```
   {
     "brokerCapacities": [
       {
         "brokerId": "-1",
         "capacity": {
           "DISK": "10000",
           "CPU": {
   ```

```
          "num.cores": "2"
        },
        "NW_IN": "5000000",
        "NW_OUT": "5000000"
      },
      "doc": "This is the default capacity. Capacity unit used for disk is in
 MB, cpu is in number of cores, network throughput is in KB."
    }
  ]
}
```

Express broker config/capacityCores.json

```
{
  "brokerCapacities":[
    {
      "brokerId": "-1",
      "capacity": {
        "DISK": "2147483647",
        "CPU": {"num.cores": "16"},
        "NW_IN": "1073741824",
        "NW_OUT": "1073741824"
      },
      "doc": "This is the default capacity. Capacity unit used for disk is in
 MB, cpu is in number of cores, network throughput is in KB."
    }
  ]
}
```

3. Cruise Control UI를 설치할 수 있습니다(선택 사항). 다운로드하려면 Cruise Control 프런트엔드 설정을 참조하세요.

4. 다음 명령을 실행하여 Cruise Control을 시작합니다. `screen` 또는 `tmux`와 같은 도구를 사용하거나 장기 실행 세션을 열어 두는 것을 고려하세요.

```
<path-to-your-CRUISE-CONTROL-installation>/bin/kafka-cruise-control-start.sh
 config/cruisecontrol.properties 9091
```

5. Cruise Control API 또는 UI를 사용하여 Cruise Control에 클러스터 로드 데이터가 있는지, 그리고 Cruise Control이 재조정을 제안하는지 확인합니다. 유효한 지표 창을 가져오는 데 몇 분 정도 소요될 수 있습니다.

> **Important**
>
> Express 브로커는 Zookeeper 엔드포인트를 노출하지 않으므로 Cruise Control 버전 2.5.60 이상만 Express 브로커와 호환됩니다.

Amazon MSK용 Cruise Control의 자동 배포 템플릿 사용

또한 이 CloudFormation 템플릿을 사용하여 Cruise Control 및 Prometheus를 쉽게 배포하여 Amazon MSK 클러스터의 성능에 대한 심층적인 분석을 얻고 리소스 사용률을 최적화할 수 있습니다.

주요 기능:

- Cruise Control 및 Prometheus가 사전 구성된 Amazon EC2 인스턴스의 자동 프로비저닝
- Amazon MSK 프로비저닝된 클러스터 지원
- PlainText 및 IAM을 사용한 유연한 인증
- Cruise Control에 대한 Zookeeper 종속성 없음
- Amazon S3 버킷에 저장된 자체 구성 파일을 제공하여 Prometheus 대상, Cruise Control 용량 설정 및 기타 구성을 쉽게 사용자 지정할 수 있습니다.

파티션 리밸런싱 지침

Kafka 파티션 재할당 지침

Kafka의 파티션 재할당은 리소스 집약적일 수 있습니다. 브로커 간에 중요한 데이터를 전송하여 네트워크 혼잡을 유발하고 클라이언트 작업에 영향을 미칠 수 있기 때문입니다. 다음 모범 사례는 제한 속도를 조정하고, 동시성 제어를 활용하고, 재할당 유형을 이해하여 클러스터 작업 중단을 최소화하여 파티션 재할당을 효과적으로 관리하는 데 도움이 됩니다.

> **Note**
>
> 새로 생성된 Express 기반 클러스터가 있는 경우 클러스터를 확장하거나 축소할 때 자동 파티션 배포에 지능형 리밸런싱을 사용합니다.

Cruise Control에서 동시성 관리

Cruise Control은 파티션 및 리더십 이동의 동시성을 제어하는 자동 조정 파라미터를 제공합니다. 다음 파라미터는 재할당 중에 허용 가능한 로드를 유지하는 데 도움이 됩니다.

- 최대 동시 파티션 이동: 과도한 네트워크 사용률을 방지하기 위해 동시 브로커 간 파티션 이동을 제한하도록 `num.concurrent.partition.movements.per.broker`를 정의합니다.

  Example 예제

  ```
  num.concurrent.partition.movements.per.broker = 5
  ```

  이 설정은 각 브로커가 지정된 시간에 10개 이하의 파티션을 이동하도록 제한하여 브로커 간에 로드의 균형을 맞춥니다.

스로틀링을 사용하여 대역폭 제어

- 스로틀 파라미터: `kafka-reassign-partitions.sh`를 사용하여 파티션 재할당을 수행할 때 `--throttle parameter`를 사용하여 브로커 간 데이터 이동에 대한 최대 전송 속도(초당 바이트)를 설정합니다.

  Example 예제

  ```
  --throttle 5000000
  ```

  이렇게 하면 최대 대역폭이 5MB/s로 설정됩니다.

- 밸런스 스로틀 설정: 적절한 스로틀링 비율을 선택하는 것이 중요합니다.

  너무 낮게 설정하면 재할당이 훨씬 더 오래 걸릴 수 있습니다.

  너무 높게 설정하면 클라이언트에서 지연 시간이 증가할 수 있습니다.

- 보수적인 스로틀링 비율로 시작하고 클러스터 성능 모니터링에 따라 조정합니다. 프로덕션 환경에 적용하기 전에 선택한 스로틀을 테스트하여 최적의 균형을 찾습니다.

스테이징 환경에서 테스트 및 검증

프로덕션 환경에서 재할당을 구현하기 전에 유사한 구성의 스테이징 환경에서 로드 테스트를 수행합니다. 이를 통해 파라미터를 미세 조정하고 라이브 프로덕션에서 예상치 못한 영향을 최소화할 수 있습니다.

Amazon MSK 클러스터의 구성 업데이트

클러스터의 구성을 업데이트하려면 클러스터가 `ACTIVE` 상태인지 확인합니다. 또한 MSK 클러스터의 브로커당 파티션 수가the section called " 클러스터 크기를 적절하게 조정: Standard 브로커당 파티션 수"에 설명된 제한 이하인지 확인해야 합니다. 제한을 초과하는 클러스터의 구성은 업데이트할 수 없습니다.

사용자 지정 구성을 생성하는 방법, 업데이트할 수 있는 속성, 기존 클러스터의 구성을 업데이트하면 나타나는 결과 등, MSK 구성에 대한 자세한 내용은 the section called "브로커 구성" 단원을 참조하십시오.

주제

- 구성 업데이트 중 브로커 가용성
- 를 사용하여 클러스터 구성 업데이트AWS CLI
- API를 사용하여 Amazon MSK 클러스터의 구성 업데이트

구성 업데이트 중 브로커 가용성

Amazon MSK는 대부분의 클러스터 구성 업데이트 중에 고가용성을 유지합니다. Amazon MSK는 한 번에 하나의 브로커를 업데이트하는 롤링 업데이트를 수행합니다. 이 프로세스 중에 구성이 업데이트되면 개별 브로커가 다시 시작되지만 클러스터는 계속 사용할 수 있습니다. 그러나 일부 구성 변경의 경우 모든 브로커를 동시에 업데이트해야 할 수 있으며, 이로 인해 클러스터 전체가 잠시 중단될 수 있습니다. 업데이트 중 브로커 가용성에 미치는 영향에 대한 자세한 내용은 Amazon MSK Provisioned 구성 섹션을 참조하세요.

프로덕션 클러스터를 업데이트하기 전에 비프로덕션 환경에서 구성 변경 사항을 테스트하고 유지 관리 기간 동안 업데이트를 예약하는 것이 좋습니다.

MSK 클러스터를 업그레이드하는 동안 문제가 발생하는 경우 Amazon MSK 클러스터를 업그레이드할 때 문제를 해결하려면 어떻게 해야 하나요?를 참조하세요.

를 사용하여 클러스터 구성 업데이트AWS CLI

1. 다음 JSON을 복사하여 파일에 저장합니다. 파일 이름을 `configuration-info.json`으로 지정합니다. *ConfigurationArn*을 클러스터 업데이트에 사용할 구성의 Amazon 리소스 이름(ARN)으로 바꿉니다. ARN 문자열은 다음 JSON에서 인용 부호로 묶여야 합니다.

   *Configuration-Revision*은 사용할 구성의 개정으로 바꿉니다. 구성 개정은 1 이상의 정수입니다. 이 정수는 다음 JSON에서 인용 부호로 묶이면 안 됩니다.

```
{
     "Arn": ConfigurationArn,
     "Revision": Configuration-Revision
}
```

2. 다음 명령을 실행하여 *ClusterArn*을 클러스터 생성 후 받은 ARN으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.

   *Path-to-Config-Info-File*은 구성 정보 파일의 경로로 바꿉니다. 이전 `configuration-info.json`단계에서 생성한 파일의 이름을 지정하여 현재 디렉터리에 저장한 경우 *Path-to-Config-Info-File*은 `configuration-info.json`입니다.

   *Current-Cluster-Version*을 클러스터의 현재 버전으로 바꿉니다.

> **Important**
>
> 클러스터 버전은 단순한 정수가 아닙니다. 클러스터의 현재 버전을 찾으려면 DescribeCluster 작업 또는 describe-clusterAWS CLI 명령을 사용합니다. 버전의 예를 들면 `KTVPDKIKX0DER`입니다.

```
aws kafka update-cluster-configuration --cluster-arn ClusterArn --configuration-
info file://Path-to-Config-Info-File --current-version Current-Cluster-Version
```

   다음은 이 명령을 사용하는 방법을 보여주는 예제입니다.

```
aws kafka update-cluster-configuration --cluster-arn "arn:aws:kafka:us-
east-1:0123456789012:cluster/exampleName/abcd1234-0123-abcd-5678-1234abcd-1" --
```

```
configuration-info file://c:\users\tester\msk\configuration-info.json --current-
version "K1X5R6FKA87"
```

이 update-cluster-configuration 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

3. update-cluster-configuration 작업 결과를 가져오려면 다음 명령을 실행하여 *ClusterOperationArn*을 update-cluster-configuration 명령의 출력에서 가져온 ARN으로 바꿉니다.

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

이 describe-cluster-operation 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "982168a3-939f-11e9-8a62-538df00285db",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-06-20T21:08:57.735Z",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_COMPLETE",
        "OperationType": "UPDATE_CLUSTER_CONFIGURATION",
        "SourceClusterInfo": {},
        "TargetClusterInfo": {
            "ConfigurationInfo": {
                "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/
ExampleConfigurationName/abcdabcd-abcd-1234-abcd-abcd123e8e8e-1",
                "Revision": 1
            }
        }
    }
```

```
}
```

이 출력에서 `OperationType`은 `UPDATE_CLUSTER_CONFIGURATION`입니다. `OperationState` 값이 `UPDATE_IN_PROGRESS`인 경우, 잠시 기다린 다음 `describe-cluster-operation` 명령을 다시 실행합니다.

API를 사용하여 Amazon MSK 클러스터의 구성 업데이트

API를 사용하여 Amazon MSK 클러스터의 구성을 업데이트하려면 UpdateClusterConfiguration을 참조하세요.

Amazon MSK 클러스터를 위한 브로커 재부팅

MSK 클러스터를 위해 브로커를 재부팅하려는 경우 이 Amazon MSK 작업을 사용합니다. 클러스터에 대한 브로커를 재부팅하려면 클러스터가 `ACTIVE` 상태인지 확인합니다.

Amazon MSK 서비스는 패치 또는 버전 업그레이드와 같은 시스템의 유지 관리 중에 MSK 클러스터의 브로커를 재부팅할 수 있습니다. 브로커를 수동으로 재부팅하면 Kafka 클라이언트의 복원력을 테스트하여 시스템의 유지 관리에 어떻게 반응하는지 확인할 수 있습니다.

AWS Management 콘솔을 사용하여 Amazon MSK 클러스터를 위해 브로커 재부팅

이 프로세스는 AWS Management 콘솔을 사용하여 Amazon MSK 클러스터의 브로커를 재부팅하는 방법을 설명합니다.

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 재부팅하려는 브로커가 있는 MSK 클러스터를 선택합니다.
3. 브로커 세부 정보 섹션까지 아래로 스크롤하여 재부팅하려는 브로커를 선택합니다.
4. 브로커 재부팅 버튼을 선택합니다.

AWS CLI을 사용하여 Amazon MSK 클러스터를 위해 브로커 재부팅

이 프로세스는 AWS CLI을 사용하여 Amazon MSK 클러스터의 브로커를 재부팅하는 방법을 설명합니다.

1. 다음 명령을 실행하여 *`ClusterArn`*을 클러스터를 생성할 때 받은 Amazon 리소스 이름(ARN)으로 바꾸고, *`BrokerId`*를 재부팅하려는 브로커의 ID로 변경합니다.

> **Note**
>
> 이 `reboot-broker` 작업은 한 번에 하나의 브로커만 재부팅할 수 있습니다.

클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 섹션을 참조하세요.

클러스터에 대한 브로커 ID가 없는 경우 브로커 노드를 나열하여 찾을 수 있습니다. 자세한 내용은 list-nodes를 참조하세요.

```
aws kafka reboot-broker --cluster-arn ClusterArn --broker-ids BrokerId
```

이 `reboot-broker` 작업의 출력은 다음 JSON과 같습니다.

```
{

    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```

2. `reboot-broker` 작업 결과를 가져오려면 다음 명령을 실행하여 *ClusterOperationArn*을 `reboot-broker` 명령 출력에서 가져온 ARN으로 바꿉니다.

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

이 `describe-cluster-operation` 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "c0b7af47-8591-45b5-9c0c-909a1a2c99ea",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2019-09-25T23:48:04.794Z",
```

```
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "REBOOT_IN_PROGRESS",
        "OperationType": "REBOOT_NODE",
        "SourceClusterInfo": {},
        "TargetClusterInfo": {}
    }
}
```

재부팅 작업이 완료되면 `OperationState`는 `REBOOT_COMPLETE`입니다.

API를 사용하여 Amazon MSK 클러스터를 위해 브로커 재부팅

API를 사용하여 클러스터에서 브로커를 재부팅하려면 RebootBroker 섹션을 참조하세요.

Amazon MSK 클러스터에 태그 지정

MSK 클러스터와 같은 Amazon MSK 리소스에 태그 형태로 고유한 메타데이터를 할당할 수 있습니다. 태그는 리소스에 대해 정의된 키-값 페어입니다. 태그를 사용하면 간단하면서도 효과적으로 AWS 리소스를 관리하고 결제 데이터를 포함하여 데이터를 구성할 수 있습니다.

주제

- Amazon MSK 클러스터의 태그 기본 사항
- 태그 지정을 사용하여 Amazon MSK 클러스터 비용 추적
- 태그 제한
- Amazon MSK API를 사용하여 리소스 태그 지정

Amazon MSK 클러스터의 태그 기본 사항

Amazon MSK API를 사용하여 다음 작업을 완료할 수 있습니다.

- Amazon MSK 리소스에 태그를 추가합니다.
- Amazon MSK 리소스에 대한 태그를 나열합니다.
- Amazon MSK 리소스에서 태그를 제거합니다.

태그를 사용하여 Amazon MSK 리소스를 분류할 수 있습니다. 예를 들어 용도, 소유자 또는 환경별로 Amazon MSK 클러스터를 분류할 수 있습니다. 각 태그에 대해 키와 값이 정의되기 때문에 특정 요구를

충족하는 사용자 지정 범주 세트를 생성할 수 있습니다. 예를 들어, 태그 세트를 정의하여 소유자 및 연관된 애플리케이션에 따라 클러스터를 추적할 수 있습니다.

다음은 몇 가지 태그의 예입니다.

- `Project: Project name`
- `Owner: Name`
- `Purpose: Load testing`
- `Environment: Production`

태그 지정을 사용하여 Amazon MSK 클러스터 비용 추적

태그를 사용하여 AWS 비용을 분류하고 추적할 수 있습니다. Amazon MSK 클러스터를 포함한 AWS 리소스에 태그를 적용하면 AWS 비용 할당 보고서에는 태그별로 집계된 사용량과 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

Amazon MSK의 태그에는 다음과 같은 제한 사항이 적용됩니다.

기본 제한

- 리소스당 최대 태그 수는 50개입니다.
- 태그 키와 값은 대/소문자를 구분합니다.
- 삭제된 리소스에 대해 태그를 변경하거나 편집할 수 없습니다.

태그 키 제한

- 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. AWS는 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

- 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 MSK API를 사용하여 리소스 태그 지정

다음 작업을 사용하여 Amazon MSK 리소스에 태그를 지정하거나 태그를 해제하거나 리소스에 대한 현재 태그 세트를 나열할 수 있습니다.

- ListTagsForResource
- TagResource
- UntagResource

Kafka 워크로드를 Amazon MSK 클러스터로 마이그레이션

Amazon MSK Replicator는 동일한 AWS 계정에서 Amazon MSK 클러스터 간 마이그레이션을 지원합니다. Apache Kafka 워크로드와 같은 비 MSK 클러스터 또는 교차 계정 Amazon MSK 클러스터 간 마이그레이션의 경우 Apache MirrorMaker 2.0을 사용해야 합니다.

MSK Replicator는 Amazon MSK로의 데이터 마이그레이션을 자동화하는 완전관리형 서버리스 솔루션입니다. MSK Replicator는 인프라 관리 필요성 없이 규모 조정, 모니터링 및 유지 관리 작업을 처리합니다. 또한 마이그레이션 중에 토픽 구성 및 컨슈머 그룹 오프셋을 유지하고 다른 AWS 서비스와 통합됩니다.

Apache MirrorMaker 2.0은 수동 설정 및 관리가 필요한 오픈 소스 도구이지만, 마이그레이션 프로세스를 세부적으로 제어할 수 있습니다. 호스팅 플랫폼과 관계없이 또는 다른 AWS 계정 간에 사용자 지정 복제 규칙을 정의하고 모든 Apache Kafka 클러스터 간에 마이그레이션할 수 있습니다. MirrorMaker를 사용하여 클러스터를 마이그레이션하는 방법에 대한 자세한 내용은 지리적 복제(클러스터 간 데이터 미러링)를 참조하세요. 고가용성 구성으로 MirrorMaker를 설정하는 것이 좋습니다.

워크로드에 적합한 복제 전략을 선택하는 방법에 대한 자세한 내용은 Amazon MSK Replicator 및 MirrorMaker2: Apache Kafka 재해 복구 및 마이그레이션에 적합한 복제 전략 선택을 참조하세요.

Amazon MSK 프로비저닝된 클러스터 삭제

> **Note**
>
> 프로비저닝된 Amazon MSK 클러스터에 오토 스케일링 정책이 있는 경우 클러스터를 삭제하기 전에 해당 정책을 제거하는 것이 좋습니다. 자세한 내용은 Amazon MSK 클러스터를 위한 자동 규모 조정 단원을 참조하십시오.

주제

를 사용하여 Amazon MSK 프로비저닝 클러스터 삭제AWS Management 콘솔

이 프로세스는 AWS Management 콘솔을 사용하여 Amazon MSK 프로비저닝된 클러스터를 삭제하는 방법을 설명합니다. MSK 클러스터를 삭제하기 전에 클러스터에 저장된 중요한 데이터의 백업이 있고 클러스터에 의존하는 예약된 태스크가 없는지 확인합니다. MSK 클러스터 삭제는 취소할 수 없습니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 옆에 있는 확인란을 선택하여 삭제하려는 MSK 클러스터를 선택합니다.
3. 삭제를 선택한 다음 삭제를 확인합니다.

를 사용하여 Amazon MSK 프로비저닝 클러스터 삭제AWS CLI

이 프로세스는 AWS CLI를 사용하여 MSK 프로비저닝된 클러스터를 삭제하는 방법을 설명합니다. MSK 클러스터를 삭제하기 전에 클러스터에 저장된 중요한 데이터의 백업이 있고 클러스터에 의존하는 예약된 태스크가 없는지 확인합니다. MSK 클러스터 삭제는 취소할 수 없습니다.

다음 명령을 실행하여 *ClusterArn*을 클러스터 생성 후 받은 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.

```
aws kafka delete-cluster --cluster-arn ClusterArn
```

API를 사용하여 Amazon MSK 프로비저닝된 클러스터 삭제

Amazon MSK API를 사용하면 자동화된 인프라 프로비저닝 또는 배포 스크립트의 일부로 MSK 프로비저닝된 클러스터를 프로그래밍 방식으로 생성하고 관리할 수 있습니다. 이 프로세스는 Amazon MSK API를 사용하여 Amazon MSK 프로비저닝된 클러스터를 삭제하는 방법을 설명합니다. Amazon MSK 클러스터를 삭제하기 전에 클러스터에 저장된 중요한 데이터의 백업이 있고 클러스터에 따라 예약된 태스크가 없는지 확인합니다. MSK 클러스터 삭제는 취소할 수 없습니다.

API를 사용하여 클러스터를 삭제하려면 DeleteCluster를 참조하십시오.

Amazon MSK 주요 기능 및 개념

Amazon MSK Provisioned 클러스터는 클러스터의 성능을 최적화하고 스트리밍 요구 사항을 충족하는 데 도움이 되는 다양한 기능을 제공합니다. 아래 주제에서는 이러한 기능에 대해 자세히 설명합니다.

- AWS Management 콘솔은
- Amazon MSK API 참조
- Amazon MSK CLI 명령 참조

주제

- Amazon MSK 브로커 유형
- Amazon MSK 브로커 크기
- Standard 브로커의 스토리지 관리
- Amazon MSK Provisioned 구성
- 클러스터의 지능형 리밸런싱
- MSK Provisioned 클러스터에 패치 적용
- 브로커 오프라인 및 클라이언트 장애 조치
- Amazon MSK의 보안
- Amazon MSK 로깅
- 메타데이터 관리
- 주제 작업
- Amazon MSK 리소스
- Apache Kafka 버전
- Amazon MSK 클러스터 문제 해결

Amazon MSK 브로커 유형

MSK Provisioned는 Standard와 Express라는 두 가지 브로커 유형을 제공합니다. Standard 브로커는 클러스터를 가장 유연하게 구성할 수 있는 반면, Express 브로커는 고성능 스트리밍 애플리케이션 실행을 위한 탄력성, 처리량, 복원력, 사용 편의성을 제공합니다.

각 유형에 대한 자세한 내용은 다음 토픽을 참조하세요. 다음 표에는 Standard 브로커와 Express 브로커 간의 주요 기능을 비교한 내용도 강조되어 있습니다.

기능	Standard 브로커	Express 브로커
스토리지 관리	고객 관리형(특징: EBS 스토리지, 계층형 스토리지, 프로비저닝된 스토리지 처리량, 오토 스케일링, 스토리지 용량 알림)	완전 MSK 관리형
지원되는 인스턴스	T3, M5, M7g	M7g
크기 및 규모 조정 고려 사항	처리량, 연결, 파티션, 스토리지	처리량, 연결, 파티션
브로커 규모 조정	수직 및 수평 규모 조정	수직 및 수평 규모 조정
Kafka 버전	Apache Kafka 버전 섹션을 참조하세요	버전 3.6에서 시작
Apache Kafka 구성	추가 구성 가능	복원력 향상을 위해 대부분 MSK 관리형임
보안	암호화, 프라이빗/퍼블릭 액세스, 인증 및 권한 부여 - IAM, SASL/SCRAM, mTLS, 일반 텍스트, Kafka ACL	암호화, 프라이빗/퍼블릭 액세스, 인증 및 권한 부여 - IAM, SASL/SCRAM, mTLS, 일반 텍스트, Kafka ACL
모니터링	CloudWatch, 오픈 모니터링	CloudWatch, 오픈 모니터링

> **Note**
>
> MSK API를 사용하여 브로커 유형을 전환하면 MSK 프로비저닝 클러스터를 Standard 브로커 유형에서 Express 브로커 유형으로 변경할 수 없습니다. 원하는 브로커 유형(Standard 또는 Express)으로 새 클러스터를 생성해야 합니다.

주제

- Amazon MSK Standard 브로커
- Amazon MSK Express 브로커

Amazon MSK Standard 브로커

MSK 프로비저닝용 Standard 브로커는 클러스터의 성능을 구성할 수 있는 가장 유연한 기능을 제공합니다. 광범위한 클러스터 구성 중에서 선택하여 애플리케이션에 필요한 가용성, 내구성, 처리량 및 지연 시간 특성을 달성할 수 있습니다. 스토리지 용량을 프로비저닝하고 필요에 따라 늘릴 수도 있습니다. Amazon MSK는 Standard 브로커 및 연결된 스토리지 리소스의 하드웨어 유지 관리를 처리하여 발생할 수 있는 하드웨어 문제를 자동으로 복구합니다. 스토리지 관리, 구성 및 유지 관리에 대한 주제를 포함하여 Standard 브로커 관련 여러 주제에 대한 자세한 내용은 이 문서에서 확인할 수 있습니다.

Amazon MSK Express 브로커

MSK용 Express 브로커 프로비저닝을 사용하면 Apache Kafka를 더 간단하게 관리하고 더 비용 효율적으로 대규모로 실행할 수 있으며, 기대를 충족하는 짧은 지연 시간으로 더 탄력적으로 운영할 수 있습니다. 브로커에는 자동으로 확장되고 크기 조정, 프로비저닝 또는 사전 예방적 모니터링이 필요하지 않은 종량제 스토리지가 포함됩니다. 선택한 인스턴스 크기에 따라 각 브로커 노드는 표준 Apache Kafka 브로커에 비해 브로커당 최대 3배 더 많은 처리량을 제공하고, 최대 20배 더 빠르게 스케일 업하고, 90% 더 빠르게 복구할 수 있습니다. Express 브로커는 Amazon MSK의 모범 사례 기본값으로 사전 구성되어 있으며, 클라이언트 처리량 할당량을 적용하여 클라이언트와 Kafka의 백그라운드 작업 간의 리소스 경합을 최소화합니다.

다음은 Express 브로커를 사용할 때 고려해야 할 몇 가지 주요 요인과 기능입니다.

- 스토리지 관리 없음: Express 브로커는 스토리지 리소스를 프로비저닝하거나 관리할 필요가 없습니다. 탄력적, 사실상 무제한, 종량제 및 완전관리형 스토리지를 사용할 수 있습니다. 처리량이 많은 사용 사례의 경우 컴퓨팅 인스턴스와 스토리지 볼륨 간의 상호 작용과 관련 처리량 병목 현상을 추론할 필요가 없습니다. 이러한 기능은 클러스터 관리를 간소화하고 스토리지 관리 운영 오버헤드를 제거합니다.

- 더 빠른 조정: Express 브로커를 사용하면 Standard 브로커보다 최대 20배 빠르게 클러스터를 조정하고 파티션을 이동할 수 있습니다. 이 기능은 예정된 로드 급증을 처리하기 위해 클러스터를 스케일 아웃하거나 비용을 줄이기 위해 클러스터를 스케일 인해야 하는 경우에 매우 중요합니다. 클러스터 규모 조정에 대한 자세한 내용은 클러스터 확장, 브로커 제거, 파티션 재할당 및 재분배를 위한 LinkedIn의 Cruise Control 설정에 대한 섹션을 참조하세요.
- 높은 처리량: Express 브로커는 Standard 브로커보다 브로커당 최대 3배 더 많은 처리량을 제공합니다. 예를 들어 각 m7g.16xlarge 크기의 Express 브로커로 최대 500MBps의 데이터를 안전하게 쓸 수 있는 데 비해 동등한 Standard 브로커의 경우 153.8MBps입니다(두 수치 모두 복제 및 리밸런싱과 같은 백그라운드 작업에 충분한 대역폭이 할당되었다고 가정한 것임).
- 높은 복원력을 위한 구성: Express 브로커는 클러스터의 복원력을 개선하기 위해 다양한 모범 사례를 자동으로 제공합니다. 여기에는 중요한 Apache Kafka 구성에 대한 가드레일, 처리량 할당량, 백그라운드 운영 및 계획되지 않은 수리를 위한 용량 예약이 포함됩니다. 이러한 기능을 사용하면 대규모 Apache Kafka 애플리케이션을 더 안전하고 쉽게 실행할 수 있습니다. 자세한 내용은 Express 브로커 구성 및 Amazon MSK Express 브로커 할당량 섹션을 참조하세요.
- 유지 관리 기간 불필요: Express 브로커에 대한 유지 관리 기간이 불필요합니다. Amazon MSK는 클러스터 하드웨어를 지속적으로 자동으로 업데이트합니다. 자세한 내용은 Express 브로커 패치 적용을 참조하세요.

Express 브로커에 대한 추가 정보

- Express 브로커는 Apache Kafka API에서 작동하지만 아직 KStreams API를 완전히 지원하지 않습니다.
- Express 브로커는 3AZ 구성에서만 사용할 수 있습니다.
- Express 브로커는 일부 인스턴스 크기에서만 사용할 수 있습니다. 업데이트된 목록은 Amazon MSK 요금을 참조하세요.
- Express 브로커는 3.6, 3.8 및 3.9 버전의 Apache Kafka에서 지원됩니다.
- Express 브로커는 Apache Kafka 버전 3.9 이상에서 KRaft 모드로 생성할 수 있습니다.

> 관련 블로그 보기
>
> MSK Express 브로커에 대한 자세한 내용과 사용 중인 Express 브로커의 실제 예를 보려면 다음 블로그를 참조하세요.
>
> - Kafka 클러스터에 높은 처리량과 더 빠른 크기 조정을 제공하는 Amazon MSK용 Express 브로커 소개

- [Amazon MSK용 Express 브로커: 최대 20배 더 빠른 고성능 Kafka 규모 조정]()

  이 블로그는 Express 브로커가 다음을 수행하는 방법을 보여줍니다.

  - 더 빠른 처리량, 신속한 규모 조정 및 장애로 인한 복구 시간 개선
  - 스토리지 관리 복잡성 제거

Amazon MSK 브로커 크기

Amazon MSK 프로비저닝된 클러스터를 생성하는 경우 클러스터에 포함할 브로커 크기를 지정합니다. [브로커 유형]()에 따라 Amazon MSK는 다음 브로커 크기를 지원합니다.

Standard 브로커 크기

- kafka.t3.small
- kafka.m5.large, kafka.m5.xlarge, kafka.m5.2xlarge, kafka.m5.4xlarge, kafka.m5.8xlarge, kafka.m5.12xlarge, kafka.m5.16xlarge, kafka.m5.24xlarge
- kafka.m7g.large, kafka.m7g.xlarge, kafka.m7g.2xlarge, kafka.m7g.4xlarge, kafka.m7g.8xlarge, kafka.m7g.12xlarge, kafka.m7g.16xlarge

Express 브로커 크기

- express.m7g.large, express.m7g.xlarge, express.m7g.2xlarge, express.m7g.4xlarge, express.m7g.8xlarge, express.m7g.12xlarge, express.m7g.16xlarge

> **Note**
>
> 일부 브로커 크기는 특정 AWS 리전에서 이용하지 못할 수 있습니다. 리전별로 이용 가능한 인스턴스의 최신 목록은 [Amazon MSK 요금 페이지]()에서 업데이트된 브로커 인스턴스 요금 테이블을 참조하세요.

브로커 크기에 대한 기타 참고 사항

- M7g 브로커는 AWS Graviton 프로세서(Amazon Web Services에서 빌드한 사용자 지정 Arm 기반 프로세서)를 사용합니다. M7g 브로커는 비슷한 M5 인스턴스에 비해 향상된 가격 대비 성능을 제공합니다. M7g 브로커는 유사한 M5 인스턴스보다 적은 전력을 소비합니다.

- Amazon MSK는 2.8.2 및 3.3.2 이상의 Kafka 버전을 실행하는 MSK 프로비저닝된 클러스터에서 M7g 브로커를 지원합니다.
- M7g 및 M5 브로커는 T3 브로커보다 기본 처리량 성능이 높으며 프로덕션 워크로드에 권장됩니다. 또한 M7g 및 M5 브로커는 T3 브로커보다 브로커당 파티션 수가 더 많을 수 있습니다. 더 대규모의 프로덕션급 워크로드를 실행 중이거나 더 많은 수의 파티션이 필요한 경우 M7g 및 M5 브로커를 사용하세요. M7g 및 M5 인스턴스 크기에 대한 자세한 내용은 Amazon EC2 범용 인스턴스를 참조하세요.
- T3 브로커는 CPU 크레딧을 사용하여 성능을 일시적으로 버스트할 수 있습니다. 저비용 개발의 경우, 중소 규모의 스트리밍 워크로드를 테스트 중인 경우 또는 처리량이 일시적으로 급증하는 낮은 처리량 스트리밍 워크로드가 있는 경우 T3 브로커를 사용하십시오. T3 브로커가 프로덕션 또는 중요한 워크로드에 충분한지 확인하기 위해 개념 증명 테스트를 실행하는 것이 좋습니다. T3 브로커 크기에 대한 자세한 내용은 Amazon EC2 T3 인스턴스를 참조하세요.

브로커 크기를 선택하는 방법에 대한 자세한 내용은 Standard 및 Express 브로커 모범 사례 단원을 참조하세요.

Standard 브로커의 스토리지 관리

Amazon MSK는 MSK 클러스터의 스토리지 관리에 도움이 되는 기능을 제공합니다.

> **Note**
>
> Express 브로커를 사용하면 데이터에 사용되는 스토리지 리소스를 프로비저닝하거나 관리할 필요가 없습니다. 이를 통해 클러스터 관리가 간소화되고 Apache Kafka 클러스터 운영 문제의 일반적인 원인 중 하나가 제거됩니다. 또한 유휴 스토리지 용량을 프로비저닝할 필요가 없고 사용한 만큼만 비용을 지불하면 되므로 비용을 절감할 수 있습니다.

Standard 브로커 유형

Standard 브로커를 사용하면 다양한 스토리지 옵션과 기능 중에서 선택할 수 있습니다. Amazon MSK는 MSK 클러스터의 스토리지 관리에 도움이 되는 기능을 제공합니다.

처리량 관리에 대한 자세한 내용은 ??? 섹션을 참조하세요.

주제

- Standard 브로커를 위한 계층형 스토리지
- Amazon MSK Standard 브로커 스토리지 스케일 업

- Amazon MSK 클러스터의 Standard 브로커에 대한 스토리지 처리량 관리

Standard 브로커를 위한 계층형 스토리지

계층형 스토리지는 사실상 무제한 스토리지로 확장할 수 있어 스트리밍 데이터 애플리케이션을 비용 효율적으로 빌드할 수 있는 Amazon MSK의 저비용 스토리지 계층입니다.

성능과 비용의 균형을 맞추는 계층형 스토리지로 구성된 Amazon MSK 클러스터를 생성할 수 있습니다. Amazon MSK는 Apache Kafka 주제 보존 한도에 도달할 때까지 스트리밍 데이터를 성능에 최적화된 기본 스토리지 계층에 저장합니다. 그러면 Amazon MSK가 자동으로 데이터를 새로운 저비용 스토리지 계층으로 이동합니다.

애플리케이션이 계층형 스토리지에서 데이터를 읽기 시작하면 처음 몇 바이트 동안 읽기 지연 시간이 늘어날 수 있습니다. 저비용 계층에서 나머지 데이터를 순차적으로 읽기 시작하면 기본 스토리지 계층과 비슷한 수준의 지연 시간을 예상할 수 있습니다. 저비용 계층형 스토리지를 위해 스토리지를 프로비저닝하거나 인프라를 관리할 필요가 없습니다. 원하는 만큼만 데이터를 저장하고 사용한 만큼만 비용을 지불할 수 있습니다. 이 기능은 KIP-405: Kafka 계층화된 스토리지에 도입된 API와 호환됩니다.

MSK 계층형 스토리지 클러스터의 규모 조정, 모니터링 및 최적화에 대한 자세한 내용은 Amazon MSK 계층형 스토리지를 사용하여 프로덕션 워크로드를 실행하는 모범 사례를 참조하세요.

다음은 계층형 스토리지의 몇 가지 기능입니다.

- 스토리지 규모를 거의 무제한으로 조정할 수 있습니다. Apache Kafka 인프라의 규모를 조정하는 방법을 추측할 필요가 없습니다.
- 브로커 수를 늘리지 않고도 Apache Kafka 주제에 데이터를 더 오래 보관하거나 주제 저장 공간을 늘릴 수 있습니다.
- 예기치 않은 처리 지연을 처리하기 위해 더 긴 기간의 안전 버퍼를 제공합니다.
- 기존 스트림 처리 코드와 Kafka API를 사용하여 정확한 생산 순서대로 오래된 데이터를 재처리할 수 있습니다.
- 보조 스토리지의 데이터는 브로커 디스크 간에 복제할 필요가 없으므로 파티션이 더 빠르게 재조정됩니다.
- 브로커와 계층화된 스토리지 간의 데이터는 VPC 내에서 이동하며 인터넷을 통해 이동하지 않습니다.
- 클라이언트 머신은 계층형 스토리지가 활성화되지 않은 클러스터에 연결할 때와 동일한 프로세스를 사용하여 계층형 스토리지가 활성화된 새 클러스터에 연결할 수 있습니다. 클라이언트 머신 생성을 참조하세요.

Amazon MSK 클러스터에 대한 계층형 스토리지 요구 사항

- 계층형 스토리지를 사용하도록 설정한 새 주제를 생성하려면 Apache Kafka 클라이언트 버전 3.0.0 이상을 사용해야 합니다. 기존 토픽을 계층형 스토리지로 전환하려면, 3.0.0(지원되는 최소 Apache Kafka 버전은 2.8.2.tiered)보다 낮은 Kafka 클라이언트 버전을 사용하는 클라이언트 머신을 재구성하여 계층형 스토리지를 사용하도록 설정할 수 있습니다. 4단계: Amazon MSK 클러스터에서 주제 생성을(를) 참조하세요.
- 계층형 스토리지가 활성화된 Amazon MSK 클러스터는 버전 3.6.0 이상 또는 2.8.2.tiered를 사용해야 합니다.

Amazon MSK 클러스터에 대한 계층형 스토리지 제약 조건 및 제한 사항

계층형 스토리지에는 다음과 같은 제약과 제한 사항이 있습니다.

- 애플리케이션이 트랜잭션 기능을 적극적으로 사용하지 않는 한 Amazon MSK의 remote_tier에서 읽을 때 클라이언트에서 `read_committed`에 구성되지 않았는지 확인합니다.
- 계층형 스토리지는AWSGovCloud(미국) 리전에서 사용할 수 없습니다.
- 계층형 스토리지는 프로비저닝 모드 클러스터에만 적용됩니다.
- 계층형 스토리지는 브로커 크기 t3.small을 지원하지 않습니다.
- 저비용 스토리지의 최소 보존 기간은 3일입니다. 기본 스토리지에는 최소 보존 기간이 없습니다.
- 계층형 스토리지는 브로커에서 다중 로그 디렉터리(JBOD 관련 기능)를 지원하지 않습니다.
- 계층형 스토리지는 압축된 주제를 지원하지 않습니다. 계층형 스토리지가 설정된 모든 주제의 cleanup.policy가 'DELETE'로만 구성되어 있는지 확인합니다.
- 계층형 스토리지 클러스터는 주제가 생성된 후 주제에 대한 log.cleanup.policy 정책 변경을 지원하지 않습니다.
- 계층형 스토리지는 개별 주제에 대해 비활성화할 수 있지만 전체 클러스터에서는 비활성화할 수 없습니다. 일단 비활성화되면 주제에 대해 계층형 스토리지를 다시 활성화할 수 없습니다.
- Amazon MSK 버전 2.8.2.tiered를 사용하는 경우 다른 계층형 스토리지를 지원하는 Apache Kafka 버전으로만 마이그레이션할 수 있습니다. 계층형 스토리지 지원 버전을 계속 사용하지 않으려면 새로운 MSK 클러스터를 생성하고 해당 클러스터로 마이그레이션하세요.
- kafka-log-dirs 도구는 계층화된 스토리지 데이터 크기를 보고할 수 없습니다. 이 도구는 기본 스토리지에 있는 로그 세그먼트의 크기만 보고합니다.

주제 수준에서 계층형 스토리지를 구성할 때 기본 설정 및 제약 조건에 대한 자세한 내용은 Amazon MSK 계층형 스토리지 주제 수준의 구성에 대한 지침 섹션을 참조하세요.

Amazon MSK 주제에 대해 로그 세그먼트를 계층형 스토리지로 복사하는 방법

새 주제 또는 기존 주제에 대해 계층형 스토리지를 활성화하면 Apache Kafka가 기본 스토리지에서 계층형 스토리지로 닫힌 로그 세그먼트를 복사합니다.

- Apache Kafka는 닫힌 로그 세그먼트만 복사합니다. 로그 세그먼트 내의 모든 메시지를 계층화된 스토리지에 복사합니다.
- 활성 세그먼트는 계층화 대상이 아닙니다. 로그 세그먼트 크기(segment.bytes) 또는 세그먼트 롤링 시간(segment.ms)은 세그먼트 닫기 속도를 제어하며 Apache Kafka가 세그먼트를 계층형 스토리지로 복사하는 속도도 제어합니다.

계층형 스토리지가 활성화된 주제의 보존 설정은 계층형 스토리지가 활성화되지 않은 주제의 설정과 다릅니다. 다음 규칙은 계층형 스토리지가 활성화된 주제의 메시지 보존을 제어합니다.

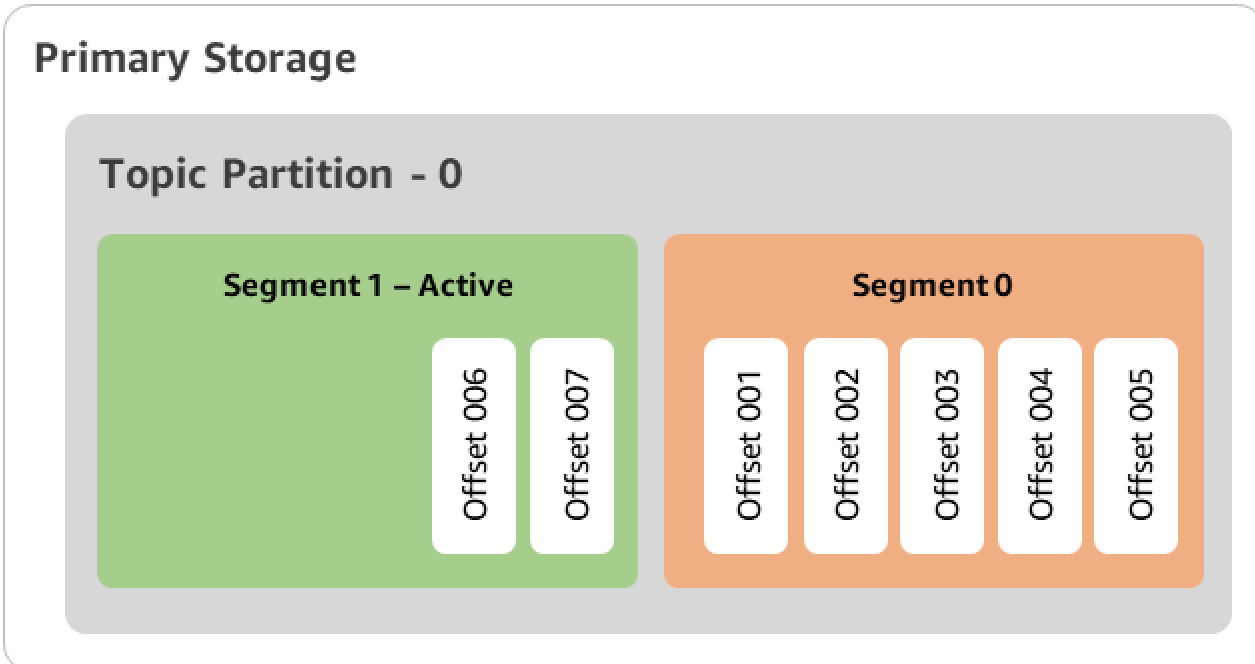
- Apache Kafka에서 보존은 log.retention.ms(시간)와 log.retention.bytes(크기)의 두 가지 설정으로 정의할 수 있습니다. 이러한 설정은 Apache Kafka가 클러스터에 보관하는 데이터의 총 기간과 크기를 결정합니다. 계층형 스토리지 모드 사용 여부에 관계없이 이러한 구성은 클러스터 수준에서 설정합니다. 주제 구성으로 주제 수준에서 설정을 재정의할 수 있습니다.
- 계층형 스토리지를 사용하도록 설정하면 기본 고성능 스토리지 계층에서 데이터를 저장하는 기간을 추가로 지정할 수 있습니다. 예를 들어 토픽의 전체 보존(log.retention.ms) 설정이 7일이고 로컬 보존(local.retention.ms)이 12시간인 경우 클러스터 기본 스토리지에는 처음 12시간 동안만 데이터를 보존합니다. 저비용 스토리지 계층은 7일 동안 데이터를 보존합니다.
- 일반 보존 설정이 전체 로그에 적용됩니다. 여기에는 계층화된 부분과 기본 부분이 포함됩니다.
- local.retention.ms 또는 local.retention.bytes 설정은 기본 저장소의 메시지 보존을 제어합니다. 전체 로그에서 데이터가 기본 스토리지 유지 설정 임계값(local.retention.ms/bytes)에 도달하면 Apache Kafka는 기본 스토리지의 데이터를 계층화된 스토리지로 복사합니다. 그러면 데이터가 만료될 수 있습니다.
- 로그 세그먼트의 메시지를 계층형 스토리지에 복사할 때 Apache Kafka는 retention.ms 또는 retention.bytes 설정에 따라 클러스터에서 메시지를 제거합니다.

Amazon MSK 계층형 스토리지 시나리오 예제

이 시나리오는 계층화된 스토리지가 활성화될 때 기본 스토리지에 메시지가 있는 기존 주제가 어떻게 작동하는지 보여줍니다. 이 주제에서 계층형 스토리지를 사용하려면 remote.storage.enable을 `true`로 설정하면 됩니다. 이 예제에서 retention.ms는 5일로 설정되고 local.retention.ms는 2일로 설정됩니다. 다음은 세그먼트가 만료될 때 발생하는 이벤트 순서입니다.

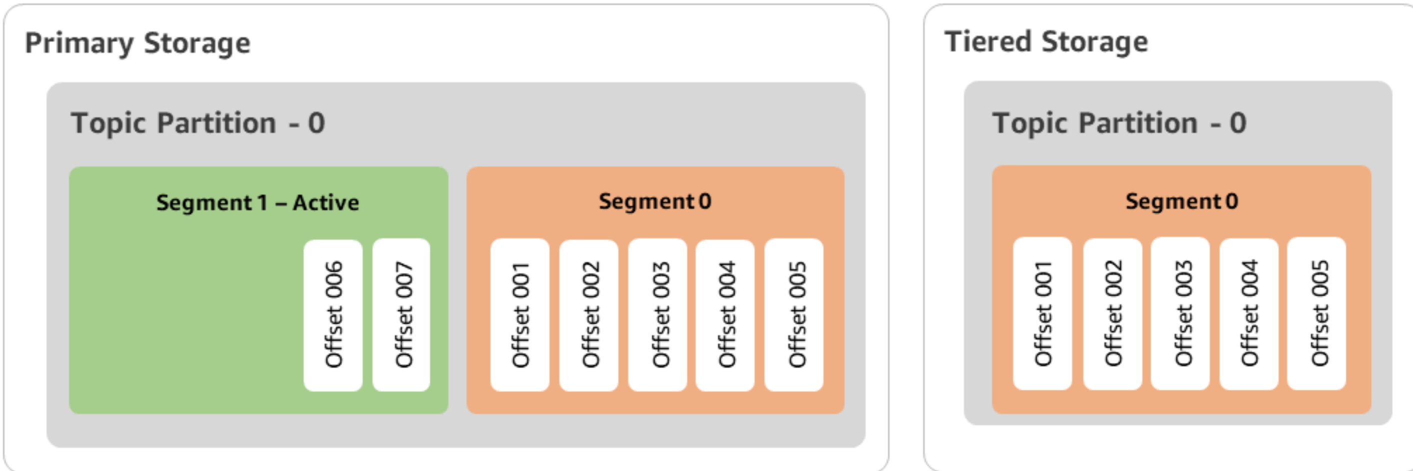
시간 T0 - 계층형 스토리지를 활성화하기 전.

이 주제에 대해 계층형 스토리지를 활성화하기 전에 2개의 로그 세그먼트가 있습니다. 세그먼트 중 하나가 기존 토픽 파티션 0에 대해 활성화되어 있습니다.



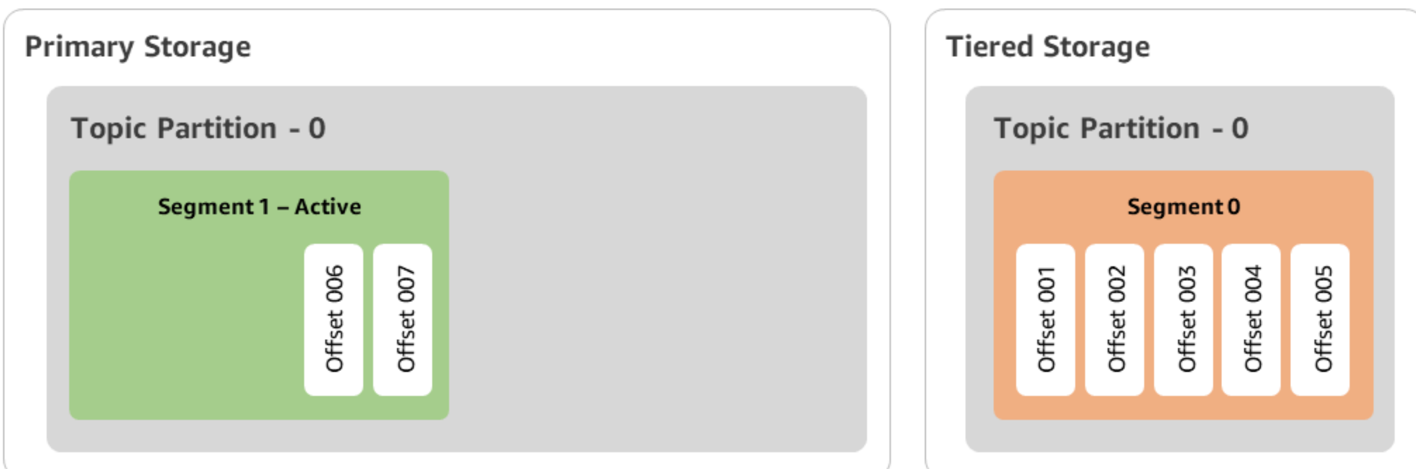
시간 T1(2일 미만) - 계층형 스토리지가 활성화되었으며, 세그먼트 0이 계층형 스토리지에 복사되었습니다.

이 주제에 대해 계층형 스토리지를 활성화하면 로그 세그먼트 0이 초기 보존 설정을 충족한 후 Apache Kafka가 계층형 스토리지에 로그 세그먼트 0을 복사합니다. 또한 Apache Kafka는 세그먼트 0의 기본 스토리지 복사본을 유지합니다. 아직 활성 세그먼트 1은 계층형 스토리지로 복사할 수 없습니다. 이 타임라인에서 Amazon MSK는 세그먼트 0 및 세그먼트 1의 메시지에 대해 아직 어떤 보존 설정도 적용하지 않습니다. (local.retention.bytes/ms, retention.ms/bytes)



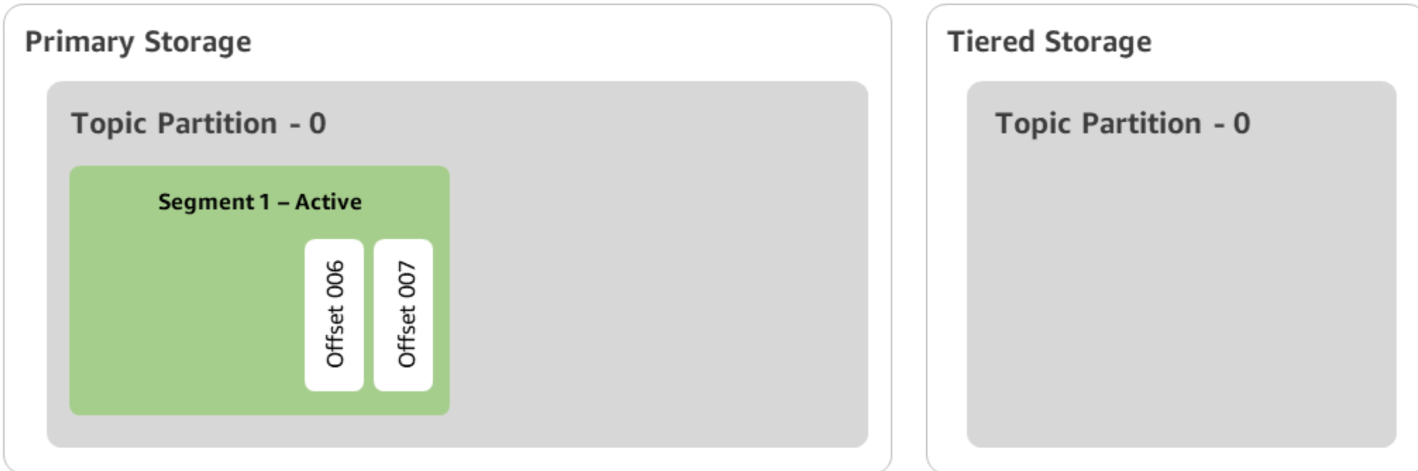
시간 T2 - 로컬 보존이 적용됨.

2일 후에는 Apache Kafka가 계층형 스토리지에 복사한 세그먼트 0에 대한 기본 보존 설정이 적용됩니다. local.retention.ms를 2일로 설정하면 이것이 결정됩니다. 이제 세그먼트 0은 기본 스토리지에서 만료됩니다. 활성 세그먼트 1은 아직 만료 대상도 아니고 계층형 스토리지로 복사할 수 있는 대상도 아닙니다.



시간 T3 - 전체 보존이 유효함.

5일 후에는 보존 설정이 적용되고 Kafka는 계층형 스토리지에서 로그 세그먼트 0과 관련 메시지를 삭제합니다. 세그먼트 1은 아직 활성이므로 만료 대상도 아니고 계층형 스토리지로 복사할 수 있는 대상도 아닙니다. 세그먼트 1은 아직 닫히지 않았으므로 세그먼트 롤에 사용할 수 없습니다.



를 사용하여 계층형 스토리지로 Amazon MSK 클러스터 생성AWS Management 콘솔

이 프로세스는 AWS Management 콘솔을 사용하여 계층형 스토리지 Amazon MSK 클러스터를 생성하는 방법을 설명합니다.

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 클러스터 생성을 선택합니다.
3. 계층형 스토리지에 대해 사용자 지정 생성을 선택합니다.
4. 클러스터의 이름을 지정합니다.
5. 클러스터 유형에서 프로비저닝을 선택합니다.
6. 클러스터를 생성하는 데 사용할 Amazon MSK 계층형 스토리지를 지원하는 Amazon Kafka 버전을 선택합니다.
7. kafka.t3.small 이외의 브로커 크기를 지정합니다.
8. 각 가용 영역에서 Amazon MSK가 생성할 브로커 수를 선택합니다. 최소는 가용 영역당 브로커 1개, 최대는 클러스터당 브로커 30개입니다.
9. 브로커가 배포되는 영역의 수를 지정합니다.
10. 영역당 배포되는 Apache Kafka 브로커의 수를 지정합니다.
11. 스토리지 옵션을 선택합니다. 여기에는 계층형 스토리지 및 EBS 스토리지가 포함되어 계층화된 스토리지 모드를 사용할 수 있습니다.
12. 클러스터 생성 마법사의 나머지 단계를 수행합니다. 완료되면 계층형 스토리지 및 EBS 스토리지가 검토 및 생성 보기에 클러스터 스토리지 모드로 나타납니다.
13. [클러스터 생성(Create cluster)]을 선택합니다.

를 사용하여 계층형 스토리지로 Amazon MSK 클러스터 생성AWS CLI

클러스터에서 계층형 스토리지를 사용하려면 계층형 스토리지에 적합한 Apache Kafka 버전과 속성을 사용하여 클러스터를 생성하세요. 아래 코드 예시를 참조하세요. 또한 다음 섹션의 단계를 를 사용하여 계층형 스토리지가 활성화된 Kafka 주제 생성AWS CLI로 완료합니다.

클러스터 생성에 지원되는 전체 속성 목록은 create-cluster를 참조하세요.

```
aws kafka create-cluster \
 —cluster-name "MessagingCluster" \
 —broker-node-group-info file://brokernodegroupinfo.json \
 —number-of-broker-nodes 3 \
--kafka-version "3.6.0" \
--storage-mode "TIERED"
```

를 사용하여 계층형 스토리지가 활성화된 Kafka 주제 생성AWS CLI

계층형 스토리지를 사용하도록 설정한 클러스터를 만들 때 시작한 프로세스를 완료하려면 이후 코드 예제에서 속성을 사용하여 계층형 스토리지를 사용하도록 설정한 주제도 생성합니다. 계층형 스토리지에 대한 구체적인 속성은 다음과 같습니다.

- 시간 기반 보존 설정의 경우 `local.retention.ms`(예: 10분), 로그 세그먼트 크기 제한의 경우 `local.retention.bytes`입니다.
- `remote.storage.enable`을 `true`로 설정하여 계층형 스토리지를 활성화합니다.

다음 구성에서는 local.retention.ms를 사용하지만 이 속성은 local.retention.bytes로 대체할 수 있습니다. 이 속성은 Apache Kafka가 기본 스토리지에서 계층형 스토리지로 데이터를 복사하기 전 Apache Kafka가 복사할 수 있는 바이트 수 또는 경과할 수 있는 시간을 제어합니다. 지원되는 구성 속성에 대한 자세한 내용은 주제 수준 구성을 참조하세요.

> **Note**
>
> Apache Kafka 클라이언트 버전 3.0.0 이상을 사용해야 합니다. 이러한 버전은 해당 클라이언트 버전 `kafka-topics.sh`에서만 `remote.storage.enable`이라는 설정을 지원합니다. 이전 버전의 Apache Kafka를 사용하는 기존 주제에서 계층형 스토리지를 활성화하려면 기존 Amazon MSK 주제에서 계층형 스토리지 활성화 섹션을 참조하세요.

```
bin/kafka-topics.sh --create --bootstrap-server $bs --replication-factor 2
 --partitions 6 --topic MSKTutorialTopic --config remote.storage.enable=true
 --config local.retention.ms=100000 --config retention.ms=604800000 --config
 segment.bytes=134217728
```

기존 Amazon MSK 주제에서 계층형 스토리지 활성화 및 비활성화

이 섹션에서는 이미 만든 주제에서 계층형 스토리지를 활성화 및 비활성화하는 방법에 대해 설명합니다. 계층형 스토리지가 활성화된 상태에서 새 클러스터와 주제를 만들려면 AWS Management 콘솔을 사용하여 계층형 스토리지로 클러스터 생성을 참조하세요.

기존 Amazon MSK 주제에서 계층형 스토리지 활성화

기존 주제에서 계층형 스토리지를 사용 설정하려면 다음 예제의 `alter` 명령 구문을 사용합니다. 이미 존재하는 주제에서 계층형 스토리지를 활성화하는 경우 특정 Apache Kafka 클라이언트 버전으로 제한되지 않습니다.

```
bin/kafka-configs.sh --bootstrap-server $bsrv --alter --entity-type topics
 --entity-name msk-ts-topic --add-config 'remote.storage.enable=true,
 local.retention.ms=604800000, retention.ms=15550000000'
```

기존 Amazon MSK 주제에서 계층형 스토리지 비활성화

기존 주제에서 계층형 스토리지를 비활성화하려면 계층형 스토리지를 사용 설정할 때와 동일한 순서로 `alter` 명령 구문을 사용합니다.

```
bin/kafka-configs.sh --bootstrap-server $bs --alter --entity-type topics --
entity-name MSKTutorialTopic --add-config 'remote.log.msk.disable.policy=Delete,
 remote.storage.enable=false'
```

> **Note**
>
> 계층형 스토리지를 비활성화하면 계층형 스토리지에서 주제 데이터가 완전히 삭제됩니다. Apache Kafka는 기본 스토리지 데이터를 보존하지만 여전히 `local.retention.ms`를 기반으로 하는 기본 보존 규칙을 적용합니다. 주제에서 계층형 스토리지를 비활성화한 후에는 다시 활성화할 수 없습니다. 기존 토픽에서 계층형 스토리지를 비활성화하려는 경우 특정 Apache Kafka 클라이언트 버전으로 제한되지 않습니다.

AWSCLI를 사용하여 기존 Amazon MSK 클러스터에서 계층형 스토리지 활성화

> **Note**
>
> 계층형 스토리지에서는 압축된 주제가 지원되지 않으므로 클러스터의 log.cleanup.policy가 `delete`로 설정된 경우에만 계층형 스토리지를 활성화할 수 있습니다. 나중에 특정 주제에서 계층형 스토리지가 활성화되어 있지 않은 경우 개별 주제의 log.cleanup.policy를 `compact`로 구성할 수 있습니다. 지원되는 구성 속성에 대한 자세한 내용은 주제 수준 구성을 참조하세요.

1. Kafka 버전 업데이트 – 클러스터 버전은 단순한 정수가 아닙니다. 클러스터의 현재 버전을 찾으려면 `DescribeCluster` 작업 또는 `describe-cluster`AWSCLI 명령을 사용합니다. 버전의 예를 들면 `KTVPDKIKX0DER`입니다.

```
aws kafka update-cluster-kafka-version --cluster-arn ClusterArn --current-version
 Current-Cluster-Version --target-kafka-version 3.6.0
```

2. 클러스터 스토리지 모드를 편집합니다. 다음 코드 예제는 `update-storage` API를 사용하여 클러스터 스토리지 모드를 `TIERED`로 편집하는 방법을 보여줍니다.

```
aws kafka update-storage --current-version Current-Cluster-Version --cluster-arn
 Cluster-arn --storage-mode TIERED
```

콘솔을 사용하여 기존 Amazon MSK 클러스터에서 계층형 스토리지 업데이트

이 프로세스는 AWS Management 콘솔을 사용하여 계층형 스토리지 Amazon MSK 클러스터를 업데이트하는 방법을 설명합니다.

MSK 클러스터의 현재 Apache Kafka 버전이 2.8.2.tiered인지 확인합니다. MSK 클러스터를 2.8.2.tiered 버전으로 업그레이드해야 하는 경우 Apache Kafka 버전 업데이트를 참조하세요.

> **Note**
>
> 계층형 스토리지에서는 압축된 주제가 지원되지 않으므로 클러스터의 log.cleanup.policy가 `delete`로 설정된 경우에만 계층형 스토리지를 활성화할 수 있습니다. 나중에 특정 주제에서 계층형 스토리지가 활성화되어 있지 않은 경우 개별 주제의 log.cleanup.policy를 `compact`로 구성할 수 있습니다. 지원되는 구성 속성에 대한 자세한 내용은 주제 수준 구성을 참조하세요.

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 클러스터 요약 페이지로 이동하여 속성을 선택합니다.
3. 스토리지 섹션으로 이동하여 클러스터 스토리지 모드 편집을 선택합니다.
4. 계층형 스토리지 및 EBS 스토리지와 변경 사항 저장을 선택합니다.

Amazon MSK Standard 브로커 스토리지 스케일 업

브로커당 EBS 스토리지의 양을 늘릴 수 있습니다. 스토리지를 줄일 수 없습니다.

스토리지 볼륨은 이 확장 작업 중에 계속 사용할 수 있습니다.

> **Important**
>
> MSK 클러스터를 위해 스토리지의 규모를 조정하면 추가 스토리지를 즉시 사용할 수 있습니다. 그러나 모든 스토리지 규모 조정 이벤트 후에 클러스터에는 휴지 기간이 필요합니다. Amazon MSK는 이 휴지 기간을 사용하여 클러스터를 다시 확장하기 전에 클러스터를 최적화합니다. 이 기간은 클러스터의 스토리지 크기, 사용률, 트래픽에 따라 최소 6시간에서 24시간 이상까지 다양합니다. 이는 Auto Scaling 이벤트와 UpdateBrokerStorage 작업을 사용하는 수동 규모 조정 모두에 적용됩니다. 적절한 스토리지 규모 조정에 대한 자세한 내용은 the section called “Standard 브로커 모범 사례” 섹션을 참조하세요.

계층형 스토리지를 사용하여 브로커의 스토리지를 무제한으로 스케일 업할 수 있습니다. Standard 브로커를 위한 계층형 스토리지 섹션을 참조하세요.

주제

- Amazon MSK 클러스터를 위한 자동 규모 조정
- Standard 브로커의 수동 규모 조정

Amazon MSK 클러스터를 위한 자동 규모 조정

사용량 증가에 대응하여 클러스터의 스토리지를 자동으로 확장하려면 Amazon MSK에 대한 애플리케이션 Auto Scaling 정책을 구성할 수 있습니다. Auto Scaling 정책에서는 목표 디스크 사용률과 최대 규모 조정 용량을 설정합니다.

Amazon MSK의 자동 규모 조정 기능을 사용하기 전에 다음을 고려해야 합니다.

- > **Important**
  >
  > 스토리지 규모 조정 작업은 6시간에 한 번만 수행할 수 있습니다.

  스토리지 요구 사항에 적합한 크기의 스토리지 용량으로 시작하는 것을 권장합니다. 적절한 클러스터 크기 조정에 대한 지침은 클러스터 크기를 적절하게 조정: 클러스터당 Standard 브로커 수 섹션을 참조하세요.
- Amazon MSK는 사용량 감소에 대응하여 클러스터 스토리지를 줄이지 않습니다. Amazon MSK는 스토리지 볼륨 크기 줄이기를 지원하지 않습니다. 클러스터 스토리지의 크기를 줄여야 하는 경우 기존 클러스터를 더 작은 스토리지가 있는 클러스터로 마이그레이션해야 합니다. 클러스터 마이그레이션에 대한 자세한 내용은 MSK 클러스터로 마이그레이션 섹션을 참조하세요.
- Amazon MSK는 아시아 태평양(오사카), 아프리카(케이프타운), 아시아 태평양(말레이시아) 리전에서의 자동 규모 조정 기능을 지원하지 않습니다.
- Auto Scaling 정책을 클러스터와 연결하면 Amazon EC2 Auto Scaling은 대상 추적을 위한 Amazon CloudWatch 경보를 자동으로 생성합니다. Auto Scaling 정책을 사용하여 클러스터를 삭제해도 이 CloudWatch 경보는 계속 유지됩니다. CloudWatch 경보를 삭제하려면 클러스터를 삭제하기 전에 클러스터에서 Auto Scaling 정책을 제거해야 합니다. 대상 추적에 대해 자세히 알아보려면 Amazon EC2 Auto Scaling 사용 설명서에서 Amazon EC2 Auto Scaling에 대한 대상 추적 규모 조정 정책을 참조하세요.

주제

- Amazon MSK에 대한 오토 스케일링 정책 세부 정보
- Amazon MSK 클러스터를 위한 오토 스케일링 설정

Amazon MSK에 대한 오토 스케일링 정책 세부 정보

Auto Scaling 정책은 클러스터에 대한 다음 파라미터를 정의합니다.

- 스토리지 사용률 목표: Auto Scaling 작업을 트리거하는 데 사용하는 Amazon MSK의 스토리지 사용률 임계값입니다. 현재 스토리지 용량의 10%에서 80% 사이에서 사용률 목표를 설정할 수 있습니다. 스토리지 사용률 목표를 50%에서 60% 사이로 설정하는 것을 권장합니다.
- 최대 스토리지 용량: Amazon MSK가 브로커 스토리지에 대해 설정할 수 있는 최대 확장 한도입니다. 브로커당 최대 스토리지 용량을 최대 16TiB까지 설정할 수 있습니다. 자세한 내용은 Amazon MSK 할당량 섹션을 참조하세요.

Amazon MSK는 `Maximum Disk Utilization` 지표가 `Storage Utilization Target` 설정보다 크거나 같다는 것을 감지하면 두 숫자, 즉 10GiB 또는 현재 스토리지의 10% 중 더 큰 양만큼 스토리지 용량을 늘립니다. 예를 들어 1000GiB가 있는 경우 해당 용량은 100GiB입니다. 이 서비스는 1분마다 스토리지 사용률을 확인합니다. 추가 규모 조정 작업은 두 숫자((10GiB 또는 현재 스토리지의 10%) 중 더 큰 값만큼 스토리지를 계속 늘립니다.

Auto Scaling 작업이 발생했는지 확인하려면 [ListClusterOperations](https://docs.aws.amazon.com/msk/1.0/apireference/clusters-clusterarn-operations.html#ListClusterOperations) 작업을 사용합니다.

Amazon MSK 클러스터를 위한 오토 스케일링 설정

Amazon MSK 콘솔, Amazon MSK API 또는 CloudFormation을 사용하거나 스토리지에 대한 Auto Scaling을 구현할 수 있습니다. Application Auto Scaling을 통해 CloudFormation 지원을 이용할 수 있습니다.

> **Note**
>
> 클러스터를 만들 때는 Auto Scaling을 구현할 수 없습니다. 먼저 클러스터를 생성한 다음 해당 클러스터에 대한 Auto Scaling 정책을 생성하고 활성화해야 합니다. 그러나 Amazon MSK 서비스가 클러스터를 생성하는 동안에도 정책을 생성할 수 있습니다.

주제

- Amazon MSK AWS Management 콘솔을 사용하여 오토 스케일링 설정
- CLI를 사용하여 자동 규모 조정 설정
- API를 사용하여 Amazon MSK에 대한 오토 스케일링 설정

Amazon MSK AWS Management 콘솔을 사용하여 오토 스케일링 설정

이 프로세스는 Amazon MSK 콘솔을 사용하여 스토리지에 대한 오토 스케일링을 구현하는 방법을 설명합니다.

1. AWS Management 콘솔에 로그인하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/에서 Amazon MSK 콘솔을 엽니다.
2. 클러스터 목록에서 클러스터를 선택합니다. 클러스터에 대한 세부 정보가 나열된 페이지로 이동합니다.
3. 스토리지용 Auto Scaling 섹션에서 구성을 선택합니다.
4. Auto Scaling 정책을 생성하고 이름을 지정합니다. 스토리지 사용률 목표, 최대 스토리지 용량, 목표 지표를 지정합니다.

5. `Save changes`를 선택합니다.

새 정책을 저장하고 활성화하면 클러스터에 대해 정책이 활성화됩니다. 그런 다음 스토리지 사용률 목표에 도달하면 Amazon MSK가 클러스터의 스토리지를 확장합니다.

CLI를 사용하여 자동 규모 조정 설정

이 프로세스는 Amazon MSK CLI를 사용하여 스토리지에 대한 오토 스케일링을 구현하는 방법을 설명합니다.

1. RegisterScalableTarget 명령을 사용하여 스토리지 사용률 목표를 등록합니다.
2. PutScalingPolicy 명령을 사용하여 자동 확장 정책을 생성합니다.

API를 사용하여 Amazon MSK에 대한 오토 스케일링 설정

이 프로세스는 Amazon MSK API를 사용하여 스토리지에 대한 오토 스케일링을 구현하는 방법을 설명합니다.

1. RegisterScalableTarget API를 사용하여 스토리지 사용률 목표를 등록합니다.
2. PutScalingPolicy API를 사용하여 자동 확장 정책을 생성합니다.

Standard 브로커의 수동 규모 조정

스토리지를 늘리려면 클러스터가 `ACTIVE` 상태가 될 때까지 기다립니다. 스토리지 규모 조정에는 이벤트 사이에 최소 6시간의 휴지 기간이 있습니다. 이 작업을 통해 추가 스토리지를 즉시 사용할 수 있지만 서비스는 클러스터에서 최대 24시간 이상 소요될 수 있는 최적화를 수행합니다. 최적화 기간은 스토리지 크기에 비례합니다.

를 사용하여 브로커 스토리지 확장AWS Management 콘솔

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 브로커 스토리지를 업데이트할 MSK 클러스터를 선택합니다.
3. 스토리지 섹션에서 편집을 선택합니다.
4. 원하는 스토리지 볼륨을 지정합니다. 스토리지 용량을 늘릴 수는 있지만 줄일 수는 없습니다.
5. 변경 사항 저장을 선택합니다.

를 사용하여 브로커 스토리지 확장AWS CLI

다음 명령을 실행하여 *ClusterArn*을 클러스터 생성 후 받은 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called "클러스터 나열" 단원을 참조하십시오.

*Current-Cluster-Version*을 클러스터의 현재 버전으로 바꿉니다.

> **Important**
>
> 클러스터 버전은 단순한 정수가 아닙니다. 클러스터의 현재 버전을 찾으려면 DescribeCluster 작업 또는 describe-clusterAWS CLI 명령을 사용합니다. 버전의 예를 들면 `KTVPDKIKX0DER`입니다.

Target-Volume-in-GiB 파라미터는 각 브로커가 보유할 스토리지의 양을 나타냅니다. 모든 브로커의 스토리지만 업데이트할 수 있습니다. 스토리지를 업데이트할 개별 브로커를 지정할 수 없습니다. *Target-Volume-in-GiB*에 지정하는 값은 100GiB보다 큰 정수여야 합니다. 업데이트 작업 후 브로커당 스토리지는 16384GiB를 초과할 수 없습니다.

```
aws kafka update-broker-storage --cluster-arn ClusterArn --current-version Current-Cluster-Version --target-broker-ebs-volume-info '{"KafkaBrokerNodeId": "All",
 "VolumeSizeGB": Target-Volume-in-GiB}'
```

API를 사용하여 브로커 스토리지 스케일 업

API를 사용하여 브로커 스토리지를 업데이트하려면 UpdateBrokerStorage를 참조하십시오.

Amazon MSK 클러스터의 Standard 브로커에 대한 스토리지 처리량 관리

Amazon MSK 콘솔, CLI 및 API를 사용하여 처리량을 프로비저닝하는 방법에 대한 자세한 내용은 ??? 섹션을 참조하세요.

주제

- Amazon MSK 브로커 처리량 병목 현상 및 최대 처리량 설정
- Amazon MSK 클러스터의 스토리지 처리량 측정
- Amazon MSK 클러스터의 프로비저닝된 스토리지에 대한 구성 업데이트 값
- Amazon MSK 클러스터의 Standard 브로커에 대한 스토리지 처리량 프로비저닝

Amazon MSK 브로커 처리량 병목 현상 및 최대 처리량 설정

브로커 처리량의 병목 현상의 원인으로는 볼륨 처리량, Amazon EC2에서 Amazon EBS로의 네트워크 처리량, Amazon EC2 송신 처리량 등이 있습니다. 프로비저닝된 스토리지 처리량을 활성화하여 볼륨 처리량을 조정할 수 있습니다. 그러나 Amazon EC2에서 Amazon EBS로의 네트워크 처리량과 Amazon EC2 송신 처리량으로 인해 브로커 처리량 제한이 발생할 수 있습니다.

Amazon EC2 송신 처리량은 소비자 그룹 및 소비자 그룹당 소비자 수에 영향을 받습니다. 또한 Amazon EC2에서 Amazon EBS로의 네트워크 처리량과 Amazon EC2 송신 처리량 모두 더 규모가 큰 브로커 크기에서 더 높습니다.

볼륨 크기가 10GiB 이상인 경우에는 초당 250MiB 이상의 스토리지 처리량을 프로비저닝할 수 있습니다. 기본값은 초당 250MiB입니다. 스토리지 처리량을 프로비저닝하려면 kafka.m5.4xlarge 이상(또는 kafka.m7g.2xlarge 이상)의 브로커 크기를 선택해야 하며 다음 표와 같이 최대 처리량을 지정할 수 있습니다.

브로커 크기	최대 스토리지 처리량(MiB/초)
kafka.m5.4xlarge	593
kafka.m5.8xlarge	850
kafka.m5.12xlarge	1000
kafka.m5.16xlarge	1000
kafka.m5.24xlarge	1000
kafka.m7g.2xlarge	312.5
kafka.m7g.4xlarge	625
kafka.m7g.8xlarge	1000
kafka.m7g.12xlarge	1000
kafka.m7g.16xlarge	1000

Amazon MSK 클러스터의 스토리지 처리량 측정

`VolumeReadBytes` 및 `VolumeWriteBytes` 지표를 사용하여 클러스터의 평균 스토리지 처리량을 측정할 수 있습니다. 이 두 지표의 합계는 평균 스토리지 처리량(바이트)을 나타냅니다. 클러스터의 평균 스토리지 처리량을 얻으려면 이 두 지표를 합계로 설정하고 기간을 1분으로 설정한 후 다음 공식을 사용합니다.

```
Average storage throughput in MiB/s = (Sum(VolumeReadBytes) + Sum(VolumeWriteBytes)) /
 (60 * 1024 * 1024)
```

`VolumeReadBytes` 및 `VolumeWriteBytes` 지표에 대한 자세한 내용은 [the section called “PER_BROKER 수준 모니터링”](#) 섹션을 참조하세요.

Amazon MSK 클러스터의 프로비저닝된 스토리지에 대한 구성 업데이트 값

프로비저닝된 처리량을 활성화하기 전이나 활성화한 후에 Amazon MSK 구성을 업데이트할 수 있습니다. 그러나 `num.replica.fetchers` 구성 파라미터를 업데이트하고 프로비저닝된 처리량을 활성화하는 두 가지 작업을 모두 수행할 때까지는 원하는 처리량을 볼 수 없습니다.

기본 Amazon MSK 구성에서 `num.replica.fetchers` 값은 2입니다. `num.replica.fetchers`를 업데이트하기 위해 다음 표의 제안된 값을 사용할 수 있습니다. 이 값은 지침을 위한 것입니다. 사용 사례에 따라 이 값을 조정하는 것을 권장합니다.

브로커 크기	num.replica.fetchers
kafka.m5.4xlarge	4
kafka.m5.8xlarge	8
kafka.m5.12xlarge	14
kafka.m5.16xlarge	16
kafka.m5.24xlarge	16

업데이트된 구성은 최대 24시간 동안 적용되지 않을 수 있으며 소스 볼륨이 완전히 사용되지 않을 경우 더 오래 걸릴 수 있습니다. 그러나 마이그레이션 기간 동안 마이그레이션 볼륨 성능은 최소한 소스 스토리지 볼륨의 성능과 동일합니다. 완전히 활용되는 1TiB 볼륨을 업데이트된 구성으로 마이그레이션하는 데는 일반적으로 약 6시간이 소요됩니다.

Amazon MSK 클러스터의 Standard 브로커에 대한 스토리지 처리량 프로비저닝

Amazon MSK 브로커는 스토리지 볼륨에 데이터를 유지합니다. 스토리지 I/O는 생산자가 클러스터에 쓸 때, 브로커 간에 데이터가 복제될 때와 소비자가 메모리에 없는 데이터를 읽을 때 소비됩니다. 볼륨 스토리지 처리량은 스토리지 볼륨에 데이터를 쓰고 읽을 수 있는 속도입니다. 프로비저닝된 스토리지 처리량은 클러스터의 브로커에 대해 해당 속도를 지정할 수 있는 기능입니다.

브로커 크기가 `kafka.m5.4xlarge` 이상이고 스토리지 볼륨이 10GiB 이상인 클러스터의 경우 프로비저닝된 처리 속도를 초당 MiB 단위로 지정할 수 있습니다. 클러스터를 생성하는 중에 프로비저닝된 처리량을 지정할 수 있습니다. 또한 `ACTIVE` 상태인 클러스터에 대해 프로비저닝된 처리량을 활성화하거나 비활성화할 수 있습니다.

처리량 관리에 대한 자세한 내용은 ??? 섹션을 참조하세요.

주제

- 를 사용하여 Amazon MSK 클러스터 스토리지 처리량 프로비저닝AWS Management 콘솔
- 를 사용하여 Amazon MSK 클러스터 스토리지 처리량 프로비저닝AWS CLI
- API를 사용하여 Amazon MSK 클러스터를 생성할 때 프로비저닝 스토리지 처리량

를 사용하여 Amazon MSK 클러스터 스토리지 처리량 프로비저닝AWS Management 콘솔

이 프로세스는 AWS Management 콘솔를 사용하여 프로비저닝된 처리량이 활성화된 Amazon MSK 클러스터를 생성하는 방법의 예를 보여줍니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 클러스터 생성을 선택합니다.
3. 사용자 지정 생성을 선택합니다.
4. 클러스터의 이름을 지정합니다.
5. 스토리지 섹션에서 활성화를 선택합니다.
6. 브로커당 스토리지 처리량에 대한 값을 선택합니다.
7. VPC, 영역 및 서브넷, 보안 그룹을 선택합니다.
8. 다음을 선택합니다.
9. 보안 단계 아래에서 다음을 선택합니다.
10. 모니터링 및 태그 단계 아래에서 다음을 선택합니다.
11. 클러스터 설정을 검토한 다음 클러스터 생성을 선택합니다.

를 사용하여 Amazon MSK 클러스터 스토리지 처리량 프로비저닝AWS CLI

이 프로세스는 AWS CLI를 사용하여 프로비저닝된 처리량이 활성화된 클러스터를 생성하는 방법의 예를 보여줍니다.

1. 다음 JSON을 복사하여 파일에 붙여넣습니다. 서브넷 ID 및 보안 그룹 ID 자리 표시자를 계정의 값으로 변경합니다. 파일 이름을 `cluster-creation.json`으로 지정하고 저장합니다.

```
{
    "Provisioned": {
        "BrokerNodeGroupInfo":{
            "InstanceType":"kafka.m5.4xlarge",
            "ClientSubnets":[
                "Subnet-1-ID",
                "Subnet-2-ID"
            ],
            "SecurityGroups":[
                "Security-Group-ID"
            ],
            "StorageInfo": {
                "EbsStorageInfo": {
                    "VolumeSize": 10,
                    "ProvisionedThroughput": {
                        "Enabled": true,
                        "VolumeThroughput": 250
                    }
                }
            }
        },
        "EncryptionInfo": {
            "EncryptionInTransit": {
                "InCluster": false,
                "ClientBroker": "PLAINTEXT"
            }
        },
        "KafkaVersion":"2.8.1",
        "NumberOfBrokerNodes": 2
    },
    "ClusterName": "provisioned-throughput-example"
}
```

2. 이전 단계에서 JSON 파일을 저장한 디렉터리에서 다음 AWS CLI명령을 실행합니다.

```
aws kafka create-cluster-v2 --cli-input-json file://cluster-creation.json
```

API를 사용하여 Amazon MSK 클러스터를 생성할 때 프로비저닝 스토리지 처리량

클러스터를 생성하는 동안 프로비저닝된 스토리지 처리량을 구성하려면 CreateClusterV2를 사용합니다.

Amazon MSK Provisioned 구성

Amazon MSK는 브로커, 주제 및 메타데이터 노드에 대한 기본 구성을 제공합니다. 또한 사용자 지정 구성을 생성하고, 이를 사용해 새 MSK 클러스터를 생성하거나 기존 클러스터를 업데이트할 수 있습니다. MSK 구성은 속성 집합과 각각에 해당하는 값으로 구성됩니다. 클러스터에서 사용하는 브로커 유형에 따라 구성 기본값 세트와 수정할 수 있는 구성 세트가 다릅니다. Standard 및 Express 브로커를 구성하는 방법에 대한 자세한 내용은 아래 섹션을 참조하세요.

주제

Standard 브로커 구성

이 섹션에서는 Standard 브로커의 구성 속성을 설명합니다.

주제

사용자 지정 Amazon MSK 구성

Amazon MSK를 사용하여 다음 Apache Kafka 구성 속성을 설정하는 사용자 지정 MSK 구성을 생성할 수 있습니다. 명시적으로 설정하지 않은 속성은 the section called “기본 Amazon MSK 구성”에 있는 값을 가져옵니다. 구성 속성에 대한 자세한 내용은 Apache Kafka 구성을 참조하십시오.

이름	설명
allow.everyone.if.no.acl.found	이 속성을 `false`로 설정하려면 먼저 클러스터에 대한 Apache Kafka ACL을 정의해야 합니다. 이 속성을 `false`로 설정하고 Apache Kafka ACL을 먼저 정의하지 않으면 클러스터에 대한 액세스 권한을 잃게 됩니다. 이 경우 구성을 다시 업데이트하고 이 속성을 `true`로 설정하여 클러스터에 다시 액세스할 수 있습니다.
auto.create.topics.enable	서버의 주제 자동 생성을 활성화합니다.
compression.type	주어진 주제에 대한 최종 압축 유형입니다. 이 속성을 표준 압축 코덱(`gzip`, `snappy`, `lz4` 및 `zstd`)으로 설정할 수 있습니다. 추가로 `uncompressed` 를 허용합니다. 이 값은 압축을 하지 않은 것과 같습니다. 값을 `producer`로 설정하면 생산자가 설정한 원본 압축 코덱을 유지한다는 의미입니다.
connections.max.idle.ms	유휴 연결 제한 시간(밀리초). 서버 소켓 프로세서 스레드는 이 속성에 설정한 값보다 더 오래 유휴 상태인 연결을 닫습니다.
default.replication.factor	자동으로 생성된 주제에 대한 기본 복제 인수입니다.
delete.topic.enable	주제 삭제 작업을 활성화합니다. 이 설정을 끄면 관리자 도구를 통해 주제를 삭제할 수 없습니다.
group.initial.rebalance.delay.ms	그룹 코디네이터가 첫 번째 재조정을 수행하기 전에 더 많은 데이터 소비자가 새 그룹에 가입할 때까지 기다리는 시간입니다. 지연 시간이 길어지면 재조정 횟수는 줄어들 수 있지만 처리가 시작될 때까지 시간이 늘어납니다.
group.max.session.timeout.ms	등록된 소비자에 대한 최대 세션 제한 시간입니다. 제한 시간이 길어지면 소비자가 하트비트 사

이름	설명
	이에 메시지를 처리할 수 있는 시간이 더 길어지지만 장애를 감지하는 데 시간이 더 오래 걸립니다.
group.min.session.timeout.ms	등록된 소비자에 대한 최소 세션 제한 시간입니다. 제한 시간이 짧아지면 소비자의 하트비트가 더 자주 발생하는 대신 장애를 더 빨리 감지할 수 있습니다. 이는 브로커 리소스에 과부하를 줄 수 있습니다.
leader.imbalance.per.broker.percentage	브로커당 허용되는 리더 불균형의 비율입니다. 컨트롤러는 브로커당 이 값을 초과하는 경우 리더 밸런스를 트리거합니다. 이 값은 백분율로 지정됩니다.
log.cleaner.delete.retention.ms	Apache Kafka가 삭제된 레코드를 유지하는 시간 길이. 최소값은 0입니다.
log.cleaner.min.cleanable.ratio	이 구성 속성의 값은 0에서 1 사이일 수 있습니다. 이 값은 로그 압축기가 로그 정리를 시도하는 빈도를 결정합니다(로그 압축이 활성화된 경우). 기본적으로 Apache Kafka는 로그의 50% 이상이 압축된 경우 로그 정리를 하지 않습니다. 이 비율은 로그가 중복으로 인해 낭비하는 최대 공간을 제한합니다(50%의 경우 로그의 최대 50%가 중복일 수 있음을 의미). 비율이 높을수록 정리 횟수가 더 적고 효율적이지만 로그에서 낭비되는 공간은 더 많아집니다.
log.cleanup.policy	보존 기간이 지난 세그먼트에 대한 기본 정리 정책입니다. 유효한 정책의 쉼표로 구분된 목록입니다. 유효한 정책은 `delete` 및 `compact`입니다. 계층형 스토리지를 사용하는 클러스터의 경우 유효한 정책은 `delete`뿐입니다.

이름	설명
log.flush.interval.messages	메시지가 디스크로 플러시되기 전에 로그 파티션에 누적되는 메시지의 수입니다.
log.flush.interval.ms	주제의 메시지가 디스크에 플러시되기 전까지 메모리에 남아 있는 최대 시간(밀리초)입니다. 이 값을 설정하지 않으면 log.flush.scheduler.interval.ms의 값이 사용됩니다. 최소값은 0입니다.
log.message.timestamp.difference.max.ms	이 구성은 Kafka 3.6.0에서 더 이상 사용되지 않습니다. `log.message.timestamp.before.max.ms` 및 `log.message.timestamp.after.max.ms` 라는 두 가지 구성이 추가되었습니다. 브로커가 메시지를 수신할 때의 타임스탬프와 메시지에 지정된 타임스탬프 사이의 최대 시간 차이입니다. log.message.timestamp.type=CreateTime인 경우 타임스탬프의 차이가 이 임계값을 초과하면 메시지가 거부됩니다. 이 구성은 log.message.timestamp.type=LogAppendTime인 경우 무시됩니다.
log.message.timestamp.type	메시지의 타임스탬프가 메시지 생성 시간인지 로그 추가 시간인지 지정합니다. 허용 값은 `CreateTime` , `LogAppendTime` 입니다.
log.retention.bytes	삭제하기 전 로그의 최대 크기입니다.
log.retention.hours	로그 파일을 삭제하기 전에 보관하는 시간(3차 log.retention.ms 속성)입니다.
log.retention.minutes	로그 파일을 삭제하기 전에 보관하는 시간(분)(2차 log.retention.ms 속성)입니다. 이 값을 설정하지 않으면 log.retention.hours의 값이 사용됩니다.

이름	설명
log.retention.ms	로그 파일을 삭제하기 전에 보관하는 시간(밀리초)입니다. 설정하지 않으면 log.retention.minutes 값이 사용됩니다.
log.roll.ms	새 로그 세그먼트가 롤아웃되기 전 최대 시간(밀리초)입니다. 이 속성을 설정하지 않으면 log.roll.hours의 값이 사용됩니다. 이 속성의 가능한 최소값은 1입니다.
log.segment.bytes	단일 로그 파일의 최대 크기입니다.
max.incremental.fetch.session.cache.slots	유지되는 최대 증분 가져오기 세션 수입니다.
message.max.bytes	Kafka가 허용하는 최대 레코드 배치 크기입니다. 이 값을 늘리고 0.10.2보다 오래된 소비자가 있는 경우 이 정도 크기의 레코드 배치를 가져올 수 있도록 소비자의 가져오기 크기도 늘려야 합니다. 최신 메시지 형식 버전은 효율성을 위해 항상 메시지를 배치로 그룹화합니다. 이전 메시지 형식 버전에서는 압축되지 않은 레코드를 배치로 그룹화하지 않으며 이러한 경우 이 제한은 단일 레코드에만 적용됩니다. 이 값은 주제 수준 max.message.bytes 구성으로 주제별로 설정할 수 있습니다.

이름	설명
min.insync.replicas	생산자가 ack를 `"all"`(또는 `"-1"`)로 설정하면 min.insync.replicas의 값은 쓰기가 성공한 것으로 간주되기 위해 쓰기를 승인해야 하는 최소 복제본 수를 지정합니다. 이 최소값이 충족되지 않으면 생산자가 예외를 발생시킵니다(NotEnoughReplicas 또는 NotEnoughReplicasAfterAppend). min.insync.replicas 및 acks의 값을 사용하여 내구성 보장을 강화할 수 있습니다. 예를 들어 복제 인수가 3인 주제를 생성하고, min.insync.replicas를 2로 설정하고, `"all"`의 acks를 사용하여 생산할 수 있습니다. 이렇게 하면 대부분의 복제본이 쓰기를 수신하지 못하는 경우 생산자가 예외를 발생시킵니다.
num.io.threads	서버가 요청을 처리하는 데 사용하는 스레드 수로서, 디스크 I/O를 포함할 수 있습니다.
num.network.threads	서버가 네트워크에서 요청을 수신하고 응답을 전송하는 데 사용하는 스레드 수입니다.
num.partitions	주제별 기본 로그 파티션 수입니다.
num.recovery.threads.per.data.dir	시작 시 로그를 복구하고 종료 시 로그를 플러시하는 데 사용할 데이터 디렉터리당 스레드 수입니다.
num.replica.fetchers	소스 브로커의 메시지를 복제하는 데 사용하는 가져오기 스레드 수입니다. 이 값을 늘리면 팔로어 브로커의 I/O 병렬 처리 정도를 늘릴 수 있습니다.

이름	설명
offsets.retention.minutes	소비자 그룹이 모든 소비자를 잃은 후(즉, 비어 있음) 해당 오프셋은 폐기되기 전까지 이 보존 기간 동안 유지됩니다. 독립 실행형 소비자(즉, 수동 할당을 사용하는 소비자)의 경우 오프셋은 마지막 커밋 시간에 이 보존 기간을 더한 후 만료됩니다.
offsets.topic.replication.factor	오프셋 주제에 대한 복제 인수입니다. 가용성을 보장하려면 이 값을 높게 설정합니다. 클러스터 크기가 이 복제 인수 요구 사항을 충족할 때까지 내부 주제 생성에 실패합니다.
replica.fetch.max.bytes	각 파티션에 대해서 가져오려고 하는 메시지 바이트 수입니다. 이 값은 절대적인 최대값이 아닙니다. 가져오기의 첫 번째 비어 있지 않은 파티션에 있는 첫 번째 레코드 배치가 이 값보다 크면 진행을 보장하기 위해 레코드 배치가 반환됩니다. message.max.bytes(브로커 구성) 또는 max.message.bytes(주제 구성)는 브로커가 허용하는 최대 레코드 배치 크기를 정의합니다.
replica.fetch.response.max.bytes	전체 가져오기 응답에 대해 예상되는 최대 바이트 수입니다. 레코드는 배치로 가져오며 가져오기의 첫 번째 비어 있지 않은 파티션의 첫 번째 레코드 배치가 이 값보다 크면 진행을 보장하기 위해 배치가 반환됩니다. 이 값은 절대적인 최대값이 아닙니다. message.max.bytes(브로커 구성) 또는 max.message.bytes(주제 구성) 속성은 브로커가 허용하는 최대 레코드 배치 크기를 지정합니다.

이름	설명
replica.lag.time.max.ms	팔로워가 가져오기 요청을 보내지 않았거나 적어도 이 시간(밀리초) 동안 리더의 로그 끝 오프셋까지 소모하지 않은 경우 리더는 ISR에서 팔로어를 제거합니다. 최소값: 10000 MaxValue = 30000
replica.selector.class	ReplicaSelector를 구현하는 정규화된 클래스 이름입니다. 브로커는 이 값을 사용하여 선호하는 읽기 복제본을 찾습니다. Apache Kafka 버전 2.4.1 이상을 사용하며 소비자가 가장 가까운 복제본에서 가져오도록 허용하려면 이 속성을 `org.apache.kafka.common.replica.RackAwareReplicaSelector` 로 설정하세요. 자세한 내용은 the section called “Apache Kafka 버전 2.4.1(대신 2.4.1.1 사용)” 단원을 참조하십시오.
replica.socket.receive.buffer.bytes	네트워크 요청에 대한 소켓 수신 버퍼입니다.
socket.receive.buffer.bytes	소켓 서버 소켓의 SO_RCVBUF 버퍼입니다. 이 속성에 설정할 수 있는 최소값은 -1입니다. 값이 -1이면 Amazon MSK는 OS 기본값을 사용합니다.
socket.request.max.bytes	소켓 요청의 최대 바이트 수입니다.
socket.send.buffer.bytes	소켓 서버 소켓의 SO_SNDBUF 버퍼입니다. 이 속성에 설정할 수 있는 최소값은 -1입니다. 값이 -1이면 Amazon MSK는 OS 기본값을 사용합니다.

이름	설명
transaction.max.timeout.ms	트랜잭션의 최대 제한 시간입니다. 클라이언트가 요청한 트랜잭션 시간이 이 값을 초과하면 브로커는 InitProducerIdRequest에 오류를 반환합니다. 이렇게 하면 클라이언트가 너무 긴 시간 제한을 방지합니다. 이로 인해 트랜잭션에 포함된 항목을 읽는 소비자가 중단될 수 있습니다.
transaction.state.log.min.isr	트랜잭션 주제에 대한 min.insync.replicas 구성을 재정의했습니다.
transaction.state.log.replication.factor	트랜잭션 주제에 대한 복제 인수입니다. 가용성을 높이려면 이 속성을 더 높은 값으로 설정합니다. 클러스터 크기가 이 복제 인수 요구 사항을 충족할 때까지 내부 주제 생성에 실패합니다.
transactional.id.expiration.ms	트랜잭션 코디네이터가 트랜잭션 ID를 만료하기 전에 현재 트랜잭션에 대한 트랜잭션 상태 업데이트를 수신하기 위해 기다리는 시간(밀리초). 이 설정은 지정된 생산자 ID로 마지막으로 쓴 후 이 시간이 경과하면 생산자 ID가 만료되므로 생산자 ID 만료에도 영향을 줍니다. 주제에 대한 보존 설정으로 인해 생산자 ID의 마지막 쓰기가 삭제되면 생산자 ID가 더 빨리 만료될 수 있습니다. 이 속성의 최소값은 1밀리초입니다.
unclean.leader.election.enable	데이터 손실이 발생할 수 있지만 ISR 세트에 없는 복제본을 최후의 수단으로 리더 역할을 해야 하는지 여부를 나타냅니다.

이름	설명
zookeeper.connection.timeout.ms	ZooKeeper 모드 클러스터. 클라이언트가 ZooKeeper에 대한 연결을 설정하기 위해 대기하는 최대 시간입니다. 이 값을 설정하지 않으면 zookeeper.session.timeout.ms의 값이 사용됩니다. 최소값 = 6000 최대값(포함) = 18000 클러스터 가동 중지 시간을 방지하려면 T3.small에서 이 값을 10,000으로 설정하는 것이 좋습니다.
zookeeper.session.timeout.ms	ZooKeeper 모드 클러스터. 밀리초 단위의 Apache ZooKeeper 세션 제한 시간입니다. 최소값 = 6000 최대값(포함) = 18000

사용자 지정 MSK 구성을 만들거나, 모든 구성을 나열하거나, 설명하는 방법을 알아보려면 the section called “브로커 구성 작업” 단원을 참조하십시오. 사용자 지정 MSK 구성으로 MSK 클러스터를 만들거나 새 사용자 지정 구성으로 클러스터를 업데이트하려면 the section called “주요 기능 및 개념” 섹션을 참조하세요.

기존 MSK 클러스터를 사용자 지정 MSK 구성으로 업데이트할 때 Amazon MSK는 필요한 경우 롤링을 다시 시작하고 모범 사례를 사용하여 고객 가동 중지를 최소화합니다. 예를 들어 Amazon MSK는 각 브로커를 다시 시작한 후 다음 브로커로 이동하기 전에 브로커가 구성 업데이트 중에 놓쳤을 수 있는 데이터를 따라잡을 수 있도록 시도합니다.

Amazon MSK 구성

Amazon MSK에서 제공하는 구성 속성 외에도 브로커를 다시 시작할 필요가 없는 클러스터 수준 및 브로커 수준 구성 속성을 동적으로 설정할 수 있습니다. 일부 구성 속성을 동적으로 설정할 수 있습니다. Apache Kafka 설명서의 브로커 구성 아래의 표에서 읽기 전용으로 표시되지 않은 속성입니다. 동적 구성과 예제 명령에 대한 자세한 내용은 Apache Kafka 설명서의 브로커 구성 업데이트를 참조하세요.

> **Note**
>
> `advertised.listeners` 속성은 설정할 수 있지만 `listeners` 속성은 설정할 수 없습니다.

주제 수준 Amazon MSK 구성

Apache Kafka 명령을 사용하여 새 주제와 기존 주제에 대한 주제 수준 구성 속성을 설정하거나 수정할 수 있습니다. 주제 수준 구성 속성과 설정 방법 예제에 대한 자세한 내용은 Apache Kafka 설명서의 [주제 수준 구성]을 참조하세요.

기본 Amazon MSK 구성

MSK 클러스터를 생성하고 사용자 지정 MSK 구성을 지정하지 않으면 Amazon MSK는 다음 표에 표시된 값으로 기본 구성을 생성하여 사용합니다. 이 표에 없는 속성의 경우 Amazon MSK는 사용 중인 Apache Kafka 버전과 관련된 기본값을 사용합니다. 이러한 기본값 목록은 [Apache Kafka 구성]을 참조하십시오.

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
allow.everyone.if.no.acl.found	특정 리소스와 일치하는 리소스 패턴이 없으면 리소스에 연결된 ACL이 없는 것입니다. 이 경우 속성을 `true`로 설정하면 수퍼유저뿐만 아니라 모든 사용자가 리소스에 액세스할 수 있습니다.	`true`	`true`
auto.create.topics.enable	서버에서 주제의 자동 생성을 활성화합니다.	`false`	`false`
auto.leader.rebalance.enable	자동 리더 밸런싱을 활성화합니다. 필요한 경우 백그라운드 스레드는 정기적으로 리더 밸	`true`	`true`

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
	런스를 확인하고 시작합니다.		
default.replication.factor	자동으로 생성된 주제에 대한 기본 복제 인수입니다.	가용 영역 3개에 있는 클러스터의 경우 3, 가용 영역 2개에 있는 클러스터의 경우 2입니다.	가용 영역 3개에 있는 클러스터의 경우 3, 가용 영역 2개에 있는 클러스터의 경우 2입니다.

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
local.retention.bytes	이전 세그먼트를 삭제하기 전 파티션에 대한 로컬 로그 세그먼트의 최대 크기입니다. 이 값을 설정하지 않으면 log.retention.bytes의 값이 사용됩니다. 유효 값은 항상 log.retention.bytes 값보다 작거나 같아야 합니다. 기본값인 -2는 로컬 보존에 제한이 없음을 나타냅니다. 이는 retention.ms/bytes 설정인 -1에 해당합니다. local.retention.ms 및 local.retention.bytes 속성은 로그 세그먼트가 로컬 스토리지에 얼마나 오래 남아 있어야 하는지 결정하는 데 사용되므로 log.retention과 유사합니다. 기존 log.retention.* 구성은 주제 파티션에 대한 보존 구성입니다. 여기에는 로컬 스토리지와 원격 스토리지가 모두 포함되어 있습니다. 유효한 값: [-2; +Inf]의 정수	무제한의 경우 -2	무제한의 경우 -2

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
local.retention.ms	삭제하기 전에 로컬 로그 세그먼트를 보존할 시간(밀리초)입니다. 이 값을 설정하지 않으면 Amazon MSK는 log.retention.ms의 값이 사용됩니다. 유효 값은 항상 log.retention.bytes 값보다 작거나 같아야 합니다. 기본값인 -2는 로컬 보존에 제한이 없음을 나타냅니다. 이는 retention.ms/bytes 설정인 -1에 해당합니다. local.retention.ms 및 local.retention.bytes 값은 log.retention과 유사합니다. MSK는 이 구성을 사용하여 로그 세그먼트가 로컬 스토리지에 얼마나 오래 남아 있어야 하는지 결정합니다. 기존 log.retention.* 구성은 주제 파티션에 대한 보존 구성입니다. 여기에는 로컬 스토리지와 원격 스토리지가 모두 포함되어 있습니다. 유효 값은 0보다 큰 정수입니다.	무제한의 경우 -2	무제한의 경우 -2

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
log.message.timestamp.difference.max.ms	이 구성은 Kafka 3.6.0에서 더 이상 사용되지 않습니다. `log.message.timestamp.before.max.ms` 및 `log.message.timestamp.after.max.ms` 라는 두 가지 구성이 추가되었습니다. 브로커가 메시지를 수신한 시점의 타임스탬프와 메시지에 지정된 타임스탬프 사이에 허용되는 최대 차이입니다. log.message.timestamp.type=CreateTime인 경우 타임스탬프의 차이가 이 임계값을 초과하면 메시지가 거부됩니다. 이 구성은 log.message.timestamp.type=LogAppendTime인 경우 무시됩니다. 허용되는 최대 타임스탬프 차이는 불필요하게 빈번한 로그 롤링을 방지하기 위해 log.retention.ms보다 크지 않아야 합니다.	9223372036854775807	86400000(Kafka 2.8.2.tiered 및 Kafka 3.7.x tiered의 경우).

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
log.segment.bytes	단일 로그 파일의 최대 크기입니다.	1073741824	134217728

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
min.insync.replicas	생산자가 acks(생산자가 Kafka 브로커로부터 받는 승인) 값을 `"all"`(또는 `"-1"`)로 설정하는 경우 min.insync.replicas의 값은 쓰기가 성공한 것으로 간주되기 위해 쓰기를 승인해야 하는 최소 복제본 수를 지정합니다. 이 값이 이 최소값을 충족하지 못하면 생산자가 예외를 발생시킵니다(NotEnoughReplicas 또는 NotEnoughReplicasAfterAppend 중 하나). min.insync.replicas와 acks의 값을 함께 사용하면 내구성을 더욱 강력하게 보장할 수 있습니다. 예를 들어 복제 인수가 3인 주제를 생성하고, min.insync.replicas를 2로 설정하고, `"all"`의 acks를 사용하여 생산할 수 있습니다. 이렇게 하면 대부분의 복제본이 쓰기를 수신하지 못하는 경우 생산자가 예외를 발생시킵니다.	가용 영역 3개에 있는 클러스터의 경우 2, 가용 영역 2개에 있는 클러스터의 경우 1입니다.	가용 영역 3개에 있는 클러스터의 경우 2, 가용 영역 2개에 있는 클러스터의 경우 1입니다.

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
num.io.threads	서버가 요청을 생성하는 데 사용하는 스레드 수로, 디스크 I/O가 포함될 수 있습니다.	8	max(8, vCPUs)이며 여기서 vCPU는 브로커의 인스턴스 크기에 따라 달라집니다.
num.network.threads	서버가 네트워크에서 요청을 수신하고 네트워크에 응답을 전송하는 데 사용하는 스레드 수입니다.	5	max(5, vCPUs / 2)이며 여기서 vCPU는 브로커의 인스턴스 크기에 따라 달라집니다.
num.partitions	주제별 기본 로그 파티션 수입니다.	1	1
num.replica.fetchers	소스 브로커에서 메시지를 복제하는 데 사용되는 가져오기 스레드 수로, 이 값을 늘리면 팔로어 브로커의 I/O 병렬 처리 정도를 높일 수 있습니다.	2	max(2, vCPUs / 4)이며 여기서 vCPU는 브로커의 인스턴스 크기에 따라 달라집니다.
remote.log.msk.disable.policy	remote.storage.enable과 함께 사용하여 계층형 스토리지를 비활성화합니다. remote.storage.enable을 false로 설정할 때 계층형 스토리지의 데이터가 삭제됨을 나타내려면 이 정책을 삭제로 설정합니다.	해당 사항 없음	없음

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
remote.log.reader.threads	원격 스토리지에서 데이터를 가져오는 작업을 예약하는 데 사용되는 원격 로그 리더 스레드 풀 크기입니다.	해당 사항 없음	max(10, vCPUs * 0.67)이며 여기서 vCPU는 브로커의 인스턴스 크기에 따라 달라집니다.
remote.storage.enable	true로 설정하면 주제에 대한 계층형 (원격) 스토리지를 활성화합니다. false로 설정되어 있고 remote.log.msk.disable.policy가 삭제로 설정되어 있는 경우 주제 수준 계층형 스토리지를 비활성화합니다. 계층형 스토리지를 비활성화하는 경우 원격 스토리지에서 데이터가 삭제됩니다. 주제에 대한 계층형 스토리지를 비활성화하면 이를 다시 활성화할 수 없습니다.	false	false
replica.lag.time.max.ms	팔로워가 가져오기 요청을 보내지 않았거나 적어도 이 시간(밀리초) 동안 리더의 로그 끝 오프셋까지 소모하지 않은 경우 리더는 ISR에서 팔로어를 제거합니다.	30000	30000

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
retention.ms	필수 필드입니다. 최소 시간은 3일입니다. 이 설정은 필수이므로 기본값이 없습니다. Amazon MSK는 retention.ms 값을 local.retention.ms와 함께 사용하여 데이터가 로컬 스토리지에서 계층형 스토리지로 이동하는 시점을 결정합니다. local.retention.ms 값은 데이터를 로컬에서 계층형 스토리지로 이동할 시기를 지정합니다. retention.ms 값은 계층형 스토리지에서 데이터를 제거(즉, 클러스터에서 제거)할 시기를 지정합니다. 유효한 값: [-1; +Inf]의 정수	최소 259,200,000밀리초(3일)이며, 무한 보존의 경우 -1입니다.	최소 259,200,000밀리초(3일)이며, 무한 보존의 경우 -1입니다.
socket.receive.buffer.bytes	소켓 서버 소켓의 SO_RCVBUF 버퍼입니다. 값이 -1이면 OS 기본값이 사용됩니다.	102400	102400
socket.request.max.bytes	소켓 요청의 최대 바이트 수입니다.	104857600	104857600

이름	설명	비계층형 스토리지 클러스터의 기본값	계층형 스토리지 활성화 클러스터의 기본값
socket.send.buffer.bytes	소켓 서버 소켓의 SO_SNDBUF 버퍼입니다. 값이 -1이면 OS 기본값이 사용됩니다.	102400	102400
unclean.leader.election.enable	데이터 손실이 발생할 수 있지만 ISR 세트에 없는 복제본을 최후의 수단으로 리더 역할을 하도록 할 것인지 여부를 나타냅니다.	true	false
zookeeper.session.timeout.ms	밀리초 단위의 Apache ZooKeeper 세션 제한 시간입니다.	18000	18000
zookeeper.set.acl	보안 ACL을 사용하도록 설정된 클라이언트입니다.	false	false

사용자 지정 구성 값을 지정하는 방법에 대한 자세한 내용은 the section called “사용자 지정 Amazon MSK 구성” 섹션을 참조하세요.

Amazon MSK 계층형 스토리지 주제 수준의 구성에 대한 지침

다음은 주제 수준에서 계층형 스토리지를 구성할 때의 기본 설정과 제한 사항입니다.

- Amazon MSK는 계층형 스토리지가 활성화된 주제에 대해서는 더 작은 로그 세그먼트 크기를 지원하지 않습니다. 세그먼트를 생성하려면 최소 로그 세그먼트 크기가 48MB이거나 최소 세그먼트 롤링 시간이 10분이어야 합니다. 이러한 값은 segment.bytes와 segment.ms 속성에 매핑됩니다.
- local.retention.ms/bytes의 값은tention.ms/bytes와 같거나 초과할 수 없습니다. 이는 계층형 스토리지 보존 설정입니다.
- local.retention.ms/bytes의 기본값은 -2입니다. 즉, retention.ms 값이 local.retention.ms/bytes에 사용됩니다. 이 경우 데이터는 로컬 스토리지와 계층형 스토리지(각각 복사본 하나씩)에 모두 남아 있으

며 함께 만료됩니다. 이 옵션의 경우 로컬 데이터의 복사본이 원격 스토리지에 유지됩니다. 이 경우 소비 트래픽에서 읽은 데이터는 로컬 스토리지에서 나옵니다.

- retention.ms의 기본값은 7일입니다. retention.bytes의 기본 크기 제한은 없습니다.
- retention.ms/bytes의 최소값은 -1입니다. 즉, 무제한 보존을 의미합니다.
- local.retention.ms/bytes의 최소값은 -2입니다. 즉, 로컬 스토리지에 대한 무제한 보존이 가능합니다. retention.ms/bytes 설정과 -1로 일치합니다.
- 계층형 스토리지가 활성화된 주제의 경우 주제 수준의 구성인 retention.ms는 필수입니다. 최소 retention.ms는 3일입니다.

계층형 스토리지 제약 조건에 대한 자세한 내용은 Amazon MSK 클러스터에 대한 계층형 스토리지 제약 조건 및 제한 사항 섹션을 참조하세요.

Express 브로커 구성

Apache Kafka에는 MSK Provisioned 클러스터의 성능을 조정하는 데 사용할 수 있는 수백 개의 브로커 구성이 있습니다. 잘못된 값이나 최적화되지 않은 값을 설정하면 클러스터 신뢰성과 성능에 영향을 미칠 수 있습니다. Express 브로커는 중요한 구성에 최적의 값을 설정하고 일반적인 잘못된 구성으로부터 보호하여 MSK Provisioned 클러스터의 가용성과 내구성을 개선합니다. 읽기 및 쓰기 액세스에 따른 구성에는 읽기/쓰기(편집 가능), 읽기 전용 및 비읽기/쓰기 구성의 세 가지 범주가 있습니다. 일부 구성은 클러스터가 실행 중인 Apache Kafka 버전에 대한 Apache Kafka의 기본값을 계속 사용합니다. 이를 Apache Kafka 기본값으로 표시합니다.

주제

- 사용자 지정 MSK Express 브로커 구성(읽기/쓰기 액세스)
- Express 브로커 읽기 전용 구성

사용자 지정 MSK Express 브로커 구성(읽기/쓰기 액세스)

Amazon MSK의 업데이트 구성 기능을 사용하거나 Apache Kafka의 AlterConfig API를 사용하여 읽기/쓰기 브로커 구성을 업데이트할 수 있습니다. Apache Kafka 브로커 구성은 정적 또는 동적입니다. 정적 구성을 적용하려면 브로커를 다시 시작해야 하지만 동적 구성을 적용하려면 브로커를 다시 시작할 필요가 없습니다. 구성 속성 및 업데이트 모드에 대한 자세한 내용은 브로커 구성 업데이트를 참조하세요.

주제

- MSK Express 브로커의 정적 구성

- Express 브로커의 동적 구성
- Express 브로커의 주제 수준 구성

MSK Express 브로커의 정적 구성

Amazon MSK를 사용하여 다음 정적 속성을 설정하는 사용자 지정 MSK 구성을 생성할 수 있습니다. Amazon MSK는 설정하지 않은 다른 모든 속성을 설정하고 관리합니다. MSK 콘솔에서 또는 구성 명령을 사용하여 정적 구성 파일을 생성하고 업데이트할 수 있습니다.

속성	설명	기본 값
allow.everyone.if.no.acl.found	이 속성을 false로 설정하려면 먼저 클러스터에 대한 Apache Kafka ACL을 정의해야 합니다. 이 속성을 false로 설정하고 Apache Kafka ACL을 먼저 정의하지 않으면 클러스터에 대한 액세스 권한을 잃게 됩니다. 이 경우 구성을 다시 업데이트하고 이 속성을 true로 설정하여 클러스터에 다시 액세스할 수 있습니다.	true
auto.create.topics.enable	서버에서 주제의 자동 생성을 활성화합니다.	false
compression.type	주어진 주제에 대한 최종 압축 유형을 지정합니다. 이 구성은 표준 압축 코덱 gzip, snappy, lz4, zstd를 허용합니다. 이 구성은 압축하지 않은 유형과 동일한 `uncompressed` , 생산자가 설정한 원래 압축 코덱을 유지한다는 의미인 `producer`를 허용합니다.	Apache Kafka 기본값

속성	설명	기본 값
connections.max.idle.ms	유휴 연결 제한 시간(밀리초). 서버 소켓 프로세서 스레드는 이 속성에 설정한 값보다 더 오래 유휴 상태인 연결을 닫습니다.	Apache Kafka 기본값
delete.topic.enable	주제 삭제 작업을 활성화합니다. 이 설정을 끄면 관리자 도구를 통해 주제를 삭제할 수 없습니다.	Apache Kafka 기본값
group.initial.rebalance.delay.ms	그룹 코디네이터가 첫 번째 재조정을 수행하기 전에 더 많은 데이터 소비자가 새 그룹에 가입할 때까지 기다리는 시간입니다. 지연 시간이 길어지면 재조정 횟수는 줄어들 수 있지만 처리가 시작될 때까지 시간이 늘어납니다.	Apache Kafka 기본값
group.max.session.timeout.ms	등록된 소비자에 대한 최대 세션 제한 시간입니다. 제한 시간이 길어지면 소비자가 하트비트 사이에 메시지를 처리할 수 있는 시간이 더 길어지지만 장애를 감지하는 데 시간이 더 오래 걸립니다.	Apache Kafka 기본값
leader.imbalance.per.broker.percentage	브로커당 허용되는 리더 불균형의 비율입니다. 컨트롤러는 브로커당 이 값을 초과하는 경우 리더 밸런스를 트리거합니다. 이 값은 백분율로 지정됩니다.	Apache Kafka 기본값

속성	설명	기본 값
log.cleanup.policy	보존 기간이 지난 세그먼트에 대한 기본 정리 정책입니다. 유효한 정책의 쉼표로 구분된 목록입니다. 유효한 정책은 `delete` 및 `compact`입니다. 계층형 스토리지 지원 클러스터를 사용하는 경우 유효한 정책은 `delete`뿐입니다.	Apache Kafka 기본값
log.message.timestamp.after.max.ms	메시지 타임스탬프와 브로커의 타임스탬프 간의 허용 가능한 타임스탬프 차이입니다. 메시지 타임스탬프는 브로커의 타임스탬프보다 크거나 같을 수 있으며, 허용되는 최대 차이는 이 구성에 설정된 값에 따라 결정됩니다. `log.message.timestamp.type=CreateTime` 인 경우 타임스탬프의 차이가 이 지정된 임계값을 초과하면 메시지가 거부됩니다. 이 구성은 `log.message.timestamp.type=LogAppendTime` 인 경우 무시됩니다.	86400000(24*60*60*1000ms, 즉 1일)

속성	설명	기본 값
log.message.timestamp.before.max.ms	브로커의 타임스탬프와 메시지 타임스탬프 간의 허용 가능한 타임스탬프 차이입니다. 메시지 타임스탬프는 브로커의 타임스탬프보다 작거나 같을 수 있으며, 허용되는 최대 차이는 이 구성에 설정된 값에 따라 결정됩니다. `log.message.timestamp.type=CreateTime` 인 경우 타임스탬프의 차이가 이 지정된 임계값을 초과하면 메시지가 거부됩니다. 이 구성은 `log.message.timestamp.type=LogAppendTime` 인 경우 무시됩니다.	86400000(24*60*60*1000ms, 즉 1일)
log.message.timestamp.type	메시지의 타임스탬프가 메시지 생성 시간인지 로그 추가 시간인지 지정합니다. 허용 값은 `CreateTime` , `LogAppendTime` 입니다.	Apache Kafka 기본값
log.retention.bytes	삭제하기 전 로그의 최대 크기입니다.	Apache Kafka 기본값
log.retention.ms	로그 파일을 삭제하기 전에 유지할 밀리초 수입니다.	Apache Kafka 기본값

속성	설명	기본 값
max.connections.per.ip	각 IP 주소에서 허용되는 최대 연결 수입니다. `max.connections.per.ip.overrides` 속성을 사용하여 재정의가 구성된 경우 이 값을 `0`으로 설정할 수 있습니다. 한도에 도달하면 IP 주소의 새 연결이 삭제됩니다.	Apache Kafka 기본값
max.incremental.fetch.session.cache.slots	유지되는 최대 증분 가져오기 세션 수입니다.	Apache Kafka 기본값
message.max.bytes	Kafka가 허용하는 최대 레코드 배치 크기입니다. 이 값을 늘리고 0.10.2보다 오래된 소비자가 있는 경우 이 정도 크기의 레코드 배치를 가져올 수 있도록 소비자의 가져오기 크기도 늘려야 합니다. 최신 메시지 형식 버전은 효율성을 위해 항상 메시지를 배치로 그룹화합니다. 이전 메시지 형식 버전에서는 압축되지 않은 레코드를 배치로 그룹화하지 않으며 이러한 경우 이 제한은 단일 레코드에만 적용됩니다. 이 값은 주제 수준 `max.message.bytes` 구성으로 주제별로 설정할 수 있습니다.	Apache Kafka 기본값
num.partitions	주제별 기본 파티션 수입니다.	1

속성	설명	기본 값
offsets.retention.minutes	소비자 그룹이 모든 소비자를 잃은 후(즉, 비어 있음) 해당 오프셋은 폐기되기 전까지 이 보존 기간 동안 유지됩니다. 독립 실행형 소비자(즉, 수동 할당을 사용하는 소비자)의 경우 오프셋은 마지막 커밋 시간에 이 보존 기간을 더한 후 만료됩니다.	Apache Kafka 기본값
replica.fetch.max.bytes	각 파티션에 대해서 가져오려고 하는 메시지 바이트 수입니다. 이 값은 절대적인 최대값이 아닙니다. 가져오기의 첫 번째 비어 있지 않은 파티션에 있는 첫 번째 레코드 배치가 이 값보다 크면 진행을 보장하기 위해 레코드 배치가 반환됩니다. message.max.bytes(브로커 구성) 또는 max.message.bytes(주제 구성)는 브로커가 허용하는 최대 레코드 배치 크기를 정의합니다.	Apache Kafka 기본값
replica.selector.class	ReplicaSelector를 구현하는 정규화된 클래스 이름입니다. 브로커는 이 값을 사용하여 선호하는 읽기 복제본을 찾습니다. 소비자가 가장 가까운 복제본에서 가져오도록 허용하려면 이 속성을 `org.apache.kafka.common.replica.RackAwareReplicaSelector` 로 설정합니다.	Apache Kafka 기본값

속성	설명	기본 값
socket.receive.buffer.bytes	소켓 서버 소켓의 SO_RCVBUF 버퍼입니다. 값이 -1이면 OS 기본값이 사용됩니다.	102400
socket.request.max.bytes	소켓 요청의 최대 바이트 수입니다.	104857600
socket.send.buffer.bytes	소켓 서버 소켓의 SO_SNDBUF 버퍼입니다. 값이 -1이면 OS 기본값이 사용됩니다.	102400
transaction.max.timeout.ms	트랜잭션의 최대 제한 시간입니다. 클라이언트가 요청한 트랜잭션 시간이 이 값을 초과하면 브로커는 InitProducerIdRequest에 오류를 반환합니다. 이렇게 하면 클라이언트가 너무 긴 시간 제한을 방지합니다. 이로 인해 트랜잭션에 포함된 항목을 읽는 소비자가 중단될 수 있습니다.	Apache Kafka 기본값

속성	설명	기본 값
transactional.id.expiration.ms	트랜잭션 코디네이터가 트랜잭션 ID를 만료하기 전에 현재 트랜잭션에 대한 트랜잭션 상태 업데이트를 수신하기 위해 기다리는 시간(밀리초). 이 설정은 지정된 생산자 ID로 마지막으로 쓴 후 이 시간이 경과하면 생산자 ID가 만료되므로 생산자 ID 만료에도 영향을 줍니다. 주제에 대한 보존 설정으로 인해 생산자 ID의 마지막 쓰기가 삭제되면 생산자 ID가 더 빨리 만료될 수 있습니다. 이 속성의 최소값은 1밀리초입니다.	Apache Kafka 기본값

Express 브로커의 동적 구성

Apache Kafka AlterConfig API 또는 Kafka-configs.sh 도구를 사용하여 다음 동적 구성을 편집할 수 있습니다. Amazon MSK는 설정하지 않은 다른 모든 속성을 설정하고 관리합니다. 브로커를 다시 시작할 필요가 없는 클러스터 수준 및 브로커 수준 구성 속성을 동적으로 설정할 수 있습니다.

속성	설명	기본값
advertised.listeners	`listeners` 구성 속성과 다른 경우 클라이언트가 사용할 수 있도록 게시할 리스너입니다. IaaS 환경에서는 브로커가 바인딩되는 인터페이스와 이 값이 달라	null

속성	설명	기본값
	야 할 수 있습니다. 설정하지 않으면 리스너 값이 사용됩니다. 리스너와 달리 0.0.0.0 메타 주소를 알리는 것은 유효하지 않습니다. 또한 `listeners` 와 달리 이 속성에는 중복 포트가 있을 수 있으므로 한 리스너가 다른 리스너의 주소를 알리도록 구성할 수 있습니다. 이는 외부 로드 밸런서가 사용되는 경우에 유용할 수 있습니다. 이 속성은 브로커당 수준에서 설정됩니다.	

속성	설명	기본값
compression.type	주어진 주제에 대한 최종 압축 유형입니다. 이 속성을 표준 압축 코덱(gzip, snappy, lz4 및 zstd)으로 설정할 수 있습니다. 추가로 uncompressed 를 허용합니다. 이 값은 압축을 하지 않은 것과 같습니다. 값을 producer로 설정하면 생산자가 설정한 원본 압축 코덱을 유지한다는 의미입니다.	Apache Kafka 기본값

속성	설명	기본값
log.cleaner.delete.retention.ms	로그 압축 주제에 대한 삭제 톰스톤 마커를 유지하는 시간입니다. 또한 이 설정은 소비자가 오프셋 0에서 시작하는 경우 최종 단계의 유효한 스냅샷을 얻기 위해 읽기를 완료해야 하는 시간에 대한 경계를 제공합니다. 그렇지 않으면 스캔을 완료하기 전에 삭제 톰스톤이 수집될 수 있습니다.	86400000(24 * 60 * 60 * 1000ms, 즉 1일), Apache Kafka 기본값
log.cleaner.min.compaction.lag.ms	로그에서 메시지가 압축되지 않은 상태로 유지되는 최소 시간입니다. 이 설정은 압축 중인 로그에만 적용됩니다.	0, Apache Kafka 기본값

속성	설명	기본값
log.cleaner.max.compaction.lag.ms	메시지가 로그의 압축에 적합하지 않은 상태로 유지되는 최대 시간입니다. 이 설정은 압축 중인 로그에만 적용됩니다. 이 구성은 [7일, Long.Max] 범위로 제한됩니다.	9223372036854775807, Apache Kafka 기본값
log.cleanup.policy	보존 기간이 지난 세그먼트에 대한 기본 정리 정책입니다. 유효한 정책의 쉼표로 구분된 목록입니다. 유효한 정책은 `delete` 및 `compact`입니다. 계층형 스토리지 지원 클러스터를 사용하는 경우 유효한 정책은 `delete`뿐입니다.	Apache Kafka 기본값

속성	설명	기본값
log.message.timestamp.after.max.ms	메시지 타임스탬프와 브로커의 타임스탬프 간의 허용 가능한 타임스탬프 차이입니다. 메시지 타임스탬프는 브로커의 타임스탬프보다 크거나 같을 수 있으며, 허용되는 최대 차이는 이 구성에 설정된 값에 따라 결정됩니다. `log.message.timestamp.type=CreateTime` 인 경우 타임스탬프의 차이가 이 지정된 임계값을 초과하면 메시지가 거부됩니다. 이 구성은 `log.message.timestamp.type=LogAppendTime` 인 경우 무시됩니다.	86400000(24*60*60*1000ms, 즉 1일)

속성	설명	기본값
log.message.timestamp.before.max.ms	브로커의 타임스탬프와 메시지 타임스탬프 간의 허용 가능한 타임스탬프 차이입니다. 메시지 타임스탬프는 브로커의 타임스탬프보다 작거나 같을 수 있으며, 허용되는 최대 차이는 이 구성에 설정된 값에 따라 결정됩니다. `log.message.timestamp.type=CreateTime` 인 경우 타임스탬프의 차이가 이 지정된 임계값을 초과하면 메시지가 거부됩니다. 이 구성은 `log.message.timestamp.type=LogAppendTime` 인 경우 무시됩니다.	86400000(24*60*60*1000ms, 즉 1일)

속성	설명	기본값
log.message.timestamp.type	메시지의 타임스탬프가 메시지 생성 시간인지 로그 추가 시간인지 지정합니다. 허용 값은 `CreateTime` , `LogAppendTime` 입니다.	Apache Kafka 기본값
log.retention.bytes	삭제하기 전 로그의 최대 크기입니다.	Apache Kafka 기본값
log.retention.ms	로그 파일을 삭제하기 전에 유지할 밀리초 수입니다.	Apache Kafka 기본값
max.connection.creation.rate	언제든지 브로커에 허용되는 최대 연결 생성 속도입니다.	Apache Kafka 기본값
max.connections	언제든지 브로커에 허용되는 최대 연결 수입니다. 이 제한은 `max.connections.per.ip` 를 사용하여 구성된 모든 IP당 제한에 추가로 적용됩니다.	Apache Kafka 기본값

속성	설명	기본값
max.connections.per.ip	각 IP 주소에서 허용되는 최대 연결 수입니다. max.connections.per.ip.overrides 속성을 사용하여 구성된 재정의가 있는경우 이 값을 `0`으로 설정할 수 있습니다. 한도에 도달하면 IP 주소의 새 연결이 삭제됩니다.	Apache Kafka 기본값
max.connections.per.ip.overrides	IP 또는 호스트 이름당 쉼표로 구분된 목록은 기본 최대 연결 수로 재정의됩니다. 예시 값은 `hostName:100,127.0.0.1:200` 입니다.	Apache Kafka 기본값

속성	설명	기본값
message.max.bytes	Kafka가 허용하는 최대 레코드 배치 크기입니다. 이 값을 늘리고 0.10.2보다 오래된 소비자가 있는 경우 이 정도 크기의 레코드 배치를 가져올 수 있도록 소비자의 가져오기 크기도 늘려야 합니다. 최신 메시지 형식 버전은 효율성을 위해 항상 메시지를 배치로 그룹화합니다. 이전 메시지 형식 버전에서는 압축되지 않은 레코드를 배치로 그룹화하지 않으며 이러한 경우 이 제한은 단일 레코드에만 적용됩니다. 이 값은 주제 수준 `max.message.bytes` 구성으로 주제별로 설정할 수 있습니다.	Apache Kafka 기본값

속성	설명	기본값
producer.id.expiration.ms	주제 파티션 리더가 생산자 ID 만료 전에 기다리는 시간(ms)입니다. 생산자 ID는 연결된 트랜잭션이 진행 중인 동안에는 만료되지 않습니다. 주제의 보존 설정으로 인해 생산자 ID의 마지막 쓰기가 삭제되면 생산자 ID가 더 빨리 만료될 수 있습니다. 이 값을 `delivery.timeout.ms` 보다 크거나 같게 설정하면 재시도 중 만료를 방지하고 메시지 중복으로부터 보호할 수 있지만 기본값은 대부분의 사용 사례에 적합해야 합니다.	Apache Kafka 기본값

Express 브로커의 주제 수준 구성

Apache Kafka 명령을 사용하여 새 주제와 기존 주제에 대한 주제 수준 구성 속성을 설정하거나 수정할 수 있습니다. 주제 수준 구성을 제공할 수 없는 경우 Amazon MSK는 브로커 기본값을 사용합니다.

브로커 수준 구성과 마찬가지로 Amazon MSK는 일부 주제 수준 구성 속성이 변경되지 않도록 보호합니다. 복제 인수 `min.insync.replicas` 및 `unclean.leader.election.enable`을 예로 들 수 있습니다. 3 이외의 복제 인수 값으로 주제를 생성하려고 시도하는 경우 Amazon MSK는 기본적으로 복제 인수가 3인 주제를 생성합니다. 주제 수준 구성 속성과 설정 방법 예제에 대한 자세한 내용은 Apache Kafka 설명서의 [주제 수준 구성]을 참조하세요.

속성	설명
cleanup.policy	이 구성은 로그 세그먼트에 사용할 보존 정책을 지정합니다. "delete" 정책(기본값)은 보존 기간 또는 크기 제한에 도달하는 경우 이전 세그먼트를 삭제합니다. "compact" 정책은 각 키에 대한 최신 값을 유지하는 로그 압축을 활성화합니다. 쉼표로 구분된 목록에서 두 정책을 모두 지정할 수도 있습니다(예: "delete,compact"). 이 경우 보존 기간 및 크기 구성에 따라 이전 세그먼트는 삭제되고 보존된 세그먼트는 압축됩니다. Express 브로커의 압축은 파티션의 데이터가 256MB에 도달하면 트리거됩니다.
compression.type	주어진 주제에 대한 최종 압축 유형을 지정합니다. 이 구성은 표준 압축 코덱(`gzip`, `snappy`, `lz4`, `zstd`)을 허용합니다. 또한 압축하지 않은 유형과 동일한 `uncompressed` , 생산자가 설정한 원래 압축 코덱을 유지한다는 의미인 `producer`를 허용합니다.
delete.retention.ms	로그 압축 주제에 대한 삭제 톰스톤 마커를 유지하는 시간입니다. 또한 이 설정은 소비자가 오프셋 0에서 시작하는 경우 최종 단계의 유효한 스냅샷을 얻기 위해 읽기를 완료해야 하는 시간에 대한 경계를 제공합니다. 그렇지 않으면 스캔을 완료하기 전에 삭제 톰스톤이 수집될 수 있습니다. 이 설정의 기본값은 86400000(24 * 60 * 60 * 1000ms, 즉 1일), Apache Kafka 기본값입니다.

속성	설명
max.message.bytes	Kafka에서 허용하는 최대 레코드 배치 크기입니다(압축이 활성화된 경우 압축 후). 이 값이 커지고 `0.10.2`보다 오래된 소비자가 있는 경우, 소비자의 가져오기 크기도 이 크기의 레코드 배치를 가져올 수 있도록 커져야 합니다. 최신 메시지 형식 버전에서 레코드는 효율성을 위해 항상 배치로 그룹화됩니다. 이전 메시지 형식 버전에서는 압축되지 않은 레코드가 배치로 그룹화되지 않으며, 이 경우 이 제한은 단일 레코드에만 적용됩니다. 이 값은 주제 수준 `max.message.bytes config` 를 통해 주제별로 설정할 수 있습니다.
message.timestamp.after.max.ms	이 구성은 메시지 타임스탬프와 브로커의 타임스탬프 간에 허용되는 타임스탬프 차이를 설정합니다. 메시지 타임스탬프는 브로커의 타임스탬프보다 크거나 같을 수 있으며, 허용되는 최대 차이는 이 구성에 설정된 값에 따라 결정됩니다. `message.timestamp.type=CreateTime` 인 경우 타임스탬프의 차이가 이 지정된 임계값을 초과하면 메시지가 거부됩니다. 이 구성은 `message.timestamp.type=LogAppendTime` 인 경우 무시됩니다.
message.timestamp.before.max.ms	이 구성은 브로커의 타임스탬프와 메시지 타임스탬프 간의 허용 가능한 타임스탬프 차이를 설정합니다. 메시지 타임스탬프는 브로커의 타임스탬프보다 작거나 같을 수 있으며, 허용되는 최대 차이는 이 구성에 설정된 값에 따라 결정됩니다. `message.timestamp.type=CreateTime` 인 경우 타임스탬프의 차이가 이 지정된 임계값을 초과하면 메시지가 거부됩니다. 이 구성은 `message.timestamp.type=LogAppendTime` 인 경우 무시됩니다.

속성	설명
message.timestamp.type	메시지의 타임스탬프가 메시지 생성 시간인지 아니면 로그 추가 시간인지 정의합니다. 값은 `CreateTime` 또는 `LogAppendTime` 이어야 합니다.
min.compaction.lag.ms	로그에서 메시지가 압축되지 않은 상태로 유지되는 최소 시간입니다. 이 설정은 압축 중인 로그에만 적용됩니다. 이 설정의 기본값은 0, Apache Kafka 기본값입니다.
max.compaction.lag.ms	메시지가 로그의 압축에 적합하지 않은 상태로 유지되는 최대 시간입니다. 이 설정은 압축 중인 로그에만 적용됩니다. 이 구성은 [7일, Long.Max] 범위로 제한됩니다. 이 설정의 기본값은 9223372036854775807, Apache Kafka 기본값입니다.
retention.bytes	이 구성은 "delete" 보존 정책을 사용하는 경우 공간을 확보하기 위해 이전 로그 세그먼트를 폐기하기 전에 파티션(로그 세그먼트로 구성)이 증가할 수 있는 최대 크기를 제어합니다. 기본적으로 크기 제한은 없고 시간 제한만 있습니다. 이 제한은 파티션 수준에서 적용되므로 파티션 수에 곱하여 주제 보존을 바이트 단위로 계산합니다. 또한 `retention.bytes configuration` 은 `segment.ms` 및 `segment.bytes` 구성과 독립적으로 작동합니다. `retention.bytes` 가 0으로 구성된 경우에는 새 세그먼트의 롤링을 트리거합니다.

속성	설명
retention.ms	이 구성은 "delete" 보존 정책을 사용하는 경우 공간을 확보하기 위해 이전 로그 세그먼트를 폐기하기 전에 로그를 보존할 최대 시간을 제어합니다. 이는 소비자가 데이터를 얼마나 빨리 읽어야 하는지에 대한 SLA를 나타냅니다. `-1`로 설정하면 시간 제한이 적용되지 않습니다. 또한 `retention.ms` 구성은 `segment.ms` 및 `segment.bytes` 구성과 독립적으로 작동합니다. `retention.ms` 조건이 충족되면 새 세그먼트의 롤링을 트리거합니다.

Express 브로커 읽기 전용 구성

Amazon MSK는 이러한 구성의 값을 설정하고 해당 값이 변경되어 클러스터의 가용성에 영향을 미치지 않도록 보호합니다. 이러한 값은 클러스터에서 실행되는 Apache Kafka 버전에 따라 변경될 수 있으므로 특정 클러스터의 값을 확인해야 합니다.

다음 표에는 Express 브로커에 대한 읽기 전용 구성이 나열되어 있습니다.

속성	설명	Express 브로커 값
broker.id	이 서버의 브로커 ID입니다.	1,2,3...
broker.rack	브로커의 랙입니다. 이는 내결함성을 위한 랙 인식 복제 할당에 사용됩니다. 예: `RACK1`, `us-east-1d`	AZ ID 또는 서브넷 ID
default.replication.factor	모든 주제에 대한 기본 복제 인수입니다.	3
fetch.max.bytes	가져오기 요청에 대해 반환할 최대 바이트 수입니다.	Apache Kafka 기본값
group.max.size	단일 소비자 그룹이 수용할 수 있는 최대 소비자 수입니다.	Apache Kafka 기본값

속성	설명	Express 브로커 값
inter.broker.listener.name	브로커 간 통신에 사용되는 리스너의 이름입니다.	REPLICATION_SECURE 또는 REPLICATION
inter.broker.protocol.version	브로커 간 프로토콜의 사용 버전을 지정합니다.	Apache Kafka 기본값
리스너	리스너 목록 - 수신할 URI 와 리스너 이름을 쉼표로 구분한 목록입니다. `advertised.listeners property`는 설정할 수 있지만 `listeners` 속성은 설정할 수 없습니다.	MSK 생성
log.message.format.version	브로커가 로그에 메시지를 추가하는 데 사용할 메시지 형식 버전을 지정합니다.	Apache Kafka 기본값

속성	설명	Express 브로커 값
min.insync.replicas	생산자가 ack를 `all`(또는 `-1`)로 설정하면 `min.insync.replicas` 의 값은 쓰기가 성공한 것으로 간주되기 위해 쓰기를 승인해야 하는 최소 복제본 수를 지정합니다. 이 최소값을 충족할 수 없는 경우 생산자는 예외(`NotEnoughReplicas` 또는 `NotEnoughReplicasAfterAppend`)를 발생시킵니다. 생산자의 ack 값을 사용하여 내구성을 강화할 수 있습니다. acks를 "all"로 설정합니다. 이렇게 하면 대부분의 복제본이 쓰기를 수신하지 못하는 경우 생산자가 예외를 발생시킵니다.	2
num.io.threads	서버가 디스크 I/O를 포함할 수 있는 요청을 생성하는 데 사용하는 스레드 수입니다. (m7g.large, 8), (m7g.xlarge, 8), (m7g.2xlarge, 16), (m7g.4xlarge, 32), (m7g.8xlarge, 64), (m7g.12xlarge, 96), (m7g.16xlarge, 128)	인스턴스 유형에 따라 다릅니다. =Math.max(8, 2 * vCPUs)

속성	설명	Express 브로커 값
num.network.threads	서버가 네트워크에서 요청을 수신하고 네트워크에 응답을 보내는 데 사용하는 스레드 수입니다. (m7g.large, 8), (m7g.xlarge, 8), (m7g.2xlarge, 8), (m7g.4xlarge, 16), (m7g.8xlarge, 32), (m7g.12xlarge, 48), (m7g.16xlarge, 64)	인스턴스 유형에 따라 다릅니다. =Math.max(8, vCPUs)
replica.fetch.response.max.bytes	전체 가져오기 응답에 대해 예상되는 최대 바이트 수입니다. 레코드는 배치로 가져오며 가져오기의 첫 번째 비어 있지 않은 파티션의 첫 번째 레코드 배치가 이 값보다 크면 진행을 보장하기 위해 배치가 반환됩니다. 이 값은 절대적인 최대값이 아닙니다. `message.max.bytes` (브로커 구성) 또는 `max.message.bytes` (주제 구성) 속성은 브로커가 허용하는 최대 레코드 배치 크기를 지정합니다.	Apache Kafka 기본값
request.timeout.ms	이 구성은 클라이언트가 요청의 응답을 기다리는 최대 시간을 제어합니다. 제한 시간이 경과하기 전에 응답이 수신되지 않으면 필요한 경우 클라이언트가 요청을 재전송하거나, 재시도가 소진되면 요청에 실패합니다.	Apache Kafka 기본값

속성	설명	Express 브로커 값
transaction.state.log.min.isr	트랜잭션 주제에 대한 `min.insync.replicas` 구성을 재정의했습니다.	2
transaction.state.log.replication.factor	트랜잭션 주제에 대한 복제 인수입니다.	Apache Kafka 기본값
unclean.leader.election.enable	데이터 손실이 발생할 수 있지만 ISR 세트에 없는 복제본이 최후의 수단으로 리더 역할을 수행할 수 있도록 허용합니다.	FALSE

브로커 구성 작업

Apache Kafka 브로커 구성은 정적 또는 동적입니다. 정적 구성을 적용하려면 브로커를 다시 시작해야 합니다. 동적 구성을 업데이트하기 위해 브로커를 다시 시작할 필요는 없습니다. 구성 속성 및 업데이트 모드에 대한 자세한 내용은 Apache Kafka 구성을 참조하세요.

이 주제에서는 사용자 지정 MSK 구성을 생성하고 이러한 구성으로 작업을 수행하는 방법을 설명합니다. MSK 구성을 사용하여 클러스터를 생성하거나 업데이트하는 방법에 대한 자세한 내용은 the section called “주요 기능 및 개념” 단원을 참조하십시오.

주제

- 구성 생성
- 구성 업데이트
- 구성 삭제
- 구성 메타데이터 가져오기
- 구성 개정에 대한 세부 정보 가져오기
- 현재 리전의 계정에 있는 구성 나열
- Amazon MSK 구성 상태

구성 생성

이 프로세스는 사용자 지정 Amazon MSK 구성을 생성하고 이러한 구성에 대한 작업을 수행하는 방법을 설명합니다.

1. 설정할 구성 속성과 해당 구성 속성에 할당할 값을 지정하는 파일을 만듭니다. 다음은 예제 구성 파일의 내용입니다.

```
auto.create.topics.enable = true

log.roll.ms = 604800000
```

2. 다음 AWS CLI명령을 실행하고 *config-file-path*를 이전 단계에서 구성을 저장한 파일의 경로로 바꿉니다.

> **Note**
>
> 구성에 대해 선택하는 이름은 "^[0-9A-Za-z][0-9A-Za-z-]{0,}$"와 일치해야 합니다.

```
aws kafka create-configuration --name "ExampleConfigurationName" --description
 "Example configuration description." --kafka-versions "1.1.1" --server-properties
 fileb://config-file-path
```

다음은 이 명령을 실행한 후 성공적인 응답의 예제입니다.

```
{
    "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
    "CreationTime": "2019-05-21T19:37:40.626Z",
    "LatestRevision": {
        "CreationTime": "2019-05-21T19:37:40.626Z",
        "Description": "Example configuration description.",
        "Revision": 1
    },
    "Name": "ExampleConfigurationName"
}
```

3. 이전 명령은 새로운 구성에 대해 Amazon 리소스 이름(ARN)을 반환합니다. 이 ARN을 저장하십시오. 다른 명령에서 이 구성을 참조할 때 필요합니다. 구성 ARN을 잃어버린 경우 계정에 있는 모든 구성을 나열하여 다시 찾을 수 있습니다.

구성 업데이트

이 프로세스는 사용자 지정 Amazon MSK 구성을 업데이트하는 방법을 설명합니다.

1. 업데이트할 구성 속성과 해당 속성에 할당할 값을 지정하는 파일을 생성합니다. 다음은 예제 구성 파일의 내용입니다.

```
auto.create.topics.enable = true

min.insync.replicas = 2
```

2. 다음 AWS CLI 명령을 실행하여 *config-file-path*를 이전 단계에서 구성을 저장한 파일의 경로로 변경합니다.

   *configuration-arn*을 구성을 생성할 때 가져온 ARN으로 변경합니다. 구성을 생성할 때 ARN을 저장하지 않은 경우 `list-configurations` 명령을 사용하여 계정의 모든 구성을 나열할 수 있습니다. 목록에서 원하는 구성이 응답에 표시됩니다. 구성의 ARN도 해당 목록에 나타납니다.

```
aws kafka update-configuration --arn configuration-arn --description "Example
 configuration revision description." --server-properties fileb://config-file-path
```

3. 다음은 이 명령을 실행한 후 성공적인 응답의 예제입니다.

```
{
    "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
    "LatestRevision": {
        "CreationTime": "2020-08-27T19:37:40.626Z",
        "Description": "Example configuration revision description.",
        "Revision": 2
    }
}
```

구성 삭제

다음 절차는 클러스터에 연결되지 않은 구성을 삭제하는 방법을 보여줍니다. 클러스터에 연결된 구성은 삭제할 수 없습니다.

1. 이 예제를 실행하려면 *configuration-arn*을 구성 생성 시에 가져온 ARN으로 변경합니다. 구성을 생성할 때 ARN을 저장하지 않은 경우 `list-configurations` 명령을 사용하여 계정의 모든 구성을 나열할 수 있습니다. 목록에서 원하는 구성이 응답에 표시됩니다. 구성의 ARN도 해당 목록에 나타납니다.

```
aws kafka delete-configuration --arn configuration-arn
```

2. 다음은 이 명령을 실행한 후 성공적인 응답의 예제입니다.

```
{
    "arn": " arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
    "state": "DELETING"
}
```

구성 메타데이터 가져오기

다음 절차에서는 Amazon MSK 구성을 설명하여 구성에 대한 메타데이터를 가져오는 방법을 보여줍니다.

1. 다음 명령은 구성에 대한 메타데이터를 반환합니다. 구성에 대한 자세한 설명을 보려면 `describe-configuration-revision` 단원을 실행합니다.

   이 예제를 실행하려면 *configuration-arn*을 구성 생성 시에 가져온 ARN으로 변경합니다. 구성을 생성할 때 ARN을 저장하지 않은 경우 `list-configurations` 명령을 사용하여 계정의 모든 구성을 나열할 수 있습니다. 목록에서 원하는 구성이 응답에 표시됩니다. 구성의 ARN도 해당 목록에 나타납니다.

```
aws kafka describe-configuration --arn configuration-arn
```

2. 다음은 이 명령을 실행한 후 성공적인 응답의 예제입니다.

```
{
    "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/abcdabcd-
abcd-1234-abcd-abcd123e8e8e-1",
```

```
    "CreationTime": "2019-05-21T00:54:23.591Z",
    "Description": "Example configuration description.",
    "KafkaVersions": [
        "1.1.1"
    ],
    "LatestRevision": {
        "CreationTime": "2019-05-21T00:54:23.591Z",
        "Description": "Example configuration description.",
        "Revision": 1
    },
    "Name": "SomeTest"
}
```

구성 개정에 대한 세부 정보 가져오기

이 프로세스는 Amazon MSK 구성 개정에 대해 자세히 설명합니다.

`describe-configuration` 명령을 사용하여 MSK 구성을 설명하면 구성의 메타데이터를 볼 수 있습니다. 구성에 대한 설명을 보려면 `describe-configuration-revision` 명령을 사용합니다.

- 다음 명령을 실행하여 *`configuration-arn`*을 구성을 생성할 때 가져온 ARN으로 변경합니다. 구성을 생성할 때 ARN을 저장하지 않은 경우 `list-configurations` 명령을 사용하여 계정의 모든 구성을 나열할 수 있습니다. 응답에 표시되는 목록에서 원하는 구성을 선택합니다. 구성의 ARN도 해당 목록에 나타납니다.

```
aws kafka describe-configuration-revision --arn configuration-arn --revision 1
```

다음은 이 명령을 실행한 후 성공적인 응답의 예제입니다.

```
{
    "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/abcdabcd-abcd-1234-abcd-abcd123e8e8e-1",
    "CreationTime": "2019-05-21T00:54:23.591Z",
    "Description": "Example configuration description.",
    "Revision": 1,
    "ServerProperties":
  "YXV0by5jcmVhdGUudG9waWNzLmVuYWJsZSA9IHRydWUKCgp6b29rZWVwZXIuY29ubmVjdGlvbi50aW1lb3V0Lm1zI
}
```

`ServerProperties` 값은 base64로 인코딩됩니다. base64 디코더(예: https://www.base64decode.org/)를 사용하여 수동으로 디코딩하는 경우 사용자 지정 구성을 만드는 데 사용한 원본 구성 파일의 내용을 가져옵니다. 이 경우 다음과 같은 결과를 얻을 수 있습니다.

```
auto.create.topics.enable = true

log.roll.ms = 604800000
```

현재 리전의 계정에 있는 구성 나열

이 프로세스는 현재 AWS리전의 계정에서 모든 Amazon MSK 구성을 나열하는 방법을 설명합니다.

- 다음 명령을 실행합니다.

```
aws kafka list-configurations
```

다음은 이 명령을 실행한 후 성공적인 응답의 예제입니다.

```
{
    "Configurations": [
        {
            "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-abcd-1234-abcd-abcd123e8e8e-1",
            "CreationTime": "2019-05-21T00:54:23.591Z",
            "Description": "Example configuration description.",
            "KafkaVersions": [
                "1.1.1"
            ],
            "LatestRevision": {
                "CreationTime": "2019-05-21T00:54:23.591Z",
                "Description": "Example configuration description.",
                "Revision": 1
            },
            "Name": "SomeTest"
        },
        {
            "Arn": "arn:aws:kafka:us-east-1:123456789012:configuration/SomeTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e-1",
            "CreationTime": "2019-05-03T23:08:29.446Z",
            "Description": "Example configuration description.",
```

```
                "KafkaVersions": [
                    "1.1.1"
                ],
                "LatestRevision": {
                    "CreationTime": "2019-05-03T23:08:29.446Z",
                    "Description": "Example configuration description.",
                    "Revision": 1
                },
                "Name": "ExampleConfigurationName"
            }
        ]
    }
```

Amazon MSK 구성 상태

Amazon MSK 구성은 다음 상태 중 하나에 해당될 수 있습니다. 구성에 대한 작업을 수행하려면 구성이 `ACTIVE` 또는 `DELETE_FAILED` 상태여야 합니다.

- `ACTIVE`
- `DELETING`
- `DELETE_FAILED`

클러스터의 지능형 리밸런싱

Amazon MSK는 Express 브로커를 사용하여 모든 새 MSK 프로비저닝 클러스터에 대한 지능형 리밸런싱을 제공합니다. 이 기능은 파티션 배포 및 클러스터 조정 작업을 자동으로 관리하므로 타사 도구가 필요하지 않습니다. 지능형 리밸런싱은 클러스터를 확장하거나 축소할 때 파티션을 자동으로 리밸런싱합니다. 또한 클러스터의 상태를 지속적으로 모니터링하여 리소스 불균형 또는 오버로드가 있는지 확인하고 워크로드를 재배포합니다.

지능형 리밸런싱은 30분 이내에 완료되며 조정 중에 클러스터 가용성에 영향을 주지 않는 빠른 조정 작업을 제공합니다. 모든 새 MSK Express 기반 프로비저닝된 클러스터에 대해 기본적으로 켜져 있으며 브로커당 파티션 20,000개의 권장 최대 파티션 한도로 작동합니다. 또한이 기능은 추가 비용 없이 사용할 수 있으며 구성이 필요하지 않습니다.

2025년 11월 20일부터 Amazon MSK Express 브로커가 지원되는 모든 AWS리전에서 Intelligent Rebalancing을 사용할 수 있습니다.

주제

지능형 리밸런싱 작동 방식

Express 브로커가 있는 모든 새 MSK 프로비저닝 클러스터에 대해 기본적으로 지능형 리밸런싱이 켜져 있습니다. 여기에는 다음 상황에 대한 지원이 포함됩니다.

- 확장 및 축소: 한 번의 클릭으로 MSK Express 기반 클러스터에 브로커를 추가하거나 제거할 수 있습니다. 추가하거나 제거할 브로커를 지정하면 지능형 리밸런싱은 내부 AWS모범 사례에 따라 새 클러스터 설정에서 파티션을 자동으로 재배포합니다.
- 안정 상태 리밸런싱:이 기능은 안정 상태에서 클러스터의 상태를 지속적으로 모니터링하고 다음과 같은 경우 파티션을 자동으로 리밸런싱합니다.
  - 리소스 사용률이 브로커 간에 왜곡됩니다.
  - 브로커가 과다 프로비저닝되거나 과소 활용됩니다.
  - 새 브로커가 추가되거나 기존 브로커가 제거됩니다.

> **Note**
>
> 지능형 리밸런싱이 켜져 있으면 파티션 리밸런싱에 Cruise Control과 같은 타사 도구를 사용할 수 없습니다. 이러한 타사 도구에서 제공하는 파티션 재할당 API를 사용하려면 먼저 지능형 재분배를 일시 중지해야 합니다.

Amazon MSK 콘솔에서이 기능을 사용할 수 있습니다. AWS CLI, Amazon MSK APIs 또는 AWSSDK 및를 사용하여이 기능을 사용할 수도 있습니다AWS CloudFormation. 자세한 내용은 Amazon MSK 클러스터 크기 조정 및 안정 상태 리밸런싱 섹션을 참조하세요.

지능형 리밸런싱 지표 모니터링

다음 Amazon CloudWatch 지표를 사용하여 진행 중인 과거 지능형 리밸런싱 작업의 상태를 모니터링할 수 있습니다.

- `RebalanceInProgress`:이 지표는 리밸런싱이 진행 중인 경우 1, 그렇지 않으면 0의 값으로 1분마다 게시됩니다.
- `UnderProvisioned`: 클러스터가 현재 프로비저닝 중이며 파티션 리밸런싱을 수행할 수 없음을 나타냅니다. 브로커를 더 추가하거나 클러스터의 인스턴스 유형을 확장해야 합니다.

MSK 프로비저닝 클러스터 모니터링에 대한 자세한 내용은 ??? 및 섹션을 참조하세요???.

지능형 리밸런싱 사용 시 고려 사항

- 지능형 리밸런싱 지원은 Express 브로커가 있는 새 MSK 프로비저닝 클러스터에서만 사용할 수 있습니다.
- 자동 파티션 재할당의 경우 브로커당 최대 20,000개의 파티션에 대한 최대 지원을 사용할 수 있습니다.
- 지능형 리밸런싱이 활성화된 경우 파티션 재할당 APIs 또는 타사 리밸런싱 도구를 사용할 수 없습니다. 이러한 APIs 또는 타사 도구를 사용하려면 먼저 MSK Express 기반 클러스터에 대한 지능형 리밸런싱을 일시 중지해야 합니다.

단일 작업으로 Amazon MSK 클러스터 확장 및 축소

지능형 리밸런싱을 사용하면 단일 작업으로 클러스터의 브로커 수를 편집하여 클러스터를 확장하거나 축소할 수 있습니다. Amazon MSK 콘솔에서 또는 AWS CLIAmazon MSK APIs 또는 AWSSDK 및를 사용하여이 작업을 수행할 수 있습니다AWS CloudFormation. 브로커 수를 변경하면 Amazon MSK는 다음을 수행합니다.

- 파티션을 새 브로커에 자동으로 배포합니다.
- 제거 중인 브로커에서 파티션을 이동합니다.

클러스터를 확장하거나 축소할 때 클라이언트가 데이터를 생성하고 소비할 수 있는 클러스터 가용성은 영향을 받지 않습니다.

Scaling clusters usingAWS Management 콘솔

1. https://console.aws.amazon.com/msk/home?region=us-east-1#/home/에서 Amazon MSK 콘솔을 엽니다.

2. 클러스터 페이지에서 새로 생성된 Express 기반 클러스터를 선택합니다. 프로비저닝된 Express 기반 클러스터 생성에 대한 자세한 내용은 섹션을 참조하세요1단계: MSK Provisioned 클러스터 생성.
3. 작업 드롭다운 목록에서 브로커 수 편집을 선택합니다.
4. 영역당 브로커 수 편집 페이지에서 다음 중 하나를 수행합니다.
    - 클러스터에 브로커를 더 추가하려면 각 가용 영역에 브로커 추가를 선택한 다음 추가할 브로커 수를 입력합니다.
    - 클러스터에서 브로커를 제거하려면 각 가용 영역에서 하나의 브로커 제거를 선택합니다.
5. 변경 사항 저장을 선택합니다.

Scaling clusters usingAWS CLI

브로커 수를 편집하여 클러스터를 확장하거나 축소할 수 있습니다. 에서이 작업을 수행하려면 다음 예제와 같이 update-broker-count 명령을 AWS CLI사용합니다. 이 명령에서 `target-broker-count` 파라미터의 클러스터에 원하는 브로커 수를 지정합니다.

```
aws msk update-broker-count --cluster-arn arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-5678-90ef-ghij-klmnopqrstuv-1 --
current-version ABCDEF1GHIJK0L --target-broker-count 6
```

Scaling clusters usingAWSSDK

브로커 수를 프로그래밍 방식으로 편집하여 클러스터를 확장하거나 축소할 수 있습니다. AWSSDK를 사용하여이 작업을 수행하려면 다음 예제와 같이 UpdateBrokerCount API를 사용합니다. `TargetNumberOfBrokerNodes` 파라미터에 클러스터에서 원하는 브로커 수를 지정합니다.

```
update_broker_count_response = client.update_broker_count(
    ClusterArn='arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-5678-90ef-ghij-klmnopqrstuv-1',
    CurrentVersion='ABCDEF1GHIJK0L',
    TargetNumberOfBrokerNodes=6
)
```

Amazon MSK 클러스터의 정상 상태 재분배

정상 상태 리밸런싱은 Express 브로커가 있는 모든 새 MSK 프로비저닝 클러스터에 대해 기본적으로 활성화되는 지능형 리밸런싱 기능의 일부입니다. 클러스터를 확장하거나 축소하면 Amazon MSK는 파

티션을 새 브로커에 배포하고 제거 예정인 브로커에서 파티션을 이동하여 파티션 관리를 자동으로 처리합니다. 브로커 간에 워크로드를 최적으로 분산하기 위해 지능형 리밸런싱은 Amazon MSK 모범 사례를 사용하여 브로커에 대한 리밸런싱을 자동으로 시작하기 위한 임계값을 결정합니다.

필요한 경우 안정 상태 재분배를 일시 중지했다가 재개할 수 있습니다. 정상 상태 리밸런싱은 클러스터를 지속적으로 모니터링하고 다음을 수행합니다.

- 브로커 리소스 사용량(CPU, 네트워크, 스토리지)을 추적합니다.
- 데이터 가용성에 영향을 주지 않고 파티션 배치를 자동으로 조정합니다.
- 표준 브로커에 비해 Express 브로커에서 최대 180배 빠른 리밸런싱 작업을 완료합니다.
- 클러스터 성능을 유지합니다.

Pause and resume steady state rebalancing inAWS Management 콘솔

1. https://console.aws.amazon.com/msk/home?region=us-east-1#/home/에서 Amazon MSK 콘솔을 엽니다.
2. 클러스터 페이지에서 Express 기반 클러스터를 선택합니다. 프로비저닝된 Express 기반 클러스터 생성에 대한 자세한 내용은 섹션을 참조하세요1단계: MSK Provisioned 클러스터 생성.
3. 클러스터 세부 정보 페이지에서 지능형 리밸런싱 상태가 활성인지 확인합니다. 지능형 리밸런싱을 사용할 수 없거나 상태가 일시 중지됨인 경우 새 Express 기반 클러스터를 생성합니다.
4. 작업 드롭다운 목록에서 지능형 재분배 편집을 선택합니다.
5. 지능형 리밸런싱 편집 페이지에서 다음을 수행합니다.
   a. 일시 중지됨을 선택합니다.
   b. 변경 사항 저장을 선택합니다.

Pause and resume steady state rebalancing usingAWS CLI

를 **ACTIVE** 사용하여 클러스터의 리밸런싱 상태를 로 설정하려면 다음 예제와 같이 update-rebalancing 명령을 AWS CLI사용합니다. 이 명령에서 `rebalancing` 파라미터를 사용하여 상태를 지정합니다.

```
aws msk update-rebalancing --cluster-arn arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-5678-90ef-ghij-klmnopqrstuv-1
 --current-version ABCDEF1GHIJK0L --rebalancing "{\"Rebalancing\":{\"Status\":
\"ACTIVE\"}}"
```

Pause and resume steady state rebalancing usingAWSSDK

UpdateRebalancingRequest API를 사용하여 클러스터의 리밸런싱 상태를 설정하여 브로커 수를 프로그래밍 방식으로 수정할 수도 있습니다. 다음 예제에서는 리밸런싱 상태를 **ACTIVE** 및 로 설정하는 방법을 보여줍니다**PAUSED**.

```
final UpdateRebalancingRequest updateRebalancingRequest = new
 UpdateRebalancingRequest()
    .withClusterArn(arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-5678-90ef-ghij-klmnopqrstuv-1)
    .withCurrentVersion(ABCDEF1GHIJK0L)
    .withRebalancing(new Rebalancing().withStatus("ACTIVE"));
```

```
final UpdateRebalancingRequest updateRebalancingRequest = new
 UpdateRebalancingRequest()
    .withClusterArn(arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-5678-90ef-ghij-klmnopqrstuv-1)
    .withCurrentVersion(ABCDEF1GHIJK0L)
    .withRebalancing(new Rebalancing().withStatus("PAUSED"));
```

MSK Provisioned 클러스터에 패치 적용

Amazon MSK는 주기적으로 클러스터 브로커의 소프트웨어를 업데이트합니다. 유지 관리에는 계획된 업데이트 또는 계획되지 않은 수리가 포함됩니다. 계획된 유지 관리에는 클러스터의 상태, 보안 및 성능을 유지하는 데 필요한 운영 체제 업데이트, 보안 업데이트 및 기타 소프트웨어 업데이트가 포함됩니다. 갑작스러운 인프라 성능 저하를 해결하기 위해 계획되지 않은 유지 관리를 수행합니다. Standard 브로커와 Express 브로커에 대한 유지 관리를 수행하지만 환경은 다릅니다.

Standard 브로커에 대한 패치 적용

Standard 브로커에 대한 업데이트는 모범 사례를 따르는 경우 애플리케이션의 쓰기 및 읽기에 영향을 주지 않습니다.

Amazon MSK는 소프트웨어에 대한 롤링 업데이트를 사용하여 클러스터의 고가용성을 유지합니다. 이 프로세스 중에 브로커는 한 번에 하나씩 재부팅되고 Kafka는 리더십을 다른 온라인 브로커로 자동으로 이동합니다. Kafka 클라이언트에는 파티션의 리더십 변화를 자동으로 감지하고 MSK 클러스터에 데이터를 계속 쓰고 읽을 수 있는 메커니즘이 내장되어 있습니다. 패치 적용 중을 포함하여 클러스터를 항상 원활하게 작동하려면 Apache Kafka 클라이언트의 모범 사례 섹션을 따르세요.

브로커가 오프라인 상태가 된 후 클라이언트에서 일시적인 연결 해제 오류가 표시되는 것은 정상입니다. 또한 짧은 기간(최대 2분이며 일반적으로 더 짧음) 동안 p99 읽기 및 쓰기 지연 시간(일반적으로 높은 밀리초, 최대 약 2초)의 어느 정도 급증도 관찰할 수 있습니다. 이러한 스파이크는 예상되며 클라이언트가 새 리더 브로커에 다시 연결해도 발생합니다. 이는 생산 또는 소비에 영향을 미치지 않으며 재연결 후 이 문제는 해결됩니다. 자세한 내용은 [오프라인 및 클라이언트 장애 조치 브로커](#)를 참조하세요.

또한 종료된 브로커의 파티션은 더 이상 데이터를 복제하지 않으므로 예상되는 `UnderReplicatedPartitions` 지표가 증가하는 것을 확인할 수 있습니다. 이는 다른 브로커에서 호스팅되는 이러한 파티션에 대한 복제본이 이제 요청을 처리하고 있으므로 애플리케이션의 쓰기 및 읽기에 영향을 주지 않습니다.

소프트웨어 업데이트 후 브로커가 다시 온라인 상태가 되면 오프라인 상태일 때 생성된 메시지를 '확인'해야 합니다. 확인 중에 볼륨 처리량 및 CPU 사용량이 증가하는 것을 관찰할 수도 있습니다. 이러한 증가는 브로커에 충분한 CPU, 메모리, 네트워크 및 볼륨 리소스가 있는 경우 클러스터에 대한 쓰기 및 읽기에는 영향을 미치지 않습니다.

Express 브로커에 대한 패치 적용

Express 브로커에는 유지 관리 기간이 없습니다. Amazon MSK는 시간 분산 방식으로 클러스터를 지속적으로 자동 업데이트합니다. 즉, 한 달 동안 간헐적이고 단수적인 브로커 재부팅을 예상할 수 있습니다. 이렇게 하면 클러스터 전체의 일회성 유지 관리 기간에 계획이나 조정을 할 필요가 없습니다. 항상 그렇듯이 브로커 재부팅 중에는 리더십이 요청을 계속 처리하는 다른 브로커로 변경되기 때문에 트래픽이 중단되지 않습니다.

Express 브로커는 유지 관리 중에 발생할 수 있는 로드 변경에 대해 클러스터를 복원할 수 있도록 모범 사례 설정 및 가드레일로 구성되어 제공됩니다. Amazon MSK는 클러스터 오버로드로 인해 브로커 재시작 중에 발생할 수 있는 문제를 완화하기 위해 Express 브로커에 처리량 할당량을 설정합니다. 이러한 개선으로 인해 Express 브로커를 사용할 때 사전 알림, 계획 및 유지 관리 기간이 필요하지 않습니다.

Express 브로커는 항상 세 가지 방법으로 데이터를 복제하므로 재부팅 중에 클라이언트가 자동으로 장애 조치를 수행합니다. 복제 인수가 1 또는 2로 설정되어 있기 때문에 주제를 사용할 수 없게 되는 문제에 대해 걱정할 필요가 없습니다. 또한 다시 시작된 Express 브로커를 따라잡는 것이 표준 브로커보다 빠릅니다. Express 브로커의 패치 적용 속도가 빨라지면 클러스터에 대해 예약한 컨트롤 플레인 활동에 대한 계획 중단이 최소화됩니다.

모든 Apache Kafka 애플리케이션과 마찬가지로 Express 브로커에 연결하는 클라이언트에 대한 공유 클라이언트-서버 계약이 여전히 있습니다. 브로커 간의 리더십 장애 조치를 처리하도록 클라이언트를 구성하는 것은 여전히 중요합니다. 패치 적용 중을 포함하여 클러스터를 항상 원활하게 작동하려면

Apache Kafka 클라이언트의 모범 사례 섹션을 따르세요. 브로커가 다시 시작된 후 클라이언트에서 일시적인 연결 해제 오류가 표시되는 것은 정상입니다. 팔로워 브로커가 파티션 리더십을 맡게 되므로 이는 생산 및 소비에 영향을 주지 않습니다. Apache Kafka 클라이언트는 자동으로 장애 조치를 수행하고 새 리더 브로커에 요청을 보내기 시작합니다.

브로커 오프라인 및 클라이언트 장애 조치

Kafka는 오프라인 브로커를 허용합니다. 모범 사례에 따라 정상적이고 균형 잡힌 클러스터의 단일 오프라인 브로커는 영향을 주거나 생산 또는 소비 실패를 초래하지 않습니다. 다른 브로커가 파티션 리더십을 인수하고 Kafka 클라이언트 lib가 자동으로 장애 조치를 취하고 새 리더 브로커에게 요청을 보내기 시작하기 때문입니다.

클라이언트 서버 계약

따라서 클라이언트 라이브러리와 서버 측 동작 간의 공유 계약이 발생합니다. 서버는 하나 이상의 새 리더를 성공적으로 할당해야 하며 클라이언트는 적시에 새로운 리더에게 요청을 보내도록 브로커를 변경해야 합니다.

Kafka는 예외를 사용하여 이 흐름을 제어합니다.

예제 절차

1. 브로커 A가 오프라인 상태가 됩니다.
2. Kafka 클라이언트에서 예외(일반적으로 네트워크 연결 해제 또는 not_leader_for_partition)를 수신합니다.
3. 이러한 예외는 Kafka 클라이언트에서 메타데이터를 업데이트하여 최신 리더에 대해 인식할 수 있도록 합니다.
4. Kafka 클라이언트는 다른 브로커의 새로운 파티션 리더에게 요청을 다시 보냅니다.

이 프로세스는 일반적으로 판매된 Java 클라이언트 및 기본 구성에서 2초 미만이 걸립니다. 클라이언트 측 오류는 상세화되고 반복적이지만 'WARN' 레벨로 표시된 정도의 우려의 대상이 아닙니다.

예제: 예외 1

```
10:05:25.306 [kafka-producer-network-thread | producer-1] WARN
o.a.k.c.producer.internals.Sender - [Producer clientId=producer-1] Got
error produce response with correlation id 864845 on topic-partition
msk-test-topic-1-0, retrying (2147483646 attempts left). Error:
NETWORK_EXCEPTION. Error Message: Disconnected from node 2
```

예제: 예외 2

```
10:05:25.306 [kafka-producer-network-thread | producer-1] WARN
o.a.k.c.producer.internals.Sender - [Producer clientId=producer-1] Received
invalid metadata error in produce request on partition msk-test-topic-1-41
due to org.apache.kafka.common.errors.NotLeaderOrFollowerException: For
requests intended only for the leader, this error indicates that the broker
is not the current leader. For requests intended for any replica, this
error indicates that the broker is not a replica of the topic partition..
Going to request metadata update now"
```

Kafka 클라이언트는 일반적으로 1초~3초 이내에 이러한 오류를 자동으로 해결합니다. 이는 클라이언트 측 지표에서 p99의 생산/소비 지연 시간(일반적으로 100초대의 높은 밀리초)으로 표시됩니다. 이 시간보다 길면 일반적으로 클라이언트 구성 또는 서버 측 컨트롤러 로드에 문제가 있음을 나타냅니다. 문제 해결 섹션을 참조하세요.

성공적인 장애 조치는 트래픽 및 리더십이 예상대로 이동했음을 입증하는 다른 브로커에 대한 `BytesInPerSec` 및 `LeaderCount` 지표 증가를 확인하여 알아볼 수 있습니다. 또한 복제본이 종료 브로커에서 오프라인 상태일 때 예상되는 `UnderReplicatedPartitions` 지표 증가도 볼 수 있습니다.

문제 해결

클라이언트-서버 계약을 위반하면 위 흐름이 중단될 수 있습니다. 가장 일반적인 문제의 이유는 다음과 같습니다.

- Kafka 클라이언트 lib의 잘못된 구성 또는 잘못된 사용
- 타사 클라이언트 lib를 사용하는 예기치 않은 기본 동작 및 버그
- 오버로드된 컨트롤러로 인해 파티션 리더 할당 속도 감소
- 새로운 컨트롤러가 선택되어 파티션 리더 할당 속도 감소

리더십 장애 조치를 처리하기 위한 올바른 동작을 보장하려면 다음을 수행하는 것이 좋습니다.

- 느린 리더십 할당을 방지하기 위해 컨트롤러 브로커를 적절하게 조정하려면 서버 측 모범 사례를 따라야 합니다.
- 클라이언트가 장애 조치를 처리하도록 하려면 클라이언트 라이브러리에서 재시도가 활성화되어 있어야 합니다.

- 연결/요청 급증을 방지하려면 클라이언트 라이브러리에 retry.backoff.ms 구성(기본값 100)이 있어야 합니다.
- 클라이언트 라이브러리에서 request.timeout.ms 및 delivery.timeout.ms 애플리케이션의 SLA와 일치하는 값으로 설정해야 합니다. 값이 높을수록 특정 장애 유형에 대한 장애 조치가 느려집니다.
- 클라이언트 라이브러리에서 초기 검색에 대한 가용성 영향을 방지하기 위해 boottrap.servers에 최소 3개의 무작위 브로커가 포함되어 있는지 확인해야 합니다.
- 일부 클라이언트 라이브러리는 다른 클라이언트 라이브러리보다 수준이 낮으며 애플리케이션 개발자가 재시도 로직 및 예외 처리 자체를 구현할 것으로 기대합니다. 사용 예제는 클라이언트 lib 관련 설명서를 참조하고 올바른 재연결/재시도 로직을 따라야 합니다.
- 생성, 성공한 요청 수 및 재시도할 수 없는 오류 수에 대한 클라이언트 측 지연 시간을 모니터링하는 것이 좋습니다.
- 이전 타사 Golang 및 Ruby 라이브러리는 생산 및 소비 요청이 영향을 받지 않음에도 불구하고 전체 브로커 오프라인 기간 동안 계속 상세화되는 것으로 관찰되었습니다. 로그에 실제 영향과 노이즈가 있는지 확인하려면 요청 지표 외에 성공 및 오류에 대한 비즈니스 수준 지표를 항상 모니터링하는 것이 좋습니다.
- network/not_leader에 대한 일시적 예외는 일반적이고 영향을 미치지 않으며 kafka 프로토콜의 일부로 예상되므로 고객은 이 예외에 대해 불안해 하지 않아도 됩니다.
- UnderReplicatedPartitions는 일반적이고 영향을 미치지 않으며 단일 오프라인 브로커 중에 예상되므로 고객은 이에 대해 불안해 하지 않아도 됩니다.

Amazon MSK의 보안

의 클라우드 보안AWS이 최우선 순위입니다. AWS고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS는 AWS클라우드에서 AWS서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. Amazon Managed Streaming for Apache Kafka에 적용되는 규정 준수 프로그램에 대해 알아보려면 규정 준수 프로그램별 범위 내 Amazon Web Services를 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS서비스에 따라 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon MSK를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 Amazon MSK를 구성하는 방법을 설명합니다. 또한 Amazon MSK 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 Amazon Web Services를 사용하는 방법도 알아봅니다.

주제

Amazon Managed Streaming for Apache Kafka의 데이터 보호

AWS공동 책임 모델 Amazon Managed Streaming for Apache Kafka의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS보안 블로그의 AWS공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고AWS 계정 AWS IAM Identity Center또는 AWS Identity and Access Management(IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다AWS CloudTrail. CloudTrail 추적을 사용하여 AWS활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS암호화 솔루션을 사용합니다AWS 서비스.

- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon MSK 또는 기타AWS 서비스에서 콘솔, API AWS CLI또는AWSSDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

주제

- Amazon MSK 암호화
- Amazon MSK 암호화 시작하기
- 인터페이스 VPC 엔드포인트와 함께 Amazon MSK API 사용

Amazon MSK 암호화

Amazon MSK는 엄격한 데이터 관리 요구 사항을 충족하는 데 사용할 수 있는 데이터 암호화 옵션을 제공합니다. Amazon MSK가 암호화에 사용하는 인증서는 13개월마다 갱신해야 합니다. Amazon MSK는 모든 클러스터에 대해 해당 인증서를 자동으로 갱신합니다. Amazon MSK가 인증서 업데이트 작업을 시작할 때 Express 브로커 클러스터는 `ACTIVE` 상태를 유지합니다. Standard 브로커 클러스터의 경우 Amazon MSK는 인증서 업데이트 작업을 시작할 때 클러스터의 상태를 `MAINTENANCE`로 설정합니다. 업데이트가 완료되면 MSK는 클러스터의 상태를 다시 `ACTIVE`로 설정합니다. 클러스터가 인증서 업데이트 작업을 진행하는 동안 계속해서 데이터를 생성하고 사용할 수 있지만 해당 클러스터에 대한 업데이트 작업은 수행할 수 없습니다.

저장 시 Amazon MSK 암호화

Amazon MSK는 AWS Key Management Service(KMS)와 통합되어 투명한 서버 측 암호화 기능을 제공합니다. Amazon MSK는 항상 저장 데이터를 암호화합니다. MSK 클러스터를 생성하는 경우 Amazon MSK가 저장 데이터를 암호화하는 데 사용할 AWS KMS key를 지정할 수 있습니다. KMS 키를 지정하지 않으면 Amazon MSK에서 AWS 관리형 키를 생성하고 사용자를 대신하여 사용합니다. KMS 키에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS keys 섹션을 참조하십시오.

전송 중 Amazon MSK 암호화

Amazon MSK는 TLS 1.2를 사용합니다. 기본적으로 MSK 클러스터의 브로커 간에 전송 중인 데이터를 암호화합니다. 클러스터를 생성할 때 이 기본값을 재정의할 수 있습니다.

클라이언트와 브로커 간 통신의 경우 다음 세 가지 설정 중 하나를 지정해야 합니다.

- TLS 암호화 데이터만 허용합니다. 기본 설정입니다.
- 일반 텍스트와 TLS 암호화 데이터를 모두 허용합니다.
- 일반 텍스트 데이터만 허용합니다.

Amazon MSK 브로커는 퍼블릭 AWS Certificate Manager인증서를 사용합니다. 따라서 Amazon Trust Services를 신뢰하는 모든 트러스트 스토어는 Amazon MSK 브로커의 인증서도 신뢰합니다.

전송 중 암호화를 활성화하는 것이 좋지만 이 경우 CPU 오버헤드와 몇 밀리초의 지연 시간이 추가될 수 있습니다. 그러나 대부분의 사용 사례는 이러한 차이에 민감하지 않으며, 클러스터, 클라이언트, 사용 프로필 구성에 따라 영향을 미치는 정도는 달라집니다.

Amazon MSK 암호화 시작하기

MSK 클러스터를 생성할 때 JSON 형식으로 암호화 설정을 지정할 수 있습니다. 다음은 예입니다.

```
{
   "EncryptionAtRest": {
       "DataVolumeKMSKeyId": "arn:aws:kms:us-east-1:123456789012:key/abcdabcd-1234-abcd-1234-abcd123e8e8e"
    },
   "EncryptionInTransit": {
        "InCluster": true,
        "ClientBroker": "TLS"
    }
}
```

`DataVolumeKMSKeyId`의 경우 고객 관리형 키를 지정하거나 계정의 MSK용 AWS 관리형 키(`alias/aws/kafka`)를 지정할 수 있습니다. 를 지정하지 않으면 `EncryptionAtRest`Amazon MSK는 여전히 에서 저장 데이터를 암호화합니다AWS 관리형 키. 클러스터에서 사용 중인 키를 확인하려면 GET 요청을 보내거나 `DescribeCluster` API 작업을 호출합니다.

`EncryptionInTransit`의 경우 `InCluster`의 기본값은 true입니다. 그러나 브로커 간에 데이터가 전달될 때 Amazon MSK가 암호화하지 않도록 하려면 이를 false로 설정할 수 있습니다.

클라이언트와 브로커 사이에 전송되는 데이터에 암호화 모드를 지정하려면 `ClientBroker`를 `TLS`, `TLS_PLAINTEXT` 또는 `PLAINTEXT`의 세 가지 값 중 하나로 설정합니다.

주제

- [Amazon MSK 클러스터를 생성할 때 암호화 설정 지정](#)
- [Amazon MSK TLS 암호화 테스트](#)

Amazon MSK 클러스터를 생성할 때 암호화 설정 지정

이 프로세스는 Amazon MSK 클러스터를 생성할 때 암호화 설정을 지정하는 방법을 설명합니다.

클러스터를 생성할 때 암호화 설정 지정

1. 이전 예제의 내용을 파일에 저장하고 파일에 원하는 이름을 지정합니다. 예를 들어, `encryption-settings.json`이라고 지정합니다.
2. `create-cluster` 명령을 실행하고 `encryption-info` 옵션을 사용하여 구성 JSON을 저장한 파일을 가리킵니다. 다음은 예입니다. *{YOUR MSK VERSION}*을 Apache Kafka 클라이언트 버전과 일치하는 버전으로 변경합니다. MSK 클러스터 버전을 찾는 방법에 대한 자세한 내용은 [MSK 클러스터 버전 확인](#) 섹션을 참조하세요. MSK 클러스터 버전과 다른 Apache Kafka 클라이언트 버전을 사용하면 Apache Kafka 데이터 손상, 손실, 가동 중지가 발생할 수 있습니다.

```
aws kafka create-cluster --cluster-name "ExampleClusterName" --broker-node-group-
info file://brokernodegroupinfo.json --encryption-info file://encryptioninfo.json
 --kafka-version "{YOUR MSK VERSION}" --number-of-broker-nodes 3
```

   다음은 이 명령을 실행한 후 성공적인 응답의 예입니다.

```
{
    "ClusterArn": "arn:aws:kafka:us-east-1:123456789012:cluster/SecondTLSTest/
abcdabcd-1234-abcd-1234-abcd123e8e8e",
    "ClusterName": "ExampleClusterName",
    "State": "CREATING"
}
```

Amazon MSK TLS 암호화 테스트

이 프로세스는 Amazon MSK에서 TLS 암호화를 테스트하는 방법을 설명합니다.

TLS 암호화를 테스트하려면

1. [the section called “클라이언트 머신 생성”](#)의 지침에 따라 클라이언트 머신을 만듭니다.
2. 클라이언트 머신에 Apache Kafka를 설치합니다.
3. 이 예제에서는 JVM 트러스트스토어를 사용하여 MSK 클러스터와 통신합니다. 이렇게 하려면 먼저 클라이언트 머신에 `/tmp`라는 폴더를 만듭니다. 그런 다음 Apache Kafka 설치 폴더인 `bin`으로 이동하여 다음 명령을 실행하십시오. (JVM 경로는 다를 수 있습니다.)

```
cp /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/lib/security/
cacerts /tmp/kafka.client.truststore.jks
```

4. 클라이언트 머신의 Apache Kafka 설치 폴더인 `bin`에 있는 동안 다음 내용으로 `client.properties`라는 텍스트 파일을 만듭니다.

```
security.protocol=SSL
ssl.truststore.location=/tmp/kafka.client.truststore.jks
```

5. 가 AWS CLI설치된 시스템에서 다음 명령을 실행하여 *clusterARN*을 클러스터의 ARN으로 바꿉니다.

```
aws kafka get-bootstrap-brokers --cluster-arn clusterARN
```

성공적인 결과는 다음과 같습니다. 다음 단계에 필요하므로 이 결과를 저장하십시오.

```
{
    "BootstrapBrokerStringTls": "a-1.example.g7oein.c2.kafka.us-
east-1.amazonaws.com:0123,a-3.example.g7oein.c2.kafka.us-
east-1.amazonaws.com:0123,a-2.example.g7oein.c2.kafka.us-east-1.amazonaws.com:0123"
}
```

6. 다음 명령을 실행하여 *BootstrapBrokerStringTls*를 이전 단계에서 얻은 브로커 엔드포인트 중 하나로 변경합니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-
list BootstrapBrokerStringTls --producer.config client.properties --topic
 TLSTestTopic
```

7. 새 명령 창을 열고 동일한 클라이언트 머신에 연결합니다. 그러면 다음 명령을 실행하여 콘솔 소비자를 생성합니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerStringTls --consumer.config client.properties --topic TLSTestTopic
```

8. 생산자 창에서 문자 메시지와 반환을 입력하고 소비자 창에서 동일한 메시지를 찾습니다. Amazon MSK는 이 메시지를 전송하는 동안 암호화했습니다.

암호화된 데이터로 작업하도록 Apache Kafka 클라이언트를 구성하는 방법에 대한 자세한 내용은 Kafka 클라이언트 구성을 참조하십시오.

인터페이스 VPC 엔드포인트와 함께 Amazon MSK API 사용

AWSPrivateLink로 구동되는 인터페이스 VPC 엔드포인트를 사용하여 Amazon VPC와 Amazon MSK APIs이 Amazon 네트워크를 벗어나지 않도록 할 수 있습니다. 인터페이스 VPC 엔드포인트에는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWSDirect Connect 연결이 필요하지 않습니다. AWSPrivateLink는 Amazon VPC의 프라이빗 IPs와 함께 탄력적 네트워크 인터페이스를 사용하여 AWS서비스 간에 프라이빗 통신을 가능하게 하는 AWS기술입니다. 자세한 내용은 Amazon Virtual Private Cloud 및 인터페이스 VPC 엔드포인트(AWSPrivateLink)를 참조하세요.

애플리케이션은AWSPrivateLink를 사용하여 Amazon MSK Provisioned 및 MSK Connect APIs에 연결할 수 있습니다. 시작하려면 Amazon VPC 리소스의 트래픽이 인터페이스 VPC 엔드포인트를 통해 흐름을 시작하도록 Amazon MSK API에 대한 인터페이스 VPC 엔드포인트를 생성합니다. FIPS 지원 인터페이스 VPC 엔드포인트는 미국 리전에서 사용할 수 있습니다. 자세한 내용은 인터페이스 엔드포인트 생성을 참조하세요.

이 기능을 사용하면 Apache Kafka 클라이언트가 인터넷을 순회하여 연결 문자열을 검색하지 않고도 연결 문자열을 동적으로 가져와 MSK Provisioned 또는 MSK Connect 리소스와 연결할 수 있습니다.

인터페이스 VPC 엔드포인트를 생성할 때 다음 서비스 이름 엔드포인트 중 하나를 선택합니다.

MSK Provisioned의 경우:

- com.amazonaws.region.kafka
- com.amazonaws.region.kafka-fips(FIPS 지원)

여기서 region은 리전 이름입니다. MSK Provisioned 호환 API를 사용하려면 이 서비스 이름을 선택합니다. 자세한 내용은 https://docs.aws.amazon.com/msk/1.0/apireference/의 작업을 참조하세요.

MSK Connect의 경우:

- com.amazonaws.region.kafkaconnect

여기서 region은 리전 이름입니다. MSK Connect 호환 API를 사용하려면 이 서비스 이름을 선택합니다. 자세한 내용은 Amazon Connect API 참조의 작업을 참조하세요.

인터페이스 VPC 엔드포인트를 생성하는 단계별 지침을 비롯한 자세한 내용은 AWS PrivateLink 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

Amazon MSK Provisioned 또는 MSK Connect API의 VPC 엔드포인트에 대한 액세스 제어

VPC 엔드포인트 정책을 사용하면 정책을 VPC 엔드포인트에 연결하거나 IAM 사용자, 그룹 또는 역할에 연결된 정책의 추가 필드를 사용하여 액세스를 제어할 수 있습니다. 이를 통해 지정된 VPC 엔드포인트를 통해서만 발생하도록 액세스를 제한할 수 있습니다. 적절한 예제 정책을 사용하여 MSK Provisioned 또는 MSK Connect 서비스에 대한 액세스 권한을 정의합니다.

엔드포인트를 생성할 때 정책을 연결하지 않으면 Amazon VPC는 서비스에 대한 전체 액세스를 허용하는 기본 정책을 자동으로 연결합니다. 엔드포인트 정책은 IAM ID 기반 정책 또는 서비스별 정책을 재정의하거나 대체하지 않습니다. 이는 엔드포인트에서 지정된 서비스로의 액세스를 제어하기 위한 별도의 정책입니다.

자세한 내용은 AWS PrivateLink 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

MSK Provisioned — VPC policy example

읽기 전용 액세스

이 샘플 정책은 VPC 엔드포인트에 연결할 수 있습니다. 자세한 내용은 Amazon VPC 리소스에 대한 액세스 제어를 참조하십시오. 연결된 VPC 엔드포인트를 통해서만 작업을 나열하고 설명할 수 있도록 작업을 제한합니다.

```
{
  "Statement": [
    {
      "Sid": "MSKReadOnly",
      "Principal": "*",
      "Action": [
        "kafka:List*",
```

```
        "kafka:Describe*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

MSK Provisioned - VPC 엔드포인트 정책 예제

특정 MSK 클러스터에 대한 액세스 제한

이 샘플 정책은 VPC 엔드포인트에 연결할 수 있습니다. 연결된 VPC 엔드포인트를 통한 특정 Kafka 클러스터에 대한 액세스를 제한합니다.

```
{
  "Statement": [
    {
      "Sid": "AccessToSpecificCluster",
      "Principal": "*",
      "Action": "kafka:*",
      "Effect": "Allow",
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/MyCluster"
    }
  ]
}
```

MSK Connect — VPC endpoint policy example

커넥터 나열 및 새 커넥터 생성

다음은 MSK Connect에 대한 엔드포인트 정책의 예입니다. 이 정책은 지정된 역할이 커넥터를 나열하고 새 커넥터를 생성하도록 허용합니다.

JSON

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

```
                "Effect": "Allow",
                "Action": [
                    "kafkaconnect:ListConnectors",
                    "kafkaconnect:CreateConnector"
                ],
                "Resource": "*",
                "Principal": {
                    "AWS": [
                        "arn:aws:iam::111122223333:role/<ExampleRole>"
                    ]
                }
            }
        ]
    }
```

MSK Connect — VPC 엔드포인트 정책 예제

지정된 VPC에 있는 특정 IP 주소의 요청만 허용

다음 예제는 지정된 VPC의 지정된 IP 주소에서 들어오는 요청만 성공하도록 허용하는 정책을 보여줍니다. 다른 IP 주소에서 들어오는 요청은 실패합니다.

```
{
    "Statement": [
        {
            "Action": "kafkaconnect:*",
            "Effect": "Allow",
            "Principal": "*",
            "Resource": "*",
            "Condition": {
                "IpAddress": {
                    "aws:VpcSourceIp": "192.0.2.123"
                },
        "StringEquals": {
                    "aws:SourceVpc": "vpc-555555555555"
                }
            }
        }
    ]
}
```

Amazon MSK API에 대한 인증 및 권한 부여

AWS Identity and Access Management(IAM)는 관리자가 AWS리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스되는 입니다. IAM 관리자는 Amazon MSK 리소스를 사용할 수 있도록 인증(로그인)하고 권한을 부여(권한 보유)할 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스있는 입니다.

주제

- Amazon MSK와 IAM의 작동 방식
- Amazon MSK ID 기반 정책 예제
- Amazon MSK의 서비스 연결 역할
- AWSAmazon MSK에 대한 관리형 정책
- Amazon MSK 자격 증명 및 액세스 문제 해결

Amazon MSK와 IAM의 작동 방식

IAM을 사용하여 Amazon MSK에 대한 액세스를 관리하기 전에 Amazon MSK에서 사용할 수 있는 IAM 기능을 이해해야 합니다. Amazon MSK 및 기타 AWS서비스에서 IAM을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWSIAM으로 작업하는 서비스를 참조하세요.

주제

- Amazon MSK ID 기반 정책
- Amazon MSK 리소스 기반 정책
- Amazon MSK 태그 기반 권한 부여
- Amazon MSK IAM 역할

Amazon MSK ID 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. Amazon MSK는 특정 작업, 리소스, 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알고 싶다면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon MSK 자격 증명 기반 정책에 대한 작업

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 엔터티가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

Amazon MSK의 정책 작업은 작업 앞에 `kafka:` 접두사를 사용합니다. 예를 들어 누군가에게 Amazon MSK `DescribeCluster` API 작업으로 MSK 클러스터를 설명할 수 있는 권한을 부여하려면 정책에 `kafka:DescribeCluster` 작업을 포함하면 됩니다. 정책 문에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다. Amazon MSK는 이 서비스로 수행할 수 있는 태스크를 설명하는 자체 작업 세트를 정의합니다.

MSK 주제 APIs에 대한 정책 작업은 작업 앞에 `kafka-cluster` 접두사를 사용합니다. 단원을 참조하십시오[IAM 권한 부여 정책 작업 및 리소스의 의미](#).

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": ["kafka:action1", "kafka:action2"]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `Describe`라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "kafka:Describe*"
```

Amazon MSK 작업 목록을 보려면 IAM 사용 설명서에서 [Amazon Managed Streaming for Apache Kafka를 위한 작업, 리소스, 조건 키](#)를 참조하세요.

Amazon MSK 자격 증명 기반 정책에 대한 리소스

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 엔터티가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 [Amazon 리소스 이름(ARN)](#)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon MSK 인스턴스 리소스의 ARN은 다음과 같습니다.

```
arn:${Partition}:kafka:${Region}:${Account}:cluster/${ClusterName}/${UUID}
```

ARN 형식에 대한 자세한 내용은 Amazon 리소스 이름(ARNs) 및 AWS서비스 네임스페이스를 참조하세요.

예를 들어 문에서 `CustomerMessages` 인스턴스를 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/CustomerMessages/abcd1234-abcd-dcba-4321-a1b2abcd9f9f-2"
```

특정 계정에 속하는 모든 인스턴스를 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/*"
```

리소스 생성 태스크와 같은 일부 Amazon MSK 태스크는 특정 리소스에서 수행할 수 없습니다. 이러한 경우, 와일드카드(*)를 사용해야 합니다.

```
"Resource": "*"
```

단일 문에서 여러 리소스를 지정하려면 ARN을 쉼표로 구분합니다.

```
"Resource": ["resource1", "resource2"]
```

Amazon MSK 리소스 유형과 해당 ARN의 목록을 보려면 IAM 사용 설명서에서 Amazon Managed Streaming for Apache Kafka로 정의된 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Managed Streaming for Apache Kafka에서 정의한 작업을 참조하세요.

Amazon MSK 자격 증명 기반 정책의 조건 키

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 엔터티가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS전역 조건 키를 보려면 IAM 사용 설명서의 AWS전역 조건 컨텍스트 키를 참조하세요.

Amazon MSK는 자체 조건 키 세트를 정의하며 일부 전역 조건 키 사용도 지원합니다. 모든 AWS전역 조건 키를 보려면 IAM 사용 설명서의 AWS전역 조건 컨텍스트 키를 참조하세요.

Amazon MSK 조건 키 목록을 보려면 IAM 사용 설명서에서 Amazon Managed Streaming for Apache Kafka를 위한 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Managed Streaming for Apache Kafka에서 정의한 작업을 참조하세요.

Amazon MSK 자격 증명 기반 정책의 예제

Amazon MSK ID 기반 정책 예제를 보려면 Amazon MSK ID 기반 정책 예제 섹션을 참조하세요.

Amazon MSK 리소스 기반 정책

Amazon MSK는 Amazon MSK 클러스터와 함께 사용할 수 있는 클러스터 정책(리소스 기반 정책이라고도 함)을 지원합니다. 클러스터 정책을 사용하여 Amazon MSK 클러스터에 대한 프라이빗 연결을 설정할 수 있는 크로스 계정 권한이 있는 IAM 주체를 정의할 수 있습니다. IAM 클라이언트 인증과 함께 사용하는 경우, 클러스터 정책을 사용하여 연결 클라이언트에 대한 Kafka 데이터 영역 권한을 세분화하여 정의할 수도 있습니다.

클러스터 정책에 지원되는 최대 크기는 20KB입니다.

클러스터 정책을 구성하는 방법의 예를 보려면 2단계: MSK 클러스터에 클러스터 정책 연결 섹션을 참조하세요.

Amazon MSK 태그 기반 권한 부여

Amazon MSK 클러스터에 태그를 연결할 수 있습니다. 태그에 근거하여 액세스를 제어하려면 `kafka:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys`조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다. Amazon MSK 리소스 태그 지정에 대한 자세한 내용은 the section called "클러스터에 태그 지정" 섹션을 참조하세요.

태그를 통해서만 클러스터 액세스를 제어할 수 있습니다. 주제 및 소비자 그룹에 태그를 지정하려면 태그 없이 정책에 별도의 문을 추가해야 합니다.

클러스터에서 태그 기반으로 해당 클러스터에 대한 액세스를 제한하는 ID 기반 정책의 예를 보려면 태그를 기반으로 Amazon MSK 클러스터에 액세스 섹션을 참조하세요.

ID 기반 정책의 조건을 사용하여 태그를 기반으로 Amazon MSK 리소스에 대한 액세스를 제어할 수 있습니다. 다음 예에서는 사용자가 클러스터를 설명하고, 부트스트랩 브로커를 가져오고, 브로커 노드를 나열, 업데이트, 삭제할 수 있도록 허용하는 정책을 보여줍니다. 그러나 이 정책은 클러스터 태그 `Owner`에 해당 사용자의 `username` 값이 있는 경우에만 권한을 부여합니다. 다음 정책의 두 번째 문은 클러스터의 주제에 대한 액세스를 허용합니다. 이 정책의 첫 번째 문은 주제 액세스를 승인하지 않습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AccessClusterIfOwner",
      "Effect": "Allow",
      "Action": [
        "kafka:Describe*",
        "kafka:Get*",
        "kafka:List*",
        "kafka:Update*",
        "kafka:Delete*"
      ],
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:*Topic*",
        "kafka-cluster:WriteData",
        "kafka-cluster:ReadData"
      ],
      "Resource": [
        "arn:aws:kafka:us-east-1:123456789012:topic/*"
      ]
    }
  ]
}
```

Amazon MSK IAM 역할

IAM 역할은 특정 권한이 있는 Amazon Web Services 계정 내의 엔터티입니다.

Amazon MSK에서 임시 자격 증명 사용

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. AssumeRole 또는 GetFederationToken과 같은 AWS STSAPI 작업을 호출하여 임시 보안 자격 증명을 얻습니다.

Amazon MSK는 임시 보안 인증 정보 사용을 지원합니다.

서비스 연결 역할

서비스 연결 역할을 사용하면 Amazon Web Services가 다른 서비스의 리소스에 액세스하여 사용자 대신 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고 서비스가 소유합니다. 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.

Amazon MSK는 서비스 연결 역할을 지원합니다. Amazon MSK 서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 the section called “서비스 연결 역할” 섹션을 참조하세요.

Amazon MSK ID 기반 정책 예제

기본적으로 IAM 사용자 및 역할에는 Amazon MSK API 작업을 실행할 수 있는 권한이 없습니다. 관리자는 지정된 리소스에서 특정 API 태스크를 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하세요.

주제

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon MSK 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다AWS 계정. 사용 사례에 맞는 AWS고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용자 설명서의 AWS 관리형 정책 또는 AWS직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용자 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다CloudFormation. 자세한 내용은 IAM 사용자 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용자 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용자 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용자 설명서의 IAM의 보안 모범 사례를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI또는 AWSAPI를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

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

```
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

하나의 Amazon MSK 클러스터 액세스

이 예에서는 Amazon Web Services 계정의 IAM 사용자에게 클러스터 중 하나인 `purchaseQueriesCluster`에 대한 액세스 권한을 부여하려고 합니다. 이 정책은 사용자가 클러스터를 설명하고, 부트스트랩 브로커를 가져오고, 브로커 노드를 나열하고 업데이트할 수 있도록 허용합니다.

JSON

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

```
            "Effect":"Allow",
            "Action":[
               "kafka:Describe*",
               "kafka:Get*",
               "kafka:List*",
               "kafka:Update*"
            ],
            "Resource":"arn:aws:kafka:us-east-1:012345678012:cluster/
purchaseQueriesCluster/abcdefab-1234-abcd-5678-cdef0123ab01-2"
         }
      ]
}
```

태그를 기반으로 Amazon MSK 클러스터에 액세스

ID 기반 정책의 조건을 사용하여 태그를 기반으로 Amazon MSK 리소스에 대한 액세스를 제어할 수 있습니다. 이 예에서는 사용자가 클러스터를 설명하고, 부트스트랩 브로커를 가져오고, 브로커 노드를 나열, 업데이트, 삭제할 수 있도록 허용하는 정책 생성 방법을 보여줍니다. 하지만 클러스터 태그 `Owner`가 해당 사용자의 사용자 이름 값을 가지고 있는 경우에만 권한이 부여됩니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AccessClusterIfOwner",
      "Effect": "Allow",
      "Action": [
        "kafka:Describe*",
        "kafka:Get*",
        "kafka:List*",
        "kafka:Update*",
        "kafka:Delete*"
      ],
      "Resource": "arn:aws:kafka:us-east-1:012345678012:cluster/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
```

```
        }
      ]
    }
```

이 정책을 계정의 IAM 사용자에게 연결할 수 있습니다. `richard-roe`라는 사용자가 MSK 클러스터를 업데이트하려고 시도하는 경우 클러스터에 `Owner=richard-roe` 또는 `owner=richard-roe` 태그를 지정해야 합니다. 그렇지 않으면 액세스가 거부됩니다. 조건 키 이름은 대소문자를 구분하지 않기 때문에 조건 태그 키 `Owner`는 `Owner` 및 `owner` 모두와 일치합니다. 자세한 정보는 IAM 사용 설명서의 [IAM JSON 정책 요소: 조건]을 참조하세요.

Amazon MSK의 서비스 연결 역할

Amazon MSK는 AWS Identity and Access Management(IAM) [서비스 연결 역할을] 사용합니다. 서비스 연결 역할은 Amazon MSK에 직접 연결되는 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon MSK에서 사전 정의하며 서비스가 사용자를 대신하여 다른 AWS서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 Amazon MSK를 더 간편하게 설정할 수 있습니다. Amazon MSK는 서비스 연결 역할의 권한을 정의합니다. 달리 정의되지 않는 한, Amazon MSK만이 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

서비스 연결 역할을 지원하는 다른 서비스에 대한 자세한 내용은 [IAM과 함께 작동하는 Amazon Web Services]를 참조하고 서비스 연결 역할 열에서 예로 표시된 서비스를 찾아보세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

주제

- Amazon MSK에 대한 서비스 연결 역할 권한
- Amazon MSK에 대한 서비스 연결 역할 생성
- Amazon MSK에 대한 서비스 연결 역할 편집
- Amazon MSK 서비스 연결 역할에 대해 지원되는 리전

Amazon MSK에 대한 서비스 연결 역할 권한

Amazon MSK는 AWSServiceRoleForKafka라는 서비스 연결 역할을 사용합니다. Amazon MSK는 해당 역할을 사용하여 리소스에 액세스하고 다음과 같은 작업을 수행합니다.

- `*NetworkInterface` – 고객 계정에서 네트워크 인터페이스를 생성하고 관리하여 고객 VPC의 클라이언트가 클러스터 브로커에 액세스할 수 있도록 합니다.
- `*VpcEndpoints` -를 사용하여 고객 VPC의 클라이언트가 클러스터 브로커에 액세스할 수 있도록 하는 고객 계정의 VPC 엔드포인트를 관리합니다AWS PrivateLink. Amazon MSK는 `DescribeVpcEndpoints`, `ModifyVpcEndpoint`, `DeleteVpcEndpoints`에 대한 권한을 사용합니다.
- `secretsmanager` -를 사용하여 클라이언트 자격 증명을 관리합니다AWS Secrets Manager.
- `GetCertificateAuthorityCertificate` - 프라이빗 인증 기관의 인증서를 검색합니다.

이 서비스 연결 역할은 관리형 정책 `KafkaServiceRolePolicy`에 연결됩니다. 이 정책에 대한 업데이트는 [KafkaServiceRolePolicy]를 참조하세요.

AWSServiceRoleForKafka 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `kafka.amazonaws.com`

역할 권한 정책을 통해 Amazon MSK는 리소스에 대해 다음 작업을 완료할 수 있습니다.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface",
    "ec2:DescribeNetworkInterfaces",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:AttachNetworkInterface",
    "ec2:DeleteNetworkInterface",
    "ec2:DetachNetworkInterface",
    "ec2:DescribeVpcEndpoints",
    "acm-pca:GetCertificateAuthorityCertificate",
    "secretsmanager:ListSecrets"
   ],
   "Resource": "*"
  },
  {
```

```
    "Effect": "Allow",
    "Action": [
     "ec2:ModifyVpcEndpoint"
    ],
    "Resource": "arn:*:ec2:*:*:subnet/*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:DeleteVpcEndpoints",
     "ec2:ModifyVpcEndpoint"
    ],
    "Resource": "arn:*:ec2:*:*:vpc-endpoint/*",
    "Condition": {
     "StringEquals": {
      "ec2:ResourceTag/AWSMSKManaged": "true"
     },
     "StringLike": {
      "ec2:ResourceTag/ClusterArn": "*"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "secretsmanager:GetResourcePolicy",
     "secretsmanager:PutResourcePolicy",
     "secretsmanager:DeleteResourcePolicy",
     "secretsmanager:DescribeSecret"
    ],
    "Resource": "*",
    "Condition": {
     "ArnLike": {
      "secretsmanager:SecretId": "arn:*:secretsmanager:*:*:secret:AmazonMSK_*"
     }
    }
   }
  ]
}
```

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 권한](#) 섹션을 참조하세요.

Amazon MSK에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Management 콘솔AWS CLI, 또는 AWSAPI에서 Amazon MSK 클러스터를 생성하면 Amazon MSK가 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. Amazon MSK 클러스터를 생성하는 경우 Amazon MSK에서 서비스 연결 역할을 다시 생성합니다.

Amazon MSK에 대한 서비스 연결 역할 편집

Amazon MSK에서는 AWSServiceRoleForKafka 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 연결 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 편집](#)을 참조하세요.

Amazon MSK 서비스 연결 역할에 대해 지원되는 리전

Amazon MSK는 서비스가 제공되는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 설명은 [AWS리전 및 엔드포인트](#)를 참조하세요.

AWSAmazon MSK에 대한 관리형 정책

AWS관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다AWS. AWS관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS관리형 정책은 모든 AWS고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 [고객 관리형 정책](#)을 정의하여 권한을 줄이는 것이 좋습니다.

AWS관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS업데이트하는AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 미칩니다. AWSAWS 서비스는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 되면 AWS관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 가이드의 [AWS 관리형 정책](#)을 참조하세요.

AWS관리형 정책: AmazonMSKFullAccess

이 정책은 모든 Amazon MSK 작업에 대한 전체 액세스 권한을 허용하는 관리 권한을 보안 주체에게 부여합니다. 해당 정책의 권한은 다음과 같이 그룹화됩니다.

- Amazon MSK 권한은 모든 Amazon MSK 작업을 허용합니다.
- **Amazon EC2** 권한 - 이 정책에서 이 권한은 API 요청에서 전달된 리소스를 검증하는 데 필요합니다. 이는 Amazon MSK가 클러스터에서 리소스를 성공적으로 사용할 수 있도록 하기 위한 것입니다. 이 정책의 나머지 Amazon EC2 권한은 Amazon MSK가 클러스터에 연결하는 데 필요한 AWS리소스를 생성하도록 허용합니다.
- **AWS KMS** 권한 - 이 정책에서 이 권한은 API 호출 중에 요청에서 전달된 리소스를 검증하는 데 사용됩니다. Amazon MSK가 전달된 키를 Amazon MSK 클러스터에서 사용할 수 있도록 하기 위해 필요합니다.
- **CloudWatch Logs, Amazon S3, and Amazon Data Firehose** 권한 - 이 권한은 Amazon MSK가 로그 전송 대상에 도달할 수 있는지와 브로커 로그 사용에 유효한지 확인하는 데 필요합니다.
- **IAM** 권한 – 이 권한은 계정에서 서비스 연결 역할을 생성하고 서비스 실행 역할을 Amazon MSK에 전달할 수 있도록 하는 데 필요합니다.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": [
    "kafka:*",
    "ec2:DescribeSubnets",
    "ec2:DescribeVpcs",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeRouteTables",
    "ec2:DescribeVpcEndpoints",
    "ec2:DescribeVpcAttribute",
    "kms:DescribeKey",
    "kms:CreateGrant",
    "logs:CreateLogDelivery",
    "logs:GetLogDelivery",
    "logs:UpdateLogDelivery",
    "logs:DeleteLogDelivery",
    "logs:ListLogDeliveries",
    "logs:PutResourcePolicy",
    "logs:DescribeResourcePolicies",
    "logs:DescribeLogGroups",
    "S3:GetBucketPolicy",
```

```
          "firehose:TagDeliveryStream"
         ],
         "Resource": "*"
        },
        {
         "Effect": "Allow",
         "Action": [
          "ec2:CreateVpcEndpoint"
         ],
         "Resource": [
          "arn:*:ec2:*:*:vpc/*",
          "arn:*:ec2:*:*:subnet/*",
          "arn:*:ec2:*:*:security-group/*"
         ]
        },
        {
         "Effect": "Allow",
         "Action": [
          "ec2:CreateVpcEndpoint"
         ],
         "Resource": [
          "arn:*:ec2:*:*:vpc-endpoint/*"
         ],
         "Condition": {
          "StringEquals": {
           "aws:RequestTag/AWSMSKManaged": "true"
          },
          "StringLike": {
           "aws:RequestTag/ClusterArn": "*"
          }
         }
        },
        {
         "Effect": "Allow",
         "Action": [
          "ec2:CreateTags"
         ],
         "Resource": "arn:*:ec2:*:*:vpc-endpoint/*",
         "Condition": {
          "StringEquals": {
           "ec2:CreateAction": "CreateVpcEndpoint"
          }
         }
        },
```

```
      {
       "Effect": "Allow",
       "Action": [
        "ec2:DeleteVpcEndpoints"
       ],
       "Resource": "arn:*:ec2:*:*:vpc-endpoint/*",
       "Condition": {
        "StringEquals": {
         "ec2:ResourceTag/AWSMSKManaged": "true"
        },
        "StringLike": {
         "ec2:ResourceTag/ClusterArn": "*"
        }
       }
      },
      {
       "Effect": "Allow",
       "Action": "iam:PassRole",
       "Resource": "*",
       "Condition": {
        "StringEquals": {
         "iam:PassedToService": "kafka.amazonaws.com"
        }
       }
      },
      {
       "Effect": "Allow",
       "Action": "iam:CreateServiceLinkedRole",
       "Resource": "arn:aws:iam::*:role/aws-service-role/kafka.amazonaws.com/
AWSServiceRoleForKafka*",
       "Condition": {
        "StringLike": {
         "iam:AWSServiceName": "kafka.amazonaws.com"
        }
       }
      },
      {
       "Effect": "Allow",
       "Action": [
        "iam:AttachRolePolicy",
        "iam:PutRolePolicy"
       ],
       "Resource": "arn:aws:iam::*:role/aws-service-role/kafka.amazonaws.com/
AWSServiceRoleForKafka*"
```

```
        },
        {
          "Effect": "Allow",
          "Action": "iam:CreateServiceLinkedRole",
          "Resource": "arn:aws:iam::*:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery*",
          "Condition": {
            "StringLike": {
              "iam:AWSServiceName": "delivery.logs.amazonaws.com"
            }
          }
        }

      ]
    }
```

AWS관리형 정책: AmazonMSKReadOnlyAccess

이 정책은 Amazon MSK의 정보를 볼 수 있는 읽기 전용 권한을 사용자에게 부여합니다. 이 정책이 첨부된 보안 주체는 기존 리소스를 업데이트하거나 삭제할 수 없으며 새 Amazon MSK 리소스를 생성할 수도 없습니다. 예를 들어 이러한 권한이 있는 주체는 자신의 계정과 연결된 클러스터 및 구성 목록을 볼 수 있지만 클러스터의 구성이나 설정을 변경할 수는 없습니다. 해당 정책의 권한은 다음과 같이 그룹화됩니다.

- **Amazon MSK** 권한 – 이 권한을 통해 Amazon MSK 리소스를 나열하고, 리소스를 설명하고, 리소스에 대한 정보를 얻을 수 있습니다.
- **Amazon EC2** 권한 – 이 권한은 클러스터와 연결된 Amazon VPC, 서브넷, 보안 그룹, ENI를 설명하는 데 사용됩니다.
- **AWS KMS** 권한 – 이 권한은 클러스터와 연결된 키를 설명하는 데 사용됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "kafka:Describe*",
                "kafka:List*",
```

```
                "kafka:Get*",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "kms:DescribeKey"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

AWS관리형 정책: KafkaServiceRolePolicy

IAM 엔터티에 KafkaServiceRolePolicy를 연결할 수 없습니다. 이 정책은 서비스 연결 역할에 첨부되어 Amazon MSK가 MSK 클러스터에서 VPC 엔드포인트(커넥터) 관리, 네트워크 인터페이스 관리, AWS Secrets Manager로 클러스터 보안 인증 정보 관리와 같은 작업을 수행할 수 있도록 합니다. 자세한 내용은 the section called "서비스 연결 역할" 단원을 참조하십시오.

AWS관리형 정책: AWSMSKReplicatorExecutionRole

이 `AWSMSKReplicatorExecutionRole` 정책은 MSK 클러스터 간에 데이터를 복제할 수 있는 권한을 Amazon MSK Replicator에 부여합니다. 해당 정책의 권한은 다음과 같이 그룹화됩니다.

- **`cluster`** – IAM 인증을 사용하여 클러스터에 연결할 수 있는 Amazon MSK Replicator 권한을 부여합니다. 또한 클러스터를 설명하고 변경할 수 있는 권한을 부여합니다.
- **`topic`** - 주제를 설명, 생성 및 변경하고 주제의 동적 구성을 변경할 수 있는 권한을 Amazon MSK Replicator에 부여합니다.
- **`consumer group`** - 소비자 그룹을 설명 및 변경하고, MSK 클러스터에서 날짜를 읽고 쓰고, Replicator에서 생성한 내부 주제를 삭제할 수 있는 권한을 Amazon MSK Replicator에 부여합니다.

JSON

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

```
     "Effect": "Allow",
     "Action": [
      "kafka-cluster:Connect",
      "kafka-cluster:DescribeCluster",
      "kafka-cluster:AlterCluster",
      "kafka-cluster:DescribeTopic",
      "kafka-cluster:CreateTopic",
      "kafka-cluster:AlterTopic",
      "kafka-cluster:WriteData",
      "kafka-cluster:ReadData",
      "kafka-cluster:AlterGroup",
      "kafka-cluster:DescribeGroup",
      "kafka-cluster:DescribeTopicDynamicConfiguration",
      "kafka-cluster:AlterTopicDynamicConfiguration",
      "kafka-cluster:WriteDataIdempotently"
     ],
     "Resource": [
      "arn:aws:kafka:*:*:cluster/*"
     ]
    },
    {
     "Sid": "TopicPermissions",
     "Effect": "Allow",
     "Action": [
      "kafka-cluster:DescribeTopic",
      "kafka-cluster:CreateTopic",
      "kafka-cluster:AlterTopic",
      "kafka-cluster:WriteData",
      "kafka-cluster:ReadData",
      "kafka-cluster:DescribeTopicDynamicConfiguration",
      "kafka-cluster:AlterTopicDynamicConfiguration",
      "kafka-cluster:AlterCluster"
     ],
     "Resource": [
      "arn:aws:kafka:*:*:topic/*/*"
     ]
    },
    {
     "Sid": "GroupPermissions",
     "Effect": "Allow",
     "Action": [
      "kafka-cluster:AlterGroup",
      "kafka-cluster:DescribeGroup"
     ],
```

```
    "Resource": [
     "arn:aws:kafka:*:*:group/*/*"
    ]
   }
  ]
}
```

AWS관리형 정책에 대한 Amazon MSK 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 Amazon MSK의 AWS관리형 정책 업데이트에 대한 세부 정보를 봅니다.

변경	설명	Date
WriteDataIdempotently 권한이 AWSMSKReplicatorExecutionRole에 추가됨 - 기존 정책의 업데이트	Amazon MSK에서 MSK 클러스터 간의 데이터 복제를 지원하기 위해 AWSMSKReplicatorExecutionRole 정책에 WriteDataIdempotently 권한을 추가했습니다.	2024년 3월 12일
AWSMSKReplicatorExecutionRole - 새로운 정책	Amazon MSK에서 Amazon MSK Replicator를 지원하기 위해 AWSMSKReplicatorExecutionRole 정책을 추가했습니다.	2023년 12월 4일
AmazonMSKFullAccess - 기존 정책에 대한 업데이트	Amazon MSK가 Amazon MSK Replicator를 지원하기 위한 권한을 추가했습니다.	2023년 9월 28일
KafkaServiceRolePolicy — 기존 정책에 대한 업데이트	Amazon MSK가 다중 VPC 프라이빗 연결을 지원하기 위한 권한을 추가했습니다.	2023년 3월 8일
AmazonMSKFullAccess - 기존 정책에 대한 업데이트	Amazon MSK가 클러스터에 연결할 수 있도록 새로운	2021년 11월 30일

변경	설명	Date
	Amazon EC2 권한을 추가했습니다.	
AmazonMSKFullAccess - 기존 정책에 대한 업데이트	Amazon MSK가 Amazon EC2 라우팅 테이블을 설명할 수 있는 새로운 권한을 추가했습니다.	2021년 11월 19일
Amazon MSK에서 변경 사항 추적 시작	Amazon MSK는 AWS관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 11월 19일

Amazon MSK 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Amazon MSK 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon MSK에서 작업을 수행할 권한이 없음

Amazon MSK에서 작업을 수행할 권한이 없음

에서 작업을 수행할 권한이 없다는 AWS Management 콘솔메시지가 표시되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 클러스터를 삭제하려고 하지만 `kafka:`*`DeleteCluster`* 권한이 없는 경우 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 kafka:DeleteCluster on resource: purchaseQueriesCluster
```

이 경우, Mateo는 `purchaseQueriesCluster` 작업을 사용하여 `kafka:DeleteCluster` 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

Apache Kafka API에 대한 인증 및 권한 부여

IAM을 사용하여 클라이언트를 인증하고 Apache Kafka 작업을 허용하거나 거부할 수 있습니다. 또는 TLS나 SASL/SCRAM을 사용하여 클라이언트를 인증하고 Apache Kafka ACL을 사용하여 작업을 허용하거나 거부할 수 있습니다.

클러스터에서 Amazon MSK 작업을 수행할 수 있는 사용자를 제어하는 방법에 대한 자세한 내용은 the section called “Amazon MSK API에 대한 인증 및 권한 부여” 섹션을 참조하세요.

주제

IAM 액세스 제어

Amazon MSK를 위한 IAM 액세스 제어를 사용하면 MSK 클러스터에 대한 인증과 권한 부여를 모두 처리할 수 있습니다. 이렇게 하면 인증에 한 메커니즘을 사용하고 권한 부여에 다른 메커니즘을 사용할 필요가 없습니다. 예를 들어 클라이언트가 클러스터에 쓰기를 시도할 때 Amazon MSK는 IAM을 사용하여 해당 클라이언트가 인증된 자격 증명인지 여부와 클러스터에 생성할 수 있는 권한이 있는지 여부를 확인합니다.

IAM 액세스 제어는 Python, Go, JavaScript 및 .NET으로 작성된 Kafka 클라이언트를 포함하여 Java 및 비 Java 클라이언트에서 작동합니다. Java가 아닌 클라이언트에 대한 IAM 액세스 제어는 Kafka 버전 2.7.1 이상의 MSK 클러스터에서 사용할 수 있습니다.

IAM 액세스 제어를 가능하게 하기 위해 Amazon MSK는 Apache Kafka 소스 코드를 약간 수정합니다. 이러한 수정 사항으로 인해 Apache Kafka 환경이 눈에 띄게 달라지지는 않습니다. Amazon MSK는 액세스 이벤트를 기록하므로 이를 감사할 수 있습니다.

IAM 액세스 제어를 사용하는 MSK 클러스터에 대해 Apache Kafka ACL API를 호출할 수 있습니다. 그러나 Apache Kafka ACL은 IAM ID에 대한 권한 부여에 영향을 미치지 않습니다. IAM 정책을 사용하여 IAM ID에 대한 액세스를 제어해야 합니다.

> **중요 고려 사항**
>
> MSK 클러스터에서 IAM 액세스 제어를 사용하는 경우 다음 중요 고려 사항에 유의하세요.

- IAM 액세스 제어는 Apache ZooKeeper 노드에는 적용되지 않습니다. 이러한 노드에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 [Amazon MSK 클러스터의 Apache ZooKeeper 노드에 대한 액세스 제어](#) 섹션을 참조하세요.
- 클러스터에서 IAM 액세스 제어를 사용하는 경우 `allow.everyone.if.no.acl.found` Apache Kafka 설정은 적용되지 않습니다.
- IAM 액세스 제어를 사용하는 MSK 클러스터에 대해 Apache Kafka ACL API를 호출할 수 있습니다. 그러나 Apache Kafka ACL은 IAM ID에 대한 권한 부여에 영향을 미치지 않습니다. IAM 정책을 사용하여 IAM ID에 대한 액세스를 제어해야 합니다.

Amazon MSK의 IAM 액세스 제어 작동 방식

Amazon MSK에 대한 IAM 액세스 제어를 사용하려면 다음 단계를 수행하세요. 이러한 단계는 이 섹션의 나머지 부분에서 자세히 설명합니다.

- IAM 액세스 제어를 사용하는 Amazon MSK 클러스터 생성
- IAM 액세스 제어를 위한 클라이언트 구성
- IAM 역할에 대한 권한 부여 정책 생성
- IAM 액세스 제어를 위한 부트스트랩 브로커 가져오기

IAM 액세스 제어를 사용하는 Amazon MSK 클러스터 생성

이 섹션에서는 AWS Management 콘솔, API 또는를 사용하여 IAM 액세스 제어를 사용하는 Amazon MSK 클러스터를 AWS CLI생성하는 방법을 설명합니다. 기존 클러스터에 대한 IAM 액세스 제어를 사용 설정하는 방법에 대한 자세한 내용은 Amazon MSK 클러스터의 보안 설정 업데이트 섹션을 참조하세요.

AWS Management 콘솔를 사용하여 IAM 액세스 제어를 사용하는 클러스터 생성

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 클러스터 생성을 선택합니다.
3. 사용자 지정 설정으로 클러스터 생성을 선택합니다.
4. 인증 섹션에서 IAM 액세스 제어를 선택합니다.
5. 클러스터 생성을 위한 나머지 워크플로를 완료합니다.

API 또는 AWS CLI를 사용하여 IAM 액세스 제어를 사용하는 클러스터 생성

- IAM 액세스 제어를 사용하도록 설정한 클러스터를 생성하려면 [CreateCluster](#) API 또는 [create-cluster](#) CLI 명령을 사용하고 `ClientAuthentication` 파라미터에 대해 다음 JSON(`"ClientAuthentication": { "Sasl": { "Iam": { "Enabled": true } }`)을 전달합니다.

IAM 액세스 제어를 위한 클라이언트 구성

클라이언트가 IAM 액세스 제어를 사용하는 MSK 클러스터와 통신할 수 있도록 하려면 다음 메커니즘 중 하나를 사용할 수 있습니다.

- SASL_OAUTHBEARER 메커니즘을 사용하여 비 Java 클라이언트 구성
- SASL_OAUTHBEARER 메커니즘 또는 AWS_MSK_IAM 메커니즘을 사용하여 Java 클라이언트 구성

SASL_OAUTHBEARER 메커니즘을 사용하여 IAM 구성

1. 다음 Python Kafka 클라이언트 예제를 사용하여 client.properties 구성 파일을 편집합니다. 구성 변경은 다른 언어에서도 비슷합니다.

```
from kafka import KafkaProducer
from kafka.errors import KafkaError
from kafka.sasl.oauth import AbstractTokenProvider
import socket
import time
from aws_msk_iam_sasl_signer import MSKAuthTokenProvider

class MSKTokenProvider():
    def token(self):
        token, _ = MSKAuthTokenProvider.generate_auth_token('<myAWS ##>')
        return token

tp = MSKTokenProvider()

producer = KafkaProducer(
    bootstrap_servers='<myBootstrapString>',
    security_protocol='SASL_SSL',
    sasl_mechanism='OAUTHBEARER',
    sasl_oauth_token_provider=tp,
```

```
    client_id=socket.gethostname(),
)

topic = "<my-topic>"
while True:
    try:
        inp=input(">")
        producer.send(topic, inp.encode())
        producer.flush()
        print("Produced!")
    except Exception:
        print("Failed to send message:", e)

producer.close()
```

2. 선택한 구성 언어의 헬퍼 라이브러리를 다운로드하고 해당 언어 라이브러리 홈페이지의 시작하기 섹션에 있는 지침을 따르세요.

   - JavaScript: https://github.com/aws/aws-msk-iam-sasl-signer-js#getting-started
   - Python: https://github.com/aws/aws-msk-iam-sasl-signer-python#get-started
   - Go: https://github.com/aws/aws-msk-iam-sasl-signer-go#getting-started
   - .NET: https://github.com/aws/aws-msk-iam-sasl-signer-net#getting-started
   - JAVA: Java용 SASL_OAUTHBEARER 지원은 `aws-msk-iam-auth` jar 파일을 통해 사용 가능

MSK 사용자 지정 AWS_MSK_IAM 메커니즘을 사용하여 IAM을 구성

1. `client.properties` 파일에 다음을 추가합니다. *<PATH_TO_TRUST_STORE_FILE>*을 클라이언트의 트러스트 스토어 파일에 대한 정규화된 경로로 변경합니다.

   > **Note**
   >
   > 특정 인증서를 사용하지 않으려면 `client.properties` 파일에서 `ssl.truststore.location=`*<PATH_TO_TRUST_STORE_FILE>*을 제거하면 됩니다. `ssl.truststore.location`의 값을 지정하지 않으면 Java 프로세스에서 기본 인증서를 사용합니다.

   ```
   ssl.truststore.location=<PATH_TO_TRUST_STORE_FILE>
   security.protocol=SASL_SSL
   ```

```
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
```

AWS자격 증명에 대해 생성한 명명된 프로필을 사용하려면 클라이언트 구성 파일에 `awsProfileName="your profile name";`를 포함합니다. 명명된 프로필에 대한 자세한 내용은 AWS CLI설명서의 명명된 프로필을 참조하세요.

2. 안정적인 최신 aws-msk-iam-auth JAR 파일을 다운로드하여 클래스 경로에 배치합니다. Maven을 사용하는 경우 필요에 따라 버전 번호를 조정하여 다음 종속성을 추가합니다.

```
<dependency>
    <groupId>software.amazon.msk</groupId>
    <artifactId>aws-msk-iam-auth</artifactId>
    <version>1.0.0</version>
</dependency>
```

Amazon MSK 클라이언트 플러그인은 Apache 2.0 라이선스에 따라 오픈 소스로 제공됩니다.

IAM 역할에 대한 권한 부여 정책 생성

권한 부여 정책을 클라이언트에 해당하는 IAM 역할에 연결합니다. 권한 부여 정책에서 역할에 대해 허용하거나 거부할 작업을 지정합니다. 클라이언트가 Amazon EC2 인스턴스를 사용하는 경우 권한 부여 정책을 해당 Amazon EC2 인스턴스의 IAM 역할에 연결합니다. 또는 명명된 프로필을 사용하도록 클라이언트를 구성한 다음 권한 부여 정책을 해당 명명된 프로필의 역할과 연결할 수 있습니다. IAM 액세스 제어를 위한 클라이언트 구성에서는 명명된 프로필을 사용하도록 클라이언트를 구성하는 방법에 대해 설명합니다.

IAM 정책을 만드는 방법에 대한 자세한 내용은 IAM 정책 생성을 참조하세요.

다음은 MyTestCluster라는 클러스터에 대한 권한 부여 정책의 예제입니다. `Action` 및 `Resource` 요소의 의미를 이해하려면 IAM 권한 부여 정책 작업 및 리소스의 의미를 참조하세요.

> **Important**
>
> IAM 정책에 대한 변경 사항은 IAM API 및 AWS CLI에 즉시 반영됩니다. 그러나 정책 변경이 적용되려면 상당한 시간이 소요될 수 있습니다. 대부분 정책 변경은 1분 이내에 적용됩니다. 네트워크 상태에 따라 지연 시간이 늘어날 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:AlterCluster",
                "kafka-cluster:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:111122223333:cluster/MyTestCluster/
abcd1234-0123-abcd-5678-1234abcd-1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:*Topic*",
                "kafka-cluster:WriteData",
                "kafka-cluster:ReadData"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:topic/MyTestCluster/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:group/MyTestCluster/*"
            ]
        }
    ]
}
```

데이터 생산 및 소비와 같은 일반적인 Apache Kafka 사용 사례에 해당하는 조치 요소가 포함된 정책을 생성하는 방법을 알아보려면 [클라이언트 권한 부여 정책의 일반적인 사용 사례](#)를 참조하세요.

Kafka 버전 2.8.0 이상에서는 WriteDataIdempotently 권한이 더 이상 사용되지 않습니다([KIP-679](#)). 기본적으로 `enable.idempotence = true`가 설정되어 있습니다. 따라서 Kafka 버전 2.8.0 이상의 경우 IAM은 Kafka ACL과 동일한 기능을 제공하지 않습니다. 해당 주제에 대한 `WriteData` 액세스 권한만 제공해서는 주제에 `WriteDataIdempotently`를 수행할 수 없습니다. 이는 모든 주제에 `WriteData`가 제공되는 경우에는 영향을 미치지 않습니다. 이 경우 `WriteDataIdempotently`가 허용됩니다. 이는 IAM 로직의 구현 방식과 Kafka ACL의 구현 방식에 차이가 있기 때문입니다. 또한 주제에 멱등적으로 쓰려면 `transactional-ids`에 대한 액세스 권한도 필요합니다.

이 문제를 해결하려면 다음 정책과 유사한 정책을 사용하는 것을 권장합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:AlterCluster",
                "kafka-cluster:DescribeCluster",
                "kafka-cluster:WriteDataIdempotently"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:cluster/MyTestCluster/
abcd1234-0123-abcd-5678-1234abcd-1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:*Topic*",
                "kafka-cluster:WriteData",
                "kafka-cluster:ReadData"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:topic/MyTestCluster/
abcd1234-0123-abcd-5678-1234abcd-1/TestTopic",
```

```
                    "arn:aws:kafka:us-east-1:123456789012:transactional-id/
MyTestCluster/abcd1234-0123-abcd-5678-1234abcd-1/*"
               ]
          }
     ]
}
```

이 경우 `WriteData`는 `TestTopic`에 대한 쓰기를 허용하고 `WriteDataIdempotently`은 클러스터에 대한 멱등성 쓰기를 허용합니다. 또한 이 정책은 필요한 `transactional-id` 리소스에 대한 액세스 권한을 추가합니다.

`WriteDataIdempotently`는 클러스터 수준 권한이므로 주제 수준에서 사용할 수 없습니다. `WriteDataIdempotently`가 주제 수준으로 제한되어 있으면 해당 정책이 작동하지 않습니다.

IAM 액세스 제어를 위한 부트스트랩 브로커 가져오기

Amazon MSK 클러스터를 위한 부트스트랩 브로커 가져오기을(를) 참조하세요.

IAM 권한 부여 정책 작업 및 리소스의 의미

현재 Amazon MSK에 대한 IAM 액세스 제어는 Kafka에 대한 내부 클러스터 작업을 지원하지 않습니다. 여기에는 Kafka가 트랜잭션을 종료하는 데 사용하는 WriteTxnMarkers API가 포함됩니다. 트랜잭션을 종료하려면 IAM 인증 대신 적절한 ACL과 함께 SCRAM 또는 mTLS 인증을 사용하는 것이 좋습니다.

이 섹션에서는 IAM 권한 부여 정책에서 사용할 수 있는 작업 및 리소스 요소의 의미에 대해 설명합니다. 정책 예제는 IAM 역할에 대한 권한 부여 정책 생성을 참조하세요.

권한 부여 정책 작업

다음 표에는 Amazon MSK를 위한 IAM 액세스 제어를 사용할 때 권한 부여 정책에 포함할 수 있는 작업이 나열되어 있습니다. 권한 부여 정책에 표의 작업 열에 있는 작업을 포함할 때는 필수 작업 열에 있는 해당 작업도 포함해야 합니다.

작업	설명	필수 작업	필수 리소스	서버리스 클러스터에 적용 가능
`kafka-cluster:Connect`	클러스터에 연결하고 인증할 수	없음	클러스터	예

작업	설명	필수 작업	필수 리소스	서버리스 클러스터에 적용 가능
	있는 권한을 부여합니다.			
kafka-cluster:DescribeCluster	클러스터의 다양한 측면을 설명할 수 있는 권한을 부여하며, 이는 Apache Kafka의 DESCRIBE CLUSTER ACL과 동일합니다.	kafka-cluster:Connect	클러스터	예
kafka-cluster:AlterCluster	클러스터의 다양한 측면을 변경할 수 있는 권한을 부여하며, 이는 Apache Kafka의 ALTER CLUSTER ACL과 동일합니다.	kafka-cluster:Connect kafka-cluster:DescribeCluster	클러스터	아니요
kafka-cluster:DescribeClusterDynamicConfiguration	클러스터의 동적 구성을 설명할 수 있는 권한을 부여하며, 이는 Apache Kafka의 DESCRIBE_CONFIGS CLUSTER ACL과 동일합니다.	kafka-cluster:Connect	클러스터	아니요

작업	설명	필수 작업	필수 리소스	서버리스 클러스터에 적용 가능
`kafka-cluster:AlterClusterDynamicConfiguration`	클러스터의 동적 구성을 변경할 수 있는 권한을 부여하며, 이는 Apache Kafka의 ALTER_CONFIGS CLUSTER ACL과 동일합니다.	`kafka-cluster:Connect` `kafka-cluster:DescribeClusterDynamicConfiguration`	클러스터	아니요
`kafka-cluster:WriteDataIdempotently`	클러스터에서 데이터를 멱등적으로 쓸 수 있는 권한을 부여하며, 이는 Apache Kafka의 IDEMPOTENT_WRITE CLUSTER ACL과 동일합니다.	`kafka-cluster:Connect` `kafka-cluster:WriteData`	클러스터	예
`kafka-cluster:CreateTopic`	클러스터에 주제를 생성할 수 있는 권한을 부여하며, 이는 Apache Kafka의 CREATE CLUSTER/TOPIC ACL과 동일합니다.	`kafka-cluster:Connect`	주제	예

작업	설명	필수 작업	필수 리소스	서버리스 클러스터에 적용 가능
`kafka-cluster:DescribeTopic`	클러스터의 주제를 설명할 수 있는 권한을 부여하며, 이는 Apache Kafka의 DESCRIBE TOPIC ACL과 동일합니다.	`kafka-cluster:Connect`	주제	예
`kafka-cluster:AlterTopic`	클러스터의 주제를 변경할 수 있는 권한을 부여하며, 이는 Apache Kafka의 ALTER TOPIC ACL과 동일합니다.	`kafka-cluster:Connect` `kafka-cluster:DescribeTopic`	주제	예
`kafka-cluster:DeleteTopic`	클러스터에서 주제를 삭제할 수 있는 권한을 부여하며, 이는 Apache Kafka의 DELETE TOPIC ACL과 동일합니다.	`kafka-cluster:Connect` `kafka-cluster:DescribeTopic`	주제	예

작업	설명	필수 작업	필수 리소스	서버리스 클러스터에 적용 가능
`kafka-cluster:DescribeTopicDynamicConfiguration`	클러스터에서 주제의 동적 구성을 설명할 수 있는 권한을 부여하며, 이는 Apache Kafka의 DESCRIBE_CONFIGS TOPIC ACL과 동일합니다.	`kafka-cluster:Connect`	주제	예
`kafka-cluster:AlterTopicDynamicConfiguration`	클러스터에서 주제의 동적 구성을 변경할 수 있는 권한을 부여하며, 이는 Apache Kafka의 ALTER_CONFIGS TOPIC ACL과 동일합니다.	`kafka-cluster:Connect` `kafka-cluster:DescribeTopicDynamicConfiguration`	주제	예

`kafka-cluster:ReadData`	클러스터의 토픽에서 데이터를 읽을 수 있는 권한을 부여하며, 이는 Apache Kafka의 READ TOPIC ACL과 동일합니다.	`kafka-cluster:Connect`
		`kafka-cluster:DescribeTopic`
		`kafka-cluster:AlterGroup`
			주제	예

작업	설명	필수 작업	필수 리소스	서버리스 클러스터에 적용 가능
kafka-cluster:WriteData	Apache Kafka의 WRITE TOPIC ACL에 해당하는 클러스터에서 주제에 데이터를 쓸 수 있는 권한을 부여합니다.	kafka-cluster:Connect kafka-cluster:DescribeTopic	주제	예
kafka-cluster:DescribeGroup	클러스터에서 그룹을 설명할 수 있는 권한을 부여하며, 이는 Apache Kafka의 DESCRIBE GROUP ACL과 동일합니다.	kafka-cluster:Connect	그룹	예
kafka-cluster:AlterGroup	클러스터의 그룹에 참여할 수 있는 권한을 부여하며, 이는 Apache Kafka의 READ GROUP ACL과 동일합니다.	kafka-cluster:Connect kafka-cluster:DescribeGroup	그룹	예
kafka-cluster:DeleteGroup	클러스터에서 그룹을 삭제할 수 있는 권한을 부여하며, 이는 Apache Kafka의 DELETE GROUP ACL과 동일합니다.	kafka-cluster:Connect kafka-cluster:DescribeGroup	그룹	예

작업	설명	필수 작업	필수 리소스	서버리스 클러스터에 적용 가능
kafka-cluster:DescribeTransactionalId	클러스터에서 트랜잭션 ID를 설명할 수 있는 권한을 부여하며, 이는 Apache Kafka의 DESCRIBE TRANSACTIONAL_ID ACL과 동일합니다.	kafka-cluster:Connect	transactional-id	예
kafka-cluster:AlterTransactionalId	클러스터의 트랜잭션 ID를 변경할 수 있는 권한을 부여하며, 이는 Apache Kafka의 WRITE TRANSACTIONAL_ID ACL과 동일합니다.	kafka-cluster:Connect kafka-cluster:DescribeTransactionalId kafka-cluster:WriteData	transactional-id	예

콜론 뒤에 오는 작업에서 별표(*) 와일드카드를 여러 번 사용할 수 있습니다. 예를 들면 다음과 같습니다.

- `kafka-cluster:*Topic`은 `kafka-cluster:CreateTopic`, `kafka-cluster:DescribeTopic`, `kafka-cluster:AlterTopic`, `kafka-cluster:DeleteTopic`을 나타냅니다. `kafka-cluster:DescribeTopicDynamicConfiguration` 또는 `kafka-cluster:AlterTopicDynamicConfiguration`은 포함되지 않습니다.
- `kafka-cluster:*`는 모든 권한을 나타냅니다.

권한 부여 정책 리소스

다음 표에는 Amazon MSK를 위한 IAM 액세스 제어를 사용할 때 권한 부여 정책에 사용할 수 있는 4가지 유형의 리소스를 보여줍니다. 에서 또는 DescribeCluster API AWS Management 콘솔또는 describe-clusterAWS CLI 명령을 사용하여 클러스터 Amazon 리소스 이름(ARN)을 가져올 수 있습니다. 그런 다음 클러스터 ARN을 사용하여 주제, 그룹, 트랜잭션 ID ARN을 구성할 수 있습니다. 권한 부여 정책에서 리소스를 지정하려면 해당 리소스의 ARN을 사용합니다.

Resource	ARN 형식
Cluster	arn:aws:kafka:*region*:*account-id* :cluster/*cluster-name* /*cluster-uuid*
Topic	arn:aws:kafka:*region*:*account-id* :topic/*cluster-name* /*cluster-uuid* /*topic-name*
Group	arn:aws:kafka:*region*:*account-id* :group/*cluster-name* /*cluster-uuid* /*group-name*
트랜잭션 ID	arn:aws:kafka:*region*:*account-id* :transactional-id/*cluster-name* /*cluster-uuid* /*transactional-id*

별표(*) 와일드카드는 ARN의 `:cluster/`, `:topic/`, `:group/`, `:transactional-id/` 뒤에 오는 부분 어디에서나 여러 번 사용할 수 있습니다. 다음은 별표(*) 와일드카드를 사용하여 여러 리소스를 참조하는 방법에 대한 몇 가지 예입니다.

- `arn:aws:kafka:us-east-1:0123456789012:topic/MyTestCluster/*`: 클러스터의 UUID에 관계없이 MyTestCluster라는 이름의 모든 클러스터에 있는 모든 주제입니다.
- `arn:aws:kafka:us-east-1:0123456789012:topic/MyTestCluster/abcd1234-0123-abcd-5678-1234abcd-1/*_test`: 이름이 MyTestCluster이고 UUID가 abcd1234-0123-abcd-5678-1234abcd-1인 클러스터에서 이름이 "_test"로 끝나는 모든 주제입니다.
- `arn:aws:kafka:us-east-1:0123456789012:transactional-id/MyTestCluster/*/5555abcd-1111-abcd-1234-abcd1234-1`: 계정에 있는 MyTestCluster라는 클러스터의 모든 구현에서 트랜잭션 ID가 5555abcd-1111-abcd-1234-abcd1234-1인 모든 트랜잭션입니다. 즉, MyTestCluster라는 이름의 클러스터를 생성한 다음 삭제한 다음 같은 이름의 다른 클러스터를 생성하는 경우 이 리소스 ARN을 사용하여 두 클러스터에서 동일한 트랜잭션 ID를 나타낼 수 있습니다. 그러나 삭제된 클러스터는 액세스할 수 없습니다.

클라이언트 권한 부여 정책의 일반적인 사용 사례

다음 표의 첫 번째 열에는 몇 가지 일반적인 사용 사례가 나와 있습니다. 클라이언트가 특정 사용 사례를 수행하도록 권한을 부여하려면 클라이언트의 권한 부여 정책에 해당 사용 사례에 필요한 작업을 포함하고 `Effect`를 `Allow`로 설정합니다.

Amazon MSK에 대한 IAM 액세스 제어의 일부인 모든 작업에 대한 자세한 내용은 [IAM 권한 부여 정책 작업 및 리소스의 의미](#) 섹션을 참조하세요.

> **Note**
>
> 기본적으로 작업이 거부됩니다. 클라이언트가 수행할 수 있도록 권한을 부여하려는 모든 작업을 명시적으로 허용해야 합니다.

사용 사례:	필수 작업
관리자	`kafka-cluster:*`
주제 생성	`kafka-cluster:Connect` `kafka-cluster:CreateTopic`

데이터 생산	`kafka-cluster:Connect`
	`kafka-cluster:DescribeTopic`
	`kafka-cluster:WriteData`

데이터 소비	`kafka-cluster:Connect`
	`kafka-cluster:DescribeTopic`
	`kafka-cluster:DescribeGroup`
	`kafka-cluster:AlterGroup`
	`kafka-cluster:ReadData`

멱등적으로 데이터 생산	`kafka-cluster:Connect`

사용 사례:	필수 작업
	kafka-cluster:DescribeTopic kafka-cluster:WriteData kafka-cluster:WriteDataIdempotently
트랜잭션 방식으로 데이터 생산	kafka-cluster:Connect kafka-cluster:DescribeTopic kafka-cluster:WriteData kafka-cluster:DescribeTransactionalId kafka-cluster:AlterTransactionalId
클러스터 구성 설명	kafka-cluster:Connect kafka-cluster:DescribeClusterDynamicConfiguration
클러스터의 구성 업데이트	kafka-cluster:Connect kafka-cluster:DescribeClusterDynamicConfiguration kafka-cluster:AlterClusterDynamicConfiguration
주제 구성 설명	kafka-cluster:Connect kafka-cluster:DescribeTopicDynamicConfiguration

사용 사례:	필수 작업
주제의 구성 업데이트	kafka-cluster:Connect kafka-cluster:DescribeTopicDynamicConfiguration kafka-cluster:AlterTopicDynamicConfiguration
주제 변경	kafka-cluster:Connect kafka-cluster:DescribeTopic kafka-cluster:AlterTopic

Amazon MSK에 대한 상호 TLS 클라이언트 인증

애플리케이션에서 Amazon MSK 브로커로의 연결을 위해 TLS를 사용하여 클라이언트 인증을 활성화할 수 있습니다. 클라이언트 인증을 사용하려면 AWS Private CA가 필요합니다. 는 클러스터AWS 계정와 동일한 또는 다른 계정에 있을 AWS Private CA수 있습니다. AWS Private CA에 대한 자세한 내용은 생성 및 관리를AWS Private CA 참조하세요.

Amazon MSK는 인증서 해지 목록(CRL)을 지원하지 않습니다. 클러스터 주제에 대한 액세스를 제어하거나 손상된 인증서를 차단하려면 Apache Kafka ACLs 및 AWS보안 그룹을 사용합니다. Apache Kafka ACL 사용에 대한 자세한 내용은 the section called “Apache Kafka ACL” 섹션을 참조하세요.

이 주제는 다음 섹션을 포함하고 있습니다.

- 클라이언트 인증을 지원하는 Amazon MSK 클러스터 생성
- 인증을 사용하도록 클라이언트 설정
- 인증을 사용하여 메시지 생성 및 사용

클라이언트 인증을 지원하는 Amazon MSK 클러스터 생성

이 절차에서는를 사용하여 클라이언트 인증을 활성화하는 방법을 보여줍니다AWS Private CA.

> **Note**
>
> 상호 TLS를 사용하여 액세스를 제어할 때는 각 MSK 클러스터AWS Private CA에 대해 독립적인를 사용하는 것이 좋습니다. 이렇게 하면 PCA가 서명한 TLS 인증서는 단일 MSK 클러스터에서만 인증됩니다.

1. 다음 콘텐츠를 가진 `clientauthinfo.json`이라는 파일을 생성합니다: *`Private-CA-ARN`*을 PCA의 ARN으로 바꿉니다.

```
{
   "Tls": {
       "CertificateAuthorityArnList": ["Private-CA-ARN"]
    }
}
```

2. the section called “를 사용하여 프로비저닝된 Amazon MSK 클러스터 생성AWS CLI”에 설명된 대로 `brokernodegroupinfo.json` 파일을 생성합니다.

3. 클라이언트 인증을 사용하려면 클라이언트와 브로커 간 전송 중 암호화를 활성화해야 합니다. 다음 콘텐츠를 가진 `encryptioninfo.json`이라는 파일을 생성합니다: *`KMS-Key-ARN`*을 KMS 키의 ARN으로 바꿉니다. `ClientBroker`를 `TLS` 또는 `TLS_PLAINTEXT`로 설정할 수 있습니다.

```
{
   "EncryptionAtRest": {
       "DataVolumeKMSKeyId": "KMS-Key-ARN"
    },
   "EncryptionInTransit": {
        "InCluster": true,
        "ClientBroker": "TLS"
    }
}
```

   암호화에 대한 자세한 내용은 the section called “Amazon MSK 암호화” 섹션을 참조하세요.

4. 가 AWS CLI설치된 시스템에서 다음 명령을 실행하여 인증 및 전송 중 암호화가 활성화된 클러스터를 생성합니다. 응답에 제공된 클러스터 ARN을 저장합니다.

```
aws kafka create-cluster --cluster-name "AuthenticationTest" --broker-node-group-
info file://brokernodegroupinfo.json --encryption-info file://encryptioninfo.json
```

```
--client-authentication file://clientauthinfo.json --kafka-version "{YOUR KAFKA
VERSION}" --number-of-broker-nodes 3
```

인증을 사용하도록 클라이언트 설정

이 프로세스는 인증을 사용할 클라이언트로 사용하도록 Amazon EC2 인스턴스를 설정하는 방법을 설명합니다.

이 프로세스는 클라이언트 머신을 생성하고, 주제를 생성하고, 필요한 보안 설정을 구성하여 인증을 사용해서 메시지를 생성하고 소비하는 방법을 설명합니다.

1. 클라이언트 머신으로 사용할 Amazon EC2 인스턴스를 생성합니다. 간단히 하기 위해 클러스터에 사용한 것과 동일한 VPC에 이 인스턴스를 생성합니다. 이러한 클라이언트 머신을 생성하는 방법에 대한 예제는 the section called “클라이언트 머신 생성” 단원을 참조하십시오.
2. 주제를 생성합니다. 예를 들어, the section called “주제 생성” 단원의 지침을 참조하십시오.
3. 가 AWS CLI설치된 시스템에서 다음 명령을 실행하여 클러스터의 부트스트랩 브로커를 가져옵니다. *Cluster-ARN*을 클러스터의 ARN으로 바꿉니다.

   ```
   aws kafka get-bootstrap-brokers --cluster-arn Cluster-ARN
   ```

   응답에서 `BootstrapBrokerStringTls`에 연결된 문자열을 저장합니다.
4. 클라이언트 머신에서 다음 명령을 실행하여 JVM 트러스트 스토어를 사용하여 클라이언트 트러스트 스토어를 만듭니다. JVM 경로가 다른 경우 그에 따라 명령을 조정하십시오.

   ```
   cp /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/lib/security/
   cacerts kafka.client.truststore.jks
   ```
5. 클라이언트 머신에서 다음 명령을 실행하여 클라이언트에 대한 프라이빗 키를 만듭니다. *Distinguished-Name*, *Example-Alias*, *Your-Store-Pass*, *Your-Key-Pass*를 원하는 문자열로 바꿉니다.

   ```
   keytool -genkey -keystore kafka.client.keystore.jks -validity 300 -storepass Your-
   Store-Pass -keypass Your-Key-Pass -dname "CN=Distinguished-Name" -alias Example-
   Alias -storetype pkcs12 -keyalg rsa
   ```
6. 클라이언트 머신에서 다음 명령을 실행하여 이전 단계에서 만든 프라이빗 키로 인증서 요청을 만듭니다.

```
keytool -keystore kafka.client.keystore.jks -certreq -file client-cert-sign-request
 -alias Example-Alias -storepass Your-Store-Pass -keypass Your-Key-Pass
```

7. `client-cert-sign-request` 파일을 열고, `-----BEGIN CERTIFICATE REQUEST-----`로 시작해 `-----END CERTIFICATE REQUEST-----`로 끝나는지 확인합니다. `-----BEGIN NEW CERTIFICATE REQUEST-----`로 시작하는 경우, 파일의 시작 부분과 끝 부분에서 단어 `NEW` 및 그 뒤의 단일 공백을 삭제합니다.

8. 가 AWS CLI설치된 시스템에서 다음 명령을 실행하여 인증서 요청에 서명합니다. *Private-CA-ARN*을 PCA의 ARN으로 바꿉니다. 원하는 경우 유효성 값을 변경할 수 있습니다. 여기에서는 300을 사용합니다.

```
aws acm-pca issue-certificate --certificate-authority-arn Private-CA-ARN --csr
 fileb://client-cert-sign-request --signing-algorithm "SHA256WITHRSA" --validity
 Value=300,Type="DAYS"
```

응답에 제공된 인증서 ARN을 저장합니다.

> **Note**
>
> 클라이언트 인증서를 검색하려면 `acm-pca get-certificate` 명령을 사용하고 사용자 인증서 ARN을 지정합니다. 자세한 내용은 AWS CLI 명령 참조에서 [get-certificate]를 참조하세요.

9. 다음 명령을 실행하여가 자동으로 AWS Private CA서명한 인증서를 가져옵니다. *Certificate-ARN*을 이전 명령에 대한 응답에서 얻은 ARN으로 바꿉니다.

```
aws acm-pca get-certificate --certificate-authority-arn Private-CA-ARN --
certificate-arn Certificate-ARN
```

10. 이전 명령을 실행한 JSON 결과에서 `Certificate` 및 `CertificateChain`에 연결된 문자열을 복사합니다. 이 두 문자열을 signed-certificate-from-acm이라는 새 파일에 붙여 넣습니다. 우선 `Certificate`에 연결된 문자열을 붙여 넣은 다음, `CertificateChain`와 연결된 문자열을 붙여 넣습니다. \n 문자를 새 줄로 바꿉니다. 다음은 인증서 및 인증서 체인을 붙여 넣은 이후의 파일 구조입니다.

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

11. 클라이언트 머신에서 다음 명령을 실행하여 MSK 브로커와 통신할 때 제공할 수 있도록 키 스토어에 이 인증서를 추가합니다.

```
keytool -keystore kafka.client.keystore.jks -import -file signed-certificate-from-acm -alias Example-Alias -storepass Your-Store-Pass -keypass Your-Key-Pass
```

12. 다음 콘텐츠를 가진 `client.properties`이라는 파일을 생성합니다: 트러스트 스토어 및 키 스토어 위치를 `kafka.client.truststore.jks`를 저장한 경로로 조정합니다. *{YOUR KAFKA VERSION}* 자리 표시자를 Kafka 클라이언트 버전으로 대체합니다.

```
security.protocol=SSL
ssl.truststore.location=/tmp/kafka_2.12-{YOUR KAFKA VERSION}/kafka.client.truststore.jks
ssl.keystore.location=/tmp/kafka_2.12-{YOUR KAFKA VERSION}/kafka.client.keystore.jks
ssl.keystore.password=Your-Store-Pass
ssl.key.password=Your-Key-Pass
```

인증을 사용하여 메시지 생성 및 사용

이 프로세스는 인증을 사용하여 메시지를 생성하고 사용하는 방법을 설명합니다.

1. 다음 명령을 실행해 주제를 생성합니다. `client.properties`라는 파일은 이전 절차에서 생성한 파일입니다.

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --bootstrap-server BootstrapBroker-String --replication-factor 3 --partitions 1 --topic ExampleTopic --command-config client.properties
```

2. 콘솔 생산자를 시작하려면 다음 명령을 실행합니다. `client.properties`라는 파일은 이전 절차에서 생성한 파일입니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --bootstrap-
server BootstrapBroker-String --topic ExampleTopic --producer.config
 client.properties
```

3. 클라이언트 머신의 새 명령 창에서 다음 명령을 실행하여 콘솔 소비자를 시작합니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
server BootstrapBroker-String --topic ExampleTopic --consumer.config
 client.properties
```

4. 생산자 창에 메시지를 입력하고 소비자 창에 표시되는지 확인합니다.

AWSSecrets Manager를 사용한 로그인 보안 인증

AWSSecrets Manager를 사용하여 저장되고 보호되는 로그인 자격 증명을 사용하여 Amazon MSK 클러스터에 대한 액세스를 제어할 수 있습니다. Secrets Manager에 사용자 보안 인증 정보를 저장하면 보안 인증 정보 감사, 업데이트, 교체와 같은 클러스터 인증의 오버헤드가 줄어듭니다. 또한 Secrets Manager를 사용하면 클러스터 간에 사용자 보안 인증 정보를 공유할 수 있습니다.

보안 암호를 MSK 클러스터와 연결하면 MSK는 자격 증명 데이터를 주기적으로 동기화합니다.

이 주제는 다음 섹션을 포함하고 있습니다.

- 로그인 자격 증명 인증의 작동 방식
- Amazon MSK 클러스터를 위해 SASL/SCRAM 인증 설정
- 사용자 작업
- SCRAM 비밀 사용 시 제한 사항

로그인 자격 증명 인증의 작동 방식

Amazon MSK의 로그인 보안 인증 정보 인증은 SASL/SCRAM(Simple Authentication and Security Layer/Salted Challenge Response Mechanism) 인증을 사용합니다. 클러스터에 대한 로그인 보안 인증 정보 인증을 설정하려면 AWS Secrets Manager에서 보안 암호 리소스를 생성하고 로그인 보안 인증 정보를 해당 보안 암호에 연결합니다.

SASL/SCRAM은 RFC 5802에 정의되어 있습니다. SCRAM은 보안 해싱 알고리즘을 사용하며 클라이언트와 서버 간에 일반 텍스트 로그인 보안 인증 정보를 전송하지 않습니다.

> **Note**
>
> 클러스터에 대해 SASL/SCRAM 인증을 설정하면 Amazon MSK는 클라이언트와 브로커 간의 모든 트래픽에 대해 TLS 암호화를 설정합니다.

Amazon MSK 클러스터를 위해 SASL/SCRAM 인증 설정

AWSSecrets Manager에서 보안 암호를 설정하려면 Secrets Manager 사용 설명서의 [보안 암호 생성 및 검색](#) 자습서를 따르세요. [AWS](#)

Amazon MSK 클러스터에 대한 보안 암호를 생성하는 경우에는 다음 요구 사항에 유의하세요.

- 암호 유형으로 다른 유형의 보안 암호(예: API 키)를 선택합니다.
- 보안 암호 이름은 접두사 AmazonMSK_로 시작해야 합니다.
- 기존 사용자 지정 AWS KMS키를 사용하거나 보안 암호에 대한 새 사용자 지정 AWS KMS키를 생성해야 합니다. Secrets Manager는 기본적으로 보안 암호에 기본 AWS KMS키를 사용합니다.

> **Important**
>
> 기본 AWS KMS키로 생성된 보안 암호는 Amazon MSK 클러스터에서 사용할 수 없습니다.

- 일반 텍스트 옵션을 사용하여 키-값 페어를 입력하려면 로그인 보안 인증 정보 데이터가 다음 형식이어야 합니다.

```
{
  "username": "alice",
  "password": "alice-secret"
}
```

- 보안 암호에 대한 Amazon 리소스 이름(ARN) 값
- > **Important**
  >
  > [the section called “ 클러스터 크기를 적절하게 조정: Standard 브로커당 파티션 수”](#)에 설명된 제한을 초과하는 클러스터에는 Secrets Manager 보안 암호를 연결할 수 없습니다.
- AWS CLI를 사용하여 보안 암호를 생성하는 경우 `kms-key-id` 파라미터의 키 ID 또는 ARN을 지정합니다. 별칭을 지정하지 마세요.

- 보안 암호를 클러스터에 연결하려면 Amazon MSK 콘솔 또는 [BatchAssociateScramSecret](#) 작업을 사용합니다.

> ⚠ Important
>
> 암호를 클러스터와 연결하면 Amazon MSK는 클러스터가 정의한 보안 암호 값에 액세스하고 읽을 수 있도록 허용하는 리소스 정책을 암호에 연결합니다. 이 리소스 정책을 수정해서는 안 됩니다. 이렇게 하면 클러스터가 보안 암호에 액세스하는 것을 방지할 수 있습니다. 보안 암호 리소스 정책 및/또는 보안 암호 암호화에 사용되는 KMS 키를 변경하는 경우 보안 암호를 MSK 클러스터에 다시 연결해야 합니다. 이를 통해 클러스터가 보안 암호에 계속 액세스할 수 있습니다.

다음 `BatchAssociateScramSecret` 작업의 예제 JSON 입력은 보안 암호를 클러스터와 연결합니다.

```
{
  "clusterArn" : "arn:aws:kafka:us-west-2:0123456789019:cluster/SalesCluster/
abcd1234-abcd-cafe-abab-9876543210ab-4",
  "secretArnList": [
    "arn:aws:secretsmanager:us-west-2:0123456789019:secret:AmazonMSK_MyClusterSecret"
  ]
}
```

로그인 보안 인증 정보를 사용하여 클러스터에 연결

보안 암호를 생성하고 클러스터에 연결하면 클라이언트를 클러스터에 연결할 수 있습니다. 다음 절차에서는 SASL/SCRAM 인증을 사용하는 클러스터에 클라이언트를 연결하는 방법을 보여줍니다. 또한 예제 주제에서 이를 생성하고 소비하는 방법을 보여줍니다.

주제

- [SASL/SCRAM 인증을 사용하여 클러스터에 클라이언트 연결](#)
- [연결 문제 해결](#)

SASL/SCRAM 인증을 사용하여 클러스터에 클라이언트 연결

1. 가 AWS CLI설치된 시스템에서 다음 명령을 실행합니다. *clusterARN*을 클러스터의 ARN으로 바꿉니다.

```
aws kafka get-bootstrap-brokers --cluster-arn clusterARN
```

이 명령의 JSON 결과에서 BootstrapBrokerStringSaslScram이라는 문자열과 연관된 값을 저장합니다. 이후 단계에서 이 값을 사용합니다.

2. 클라이언트 머신에서 암호에 저장된 사용자 보안 인증 정보가 포함된 JAAS 구성 파일을 생성합니다. 예를 들어 사용자 alice에 대해 다음과 같은 내용으로 users_jaas.conf라는 파일을 생성합니다.

```
KafkaClient {
   org.apache.kafka.common.security.scram.ScramLoginModule required
   username="alice"
   password="alice-secret";
};
```

3. 다음 명령을 사용하여 JAAS 구성 파일을 KAFKA_OPTS 환경 파라미터로 내보냅니다.

```
export KAFKA_OPTS=-Djava.security.auth.login.config=<path-to-jaas-file>/
users_jaas.conf
```

4. /tmp 디렉터리에 kafka.client.truststore.jks라는 파일을 생성합니다.

5. (선택 사항) 다음 명령을 사용하여 JVM cacerts 폴더에 이전 단계에서 생성한 kafka.client.truststore.jks 파일로 JDK 키 저장소 파일을 복사합니다. *JDKFolder*를 인스턴스의 JDK 폴더 이름으로 변경합니다. 예를 들어 JDK 폴더의 이름은 java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64일 수 있습니다.

```
cp /usr/lib/jvm/JDKFolder/lib/security/cacerts /tmp/kafka.client.truststore.jks
```

6. Apache Kafka 설치의 bin 디렉터리에 다음 내용으로 client_sasl.properties라는 클라이언트 속성 파일을 생성합니다. 이 파일은 SASL 메커니즘과 프로토콜을 정의합니다.

```
security.protocol=SASL_SSL
sasl.mechanism=SCRAM-SHA-512
```

7. 예제 주제를 생성하려면 다음 명령을 실행합니다. *BootstrapBrokerStringSaslScram*을 이 주제의 1단계에서 얻은 부트스트랩 브로커 문자열로 바꿉니다.

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --bootstrap-
server BootstrapBrokerStringSaslScram --command-config <path-to-client-
```

```
properties>/client_sasl.properties --replication-factor 3 --partitions 1 --topic
 ExampleTopicName
```

8. 생성한 예제 주제로 생성하려면 클라이언트 머신에서 다음 명령을 실행합니다. *BootstrapBrokerStringSaslScram*을 이 주제의 1단계에서 검색한 부트스트랩 브로커 문자열로 바꿉니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-
list BootstrapBrokerStringSaslScram --topic ExampleTopicName --producer.config
 client_sasl.properties
```

9. 생성한 주제에서 사용하려면 클라이언트 머신에서 다음 명령을 실행합니다. *BootstrapBrokerStringSaslScram*을 이 주제의 1단계에서 얻은 부트스트랩 브로커 문자열로 바꿉니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
server BootstrapBrokerStringSaslScram --topic ExampleTopicName --from-beginning --
consumer.config client_sasl.properties
```

연결 문제 해결

Kafka 클라이언트 명령을 실행하는 경우, 특히 대규모 주제 또는 데이터세트로 작업할 때 Java 힙 메모리 오류가 발생할 수 있습니다. 이러한 오류는 Kafka 도구가 워크로드에 충분하지 않을 수 있는 기본 메모리 설정이 있는 Java 애플리케이션으로 실행되기 때문에 발생합니다.

`Out of Memory Java Heap` 오류를 해결하려면 메모리 설정을 포함하도록 KAFKA_OPTS 환경 변수를 수정하여 Java 힙 크기를 늘릴 수 있습니다.

다음 예제에서는 최대 힙 크기를 1GB(-Xmx1G)로 설정합니다. 사용 가능한 시스템 메모리 및 요구 사항에 따라 이 값을 조정할 수 있습니다.

```
export KAFKA_OPTS="-Djava.security.auth.login.config=<path-to-jaas-file>/
users_jaas.conf -Xmx1G"
```

대규모 주제를 사용하는 경우 메모리 사용량을 제한하는 `--from-beginning` 대신 시간 기반 또는 오프셋 기반 파라미터를 사용하는 것이 좋습니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
server BootstrapBrokerStringSaslScram --topic ExampleTopicName --max-messages 1000 --
consumer.config client_sasl.properties
```

사용자 작업

사용자 생성: 보안 암호에 키-값 페어로 사용자를 생성합니다. Secrets Manager 콘솔에서 일반 텍스트 옵션을 사용하는 경우 로그인 보안 인증 정보 데이터를 다음 형식으로 지정해야 합니다.

```
{
  "username": "alice",
  "password": "alice-secret"
}
```

사용자 액세스 취소: 클러스터에 액세스하기 위한 사용자의 보안 인증 정보를 취소하려면 먼저 클러스터에서 ACL을 제거하거나 적용한 다음 보안 암호 연결을 해제하는 것을 권장합니다. 이는 다음과 같은 이유 때문입니다.

- 사용자를 제거해도 기존 연결은 닫히지 않습니다.
- 보안 암호에 대한 변경 사항이 전파되는 데에는 최대 10분이 소요됩니다.

Amazon MSK에서 ACL을 사용하는 방법에 대한 자세한 내용은 Apache Kafka ACL 섹션을 참조하세요.

ZooKeeper 모드를 사용하는 클러스터의 경우 사용자가 ACL을 수정하지 못하도록 ZooKeeper 노드에 대한 액세스를 제한하는 것이 좋습니다. 자세한 내용은 Amazon MSK 클러스터의 Apache ZooKeeper 노드에 대한 액세스 제어 단원을 참조하십시오.

SCRAM 비밀 사용 시 제한 사항

SCRAM 보안 암호를 사용할 때는 다음 제한 사항에 유의하세요.

- Amazon MSK는 SCRAM-SHA-512 인증만 지원합니다.
- Amazon MSK 클러스터는 최대 1,000명의 사용자를 보유할 수 있습니다.
- 보안 암호와 AWS KMS key함께를 사용해야 합니다. 기본 시크릿 관리자 암호화 키를 사용하는 보안 암호는 Amazon MSK와 함께 사용할 수 없습니다. KMS 키 생성에 대한 자세한 내용은 대칭 암호화 KMS 키 생성을 참조하세요.
- Secrets Manager에서는 비대칭 KMS 키를 사용할 수 없습니다.
- BatchAssociateScramSecret 작업을 사용하여 한 번에 최대 10개의 보안 암호를 클러스터에 연결할 수 있습니다.
- Amazon MSK 클러스터와 연결된 보안 암호의 이름에는 접두사 AmazonMSK_가 있어야 합니다.

- Amazon MSK 클러스터와 연결된 보안 암호는 클러스터와 동일한 Amazon Web Services 계정 및 AWS리전에 있어야 합니다.

Apache Kafka ACL

Apache Kafka에는 플러그형 권한 부여자가 있으며, 기본 제공 권한 부여자 구현이 함께 제공됩니다. Amazon MSK는 브로커의 `server.properties` 파일에서 이 권한 부여자를 활성화합니다.

Apache Kafka ACL의 형식은 "Principal P is [Allowed/Denied] Operation O From Host H on any Resource R matching ResourcePattern RP"입니다. RP가 특정 리소스 R과 일치하지 않으면 R에 연결된 ACL이 없으므로 수퍼유저 이외의 누구도 R에 액세스할 수 없습니다. 이 Apache Kafka 동작을 변경하려면 속성 `allow.everyone.if.no.acl.found`를 true로 설정합니다. Amazon MSK는 이 속성을 기본적으로 true로 설정합니다. 즉, Amazon MSK 클러스터를 사용할 때 리소스에 ACL을 명시적으로 설정하지 않으면 모든 보안 주체가 이 리소스에 액세스할 수 있습니다. 리소스에 대해 ACL을 활성화하면 권한이 부여된 보안 주체만 ACL에 액세스할 수 있습니다. TLS 상호 인증을 사용하여 주제에 대한 액세스를 제한하고 클라이언트에 권한을 부여하려면 Apache Kafka 권한 부여자 CLI를 사용하여 ACL을 추가합니다. ACL 추가, 제거 및 나열 방법에 대한 자세한 내용은 Kafka 인증 명령줄 인터페이스를 참조하십시오.

Amazon MSK는 브로커를 슈퍼 사용자로 구성하므로 모든 주제에 액세스할 수 있습니다. 이를 통해 브로커가 클러스터 구성에 `allow.everyone.if.no.acl.found` 속성이 정의되어 있는지 여부에 관계없이 기본 파티션에서 메시지를 복제할 수 있습니다.

주제에 대한 읽기 및 쓰기 액세스 권한을 추가하거나 제거하려면

1. 브로커가 ACL이 있는 모든 주제에서 읽을 수 있도록 ACL 테이블에 브로커를 추가합니다. 브로커에게 주제에 대한 읽기 액세스 권한을 부여하려면 MSK 클러스터와 통신할 수 있는 클라이언트 머신에서 다음 명령을 실행합니다.

   *`Distinguished-Name`*을 해당 클러스터의 부트스트랩 브로커의 DNS로 바꾼 다음 이 고유 이름의 첫 번째 마침표 앞에 있는 문자열을 별표(*)로 바꿉니다. 예를 들어, 클러스터의 부트스트랩 브로커 중 하나에 DNS `b-6.mytestcluster.67281x.c4.kafka.us-east-1.amazonaws.com`이 있는 경우 다음 명령의 *`Distinguished-Name`*을 `*.mytestcluster.67281x.c4.kafka.us-east-1.amazonaws.com`으로 바꿉니다. 부트스트랩 브로커를 가져오는 방법에 대한 자세한 내용은 the section called “부트스트랩 브로커 가져오기” 단원을 참조하십시오.

```
<path-to-your-kafka-installation>/bin/kafka-acls.sh --bootstrap-server
 BootstrapServerString --add --allow-principal "User:CN=Distinguished-Name" --
operation Read --group=* --topic Topic-Name
```

2. 주제에 대한 클라이언트 애플리케이션 읽기 액세스 권한을 부여하려면 클라이언트 머신에서 다음 명령을 실행합니다. 상호 TLS 인증을 사용하는 경우 프라이빗 키를 생성할 때 사용한 것과 동일한 *Distinguished-Name*을 사용합니다.

```
<path-to-your-kafka-installation>/bin/kafka-acls.sh --bootstrap-server
 BootstrapServerString --add --allow-principal "User:CN=Distinguished-Name" --
operation Read --group=* --topic Topic-Name
```

   읽기 액세스 권한을 제거하려면 `--add`를 `--remove`로 바꾸어 같은 명령을 실행하면 됩니다.

3. 주제에 대한 쓰기 액세스 권한을 부여하려면 클라이언트 머신에서 다음 명령을 실행합니다. 상호 TLS 인증을 사용하는 경우 프라이빗 키를 생성할 때 사용한 것과 동일한 *Distinguished-Name*을 사용합니다.

```
<path-to-your-kafka-installation>/bin/kafka-acls.sh --bootstrap-server
 BootstrapServerString --add --allow-principal "User:CN=Distinguished-Name" --
operation Write --topic Topic-Name
```

   쓰기 액세스 권한을 제거하려면 `--add`를 `--remove`로 바꾸어 같은 명령을 실행하면 됩니다.

Amazon MSK 클러스터의 보안 그룹 변경

이 페이지에서는 기존 MSK 클러스터의 보안 그룹을 변경하는 방법을 설명합니다. 특정 사용자 세트에 액세스 권한을 제공하거나 클러스터에 대한 액세스를 제한하기 위해 클러스터의 보안 그룹을 변경해야 할 수도 있습니다. 보안 그룹에 대한 자세한 내용은 Amazon VPC 사용 설명서에서 [VPC의 보안 그룹](#)을 참조하세요.

1. [ListNodes](#) API 또는의 [list-nodes](#) 명령을 사용하여 클러스터의 브로커 목록을 AWS CLI가져옵니다. 이 작업의 결과에는 브로커와 연결된 탄력적 네트워크 인터페이스(ENI)의 ID가 포함됩니다.
2. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
3. 화면 오른쪽 상단에 있는 드롭다운 목록을 사용하여 클러스터가 배포된 리전을 선택합니다.
4. 왼쪽 창의 네트워크 및 보안에서 네트워크 인터페이스를 선택합니다.

5. 첫 번째 단계에서 가져온 첫 번째 ENI를 선택합니다. 화면 상단의 작업 메뉴를 선택한 다음 보안 그룹 변경을 선택합니다. 해당 ENI에 새 보안 그룹을 할당합니다. 첫 번째 단계에서 가져온 각 ENI에 대해 이 단계를 반복합니다.

> **Note**
>
> Amazon EC2 콘솔을 사용하여 클러스터의 보안 그룹에 대한 변경 사항은 네트워크 설정의 MSK 콘솔에 반영되지 않습니다.

6. 새 보안 그룹의 규칙을 구성하여 고객이 브로커에 액세스할 수 있도록 합니다. 보안 그룹 규칙 설정에 대한 자세한 내용은 Amazon VPC 사용 설명서에서 규칙 추가, 제거, 업데이트를 참조하세요.

> **Important**
>
> 클러스터의 브로커와 연결된 보안 그룹을 변경한 다음 해당 클러스터에 새 브로커를 추가하면 Amazon MSK는 새 브로커를 클러스터가 생성될 때 클러스터와 연결된 원래 보안 그룹에 연결합니다. 그러나 클러스터가 올바르게 작동하려면 모든 브로커가 동일한 보안 그룹에 연결되어 있어야 합니다. 따라서 보안 그룹을 변경한 후 새 브로커를 추가하는 경우 이전 절차를 다시 수행하여 새 브로커의 ENI를 업데이트해야 합니다.

Amazon MSK 클러스터의 Apache ZooKeeper 노드에 대한 액세스 제어

보안상의 이유로 Amazon MSK 클러스터의 일부인 Apache ZooKeeper 노드에 대한 액세스를 제한할 수 있습니다. 노드에 대한 액세스를 제한하기 위해 별도의 보안 그룹을 할당할 수 있습니다. 그런 다음 해당 보안 그룹에 대한 액세스 권한을 결정할 수 있습니다.

> **Important**
>
> 이 섹션은 KRaft 모드에서 실행되는 클러스터에는 적용되지 않습니다. the section called “KRaft 모드 ”을(를) 참조하세요.

이 주제는 다음 섹션을 포함하고 있습니다.

- Apache ZooKeeper 노드를 별도의 보안 그룹에 배치하려면
- Apache ZooKeeper에서 TLS 보안 사용

Apache ZooKeeper 노드를 별도의 보안 그룹에 배치하려면

Apache ZooKeeper 노드에 대한 액세스를 제한하기 위해 별도의 보안 그룹을 할당할 수 있습니다. 보안 그룹 규칙을 설정하여 이 새 보안 그룹에 액세스할 수 있는 사용자를 선택할 수 있습니다.

1. 클러스터에 대한 Apache ZooKeeper 연결 문자열을 가져옵니다. 자세한 방법은 the section called "ZooKeeper 모드"을 참조하세요. 이 연결 문자열은 Apache ZooKeeper 노드의 DNS 이름을 포함합니다.
2. `host` 또는 `ping` 같은 도구를 사용하여 이전 단계에서 얻은 DNS 이름을 IP 주소로 변환합니다. 이 절차의 뒷부분에 필요하므로 이 IP 주소를 저장하십시오.
3. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
4. 탐색 창의 NETWORK & SECURITY(네트워크 및 보안) 아래에서 Network Interfaces(네트워크 인터페이스)를 선택합니다.
5. 네트워크 인터페이스 테이블 위의 검색 필드에 클러스터 이름을 입력한 다음 return을 입력합니다. 이렇게 하면 테이블에 표시되는 네트워크 인터페이스 수가 클러스터와 연결된 인터페이스로 제한됩니다.
6. 목록의 첫 번째 네트워크 인터페이스에 해당하는 행의 시작 부분에 있는 확인란을 선택합니다.
7. 페이지 하단의 세부 정보 창에서 주 프라이빗 IPv4 IP를 찾습니다. 이 IP 주소가 이 절차의 첫 번째 단계에서 얻은 IP 주소 중 하나와 일치하면 이 네트워크 인터페이스가 클러스터의 일부인 Apache ZooKeeper 노드에 할당된다는 뜻입니다. 그렇지 않으면 이 네트워크 인터페이스 옆의 확인란 선택을 취소하고, 목록에서 다음 네트워크 인터페이스를 선택합니다. 네트워크 인터페이스를 선택하는 순서는 중요하지 않습니다. 다음 단계에는 Apache ZooKeeper 노드에 할당된 모든 네트워크 인터페이스에서 일일이 동일한 작업을 수행합니다.
8. Apache ZooKeeper 노드에 해당하는 네트워크 인터페이스를 선택하는 경우 페이지 상단의 작업 메뉴를 선택한 다음, 보안 그룹 변경을 선택합니다. 이 네트워크 인터페이스에 새 보안 그룹을 할당합니다. 보안 그룹 생성에 대한 자세한 내용은 Amazon VPC 설명서에서 보안 그룹 생성을 참조하세요.
9. 이전 단계를 반복하여 클러스터의 Apache ZooKeeper 노드와 연결된 모든 네트워크 인터페이스에 동일한 새 보안 그룹을 할당합니다.
10. 이제 이 새 보안 그룹에 액세스할 수 있는 사용자를 선택할 수 있습니다. 보안 그룹 규칙 설정에 대한 자세한 내용은 Amazon VPC 설명서에서 규칙 추가, 제거, 업데이트를 참조하세요.

Apache ZooKeeper에서 TLS 보안 사용

클라이언트와 Apache ZooKeeper 노드 간의 전송 시 암호화에 TLS 보안을 사용할 수 있습니다. Apache ZooKeeper 노드로 TLS 보안을 구현하려면 다음을 수행합니다.

- Apache ZooKeeper에서 TLS 보안을 사용하려면 클러스터는 Apache Kafka 버전 2.5.1 이상을 사용해야 합니다.
- 클러스터를 생성하거나 구성할 때 TLS 보안을 활성화합니다. TLS가 활성화된 Apache Kafka 버전 2.5.1 이상에서 생성된 클러스터는 Apache ZooKeeper 엔드포인트에서 자동으로 TLS 보안을 사용합니다. TLS 보안 설정에 대한 자세한 내용은 Amazon MSK 암호화 시작하기 섹션을 참조하세요.
- DescribeCluster 작업을 사용하여 TLS Apache ZooKeeper 엔드포인트를 검색합니다.
- `kafka-configs.sh` 및 `kafka-acls.sh` 도구 또는 ZooKeeper 셸과 함께 사용할 Apache ZooKeeper 구성 파일을 생성합니다. 각 도구에서 `--zk-tls-config-file` 파라미터를 사용하여 Apache ZooKeeper 구성을 지정합니다.

  다음 예제는 일반적인 Apache ZooKeeper 구성 파일을 보여줍니다.

```
zookeeper.ssl.client.enable=true
zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
zookeeper.ssl.keystore.location=kafka.jks
zookeeper.ssl.keystore.password=test1234
zookeeper.ssl.truststore.location=truststore.jks
zookeeper.ssl.truststore.password=test1234
```

- 다른 명령(예: `kafka-topics`)의 경우 `KAFKA_OPTS` 환경 변수를 사용하여 Apache ZooKeeper 파라미터를 구성해야 합니다. 다음 예제는 Apache ZooKeeper 파라미터를 다른 명령에 전달하도록 `KAFKA_OPTS` 환경 변수를 구성하는 방법을 보여줍니다.

```
export KAFKA_OPTS="
-Dzookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
-Dzookeeper.client.secure=true
-Dzookeeper.ssl.trustStore.location=/home/ec2-user/kafka.client.truststore.jks
-Dzookeeper.ssl.trustStore.password=changeit"
```

  `KAFKA_OPTS` 환경 변수를 구성한 후에는 CLI 명령을 정상적으로 사용할 수 있습니다. 다음 예제는 `KAFKA_OPTS` 환경 변수의 Apache ZooKeeper 구성을 사용하여 Apache Kafka 주제를 생성합니다.

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --
zookeeper ZooKeeperTLSConnectString --replication-factor 3 --partitions 1 --topic
 AWSKafkaTutorialTopic
```

> **Note**
>
> Apache ZooKeeper 구성 파일에서 사용하는 파라미터 KAFKA_OPTS의 이름과 환경 변수에서 사용하는 파라미터의 이름이 일치하지 않습니다. 구성 파일 및 KAFKA_OPTS 환경 변수에서 어떤 이름을 어떤 파라미터와 함께 사용하는지에 대해 주의하세요.

TLS로 Apache ZooKeeper 노드에 액세스하는 방법에 대한 자세한 내용은 KIP-515: ZK 클라이언트에서 새로운 TLS 지원 인증 활성화를 참조하세요.

Amazon Managed Streaming for Apache Kafka에 대한 규정 준수 검증

타사 감사자는 AWS 규정 준수 프로그램의 일환으로 Amazon Managed Streaming for Apache Kafka의 보안 및 규정 준수 여부를 평가합니다. 여기에는 PCI 및 HIPAA BAA가 포함됩니다.

특정 규정 준수 프로그램 범위의 AWS서비스 목록은 규정 준수 프로그램 제공 범위 내 Amazon 서비스 규정 준수 프로그램 제공 . 일반 정보는 AWS규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다AWS Artifact. 자세한 내용은 Downloading Reports inDownloadingAWS Artifact을 참조하세요.

Amazon MSK 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다.는 규정 준수를 지원하기 위해 다음 리소스를 AWS제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서: 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수를 위한 설계 백서 -이 백서에서는 기업이 AWS를 사용하여 HIPAA 준수 애플리케이션을 생성하는 방법을 설명합니다.
- AWS규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- AWS Config개발자 안내서의 규칙을 사용하여 리소스 평가 -이 AWS Config서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub CSPM -이 AWS서비스는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 AWS되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

Amazon Managed Streaming for Apache Kafka의 복원력

AWS글로벌 인프라는 AWS리전 및 가용 영역을 중심으로 구축됩니다. AWS리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS리전 및 가용 영역에 대한 자세한 내용은 AWS글로벌 인프라를 참조하세요.

Amazon Managed Streaming for Apache Kafka의 인프라 보안

관리형 서비스인 Amazon Managed Streaming for Apache Kafka는 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS글로벌 네트워크 보안 절차로 보호됩니다.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 Amazon MSK에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 시크릿 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

Amazon MSK 로깅

Apache Kafka 브로커 로그를 Amazon CloudWatch Logs, Amazon S3, Amazon Data Firehose 등의 대상 유형 중 하나 이상에 전달할 수 있습니다. 를 사용하여 Amazon MSK API 호출을 로깅할 수도 있습니다AWS CloudTrail.

> **Note**
>
> Express 브로커에서는 브로커 로그를 사용할 수 없습니다.

브로커 로그

브로커 로그를 통해 Apache Kafka 애플리케이션의 문제를 해결하고 MSK 클러스터와의 통신을 분석할 수 있습니다. INFO 수준 브로커 로그를 CloudWatch 로그 그룹, S3 버킷, Firehose 전송 스트림 등의

대상 리소스 유형 중 하나 이상에 전송하도록 신규 또는 기존 MSK 클러스터를 구성할 수 있습니다. 그런 다음 Firehose를 통해 전송 스트림의 로그 데이터를 OpenSearch Service로 전송할 수 있습니다.

클러스터에 브로커 로그를 전달하도록 클러스터를 구성하기 전에 대상 리소스를 생성해야 합니다. Amazon MSK는 이러한 대상 리소스가 아직 존재하지 않는 경우 이를 생성하지 않습니다. 이러한 세 가지 유형의 대상 리소스와 이를 생성하는 방법에 대한 자세한 내용은 다음 설명서를 참조하십시오.

- Amazon CloudWatch Logs
- Amazon S3
- Amazon Data Firehose

필수 권한

Amazon MSK 브로커 로그의 대상을 구성하려면 Amazon MSK 작업에 사용하는 IAM ID에 AWS관리형 정책: AmazonMSKFullAccess 정책에 설명된 권한이 있어야 합니다.

브로커 로그를 S3 버킷으로 스트리밍하려면 `s3:PutBucketPolicy` 권한도 필요합니다. S3 버킷 정책에 대한 자세한 내용은 Amazon S3 사용 설명서에서 S3 버킷 정책을 추가하려면 어떻게 해야 하나요?를 참조하세요. IAM 정책 전반에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리를 참조하세요.

SSE-KMS 버킷과 함께 사용하기 위한 필수 KMS 키 정책

고객 관리형 키와 함께 AWS KMS관리형 키(SSE-KMS)를 사용하여 S3 버킷에 대한 서버 측 암호화를 활성화한 경우 Amazon MSK가 버킷에 브로커 파일을 쓸 수 있도록 KMS 키의 키 정책에 다음을 추가합니다.

```
{
  "Sid": "Allow Amazon MSK to use the key.",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "delivery.logs.amazonaws.com"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
```

```
  ],
  "Resource": "*"
}
```

를 사용하여 브로커 로그 구성AWS Management 콘솔

새 클러스터를 생성하는 경우 모니터링 섹션에서 브로커 로그 전달 제목을 찾습니다. Amazon MSK에서 브로커 로그를 전달할 대상을 지정할 수 있습니다.

기존 클러스터의 경우 클러스터 목록에서 클러스터를 선택한 다음 속성 탭을 선택합니다. 모니터링 섹션까지 아래로 스크롤한 다음 편집 버튼을 선택합니다. Amazon MSK에서 브로커 로그를 전달할 대상을 지정할 수 있습니다.

를 사용하여 브로커 로그 구성AWS CLI

`create-cluster` 또는 `update-monitoring` 명령을 사용하면 선택적으로 `logging-info` 파라미터를 지정하고 다음 예제와 같이 JSON 구조를 전달할 수 있습니다. 이 JSON에서 세 가지 대상 유형은 모두 선택 사항입니다.

> **Note**
>
> 로그 전송을 설정하려면 Firehose 스트림에서 `LogDeliveryEnabled` 태그를 `true`로 설정해야 합니다. CloudWatch 로그에 대해가 AWS생성하는 서비스 연결 역할은이 태그를 사용하여 모든 Firehose 전송 스트림에 대한 권한을 부여합니다. 이 태그를 제거하면 서비스 연결 역할이 Firehose 스트림에 로그를 전송할 수 없습니다. 서비스 연결 역할에 포함된 권한을 보여주는 IAM 정책의 예를 보려면 Amazon CloudWatch 사용 설명서의 [리소스 권한에 사용되는 IAM 역할](#)을 참조하세요.

```
{
  "BrokerLogs": {
    "S3": {
      "Bucket": "amzn-s3-demo-bucket",
      "Prefix": "ExamplePrefix",
      "Enabled": true
    },
    "Firehose": {
      "DeliveryStream": "ExampleDeliveryStreamName",
      "Enabled": true
    },
    "CloudWatchLogs": {
```

```
            "Enabled": true,
            "LogGroup": "ExampleLogGroupName"
        }
    }
}
```

API를 사용하여 브로커 로그 구성

CreateCluster 또는 UpdateMonitoring 작업에 전달하는 JSON에 선택적 `loggingInfo` 구조를 지정할 수 있습니다.

> **Note**
>
> 기본적으로 브로커 로깅을 활성화하면 Amazon MSK는 `INFO` 수준 로그를 지정된 대상에 기록합니다. 그러나 Apache Kafka 2.4.X 이상 사용자는 브로커 로그 수준을 log4j log 수준 중 어느 것으로든 동적으로 설정할 수 있습니다. 브로커 로그 수준을 동적으로 설정하는 방법에 대한 자세한 내용은 KIP-412: 동적 애플리케이션 로그 수준을 지원하도록 관리자 API 확장을 참조하세요. 로그 수준을 `DEBUG` 또는 `TRACE`로 동적으로 설정하는 경우 Amazon S3 또는 Firehose를 로그 대상으로 사용하는 것이 좋습니다. CloudWatch Logs를 로그 대상으로 사용하고 `DEBUG` 또는 `TRACE` 수준 로깅을 동적으로 활성화하는 경우 Amazon MSK는 로그 샘플을 지속적으로 전달할 수 있습니다. 이는 브로커 성능에 상당한 영향을 미칠 수 있으므로 `INFO` 로그 수준이 문제의 근본 원인을 파악하기에 충분히 상세하지 않은 경우에만 사용해야 합니다.

를 사용하여 API 호출 로깅AWS CloudTrail

> **Note**
>
> AWS CloudTrail로그는를 사용하는 경우에만 Amazon MSK에 사용할 수 있습니다IAM 액세스 제어.

Amazon MSK는 Amazon MSK에서 사용자AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS서비스와 통합됩니다. CloudTrail은 에 대한 API 직접 호출을 이벤트로 캡처합니다. 캡처된 호출에는 Amazon MSK 콘솔에서의 호출과 Amazon MSK API 작업에 대한 코드 호출이 포함되어 있습니다. 또한 주제 및 그룹 생성 및 변경과 같은 Apache Kafka 작업을 캡처합니다.

트레일을 만들면 Amazon MSK용 이벤트를 포함한 CloudTrail 이벤트를 Amazon S3 버킷에 지속적으로 전달할 수 있습니다. 추적을 구성하지 않은 경우에도 이벤트 기록에서 CloudTrail 콘솔의 최신 이벤

트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon MSK 또는 Apache Kafka 작업으로 이루어진 요청, 요청이 이루어진 IP 주소, 요청을 한 사람, 요청이 이루어진 시기, 추가 세부 정보를 확인할 수 있습니다.

구성 및 활성화 방법을 포함하여 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용자 안내서를 참조하세요.

CloudTrail의 Amazon MSK 정보

CloudTrail은 계정 생성 시 Amazon Web Services 계정에서 활성화됩니다. 지원되는 이벤트 활동이 MSK 클러스터에서 발생하면 해당 활동은 이벤트 기록의 다른 AWS서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. Amazon Web Services 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 설명은 CloudTrail 이벤트 기록으로 이벤트 보기를 참조하세요.

Amazon MSK에 대한 이벤트를 포함하여 Amazon Web Services 계정의 이벤트에 대한 지속적인 레코드를 보려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 Region에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 지역의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석하고 처리하도록 다른 Amazon 서비스를 구성할 수도 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 MSK는 모든 Amazon MSK 작업을 CloudTrail 로그 파일에 이벤트로 기록합니다. 또한 다음과 같은 Apache Kafka 작업을 기록합니다.

- kafka-cluster:DescribeClusterDynamicConfiguration
- kafka-cluster:AlterClusterDynamicConfiguration
- kafka-cluster:CreateTopic
- kafka-cluster:DescribeTopicDynamicConfiguration
- kafka-cluster:AlterTopic
- kafka-cluster:AlterTopicDynamicConfiguration
- kafka-cluster:DeleteTopic

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 사용자 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 AWS서비스에서 이루어졌는지 여부입니다.

자세한 설명은 [CloudTrail userIdentity 요소](#)를 참조하세요.

예제: Amazon MSK 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 호출과 Apache Kafka 작업의 스택 기록이 정렬되어 있지 않으므로 특정 순서로 표시되지 않습니다.

다음 예제는 `DescribeCluster` 및 `DeleteCluster` Amazon MSK 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEF0123456789ABCDE",
        "arn": "arn:aws:iam::012345678901:user/Joe",
        "accountId": "012345678901",
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "userName": "Joe"
      },
      "eventTime": "2018-12-12T02:29:24Z",
      "eventSource": "kafka.amazonaws.com",
      "eventName": "DescribeCluster",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "aws-cli/1.14.67 Python/3.6.0 Windows/10 botocore/1.9.20",
      "requestParameters": {
```

```
        "clusterArn": "arn%3Aaws%3Akafka%3Aus-east-1%3A012345678901%3Acluster
%2Fexamplecluster%2F01234567-abcd-0123-abcd-abcd0123efa-2"
      },
      "responseElements": null,
      "requestID": "bd83f636-fdb5-abcd-0123-157e2fbf2bde",
      "eventID": "60052aba-0123-4511-bcde-3e18dbd42aa4",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "recipientAccountId": "012345678901"
    },
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEF0123456789ABCDE",
        "arn": "arn:aws:iam::012345678901:user/Joe",
        "accountId": "012345678901",
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "userName": "Joe"
      },
      "eventTime": "2018-12-12T02:29:40Z",
      "eventSource": "kafka.amazonaws.com",
      "eventName": "DeleteCluster",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "aws-cli/1.14.67 Python/3.6.0 Windows/10 botocore/1.9.20",
      "requestParameters": {
        "clusterArn": "arn%3Aaws%3Akafka%3Aus-east-1%3A012345678901%3Acluster
%2Fexamplecluster%2F01234567-abcd-0123-abcd-abcd0123efa-2"
      },
      "responseElements": {
        "clusterArn": "arn:aws:kafka:us-east-1:012345678901:cluster/
examplecluster/01234567-abcd-0123-abcd-abcd0123efa-2",
        "state": "DELETING"
      },
      "requestID": "c6bfb3f7-abcd-0123-afa5-293519897703",
      "eventID": "8a7f1fcf-0123-abcd-9bdb-1ebf0663a75c",
      "readOnly": false,
      "eventType": "AwsApiCall",
      "recipientAccountId": "012345678901"
    }
  ]
}
```

다음은 `kafka-cluster:CreateTopic` 작업을 설명하는 CloudTrail 로그 항목을 보여 주는 예시입니다.


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "ABCDEFGH1IJKLMN2P34Q5",
    "arn": "arn:aws:iam::111122223333:user/Admin",
    "accountId": "111122223333",
    "accessKeyId": "CDEFAB1C2UUUUU3AB4TT",
    "userName": "Admin"
  },
  "eventTime": "2021-03-01T12:51:19Z",
  "eventSource": "kafka-cluster.amazonaws.com",
  "eventName": "CreateTopic",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "198.51.100.0/24",
  "userAgent": "aws-msk-iam-auth/unknown-version/aws-internal/3 aws-sdk-java/1.11.970
 Linux/4.14.214-160.339.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/25.272-b10 java/1.8.0_272
 scala/2.12.8 vendor/Red_Hat,_Inc.",
  "requestParameters": {
    "kafkaAPI": "CreateTopics",
    "resourceARN": "arn:aws:kafka:us-east-1:111122223333:topic/IamAuthCluster/3ebafd8e-
dae9-440d-85db-4ef52679674d-1/Topic9"
  },
  "responseElements": null,
  "requestID": "e7c5e49f-6aac-4c9a-a1d1-c2c46599f5e4",
  "eventID": "be1f93fd-4f14-4634-ab02-b5a79cb833d2",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333"
}
```


메타데이터 관리

Amazon MSK는 Apache ZooKeeper 또는 KRaft 메타데이터 관리 모드를 지원합니다.

Amazon MSK의 Apache Kafka 버전 3.7.x에서 ZooKeeper 모드 대신 KRaft 모드를 사용하는 클러스터를 생성할 수 있습니다. KRaft 기반 클러스터는 Kafka 내의 컨트롤러를 사용하여 메타데이터를 관리합니다.

주제

- ZooKeeper 모드
- KRaft 모드

ZooKeeper 모드

Apache ZooKeeper는 구성 정보 유지, 명명, 분산 동기화 제공 및 그룹 서비스 제공을 위한 중앙화된 서비스입니다. 이러한 모든 종류의 서비스는 Apache Kafka를 비롯한 분산 애플리케이션에서 몇몇 또는 다른 형태로 사용됩니다.

클러스터에서 ZooKeeper 모드를 사용하는 경우 아래 단계를 사용하여 Apache ZooKeeper 연결 문자열을 가져올 수 있습니다. 하지만 Kafka 2.5에서는 `--zookeeper` 플래그가 더 이상 사용되지 않고 Kafka 3.0에서는 제거되므로 `BootstrapServerString`을 사용하여 클러스터에 연결하고 관리자 작업을 수행하는 것이 좋습니다.

를 사용하여 Apache ZooKeeper 연결 문자열 가져오기AWS Management 콘솔

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 표에 이 계정에 속한 현재 리전의 모든 클러스터가 나열됩니다. 설명을 보려면 클러스터의 이름을 선택합니다.
3. 클러스터 요약 페이지에서 클라이언트 정보 보기를 선택합니다. 여기에는 부트스트랩 브로커와 Apache ZooKeeper 연결 문자열이 표시됩니다.

AWS CLI를 사용하여 Apache ZooKeeper 연결 문자열 가져오기

1. 클러스터의 Amazon 리소스 이름(ARN)을 모르는 경우, 계정의 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.
2. 클러스터에 대한 다른 정보와 함께 Apache ZooKeeper 연결 문자열을 가져오려면 *ClusterArn*을 클러스터의 ARN으로 바꾸어 다음 명령을 실행합니다.

```
aws kafka describe-cluster --cluster-arn ClusterArn
```

이 `describe-cluster` 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "ClusterInfo": {
        "BrokerNodeGroupInfo": {
            "BrokerAZDistribution": "DEFAULT",
            "ClientSubnets": [
                "subnet-0123456789abcdef0",
                "subnet-2468013579abcdef1",
                "subnet-1357902468abcdef2"
            ],
            "InstanceType": "kafka.m5.large",
            "StorageInfo": {
                "EbsStorageInfo": {
                    "VolumeSize": 1000
                }
            }
        },
        "ClusterArn": "arn:aws:kafka:us-east-1:111122223333:cluster/
testcluster/12345678-abcd-4567-2345-abcdef123456-2",
        "ClusterName": "testcluster",
        "CreationTime": "2018-12-02T17:38:36.75Z",
        "CurrentBrokerSoftwareInfo": {
            "KafkaVersion": "2.2.1"
        },
        "CurrentVersion": "K13V1IB3VIYZZH",
        "EncryptionInfo": {
            "EncryptionAtRest": {
                "DataVolumeKMSKeyId": "arn:aws:kms:us-
east-1:555555555555:key/12345678-abcd-2345-ef01-abcdef123456"
            }
        },
        "EnhancedMonitoring": "DEFAULT",
        "NumberOfBrokerNodes": 3,
        "State": "ACTIVE",
        "ZookeeperConnectString": "10.0.1.101:2018,10.0.2.101:2018,10.0.3.101:2018"
    }
}
```

이전 JSON 예제는 `describe-cluster` 명령 출력에 있는 `ZookeeperConnectString` 키를 보여줍니다. 이 키에 해당하는 값을 복사하고, 클러스터에서 주제를 생성해야 할 때를 대비해 저장하십시오.

> **Important**
>
> Apache ZooKeeper 연결 문자열을 가져오려면 Amazon MSK 클러스터가 `ACTIVE` 상태여야 합니다. 클러스터가 여전히 `CREATING` 상태에 있으면 `describe-cluster` 명령의 출력에 `ZookeeperConnectString`이 포함되지 않습니다 이 경우, 몇 분 정도 기다린 다음 클러스터가 `ACTIVE` 상태에 도달한 후 `describe-cluster`를 다시 실행합니다.

API를 사용하여 Apache ZooKeeper 연결 문자열 가져오기

API를 사용하여 Apache ZooKeeper 연결 문자열을 가져오려면 DescribeCluster를 참조하십시오.

KRaft 모드

Amazon MSK는 Kafka 버전 3.7.x에서 KRaft(Apache Kafka Raft)에 대한 지원을 도입했습니다. Apache Kafka 커뮤니티는 Apache Kafka 클러스터의 메타데이터 관리를 위해 Apache ZooKeeper를 대체하도록 KRaft를 개발했습니다. KRaft 모드에서 클러스터 메타데이터는 ZooKeeper 노드 대신 Kafka 클러스터의 일부인 Kafka 컨트롤러 그룹 내에서 전파됩니다. KRaft 컨트롤러는 추가 비용 없이 포함되어 있으므로 추가 설정이나 관리가 필요하지 않습니다. KRaft에 대한 자세한 내용은 KIP-500을 참조하세요.

다음은 MSK의 KRaft 모드에 대해 유의해야 할 몇 가지 사항입니다.

- KRaft 모드는 새로운 클러스터에만 사용할 수 있습니다. 클러스터가 생성된 후에는 메타데이터 모드를 전환할 수 없습니다.
- MSK 콘솔에서 Kafka 버전 3.7.x를 선택하고 클러스터 생성 창에서 KRaft 확인란을 선택하여 Kraft 기반 클러스터를 생성할 수 있습니다.
- MSK API `CreateCluster` 또는 `CreateClusterV2` 작업을 사용하여 KRaft 모드에서 클러스터를 생성하려면 `3.7.x.kraft`를 버전으로 사용해야 합니다. `3.7.x`를 버전으로 사용하여 ZooKeeper 모드에서 클러스터를 생성합니다.
- 브로커당 파티션 수는 KRaft 및 ZooKeeper 기반 클러스터에서 동일합니다. 하지만 KRaft를 사용하면 클러스터에 더 많은 브로커를 프로비저닝하여 클러스터당 더 많은 파티션을 호스팅할 수 있습니다.
- Amazon MSK에서 KRaft 모드를 사용하기 위해 API를 변경할 필요는 없습니다. 그러나 클라이언트에서 지금도 여전히 `--zookeeper` 연결 문자열을 사용하는 경우 `--bootstrap-server` 연결 문자열을 사용하여 클러스터에 연결하도록 클라이언트를 업데이트해야 합니다. `--zookeeper` 플래그는 Apache Kafka 버전 2.5에서 더 이상 사용되지 않으며 Kafka 버전 3.0부터 제거된다는 점

에 유의하세요. 따라서 클러스터에 대한 모든 연결에는 최신 Apache Kafka 클라이언트 버전과 `--bootstrap-server` 연결 문자열을 사용하는 것이 좋습니다.

- ZooKeeper 모드는 Apache Kafka에서 ZooKeeper도 지원하는 모든 릴리스 버전에서 계속 사용할 수 있습니다. Apache Kafka 버전 및 향후 업데이트에 대한 지원 종료에 대한 자세한 내용은 지원되는 Apache Kafka 버전 단원을 참조하세요.
- 사용하는 모든 도구에서 ZooKeeper 연결 없이 Kafka Admin API를 사용할 수 있는지 확인해야 합니다. 클러스터를 Cruise Control에 연결하는 업데이트된 단계는 Amazon MSK에서 Apache Kafka용 LinkedIn의 Cruise Control 사용 단원을 참조하세요. Cruise Control에는 ZooKeeper 없이 Cruise Control을 실행하는 방법도 있습니다.
- 관리 작업을 위해 클러스터의 KRaft 컨트롤러에 직접 액세스할 필요는 없습니다. 그러나 오픈 모니터링을 사용하여 지표를 수집하는 경우 클러스터에 대한 일부 비컨트롤러 관련 지표를 수집하려면 컨트롤러의 DNS 엔드포인트도 필요합니다. MSK 콘솔에서 또는 ListNodes API 작업을 사용하여 이러한 DNS 엔드포인트를 가져올 수 있습니다. KRaft 기반 클러스터에 대한 오픈 모니터링 설정을 위한 업데이트된 단계는 Prometheus를 사용하여 MSK 프로비저닝 클러스터 모니터링 단원을 참조하세요.
- ZooKeeper 모드 클러스터를 통해 KRaft 모드 클러스터를 모니터링하는 데 필요한 추가 CloudWatch 지표는 없습니다. MSK는 클러스터에 사용되는 KRaft 컨트롤러를 관리합니다.
- `--bootstrap-server` 연결 문자열을 사용하여 KRaft 모드 클러스터에서 ACL을 계속 관리할 수 있습니다. `--zookeeper` 연결 문자열을 사용하여 ACL을 관리해서는 안 됩니다. Apache Kafka ACL을(를) 참조하세요.
- KRaft 모드에서 클러스터의 메타데이터는 외부 ZooKeeper 노드가 아닌 Kafka 내의 KRaft 컨트롤러에 저장됩니다. 따라서 ZooKeeper 노드와 마찬가지로 컨트롤러 노드에 대한 액세스를 별도로 제어할 필요는 없습니다.

주제 작업

Amazon MSK APIs 사용하여 MSK 프로비저닝 클러스터의 주제에 대한 정보를 볼 수 있습니다. 이러한 APIs 파티션 수, 복제 인자, 구성 및 파티션 세부 정보를 포함한 주제 메타데이터에 대한 읽기 전용 액세스를 제공합니다. 이 정보는 Kafka 주제의 구조를 모니터링, 문제 해결 및 이해하는 데 유용합니다.

> **Important**
>
> `ListTopics`, `DescribeTopic`및 `DescribeTopicPartitions` APIs 응답에는 약 1분마다 업데이트되는 데이터가 반영됩니다. 변경 후 최신 주제 상태의 경우 쿼리하기 전에 약 1분 정도 기다려야 합니다.

> **Note**
>
> 이러한 APIs 주제 메타데이터에 대한 읽기 전용 액세스를 제공합니다. 주제를 생성하거나 수정하려면 Apache Kafka 도구 또는 Kafka AdminClient를 사용합니다. 자세한 내용은 4단계: Amazon MSK 클러스터에서 주제 생성 단원을 참조하십시오.

주제 정보 보기 요구 사항

- 클러스터는 MSK 프로비저닝 클러스터여야 합니다. MSK Serverless 클러스터에는 이러한 APIs를 사용할 수 없습니다.
- 클러스터가 Apache Kafka 버전 3.6.0 이상을 실행 중이어야 합니다. 지원되는 버전에 대한 자세한 내용은 섹션을 참조하세요the section called “지원되는 Apache Kafka 버전”.
- 클러스터가 `ACTIVE` 상태여야 합니다. 클러스터 상태에 대한 자세한 내용은 MSK 프로비저닝된 클러스터의 상태 이해 섹션을 참조하세요.
- 적절한 IAM 권한이 있어야 합니다. 자세한 내용은 the section called “IAM 권한” 단원을 참조하십시오.

주제 정보를 볼 수 있는 IAM 권한

이러한 APIs 호출하려면 적절한 IAM 권한이 있어야 합니다. 다음 표에는 각 API에 필요한 권한이 나열되어 있습니다.

주제 정보를 보는 데 필요한 권한

API	필수 권한	Resource
ListTopics	`kafka-cluster:Connect` `kafka-cluster:DescribeTopic`	클러스터 ARN

DescribeTopic	`kafka-cluster:Connect`
	`kafka-cluster:DescribeTopic`
	`kafka-cluster:DescribeTopicDynamicConfiguration`
		주제 ARN

API	필수 권한	Resource

DescribeTopicPartitions	`kafka-cluster:Connect`
	`kafka-cluster:DescribeTopic`
	`kafka-cluster:DescribeTopicDynamicConfiguration`
		주제 ARN

> **Note**
>
> 의 경우 IAM 정책에서 클러스터 ARN을 `ListTopics`지정합니다. `DescribeTopic` 및의 경우 IAM 정책에서 주제 ARN을 `DescribeTopicPartitions`지정합니다.

Amazon MSK의 IAM 액세스 제어에 대한 자세한 내용은 섹션을 참조하세요the section called “IAM 액세스 제어”.

주제

- Amazon MSK 클러스터의 주제 나열
- 주제에 대한 자세한 정보 가져오기
- 주제에 대한 파티션 정보 보기

Amazon MSK 클러스터의 주제 나열

MSK 프로비저닝 클러스터의 모든 주제를 나열하여 파티션 수 및 복제 인자와 같은 기본 메타데이터를 볼 수 있습니다. 이는 클러스터의 주제를 모니터링하거나, 인벤토리 검사를 수행하거나, 추가 조사를 위한 주제를 식별하는 데 유용합니다.

> **Note**
>
> `ListTopics` API는 기본 주제 메타데이터를 제공합니다. 현재 상태 및 구성을 포함하여 특정 주제에 대한 자세한 정보를 가져오려면 `DescribeTopic` API를 사용합니다. 자세한 내용은 주제에 대한 자세한 정보 가져오기 단원을 참조하십시오.

> **Note**
>
> 이 API 응답은 약 1분마다 업데이트되는 데이터를 반영합니다. 변경 후 최신 주제 상태의 경우 쿼리하기 전에 약 1분 정도 기다려야 합니다.

주제

- 를 사용하여 주제 나열AWS Management 콘솔
- 를 사용하여 주제 나열AWS CLI
- API를 사용하여 주제 나열

를 사용하여 주제 나열AWS Management 콘솔

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 클러스터 목록에서 주제를 나열할 클러스터의 이름을 선택합니다.
3. 클러스터 세부 정보 페이지에서 주제 탭을 선택합니다.
4. 이 표에는 주제 이름, 파티션 수, 복제 인수, out-of-sync 복제본 수를 포함하여 클러스터의 모든 주제가 표시됩니다.

를 사용하여 주제 나열AWS CLI

다음 명령을 실행하여 *ClusterArn*을 클러스터의 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.

```
aws kafka list-topics --cluster-arn ClusterArn
```

이 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "topics": [
        {
            "topicArn": "arn:aws:kafka:us-east-1:123456789012:topic/MyCluster/abcd1234-
abcd-dcba-4321-a1b2abcd9f9f-2/MyTopic",
            "topicName": "MyTopic",
            "partitionCount": 3,
```

```
            "replicationFactor": 3,
            "outOfSyncReplicaCount": 0
        },
        {
            "topicArn": "arn:aws:kafka:us-east-1:123456789012:topic/MyCluster/abcd1234-
abcd-dcba-4321-a1b2abcd9f9f-2/AnotherTopic",
            "topicName": "AnotherTopic",
            "partitionCount": 6,
            "replicationFactor": 3,
            "outOfSyncReplicaCount": 1
        }
    ]
}
```

결과 페이지 매김

클러스터에 주제가 많은 경우 페이지 매김을 사용하여 더 작은 배치로 결과를 검색할 수 있습니다. `--max-results` 파라미터를 사용하여 반환할 최대 주제 수를 지정하고 `--next-token` 파라미터를 사용하여 결과의 다음 페이지를 검색합니다.

```
aws kafka list-topics --cluster-arn ClusterArn --max-results 10
```

사용 가능한 결과가 더 있는 경우 응답에 `nextToken` 값이 포함됩니다. 이 토큰을 사용하여 결과의 다음 페이지를 검색합니다.

```
aws kafka list-topics --cluster-arn ClusterArn --max-results 10 --next-token NextToken
```

이름을 기준으로 주제 필터링

`--topic-name-filter` 파라미터를 사용하여 접두사를 지정하여 주제 목록을 필터링할 수 있습니다. 그러면 이름이 지정된 접두사로 시작하는 주제만 반환됩니다.

```
aws kafka list-topics --cluster-arn ClusterArn --topic-name-filter "prod-"
```

이 명령은 `prod-orders` 또는 `prod-`와 같이 이름이 로 시작하는 주제만 반환합니다`prod-inventory`.

API를 사용하여 주제 나열

API를 사용하여 주제를 나열하려면 [ListTopics](#)를 참조하세요.

주제에 대한 자세한 정보 가져오기

현재 상태, 파티션 수, 복제 인수 및 구성을 포함하여 MSK 프로비저닝 클러스터의 특정 주제에 대한 자세한 정보를 검색할 수 있습니다. 이는 문제 해결, 주제 설정 검증 또는 작업 중 주제 상태 모니터링에 유용합니다.

> **Note**
>
> 이 API 응답은 약 1분마다 업데이트되는 데이터를 반영합니다. 변경 후 최신 주제 상태의 경우 쿼리하기 전에 약 1분 정도 기다려야 합니다.

주제

- 를 사용하여 주제 설명AWS Management 콘솔
- 를 사용하여 주제 설명AWS CLI
- API를 사용하여 주제 설명

를 사용하여 주제 설명AWS Management 콘솔

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 클러스터 목록에서 설명할 주제가 포함된 클러스터의 이름을 선택합니다.
3. 클러스터 세부 정보 페이지에서 주제 탭을 선택합니다.
4. 주제 목록에서 보려는 주제의 이름을 선택합니다.
5. 주제 세부 정보 페이지에는 상태, 파티션 수, 복제 인수, 구성 설정 등 주제에 대한 정보가 표시됩니다.

를 사용하여 주제 설명AWS CLI

다음 명령을 실행하여 *ClusterArn*을 클러스터의 Amazon 리소스 이름(ARN)으로 바꾸고 *TopicName*을 설명하려는 주제의 이름으로 바꿉니다.

```
aws kafka describe-topic --cluster-arn ClusterArn --topic-name TopicName
```

이 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "topicArn": "arn:aws:kafka:us-east-1:123456789012:topic/MyCluster/abcd1234-abcd-dcba-4321-a1b2abcd9f9f-2/MyTopic",
    "topicName": "MyTopic",
    "partitionCount": 3,
    "replicationFactor": 3,
    "configs": "Y29tcHJlc3Npb24udHlwZT1wcm9kdWNlcgpyZXRlbnRpb24ubXM9NjA0ODAwMDAw",
    "status": "ACTIVE"
}
```

주제 상태 이해

`status` 필드는 주제의 현재 상태를 나타냅니다. 다음 표에서는 가능한 상태 값을 설명합니다.

주제 상태 값

Status	설명
생성	주제가 생성 중입니다.
ACTIVE	주제가 활성 상태이고 사용할 준비가 되었습니다.
업데이트 중	주제 구성이 업데이트되고 있습니다.
DELETING	주제가 삭제되고 있습니다.

주제 구성 이해

`configs` 필드에는 Base64 형식으로 인코딩된 주제의 Kafka 구성 속성이 포함되어 있습니다. 읽기 가능한 형식으로 구성을 보려면 Base64 문자열을 디코딩해야 합니다.

다음 예제에서는 Linux 또는 macOS에서 `base64` 명령을 사용하여 구성을 디코딩하는 방법을 보여줍니다.

```
echo "Y29tcHJlc3Npb24udHlwZT1wcm9kdWNlcgpyZXRlbnRpb24ubXM9NjA0ODAwMDAw" | base64 --decode
```

디코딩된 출력에는 주제 구성 속성이 키-값 형식으로 표시됩니다.

```
compression.type=producer
```

```
retention.ms=604800000
```

주제 수준 구성 속성에 대한 자세한 내용은 섹션을 참조하세요the section called “주제 수준 Amazon MSK 구성”.

API를 사용하여 주제 설명

API를 사용하여 주제를 설명하려면 DescribeTopic을 참조하세요.

주제에 대한 파티션 정보 보기

MSK 프로비저닝된 클러스터에서 특정 주제의 파티션에 대한 자세한 정보를 검색할 수 있습니다. 이 정보에는 파티션 번호, 리더 브로커, 복제본 브로커 및 동기화 내 복제본(ISR)이 포함됩니다. 이는 파티션 배포를 모니터링하거나, 덜 복제된 파티션을 식별하거나, 복제 문제를 해결하는 데 유용합니다.

> **Note**
>
> 이 API 응답은 약 1분마다 업데이트되는 데이터를 반영합니다. 변경 후 최신 주제 상태의 경우 쿼리하기 전에 약 1분 정도 기다려야 합니다.

주제

- 를 사용하여 파티션 정보 보기AWS Management 콘솔
- 를 사용하여 파티션 정보 보기AWS CLI
- API를 사용하여 파티션 정보 보기

를 사용하여 파티션 정보 보기AWS Management 콘솔

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 클러스터 목록에서 주제가 포함된 클러스터의 이름을 선택합니다.
3. 클러스터 세부 정보 페이지에서 주제 탭을 선택합니다.
4. 주제 목록에서 파티션 정보를 보려는 주제의 이름을 선택합니다.
5. 주제 세부 정보 페이지에 파티션 정보가 표시되며 각 파티션의 파티션 번호, 리더 브로커, 복제본 및 동기화된 복제본이 표시됩니다.

를 사용하여 파티션 정보 보기AWS CLI

다음 명령을 실행하여 *ClusterArn*을 클러스터의 Amazon 리소스 이름(ARN)으로 바꾸고 *TopicName*을 주제 이름으로 바꿉니다.

```
aws kafka describe-topic-partitions --cluster-arn ClusterArn --topic-name TopicName
```

이 명령의 출력은 다음 JSON 예제와 같습니다.

```
{
    "partitions": [
        {
            "partition": 0,
            "leader": 1,
            "replicas": [1, 2, 3],
            "isr": [1, 2, 3]
        },
        {
            "partition": 1,
            "leader": 2,
            "replicas": [2, 3, 1],
            "isr": [2, 3, 1]
        },
        {
            "partition": 2,
            "leader": 3,
            "replicas": [3, 1, 2],
            "isr": [3, 1]
        }
    ]
}
```

파티션 정보 이해

응답에는 각 파티션에 대한 다음 정보가 포함됩니다.

- 파티션 - 파티션 번호입니다. 파티션은 0부터 번호가 매겨집니다.
- leader -이 파티션에 대한 리더의 브로커 ID입니다. 리더는 파티션에 대한 모든 읽기 및 쓰기 요청을 처리합니다.
- 복제본 -이 파티션의 복제본이 있는 브로커 IDs 목록입니다. 여기에는 동기화 내 복제본과 out-of-sync 복제본이 모두 포함됩니다.

- isr - 동기화된 복제본인 브로커 IDs 목록입니다. 이러한 복제본은 리더와 완전히 연결되며 필요한 경우 리더 역할을 맡을 수 있습니다.

위 예제에서 파티션 2에는 out-of-sync 복제본이 있습니다. `replicas` 목록에는 브로커 2가 포함되지만 `isr` 목록에는 포함되지 않습니다. 이는 브로커 2가 이 파티션의 리더와 완전히 연결되지 않았음을 나타냅니다.

결과 페이지 매김

주제에 파티션이 많은 경우 페이지 매김을 사용하여 더 작은 배치로 결과를 검색할 수 있습니다. `--max-results` 파라미터를 사용하여 반환할 최대 파티션 수를 지정하고 `--next-token` 파라미터를 사용하여 결과의 다음 페이지를 검색합니다.

```
aws kafka describe-topic-partitions --cluster-arn ClusterArn --topic-name TopicName --
max-results 10
```

사용 가능한 결과가 더 있는 경우 응답에 `nextToken` 값이 포함됩니다. 이 토큰을 사용하여 결과의 다음 페이지를 검색합니다.

```
aws kafka describe-topic-partitions --cluster-arn ClusterArn --topic-name TopicName --
max-results 10 --next-token NextToken
```

일반 사용 사례

파티션 정보 보기는 여러 시나리오에 유용합니다.

- 과소 복제된 파티션 식별 - `replicas` 및 `isr` 목록을 비교하여 일부 복제본이 동기화되지 않은 파티션을 식별합니다. 이는 성능 문제 또는 브로커 문제를 나타낼 수 있습니다.
- 파티션 배포 모니터링 - 파티션 리더가 브로커에 고르게 분산되어 있는지 확인하여 로드가 균형 잡힌지 확인합니다.
- 복제 문제 해결 - ISR 목록을 검사하여 복제를 따라잡는 데 문제가 있는 브로커를 식별합니다.
- 파티션 리밸런싱 계획 - 리밸런싱 작업을 수행하기 전에 이 정보를 사용하여 현재 파티션 레이아웃을 이해합니다.

API를 사용하여 파티션 정보 보기

API를 사용하여 파티션 정보를 보려면 DescribeTopicPartitions를 참조하세요.

Amazon MSK 리소스

Amazon MSK에서는 상황에 따라 리소스라는 용어에 두 가지 의미가 있습니다. API의 컨텍스트에서 리소스는 작업을 간접적으로 호출할 수 있는 구조입니다. 이러한 리소스의 목록과 해당 리소스에 대해 호출할 수 있는 작업은 Amazon MSK API 참조의 리소스를 참조하세요. the section called “IAM 액세스 제어”의 컨텍스트에서 리소스는 the section called “권한 부여 정책 리소스” 섹션에 정의된 대로 액세스를 허용하거나 거부할 수 있는 엔터티입니다.

Apache Kafka 버전

Amazon MSK 클러스터를 생성하는 경우 클러스터에 설치할 Apache Kafka 버전을 지정합니다. 또한 기존 클러스터의 Apache Kafka 버전을 업데이트할 수 있습니다. 이 장의 주제는 Kafka 버전 지원의 타임라인과 모범 사례에 대한 제안을 이해하는 데 도움이 됩니다.

주제

- 지원되는 Apache Kafka 버전
- Amazon MSK 버전 지원

지원되는 Apache Kafka 버전

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 다음 Apache Kafka 및 Amazon MSK 버전을 지원합니다. Apache Kafka 커뮤니티는 릴리스 날짜 이후 버전에 대해 약 12개월 동안 지원합니다. 자세한 내용은 Apache Kafka 수명 종료(EOL) 정책을 참조하세요.

다음 표에는 Amazon MSK가 지원하는 Apache Kafka 버전이 나열되어 있습니다.

Apache Kafka 버전	MSK 릴리스 날짜	지원 종료일
1.1.1	--	2024-06-05
2.1.0	--	2024-06-05
2.2.1	2019년 7월 31일	2024-06-08
2.3.1	2019년 12월 19일	2024-06-08
2.4.1	2020년 4월 2일	2024-06-08
2.4.1.1	2020-09-09	2024-06-08

Apache Kafka 버전	MSK 릴리스 날짜	지원 종료일
2.5.1	2020-09-30	2024-06-08
2.6.0	2020-10-21	2024-09-11
2.6.1	2021-01-19	2024-09-11
2.6.2	2021-04-29	2024-09-11
2.6.3	2021-12-21	2024-09-11
2.7.0	2020-12-29	2024-09-11
2.7.1	2021-05-25	2024-09-11
2.7.2	2021-12-21	2024-09-11
2.8.0	2021-05-19	2024-09-11
2.8.1	2022-10-28	2024-09-11
2.8.2-tiered	2022-10-28	2025-01-14
3.1.1	2022-06-22	2024-09-11
3.2.0	2022-06-22	2024-09-11
3.3.1	2022-10-26	2024-09-11
3.3.2	2023-03-02	2024-09-11
3.4.0	2023-05-04	2025-08-04
3.5.1	2023-09-26	2025-10-23
3.6.0	2023-11-16	2026-06-01
3.7.x	2024-05-29	--
3.8.x(권장)	2025-02-20	--

Apache Kafka 버전	MSK 릴리스 날짜	지원 종료일
3.9.x	2025-04-21	--
4.0.x	2025-05-16	--
4.1.x	2025-10-15	--

Amazon MSK 버전 지원 정책에 대한 자세한 내용은 Amazon MSK 버전 지원 정책 섹션을 참조하세요.

Amazon MSK 버전 4.1.x

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 미리 보기 기능으로 대기열, 조기 액세스의 새로운 스트림 리밸런싱 프로토콜, 적격 리더 복제본(ELR)을 도입하는 Apache Kafka 버전 4.1을 지원합니다. 이러한 기능과 함께 Apache Kafka 버전 4.1에는 다양한 버그 수정 및 개선 사항이 포함되어 있습니다.

Kafka 4.1의 주요 하이라이트는 미리 보기 기능으로 대기열을 도입하는 것입니다. 여러 컨슈머를 사용하여 동일한 토픽 파티션의 메시지를 처리하여 포인트 투 포인트 메시지 전송이 필요한 워크로드의 병렬 처리 및 처리량을 개선할 수 있습니다. 새로운 스트림 리밸런싱 프로토콜은 Kafka 4.0의 컨슈머 리밸런싱 프로토콜을 기반으로 하며 최적화된 작업 할당 및 리밸런싱을 위해 브로커 규모 조정 기능을 Kafka Streams로 확장합니다. 또한 이제 가용성을 강화하기 위해 ELR이 기본적으로 활성화됩니다.

자세한 사항 및 전체 개선 사항, 버그 수정 목록은 Apache Kafka 버전 4.1에 대한 릴리스 정보를 참조하세요.

Amazon MSK에서 Apache Kafka 4.1 사용을 시작하려면 AWS Management 콘솔AWS CLI또는 AWSSDKs를 통해 새 클러스터를 생성할 때 버전 4.1.x를 선택합니다. 인플레이스 롤링 업데이트를 사용하여 기존 MSK 프로비저닝 클러스터를 업그레이드할 수도 있습니다. Amazon MSK는 브로커를 다시 시작하여 가용성을 유지하고 업그레이드 중에 데이터를 보호합니다. Kafka 버전 4.1 지원은 Amazon MSK가 제공되는 모든 AWS 리전곳에서 사용할 수 있습니다.

Amazon MSK 버전 4.0.x

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 Apache Kafka 버전 4.0을 지원합니다. 이 버전은 클러스터 관리 및 성능의 최신 발전을 MSK Provisioned에 제공합니다. Kafka 4.0은 이제 일반적으로 사용할 수 있는 새로운 컨슈머 리밸런싱 프로토콜을 도입하여 더 원활하고 빠른 그룹 리밸런싱을 보장합니다. 또한 Kafka 4.0은 Java 17을 사용하기 위해 브로커와 도구가 필요하며, 향상된 보안 및 성능을 제공하고, 다양한 버그 수정 및 개선 사항을 포함하고, Apache ZooKeeper를 통한 메타데이터 관리를 중단합니다.

자세한 사항 및 전체 개선 사항, 버그 수정 목록은 Apache Kafka 버전 4.0에 대한 릴리스 정보를 참조하세요.

Amazon MSK 버전 3.9.x

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 Apache Kafka 버전 3.9을 지원합니다. 이 버전은 토픽 수준에서 계층형 스토리지를 비활성화할 때 계층형 데이터를 유지할 수 있도록 하여 계층형 스토리지 기능을 개선합니다. 컨슈머 애플리케이션은 로컬 및 원격 스토리지에서 연속 로그 오프셋을 유지하면서 원격 로그 시작 오프셋(Rx)에서 기록 데이터를 읽을 수 있습니다.

버전 3.9는 ZooKeeper 및 KRaft 메타데이터 관리 시스템을 모두 지원하는 마지막 버전입니다. Amazon MSK는 릴리스 날짜로부터 최소 2년 동안 버전 3.9에 대한 추가 지원을 제공합니다.

자세한 사항 및 전체 개선 사항, 버그 수정 목록은 Apache Kafka 버전 3.9.x에 대한 릴리스 정보를 참조하세요.

Amazon MSK 버전 3.8.x(권장)

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 Apache Kafka 버전 3.8을 지원합니다. 이제 메타데이터 관리를 위해 KRAFT 또는 ZooKeeper 모드와 함께 버전 3.8을 사용하여 새 클러스터를 생성하거나 버전 3.8을 사용하도록 기존 ZooKeeper 기반 클러스터를 업그레이드할 수 있습니다. Apache Kafka 버전 3.8에는 성능을 개선하는 몇 가지 버그 수정 사항과 새로운 기능이 포함되어 있습니다. 새로운 주요 기능에는 압축 수준 구성에 대한 지원이 포함됩니다. 이렇게 하면 기본 압축 수준을 변경할 수 있으므로 lz4, zstd 및 gzip과 같은 압축 유형을 사용할 때 성능을 더욱 최적화할 수 있습니다.

자세한 사항 및 전체 개선 사항, 버그 수정 목록은 Apache Kafka 버전 3.8.x에 대한 릴리스 정보를 참조하세요.

Apache Kafka 버전 3.7.x(프로덕션 환경 사용 가능 계층형 스토리지 포함)

MSK의 Apache Kafka 버전 3.7.x에는 Apache Kafka 버전 3.7.0에 대한 지원이 포함되어 있습니다. 새로운 3.7.x 버전을 사용하도록 클러스터를 생성하거나 기존 클러스터를 업그레이드할 수 있습니다. 버전 이름이 변경되면 Apache Kafka 커뮤니티에서 릴리스할 때 3.7.1과 같은 최신 패치 수정 버전을 더 이상 채택할 필요가 없습니다. Amazon MSK는 향후 패치 버전이 출시되면 해당 버전을 지원하도록 3.7.x를 자동으로 업데이트합니다. 자동 업데이트 덕분에 버전 업그레이드를 트리거하지 않고도 패치 수정 버전을 통해 사용할 수 있는 보안 및 버그 수정의 이점을 누릴 수 있습니다. Apache Kafka에서 릴리스한 이러한 패치 수정 버전은 버전 호환성을 저하시키지 않으며 클라이언트 애플리케이션의 읽기 또는 쓰기 오류에 대한 걱정 없이 새로운 패치 수정 버전의 이점을 누릴 수 있습니다. CloudFormation과 같은 인프라 자동화 도구가 버전 이름 변경 사항을 반영하도록 업데이트되었는지 확인하세요.

Amazon MSK는 이제 Apache Kafka 버전 3.7.x에서 KRaft 모드(Apache Kafka Raft)를 지원합니다. Amazon MSK에서는 ZooKeeper 노드와 마찬가지로 KRaft 컨트롤러가 추가 비용 없이 포함되어 있으므로 추가 설정이나 관리가 필요하지 않습니다. 이제 Apache Kafka 버전 3.7.x의 KRaft 모드 또는 ZooKeeper 모드에서 클러스터를 생성할 수 있습니다. Kraft 모드에서는 Zookeeper 기반 클러스터의 30개 브로커 할당량에 비해 제한 증가를 요청하지 않고 클러스터당 더 많은 파티션을 호스팅할 수 있도록 최대 60개의 브로커를 추가할 수 있습니다. MSK의 KRaft에 대한 자세한 내용은 KRaft 모드를 참조하세요.

Apache Kafka 버전 3.7.x에는 성능을 개선하는 몇 가지 버그 수정 사항과 새로운 기능도 포함되어 있습니다. 주요 개선 사항으로는 클라이언트를 위한 리더 검색 최적화와 로그 세그먼트 플러시 최적화 옵션이 있습니다. 전체 개선 사항 및 버그 수정 목록은 3.7.0에 대한 Apache Kafka 릴리스 정보를 참조하세요.

Apache Kafka 버전 3.6.0(프로덕션 환경 사용 가능 계층형 스토리지 포함)

Apache Kafka 버전 3.6.0(프로덕션 환경 사용 가능 계층형 스토리지 포함)에 대한 자세한 내용은 Apache Kafka 다운로드 사이트의 릴리스 정보를 참조하세요.

Amazon MSK에서는 안정성을 위해 이번 릴리스에서도 쿼럼 관리용으로 ZooKeeper를 계속 사용하고 관리할 예정입니다.

Amazon MSK 버전 3.5.1

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 새로운 클러스터와 기존 클러스터에 대해 Apache Kafka 버전 3.5.1을 지원합니다. Apache Kafka 3.5.1에는 성능을 개선하는 몇 가지 버그 수정 사항과 새로운 기능이 포함되어 있습니다. 주요 기능으로는 소비자를 위한 새로운 랙 인식 파티션 할당 도입이 포함됩니다. Amazon MSK는 이번 릴리스에서도 쿼럼 관리용으로 ZooKeeper를 계속 사용하고 관리할 예정입니다. 전체 개선 사항 및 버그 수정 목록은 3.5.1에 대한 Apache Kafka 릴리스 정보를 참조하세요.

Apache Kafka 버전 3.5.1에 대한 자세한 내용은 Apache Kafka 다운로드 사이트의 릴리스 정보를 참조하세요.

Amazon MSK 버전 3.4.0

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 새로운 클러스터와 기존 클러스터에 대해 Apache Kafka 버전 3.4.0을 지원합니다. Apache Kafka 3.4.0에는 성능을 개선하는 몇 가지 버그 수정 사항과 새로운 기능이 포함되어 있습니다. 주요 기능으로는 가장 가까운 복제본에서 가져올 수 있도록 안정성을 개선하는 수정 사항이 있습니다. Amazon MSK는 이번 릴리스에서도 쿼럼 관리

용으로 ZooKeeper를 계속 사용하고 관리할 예정입니다. 전체 개선 사항 및 버그 수정 목록은 3.4.0에 대한 Apache Kafka 릴리스 정보를 참조하세요.

Apache Kafka 버전 3.4.0에 대한 자세한 내용은 Apache Kafka 다운로드 사이트의 릴리스 정보를 참조하세요.

Amazon MSK 버전 3.3.2

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 새로운 클러스터와 기존 클러스터에 대해 Apache Kafka 버전 3.3.2를 지원합니다. Apache Kafka 3.3.2에는 성능을 개선하는 몇 가지 버그 수정 사항과 새로운 기능이 포함되어 있습니다. 주요 기능으로는 가장 가까운 복제본에서 가져올 수 있도록 안정성을 개선하는 수정 사항이 있습니다. Amazon MSK는 이번 릴리스에서도 쿼럼 관리용으로 ZooKeeper를 계속 사용하고 관리할 예정입니다. 전체 개선 사항 및 버그 수정 목록은 3.3.2에 대한 Apache Kafka 릴리스 정보를 참조하세요.

Apache Kafka 버전 3.3.2에 대한 자세한 내용은 Apache Kafka 다운로드 사이트의 릴리스 정보를 참조하세요.

Amazon MSK 버전 3.3.1

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 새로운 클러스터와 기존 클러스터에 대해 Apache Kafka 버전 3.3.1을 지원합니다. Apache Kafka 3.3.1에는 성능을 개선하는 몇 가지 버그 수정 사항과 새로운 기능이 포함되어 있습니다. 일부 주요 기능으로는 지표 및 파티셔너에 대한 개선 사항이 있습니다. Amazon MSK에서는 안정성을 위해 이번 릴리스에서도 쿼럼 관리용으로 ZooKeeper를 계속 사용하고 관리할 예정입니다. 전체 개선 사항 및 버그 수정 목록은 3.3.1에 대한 Apache Kafka 릴리스 정보를 참조하세요.

Apache Kafka 버전 3.3.1에 대한 자세한 내용은 Apache Kafka 다운로드 사이트의 릴리스 정보를 참조하세요.

Amazon MSK 버전 3.1.1

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 이제 새로운 클러스터와 기존 클러스터에 대해 Apache Kafka 버전 3.1.1 및 3.2.0을 지원합니다. Apache Kafka 3.1.1 및 Apache Kafka 3.2.0에는 성능을 개선하는 몇 가지 버그 수정 사항과 새로운 기능이 포함되어 있습니다. 일부 주요 기능으로는 지표 개선 사항과 주제 ID 사용이 있습니다. MSK에서는 안정성을 위해 이번 릴리스에서도 쿼럼 관리용으로 ZooKeeper를 계속 사용하고 관리할 예정입니다. 전체 개선 사항 및 버그 수정 목록은 3.1.1 및 3.2.0에 대한 Apache Kafka 릴리스 정보를 참조하세요.

Apache Kafka 버전 3.1.1 및 3.2.0에 대한 자세한 내용은 Apache Kafka 다운로드 사이트의 3.2.0 릴리스 정보 및 3.1.1 릴리스 정보를 참조하세요.

Amazon MSK 계층형 스토리지 버전 2.8.2.tiered

이번 릴리즈는 Apache Kafka 버전 2.8.2의 Amazon MSK 전용 버전이며, 오픈 소스 Apache Kafka 클라이언트와 호환됩니다.

2.8.2.tiered 릴리즈에는 Apache Kafka용 KIP-405에 도입된 API와 호환되는 계층형 스토리지 기능이 포함되어 있습니다. Amazon MSK 계층형 스토리지 기능에 대한 자세한 내용은 Standard 브로커를 위한 계층형 스토리지 섹션을 참조하세요.

Apache Kafka 버전 2.5.1

Apache Kafka 버전 2.5.1에는 몇 가지 버그 수정과 새로운 기능이 포함되어 있으며, 여기에는 Apache ZooKeeper 및 관리 클라이언트를 위한 전송 중 암호화가 포함됩니다. Amazon MSK는 DescribeCluster 작업으로 쿼리할 수 있는 TLS ZooKeeper 엔드포인트를 제공합니다.

DescribeCluster 작업의 출력에는 TLS ZooKeeper 엔드포인트를 나열하는 `ZookeeperConnectStringTls` 노드가 포함되어 있습니다.

다음 예제는 `DescribeCluster` 작업에 대한 응답의 `ZookeeperConnectStringTls` 노드를 보여 줍니다.

```
"ZookeeperConnectStringTls": "z-3.awskafkatutorialc.abcd123.c3.kafka.us-
east-1.amazonaws.com:2182,z-2.awskafkatutorialc.abcd123.c3.kafka.us-
east-1.amazonaws.com:2182,z-1.awskafkatutorialc.abcd123.c3.kafka.us-
east-1.amazonaws.com:2182"
```

Zookeeper를 사용한 TLS 암호화 사용에 대한 자세한 내용은 Apache ZooKeeper에서 TLS 보안 사용 섹션을 참조하세요.

Apache Kafka 버전 2.5.1에 대한 자세한 내용은 Apache Kafka 다운로드 사이트의 릴리스 정보를 참조하세요.

Amazon MSK 버그 수정 버전 2.4.1.1

이 릴리스는 Apache Kafka 버전 2.4.1의 Amazon MSK 전용 버그 수정 버전입니다. 이 버그 수정 릴리스에는 소비자 그룹이 지속적으로 재조정되어 `PreparingRebalance` 상태를 유지하는 드문 문제인 KAFKA-9752 관련 수정 사항이 포함되어 있습니다. 이 문제는 Apache Kafka 버전 2.3.1 및 2.4.1을 실행하는 클러스터에 영향을 미칩니다. 이 릴리스에는 Apache Kafka 버전 2.5.0에서 사용 가능한 커뮤니티 제작 수정 사항이 포함되어 있습니다.

> **Note**
>
> 버전 2.4.1.1을 실행하는 Amazon MSK 클러스터는 Apache Kafka 버전 2.4.1과 호환되는 모든 Apache Kafka 클라이언트와 호환됩니다.

Apache Kafka 2.4.1을 사용하시려면 새로운 Amazon MSK 클러스터에 MSK 버그 수정 버전 2.4.1.1을 사용하는 것을 권장합니다. 이 수정 사항을 적용하려면 Apache Kafka 버전 2.4.1을 실행하는 기존 클러스터를 이 버전으로 업데이트하면 됩니다. 기존 클러스터 업그레이드에 대한 자세한 내용은 Apache Kafka 버전 업그레이드 섹션을 참조하세요.

클러스터를 버전 2.4.1.1로 업그레이드하지 않고 이 문제를 해결하려면 Amazon MSK 클러스터 문제 해결 설명서의 `PreparingRebalance` 상태에 멈춘 소비자 그룹 섹션을 참조하세요.

Apache Kafka 버전 2.4.1(대신 2.4.1.1 사용)

> **Note**
>
> Apache Kafka 버전 2.4.1에서는 더 이상 MSK 클러스터를 생성할 수 없습니다. 대신 Apache Kafka 버전 2.4.1과 호환되는 클라이언트에서 Amazon MSK 버그 수정 버전 2.4.1.1을 사용할 수 있습니다. 또한 이미 Apache Kafka 버전 2.4.1이 설치된 MSK 클러스터가 있는 경우 대신 Apache Kafka 버전 2.4.1.1을 사용하도록 업데이트하는 것을 권장합니다.

KIP-392는 Apache Kafka 2.4.1 릴리스에 포함된 주요 Kafka 개선 제안 중 하나입니다. 이러한 개선 사항을 통해 소비자는 가장 가까운 복제본에서 가져올 수 있게 되었습니다. 이 기능을 사용하려면 소비자 속성의 `client.rack`을 소비자의 가용 영역 ID로 설정합니다. AZ ID의 예는 `use1-az1`입니다. Amazon MSK는 `broker.rack`을 브로커의 가용 영역 ID로 설정합니다. 또한 `replica.selector.class` 구성 속성을 Apache Kafka에서 제공하는 랙 인식의 구현인 `org.apache.kafka.common.replica.RackAwareReplicaSelector`로 설정해야 합니다.

이 버전의 Apache Kafka를 사용하면 `PER_TOPIC_PER_BROKER` 모니터링 수준의 지표는 해당 값이 처음으로 0이 아닌 값이 된 후에만 나타납니다. 자세한 내용은 the section called “`PER_TOPIC_PER_BROKER` 수준 모니터링” 단원을 참조하십시오.

가용 영역 IDs를 찾는 방법에 대한 자세한 내용은 AWS Resource Access Manager사용 설명서의 리소스의 AZ IDs를 참조하세요.

구성 속성 설정에 대한 자세한 내용은 the section called “브로커 구성” 단원을 참조하십시오.

KIP-392에 대한 자세한 내용은 Confluence 페이지의 Allow Consumers to Fetch from Closest Replica를 참조하십시오.

Apache Kafka 버전 2.4.1에 대한 자세한 내용은 Apache Kafka 다운로드 사이트의 릴리스 정보를 참조하십시오.

Amazon MSK 버전 지원

이 주제에서는 Amazon MSK 버전 지원 정책 및 Apache Kafka 버전 업그레이드에 대한 절차를 설명합니다. Kafka 버전을 업그레이드하는 경우 버전 업그레이드의 모범 사례에 설명된 모범 사례를 따르세요.

주제

- Amazon MSK 버전 지원 정책
- Apache Kafka 버전 업그레이드
- 버전 업그레이드의 모범 사례

Amazon MSK 버전 지원 정책

이 섹션에서는 Amazon MSK 지원 Kafka 버전에 대한 지원 정책을 설명합니다.

- 모든 Kafka 버전은 지원 종료일에 도달할 때까지 지원됩니다. 지원 종료일에 대한 자세한 내용은 지원되는 Apache Kafka 버전 단원을 참조하세요. MSK 클러스터를 지원 종료일 전에 권장 Kafka 버전 이상으로 업그레이드합니다. Apache Kafka 버전 업그레이드에 대한 자세한 내용은 Apache Kafka 버전 업그레이드 섹션을 참조하세요. 지원 종료일 후 Kafka 버전을 사용하는 클러스터는 권장 Kafka 버전으로 자동 업그레이드됩니다. 자동 업그레이드는 지원 종료 날짜 이후 언제든지 발생할 수 있습니다. 업그레이드 전에는 알림이 제공되지 않습니다.
- MSK는 새로 생성된 클러스터에서 지원 종료일이 공개된 Kafka 버전을 사용하는 경우 이러한 클러스터에 대한 지원을 단계적으로 중단합니다.

Apache Kafka 버전 업그레이드

기존 MSK 클러스터를 최신 버전의 Apache Kafka로 업그레이드할 수 있습니다. 클러스터의 Kafka 버전을 업그레이드하기 전에 클라이언트 측 소프트웨어 버전이 새 Kafka 버전의 기능을 지원하는지 확인합니다.

업그레이드 중에 클러스터를 고가용성으로 만드는 방법에 대한 자세한 내용은 the section called “고가용성 클러스터 빌드” 섹션을 참조하세요.

를 사용하여 Apache Kafka 버전 업그레이드AWS Management 콘솔

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 탐색 모음에서 MSK 클러스터를 생성한 리전을 선택합니다.
3. 업그레이드하려는 MSK 클러스터를 선택합니다.
4. 속성 탭의 Apache Kafka 버전 섹션에서 업그레이드를 선택합니다.
5. Apache Kafka 버전 섹션에서 다음을 수행합니다.

    a. Apache Kafka 버전 선택 드롭다운 목록에서 업그레이드할 대상 버전을 선택합니다. 예를 들어 **`3.9.x`**를 선택합니다.

    b. (선택 사항) 버전 호환성 보기를 선택하여 클러스터의 현재 버전과 사용 가능한 업그레이드 버전 간의 호환성을 확인합니다. 그런 다음 선택을 선택하여 계속 진행합니다.

    > **Note**
    >
    > Amazon MSK는 대부분의 Apache Kafka 버전으로의 현재 위치 업그레이드를 지원합니다. 그러나 ZooKeeper 기반 Kafka 버전에서 KRaft 기반 버전으로 업그레이드할 때는 새 클러스터를 생성해야 합니다. 그런 다음 데이터를 새 클러스터에 복사하고 클라이언트를 새 클러스터로 전환합니다.

    c. (선택 사항) 클러스터 구성 업데이트 확인란을 선택하여 새 버전과 호환되는 구성 업데이트를 적용합니다. 이렇게 하면 새 버전의 기능과 개선 사항이 활성화됩니다.

    기존 사용자 지정 구성을 유지 관리해야 하는 경우 이 단계를 건너뛸 수 있습니다.

    > **Note**
    >
    > - 서버 측 업그레이드는 클라이언트 애플리케이션을 자동으로 업데이트하지 않습니다.
    > - 클러스터 안정성을 유지하기 위해 버전 다운그레이드는 지원되지 않습니다.

    d. 업그레이드를 선택하여 프로세스를 시작합니다.

를 사용하여 Apache Kafka 버전 업그레이드AWS CLI

1. 다음 명령을 실행하여 *ClusterArn*을 클러스터 생성 후 받은 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.

```
aws kafka get-compatible-kafka-versions --cluster-arn ClusterArn
```

이 명령의 출력에는 클러스터를 업그레이드할 수 있는 Apache Kafka 버전 목록이 포함되어 있습니다. 다음 예제와 같습니다.

```
{
    "CompatibleKafkaVersions": [
        {
            "SourceVersion": "2.2.1",
            "TargetVersions": [
                "2.3.1",
                "2.4.1",
                "2.4.1.1",
                "2.5.1"
            ]
        }
    ]
}
```

2. 다음 명령을 실행하여 *ClusterArn*을 클러스터 생성 후 받은 Amazon 리소스 이름(ARN)으로 바꿉니다. 클러스터에 대한 ARN이 없는 경우, 모든 클러스터를 나열하여 찾을 수 있습니다. 자세한 내용은 the section called “클러스터 나열” 단원을 참조하십시오.

*Current-Cluster-Version*을 클러스터의 현재 버전으로 바꿉니다. *TargetVersion*의 경우 이전 명령의 출력에서 대상 버전을 지정할 수 있습니다.

> **Important**
>
> 클러스터 버전은 단순한 정수가 아닙니다. 클러스터의 현재 버전을 찾으려면 DescribeCluster 작업 또는 describe-clusterAWS CLI 명령을 사용합니다. 버전의 예를 들면 `KTVPDKIKX0DER`입니다.

```
aws kafka update-cluster-kafka-version --cluster-arn ClusterArn --current-
version Current-Cluster-Version --target-kafka-version TargetVersion
```

이전 명령의 출력은 다음 JSON과 같습니다.


```
{

    "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
abcdefab-1234-abcd-5678-cdef0123ab01-2",
    "ClusterOperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef"
}
```


3. update-cluster-kafka-version 작업 결과를 가져오려면 다음 명령을 실행하여 *ClusterOperationArn*을 update-cluster-kafka-version 명령의 출력에서 가져온 ARN으로 바꿉니다.

```
aws kafka describe-cluster-operation --cluster-operation-arn ClusterOperationArn
```

이 describe-cluster-operation 명령의 출력은 다음 JSON 예제와 같습니다.


```
{
    "ClusterOperationInfo": {
        "ClientRequestId": "62cd41d2-1206-4ebf-85a8-dbb2ba0fe259",
        "ClusterArn": "arn:aws:kafka:us-east-1:012345678012:cluster/
exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2",
        "CreationTime": "2021-03-11T20:34:59.648000+00:00",
        "OperationArn": "arn:aws:kafka:us-east-1:012345678012:cluster-
operation/exampleClusterName/abcdefab-1234-abcd-5678-cdef0123ab01-2/0123abcd-
abcd-4f7f-1234-9876543210ef",
        "OperationState": "UPDATE_IN_PROGRESS",
        "OperationSteps": [
            {
                "StepInfo": {
                    "StepStatus": "IN_PROGRESS"
                },
                "StepName": "INITIALIZE_UPDATE"
            },
            {
```

```
                "StepInfo": {
                    "StepStatus": "PENDING"
                },
                "StepName": "UPDATE_APACHE_KAFKA_BINARIES"
            },
            {
                "StepInfo": {
                    "StepStatus": "PENDING"
                },
                "StepName": "FINALIZE_UPDATE"
            }
        ],
        "OperationType": "UPDATE_CLUSTER_KAFKA_VERSION",
        "SourceClusterInfo": {
            "KafkaVersion": "2.4.1"
        },
        "TargetClusterInfo": {
            "KafkaVersion": "2.6.1"
        }
    }
}
```

`OperationState` 값이 `UPDATE_IN_PROGRESS`인 경우, 잠시 기다린 다음 `describe-cluster-operation` 명령을 다시 실행합니다. 작업이 완료되면 `OperationState`의 값이 `UPDATE_COMPLETE`가 됩니다. Amazon MSK가 작업을 완료하는 데 걸리는 시간은 다양하므로 작업이 완료될 때까지 반복해서 확인해야 할 수 있습니다.

API를 사용하여 Apache Kafka 버전 업그레이드

1. 클러스터를 업그레이드할 수 있는 Apache Kafka 버전 목록을 가져오려면 GetCompatibleKafkaVersions 작업을 호출합니다.
2. 호환되는 Apache Kafka 버전 중 하나에 클러스터를 업그레이드하려면 UpdateClusterKafkaVersion 작업을 호출합니다.

버전 업그레이드의 모범 사례

Kafka 버전 업그레이드 프로세스의 일부로 수행되는 롤링 업데이트 중에 클라이언트 연속성을 보장하려면 다음과 같은 클라이언트 및 Apache Kafka 주제의 구성을 검토합니다.

- 주제 복제 계수(RF)를 2개의 AZ 클러스터의 경우 2의 최소값으로 설정하고 3개의 AZ 클러스터의 경우 3의 최소값으로 설정합니다. 2의 RF 값은 패치 적용 중에 오프라인 파티션으로 이어질 수 있습니다.
- 최소 동기화 내 복제본(minISR)을 `miniISR = (RF) - 1`인 Replication Factor(RF)보다 1 작은 최대값으로 설정합니다. 이렇게 하면 파티션 복제본 세트가 한 복제본이 오프라인 또는 과소 복제되는 것을 허용할 수 있습니다.
- 여러 브로커 연결 문자열을 사용하도록 클라이언트를 구성합니다. 클라이언트의 연결 문자열에 여러 브로커가 있으면 클라이언트 I/O를 지원하는 특정 브로커가 패치되기 시작하는 경우 장애 조치가 가능합니다. 여러 브로커가 있는 연결 문자열을 가져오는 방법에 대한 자세한 내용은 [Amazon MSK 클러스터의 부트스트랩 브로커 가져오기]를 참조하세요.
- 새로운 버전에서 사용 가능한 기능을 활용하려면 연결 클라이언트를 권장 버전 이상으로 업그레이드하는 것이 좋습니다. 클라이언트 업그레이드는 MSK 클러스터 Kafka 버전의 수명 종료(EOL) 날짜의 적용을 받지 않으며 EOL 날짜까지 완료할 필요가 없습니다. Apache Kafka는 이전 클라이언트가 최신 클러스터에서 작업할 수 있도록 허용하는 [양방향 클라이언트 호환성 정책]을 제공하며 그 반대의 경우도 마찬가지입니다.
- 버전 3.x.x를 사용하는 Kafka 클라이언트에는 `acks=all` 및 `enable.idempotence=true` 기본값이 적용될 수 있습니다. `acks=all`은 `acks=1`의 이전 기본값과 다르며 모든 동기화 내 복제본이 생산 요청을 승인하도록 하여 더 강화된 내구성을 제공합니다. 마찬가지로 `enable.idempotence`의 기본값은 이전에 `false`였습니다. `enable.idempotence=true`를 기본값으로 변경하면 중복 메시지의 가능성이 낮아집니다. 이러한 변경 사항은 모범 사례 설정으로 간주되며 정상 성능 파라미터 내에 약간의 추가 지연 시간이 발생할 수 있습니다.
- 새로운 MSK 클러스터를 생성할 때는 권장 Kafka 버전을 사용합니다. 권장 Kafka 버전을 사용하면 최신 Kafka 및 MSK 기능을 활용할 수 있습니다.

Amazon MSK 클러스터 문제 해결

다음 정보는 Amazon MSK 클러스터에서 발생할 수 있는 문제를 해결하는 데 도움이 될 수 있습니다. 또한 [AWS re:Post]에 문제를 게시할 수 있습니다. Amazon MSK Replicator 문제 해결은 [MSK Replicator 문제 해결] 단원을 참조하세요.

주제

- [볼륨 교체 때문에 복제 과부하로 인해 디스크 포화 발생]
- [PreparingRebalance 상태에 멈춘 소비자 그룹]
- [브로커 로그를 Amazon CloudWatch Logs에 전달하는 중 오류 발생]
- [기본 보안 그룹 없음]

- 클러스터가 CREATING 상태에 정체된 것으로 표시됨
- 클러스터 상태가 CREATING에서 FAILED로 바뀜
- 클러스터가 ACTIVE 상태에 있지만 생산자가 데이터를 보낼 수 없거나 소비자가 데이터를 받을 수 없음
- AWS CLI가 Amazon MSK를 인식하지 못함
- 파티션이 오프라인으로 전환되거나 복제본이 동기화되지 않음
- 디스크 공간이 부족함
- 메모리 부족
- 생산자가 NotLeaderForPartitionException을 받음
- 복제되지 않은 파티션(URP)이 0보다 큼
- 클러스터에는 __amazon_msk_canary와 __amazon_msk_canary_state라는 주제가 있습니다.
- 파티션 복제 실패
- 퍼블릭 액세스가 활성화된 클러스터에 액세스할 수 없음
- 내에서 클러스터에 액세스할 수 없음AWS: 네트워킹 문제
- 인증 실패: 연결 횟수가 너무 많음
- 인증 실패: 세션이 너무 짧음
- MSK 서버리스: 클러스터 생성 실패
- MSK 구성에서 KafkaVersionsList를 업데이트할 수 없음

볼륨 교체 때문에 복제 과부하로 인해 디스크 포화 발생

예기치 않은 볼륨 하드웨어 장애 발생 시 Amazon MSK에서 볼륨을 새 인스턴스로 교체할 수 있습니다. Kafka는 클러스터의 다른 브로커에서 파티션을 복제하여 새로운 볼륨을 다시 채웁니다. 파티션이 복제되고 인식되면 리더십 및 비동기 복제본(ISR) 멤버가 될 수 있습니다.

문제

볼륨 교체에서 복구하는 브로커의 경우 크기가 다양한 일부 파티션이 다른 파티션보다 먼저 온라인 상태로 다시 전환될 수 있습니다. 이러한 파티션은 여전히 다른 파티션을 추적하고 있는(복제하고 있는) 동일한 브로커의 트래픽을 제공할 수 있으므로 문제가 될 수 있습니다. 이 복제 트래픽은 경우에 따라 기본 볼륨 처리량 한도를 포화시킬 수 있습니다. 기본 사례에서는 초당 250MiB입니다. 이 포화가 발생하면 이미 인식된 모든 파티션이 영향을 받아 ISR을 해당 파티션과 공유하는 모든 브로커(원격 승인

`acks=all`으로 인한 리더 파티션뿐만 아니라)의 클러스터 전체에서 지연 시간이 발생합니다. 이 문제는 다양한 크기의 파티션 수가 많은 더 큰 클러스터에서 더 일반적입니다.

권장 사항

- 복제 I/O 상태를 개선하려면 모범 사례 스레드 설정이 마련되어 있어야 합니다.
- 기본 볼륨 포화 가능성을 줄이려면 처리량이 더 높은 프로비저닝된 스토리지를 활성화합니다. 처리량이 많은 복제 사례의 경우 최소 처리량 값 500MiB/s가 권장되지만 필요한 실제 값은 처리량 및 사용 사례에 따라 달라집니다. Amazon MSK 클러스터의 Standard 브로커에 대한 스토리지 처리량 프로비저닝
- 복제 압력을 최소화하려면 `num.replica.fetchers`를 기본값인 2로 낮춥니다.

`PreparingRebalance` 상태에 멈춘 소비자 그룹

하나 이상의 소비자 그룹이 영구 재조정 상태에 머무는 경우, Apache Kafka 버전 2.3.1 및 2.4.1에 영향을 주는 Apache Kafka 문제 KAFKA-9752가 원인일 수 있습니다.

문제를 해결하려면 문제에 대한 수정 사항이 포함된 Amazon MSK 버그 수정 버전 2.4.1.1로 클러스터를 업그레이드하는 것을 권장합니다. 기존 클러스터를 Amazon MSK 버그 수정 버전 2.4.1.1로 업데이트하는 방법에 대한 자세한 내용은 Apache Kafka 버전 업그레이드 섹션을 참조하세요.

클러스터를 Amazon MSK 버그 수정 버전 2.4.1.1로 업그레이드하지 않고 이 문제를 해결하기 위한 해결 방법은 Kafka 클라이언트가 고정 멤버십 프로토콜을 사용하도록 설정하거나 멈춘 소비자 그룹의 조정 브로커 노드에 식별 및 재부팅 작업을 수행하는 것입니다.

정적 멤버십 프로토콜 구현

클라이언트에서 정적 멤버십 프로토콜을 구현하려면 다음을 수행합니다.

1. Kafka Consumers 구성의 `group.instance.id` 속성을 그룹 내 소비자를 식별하는 정적 문자열로 설정합니다.
2. 구성의 다른 인스턴스가 정적 문자열을 사용하도록 업데이트되었는지 확인합니다.
3. 변경 사항을 Kafka 소비자에게 배포합니다.

클라이언트 구성에서 세션 시간 제한을 소비자 그룹 재조정을 조기에 트리거하지 않고 소비자가 복구할 수 있는 기간으로 설정하는 경우 정적 멤버십 프로토콜을 사용하는 것이 더 효과적입니다. 예를 들어 소비자 애플리케이션이 5분 동안의 사용 불가 상태를 견딜 수 있는 경우 세션 시간 제한의 합리적인 값은 기본값인 10초가 아닌 4분이 될 수 있습니다.

> **Note**
>
> 정적 멤버십 프로토콜을 사용하면 해당 문제가 발생할 확률이 줄어듭니다. 정적 멤버십 프로토콜을 사용하는 경우에도 이 문제가 발생할 수 있습니다.

조정 브로커 노드 재부팅

조정 브로커 노드를 재부팅하려면 다음을 수행합니다.

1. `kafka-consumer-groups.sh` 명령을 사용하여 그룹 코디네이터를 식별합니다.
2. RebootBroker API 작업을 사용하여 중단된 소비자 그룹의 그룹 코디네이터를 다시 시작합니다.

브로커 로그를 Amazon CloudWatch Logs에 전달하는 중 오류 발생

브로커 로그를 Amazon CloudWatch Logs로 전송하도록 클러스터를 설정하려고 할 때 두 가지 예외 중 하나가 발생할 수 있습니다.

`InvalidInput.LengthOfCloudWatchResourcePolicyLimitExceeded` 예외가 발생하는 경우 다시 시도하되 `/aws/vendedlogs/`로 시작하는 로그 그룹을 사용하십시오. 자세한 내용은 특정 Amazon Web Services에서 로깅 활성화를 참조하세요.

`InvalidInput.NumberOfCloudWatchResourcePoliciesLimitExceeded` 예외가 발생하면 계정에서 기존 Amazon CloudWatch Logs 정책을 선택하고 다음 JSON을 추가합니다.

```
{"Sid":"AWSLogDeliveryWrite","Effect":"Allow","Principal":
{"Service":"delivery.logs.amazonaws.com"},"Action":
["logs:CreateLogStream","logs:PutLogEvents"],"Resource":["*"]}
```

위의 JSON을 기존 정책에 추가하려고 하는데 선택한 정책의 최대 길이에 도달했다는 오류가 표시되는 경우 다른 Amazon CloudWatch Logs 정책 중 하나에 JSON을 추가해 보세요. 기존 정책에 JSON을 추가한 후 다시 한번 브로커 로그 전달을 Amazon CloudWatch Logs로 설정합니다.

기본 보안 그룹 없음

클러스터를 생성하려고 하는데 기본 보안 그룹이 없다는 오류가 표시되는 경우, 본인의 공유 VPC를 사용하고 있기 때문일 수 있습니다. 관리자에게 이 VPC의 보안 그룹을 설명할 수 있는 권한을 요청하고 다시 시도하십시오. 이 작업을 허용하는 정책의 예는 Amazon EC2: 특정 VPC와 연결된 EC2 보안 그룹을 콘솔에서 프로그래밍 방식으로 관리할 수 있도록 허용을 참조하십시오.

클러스터가 CREATING 상태에 정체된 것으로 표시됨

간혹 클러스터 생성에 최대 30분이 소요될 수 있습니다. 30분 동안 기다렸다가 클러스터의 상태를 다시 확인하십시오.

클러스터 상태가 CREATING에서 FAILED로 바뀜

클러스터를 다시 생성해 보십시오.

클러스터가 ACTIVE 상태에 있지만 생산자가 데이터를 보낼 수 없거나 소비자가 데이터를 받을 수 없음

- 클러스터 생성에 성공했지만(클러스터 상태는 `ACTIVE`) 데이터를 보내거나 받을 수 없는 경우, 생산자 및 소비자 애플리케이션이 클러스터에 액세스할 수 있는지 확인합니다. 자세한 내용은 the section called "클라이언트 머신 생성"의 지침을 참조하십시오.

- 생산자와 소비자가 클러스터에 액세스할 수 있지만 데이터 생산과 소비에 계속 문제가 발생하는 경우 가능한 원인은 KAFKA-7697로, 이 경우 Apache Kafka 버전 2.1.0에 영향을 미치고 이로 인해 하나 이상의 브로커가 교착 상태에 빠질 수 있습니다. 이 버그의 영향을 받지 않는 Apache Kafka 2.2.1로 마이그레이션하는 것을 고려하십시오. 마이그레이션 방법에 대한 자세한 내용은 the section called "MSK 클러스터로 마이그레이션" 단원을 참조하십시오.

AWS CLI가 Amazon MSK를 인식하지 못함

가 AWS CLI설치되어 있지만 Amazon MSK 명령을 인식하지 못하는 경우 AWS CLI를 최신 버전으로 업그레이드합니다. 업그레이드 방법에 대한 자세한 지침은 설치를 AWS CLI참조하세요. AWS Command Line Interface 를 사용하여 Amazon MSK 명령을 실행하는 방법에 대한 자세한 내용은 섹션을 참조AWS CLI하세요the section called "주요 기능 및 개념".

파티션이 오프라인으로 전환되거나 복제본이 동기화되지 않음

디스크 공간 부족 증상일 수 있습니다. the section called "디스크 공간이 부족함"을(를) 참조하세요.

디스크 공간이 부족함

디스크 공간 관리의 모범 사례인 the section called "디스크 공간 모니터링" 및 the section called "데이터 보존 파라미터 조정" 단원을 참조하십시오.

메모리 부족

`MemoryUsed` 지표가 높거나 `MemoryFree`가 낮다고 해서 문제가 있다는 뜻은 아닙니다. Apache Kafka는 최대한 많은 메모리를 사용하도록 설계되었으며 최적으로 관리합니다.

생산자가 NotLeaderForPartitionException을 받음

종종 발생하는 일시적인 오류입니다. 생산자의 `retries` 구성 파라미터를 현재 값보다 큰 값으로 설정합니다.

복제되지 않은 파티션(URP)이 0보다 큼

`UnderReplicatedPartitions` 지표는 모니터링해야 하는 중요한 지표입니다. 정상 MSK 클러스터에서 이 지표의 값은 0입니다. 0보다 크면 다음 이유 중 하나 때문일 수 있습니다.

- `UnderReplicatedPartitions`가 급증하는 경우, 클러스터가 수신 트래픽과 발신 트래픽을 처리하기에 적합한 크기로 프로비저닝되지 않았기 때문일 수 있습니다. the section called "Standard 브로커 모범 사례"을(를) 참조하세요.
- 트래픽이 낮은 기간에도 `UnderReplicatedPartitions` 값이 0보다 큰 증상이 지속된다면, 주제에 브로커 액세스 권한을 부여하지 않는 제한적인 ACL을 설정했기 때문일 수 있습니다. 파티션을 복제하려면 브로커가 READ 및 DESCRIBE 주제 모두에 대한 권한이 있어야 합니다. DESCRIBE는 READ 권한과 함께 기본적으로 부여됩니다. ACL 설정에 대한 자세한 내용은 Apache Kafka 설명서의 권한 부여 및 ACL을 참조하십시오.

클러스터에는 __amazon_msk_canary와 __amazon_msk_canary_state라는 주제가 있습니다.

MSK 클러스터에 `__amazon_msk_canary`라는 이름의 주제와 `__amazon_msk_canary_state`라는 이름의 주제가 있는 것을 볼 수 있습니다. 이는 Amazon MSK가 클러스터 상태 및 진단 지표에 대해 생성하고 사용하는 내부 주제입니다. 이러한 주제는 크기가 무시할 수 있을 정도로 작으며 삭제할 수 없습니다.

파티션 복제 실패

CLUSTER_ACTIONS에 ACL을 설정하지 않았는지 확인합니다.

퍼블릭 액세스가 활성화된 클러스터에 액세스할 수 없음

클러스터에 공용 액세스가 활성화되어 있지만 여전히 인터넷에서 클러스터에 액세스할 수 없는 경우 다음 단계를 수행합니다.

1. 클러스터의 보안 그룹 인바운드 규칙이 IP 주소와 클러스터의 포트를 허용하는지 확인합니다. 클러스터 포트 번호 목록은 the section called “포트 정보” 섹션을 참조하세요. 또한 보안 그룹의 아웃바운드 규칙이 아웃바운드 통신을 허용하는지 확인합니다. 보안 그룹 및 해당 인바운드 및 아웃바운드 규칙에 대한 자세한 내용은 Amazon VPC 사용 설명서에서 VPC의 보안 그룹을 참조하세요.
2. IP 주소와 클러스터 포트가 클러스터 VPC 네트워크 ACL의 인바운드 규칙에 허용되는지 확인합니다. 보안 그룹과 달리 네트워크 ACL은 상태 비저장입니다. 즉, 인바운드 및 아웃바운드 규칙을 모두 구성해야 합니다. 아웃바운드 규칙에서 모든 트래픽(포트 범위: 0~65535)을 사용자 IP 주소로 허용합니다. 자세한 내용은 Amazon VPC 사용 설명서의 규칙 추가 및 삭제를 참조하세요.
3. 퍼블릭 액세스 부트스트랩 브로커 문자열을 사용하여 클러스터에 액세스하고 있는지 확인합니다. 퍼블릭 액세스가 활성화된 MSK 클러스터에는 두 개의 서로 다른 부트스트랩 브로커 문자열이 있는데, 하나는 퍼블릭 액세스를 위한 것이고 다른 하나는 AWS 내에서 액세스하기 위한 것입니다. 자세한 내용은 the section called “를 사용하여 부트스트랩 브로커 가져오기AWS Management 콘솔” 단원을 참조하십시오.

내에서 클러스터에 액세스할 수 없음AWS: 네트워킹 문제

MSK 클러스터와 성공적으로 통신할 수 없는 Apache Kafka 애플리케이션이 있는 경우 다음 연결 테스트를 수행하여 시작합니다.

1. the section called “부트스트랩 브로커 가져오기”에 설명된 방법 중 하나를 사용하여 부트스트랩 브로커의 주소를 가져옵니다.
2. 다음 명령에서 *bootstrap-broker*를 이전 단계에서 얻은 브로커 주소 중 하나로 바꿉니다. 클러스터가 TLS 인증을 사용하도록 설정된 경우 *port-number*를 9094로 바꿉니다. 클러스터에서 TLS 인증을 사용하지 않는 경우 *port-number*를 9092로 바꿉니다. 클라이언트 시스템에서 명령을 실행합니다.

   ```
   telnet bootstrap-broker port-number
   ```

   포트 번호는 다음과 같습니다.

   - 클러스터가 TLS 인증을 사용하도록 설정된 경우 9094
   - 클러스터에서 TLS 인증을 사용하지 않는 경우 9092
   - 퍼블릭 액세스가 활성화된 경우에는 다른 포트 번호가 필요

   클라이언트 시스템에서 명령을 실행합니다.
3. 모든 부트스트랩 브로커에 대해 이전 명령을 반복합니다.

클라이언트 머신이 브로커에 액세스할 수 있는 경우 이는 연결 문제가 없음을 의미합니다. 이 경우 다음 명령을 실행하여 Apache Kafka 클라이언트가 올바르게 설정되어 있는지 확인합니다. *bootstrap-brokers*를 가져오려면 the section called "부트스트랩 브로커 가져오기"에 설명된 방법 중 하나를 사용합니다. *topic*을 해당 주제의 이름으로 바꿉니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-
list bootstrap-brokers --producer.config client.properties --topic topic
```

이전 명령이 성공하면 클라이언트가 올바르게 설정되었음을 의미합니다. 여전히 애플리케이션에서 생성 및 소비할 수 없는 경우 애플리케이션 수준에서 문제를 디버그하십시오.

클라이언트 머신이 브로커에 액세스할 수 없는 경우 클라이언트 시스템 설정에 따른 지침은 다음 하위 섹션을 참조하세요.

동일한 VPC의 Amazon EC2 클라이언트와 MSK 클러스터

클라이언트 머신이 MSK 클러스터와 동일한 VPC에 있는 경우 클러스터의 보안 그룹에 클라이언트 머신의 보안 그룹으로부터의 트래픽을 허용하는 인바운드 규칙이 있는지 확인합니다. 이러한 규칙 설정에 대한 자세한 내용은 보안 그룹 규칙을 참조하십시오. 클러스터와 동일한 VPC에 있는 Amazon EC2 인스턴스에서 클러스터에 액세스하는 방법의 예제는 the section called "시작하기" 섹션을 참조하세요.

서로 다른 VPC의 Amazon EC2 클라이언트와 MSK 클러스터

클라이언트 시스템과 클러스터가 서로 다른 두 VPC에 있는 경우 다음을 확인합니다.

- 두 VPC가 피어링되어 있습니다.
- 피어링 연결의 상태가 활성 상태입니다.
- 두 VPC의 라우팅 테이블이 올바르게 설정되어 있습니다.

VPC 피어링에 대한 자세한 내용은 VPC 피어링 연결 작업을 참조하십시오.

온프레미스 클라이언트

를 사용하여 MSK 클러스터에 연결하도록 설정된 온프레미스 클라이언트의 경우 다음을 Site-to-Site VPN확인합니다.

- VPN 연결 상태가 UP입니다. VPN 연결 상태를 확인하는 방법에 대한 자세한 내용은 VPN 터널의 현재 상태를 확인하려면 어떻게 합니까?를 참조하십시오.

- 클러스터 VPC의 라우팅 테이블에는 대상의 형식이 `Virtual private gateway(vgw-xxxxxxxx)`인 온프레미스 CIDR에 대한 라우팅이 포함되어 있습니다.
- MSK 클러스터의 보안 그룹은 포트 2181, 포트 9092(클러스터가 일반 텍스트 트래픽을 허용하는 경우), 포트 9094(클러스터가 TLS 암호화 트래픽을 허용하는 경우)의 트래픽을 허용합니다.

자세한 Site-to-Site VPN문제 해결 지침은 [Client VPN 문제 해결을 참조하세요](#).

Direct Connect

클라이언트가를 사용하는 경우 문제 해결을 Direct Connect참조하세요. [Direct Connect](#)

이전 문제 해결 지침으로 문제가 해결되지 않으면 네트워크 트래픽을 차단하는 방화벽이 없는지 확인하십시오. 추가 디버깅을 위해 `tcpdump` 및 `Wireshark`와 같은 도구를 사용하여 트래픽을 분석하고 트래픽이 MSK 클러스터에 도달하고 있는지 확인합니다.

인증 실패: 연결 횟수가 너무 많음

`Failed authentication ... Too many connects` 오류는 하나 이상의 IAM 클라이언트가 공격적인 속도로 연결을 시도하므로 브로커가 스스로를 보호하고 있음을 나타냅니다. 브로커가 더 높은 속도의 새 IAM 연결을 수락하도록 하려면 [`reconnect.backoff.ms`](#) 구성 파라미터를 늘릴 수 있습니다.

브로커별 새 연결 속도 제한에 대한 자세한 내용은 [Amazon MSK 할당량](#) 페이지를 참조하세요.

인증 실패: 세션이 너무 짧음

이 `Failed authentication ... Session too short` 오류는 클라이언트가 곧 만료될 IAM 자격 증명을 사용하여 클러스터에 연결을 시도하는 경우 발생합니다. IAM 자격 증명이 새로 고쳐지는 방식을 확인해야 합니다. 대부분의 경우 자격 증명이 세션 만료에 너무 가깝게 교체되어 서버 측에 문제가 발생하고 인증에 실패할 수 있습니다.

MSK 서버리스: 클러스터 생성 실패

MSK 서버리스 클러스터를 생성하려고 하는데 워크플로우가 실패하는 경우 VPC 엔드포인트를 생성할 수 있는 권한이 없는 것일 수 있습니다. 관리자가 `ec2:CreateVpcEndpoint` 작업을 허용하여 VPC 엔드포인트를 만들 수 있는 권한을 부여했는지 확인합니다.

모든 Amazon MSK 작업을 수행하는 데 필요한 전체 권한 목록은 [AWS관리형 정책: AmazonMSKFullAccess](#) 섹션을 참조하세요.

MSK 구성에서 KafkaVersionsList를 업데이트할 수 없음

AWS::MSK::Configuration 리소스에서 KafkaVersionsList 속성을 업데이트하면 다음 오류와 함께 업데이트가 실패합니다.

```
Resource of type 'AWS::MSK::Configuration' with identifier '<identifierName>' already
 exists.
```

`KafkaVersionsList` 속성을 업데이트하면는 이전 구성을 삭제하기 전에 업데이트된 속성으로 새 구성을 AWS CloudFormation다시 생성합니다. 새 구성이 기존 구성과 동일한 이름을 사용하기 때문에 CloudFormation스택 업데이트가 실패합니다. 이러한 업데이트에는 리소스 교체가 필요합니다. `KafkaVersionsList`를 성공적으로 업데이트하려면 동일한 작업에서 이름 속성도 업데이트해야 합니다.

또한 또는를 사용하여 생성된 클러스터에 구성이 연결된 경우 AWS Management 콘솔AWS CLI구성 리소스에 다음을 추가하여 실패한 리소스 삭제 시도를 방지합니다.

```
UpdateReplacePolicy: Retain
```

업데이트가 성공하면 Amazon MSK 콘솔로 이동하여 이전 구성을 삭제합니다. MSK 구성에 대한 자세한 내용은 Amazon MSK Provisioned 구성 단원을 참조하십시오.

Standard 및 Express 브로커 모범 사례

이 섹션에서는 Standard 브로커 및 Express 브로커에 대해 따라야 할 모범 사례를 설명합니다. Amazon MSK Replicator 모범 사례에 대한 자세한 내용은 MSK Replicator 사용 모범 사례 단원을 참조하세요.

주제

Standard 브로커 모범 사례

이 주제에서는 Amazon MSK를 사용할 때 따라야 할 몇 가지 모범 사례를 간략하게 설명합니다. Amazon MSK Replicator 모범 사례에 대한 자세한 내용은 MSK Replicator 사용 모범 사례 단원을 참조하세요.

클라이언트 측 고려 사항

애플리케이션의 가용성과 성능은 서버 측 설정뿐만 아니라 클라이언트 설정에 따라서도 달라집니다.

- 고가용성을 위해 클라이언트를 구성합니다. Apache Kafka와 같은 분산 시스템에서는 고가용성을 보장하는 것이 신뢰할 수 있고 내결함성 있는 메시징 인프라를 유지하는 데 매우 중요합니다. 브로커는 업그레이드, 패치 적용, 하드웨어 장애, 네트워크 문제 등와 같은 계획된 이벤트와 계획되지 않은 이벤트 모두에 대해 오프라인 상태가 됩니다. Kafka 클러스터는 오프라인 브로커 오류를 처리할 수 있으므로 Kafka 클라이언트도 브로커 장애 조치를 정상적으로 처리해야 합니다. Apache Kafka 클라이언트의 모범 사례에 대한 전체 세부 정보를 참조하세요.
- 클라이언트 연결 문자열에 각 가용 영역의 브로커가 하나 이상 포함되어 있는지 확인합니다. 클라이언트의 연결 문자열에 여러 브로커가 있으면 특정 브로커가 업데이트를 위해 오프라인 상태일 때 장애 조치를 수행할 수 있습니다. 여러 브로커를 통해 연결 문자열을 가져오는 방법에 대한 자세한 내용은 Amazon MSK 클러스터를 위한 부트스트랩 브로커 가져오기 섹션을 참조하세요.
- 성능 테스트를 실행하여 클라이언트 구성에서 성능 목표를 충족할 수 있는지 확인합니다.

서버 측 고려 사항

클러스터 크기를 적절하게 조정: Standard 브로커당 파티션 수

다음 표는 Standard 브로커당 권장되는 파티션 수(리더 및 팔로어 복제본 포함)를 보여줍니다. 권장 파티션 수는 적용되지 않으며 프로비저닝된 모든 토픽 파티션에서 트래픽을 전송하는 시나리오에 가장 적합합니다.

브로커 크기	브로커당 권장 파티션 수(리더 및 팔로어 복제본 포함)	업데이트 작업을 지원하는 최대 파티션 수
`kafka.t3.small`	300	300
`kafka.m5.large` 'or' `kafka.m5.xlarge`	1000	1500
`kafka.m5.2xlarge`	2000	3000
`kafka.m5.4xlarge` `kafka.m5.8xlarge` ,`kafka.m5.`	4000	6000

브로커 크기	브로커당 권장 파티션 수(리더 및 팔로어 복제본 포함)	업데이트 작업을 지원하는 최대 파티션 수
12xlarge ,kafka.m5.16xlarge ,또는 kafka.m5.24xlarge		
kafka.m7g.large ‘or’kafka.m7g.xlarge	1000	1500
kafka.m7g.2xlarge	2000	3000
kafka.m7g.4xlarge , kafka.m7g.8xlarge , kafka.m7g.12xlarge 또는 kafka.m7g.16xlarge	4000	6000

파티션 수가 많고 처리량이 낮은 사용 사례가 있지만, 모든 파티션에서 트래픽을 전송하지 않는 경우 클러스터가 더 많은 파티션 수로 정상 상태를 유지하는지 확인하기 위해 충분한 테스트 및 성능 테스트를 수행했다면 브로커당 더 많은 파티션을 패키징할 수 있습니다. 브로커당 파티션 수가 최대 허용 값을 초과하여 클러스터에 과부하가 주어지면 다음 작업을 수행하지 못할 수 있습니다.

- 클러스터 구성 업데이트
- 더 작은 브로커 크기로 클러스터 업데이트
- AWS Secrets Manager 암호를 SASL/SCRAM 인증이 있는 클러스터와 연결

파티션 수가 많으면 CloudWatch 및 Prometheus 스크레이핑에서 Kafka 지표가 누락될 수도 있습니다.

파티션 수를 선택하는 방법에 대한 지침은 Apache Kafka는 클러스터당 200K 파티션 지원을 참조하세요. 또한 직접 테스트를 수행하여 자신에게 적합한 브로커 크기를 결정하는 것을 권장합니다. 다양한 브로커 크기에 대한 자세한 내용은 the section called “브로커 유형” 단원을 참조하세요.

클러스터 크기를 적절하게 조정: 클러스터당 Standard 브로커 수

MSK Provisioned 클러스터에 적합한 Standard 브로커 수를 결정하고 비용을 이해하려면 MSK 크기 조정 및 요금 스프레드시트를 참조하세요. 이 스프레드시트는 유사한 자체 관리형 EC2 기반 Apache Kafka 클러스터와 비교하여 MSK Provisioned 클러스터의 크기 조정 및 관련 비용에 대한 추정치를 제

공합니다. 스프레드시트의 입력 파라미터에 대한 자세한 내용을 보려면 파라미터 설명 위에 커서를 대십시오. 이 시트에서 제공하는 추정치는 보수적인 수치이며 새로운 MSK Provisioned 클러스터를 위한 시작점을 제공합니다. 클러스터 성능, 크기, 비용은 사용 사례에 따라 다르므로 실제 테스트를 통해 확인하는 것이 좋습니다.

기본 인프라가 Apache Kafka 성능에 어떤 영향을 미치는지 이해하려면 AWS 빅 데이터 블로그에서 Best practices for right-sizing your Apache Kafka clusters to optimize performance and cost를 참조하세요. 이 블로그 게시물에서는 처리량, 가용성, 지연 시간 요구 사항을 충족하기 위해 클러스터 크기를 조정하는 방법에 대해 설명합니다. 또한 스케일 업 및 스케일 아웃이 필요한 시기 등의 질문에 대한 답변과 프로덕션 클러스터의 크기를 지속적으로 확인하는 방법에 대한 지침을 제공합니다. 계층형 스토리지 기반 클러스터에 대한 자세한 내용은 Amazon MSK 계층형 스토리지를 사용하여 프로덕션 워크로드를 실행하는 모범 사례를 참조하세요.

m5.4xl, m7g.4xl 또는 더 큰 인스턴스에 대한 클러스터 처리량 최적화

m5.4xl, m7g.4xl 또는 더 큰 인스턴스를 사용하는 경우 num.io.threads 및 num.network.threads 구성을 조정하여 MSK Provisioned 클러스터 처리량을 최적화할 수 있습니다.

Num.io.threads는 Standard 브로커가 요청을 처리하는 데 사용하는 스레드 수입니다. 인스턴스 크기에 지원되는 CPU 코어 수까지 스레드를 더 추가하면 클러스터 처리량을 개선하는 데 도움이 될 수 있습니다.

Num.network.threads는 Standard 브로커가 수신되는 모든 요청을 수신하고 응답을 반환하는 데 사용하는 스레드 수입니다. 네트워크 스레드는 들어오는 요청을 요청 대기열에 배치하여 io.threads에서 처리합니다. num.network.threads를 인스턴스 크기에 지원되는 CPU 코어 수의 절반으로 설정하면 새 인스턴스 크기를 완전하게 사용할 수 있습니다.

> **Important**
>
> 대기열 포화로 인한 혼잡이 발생할 수 있으므로 num.io.threads를 늘리기 전에 num.network.threads를 늘리지 마세요.

다음 표에서는 각 인스턴스 크기에 대한 권장 설정을 설명합니다.

인스턴스 크기	num.io.threads의 권장값	num.network.threads의 권장값
m5.4xl	16	8

인스턴스 크기	num.io.threads의 권장값	num.network.threads의 권장값
m5.8xl	32	16
m5.12xl	48	24
m5.16xl	64	32
m5.24xl	96	48
m7g.4xlarge	16	8
m7g.8xlarge	32	16
m7g.12xlarge	48	24
m7g.16xlarge	64	32

최신 Kafka AdminClient를 사용하여 주제 ID 불일치 문제를 방지합니다.

2.8.0보다 낮은 Kafka AdminClient 버전과 `--zookeeper` 플래그를 사용하여 Kafka 버전 2.8.0 이상을 사용하는 MSK Provisioned 클러스터의 토픽 파티션을 늘리거나 재할당할 때 토픽의 ID가 손실됩니다 (오류: 파티션의 토픽 ID와 일치하지 않음). `--zookeeper` 플래그는 Kafka 2.5에서 더 이상 사용되지 않으며 Kafka 3.0부터 제거된다는 점에 유의하세요. [0.8.x~2.4.x 버전에서 2.5.0으로 업그레이드](#)를 참조하세요.

주제 ID 불일치를 방지하려면 Kafka 관리자 작업에 Kafka 클라이언트 버전 2.8.0 이상을 사용하세요. 또는 클라이언트 2.5 이상에서는 `--zookeeper` 플래그 대신 `--bootstrap-servers` 플래그를 사용할 수 있습니다.

고가용성 클러스터 빌드

다음 권장 사항을 사용하여 업데이트 중(예: 브로커 크기 또는 Apache Kafka 버전을 업데이트하는 경우) 또는 Amazon MSK가 브로커를 교체하는 동안 MSK Provisioned 클러스터의 고가용성을 유지할 수 있습니다.

- 3개의 AZ 클러스터를 설정합니다.
- 복제 인수(RF)가 3 이상인지 확인합니다. RF가 1이면 롤링 업데이트 중에 오프라인 파티션이 발생할 수 있고 RF가 2이면 데이터 손실이 발생할 수 있다는 점에 유의하세요.

- 최소 동기화 복제본(minISR)을 최대 RF - 1로 설정합니다. RF와 동일한 minISR은 롤링 업데이트 중에 클러스터 생성을 방해할 수 있습니다. 2개의 minISR을 사용하면 하나의 복제본이 오프라인 상태일 때 3방향 복제된 항목이 가능합니다.

CPU 사용량 모니터링

Amazon MSK는 브로커의 총 CPU 사용률(`CPU User + CPU System`으로 정의됨)을 60% 미만으로 유지할 것을 강력하게 권장합니다. 이를 통해 클러스터가 브로커 장애, 패치 적용 및 롤링 업그레이드와 같은 운영 이벤트를 처리하기에 충분한 CPU 헤드룸을 유지할 수 있습니다.

Apache Kafka는 필요한 경우 클러스터의 브로커 간에 CPU 로드를 재배포할 수 있습니다. 예를 들어 Amazon MSK가 브로커 결함을 감지하고 복구하면 패치 적용과 같은 자동 유지 관리를 수행합니다. 마찬가지로, 사용자가 브로커 크기 변경 또는 버전 업그레이드를 요청하는 경우 Amazon MSK는 한 번에 하나의 브로커를 오프라인으로 전환하는 롤링 워크플로를 시작합니다. 리드 파티션이 있는 브로커가 오프라인 상태가 되면 Apache Kafka는 파티션 책임자를 재할당하여 클러스터 내 다른 브로커에게 작업을 재분배합니다. 이 모범 사례를 따르면 클러스터에 이와 같은 운영 이벤트를 견딜 수 있는 충분한 CPU 헤드룸을 확보할 수 있습니다.

> **Note**
>
> CPU 사용률을 모니터링하는 동안 총 CPU 사용량에는 `CPU User` 및 `CPU System` 이상이 포함됨을 유의하세요. `iowait`, `irq`, `softirq` 및 `steal`과 같은 다른 범주도 전체 CPU 활동에 기여합니다. 따라서 CPU 유휴가 항상 *`100% - CPU User - CPU System`*과 같지는 않습니다.

Amazon CloudWatch Metric Math를 사용하여 복합 지표(`CPU User + CPU System`)를 생성하고 평균 사용량이 60%를 초과할 때 트리거하도록 경보를 설정할 수 있습니다. 이 경보가 트리거되면 다음 옵션 중 하나를 사용하여 클러스터를 확장합니다.

- 옵션 1(권장): 다음으로 큰 크기로 브로커 크기를 업데이트합니다. 예를 들어 현재 크기가 `kafka.m5.large`인 경우 `kafka.m5.xlarge`를 사용하도록 클러스터를 업데이트합니다. 클러스터에서 브로커 크기를 업데이트하면 Amazon MSK는 롤링 방식으로 브로커를 오프라인 상태로 전환하고 파티션 리더십을 다른 브로커에게 일시적으로 재할당한다는 점에 유의하세요. 크기 업데이트는 일반적으로 브로커당 10~15분 정도 소요됩니다.
- 옵션 2: 생산자로부터 수집된 모든 메시지가 라운드 로빈 쓰기를 사용하는 주제가 있는 경우(즉, 메시지에 키가 지정되지 않고 순서가 소비자에게 중요하지 않은 경우) 브로커를 추가하여 클러스터를 확장합니다. 또한 처리량이 가장 많은 기존 주제에 파티션을 추가할 수도 있습니다. 그런 다음

`kafka-topics.sh --describe`를 사용하여 새로 추가된 파티션이 새 브로커에 할당되었는지 확인합니다. 이전 옵션에 비해 이 옵션의 가장 큰 이점은 리소스와 비용을 더 세부적으로 관리할 수 있다는 점입니다. 또한 이러한 형태의 규모 조정은 일반적으로 기존 브로커의 부하를 증가시키지 않으므로 CPU 부하가 60%를 크게 초과하는 경우 이 옵션을 사용할 수 있습니다.

- 옵션 3: 브로커를 추가하여 클러스터를 확장한 다음 `kafka-reassign-partitions.sh`라는 이름의 파티션 재할당 도구를 사용하여 기존 파티션을 재할당합니다. 그러나 이 옵션을 사용하면 파티션이 재할당된 후 클러스터가 브로커 간에 데이터를 복제하는 데 리소스를 사용해야 합니다. 이전 두 옵션에 비해 처음에는 클러스터의 부하가 크게 증가할 수 있습니다. 따라서 Amazon MSK는 복제로 인해 추가 CPU 부하 및 네트워크 트래픽이 발생하여 CPU 사용률이 70%를 초과하는 경우에는 이 옵션을 사용하지 않는 것을 권장합니다. Amazon MSK는 앞의 두 가지 옵션을 사용할 수 없는 경우에만 이 옵션 사용을 권장합니다.

기타 권장 사항:

- 부하 분산을 위한 프록시로 브로커당 총 CPU 사용률을 모니터링합니다. 브로커의 CPU 사용률이 지속적으로 고르지 않다면 클러스터 내에서 부하가 고르게 분산되지 않았다는 신호일 수 있습니다. Cruise Control을 사용하여 파티션 할당을 통한 부하 분산을 지속적으로 관리할 것을 권장합니다.
- 생산 및 소비 지연을 모니터링합니다. 생산 및 소비 지연 시간은 CPU 사용률에 따라 선형적으로 증가할 수 있습니다.
- JMX 스크레입 간격: Prometheus 기능으로 개방형 모니터링을 사용하도록 설정하는 경우 Prometheus 호스트 구성(prometheus.yml)에 60초 이상의 스크레입 간격(scrape_interval: 60초)을 사용하는 것을 권장합니다. 스크랩 간격을 줄이면 클러스터의 CPU 사용량이 늘어날 수 있습니다.

디스크 공간 모니터링

메시지를 위한 디스크 공간이 부족하지 않도록 하려면 `KafkaDataLogsDiskUsed` 지표를 감시하는 CloudWatch 경보를 생성합니다. 이 지표의 값이 85% 이상에 도달하면 다음 작업 중 하나 이상을 수행합니다.

- 를 사용합니다.the section called “클러스터의 자동 규모 조정” the section called “수동 조정”에 설명된 대로 브로커 스토리지를 수동으로 늘릴 수도 있습니다.
- 메시지 보존 기간 또는 로그 크기를 줄입니다. 이렇게 하는 방법에 대한 정보는 the section called “데이터 보존 파라미터 조정” 섹션을 참조하세요.
- 사용되지 않는 항목을 삭제합니다.

경보를 설정하고 사용하는 방법에 대한 자세한 내용은 [Amazon CloudWatch 경보 사용]을 참조하세요. Amazon MSK 지표의 전체 목록은 [the section called “클러스터 모니터링”] 섹션을 참조하세요.

데이터 보존 파라미터 조정

메시지를 소비해도 로그에서 제거되지 않습니다. 정기적으로 디스크 공간을 확보하려면 메시지가 로그에 유지되는 기간인 보존 기간을 명시적으로 지정하면 됩니다. 보존 로그 크기를 지정할 수도 있습니다. 보존 기간 또는 보존 로그 크기에 도달하면 Apache Kafka는 로그에서 비활성 세그먼트를 제거하기 시작합니다.

클러스터 수준에서 보존 정책을 지정하려면, `log.retention.hours`, `log.retention.minutes`, `log.retention.ms` 또는 `log.retention.bytes` 파라미터 중 하나 이상을 설정합니다. 자세한 내용은 [the section called “사용자 지정 Amazon MSK 구성”] 섹션을 참조하세요.

주제 수준에서 보존 파라미터를 지정할 수도 있습니다.

- 주제별로 보존 기간을 지정하려면 다음 명령을 사용합니다.

```
kafka-configs.sh --bootstrap-server $bs --alter --entity-type topics --entity-
name TopicName --add-config retention.ms=DesiredRetentionTimePeriod
```

- 주제별로 보존 로그 크기를 지정하려면 다음 명령을 사용합니다.

```
kafka-configs.sh --bootstrap-server $bs --alter --entity-type topics --entity-
name TopicName --add-config retention.bytes=DesiredRetentionLogSize
```

주제 수준에서 지정하는 보존 파라미터는 클러스터 수준 파라미터보다 우선합니다.

비정상 종료 후 로그 복구 속도 향상

비정상 종료 후 브로커는 로그 복구를 수행하므로 다시 시작하는 데 시간이 걸릴 수 있습니다. 기본적으로 Kafka는 로그 디렉터리당 단일 스레드만 사용하여 해당 복구를 수행합니다. 예를 들어 수천 개의 파티션이 있는 경우 로그 복구를 완료하는 데 몇 시간이 소요될 수 있습니다. 로그 복구 속도를 높이려면 구성 속성 [`num.recovery.threads.per.data.dir`]을 사용하여 스레드 수를 늘리는 것을 권장합니다. CPU 코어 수로 설정할 수 있습니다.

Apache Kafka 메모리 모니터링

Apache Kafka가 사용하는 메모리를 모니터링하는 것을 권장합니다. 그렇지 않으면 클러스터를 사용할 수 없게 될 수 있습니다.

Apache Kafka가 사용하는 메모리 양을 확인하려면 `HeapMemoryAfterGC` 지표를 모니터링하면 됩니다. `HeapMemoryAfterGC`는 가비지 수집 후 사용 중인 전체 힙 메모리의 백분율입니다. `HeapMemoryAfterGC`가 60% 이상 증가하면 조치를 취하는 CloudWatch 경보를 생성하는 것이 좋습니다.

메모리 사용량을 줄이기 위해 취할 수 있는 조치는 다양합니다. Apache Kafka를 구성하는 방식에 따라 달라집니다. 예를 들어 트랜잭션 메시지 전송을 사용하는 경우 Apache Kafka 구성에서 `transactional.id.expiration.ms` 값을 `604800000`밀리초에서 `86400000`밀리초(7일에서 1일로)로 줄일 수 있습니다. 이를 통해 각 트랜잭션의 메모리 사용량이 줄어듭니다.

MSK 이외의 브로커 추가 금지

ZooKeeper 기반 클러스터의 경우 Apache ZooKeeper 명령을 사용하여 브로커를 추가하면 해당 브로커는 MSK 클러스터에 추가되지 않으며 Apache ZooKeeper에는 클러스터에 대한 잘못된 정보가 포함됩니다. 이로 인해 데이터가 손실될 수 있습니다. 지원되는 MSK 프로비저닝된 클러스터 작업은 the section called “주요 기능 및 개념” 섹션을 참조하세요.

전송 중 암호화 활성화

전송 중 암호화와 활성화하는 방법에 대한 자세한 내용은 the section called “전송 중 Amazon MSK 암호화” 섹션을 참조하세요.

파티션 재할당

파티션을 동일한 MSK Provisioned 클러스터에 있는 다른 브로커로 이동하기 위해 `kafka-reassign-partitions.sh`라는 파티션 재할당 도구를 사용할 수 있습니다. 안전한 작업을 위해 한 번의 `kafka-reassign-partitions` 직접 호출로 10개 이상의 파티션을 재할당하지 않는 것이 좋습니다. 예를 들어, 클러스터를 확장하거나 브로커를 제거하기 위해 파티션을 이동하도록 새로운 브로커를 추가한 후에는 새로운 브로커에 파티션을 다시 할당하여 해당 클러스터를 재분배할 수 있습니다. MSK Provisioned 클러스터에 브로커를 추가하는 방법에 대한 자세한 내용은 the section called “클러스터 확장” 섹션을 참조하세요. MSK Provisioned 클러스터에서 브로커를 제거하는 방법에 대한 자세한 내용은 the section called “브로커 제거” 섹션을 참조하세요. 파티션 재할당 도구에 대한 자세한 내용은 Apache Kafka 설명서의 클러스터 확장을 참조하세요.

Express 브로커 모범 사례

이 주제에서는 Express 브로커를 사용할 때 따라야 할 몇 가지 모범 사례에 대해 간략하게 설명합니다. Express 브로커는 고가용성 및 내구성을 위해 사전 구성되어 제공됩니다. 사용자의 데이터는 기본적으로 3개의 가용 영역에 분산되며 복제는 항상 3으로 설정되고 최소 동기화된 복제본은 항상 2로 설정됩니다. 그러나 클러스터의 신뢰성 및 성능을 최적화하기 위해 고려해야 할 몇 가지 요소가 있습니다.

클라이언트 측 고려 사항

애플리케이션의 가용성과 성능은 서버 측 설정뿐만 아니라 클라이언트 설정에 따라서도 달라집니다.

- 고가용성을 위해 클라이언트를 구성합니다. Apache Kafka와 같은 분산 시스템에서는 고가용성을 보장하는 것이 신뢰할 수 있고 내결함성 있는 메시징 인프라를 유지하는 데 매우 중요합니다. 브로커는 업그레이드, 패치 적용, 하드웨어 장애, 네트워크 문제 등와 같은 계획된 이벤트와 계획되지 않은 이벤트 모두에 대해 오프라인 상태가 됩니다. Kafka 클러스터는 오프라인 브로커 오류를 처리할 수 있으므로 Kafka 클라이언트도 브로커 장애 조치를 정상적으로 처리해야 합니다. Apache Kafka 클라이언트에 대한 모범 사례 권장 사항의 전체 세부 정보를 참조하세요.
- 성능 테스트를 실행하여 클라이언트 구성이 최대 부하 상태에서 브로커를 다시 시작할 때도 성능 목표를 충족할 수 있는지 확인합니다. MSK 콘솔에서 또는 MSK API를 사용하여 클러스터의 브로커를 재부팅할 수 있습니다.

서버 측 고려 사항

주제

- 클러스터 크기를 적절하게 조정: 클러스터당 브로커 수
- CPU 사용량 모니터링
- 클러스터 크기 조정: Express 브로커당 파티션 수
- 연결 수 모니터링
- 파티션 재할당

클러스터 크기를 적절하게 조정: 클러스터당 브로커 수

Express 기반 클러스터의 브로커 수는 쉽게 선택할 수 있습니다. 각 Express 브로커에는 수신 및 송신에 대해 정의된 처리량 용량이 제공됩니다. 이 처리량 용량을 클러스터 크기 조정의 기본 수단으로 사용해야 합니다(그런 다음 아래에 설명된 파티션 및 연결 수와 같은 다른 요소 고려).

예를 들어 스트리밍 애플리케이션에 45MBps의 데이터 수신(쓰기)과 90MBps의 데이터 송신(읽기) 용량이 필요한 경우 3개의 express.m7g.large 브로커를 사용하여 처리량 요구 사항을 충족할 수 있습니다. 각 express.m7g.large 브로커는 수신 15MBps, 송신은 30MBps로 처리합니다. 각 Express 브로커 크기에 대한 권장 처리량 한도는 다음 표를 참조하세요. 처리량이 권장 한도를 초과하는 경우 성능이 저하될 수 있으므로 트래픽을 줄이거나 클러스터를 확장해야 합니다. 처리량이 권장 한도를 초과하고 브로커당 할당량에 도달하는 경우 MSK는 클라이언트 트래픽을 제한하여 추가 과부하를 방지합니다.

MSK 크기 조정 및 요금 스프레드시트를 사용하여 여러 시나리오를 평가하고 파티션 수와 같은 다른 요소를 고려할 수도 있습니다.

다음 표에는 각 인스턴스 크기에 대한 브로커당 권장 최대 처리량이 나열되어 있습니다.

인스턴스 크기	수신(MBps)	송신(MBps)
`express.m7g.large`	15.6	31.2
`express.m7g.xlarge`	31.2	62.5
`express.m7g.2xlarge`	62.5	125.0
`express.m7g.4xlarge`	124.9	249.8
`express.m7g.8xlarge`	250.0	500.0
`express.m7g.12xlarge`	375.0	750.0
`express.m7g.16xlarge`	500.0	1000.0

CPU 사용량 모니터링

브로커의 총 CPU 사용률(CPU 사용자 + CPU 시스템으로 정의됨)을 60% 미만으로 유지할 것을 권장합니다. 클러스터 총 CPU의 40% 이상을 사용할 수 있는 경우 Apache Kafka는 필요한 경우 클러스터의 브로커 간에 CPU 부하를 재분배할 수 있습니다. 이는 계획되거나 계획되지 않은 이벤트로 인해 필요할 수 있습니다. 계획된 이벤트의 예로는 MSK가 클러스터의 브로커를 한 번에 하나씩 다시 시작하여 업데이트하는 클러스터 버전 업그레이드가 있습니다. 계획되지 않은 이벤트의 예로는 브로커의 하드웨어 장애 또는 최악의 경우 AZ의 모든 브로커가 영향을 받는 AZ 장애가 있습니다. 파티션 리드 복제본이 있는 브로커가 오프라인 상태가 되면 Apache Kafka는 파티션 리더십을 재할당하여 클러스터 내 다른 브로커에 작업을 재분배합니다. 이 모범 사례를 따르면 클러스터에 이와 같은 운영 이벤트를 견딜 수 있는 충분한 CPU 헤드룸을 확보할 수 있습니다.

Amazon CloudWatch 사용 설명서의 CloudWatch 지표와 함께 수학 표현식 사용을 통해 CPU 사용자 + CPU 시스템인 복합 지표를 생성할 수 있습니다. 복합 지표가 평균 CPU 사용률 60%에 도달할 때 트리거되는 경보를 설정합니다. 이 경보가 트리거되면 다음 옵션 중 하나를 사용하여 클러스터를 확장합니다.

- 옵션 1: 한 단계 위의 큰 크기로 브로커 크기를 업데이트합니다. 클러스터에서 브로커 크기를 업데이트하면 Amazon MSK는 롤링 방식으로 브로커를 오프라인 상태로 전환하고 파티션 리더십을 다른 브로커에게 일시적으로 재할당한다는 점에 유의하세요.
- 옵션 2: 브로커를 추가하여 클러스터를 확장한 다음 `kafka-reassign-partitions.sh`라는 이름의 파티션 재할당 도구를 사용하여 기존 파티션을 재할당합니다.

기타 권장 사항

- 부하 분산을 위한 프록시로 브로커당 총 CPU 사용률을 모니터링합니다. 브로커의 CPU 사용률이 지속적으로 고르지 않다면 클러스터 내에서 부하가 고르게 분산되지 않았다는 신호일 수 있습니다. Cruise Control을 사용하여 파티션 할당을 통한 부하 분산을 지속적으로 관리할 것을 권장합니다.
- 생산 및 소비 지연을 모니터링합니다. 생산 및 소비 지연 시간은 CPU 사용률에 따라 선형적으로 증가할 수 있습니다.
- JMX 스크레이프 간격: Prometheus 기능으로 개방형 모니터링을 사용하도록 설정하는 경우 Prometheus 호스트 구성(`prometheus.yml`)에 60초 이상의 스크레이프 간격(`scrape_interval: 60s`)을 사용하는 것을 권장합니다. 스크랩 간격을 줄이면 클러스터의 CPU 사용량이 늘어날 수 있습니다.

클러스터 크기 조정: Express 브로커당 파티션 수

파티션 사용량이 많고 처리량이 낮은 사용 사례가 있지만 모든 파티션에 트래픽을 전송하지 않는 경우, 클러스터가 더 높은 파티션 수에서도 정상 상태를 유지하는지 확인하기 위해 충분한 테스트 및 성능 테스트를 수행했다는 전제 하에 브로커당 더 많은 파티션을 패키징할 수 있습니다. 브로커당 파티션 수가 최대 허용 값을 초과하여 클러스터에 과부하가 주어지면 다음 작업을 수행하지 못하게 될 수 있습니다.

- 클러스터 구성 업데이트
- 더 작은 브로커 크기로 클러스터 업데이트
- SASL/SCRAM 인증이 있는 클러스터와 AWS Secrets Manager보안 암호 연결

파티션 수가 많아 클러스터가 과부하되면 CloudWatch 및 Prometheus 스크래핑에서 Kafka 지표가 누락될 수도 있습니다.

파티션 수를 선택하는 방법에 대한 지침은 Apache Kafka는 클러스터당 200K 파티션 지원을 참조하세요. 또한 직접 테스트를 수행하여 자신에게 적합한 브로커 크기를 결정하는 것을 권장합니다. 다양한 브로커 크기에 대한 자세한 내용은 Amazon MSK 브로커 크기 단원을 참조하세요.

각 Express 브로커에 권장되는 파티션 수(리더 및 팔로어 복제본 포함)에 대한 자세한 내용은 Express 브로커 파티션 할당량 섹션을 참조하세요. 권장되는 파티션 수는 강제로 적용되지 않으며 프로비저닝된 모든 토픽 파티션에서 트래픽을 전송하는 시나리오에 가장 적합합니다.

연결 수 모니터링

브로커에 대한 클라이언트 연결은 메모리 및 CPU와 같은 시스템 리소스를 사용합니다. 인증 메커니즘에 따라 모니터링하여 적용 가능한 한도 내에 있는지 확인해야 합니다. 실패한 연결에 대한 재시도를 처리하려면 클라이언트 측에서 `reconnect.backoff.ms` 구성 파라미터를 설정하면 됩니다. 예를 들어 클라이언트가 1초 후에 연결을 다시 시도하도록 하려면 `reconnect.backoff.ms`를 `1000`으로 설정합니다. 재시도 구성에 대한 자세한 내용은 Apache Kafka 설명서를 참조하세요.

측정 기준	할당량
브로커당 최대 TCP 연결 수(IAM 액세스 제어)	3000
브로커당 최대 TCP 연결 수(IAM)	100/초
브로커당 최대 TCP 연결 수(비 IAM)	MSK는 비 IAM 인증에 연결 한도를 강제로 적용하지 않습니다. 그러나 CPU 및 메모리 사용량과 같은 다른 지표를 모니터링하여 과도한 연결로 인해 클러스터에 과부하가 걸리지 않도록 해야 합니다.

파티션 재할당

파티션을 동일한 MSK Provisioned 클러스터에 있는 다른 브로커로 이동하기 위해 `kafka-reassign-partitions.sh`라는 파티션 재할당 도구를 사용할 수 있습니다. 안전한 작업을 위해 한 번의 `kafka-reassign-partitions` 직접 호출로 20개 이상의 파티션을 재할당하지 않는 것이 좋습니다. 예를 들어, 클러스터를 확장하거나 브로커를 제거하기 위해 파티션을 이동하도록 새로운 브로커를 추가한 후에는 새로운 브로커에 파티션을 다시 할당하여 해당 클러스터를 재분배할 수 있습니다. MSK Provisioned 클러스터에 브로커를 추가하는 방법에 대한 자세한 내용은 the section called “클러스터 확장” 섹션을 참조하세요. MSK Provisioned 클러스터에서 브로커를 제거하는 방법에 대한 자세한 내용은 the section called “브로커 제거” 섹션을 참조하세요. 파티션 재할당 도구에 대한 자세한 내용은 Apache Kafka 설명서의 클러스터 확장을 참조하세요.

Apache Kafka 클라이언트의 모범 사례

Apache Kafka 및 Amazon MSK에서 작업할 때는 최적의 성능과 신뢰성을 위해 클라이언트와 서버를 올바르게 구성하는 것이 중요합니다. 이 가이드에서는 Amazon MSK에 대한 모범 사례 클라이언트 측 구성에 대한 권장 사항을 제공합니다.

Amazon MSK Replicator 모범 사례에 대한 자세한 내용은 MSK Replicator 사용 모범 사례 단원을 참조하세요. Standard 및 Express 브로커 모범 사례는 Standard 및 Express 브로커 모범 사례 섹션을 참조하세요.

주제

Apache Kafka 클라이언트 가용성

Apache Kafka와 같은 분산 시스템에서는 고가용성을 보장하는 것이 신뢰할 수 있고 내결함성 있는 메시징 인프라를 유지하는 데 매우 중요합니다. 브로커는 업그레이드, 패치 적용, 하드웨어 장애, 네트워크 문제 등와 같은 예정된 이벤트와 예정되지 않은 이벤트 모두에 대해 오프라인 상태가 됩니다. Kafka 클러스터는 오프라인 브로커 오류를 처리할 수 있으므로 Kafka 클라이언트도 브로커 장애 조치를 정상적으로 처리해야 합니다. Kafka 클라이언트의 고가용성을 보장하려면 다음 모범 사례를 사용하는 것이 좋습니다.

생산자 가용성

- 브로커 장애 조치 중에 실패한 메시지 전송을 다시 시도하도록 생산자에게 지시하도록 `retries`를 설정합니다. 대부분의 사용 사례에는 최대 정수 값 또는 이와 유사한 높은 값을 사용하는 것이 좋습니다. 이렇게 하지 못하면 Kafka의 고가용성이 저하됩니다.
- 메시지를 보내는 시간과 브로커로부터 확인을 받는 시간 사이의 총 시간에 대한 상한을 지정하도록 `delivery.timeout.ms`를 설정합니다. 이는 메시지가 유효한 기간의 비즈니스 요구 사항을 반영해야 합니다. 장애 조치 작업을 완료하는 데 충분한 재시도를 허용할 수 있도록 시간 제한을 충분히 높게 설정합니다. 대부분의 사용 사례에는 60초 이상의 값을 사용하는 것이 좋습니다.
- 재전송을 시도하기 전에 단일 요청이 기다려야 하는 최대 값으로 `request.timeout.ms`를 설정합니다. 대부분의 사용 사례에는 10초 이상의 값을 사용하는 것이 좋습니다.

- 재시도 급증 및 가용성 영향을 방지하기 위해 재시도 간의 지연을 구성하도록 `retry.backoff.ms`를 설정합니다. 대부분의 사용 사례에는 200ms의 최소값을 사용하는 것이 좋습니다.
- 높은 내구성을 구성하도록 `acks=all`을 설정합니다. 이는 ISR의 모든 파티션이 쓰기를 승인하도록 하려면 `RF=3` 및 `min.isr=2`의 서버 측 구성과 일치해야 합니다. 단일 브로커 오프라인 중에 이는 `min.isr`, 즉 `2`입니다.

소비자 가용성

- 새로운 소비자 그룹 또는 재생성된 소비자 그룹에 대해 `auto.offset.reset`을 처음에 `latest`로 설정합니다. 이렇게 하면 전체 주제를 소비하여 클러스터 부하가 추가될 위험을 피할 수 있습니다.
- `enable.auto.commit`를 사용할 때 `auto.commit.interval.ms`를 설정합니다. 대부분의 사용 사례에서는 추가 로드 위험을 방지하기 위해 최소 5초의 값을 사용하는 것이 좋습니다.
- 소비자의 메시지 처리 코드 내에서 예외 처리를 구현하여 회로 차단기 또는 지수 백오프가 있는 절전 모드와 같은 일시적인 오류를 처리합니다. 이렇게 하지 않으면 애플리케이션 충돌이 발생하여 과도한 재조정이 발생할 수 있습니다.
- `isolation.level`을 설정하여 트랜잭션 메시지를 읽는 방법을 제어합니다.

  기본적으로 `read_uncommitted`를 항상 간결하게 설정하는 것이 좋습니다. 이는 일부 클라이언트 구현에서 누락됩니다.

  계층형 스토리지를 사용할 때는 `read_uncommitted`의 값을 사용하는 것이 좋습니다.
- 가장 가까운 복제본 읽기를 사용하도록 `client.rack`을 설정합니다. 네트워크 트래픽 비용과 지연 시간을 최소화하도록 `az id` 를 설정하는 것이 좋습니다. [랙 인식을 통해 Amazon MSK 소비자의 네트워크 트래픽 비용 절감]을 참조하세요.

소비자 재조정

- 구현된 시작 지터를 포함하여 애플리케이션의 시작 시간보다 큰 값으로 `session.timeout.ms`를 설정합니다. 대부분의 사용 사례에는 60초의 값을 사용하는 것이 좋습니다.
- 그룹 조정자가 소비자를 정상으로 보는 방법을 미세 조정하도록 `heartbeat.interval.ms`를 설정합니다. 대부분의 사용 사례에는 10초의 값을 사용하는 것이 좋습니다.
- 세션 제한 시간을 사용하여 소비자가 그룹을 떠날 때를 식별하는 대신 애플리케이션에서 종료 후크를 설정하여 SIGTERM에서 소비자를 확실하게 닫습니다. Kstream 애플리케이션은 `internal.leave.group.on.close`를 `true`의 값으로 설정할 수 있습니다.

- 소비자 그룹 내에서 고유한 값으로 `group.instance.id`를 설정합니다. 이상적으로는 호스트 이름, task-id 또는 pod-id가 적합합니다. 문제 해결 중에 더 결정적인 동작과 더 나은 클라이언트/서버 로그 상관관계를 위해 항상 이 설정을 지정하는 것이 좋습니다.
- `group.initial.rebalance.delay.ms`를 평균 배포 시간과 일치하는 값으로 설정합니다. 이렇게 하면 배포 중에 지속적인 재조정이 중지됩니다.
- 고정 할당자를 사용하도록 `partition.assignment.strategy`를 설정합니다. `StickyAssignor` 또는 `CooperativeStickyAssignor`를 사용하는 것이 좋습니다.

Apache Kafka 클라이언트 성능

Kafka 클라이언트의 고성능을 보장하려면 다음 모범 사례를 사용하는 것이 좋습니다.

생산자 성능

- 배치가 채워질 때까지 생산자가 기다리는 시간을 제어하도록 `linger.ms`를 설정합니다. 더 작은 배치는 한 번에 더 많은 스레드 및 I/O 작업으로 변환되므로 Kafka에 컴퓨팅 비용이 많이 듭니다. 다음 값을 사용하는 것이 좋습니다:

  지연 시간이 짧은 모든 사용 사례에 대한 최소값은 5밀리초입니다.

  대부분의 사용 사례에서는 25밀리초의 더 높은 값을 사용하는 것이 좋습니다.

  지연 시간이 짧은 사용 사례에서는 0의 값을 사용하지 않는 것이 좋습니다. (0의 값을 사용하면 일반적으로 IO 오버헤드로 인해 지연 시간이 발생합니다.)
- 클러스터로 전송된 배치 크기를 제어하도록 `batch.size`를 설정합니다. 이 값을 64KB 또는 128KB의 값으로 늘리는 것이 좋습니다.
- 더 큰 배치 크기를 사용할 때에는 `buffer.memory`를 설정합니다. 대부분의 사용 사례에는 64MB의 값을 사용하는 것이 좋습니다.
- 바이트 수신에 사용되는 TCP 버퍼를 제어하도록 `send.buffer.bytes`를 설정합니다. 지연 시간이 긴 네트워크에서 생산자를 실행할 때 OS에서 이 버퍼를 관리하도록 하려면 -1 값을 사용하는 것이 좋습니다.
- 배치의 압축을 제어하도록 compression.type을 설정합니다. 지연 시간이 긴 네트워크에서 생산자를 실행하는 lz4 또는 zstd를 사용하는 것이 좋습니다.

소비자 성능

- 최소 가져오기 크기를 제어하여 가져오기 및 클러스터 부하 수를 유효하게 줄이도록 `fetch.min.bytes`를 설정합니다.

  모든 사용 사례에 대해 32바이트의 최소값을 사용하는 것이 좋습니다.

  대부분의 사용 사례에서는 128바이트의 더 높은 값을 사용하는 것이 좋습니다.
- fetch.min.bytes가 무시되기 전까지 소비자가 대기하는 시간을 결정하도록 fetch.max.wait.ms를 설정합니다. 대부분의 사용 사례에서는 1,000밀리초의 값을 사용하는 것이 좋습니다.
- 더 나은 병렬 처리와 복원력을 위해 컨슈머 수는 최소한 파티션 수와 같도록 지정하는 것이 좋습니다. 경우에 따라 처리량이 적은 토픽에 대해 파티션 수보다 적은 컨슈머를 선택할 수 있습니다.
- 바이트 수신에 사용되는 TCP 버퍼를 제어하도록 `receive.buffer.bytes`를 설정합니다. 대기 시간이 긴 네트워크에서 소비자를 실행할 때 OS에서 이 버퍼를 관리하도록 하려면 -1 값을 사용하는 것이 좋습니다.

클라이언트 연결

연결 수명 주기에는 Kafka 클러스터에 대한 계산 및 메모리 비용이 발생합니다. 한 번에 너무 많은 연결이 생성되면 Kafka 클러스터의 가용성에 영향을 미칠 수 있는 부하가 발생합니다. 이러한 가용성 영향으로 인해 애플리케이션이 훨씬 더 많은 연결을 생성해서 캐스케이드 실패를 유발하여 완전 중단이 발생할 수 있습니다. 합리적인 속도로 생성하면 많은 수의 연결을 실현할 수 있습니다.

높은 연결 생성률을 관리하려면 다음과 같은 완화 조치를 사용하는 것이 좋습니다.

- 애플리케이션 배포 메커니즘이 모든 생산자/소비자를 한 번에 다시 시작하지는 않지만 더 작은 배치로 시작하는 것이 좋습니다.
- 애플리케이션 계층에서 개발자는 관리자 클라이언트, 생산자 클라이언트 또는 소비자 클라이언트를 생성하기 전에 무작위 지터(랜덤 절전 모드)가 수행되도록 해야 합니다.
- SIGTERM에서 연결을 종료할 때 모든 Kafka 클라이언트의 연결이 동시에 종료되지 않도록 무작위 절전 모드를 실행해야 합니다. 무작위 절전 모드는 SIGKILL이 발생하기 전까지 제한 시간 내에 있어야 합니다.

  Example 예제 A(Java)

  ```
  sleepInSeconds(randomNumberBetweenOneAndX);
                              this.kafkaProducer = new KafkaProducer<>(this.props);
  ```

Example 예제 B(Java)

```
Runtime.getRuntime().addShutdownHook(new Thread(() -> {
    sleepInSeconds(randomNumberBetweenOneAndTwentyFive);
    kafkaProducer.close(Duration.ofSeconds(5));
});
```

- 애플리케이션 계층에서 개발자는 클라이언트가 싱글톤 패턴으로 애플리케이션당 한 번만 생성되도록 해야 합니다. 예를 들어 lambda를 사용하는 경우 메서드 핸들러 내에서가 아닌 전역 범위에서 클라이언트를 생성해야 합니다.
- 안정적인 상태를 유지하기 위해 연결 수를 모니터링하는 것이 좋습니다. 배포 및 브로커 장애 조치 중 연결 생성/종료/교대는 정상입니다.

Kafka 클라이언트 모니터링

Kafka 클라이언트 모니터링은 Kafka 에코시스템의 상태와 효율성을 유지하는 데 매우 중요합니다. Kafka 관리자, 개발자 또는 운영 팀원이든 관계없이 클라이언트 측 지표를 활성화하면 예정된 이벤트와 예정되지 않은 이벤트 중에 비즈니스 영향을 이해하는 데 매우 중요합니다.

선호하는 지표 캡처 메커니즘을 사용하여 다음 클라이언트 측 지표를 모니터링하는 것이 좋습니다.

AWS에서 지원 티켓을 생성할 때는 인시던트 중에 관찰된 모든 비정상 값을 포함하세요. 또한 오류(경고 아님)를 자세히 설명하는 클라이언트 애플리케이션 로그 샘플도 포함하세요.

생산자 지표

- byte-rate
- record-send-rate
- records-per-request-avg
- acks-latency-avg
- request-latency-avg
- request-latency-max
- record-error-rate
- record-retry-rate
- error-rate

> **Note**
>
> 참고: 리더 장애 조치 또는 네트워크 재전송과 같은 일시적 문제를 처리하기 위한 Kafka 프로토콜의 일부이므로 재시도 시 발생하는 일시적 오류는 우려할 사항이 아닙니다. `record-send-rate`는 프로듀서가 여전히 재시도를 진행하고 있는지 확인합니다.

소비자 지표

- records-consumed-rate
- bytes-consumed-rate
- fetch-rate
- records-lag-max
- record-error-rate
- fetch-error-rate
- poll-rate
- rebalance-latency-avg
- commit-rate

> **Note**
>
> 가져오기 속도와 커밋 속도가 높으면 클러스터에 불필요한 부하가 발생합니다. 더 큰 배치로 요청을 수행하는 것이 가장 적합합니다.

공통 지표

- connection-close-rate
- connection-creation-rate
- connection-count

> **Note**
>
> 연결 생성/종료가 많으면 클러스터에 불필요한 부하가 발생합니다.

MSK Serverless란 무엇인가요?

> **Note**
>
> MSK 서버리스는 미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(오레곤), 캐나다(중부), 아시아 태평양(뭄바이), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 아시아 태평양(서울), 유럽(프랑크푸르트), 유럽(스톡홀름), 유럽(아일랜드), 유럽(파리), 유럽(런던) 리전에서 사용할 수 있습니다.

MSK 서버리스는 클러스터 용량을 관리하고 규모를 조정할 필요 없이 Apache Kafka를 실행할 수 있도록 해주는 Amazon MSK의 클러스터 유형입니다. 주제의 파티션을 관리하면서 용량을 자동으로 프로비저닝하고 확장하므로 클러스터의 크기를 조정하거나 확장할 필요 없이 데이터를 스트리밍할 수 있습니다. MSK 서버리스는 처리량 기반 요금 모델을 제공하므로 사용한 만큼만 비용을 지불하면 됩니다. 애플리케이션에 자동으로 확장되고 축소되는 온디맨드 스트리밍 용량이 필요한 경우 서버리스 클러스터 사용을 고려하세요.

MSK 서버리스는 Apache Kafka와 완벽하게 호환되므로 호환되는 모든 클라이언트 애플리케이션을 사용하여 데이터를 생성하고 소비할 수 있습니다. 또한 다음 서비스와도 통합됩니다.

- AWS PrivateLink프라이빗 연결을 제공하기 위해
- AWS Identity and Access ManagementJava 및 비 Java 언어를 사용한 인증 및 권한 부여를 위한 (IAM) IAM용 클라이언트 구성에 대한 지침은 IAM 액세스 제어를 위한 클라이언트 구성 섹션을 참조하세요.
- AWS Glue스키마 관리를 위한 스키마 레지스트리
- Apache Flink 기반 스트림 처리를 위한 Amazon Managed Service for Apache Flink
- AWS Lambda이벤트 처리를 위한

> **Note**
>
> MSK 서버리스에는 모든 클러스터에 대한 IAM 액세스 제어가 필요합니다. Apache Kafka 액세스 제어 목록(ACL)은 지원되지 않습니다. 자세한 내용은 the section called “IAM 액세스 제어” 단원을 참조하십시오.
> MSK 서버리스에 적용되는 서비스 할당량에 대한 자세한 내용은 the section called “서버리스 클러스터에 대한 할당량” 섹션을 참조하세요.

서버리스 클러스터를 시작하는 데 도움을 주고 서버리스 클러스터의 구성 및 모니터링 옵션에 대해 자세히 알아보려면 다음을 참조하세요.

주제

MSK Serverless 클러스터 사용

이 자습서에서는 MSK 서버리스 클러스터를 생성하고, 클러스터에 액세스할 수 있는 클라이언트 시스템을 생성하고, 클라이언트를 사용하여 클러스터에 주제를 생성하고, 해당 주제에 데이터를 쓰는 방법의 예제를 보여줍니다. 이 예제는 서버리스 클러스터를 생성할 때 선택할 수 있는 모든 옵션을 대표하지는 않습니다. 이 연습의 다른 부분에서는 간단한 설명을 위해 기본 옵션을 선택합니다. 그렇다고 해서 서버리스 클러스터를 설정하는 데 이 옵션이 유일한 옵션이라는 의미는 아닙니다. AWS CLI또는 Amazon MSK API를 사용할 수도 있습니다. 자세한 내용은 Amazon MSK API 참조 2.0을 참조하세요.

주제

MSK Serverless 클러스터 생성

이 단계에서는 두 가지 태스크를 수행합니다. 먼저 기본 설정으로 MSK 서버리스 클러스터를 생성합니다. 둘째, 클러스터에 대한 정보를 수집합니다. 이 정보는 이후 단계에서 클러스터에 데이터를 보낼 수 있는 클라이언트를 만들 때 필요한 정보입니다.

서버리스 클러스터를 만들려면 다음을 수행합니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home Amazon MSK 콘솔을 엽니다.

2. 클러스터 생성을 선택합니다.
3. 생성 방법에서는 빠른 생성 옵션을 선택한 상태로 둡니다. 빠른 생성 옵션을 사용하면 기본 설정으로 서버리스 클러스터를 생성할 수 있습니다.
4. 클러스터 이름에 설명이 포함된 이름을 입력합니다(예: **`msk-serverless-tutorial-cluster`**).
5. 일반 클러스터 속성의 경우 클러스터 유형으로 서버리스를 선택합니다. 나머지 일반 클러스터 속성에는 기본값을 사용합니다.
6. 모든 클러스터 설정 아래의 표를 참고하세요. 이 표에는 네트워킹 및 가용성과 같은 중요한 설정의 기본값이 나열되어 있으며 클러스터 생성 후 각 설정을 변경할 수 있는지 여부가 표시되어 있습니다. 클러스터를 생성하기 전에 설정을 변경하려면 생성 방법에서 사용자 지정 생성 옵션을 선택해야 합니다.

   > **Note**
   >
   > MSK 서버리스 클러스터를 사용하면 최대 5개의 서로 다른 VPC의 클라이언트를 연결할 수 있습니다. 중단 시 클라이언트 애플리케이션이 다른 가용 영역으로 전환할 수 있도록 하려면 각 VPC에 서브넷을 2개 이상 지정해야 합니다.

7. 클러스터 생성을 선택합니다.

클러스터에 대한 정보를 수집하려면 다음을 수행합니다.

1. 클러스터 요약 섹션에서 클라이언트 정보 보기를 선택합니다. 이 버튼은 Amazon MSK가 클러스터 생성을 완료할 때까지 회색으로 표시된 상태로 유지됩니다. 버튼이 활성화되어 사용할 수 있을 때까지 몇 분 정도 기다려야 할 수도 있습니다.
2. 엔드포인트 레이블 아래의 문자열을 복사합니다. 이것은 부트스트랩 서버 문자열입니다.
3. 속성 탭을 선택합니다.
4. 네트워킹 설정 섹션에서 서브넷의 ID와 보안 그룹을 복사하고 저장합니다. 이 정보는 나중에 클라이언트 시스템을 생성하는 데 필요하기 때문입니다.
5. 서브넷 중 하나를 선택합니다. 그러면 Amazon VPC 콘솔이 열립니다. 서브넷과 연결된 Amazon VPC의 ID를 찾습니다. 나중에 사용할 수 있도록 이 Amazon VPC ID를 저장합니다.

다음 단계

MSK Serverless 클러스터의 주제에 대한 IAM 역할 생성

MSK Serverless 클러스터의 주제에 대한 IAM 역할 생성

이 단계에서는 두 가지 태스크를 수행합니다. 첫 번째 태스크는 클러스터에서 주제를 생성하고 해당 주제에 데이터를 전송할 수 있는 액세스 권한을 부여하는 IAM 정책을 생성하는 것입니다. 두 번째 태스크는 IAM 역할을 생성하고 해당 정책을 여기에 연결하는 것입니다. 이후 단계에서 이 역할을 맡는 클라이언트 머신을 생성하고, 이를 사용하여 클러스터에 주제를 생성하고, 해당 주제로 데이터를 전송합니다.

주제를 생성하고 주제에 글을 쓸 수 있는 IAM 정책을 만들려면 다음을 수행합니다.

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택하세요.
4. JSON 탭을 선택한 다음 편집기 창의 JSON을 다음 JSON으로 변경합니다.

   다음 예제에서 다음을 변경합니다.

   - 클러스터를 생성한 AWS 리전의 코드가 포함된 *region*.
   - 계정 ID 예: *123456789012*, AWS 계정ID 포함.
   - *msk-serverless-tutorial-cluster*/*c07c74ea-5146-4a03-add1-9baa787a5b14-s3* 및 *msk-serverless-tutorial-cluster*를 사용자의 서버리스 클러스터 ID 및 토픽 이름으로 변경합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:cluster/msk-serverless-
tutorial-cluster/c07c74ea-5146-4a03-add1-9baa787a5b14-s3"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:CreateTopic",
                "kafka-cluster:WriteData",
                "kafka-cluster:DescribeTopic"
            ],
            "Resource": [
            "arn:aws:kafka:us-east-1:123456789012:topic/msk-serverless-
tutorial-cluster/*"
            ]
        }
    ]
}
```

보안 정책을 작성하는 방법에 대한 지침은 the section called “IAM 액세스 제어” 섹션을 참조하세요.

5. 다음: 태그를 선택합니다.
6. 다음: 검토를 선택합니다.
7. 정책 이름에 설명이 포함된 이름을 입력합니다(예: **msk-serverless-tutorial-policy**).
8. 정책 생성을 선택합니다.

IAM 역할을 생성하여 여기에 정책을 연결하려면 다음을 수행합니다.

1. 탐색 창에서 역할을 선택합니다.
2. 역할 생성을 선택합니다.
3. 일반 사용 사례에서 EC2를 선택한 다음 다음: 권한을 선택합니다.
4. 검색 상자에 이 자습서를 위해 이전에 생성한 정책의 이름을 입력합니다. 그런 다음 정책의 왼쪽에 있는 확인란을 선택합니다.
5. 다음: 태그를 선택합니다.
6. 다음: 검토를 선택합니다.
7. 역할 이름에 설명이 포함된 이름을 입력합니다(예: **msk-serverless-tutorial-role**).
8. 역할 생성을 선택합니다.

다음 단계

[MSK Serverless 클러스터에 액세스할 클라이언트 머신 생성]

MSK Serverless 클러스터에 액세스할 클라이언트 머신 생성

이 단계에서는 두 가지 태스크를 수행합니다. 첫 번째 태스크는 Apache Kafka 클라이언트 머신으로 사용할 Amazon EC2 인스턴스를 생성하는 것입니다. 두 번째 작업은 머신에 Java 및 Apache Kafka 도구를 설치하는 것입니다.

클라이언트 머신을 만들려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스 시작을 선택합니다.
3. 클라이언트 머신에 대한 설명이 포함된 이름(예: **`msk-serverless-tutorial-client`**)을 입력합니다.
4. Amazon Linux 2 AMI(HVM) - 커널 5.10, SSD 볼륨 유형이 Amazon Machine Image(AMI) 유형으로 선택된 상태로 둡니다.
5. t2.micro 인스턴스 유형을 선택된 상태로 둡니다.
6. 키 페어(로그인)에서 키 페어 생성을 선택합니다. 키 페어 이름에 **`MSKServerlessKeyPair`**를 입력합니다. 키 페어 다운로드를 선택합니다. 또는 기존 키 페어를 사용할 수 있습니다.
7. 네트워크 설정에서 편집을 선택합니다.
8. VPC에서 서버리스 클러스터의 Virtual Private Cloud(VPC) ID를 입력합니다. 클러스터를 생성한 후 ID를 저장한 Amazon VPC 서비스를 기반으로 하는 VPC입니다.
9. 서브넷에서 클러스터를 생성한 후 ID를 저장한 서브넷을 선택합니다.
10. 방화벽(보안 그룹)의 경우 클러스터와 연결된 보안 그룹을 선택합니다. 이 값은 해당 보안 그룹에 보안 그룹에서 자신으로 트래픽을 허용하는 인바운드 규칙이 있는 경우에 작동합니다. 이러한 규칙을 사용하면 동일한 보안 그룹의 구성원이 서로 통신할 수 있습니다. 자세한 내용은 Amazon VPC 개발자 안내서의 [보안 그룹 규칙]을 참조하세요.
11. 고급 세부 정보 섹션을 확장하고 [MSK Serverless 클러스터의 주제에 대한 IAM 역할 생성]에서 생성한 IAM 역할을 선택합니다.
12. 시작을 선택합니다.
13. 왼쪽 탐색 창에서 인스턴스를 선택합니다. 그런 다음 새로 생성한 Amazon EC2 인스턴스를 나타내는 행에서 확인란을 선택합니다. 이 시점부터 이 인스턴스를 클라이언트 머신이라고 부릅니다.
14. 연결을 선택하고 지침에 따라 클라이언트 머신에 연결합니다.

클라이언트 머신에 Apache Kafka 클라이언트 도구를 설정하려면 다음을 수행합니다.

1. Java를 설치하려면 클라이언트 머신에서 다음 명령을 실행합니다.

```
sudo yum -y install java-11
```

2. 주제를 생성하고 데이터를 전송하는 데 필요한 Apache Kafka 도구를 가져오려면 다음 명령을 실행합니다.

```
wget https://archive.apache.org/dist/kafka/2.8.1/kafka_2.12-2.8.1.tgz
```

```
tar -xzf kafka_2.12-2.8.1.tgz
```

> **Note**
>
> Kafka 아카이브를 추출한 후 bin 디렉터리의 스크립트에 적합한 실행 권한이 있는지 확인합니다. 다음 명령으로 실행하세요.
>
> ```
> chmod +x kafka_2.12-2.8.1/bin/*.sh
> ```

3. kafka_2.12-2.8.1/libs 디렉터리로 이동하고 다음 명령을 실행하여 Amazon MSK IAM JAR 파일을 다운로드합니다. Amazon MSK IAM JAR을 사용하면 클라이언트 머신이 클러스터에 액세스할 수 있습니다.

```
wget https://github.com/aws/aws-msk-iam-auth/releases/download/v2.3.0/aws-msk-iam-auth-2.3.0-all.jar
```

이 명령을 사용하여 Amazon MSK IAM JAR 파일의 다른 버전 또는 최신 버전을 다운로드할 수도 있습니다.

4. kafka_2.12-2.8.1/bin 디렉터리로 이동합니다. 다음 속성 설정을 복사하여 새 파일에 붙여넣습니다. 파일 이름을 client.properties로 지정하고 저장합니다.

```
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
```

다음 단계

[Apache Kafka 주제 생성](#)

Apache Kafka 주제 생성

이 단계에서는 이전에 생성한 클라이언트 머신을 사용하여 서버리스 클러스터에 주제를 생성합니다.

주제

- 토픽 생성을 위한 환경 설정
- 토픽 생성 및 해당 위치에 데이터 쓰기

토픽 생성을 위한 환경 설정

- 주제를 생성하기 전에 AWSMSK IAM JAR 파일을 Kafka 설치의 `libs/` 디렉터리에 다운로드했는지 확인합니다. 아직 수행하지 않은 경우 Kafka의 `libs/` 디렉터리에서 다음 명령을 실행합니다.

  ```
  wget https://github.com/aws/aws-msk-iam-auth/releases/download/v2.3.0/aws-msk-iam-auth-2.3.0-all.jar
  ```

  이 JAR 파일은 MSK 서버리스 클러스터를 사용한 IAM 인증에 필요합니다.

- Kafka 명령을 실행할 때에 AWSMSK IAM JAR 파일이 `classpath` 포함되어 있는지 확인해야 할 수 있습니다. 이렇게 하려면 다음 중 한 가지를 수행합니다.
  - 다음 예제와 같이 Kafka 라이브러리가 포함되도록 `CLASSPATH` 환경 변수를 설정합니다.

    ```
    export CLASSPATH=<path-to-your-kafka-installation>/libs/*:<path-to-your-kafka-installation>/libs/aws-msk-iam-auth-2.3.0-all.jar
    ```

  - 다음 예제와 같이 명시적 `classpath`와 함께 전체 Java 명령을 사용하여 Kafka 명령을 실행합니다.

    ```
    java -cp "<path-to-your-kafka-installation>/libs/*:<path-to-your-kafka-installation>/libs/aws-msk-iam-auth-2.3.0-all.jar"
     org.apache.kafka.tools.TopicCommand --bootstrap-server $BS --command-config
     client.properties --create --topic msk-serverless-tutorial --partitions 6
    ```

토픽 생성 및 해당 위치에 데이터 쓰기

1. 다음 export 명령에서 *my-endpoint*를 클러스터를 생성한 후 저장한 부트스트랩 서버 문자열로 변경합니다. 그런 다음 클라이언트 머신의 kafka_2.12-2.8.1/bin 디렉터리로 이동하여 export 명령을 실행합니다.

```
export BS=my-endpoint
```

2. 다음 명령을 실행하여 msk-serverless-tutorial이라는 주제를 생성합니다.

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --bootstrap-server $BS
 --command-config client.properties --create --topic msk-serverless-tutorial --
partitions 6
```

다음 단계

[MSK Serverless에서 데이터 생성 및 소비](#)

MSK Serverless에서 데이터 생성 및 소비

이 단계에서는 이전 단계에서 생성한 주제를 사용하여 데이터를 생성하고 소비합니다.

메시지를 생산하고 소비하려면

1. 다음 명령을 실행하여 콘솔 생산자를 생성합니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-list $BS
 --producer.config client.properties --topic msk-serverless-tutorial
```

2. 원하는 메시지를 입력하고 Enter키를 누릅니다. 이 단계를 두 번 또는 세 번 반복하십시오. 한 줄을 입력하고 Enter를 누를 때마다 해당 줄이 별도의 메시지로 클러스터에 전송됩니다.
3. 클라이언트 머신에 대한 연결을 열어 둔 다음, 새 창에서 해당 머신에 대한 별도의 두 번째 연결을 엽니다.
4. 클라이언트 머신에 대한 두 번째 연결을 사용하여 다음 명령으로 콘솔 소비자를 생성합니다. *my-endpoint*를 클러스터를 생성한 후 저장한 부트스트랩 서버 문자열로 변경합니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-consumer.sh --bootstrap-
server my-endpoint --consumer.config client.properties --topic msk-serverless-
tutorial --from-beginning
```

콘솔 생산자 명령을 사용할 때 앞서 입력한 메시지가 표시되기 시작합니다.

5. 생산자 창에 메시지를 더 입력하고 소비자 창에 메시지가 표시되는지 확인합니다.

해당 명령을 실행하는 동안 `classpath` 문제가 발생하는 경우 올바른 디렉터리에서 실행 중인지 확인합니다. 또한 AWSMSK IAM JAR이 `libs` 디렉터리에 있는지 확인합니다. 또는 다음 예제와 같이 명시적 `classpath`와 함께 전체 Java 명령을 사용하여 Kafka 명령을 실행할 수 있습니다.

```
java -cp "kafka_2.12-2.8.1/libs/*:kafka_2.12-2.8.1/libs/aws-msk-iam-auth-2.3.0-
all.jar" org.apache.kafka.tools.ConsoleProducer –broker-list $BS –producer.config
 client.properties –topic msk-serverless-tutorial
```

다음 단계

[MSK Serverless를 위해 생성한 리소스 삭제](#)

MSK Serverless를 위해 생성한 리소스 삭제

이 단계에서는 자습서에서 생성한 리소스를 삭제합니다.

클러스터를 삭제하려면 다음을 수행합니다.

1. https://console.aws.amazon.com/msk/home에서 Amazon MSK 콘솔을 엽니다.
2. 클러스터 목록에서 이 자습서를 위해 생성한 클러스터를 선택합니다.
3. 작업에서 클러스터 삭제를 선택합니다.
4. 상자에 `delete`를 입력한 다음 삭제를 선택합니다.

클라이언트 머신을 중지하려면 다음을 수행합니다.

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. Amazon EC2 인스턴스 목록에서 이 자습서를 위해 생성한 클라이언트 머신을 선택합니다.
3. 인스턴스 상태를 선택한 다음 인스턴스 종료를 선택합니다.
4. 종료를 선택합니다.

IAM 정책 및 역할을 삭제하려면 다음을 수행합니다.

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.

2. 탐색 창에서 역할을 선택합니다.
3. 검색 상자에 이 자습서를 위해 생성한 IAM 역할의 이름을 입력합니다.
4. 역할을 선택합니다. 그런 다음 역할 삭제를 선택하고 삭제를 확인합니다.
5. 탐색 창에서 정책을 선택합니다.
6. 검색 상자에 이 자습서를 위해 생성한 정책의 이름을 입력합니다.
7. 정책을 선택하면 해당 요약 페이지가 열립니다. 정책의 요약 페이지에서 정책 삭제를 선택합니다.
8. 삭제를 선택합니다.

MSK Serverless 클러스터에 대한 구성 속성

Amazon MSK는 서버리스 클러스터에 대한 브로커 구성 속성을 설정합니다. 이러한 브로커 구성 속성 설정은 변경할 수 없지만 그러나 다음 주제 수준의 구성 속성을 설정하거나 수정할 수 있습니다. 다른 모든 주제 수준의 구성 속성은 구성할 수 없습니다.

구성 속성	기본값	편집 가능	최대 허용 값
cleanup.policy	Delete	예, 그러나 주제를 생성할 때만 가능합니다.	
compression.type	생산자	예	
max.message.bytes	1048588	예	8388608(8MiB)
message.timestamp.difference.max.ms	long.max	예	
message.timestamp.type	CreateTime	예	
retention.bytes	250GiB	예	무제한, 무제한 보존을 위해 -1로 설정
retention.ms	7일	예	무제한, 무제한 보존을 위해 -1로 설정

이러한 주제 수준의 구성 속성을 설정하거나 수정하기 위해 Apache Kafka 명령줄 도구를 사용할 수 있습니다. 자세한 내용과 설정 방법은 공식 Apache Kafka 설명서의 3.2 주제 수준 구성을 참조하세요.

> **Note**
>
> MSK Serverless의 토픽에 대한 segment.bytes 구성은 수정할 수 없습니다. 그러나 Kafka Streams 애플리케이션은 MSK Serverless에서 허용하는 것과 다른 segment.bytes 구성 값으로 내부 토픽 생성을 시도할 수 있습니다. MSK Serverless로 Kafka Streams를 구성하는 방법에 대한 자세한 내용은 MSK Express 브로커 및 MSK Serverless를 통해 Kafka Streams 사용 섹션을 참조하세요.

Amazon MSK Serverless에서 Apache Kafka 명령줄 도구를 사용하는 경우 Amazon MSK Serverless 시작하기 설명서의 클라이언트 머신에 Apache Kafka 클라이언트 도구를 설정하려면 섹션에서 1~4단계를 완료했는지 확인합니다. 또한 명령에 `--command-config client.properties` 파라미터를 포함해야 합니다.

예를 들어 다음 명령을 사용하여 retention.bytes 주제의 구성 속성을 수정하여 무제한 보존을 설정할 수 있습니다.

```
<path-to-your-kafka-client-installation>/bin/kafka-configs.sh —bootstrap-
server <bootstrap_server_string> —command-config client.properties --entity-type topics
 --entity-name <topic_name> --alter --add-config retention.bytes=-1
```

이 예제에서는 *<bootstrap server string>*을 Amazon MSK Serverless 클러스터의 부트스트랩 서버 엔드포인트로 바꾸고, *<topic_name>*을 수정하려는 주제 이름으로 바꿉니다.

`--command-config client.properties` 파라미터는 Kafka 명령줄 도구에서 적절한 구성 설정을 사용하여 Amazon MSK Serverless 클러스터와 통신하도록 합니다.

MSK Serverless 클러스터 모니터링

Amazon MSK는 Amazon CloudWatch와 통합되므로 MSK 서버리스 클러스터에 대한 지표를 수집, 확인, 분석할 수 있습니다. 다음 테이블에 나와 있는 지표는 모든 서버리스 클러스터에 사용할 수 있습니다. 이러한 지표는 주제의 각 파티션에 대한 개별 데이터 포인트로 게시되므로 주제 수준에서 보려면 '합계' 통계로 보는 것을 권장합니다.

Amazon MSK는 1분에 한 번씩 CloudWatch에 `PerSec` 지표를 게시합니다. 즉, 1분 동안의 '합계' 통계는 `PerSec` 지표에 대한 초당 데이터를 정확하게 나타냅니다. 1분 이상의 기간 동안 초당 데이터를 수집하려면 CloudWatch 수학 표현식(`m1 * 60/PERIOD(m1)`)을 사용하세요.

기본 모니터링 수준에서 사용할 수 있는 지표

이름	표시되는 경우	측정 기준	설명
`BytesInPerSec`	생산자가 주제에 쓴 후	클러스터 이름, 주제	클라이언트로부터 받은 초당 바이트 수입니다. 이 지표는 각 주제에 대해 사용할 수 있습니다.
`BytesOutPerSec`	소비자 그룹이 주제에서 소비한 후	클러스터 이름, 주제	클라이언트에 전송된 초당 바이트 수입니다. 이 지표는 각 주제에 대해 사용할 수 있습니다.
`FetchMessageConversionsPerSec`	소비자 그룹이 주제에서 소비한 후	클러스터 이름, 주제	주제에 대한 초당 가져오기 메시지 전환 횟수
`EstimatedMaxTimeLag`	소비자 그룹이 주제에서 소비한 후	클러스터 이름, 소비자 그룹, 주제	MaxOffsetLag 지표의 시간 추정치.
`MaxOffsetLag`	소비자 그룹이 주제에서 소비한 후	클러스터 이름, 소비자 그룹, 주제	주제의 모든 파티션에 대한 최대 오프셋 지연.
`MessagesInPerSec`	생산자가 주제에 쓴 후	클러스터 이름, 주제	주제에 대해 초당 수신되는 메시지 수
`ProduceMessageConversionsPerSec`	생산자가 주제에 쓴 후	클러스터 이름, 주제	주제에 대한 초당 생성 메시지 전환 횟수
`SumOffsetLag`	소비자 그룹이 주제에서 소비한 후	클러스터 이름, 소비자 그룹, 주제	주제의 모든 파티션에 대한 집계된 오프셋 지연

MSK 서비리스 지표를 보려면 다음을 수행합니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 탐색 창의 지표에서 모든 지표를 선택합니다.
3. 지표에서 용어(**kafka**)를 검색합니다.
4. 다른 지표를 보려면 AWS/Kafka/클러스터 이름, 주제 또는 AWS/Kafka/클러스터 이름, 소비자 그룹, 주제를 선택하세요.

MSK Connect 이해

MSK Connect는 개발자가 Apache Kafka 클러스터에서 데이터를 간편하게 스트리밍할 수 있도록 해주는 Amazon MSK의 기능입니다. MSK Connect는 데이터베이스, 검색 인덱스, 파일 시스템과 같은 외부 시스템과 Apache Kafka 클러스터를 연결하기 위한 오픈 소스 프레임워크인 Kafka Connect 버전 2.7.1 또는 3.7.x를 사용합니다. MSK Connect를 사용하면 Amazon S3 및 Amazon OpenSearch Service와 같은 널리 사용되는 데이터 저장소로 데이터를 이동하거나 데이터 저장소에서 데이터를 가져오는 Kafka Connect를 위해 구축된 완전 관리형 커넥터를 배포할 수 있습니다. 데이터베이스의 변경 로그를 Apache Kafka 클러스터로 스트리밍하기 위해 Debezium과 같은 타사 개발 커넥터를 배포하거나 코드 변경 없이 기존 커넥터를 배포할 수 있습니다. 커넥터는 부하 변화에 따라 자동으로 규모를 조정하며 사용한 리소스에 대해서만 비용을 지불합니다.

소스 커넥터를 사용하여 외부 시스템에서 주제로 데이터를 가져올 수 있습니다. 싱크 커넥터를 사용하면 주제의 데이터를 외부 시스템으로 내보낼 수 있습니다.

MSK Connect는 MSK 클러스터든 독립적으로 호스팅되는 Apache Kafka 클러스터든 상관없이 Amazon VPC에 연결할 수 있는 모든 Apache Kafka 클러스터에 대한 커넥터를 지원합니다.

MSK Connect는 커넥터 상태 및 전송 상태를 지속적으로 모니터링하고, 기본 하드웨어를 패치 및 관리하며, 처리량 변화에 맞추어 커넥터의 규모를 자동으로 조정합니다.

MSK Connect 사용을 시작하려면 the section called “시작하기” 섹션을 참조하세요.

MSK Connect를 사용하여 생성할 수 있는 AWS리소스에 대한 자세한 내용은 the section called “커넥터 이해”, the section called “사용자 지정 플러그인 생성”및 섹션을 참조하세요the section called “MSK Connect 작업자 이해”.

MSK Connect API에 대한 자세한 내용은 Amazon MSK Connect API 참조를 참조하세요.

Amazon MSK Connect를 사용할 경우의 이점

Apache Kafka는 실시간 데이터 스트림을 수집 및 처리하기 위해 가장 널리 사용되는 오픈 소스 스트리밍 플랫폼 중 하나입니다. Apache Kafka를 사용하면 데이터 생성 및 데이터 소비 애플리케이션을 분리하여 독립적으로 규모 조정할 수 있습니다.

Kafka Connect는 Apache Kafka를 사용하여 스트리밍 애플리케이션을 빌드하고 실행하는 데 중요한 구성 요소입니다. Kafka Connect는 Kafka와 외부 시스템 간에 데이터를 이동하는 표준화된 방법을 제공합니다. Kafka Connect는 확장성이 뛰어나며 대용량 데이터를 처리할 수 있습니다. Kafka Connect는 Kafka 주제와 외부 시스템 간에 데이터를 이동하는 커넥터를 구성, 배포 및 모니터링하기 위한 강력

한 API 작업 및 도구 세트를 제공합니다. 이러한 도구를 사용하여 Kafka Connect의 기능을 사용자 지정하고 확장하여 스트리밍 애플리케이션의 특정 요구 사항을 충족할 수 있습니다.

Apache Kafka Connect 클러스터를 단독으로 운영하거나 오픈 소스 Apache Kafka Connect 애플리케이션을 AWS로 마이그레이션하려고 할 때 문제가 발생할 수 있습니다. 이러한 문제로는 인프라를 설정하고 애플리케이션을 배포하는 데 필요한 시간, 자체 관리형 Apache Kafka Connect 클러스터를 설정할 때 발생하는 엔지니어링 장애 요소, 관리 운영 오버헤드가 있습니다.

이러한 문제를 해결하려면 Amazon Managed Streaming for Apache Kafka Connect(Amazon MSK Connect)를 사용하여 오픈 소스 Apache Kafka Connect 애플리케이션을 AWS로 마이그레이션하는 것이 좋습니다. Amazon MSK Connect는 Kafka Connect를 사용하여 Apache Kafka 클러스터와 외부 시스템(예: 데이터베이스, 검색 인덱스 및 파일 시스템) 간에 데이터를 스트리밍하는 작업을 간소화합니다.

다음은 Amazon MSK Connect로 마이그레이션할 경우 얻을 수 있는 몇 가지 이점입니다.

- 운영 오버헤드 제거 - Amazon MSK Connect는 Apache Kafka Connect 클러스터의 패치 적용, 프로비저닝 및 확장과 관련된 운영 부담을 덜어줍니다. Amazon MSK Connect는 Connect 클러스터의 상태를 지속적으로 모니터링하고 워크로드 중단을 일으키지 않으면서 패치 적용 및 버전 업그레이드를 자동화합니다.
- Connect 작업의 자동 재시작 - Amazon MSK Connect는 실패한 작업을 자동으로 복구하여 프로덕션 중단을 줄일 수 있습니다. 작업 실패는 Kafka에 대한 TCP 연결 제한 위반, 싱크 커넥터에 대한 소비자 그룹에 새 작업자가 조인할 때 작업 재조정 등의 일시적인 오류로 인해 발생할 수 있습니다.
- 자동 수평적 및 수직적 스케일링 - Amazon MSK Connect를 사용하면 커넥터 애플리케이션이 더 높은 처리량을 지원하도록 자동으로 규모를 조정할 수 있습니다. Amazon MSK Connect에서 자동으로 규모 조정을 관리합니다. Auto Scaling 그룹의 작업자 수와 사용률 임곗값만 지정하면 됩니다. Amazon MSK Connect `UpdateConnector` API 작업을 사용하여 가변 처리량을 지원하기 위해 vCPUs를 1~8개의 vCPUs로 수직적으로 스케일 업하거나 스케일 다운할 수 있습니다.
- 프라이빗 네트워크 연결 - Amazon MSK Connect는AWSPrivateLink 및 프라이빗 DNS 이름을 사용하여 소스 및 싱크 시스템에 비공개로 연결합니다.

MSK Connect 시작하기

이 자습서는 AWS Management 콘솔를 사용하여 MSK 클러스터를 생성하고 클러스터에서 S3 버킷으로 데이터를 전송하는 싱크 커넥터를 생성하는 step-by-step 자습서입니다.

주제

MSK Connect에 필요한 리소스 설정

이 단계에서는 시작하기 시나리오에 필요한 다음 리소스를 생성합니다.

- 커넥터에서 데이터를 수신하는 대상 역할을 하는 Amazon S3 버킷.
- 데이터를 전송할 MSK 클러스터로, 이후 커넥터가 해당 클러스터에서 데이터를 읽고 대상 S3 버킷으로 전송함
- 대상 S3 버킷에 작성할 수 있는 권한이 포함된 IAM 정책.
- 커넥터가 대상 S3 버킷에 쓸 수 있도록 허용하는 IAM 역할 생성하는 IAM 정책을 이 역할에 추가합니다.
- 클러스터와 커넥터가 있는 Amazon VPC에서 Amazon S3로 데이터를 전송할 수 있도록 하는 Amazon VPC 엔드포인트

S3 버킷을 만들려면 다음을 수행합니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 버킷 생성을 선택합니다.
3. 버킷의 이름에 설명이 포함된 이름을 입력합니다(예: `amzn-s3-demo-bucket-mkc-tutorial`).
4. 아래로 스크롤하여 버킷 생성을 선택합니다.
5. 버킷 목록에서 새로 생성한 버킷을 선택합니다.
6. 폴더 생성을 선택합니다.
7. 폴더 이름에 `tutorial`을 입력한 다음 아래로 스크롤하여 폴더 생성을 선택합니다.

클러스터를 생성하려면 다음을 수행합니다.

1. https://console.aws.amazon.com/msk/home?region=us-east-1#/home/에서 Amazon MSK 콘솔을 엽니다.
2. 왼쪽 창의 MSK 클러스터에서 클러스터를 선택합니다.
3. 클러스터 생성을 선택합니다.
4. 생성 방법에서 사용자 지정 생성을 선택합니다.
5. 클러스터 이름에 **`mkc-tutorial-cluster`**를 입력합니다.
6. 클러스터 유형에서 프로비저닝을 선택합니다.
7. 다음을 선택합니다.
8. 네트워킹에서 Amazon VPC를 선택합니다. 그런 다음 사용하려는 가용 영역과 서브넷을 선택합니다. 자습서의 뒷부분에서 필요하므로 선택한 Amazon VPC와 서브넷의 ID를 기억해 둡니다.
9. 다음을 선택합니다.
10. 액세스 제어 방법에서는 인증되지 않은 액세스만 선택되도록 합니다.
11. 암호화에서 일반 텍스트만 선택했는지 확인합니다.
12. 마법사를 계속 진행한 다음 클러스터 생성을 선택합니다. 그러면 해당 클러스터에 대한 세부 정보 페이지로 이동합니다. 해당 페이지의 적용된 보안 그룹에서 보안 그룹 ID를 찾습니다. 자습서의 뒷부분에서 필요하므로 해당 ID를 기억해 둡니다.

S3 버킷에 작성할 수 있는 권한이 있는 IAM 정책 생성

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 생성을 선택합니다.
4. 정책 편집기에서 JSON을 선택한 후 편집기 창의 JSON을 다음 JSON으로 변경합니다.

   다음 예제에서는 *`<amzn-s3-demo-bucket-my-tutorial>`*을 S3 버킷의 이름으로 바꿉니다.

   JSON

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

```
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::<amzn-s3-demo-bucket-my-tutorial>"
    },
    {
      "Sid": "AllowObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:AbortMultipartUpload",
        "s3:ListMultipartUploadParts",
        "s3:ListBucketMultipartUploads"
      ],
      "Resource": "arn:aws:s3:::<amzn-s3-demo-bucket-my-tutorial>/*"
    }
  ]
}
```

보안 정책을 작성하는 방법에 대한 지침은 the section called “IAM 액세스 제어” 섹션을 참조하세요.

5. 다음을 선택합니다.
6. 검토 및 생성 페이지에서 다음을 수행합니다.

   a. 정책 이름에 설명이 포함된 이름을 입력합니다(예: **mkc-tutorial-policy**).

   b. 이 정책에 정의된 권한에서 정책에 정의된 권한을 검토 및/또는 편집합니다.

   c. (선택 사항) 정책을 식별, 구성 또는 검색하는 데 도움이 되도록 새 태그 추가를 선택하여 태그를 키-값 페어로 추가합니다. 예를 들어 키-값 페어가 **Environment** 및 **Test**인 태그를 정책에 추가합니다.

      태그 사용에 대한 자세한 내용은 IAM 사용 설명서의 AWS Identity and Access Management 리소스 태그를 참조하세요.

7. 정책 생성을 선택합니다.

대상 버킷에 쓸 수 있는 IAM 역할을 생성하려면 다음을 수행합니다.

1. IAM 콘솔의 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
2. 신뢰할 수 있는 엔터티 선택 페이지에서 다음을 수행합니다.
   a. 신뢰할 수 있는 엔터티 유형에서 AWS 서비스를 선택합니다.
   b. 서비스 또는 사용 사례에서 S3를 선택합니다.
   c. 사용 사례에서 S3를 선택합니다.
3. 다음을 선택합니다.
4. 권한 추가 페이지에서 다음을 수행합니다.
   a. 권한 정책 검색 상자에 이 자습서를 위해 이전에 생성한 정책의 이름을 입력합니다. 예: mkc-tutorial-policy. 그런 다음 정책 이름 왼쪽에서 확인란을 선택합니다.
   b. (선택 사항) 권한 경계를 선택합니다. 이는 서비스 역할에서 사용 가능한 고급 기능이며 서비스에 연결된 역할은 아닙니다. 권한 경계 설정에 대한 자세한 내용은 IAM 사용 설명서의 역할 생성 및 정책 연결(콘솔)을 참조하세요.
5. 다음을 선택합니다.
6. 이름, 검토 및 생성 페이지에서 다음을 수행합니다.
   a. 역할 이름에 설명이 포함된 이름을 입력합니다(예: **mkc-tutorial-role**).

      > **Important**
      >
      > 역할 이름을 지정할 때는 다음 사항에 유의하세요.
      >
      > - 역할 이름은 내에서 고유해야 하며 대/소문자를 구분할 AWS 계정수 없습니다.
      >
      >   예를 들어, 이름이 **PRODROLE**과 **prodrole**, 두 가지로 지정된 역할을 만들지 마세요. 역할 이름이 정책 또는 ARN의 일부로 사용되는 경우 역할 이름은 대소문자를 구분합니다. 그러나 로그인 프로세스와 같이 콘솔에서 역할 이름이 고객에게 표시되는 경우에는 역할 이름이 대소문자를 구분하지 않습니다.
      > - 다른 엔터티가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.

   b. (선택 사항) 설명에 역할에 대한 설명을 입력합니다.
   c. (선택 사항) 역할에 대한 사용 사례와 권한을 편집하려면 1단계: 신뢰할 수 있는 엔터티 선택 또는 2단계: 권한 추가 섹션에서 편집을 선택합니다.

d. (선택 사항) 역할을 식별, 구성 또는 검색하는 데 도움이 되도록 새 태그 추가를 선택하여 태그를 키-값 페어로 추가합니다. 예를 들어 **ProductManager** 및 **John**의 키-값 페어를 사용하여 역할에 태그를 추가합니다.

   태그 사용에 대한 자세한 내용은 IAM 사용 설명서의 [AWS Identity and Access Management 리소스 태그를 참조하세요](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_tags.html).

7. 역할을 검토한 다음 역할 생성을 선택합니다.

MSK Connect가 역할을 수행하도록 허용하려면 다음을 수행합니다.

1. IAM 콘솔의 왼쪽 창에 있는 액세스 관리에서 역할을 선택합니다.
2. `mkc-tutorial-role`을 찾아 선택합니다.
3. 역할의 요약에서 신뢰 관계 탭을 선택합니다.
4. 신뢰 관계 편집을 선택합니다.
5. 기존 신뢰 정책을 다음 JSON으로 변경합니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. 신뢰 정책 업데이트를 선택합니다.

클러스터의 VPC에서 Amazon S3로 Amazon VPC 엔드포인트를 생성하려면 다음을 수행합니다.

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 왼쪽 창에서 엔드포인트를 선택합니다.

3. 엔드포인트 생성을 선택합니다.
4. 서비스 이름에서 com.amazonaws.us-east-1.s3 서비스 및 게이트웨이 유형을 선택합니다.
5. 클러스터의 VPC를 선택한 다음 클러스터의 서브넷과 연결된 라우팅 테이블 왼쪽에 있는 상자를 선택합니다.
6. 엔드포인트 생성을 선택합니다.

다음 단계

사용자 지정 플러그인 생성

사용자 지정 플러그인 생성

플러그인에는 커넥터의 로직을 정의하는 코드가 포함되어 있습니다. 이 단계에서는 렌즈 Amazon S3 싱크 커넥터용 코드가 포함된 사용자 지정 플러그인을 생성합니다. 이후 단계에서 MSK 커넥터를 생성할 때 해당 코드가 이 사용자 지정 플러그인에 있도록 지정합니다. 동일한 플러그인을 사용하여 서로 다른 구성으로 여러 개의 MSK 커넥터를 생성할 수 있습니다.

사용자 지정 플러그인을 생성하려면 다음을 수행합니다.

1. S3 커넥터를 다운로드합니다.
2. 액세스 권한이 있는 S3 버킷에 ZIP 파일을 업로드합니다. Amazon S3에 파일을 업로드하는 방법에 대한 자세한 내용은 Amazon S3 사용 설명서의 객체 업로드를 참조하세요.
3. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
4. 왼쪽 창에서 MSK Connect를 확장한 다음 사용자 지정 플러그인을 선택합니다.
5. 사용자 지정 플러그인 생성을 선택합니다.
6. S3 찾아보기(Browse S3)를 선택합니다.
7. 버킷 목록에서 ZIP 파일을 업로드한 버킷을 찾아 해당 버킷을 선택합니다.
8. 버킷의 개체 목록에서 ZIP 파일 왼쪽에 있는 라디오 버튼을 선택한 다음 선택이라는 레이블이 지정된 버튼을 선택합니다.
9. 사용자 지정 플러그인 이름에 `mkc-tutorial-plugin`을 입력한 다음 사용자 지정 플러그인 생성을 선택합니다.

사용자 지정 플러그인 생성을 완료하는 데 AWS몇 분 정도 걸릴 수 있습니다. 생성 프로세스가 완료되면 브라우저 창 상단의 배너에 다음 메시지가 표시됩니다.

```
Custom plugin mkc-tutorial-plugin was successfully created
The custom plugin was created. You can now create a connector using this custom plugin.
```

다음 단계

클라이언트 머신 및 Apache Kafka 주제 생성

클라이언트 머신 및 Apache Kafka 주제 생성

이 단계에서는 Apache Kafka 클라이언트 인스턴스로 사용할 Amazon EC2 인스턴스를 생성합니다. 그런 다음 해당 인스턴스를 사용하여 클러스터에 주제를 생성합니다.

클라이언트 머신을 만들려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스 시작을 선택합니다.
3. 클라이언트 머신의 이름을 입력합니다(예: **mkc-tutorial-client**).
4. Amazon Linux 2 AMI(HVM) - 커널 5.10, SSD 볼륨 유형이 Amazon Machine Image(AMI) 유형으로 선택된 상태로 둡니다.
5. t2.xlarge 인스턴스 유형을 선택합니다.
6. 키 페어(로그인)에서 키 페어 생성을 선택합니다. 키 페어 이름에 **mkc-tutorial-key-pair**를 입력한 다음 키 페어 다운로드를 선택합니다. 또는 기존 키 페어를 사용할 수 있습니다.
7. 인스턴스 시작을 선택합니다.
8. 인스턴스 보기를 선택합니다. 그런 다음 보안 그룹 열에서 새 인스턴스와 연결된 보안 그룹을 선택합니다. 보안 그룹의 ID를 복사하여 나중에 사용할 수 있도록 저장합니다.

새로 생성된 클라이언트가 클러스터에 데이터를 전송할 수 있도록 허용하려면 다음을 수행합니다.

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 왼쪽 창의 보안에서 보안 그룹을 선택합니다. 보안 그룹 ID 열에서 클러스터의 보안 그룹을 찾습니다. the section called “MSK Connect에 필요한 리소스 설정”에 클러스터를 생성할 때 이 보안 그룹의 ID를 저장했습니다. 행 왼쪽에 있는 상자를 선택하여 해당 보안 그룹을 선택합니다. 다른 보안 그룹이 동시에 선택되지 않았는지 확인합니다.
3. 화면 하단에서 인바운드 규칙 탭을 선택합니다.
4. 인바운드 규칙 편집을 선택합니다.
5. 화면 왼쪽 하단에서 규칙 추가를 선택합니다.

6. 새 규칙의 유형 열에서 모든 트래픽을 선택합니다. 소스 열의 오른쪽 필드에 클라이언트 컴퓨터의 보안 그룹 ID를 입력합니다. 클라이언트 머신을 생성한 후 저장한 보안 그룹 ID입니다.
7. 규칙 저장을 선택합니다. 이제 MSK 클러스터가 이전 절차에서 생성한 클라이언트의 모든 트래픽을 허용합니다.

주제를 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스 표에서 `mkc-tutorial-client`를 선택합니다.
3. 화면 상단에서 연결을 선택한 다음 지침에 따라 인스턴스에 연결합니다.
4. 다음 명령을 실행하여 클라이언트 인스턴스에 Java를 설치합니다.

```
sudo yum install java-1.8.0
```

5. Apache Kafka를 다운로드하려면 다음 명령을 실행합니다.

```
wget https://archive.apache.org/dist/kafka/2.2.1/kafka_2.12-2.2.1.tgz
```

> **Note**
> 이 명령에 사용된 사이트 이외의 미러 사이트를 사용하려면 Apache 웹사이트에서 다른 것을 선택할 수 있습니다.

6. 이전 단계에 TAR 파일을 다운로드한 디렉토리에서 다음 명령을 실행합니다.

```
tar -xzf kafka_2.12-2.2.1.tgz
```

7. kafka_2.12-2.2.1 디렉터리로 이동하십시오.
8. https://console.aws.amazon.com/msk/home?region=us-east-1#/home/에서 Amazon MSK 콘솔을 엽니다.
9. 왼쪽 창에서 클러스터를 선택한 다음 이름 `mkc-tutorial-cluster`를 선택합니다.
10. 클라이언트 정보 보기를 선택합니다.
11. 일반 텍스트 연결 문자열을 복사합니다.
12. 완료를 선택합니다.
13. 클라이언트 인스턴스(`mkc-tutorial-client`)에서 다음 명령을 실행하여 *`bootstrapServerString`*을 클러스터의 클라이언트 정보를 볼 때 저장한 값으로 변경합니다.

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --create --bootstrap-server bootstrapServerString --replication-factor 2 --partitions 1 --topic mkc-tutorial-topic
```

명령이 성공하면 `Created topic mkc-tutorial-topic.` 메시지가 표시됩니다.

다음 단계

[커넥터 생성](#)

커넥터 생성

이 절차에서는 AWS Management 콘솔을 사용하여 커넥터를 생성하는 방법을 설명합니다.

커넥터를 생성하려면 다음을 수행합니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 왼쪽 창에서 MSK Connect를 확장하고 커넥터를 선택합니다.
3. 커넥터 생성을 선택합니다.
4. 플러그인 목록에서 `mkc-tutorial-plugin`을 선택하고 다음을 선택합니다.
5. 커넥터 이름에 `mkc-tutorial-connector`를 입력합니다.
6. 클러스터 목록에서 `mkc-tutorial-cluster`를 선택합니다.
7. 커넥터 네트워크 설정 섹션에서 네트워크 유형에 대해 다음 중 하나를 선택합니다.
   - IPv4(기본값) - IPv4를 통한 대상 연결 전용
   - 듀얼 스택 - IPv4 및 IPv6를 통해 대상에 연결(서브넷에 IPv4 및 IPv6 CIDR 블록이 연결된 경우에만 사용 가능)
8. 다음 구성을 복사하여 커넥터 구성 필드에 붙여넣습니다.

   리전을 커넥터를 생성하는 AWS 리전의 코드로 바꿔야 합니다. 또한 다음 예제에서 Amazon S3 버킷 이름 *<amzn-s3-demo-bucket-my-tutorial>*을 사용자의 버킷 이름으로 바꿉니다.

```
connector.class=io.confluent.connect.s3.S3SinkConnector
s3.region=us-east-1
format.class=io.confluent.connect.s3.format.json.JsonFormat
```

```
flush.size=1
schema.compatibility=NONE
tasks.max=2
topics=mkc-tutorial-topic
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner
storage.class=io.confluent.connect.s3.storage.S3Storage
s3.bucket.name=<amzn-s3-demo-bucket-my-tutorial>
topics.dir=tutorial
```

9. 액세스 권한에서 `mkc-tutorial-role`을 선택합니다.
10. 다음을 선택합니다. 보안 페이지에서 다음을 다시 선택합니다.
11. 로그 페이지에서 다음을 선택합니다.
12. 검토 및 생성 페이지에서 커넥터 구성을 검토하고 커넥터 생성을 선택합니다.

다음 단계

[MSK 클러스터로 데이터 전송]

MSK 클러스터로 데이터 전송

이 단계에서는 이전에 생성한 Apache Kafka 주제로 데이터를 전송한 다음 대상 S3 버킷에서 동일한 데이터를 찾습니다.

MSK 클러스터로 데이터를 전송하려면 다음을 수행합니다.

1. 클라이언트 인스턴스에 설치된 Apache Kafka의 `bin` 폴더에 다음 내용으로 `client.properties`라는 이름의 텍스트 파일을 생성합니다.

```
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
```

2. 다음 명령을 실행하여 콘솔 생산자를 생성합니다. *`BootstrapBrokerString`*을 이전 명령을 실행할 때 가져온 값으로 변경합니다.

```
<path-to-your-kafka-installation>/bin/kafka-console-producer.sh --broker-
list BootstrapBrokerString --producer.config client.properties --topic mkc-
tutorial-topic
```

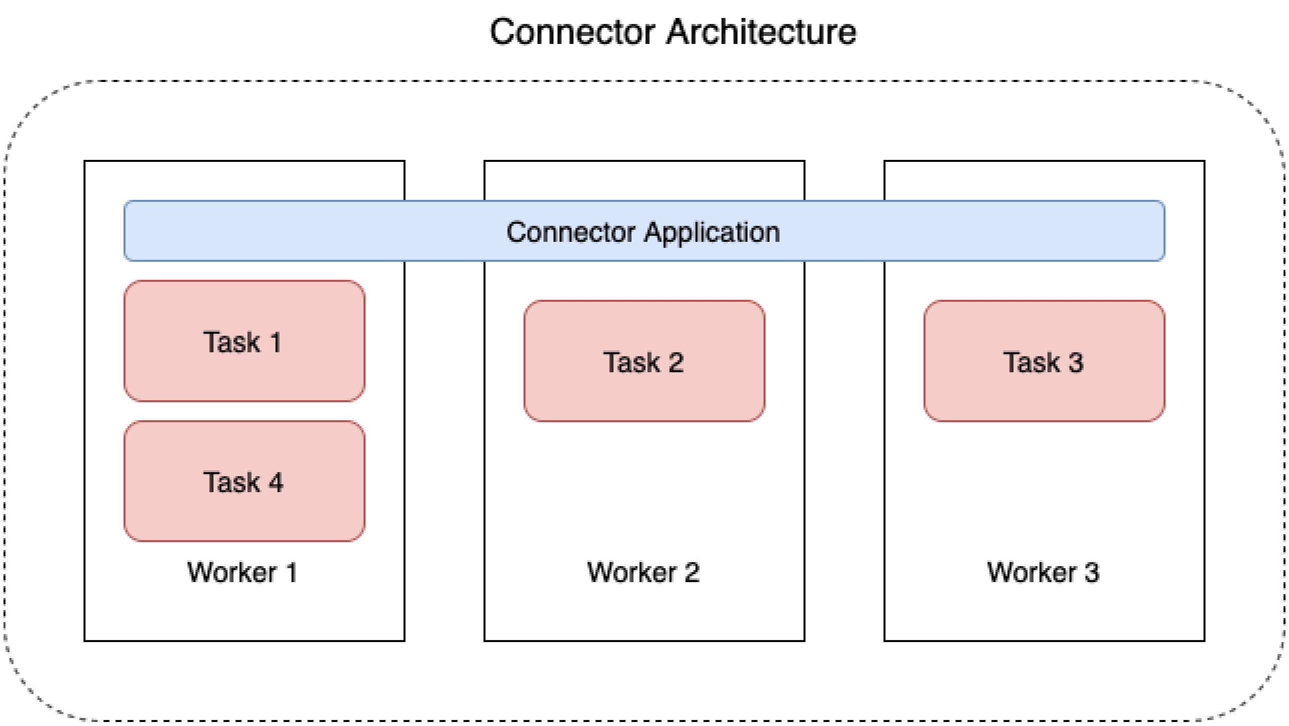
3. 원하는 메시지를 입력하고 Enter키를 누릅니다. 이 단계를 두 번 또는 세 번 반복하십시오. 한 줄을 입력하고 Enter를 누를 때마다, 그 줄은 Apache Kafka 클러스터에 별도의 메시지로 전송됩니다.

4. 대상 Amazon S3 버킷에서 이전 단계에서 보낸 메시지를 찾습니다.

커넥터 이해

커넥터는 데이터 소스의 스트리밍 데이터를 Apache Kafka 클러스터로 지속적으로 복사하거나 클러스터의 데이터를 데이터 싱크로 지속적으로 복사하여 외부 시스템과 Amazon 서비스를 Apache Kafka와 통합합니다. 커넥터는 데이터를 대상에 전달하기 전에 변환, 형식 변환 또는 데이터 필터링과 같은 간단한 로직을 수행할 수도 있습니다. 소스 커넥터는 데이터 소스에서 데이터를 가져와서 해당 데이터를 클러스터로 푸시하고, 싱크 커넥터는 클러스터에서 데이터를 가져와서 해당 데이터를 데이터 싱크로 푸시합니다.

다음 다이어그램은 커넥터의 아키텍처를 보여줍니다. 작업자는 커넥터 로직을 실행하는 Java 가상 머신(JVM) 프로세스입니다. 각 작업자는 병렬 스레드에서 실행되는 일련의 작업을 생성하고 데이터 복사 작업을 수행합니다. 작업은 상태를 저장하지 않으므로 탄력적이고 규모를 조정할 수 있는 데이터 파이프라인을 제공하기 위해 언제든지 시작, 중지 또는 다시 시작할 수 있습니다.



커넥터 용량 이해

커넥터의 총 용량은 커넥터가 보유한 작업자 수와 작업자당 MSK 연결 단위(MCU)의 수에 따라 달라집니다. 각 MCU는 1vCPU의 컴퓨팅과 4기가바이트의 메모리를 나타냅니다. MCU 메모리는 사용 중인 힙 메모리가 아닌 작업자 인스턴스의 전체 메모리와 관련이 있습니다.

MSK Connect 작업자는 고객 제공 서브넷의 IP 주소를 사용합니다. 각 작업자는 고객이 제공한 서브넷 중 하나에서 하나의 IP 주소를 사용합니다. 특히 작업자 수가 변할 수 있는 커넥터를 자동 조정할 때 지정된 용량을 고려하기 위해 CreateConnector 요청에 제공된 서브넷에 사용 가능한 IP 주소가 충분한지 확인해야 합니다.

커넥터를 생성하려면 다음 두 가지 용량 모드 중 하나를 선택해야 합니다.

- 프로비저닝 - 커넥터의 용량 요구 사항을 알고 있는 경우 이 모드를 선택합니다. 두 가지 값을 지정합니다.
  - 작업자 수
  - 작업자당 MCU 수
- 자동 크기 조정 - 커넥터의 용량 요구 사항이 가변적이거나 이를 미리 알 수 없는 경우 이 모드를 선택합니다. 자동 규모 조정 모드를 사용하는 경우 Amazon MSK Connect는 커넥터에서 실행 중인 작업자 수와 작업자당 MCU 수에 비례하는 값으로 커넥터의 `tasks.max` 속성을 재정의합니다.

  세 가지의 값 세트를 지정합니다.
  - 최소 및 최대 작업자 수
  - `CpuUtilization` 지표에 의해 결정되는 CPU 사용률에 대한 스케일 인 및 스케일 아웃 비율로, 커넥터에 대한 `CpuUtilization` 지표가 스케일 아웃 비율을 초과하면 MSK Connect는 커넥터에서 실행 중인 작업자 수를 늘립니다. `CpuUtilization` 지표가 스케일 인 비율 아래로 떨어지면 MSK Connect는 작업자 수를 줄입니다. 작업자 수는 항상 커넥터를 만들 때 지정한 최소 및 최대 수 이내로 유지됩니다.
  - 작업자당 MCU 수

작업자에 대한 자세한 내용은 [the section called “MSK Connect 작업자 이해”](#) 섹션을 참조하세요. MSK Connect 지표에 대한 자세한 내용은 [the section called “MSK Connect 모니터링”](#)를 참조하세요.

듀얼 스택 네트워크 유형 구성

Amazon MSK Connect는 새 커넥터에 대해 듀얼 스택 네트워크 유형을 지원합니다. 듀얼 스택 네트워킹을 사용하면 커넥터가 IPv4 및 IPv6를 통해 대상에 연결할 수 있습니다. IPv6 연결은 듀얼 스택 모드(IPv4 + IPv6)에서만 사용할 수 있습니다. IPv6-only 네트워킹은 지원되지 않습니다.

기본적으로 새 커넥터는 IPv4 네트워크 유형을 사용합니다. 듀얼 스택 네트워크 유형의 커넥터를 생성하려면 다음 섹션에 설명된 사전 조건을 충족했는지 확인합니다. 듀얼 스택 네트워크 유형을 사용하여 커넥터를 생성한 후에는 해당 네트워크 유형을 수정할 수 없습니다. 네트워크 유형을 변경하려면 커넥터를 삭제하고 다시 생성해야 합니다.

Amazon MSK Connect는 IPv6 및 IPv4를 통한 서비스 API 엔드포인트 연결도 지원합니다. API 호출에 IPv6 연결을 사용하려면 듀얼 스택 엔드포인트를 사용해야 합니다. MSK Connect 서비스 엔드포인트에 대한 자세한 내용은 Amazon MSK Connect 엔드포인트 및 할당량을 참조하세요.

듀얼 스택 네트워크 유형을 사용하기 위한 사전 조건

커넥터에 대해 듀얼 스택 네트워크 유형을 구성하기 전에 커넥터 생성 중에 제공하는 모든 서브넷에 IPv6 및 IPv4 CIDR 블록이 할당되어 있는지 확인합니다.

듀얼 스택 네트워크 유형 사용에 대한 고려 사항

- IPv6 지원은 현재 IPv6 전용이 아닌 듀얼 스택 모드(IPv4 + IPv6-only)에서만 사용할 수 있습니다.
- 듀얼 스택이 활성화된 커넥터는 IPv4와 IPv6를 통해 MSK와 싱크 또는 소스 데이터 시스템 모두에 연결할 수 있습니다.
- 커넥터 생성 후에는 네트워크 유형을 수정할 수 없습니다. 네트워크 유형을 변경하려면 커넥터를 삭제하고 다시 생성해야 합니다.
- 커넥터 생성 중에 지정된 모든 서브넷은 커넥터 생성이 듀얼 스택 네트워크 유형으로 성공하려면 듀얼 스택을 지원해야 합니다.
- 듀얼 스택 서브넷을 사용하지만 네트워크 유형을 지정하지 않은 경우 커넥터는 이전 버전과의 호환성을 위해 기본적으로 IPv4-only으로 설정됩니다.
- 기존 커넥터의 경우 네트워크 유형을 업데이트할 수 없습니다. 네트워크 유형을 변경하려면 커넥터를 삭제하고 다시 생성해야 합니다.
- 듀얼 스택 네트워킹을 사용하면 추가 비용이 발생하지 않습니다.

커넥터 생성

이 절차에서는 AWS Management 콘솔을 사용하여 커넥터를 생성하는 방법을 설명합니다.

를 사용하여 커넥터 생성AWS Management 콘솔

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 왼쪽 창의 MSK Connect에서 커넥터를 선택합니다.
3. 커넥터 생성을 선택합니다.
4. 기존 사용자 지정 플러그인을 사용하여 커넥터를 생성하거나 새 사용자 지정 플러그인을 먼저 생성하는 것 중에서 선택할 수 있습니다. 사용자 정의 플러그인 및 플러그인 생성 방법에 대한 자세한 내용은 the section called “사용자 지정 플러그인 생성” 섹션을 참조하세요. 이 절차에서는 사용

하려는 사용자 지정 플러그인이 있다고 가정해 보겠습니다. 사용자 지정 플러그인 목록에서 사용하려는 플러그인을 찾아 왼쪽에 있는 상자를 선택한 후 다음을 선택합니다.

5. 이름과 설명(선택 사항)을 입력합니다.
6. 연결하려는 클러스터를 선택합니다.
7. 커넥터 네트워크 설정 섹션에서 네트워크 유형에 대해 다음 중 하나를 선택합니다.
   - IPv4(기본값) - IPv4를 통한 대상 연결 전용
   - 듀얼 스택 - IPv4 및 IPv6를 통해 대상에 연결(서브넷에 연결된 IPv4 및 IPv6 CIDR 블록이 있는 경우에만 사용 가능)
8. 커넥터 구성을 지정합니다. 지정해야 하는 구성 파라미터는 생성하려는 커넥터 유형에 따라 달라집니다. 그러나 일부 파라미터(예: `connector.class` 및 `tasks.max` 파라미터)는 모든 커넥터에 공통으로 적용됩니다. 다음은 Confluent Amazon S3 Sink Connector 구성의 예제입니다.

```
connector.class=io.confluent.connect.s3.S3SinkConnector
tasks.max=2
topics=my-example-topic
s3.region=us-east-1
s3.bucket.name=amzn-s3-demo-bucketflush.size=1
storage.class=io.confluent.connect.s3.storage.S3Storage
format.class=io.confluent.connect.s3.format.json.JsonFormat
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter
schema.compatibility=NONE
```

9. 그런 다음 커넥터 용량을 구성합니다. 프로비저닝과 자동 규모 조정이라는 두 가지 용량 모드 중에서 선택할 수 있습니다. 이러한 두 가지 옵션에 대한 자세한 내용은 the section called “커넥터 용량 이해” 단원을 참조하십시오.
10. 기본 작업자 구성 또는 사용자 지정 작업자 구성 중 하나를 선택합니다. 사용자 지정 작업자 구성 생성에 대한 자세한 내용은 the section called “MSK Connect 작업자 이해” 섹션을 참조하세요.
11. 그런 다음 서비스 실행 역할을 지정합니다. 이는 MSK Connect가 수임할 수 있고 커넥터에 필요한 AWS리소스에 액세스하는 데 필요한 모든 권한을 부여하는 IAM 역할이어야 합니다. 이러한 권한은 커넥터의 로직에 따라 달라집니다. 이 역할 생성 방법에 대한 자세한 내용은 the section called “서비스 실행 역할 이해” 단원을 참조하십시오.
12. 다음을 선택하고 보안 정보를 검토한 후 다음을 다시 선택합니다.
13. 원하는 로깅 옵션을 지정한 후 다음을 선택합니다. 로깅에 대한 추가 정보는 the section called “로깅” 섹션을 참조하세요.

14. 검토 및 생성 페이지에서 커넥터 구성을 검토하고 커넥터 생성을 선택합니다.

MSK Connect API를 사용하여 커넥터를 생성하려면 CreateConnector를 참조하세요.

`UpdateConnector` API를 사용하여 커넥터의 구성을 수정할 수 있습니다. 자세한 내용은 the section called “커넥터 업데이트” 단원을 참조하십시오.

커넥터 업데이트

이 절차에서는 AWS Management 콘솔을 사용하여 기존 MSK Connect 커넥터의 구성을 업데이트하는 방법을 설명합니다.

를 사용하여 커넥터 구성 업데이트AWS Management 콘솔

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 왼쪽 창의 MSK Connect에서 커넥터를 선택합니다.
3. 기존 커넥터를 선택합니다.
4. 커넥터 구성 편집을 선택합니다.
5. 커넥터 구성을 업데이트합니다. UpdateConnector를 사용하여 `connector.class`를 재정의할 수 없습니다. 다음은 Confluent Amazon S3 Sink Connector 구성 예제의 예제입니다.

```
connector.class=io.confluent.connect.s3.S3SinkConnector
tasks.max=2
topics=my-example-topic
s3.region=us-east-1
s3.bucket.name=amzn-s3-demo-bucketflush.size=1
storage.class=io.confluent.connect.s3.storage.S3Storage
format.class=io.confluent.connect.s3.format.json.JsonFormat
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter
schema.compatibility=NONE
```

6. 제출을 선택합니다.
7. 그런 다음 커넥터의 작업 탭에서 작업의 현재 상태를 모니터링할 수 있습니다.

MSK 연결 API를 사용하여 커넥터 구성을 업데이트하려면 UpdateConnector를 참조하세요.

커넥터에서 연결

다음 모범 사례를 통해 Amazon MSK Connect에 대한 연결 성능을 개선할 수 있습니다.

Amazon VPC 피어링 또는 Transit Gateway의 IP가 겹치지 않도록 하세요.

Amazon MSK Connect와 함께 Amazon VPC 피어링 또는 Transit Gateway를 사용하는 경우 CIDR 범위의 IP를 사용하여 피어링된 VPC 리소스에 도달하도록 커넥터를 구성하지 마세요.

- “10.99.0.0/16”
- “192.168.0.0/16”
- “172.21.0.0/16”

사용자 지정 플러그인 생성

플러그인은 커넥터 로직을 정의하는 코드를 포함하는 AWS 리소스입니다. S3 버킷에 JAR 파일(또는 하나 이상의 JAR 파일이 포함된 ZIP 파일)을 업로드하고 플러그인을 생성할 때 버킷의 위치를 지정합니다. 커넥터를 생성할 때 MSK Connect에서 사용할 플러그인을 지정합니다. 플러그인과 커넥터의 관계는 일대다 관계입니다. 동일한 플러그인에서 하나 이상의 커넥터를 생성할 수 있습니다.

커넥터를 위한 코드를 개발하는 방법에 대한 자세한 내용은 Apache Kafka 설명서의 커넥터 개발 안내서를 참조하세요.

AWS Management 콘솔을 사용하여 사용자 지정 플러그인 생성

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 왼쪽 창의 MSK Connect에서 사용자 지정 플러그인을 선택합니다.
3. 사용자 지정 플러그인 생성을 선택합니다.
4. S3 찾아보기(Browse S3)를 선택합니다.
5. S3 버킷 목록에서 플러그인을 위한 JAR 또는 ZIP 파일이 있는 버킷을 선택합니다.
6. 개체 목록에서 플러그인의 JAR 또는 ZIP 파일 왼쪽에 있는 상자를 선택한 다음 선택을 선택합니다.
7. 사용자 지정 플러그인 생성을 선택합니다.

MSK Connect API를 사용하여 사용자 지정 플러그인을 생성하려면 CreateCustomPlugin을 참조하세요.

MSK Connect 작업자 이해

작업자는 커넥터 로직을 실행하는 Java 가상 머신(JVM) 프로세스입니다. 각 작업자는 병렬 스레드에서 실행되는 일련의 작업을 생성하고 데이터 복사 작업을 수행합니다. 작업은 상태를 저장하지 않으므로 탄력적이고 규모 조정 가능한 데이터 파이프라인을 제공하기 위해 언제든지 시작, 중지 또는 다시 시작할 수 있습니다. 확장 이벤트 또는 예기치 못한 장애로 인한 작업자 수 변경은 남은 작업자가 자동으로 감지합니다. 이들은 남은 작업자 세트에서 작업의 균형을 재조정하기 위해 조정합니다. Connect 작업자는 Apache Kafka의 소비자 그룹을 사용하여 조정하고 재조정합니다.

커넥터의 용량 요구 사항이 가변적이거나 예측하기 어려운 경우 MSK Connect에서 필요에 따라 지정한 하한과 상한 사이에서 작업자 수 규모를 조정할 수 있습니다. 또는 커넥터 로직을 실행할 정확한 작업자 수를 지정할 수 있습니다. 자세한 내용은 the section called “커넥터 용량 이해” 섹션을 참조하세요.

IP 주소를 사용하는 MSK Connect 작업자

MSK Connect 작업자는 고객 제공 서브넷의 IP 주소를 사용합니다. 각 작업자는 고객이 제공한 서브넷 중 하나에서 하나의 IP 주소를 사용합니다. 특히 작업자 수가 변할 수 있는 커넥터를 자동 조정할 때 지정된 용량을 고려하기 위해 CreateConnector 요청에 제공된 서브넷에 사용 가능한 IP 주소가 충분한지 확인해야 합니다.

기본 작업자 구성

MSK Connect는 다음과 같은 기본 작업자 구성을 제공합니다.

```
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter
```

지원되는 작업자 구성 속성

MSK Connect는 기본 작업자 구성을 제공합니다. 커넥터와 함께 사용할 사용자 지정 작업자 구성을 생성하는 옵션도 있습니다. 다음 목록에는 Amazon MSK Connect에서 지원하거나 지원하지 않는 작업자 구성 속성에 대한 정보가 포함되어 있습니다.

- `key.converter` 및 `value.converter` 속성은 필수입니다.
- MSK Connect에서는 다음과 같은 `producer.` 구성 속성을 지원합니다.

  ```
  producer.acks
  producer.batch.size
  ```

```
producer.buffer.memory
producer.compression.type
producer.enable.idempotence
producer.key.serializer
producer.linger.ms
producer.max.request.size
producer.metadata.max.age.ms
producer.metadata.max.idle.ms
producer.partitioner.class
producer.reconnect.backoff.max.ms
producer.reconnect.backoff.ms
producer.request.timeout.ms
producer.retry.backoff.ms
producer.value.serializer
```

- MSK Connect에서는 다음과 같은 `consumer.` 구성 속성을 지원합니다.

```
consumer.allow.auto.create.topics
consumer.auto.offset.reset
consumer.check.crcs
consumer.fetch.max.bytes
consumer.fetch.max.wait.ms
consumer.fetch.min.bytes
consumer.heartbeat.interval.ms
consumer.key.deserializer
consumer.max.partition.fetch.bytes
consumer.max.poll.interval.ms
consumer.max.poll.records
consumer.metadata.max.age.ms
consumer.partition.assignment.strategy
consumer.reconnect.backoff.max.ms
consumer.reconnect.backoff.ms
consumer.request.timeout.ms
consumer.retry.backoff.ms
consumer.session.timeout.ms
consumer.value.deserializer
```

- 다음 속성을 제외하고 `producer.` 또는 `consumer.` 접두사로 시작하지 않는 다른 모든 구성 속성이 지원됩니다.

```
access.control.
admin.
admin.listeners.https.
```

```
client.
connect.
inter.worker.
internal.
listeners.https.
metrics.
metrics.context.
rest.
sasl.
security.
socket.
ssl.
topic.tracking.
worker.
bootstrap.servers
config.storage.topic
connections.max.idle.ms
connector.client.config.override.policy
group.id
listeners
metric.reporters
plugin.path
receive.buffer.bytes
response.http.headers.config
scheduled.rebalance.max.delay.ms
send.buffer.bytes
status.storage.topic
```

작업자 구성 속성 및 해당 속성이 나타내는 내용에 대한 자세한 내용은 Apache Kafka 설명서의 Kafka Connect Configs를 참조하세요.

사용자 지정 작업자 구성 생성

이 절차에서는 AWS Management 콘솔을 사용하여 사용자 지정 작업자 구성을 생성하는 방법을 설명합니다.

AWS Management 콘솔을 사용하여 사용자 지정 작업자 구성 생성

1. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.
2. 왼쪽 창의 MSK Connect에서 작업자 구성을 선택합니다.
3. 작업자 구성 생성을 선택합니다.

4. 이름과 설명(선택 사항)을 입력한 다음 설정할 속성 및 값을 추가합니다.
5. 작업자 구성 생성을 선택합니다.

MSK Connect API를 사용하여 작업자 구성을 생성하려면 CreateWorkerConfiguration을 참조하세요.

`offset.storage.topic`을 사용하여 소스 커넥터 오프셋 관리

이 섹션에서는 오프셋 스토리지 주제를 사용하여 소스 커넥터 오프셋을 관리하는 데 도움이 되는 정보를 제공합니다. 오프셋 스토리지 주제는 Kafka Connect에서 커넥터 및 작업 구성 오프셋을 저장하는 데 사용하는 내부 주제입니다.

고려 사항

소스 커넥터 오프셋을 관리할 때는 다음을 고려하세요.

- 오프셋 스토리지 주제를 지정하려면 커넥터 오프셋이 저장되는 Kafka 주제의 이름을 작업자 구성에서 `offset.storage.topic`의 값으로 제공합니다.
- 커넥터 구성을 변경할 때는 주의하세요. 구성 값을 변경하면 소스 커넥터가 구성 값을 키 오프셋 레코드에 사용하는 경우 의도하지 않은 커넥터 동작이 발생할 수 있습니다. 플러그인 설명서를 참조하여 지침을 확인하는 것을 권장합니다.
- 기본 파티션 수 사용자 지정 - `offset.storage.topic`을 추가하여 작업자 구성을 사용자 지정하는 것 외에도 오프셋 및 상태 스토리지 주제에 대한 파티션 수를 사용자 지정할 수 있습니다. 내부 주제의 기본 파티션은 다음과 같습니다.
  - `config.storage.topic`: 1, 구성 불가능, 단일 파티션 주제여야 함
  - `offset.storage.topic`: 25, `offset.storage.partitions`를 제공하여 구성 가능
  - `status.storage.topic`: 5, `status.storage.partitions`를 제공하여 구성 가능
- 주제 수동 삭제 – Amazon MSK Connect는 커넥터를 배포할 때마다 새로운 Kafka Connect 내부 주제(주제 이름은 `__amazon_msk_connect`로 시작)를 생성합니다. 삭제된 커넥터에 연결된 이전 주제는 자동으로 제거되지 않는데 내부 주제(예: `offset.storage.topic`)와 같은 주제는 커넥터 간에 다시 사용될 수 있기 때문입니다. 그러나 MSK Connect에서 생성한 사용하지 않는 내부 주제는 수동으로 삭제할 수 있습니다. 내부 주제의 이름은 `__amazon_msk_connect_<offsets|status|configs>_connector_name_connector_id` 형식에 따라 지정됩니다.

  `__amazon_msk_connect_<offsets|status|configs>_connector_name_connector_id` 정규 표현식을 사용하여 내부 주제를 삭제할 수 있습니다. 실행 중인 커넥터에서 현재 사용 중인 내부 주제를 삭제해서는 안 됩니다.

- MSK Connect에서 생성한 내부 주제에 동일한 이름 사용 - 이전에 만든 커넥터에서 오프셋을 사용하기 위해 오프셋 스토리지 주제를 다시 사용하려면 새 커넥터에 이전 커넥터와 동일한 이름을 지정해야 합니다. 작업자 구성을 사용하여 `offset.storage.topic` 속성을 설정하고 `offset.storage.topic`에 동일한 이름을 할당하여 다른 커넥터 간에 다시 사용할 수 있습니다. 이 구성은 [커넥터 오프셋 관리]에 설명되어 있습니다. MSK Connect는 서로 다른 커넥터가 `config.storage.topic` 및 `status.storage.topic`을 공유하는 것을 허용하지 않습니다. 이러한 주제는 MSKC에서 새 커넥터를 생성할 때마다 생성됩니다. 이러한 커넥터는 `__amazon_msk_connect_<status|configs>_connector_name_connector_id` 형식을 따라 자동으로 이름이 지정되므로 생성하는 커넥터마다 다릅니다.

기본 오프셋 스토리지 주제 사용

기본적으로 Amazon MSK Connect는 사용자가 생성하는 각 커넥터에 대해 Kafka 클러스터에 새 오프셋 스토리지 주제를 생성합니다. MSK는 커넥터 ARN의 일부를 사용하여 기본 주제 이름을 구성합니다. 예를 들어 `__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-4abc-8be8-c657f7e4ff32-2`입니다.

사용자 지정 오프셋 스토리지 주제 사용

소스 커넥터 간에 오프셋 연속성을 제공하려면 기본 주제 대신 원하는 오프셋 스토리지 주제를 사용할 수 있습니다. 오프셋 스토리지 주제를 지정하면 이전 커넥터의 마지막 오프셋에서 읽기를 다시 시작하는 소스 커넥터를 생성하는 것과 같은 작업을 수행하는 데 도움이 됩니다.

오프셋 스토리지 주제를 지정하면 커넥터를 생성하기 전에 작업자 구성에서 `offset.storage.topic` 속성 값을 제공해야 합니다. 오프셋 스토리지 주제를 다시 사용하여 이전에 만든 커넥터의 오프셋을 사용하려면 새 커넥터에 이전 커넥터와 동일한 이름을 지정해야 합니다. 사용자 지정 오프셋 스토리지 주제를 만드는 경우 주제 구성에서 [`cleanup.policy`]를 `compact`로 설정해야 합니다.

> **Note**
>
> 싱크 커넥터를 생성할 때 오프셋 스토리지 주제를 지정하는 경우 해당 주제가 아직 존재하지 않으면 MSK Connect에서 해당 주제를 생성합니다. 그러나 이 주제는 커넥터 오프셋을 저장하는 데 사용되지 않습니다.
> 싱크 커넥터 오프셋은 대신 Kafka 소비자 그룹 프로토콜을 사용하여 관리됩니다. 각 싱크 커넥터는 `connect-{CONNECTOR_NAME}`이라는 그룹을 생성합니다. 소비자 그룹이 존재하는 한

동일한 CONNECTOR_NAME 값으로 생성되는 모든 후속 싱크 커넥터는 마지막으로 커밋된 오프셋부터 계속됩니다.

Example : 업데이트된 구성으로 소스 커넥터를 다시 생성하기 위한 오프셋 스토리지 주제 지정

변경 데이터 캡처(CDC) 커넥터가 있고 CDC 스트림에서 위치를 잃지 않고 커넥터 구성을 수정하고 싶다고 가정해 보겠습니다. 기존 커넥터 구성을 업데이트할 수는 없지만 커넥터를 삭제하고 동일한 이름의 새 커넥터를 생성할 수는 있습니다. 새 커넥터가 CDC 스트림에서 읽기를 시작할 위치를 알려주려면 작업자 구성에서 이전 커넥터의 오프셋 스토리지 주제를 지정하면 됩니다. 다음 단계에서는 이 작업을 수행하는 방법을 설명합니다.

1. 클라이언트 머신에서 다음 명령을 실행하여 커넥터의 오프셋 스토리지 주제 이름을 찾습니다. *<bootstrapBrokerString>*을 클러스터의 부트스트랩 브로커 문자열로 변경합니다. 부트스트랩 브로커 문자열을 가져오는 방법에 대한 지침은 [Amazon MSK 클러스터를 위한 부트스트랩 브로커 가져오기]() 섹션을 참조하세요.

```
<path-to-your-kafka-installation>/bin/kafka-topics.sh --list --bootstrap-
server <bootstrapBrokerString>
```

   다음 출력에는 기본 내부 커넥터 주제를 포함한 모든 클러스터 주제 목록이 나와 있습니다. 이 예제에서는 기존 CDC 커넥터가 MSK Connect에서 생성한 [기본 오프셋 스토리지 주제]()를 사용합니다. 이것이 오프셋 저장소 주제를 `__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-4abc-8be8-c657f7e4ff32-2`라고 부르는 이유입니다.

```
__consumer_offsets
__amazon_msk_canary
__amazon_msk_connect_configs_my-mskc-connector_12345678-09e7-4abc-8be8-
c657f7e4ff32-2
__amazon_msk_connect_offsets_my-mskc-connector_12345678-09e7-4abc-8be8-
c657f7e4ff32-2
__amazon_msk_connect_status_my-mskc-connector_12345678-09e7-4abc-8be8-
c657f7e4ff32-2
my-msk-topic-1
my-msk-topic-2
```

2. https://console.aws.amazon.com/msk/에서 Amazon MSK 콘솔을 엽니다.

3. 커넥터 목록에서 커넥터를 선택합니다. 커넥터 구성 필드의 내용을 복사하여 저장하면 이를 수정하고 새 커넥터를 생성하는 데 사용할 수 있습니다.

4. 커넥터를 삭제하려면 삭제를 선택합니다. 그런 다음 텍스트 입력 필드에 커넥터 이름을 입력하여 삭제를 확인합니다.
5. 시나리오에 적합한 값으로 사용자 지정 작업자 구성을 생성합니다. 지침은 사용자 지정 작업자 구성 생성 섹션을 참조하세요.

   작업자 구성에서 다음 구성과 같이 이전에 검색한 오프셋 스토리지 주제의 이름을 `offset.storage.topic`의 값으로 지정해야 합니다.

   ```
   config.providers.secretManager.param.aws.region=eu-west-3
   key.converter=<org.apache.kafka.connect.storage.StringConverter>
   value.converter=<org.apache.kafka.connect.storage.StringConverter>
   config.providers.secretManager.class=com.github.jcustenborder.kafka.config.aws.SecretsManag
   config.providers=secretManager
   offset.storage.topic=__amazon_msk_connect_offsets_my-mskc-
   connector_12345678-09e7-4abc-8be8-c657f7e4ff32-2
   ```
6.
   Important

   새 커넥터에 이전 커넥터와 동일한 이름을 지정해야 합니다.

   이전 단계에서 설정한 작업자 구성을 사용하여 새 커넥터를 생성합니다. 지침은 커넥터 생성 단원을 참조하세요.

튜토리얼: 구성 공급자를 사용하여 민감한 정보 외부화

이 예제는 오픈 소스 구성 공급자를 사용하여 Amazon MSK Connect의 민감한 정보를 외부화하는 방법을 보여줍니다. 구성 공급자를 사용하면 커넥터 또는 작업자 구성에서 일반 텍스트 대신 변수를 지정할 수 있으며 커넥터에서 실행 중인 작업자는 런타임 시 이러한 변수를 확인합니다. 이렇게 하면 보안 인증 정보 및 기타 비밀이 일반 텍스트로 저장되는 것을 방지할 수 있습니다. 예제의 구성 공급자는 AWSSecrets Manager, Amazon S3 및 Systems Manager(SSM)에서 구성 파라미터 검색을 지원합니다. 2단계에서는 구성하려는 서비스에 대한 민감한 정보의 저장 및 검색을 설정하는 방법을 확인할 수 있습니다.

고려 사항

Amazon MSK Connect에서 MSK 구성 공급자를 사용하는 경우 다음을 고려하세요.

- 구성 공급자를 IAM 서비스 실행 역할에 사용하는 경우 적절한 권한을 할당합니다.

- 작업자 구성에서 구성 공급자를 정의하고 커넥터 구성에서 해당 구현을 정의합니다.
- 플러그인이 해당 값을 암호로 정의하지 않은 경우 커넥터 로그에 민감한 구성 값이 표시될 수 있습니다. Kafka Connect는 정의되지 않은 구성 값을 기타 일반 텍스트 값과 동일하게 취급합니다. 자세한 내용은 커넥터 로그에 비밀이 표시되지 않도록 하기를 참조하세요.
- 기본적으로 MSK Connect는 커넥터가 구성 공급자를 사용하는 경우 커넥터를 자주 다시 시작합니다. 이러한 다시 시작 동작을 해제하려면 커넥터 구성에서 `config.action.reload` 값을 none으로 설정하면 됩니다.

사용자 지정 플러그인을 생성하고 S3에 업로드

사용자 지정 플러그인을 생성하려면 로컬 컴퓨터에서 다음 명령을 실행하여 커넥터와 msk-config-provider가 포함된 zip 파일을 생성합니다.

터미널 창과 Debezium을 커넥터로 사용하여 사용자 지정 플러그인을 만들려면 다음을 수행합니다.

AWSCLI를 사용하여 AWSS3 버킷에 액세스할 수 있는 자격 증명을 사용하여 슈퍼 사용자로 명령을 실행합니다. AWSCLI 설치 및 설정에 대한 자세한 내용은 AWS Command Line Interface사용 설명서의 AWSCLI 시작하기를 참조하세요. Amazon S3에서 AWSCLI를 사용하는 방법에 대한 자세한 내용은 AWS Command Line Interface사용 설명서의 AWSCLI에서 Amazon S3 사용을 참조하세요.

1. 터미널 창에서 다음 명령을 사용하여 Workspace에 `custom-plugin`이라는 이름의 폴더를 생성합니다.

   ```
   mkdir custom-plugin && cd custom-plugin
   ```

2. 다음 명령을 사용하여 Debezium 사이트에서 MySQL 커넥터 플러그인의 안정적인 최신 릴리스를 다운로드하세요.

   ```
   wget https://repo1.maven.org/maven2/io/debezium/debezium-connectormysql/
   2.2.0.Final/debezium-connector-mysql-2.2.0.Final-plugin.tar.gz
   ```

   다음 명령을 사용하여 `custom-plugin` 폴더에서 다운로드한 gzip 파일의 압축을 풉니다.

   ```
   tar xzf debezium-connector-mysql-2.2.0.Final-plugin.tar.gz
   ```

3. 다음 명령을 사용하여 MSK 구성 공급자 zip 파일을 다운로드합니다.

```
wget https://github.com/aws-samples/msk-config-providers/releases/download/r0.4.0/
msk-config-providers-0.4.0-with-dependencies.zip
```

다음 명령을 사용하여 `custom-plugin` 폴더에서 다운로드한 zip 파일의 압축을 풉니다.

```
unzip msk-config-providers-0.4.0-with-dependencies.zip
```

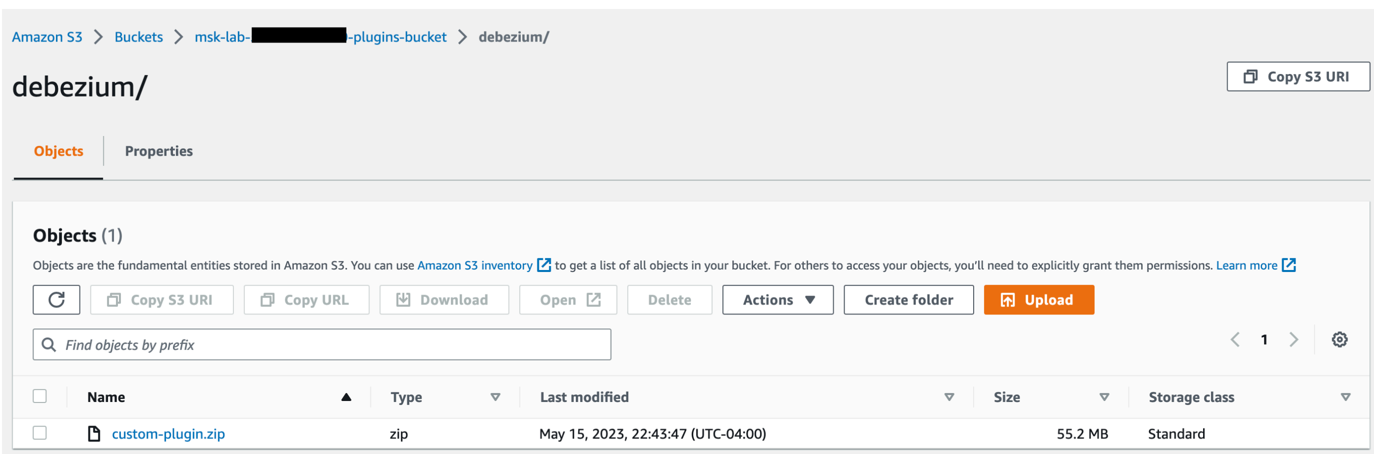
4. 위 단계의 MSK 구성 공급자 콘텐츠와 사용자 지정 커넥터를 `custom-plugin.zip`이라는 단일 파일로 압축합니다.

```
zip -r ../custom-plugin.zip *
```

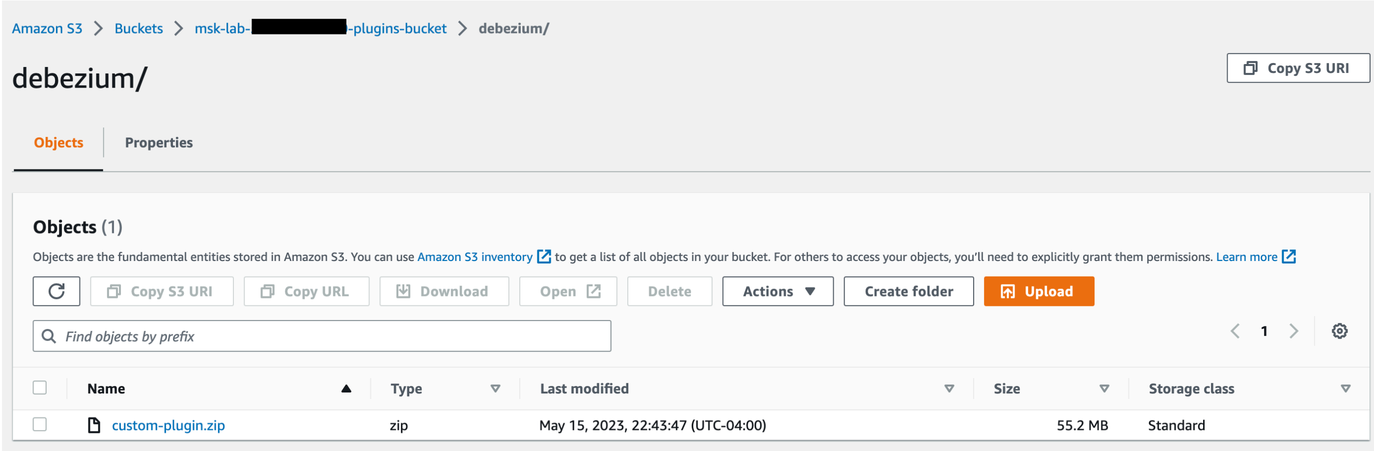
5. 나중에 참조할 수 있도록 S3에 파일을 업로드합니다.

```
aws s3 cp ../custom-plugin.zip s3:<S3_URI_BUCKET_LOCATION>
```

6. Amazon MSK 콘솔의 MSK Connect 섹션에서 사용자 지정 플러그인을 선택한 후 사용자 지정 플러그인 생성을 선택하고, s3:<***S3_URI_BUCKET_LOCATION***> S3 버킷을 탐색하여 방금 업로드한 사용자 지정 플러그인 ZIP 파일을 선택합니다.



7. 플러그인 이름으로 **debezium-custom-plugin**을 입력합니다. 설명을 입력하고 사용자 지정 플러그인 생성을 선택합니다(선택 사항).



다양한 공급자에 대한 파라미터 및 권한 구성

다음 3가지 서비스에서 파라미터 값을 구성할 수 있습니다.

- Secrets Manager
- Systems Manager Parameter Store
- S3 - Simple Storage Service

아래 탭 중 하나를 선택하면 해당 서비스에 대한 파라미터 및 관련 권한 설정에 대한 지침을 확인할 수 있습니다.

Configure in Secrets Manager

Secrets Manager에서 파라미터 값을 구성하려면 다음을 수행합니다.

1. Secrets Manager 콘솔을 엽니다.
2. 보안 인증 정보 또는 비밀을 저장할 새 비밀을 생성합니다. 지침은 AWS Secrets Manager사용 설명서의 AWS Secrets Manager보안 암호 생성을 참조하세요.
3. 보안 암호의 ARN을 복사합니다.
4. 다음 예제 정책의 Secrets Manager 권한을 서비스 실행 역할에 추가합니다. 예제 ARN `arn:aws:secretsmanager:us-east-1:123456789012:secret:MySecret-1234`를 보안 암호의 ARN으로 바꿉니다.
5. 작업자 구성과 커넥터 지침을 추가합니다.

```
{
```

```
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "secretsmanager:GetResourcePolicy",
                    "secretsmanager:GetSecretValue",
                    "secretsmanager:DescribeSecret",
                    "secretsmanager:ListSecretVersionIds"
                ],
                "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:MySecret-1234"
                ]
            }
        ]
    }
```

6. Secrets Manager 구성 공급자를 사용하려면 3단계의 작업자 구성 텍스트 상자에 다음 코드 줄을 복사합니다.

```
# define name of config provider:

config.providers = secretsmanager

# provide implementation classes for secrets manager:

config.providers.secretsmanager.class =
 com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider

# configure a config provider (if it needs additional initialization), for
 example you can provide a region where the secrets or parameters are located:

config.providers.secretsmanager.param.region = us-east-1
```

7. Secrets Manager 구성 공급자 4단계의 커넥터 구성에 다음 코드 줄을 복사합니다.

```
#Example implementation for secrets manager variable
database.user=${secretsmanager:MSKAuroraDBCredentials:username}
database.password=${secretsmanager:MSKAuroraDBCredentials:password}
```

위의 단계를 더 많은 구성 공급자와 함께 사용할 수도 있습니다.

Configure in Systems Manager Parameter Store

Systems Manager Parameter Store에서 파라미터 값을 구성하려면 다음을 수행합니다.

1. Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 파라미터 스토어를 선택합니다.
3. Systems Manager에 저장할 새 파라미터를 생성합니다. 지침은 AWS Systems Manager사용 설명서의 Systems Manager 파라미터 생성(콘솔)을 참조하세요.
4. 파라미터의 ARN을 복사합니다.
5. 다음 예제 정책의 Systems Manager 권한을 서비스 실행 역할에 추가합니다. *<arn:aws:ssm:us-east-1:123456789000:parameter/MyParameterName>*을 파라미터의 ARN으로 변경합니다.

```
{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Sid": "VisualEditor0",
                "Effect": "Allow",
                "Action": [
                    "ssm:GetParameterHistory",
                    "ssm:GetParametersByPath",
                    "ssm:GetParameters",
                    "ssm:GetParameter"
                ],
                "Resource": "arn:aws:ssm:us-east-1:123456789000:parameter/
MyParameterName"
            }
        ]
    }
```

6. 파라미터 저장소 구성 공급자를 사용하려면 3단계의 작업자 구성 텍스트 상자에 다음 코드 줄을 복사합니다.

```
# define name of config provider:

config.providers = ssm

# provide implementation classes for parameter store:
```

```
config.providers.ssm.class =
 com.amazonaws.kafka.config.providers.SsmParamStoreConfigProvider

# configure a config provider (if it needs additional initialization), for
 example you can provide a region where the secrets or parameters are located:

config.providers.ssm.param.region = us-east-1
```

7. 파라미터 저장소 구성 공급자의 경우 5단계의 커넥터 구성에서 다음 코드 줄을 복사합니다.

```
#Example implementation for parameter store variable
schema.history.internal.kafka.bootstrap.servers=
${ssm::MSKBootstrapServerAddress}
```

위의 두 단계를 더 많은 구성 공급자와 함께 번들로 제공할 수도 있습니다.

Configure in Amazon S3

Amazon S3에서 객체/파일을 구성하려면 다음을 수행합니다.

1. Amazon S3 콘솔을 엽니다.
2. S3의 버킷에 객체를 업로드합니다. 지침은 객체 업로드를 참조하세요.
3. 객체의 ARN을 복사합니다.
4. 다음 예제 정책의 Amazon S3 객체 읽기 권한을 서비스 실행 역할에 추가합니다. 예제 ARN `arn:aws:s3:::<MY_S3_BUCKET/path/to/custom-plugin.zip>`을 객체의 ARN으로 바꿉니다.

```
{
    "Version":"2012-10-17",
    "Statement": [
            {
                "Sid": "VisualEditor0",
                "Effect": "Allow",
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3:::<MY_S3_BUCKET/path/to/custom-
plugin.zip>"
            }
        ]
    }
```

5. Amazon S3 구성 공급자를 사용하려면 3단계의 작업자 구성 텍스트 상자에 다음 코드 줄을 복사합니다.

```
# define name of config provider:

config.providers = s3import
# provide implementation classes for S3:

config.providers.s3import.class =
 com.amazonaws.kafka.config.providers.S3ImportConfigProvider
```

6. Amazon S3 구성 공급자의 경우 4단계의 커넥터 구성에 다음 코드 줄을 복사합니다.

```
#Example implementation for S3 object

database.ssl.truststore.location = ${s3import:us-west-2:my_cert_bucket/path/to/
trustore_unique_filename.jks}
```

위의 두 단계를 더 많은 구성 공급자와 함께 번들로 제공할 수도 있습니다.

구성 제공자에 대한 정보를 사용하여 사용자 지정 작업자 구성 생성

1. Amazon MSK Connect 섹션에서 작업자 구성을 선택합니다.
2. 작업자 구성 생성을 선택합니다.
3. 작업자 구성 이름 텍스트 상자에 `SourceDebeziumCustomConfig`를 입력합니다. 설명은 선택 사항입니다.
4. 원하는 공급자에 따라 관련 구성 코드를 복사하여 작업자 구성 텍스트 상자에 붙여넣습니다.
5. 다음은 세 가지 제공자 모두에 대한 작업자 구성의 예입니다.

```
key.converter=org.apache.kafka.connect.storage.StringConverter
key.converter.schemas.enable=false
value.converter=org.apache.kafka.connect.json.JsonConverter
value.converter.schemas.enable=false
offset.storage.topic=offsets_my_debezium_source_connector

# define names of config providers:

config.providers=secretsmanager,ssm,s3import
```

```
# provide implementation classes for each provider:

config.providers.secretsmanager.class    =
 com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider
config.providers.ssm.class               =
 com.amazonaws.kafka.config.providers.SsmParamStoreConfigProvider
config.providers.s3import.class          =
 com.amazonaws.kafka.config.providers.S3ImportConfigProvider

# configure a config provider (if it needs additional initialization), for example
 you can provide a region where the secrets or parameters are located:


config.providers.secretsmanager.param.region = us-east-1
config.providers.ssm.param.region = us-east-1
```

6. 작업자 구성 생성을 클릭합니다.

커넥터 생성

1. 새 커넥터 생성의 지침에 따라 새 커넥터를 생성합니다.
2. ???에서 S3 버킷에 업로드한 `custom-plugin.zip` 파일을 사용자 지정 플러그인의 소스로 선택합니다.
3. 원하는 공급자에 따라 관련 구성 코드를 복사하여 커넥터 구성 필드에 붙여넣습니다.
4. 다음은 세 가지 제공자 모두에 대한 커넥터 구성의 예입니다.

```
#Example implementation for parameter store variable
schema.history.internal.kafka.bootstrap.servers=${ssm::MSKBootstrapServerAddress}

#Example implementation for secrets manager variable
database.user=${secretsmanager:MSKAuroraDBCredentials:username}
database.password=${secretsmanager:MSKAuroraDBCredentials:password}

#Example implementation for Amazon S3 file/object
database.ssl.truststore.location = ${s3import:us-west-2:my_cert_bucket/path/to/
trustore_unique_filename.jks}
```

5. 사용자 지정 구성 사용을 선택하고 작업자 구성 드롭다운에서 SourceDebeziumCustomConfig를 선택합니다.

6. 커넥터 생성에 나와 있는 지침의 나머지 단계를 수행합니다.

MSK Connect의 IAM 역할 및 정책

이 섹션에서는 적절한 IAM 정책 및 역할을 설정하여 AWS환경 내에서 Amazon MSK Connect를 안전하게 배포하고 관리하는 데 도움이 됩니다. 다음 섹션에서는 IAM 인증 MSK 클러스터에 연결할 때 필요한 신뢰 정책 및 추가 권한을 비롯하여 MSK Connect에 사용해야 하는 서비스 실행 역할에 대해 설명합니다. 또한이 페이지에서는 MSK Connect 기능에 대한 전체 액세스 권한을 부여하는 포괄적인 IAM 정책의 예와 서비스에 사용할 수 있는 AWS관리형 정책에 대한 세부 정보를 제공합니다.

주제

서비스 실행 역할 이해

> **Note**
>
> Amazon MSK Connect는 서비스 연결 역할을 서비스 실행 역할로 사용하는 것을 지원하지 않습니다. 별도의 서비스 실행 역할을 생성해야 합니다. 사용자 지정 IAM 역할을 생성하는 방법에 대한 지침은 IAM 사용 설명서의 AWS서비스에 권한을 위임할 역할 생성을 참조하세요.

MSK Connect로 커넥터를 생성할 때는 커넥터와 함께 사용할 AWS Identity and Access Management(IAM) 역할을 지정해야 합니다. 서비스 실행 역할에 다음 신뢰 정책이 있어야 MSK Connect에서 해당 역할을 맡을 수 있습니다. 해당 정책의 조건 컨텍스트 키에 대한 자세한 내용은 the section called “교차 서비스 혼동된 대리자 문제 방지” 섹션을 참조하세요.

JSON

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:kafkaconnect:us-
east-1:123456789012:connector/myConnector/abc12345-abcd-4444-a8b9-123456f513ed-2"
        }
      }
    }
  ]
}
```

커넥터와 함께 사용하려는 Amazon MSK 클러스터가 IAM 인증을 사용하는 클러스터인 경우 커넥터의 서비스 실행 역할에 다음 사용 권한 정책을 추가해야 합니다. 클러스터의 UUID를 찾는 방법과 토픽 ARN을 구성하는 방법에 대한 자세한 내용은 the section called “권한 부여 정책 리소스” 섹션을 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:000000000001:cluster/
testClusterName/300d0000-0000-0005-000f-00000000000b-1"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:ReadData",
                "kafka-cluster:DescribeTopic"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:topic/
myCluster/300a0000-0000-0003-000a-00000000000b-6/__amazon_msk_connect_read"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:WriteData",
                "kafka-cluster:DescribeTopic"
            ],
            "Resource": [
                "arn:aws:kafka:us-east-1:123456789012:topic/
testCluster/300f0000-0000-0008-000d-00000000000m-7/__amazon_msk_connect_write"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:CreateTopic",
                "kafka-cluster:WriteData",
                "kafka-cluster:ReadData",
                "kafka-cluster:DescribeTopic"
            ],
            "Resource": [
                "arn:aws:kafka:us-
east-1:123456789012:topic/testCluster/300f0000-0000-0008-000d-00000000000m-7/
__amazon_msk_connect_*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
```

```
                "arn:aws:kafka:us-
east-1:123456789012:group/testCluster/300d0000-0000-0005-000f-00000000000b-1/
__amazon_msk_connect_*",
                "arn:aws:kafka:us-
east-1:123456789012:group/testCluster/300d0000-0000-0005-000f-00000000000b-1/
connect-*"
            ]
        }
    ]
}
```

커넥터 종류에 따라 AWS리소스에 액세스할 수 있도록 허용하는 권한 정책을 서비스 실행 역할에 연결해야 할 수도 있습니다. 예를 들어 커넥터에서 S3 버킷으로 데이터를 전송해야 하는 경우 서비스 실행 역할에 해당 버킷에 쓸 수 있는 권한을 부여하는 권한 정책이 있어야 합니다. 테스트 목적으로 전체 액세스 권한을 부여하는 사전 구축된 IAM 정책 중 하나를 사용할 수 있습니다(예: `arn:aws:iam::aws:policy/AmazonS3FullAccess`). 그러나 보안을 위해 커넥터가 AWS소스에서 읽거나 AWS싱크에 쓸 수 있도록 허용하는 가장 제한적인 정책을 사용하는 것이 좋습니다.

MSK Connect의 IAM 정책 예제

관리자가 아닌 사용자에게 모든 MSK Connect 기능에 대한 전체 액세스 권한을 부여하려면 사용자의 IAM 역할에 다음과 같은 정책을 연결하세요.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "MSKConnectFullAccess",
      "Effect": "Allow",
      "Action": [
        "kafkaconnect:CreateConnector",
        "kafkaconnect:DeleteConnector",
        "kafkaconnect:DescribeConnector",
        "kafkaconnect:ListConnectors",
        "kafkaconnect:UpdateConnector",
        "kafkaconnect:CreateCustomPlugin",
        "kafkaconnect:DeleteCustomPlugin",
        "kafkaconnect:DescribeCustomPlugin",
```

```
        "kafkaconnect:ListCustomPlugins",
        "kafkaconnect:CreateWorkerConfiguration",
        "kafkaconnect:DeleteWorkerConfiguration",
        "kafkaconnect:DescribeWorkerConfiguration",
        "kafkaconnect:ListWorkerConfigurations"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/MSKConnectServiceRole",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "kafkaconnect.amazonaws.com"
        }
      }
    },
    {
      "Sid": "EC2NetworkAccess",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
      ],
      "Resource": "*"
    },
    {
      "Sid": "MSKClusterAccess",
      "Effect": "Allow",
      "Action": [
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka:GetBootstrapBrokers"
      ],
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/myCluster/"
    },
    {
      "Sid": "MSKLogGroupAccess",
```

```
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:/aws/msk-connect/*"
      ]
    },
    {
      "Sid": "S3PluginAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1-custom-plugins",
        "arn:aws:s3:::amzn-s3-demo-bucket1-custom-plugins/*"
      ]
    }
  ]
}
```

교차 서비스 혼동된 대리자 문제 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 위탁자를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 [`aws:SourceArn`] 및 [`aws:SourceAccount`] 전역 조건 컨텍스트 키를 사용하여 MSK Connect가 다른 서비스에 리소스에 부여하는 권한을 제한하는 것을 권장합니다. `aws:SourceArn` 값에 계정 ID가 포함되어 있지 않은 경우(예: Amazon S3 버킷 ARN에 계정 ID가 포함되어 있지

않은 경우) 두 전역 조건 컨텍스트 키를 모두 사용하여 권한을 제한해야 합니다. 두 전역 조건 컨텍스트 키와 계정을 포함한 `aws:SourceArn` 값을 모두 사용하는 경우, `aws:SourceAccount` 값 및 `aws:SourceArn` 값의 계정은 동일한 정책 명령문에서 사용할 경우 반드시 동일한 계정 ID를 사용해야 합니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 `aws:SourceArn`을 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 `aws:SourceAccount`를 사용하세요.

MSK Connect의 경우 `aws:SourceArn` 값은 MSK 커넥터여야 합니다.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 `aws:SourceArn`글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모를 경우 또는 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 `aws:SourceArn`전역 조건 컨텍스트 키를 사용합니다. 예를 들어 *`arn:aws:kafkaconnect:us-east-1:123456789012:connector/*`*는 미국 동부(버지니아 북부) 리전의 ID가 123456789012인 계정에 속한 모든 커넥터를 나타냅니다.

다음 예에서는 MSK Connect에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 대리자 혼동 문제를 방지하는 방법을 보여줍니다. *`123456789012`* 및 arn:aws:kafkaconnect:*`us-east-1`*:*`123456789012`*:connector/*`my-S3-Sink-Connector`*/*`abcd1234-5678-90ab-cdef-1234567890ab`*를 AWS 계정및 커넥터 정보로 바꿉니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": " kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
```

```
        "aws:SourceArn": "arn:aws:kafkaconnect:us-
east-1:123456789012:connector/my-S3-Sink-Connector/abcd1234-5678-90ab-
cdef-1234567890ab"
        }
      }
    }
  ]
}
```

AWSMSK Connect에 대한 관리형 정책

AWS관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다AWS. AWS관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS관리형 정책은 모든 AWS고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS업데이트하는AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 미칩니다. AWSAWS 서비스는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 되면 AWS관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 가이드의 AWS 관리형 정책을 참조하세요.

AWS관리형 정책: AmazonMSKConnectReadOnlyAccess

해당 정책은 사용자에게 MSK Connect 리소스를 나열하고 설명하는 데 필요한 권한을 부여합니다.

`AmazonMSKConnectReadOnlyAccess` 정책을 IAM ID에 연결할 수 있습니다.

JSON

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

```
                "kafkaconnect:ListConnectors",
                "kafkaconnect:ListCustomPlugins",
                "kafkaconnect:ListWorkerConfigurations"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafkaconnect:DescribeConnector"
            ],
            "Resource": [
                "arn:aws:kafkaconnect:*:*:connector/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafkaconnect:DescribeCustomPlugin"
            ],
            "Resource": [
                "arn:aws:kafkaconnect:*:*:custom-plugin/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafkaconnect:DescribeWorkerConfiguration"
            ],
            "Resource": [
                "arn:aws:kafkaconnect:*:*:worker-configuration/*"
            ]
        }
    ]
}
```

AWS관리형 정책: KafkaConnectServiceRolePolicy

해당 정책은 MSK Connect 서비스에 `AmazonMSKConnectManaged:true` 태그가 있는 네트워크 인터페이스를 생성하고 관리하는 데 필요한 권한을 부여합니다. 이러한 네트워크 인터페이스를 통해 MSK Connect 네트워크는 Apache Kafka 클러스터나 소스 또는 싱크와 같은 Amazon VPC의 리소스에 액세스할 수 있습니다.

KafkaConnectServiceRolePolicy를 IAM 엔터티에 연결할 수 없습니다. 이 정책은 MSK Connect가 사용자를 대신하여 작업을 수행할 수 있도록 하는 서비스 연결 역할에 연결되어 있습니다.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface"
   ],
   "Resource": "arn:aws:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "aws:RequestTag/AmazonMSKConnectManaged": "true"
    },
    "ForAllValues:StringEquals": {
     "aws:TagKeys": "AmazonMSKConnectManaged"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:subnet/*",
    "arn:aws:ec2:*:*:security-group/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": "arn:aws:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": "CreateNetworkInterface"
    }
```

```
      }
    },
    {
     "Effect": "Allow",
     "Action": [
      "ec2:DescribeNetworkInterfaces",
      "ec2:CreateNetworkInterfacePermission",
      "ec2:AttachNetworkInterface",
      "ec2:DetachNetworkInterface",
      "ec2:DeleteNetworkInterface"
     ],
     "Resource": "arn:aws:ec2:*:*:network-interface/*",
     "Condition": {
      "StringEquals": {
       "ec2:ResourceTag/AmazonMSKConnectManaged": "true"
      }
     }
    }
  ]
}
```

AWS관리형 정책에 대한 MSK Connect 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 MSK Connect의 AWS관리형 정책 업데이트에 대한 세부 정보를 봅니다.

변경	설명	Date
MSK Connect 읽기 전용 정책 업데이트	MSK Connect는 리스팅 작업에 대한 제한을 제거하기 위해 AmazonMSKConnectReadOnlyAccess 정책을 업데이트했습니다.	2021년 10월 13일
MSK Connect에서 변경 사항 추적 시작	MSK Connect가 AWS관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 9월 14일

MSK Connect에 서비스 연결 역할 사용

Amazon MSK Connect는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 MSK Connect에 직접 연결되는 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 MSK Connect에서 사전 정의하며 서비스가 사용자를 대신하여 다른 AWS서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 MSK Connect를 더 간편하게 설정할 수 있습니다. MSK Connect는 서비스 연결 역할의 권한을 정의하며 달리 정의되지 않는 한 MSK Connect만이 해당 역할을 수행할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 Service-Linked Role(서비스 연결 역할) 열에 Yes(예)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

MSK Connect에 대한 서비스 연결 역할 권한

MSK Connect는 AWSServiceRoleForKafkaConnect라는 서비스 연결 역할을 사용하며 이 역할을 통해 Amazon MSK Connect가 사용자를 대신하여 Amazon 리소스에 액세스할 수 있습니다.

AWSServiceRoleForKafkaConnect 서비스 연결 역할은 `kafkaconnect.amazonaws.com` 서비스를 신뢰하여 역할을 맡습니다.

역할이 사용하는 권한 정책에 대한 자세한 내용은 the section called “KafkaConnectServiceRolePolicy” 섹션을 참조하세요.

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한 섹션을 참조하세요.

MSK Connect에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Management 콘솔AWS CLI, 또는 AWSAPI에서 커넥터를 생성하면 MSK Connect가 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 커넥터를 생성하면 MSK Connect에서 서비스 연결 역할을 다시 생성합니다.

MSK Connect에 대한 서비스 연결 역할 편집

MSK Connect에서는 AWSServiceRoleForKafkaConnect 서비스 연결 역할을 편집할 수 없습니다. 서비스 연결 역할을 생성한 후에는 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

MSK Connect의 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI또는 AWSAPI를 사용하여 서비스 연결 역할을 수동으로 삭제할 수 있습니다. 이렇게 하려면 먼저 모든 MSK Connect 커넥터를 수동으로 삭제한 다음 역할을 수동으로 삭제해야 합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

MSK Connect 서비스 연결 역할에 대해 지원되는 리전

MSK Connect는 서비스를 이용할 수 있는 모든 리전에서 서비스 연결 역할을 사용할 수 있도록 지원합니다. 자세한 설명은 AWS리전 및 엔드포인트를 참조하세요.

Amazon MSK Connect에 대한 인터넷 액세스 활성화

Amazon MSK Connect용 커넥터에 인터넷 액세스가 필요한 경우 다음 Amazon Virtual Private Cloud(VPC) 설정을 사용하여 해당 액세스를 사용하도록 설정하는 것을 권장합니다.

- 커넥터를 프라이빗 서브넷으로 구성합니다.
- 퍼블릭 서브넷에서 VPC를 위한 퍼블릭 NAT 게이트웨이 또는 NAT 인스턴스를 생성합니다. 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 NAT 디바이스를 사용하여 인터넷 또는 다른 VPC에 서브넷 연결 페이지를 참조하세요.
- 프라이빗 서브넷에서 NAT 게이트웨이 또는 인스턴스로의 아웃바운드 트래픽을 허용합니다.

Amazon MSK Connect에 대한 NAT 게이트웨이 설정

다음 단계에서는 커넥터의 인터넷 액세스를 사용하도록 NAT 게이트웨이를 설정하는 방법을 설명합니다. 프라이빗 서브넷에서 커넥터를 생성하기 전에 이 단계를 완료해야 합니다.

NAT 게이트웨이 설정을 위한 전체 사전 조건

다음 항목이 준비되어 있는지 확인하세요.

- 클러스터와 관련된 Amazon Virtual Private Cloud(VPC) ID. 예를 들어 vpc-123456ab입니다.

- VPC에 있는 프라이빗 서브넷 ID. 예를 들어 subnet-a1b2c3de, subnet-f4g5h6ij 등입니다. 커넥터를 프라이빗 서브넷으로 구성해야 합니다.

커넥터에 대한 인터넷 액세스를 활성화하는 단계

커넥터에 인터넷 액세스를 활성화하려면 다음을 수행합니다.

1. https://console.aws.amazon.com/vpc/에서 Amazon Virtual Private Cloud 콘솔을 엽니다.
2. 설명이 포함된 이름으로 NAT 게이트웨이의 퍼블릭 서브넷을 생성하고 서브넷 ID를 기록합니다. 자세한 지침은 VPC에서 서브넷 생성을 참조하세요.
3. VPC가 인터넷과 통신할 수 있도록 인터넷 게이트웨이를 생성하고 게이트웨이 ID를 기록해 두세요. 인터넷 게이트웨이를 VPC에 연결합니다. 지침은 인터넷 게이트웨이 생성 및 연결을 참조하세요.
4. 프라이빗 서브넷의 호스트가 퍼블릭 서브넷에 연결될 수 있도록 퍼블릭 NAT 게이트웨이를 프로비저닝합니다. NAT 게이트웨이를 생성하는 경우 이전에 생성한 퍼블릭 서브넷을 선택합니다. 지침은 NAT 게이트웨이 생성을 참조하세요.
5. 라우팅 테이블을 구성합니다. 이 설정을 완료하려면 총 2개의 라우팅 테이블이 있어야 합니다. 이미 VPC와 동시에 자동으로 생성된 기본 라우팅 테이블이 있어야 합니다. 이 단계에서는 퍼블릭 서브넷에 대한 추가 라우팅 테이블을 생성합니다.

    a. 다음 설정을 사용하여 프라이빗 서브넷이 NAT 게이트웨이로 트래픽을 라우팅하도록 VPC의 기본 라우팅 테이블을 수정합니다. 지침은 Amazon Virtual Private Cloud 사용 설명서에서 라우팅 테이블로 작업을 참조하세요.

    프라이빗 MSKC 라우팅 테이블

속성	값
[Name tag]	이 라우팅 테이블을 식별하는 데 도움이 되도록 설명이 포함된 이름 태그 지정을 권장합니다. 프라이빗 MSKC를 예로 들 수 있습니다.
관련 서브넷	프라이빗 서브넷
MSK Connect에 대한 인터넷 액세스를 활성화하는 경로	• 대상 주소: 0.0.0.0/0

속성	값
	• 대상: 사용자의 NAT 게이트웨이 ID. nat-12a345bc6789efg1h를 예로 들 수 있습니다.
내부 트래픽을 위한 로컬 경로	• 대상 주소: 10.0.0.0/16. 이 값은 VPC의 CIDR 블록에 따라 다를 수 있습니다. • 대상: 로컬

b. 사용자 지정 라우팅 테이블 생성의 지침에 따라 퍼블릭 서브넷에 대한 라우팅 테이블을 생성합니다. 테이블을 생성할 때 테이블이 연결된 서브넷을 식별할 수 있도록 이름 태그 필드에 설명이 포함된 이름을 입력합니다. 퍼블릭 MSKC를 예로 들 수 있습니다.

c. 다음 설정을 사용하여 퍼블릭 MSKC 라우팅 테이블을 구성합니다.

속성	값
[Name tag]	퍼블릭 MSKC 또는 사용자가 선택한 설명이 포함된 다른 이름
관련 서브넷	NAT 게이트웨이가 포함된 퍼블릭 서브넷
MSK Connect에 대한 인터넷 액세스를 활성화하는 경로	• 대상 주소: 0.0.0.0/0 • 대상: 사용자의 인터넷 게이트웨이 ID. igw-1a234bc5를 예로 들 수 있습니다.
내부 트래픽을 위한 로컬 경로	• 대상 주소: 10.0.0.0/16. 이 값은 VPC의 CIDR 블록에 따라 다를 수 있습니다. • 대상: 로컬

프라이빗 DNS 호스트 이름 이해

MSK Connect에서 프라이빗 DNS 호스트 이름을 지원하므로 퍼블릭 또는 프라이빗 도메인 이름을 참조하도록 커넥터를 구성할 수 있습니다. 지원 여부는 VPC DHCP 옵션 세트에 지정된 DNS 서버에 따라 달라집니다.

DHCP 옵션 세트는 EC2 인스턴스가 VPC 네트워크를 통해 통신하기 위해 VPC에서 사용하는 네트워크 구성 그룹입니다. 각 VPC에는 기본 DHCP 옵션 세트가 있지만 VPC의 인스턴스가 도메인 이름 확인에 Amazon 제공 DNS 서버 대신 다른 DNS 서버를 사용하도록 하려는 경우 사용자 지정 DHCP 옵션 세트를 생성할 수 있습니다. Amazon VPC의 DHCP 옵션 세트를 참조하세요.

프라이빗 DNS 확인 기능/특성이 MSK Connect에 포함되기 전에는 커넥터가 고객 커넥터의 DNS 쿼리에 서비스 VPC DNS 해석기를 사용했습니다. 커넥터가 DNS 확인을 위해 고객 VPC DHCP 옵션 세트에 정의된 DNS 서버를 사용하지 않았습니다.

커넥터는 공개적으로 확인할 수 있는 고객 커넥터 구성 또는 플러그인에서 호스트 이름만 참조할 수 있었습니다. 프라이빗 호스팅 영역에 정의된 프라이빗 호스트 이름을 확인하거나 다른 고객 네트워크의 DNS 서버를 사용할 수 없었습니다.

프라이빗 DNS가 없으면 데이터베이스, 데이터 웨어하우스, 자체 VPC의 Secrets Manager와 같은 시스템을 인터넷에서 액세스할 수 없도록 설정한 고객은 MSK 커넥터로 작업할 수 없습니다. 고객은 기업 보안 태세를 준수하기 위해 종종 프라이빗 DNS 호스트 이름을 사용합니다.

커넥터에 대한 VPC DHCP 옵션 세트 구성

커넥터는 커넥터가 생성될 때 자동으로 VPC DHCP 옵션 세트에 정의된 DNS 서버를 사용합니다. 커넥터를 생성하기 전에 커넥터의 DNS 호스트 이름 확인 요구 사항에 맞도록 VPC DHCP 옵션 세트를 구성해야 합니다.

MSK Connect에서 프라이빗 DNS 호스트 이름 기능을 사용할 수 있게 되기 전에 생성한 커넥터는 수정할 필요 없이 이전 DNS 확인 구성을 계속 사용합니다.

커넥터에서 공개적으로 확인 가능한 DNS 호스트 이름 확인만 필요한 경우 더 간편하게 설정하려면 커넥터를 만들 때 계정의 기본 VPC를 사용하는 것을 권장합니다. Amazon에서 제공하는 DNS 서버 또는 Amazon Route 53 Resolver에 대한 자세한 내용은 Amazon VPC 사용 설명서의 Amazon DNS 서버를 참조하세요.

프라이빗 DNS 호스트 이름을 확인해야 하는 경우 커넥터 생성 시 전달된 VPC에 DHCP 옵션이 올바르게 설정되어 있는지 확인합니다. 자세한 내용은 Amazon VPC 사용 설명서에서 DHCP 옵션 세트를 사용하여 작업을 참조하세요.

프라이빗 DNS 호스트 이름 확인을 위한 DHCP 옵션 세트를 구성하는 경우 커넥터가 DHCP 옵션 세트에서 구성한 사용자 지정 DNS 서버에 연결할 수 있는지 확인합니다. 그렇지 않으면 커넥터 생성에 실패합니다.

VPC DHCP 옵션 세트를 사용자 지정하면 이후 해당 VPC에서 생성되는 커넥터는 옵션 세트에서 지정한 DNS 서버를 사용합니다. 커넥터를 생성한 후 옵션 세트를 변경하면 커넥터는 몇 분 내에 새 옵션 세트의 설정을 적용합니다.

VPC의 DNS 속성 구성

Amazon VPC 사용 설명서의 VPC의 DNS 속성 및 DNS 호스트 이름에 설명된 대로 VPC DNS 속성을 올바르게 구성했는지 확인합니다.

인바운드 및 아웃바운드 해석기 엔드포인트를 사용하여 다른 네트워크를 VPC에 연결하여 커넥터와 함께 작업하는 방법에 대한 자세한 내용은 Amazon Route 53 개발자 안내서의 VPC와 네트워크 간의 DNS 쿼리 확인을 참조하세요.

커넥터 생성 오류 처리

이 섹션에서는 DNS 확인과 관련하여 발생할 수 있는 커넥터 생성 실패 및 문제 해결을 위한 권장 조치에 대해 설명합니다.

실패	권장 조치
DNS 확인 쿼리가 실패하거나 커넥터에서 DNS 서버에 연결할 수 없는 경우 커넥터 생성에 실패합니다.	커넥터에 대해 이러한 로그를 구성한 경우 실패한 DNS 확인 쿼리로 인한 커넥터 생성 실패를 CloudWatch Logs에서 확인할 수 있습니다. DNS 서버 구성을 확인하고 커넥터에서 DNS 서버에 네트워크가 연결되어 있는지 확인합니다.
커넥터가 실행 중일 때 VPC DHCP 옵션 세트에서 DNS 서버 구성을 변경하면 커넥터의 DNS 확인 쿼리가 실패할 수 있습니다. DNS 확인에 실패하면 일부 커넥터 작업이 실패 상태로 전환될 수 있습니다.	커넥터에 대해 이러한 로그를 구성한 경우 실패한 DNS 확인 쿼리로 인한 커넥터 생성 실패를 CloudWatch Logs에서 확인할 수 있습니다. 실패한 작업은 자동으로 다시 시작되어 커넥터를 다시 활성화해야 합니다. 그렇지 않은 경우 지원팀에 문의하여 해당 커넥터의 실패한 작업을 다시 시작하거나 커넥터를 다시 생성할 수 있습니다.

MSK Connect에 대한 보안

AWS PrivateLink로 구동되는 인터페이스 VPC 엔드포인트를 사용하여 Amazon VPC와 Amazon MSK-Connect 호환 API 간의 트래픽이 Amazon 네트워크를 벗어나지 않도록 방지할 수 있습니다. 인터페이스 VPC 엔드포인트에는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결이 필요하지 않습니다. 자세한 내용은 인터페이스 VPC 엔드포인트와 함께 Amazon MSK API 사용 섹션을 참조하세요.

MSK Connect 로깅

MSK Connect는 커넥터 디버깅에 사용할 수 있는 로그 이벤트를 작성할 수 있습니다. 커넥터를 생성할 때 다음 로그 대상을 0개 이상 지정할 수 있습니다.

- Amazon CloudWatch Logs: MSK Connect가 커넥터의 로그 이벤트를 전송할 로그 그룹을 지정합니다. 로그 그룹을 생성하는 방법에 대한 자세한 내용은 CloudWatch Logs 사용 설명서에서 로그 그룹 생성을 참조하세요.
- Amazon S3: MSK Connect가 커넥터의 로그 이벤트를 전송할 S3 버킷을 지정합니다. S3 버킷을 생성하는 방법에 대한 자세한 내용은 Amazon S3 사용 설명서에서 버킷 생성을 참조하세요.
- Amazon Data Firehose: MSK Connect가 커넥터의 로그 이벤트를 전송할 전송 스트림을 지정합니다. 전송 스트림을 생성하는 방법에 대한 자세한 내용은 Firehose 사용 설명서에서 Amazon Data Firehose 전송 스트림 생성을 참조하세요.

로깅 설정에 대해 자세히 알아보려면 Amazon CloudWatch Logs 사용 설명서의 특정 AWS 서비스에서 로깅 활성화를 참조하세요.

MSK Connect는 다음 유형의 로그 이벤트를 내보냅니다.

수준	설명
INFO	시작과 종료 시 관심 있는 런타임 이벤트
WARN	오류는 아니지만 바람직하지 않거나 예상하지 못한 런타임 상황
FATAL	조기 종료의 원인이 되는 심각한 오류

수준	설명
ERROR	치명적이지 않은 예상하지 못한 조건 및 런타임 오류

다음은 CloudWatch Logs로 전송되는 로그 이벤트의 예제입니다.

```
[Worker-0bb8afa0b01391c41] [2021-09-06 16:02:54,151] WARN [Producer
 clientId=producer-1] Connection to node 1 (b-1.my-test-cluster.twwhtj.c2.kafka.us-
east-1.amazonaws.com/INTERNAL_IP) could not be established. Broker may not be
 available. (org.apache.kafka.clients.NetworkClient:782)
```

커넥터 로그에 비밀이 표시되지 않도록 하기

Note

플러그인이 해당 값을 보안 암호로 정의하지 않은 경우 커넥터 로그에 민감한 구성 값이 표시될 수 있습니다. Kafka Connect는 정의되지 않은 구성 값을 기타 일반 텍스트 값과 동일하게 취급합니다.

플러그인에서 속성을 비밀로 정의하는 경우 Kafka Connect는 커넥터 로그에서 속성 값을 삭제합니다. 예를 들어 다음 커넥터 로그는 플러그인이 `aws.secret.key`를 `PASSWORD` 유형으로 정의하면 해당 값이 **`[hidden]`**으로 대체됨을 보여줍니다.

```
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] [2022-01-11
 15:18:55,150] INFO SecretsManagerConfigProviderConfig values:
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] aws.access.key =
 my_access_key
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] aws.region = us-east-1
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] aws.secret.key
 = [hidden]
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] secret.prefix =
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b] secret.ttl.ms = 300000
    2022-01-11T15:18:55.000+00:00    [Worker-05e6586a48b5f331b]
 (com.github.jcustenborder.kafka.config.aws.SecretsManagerConfigProviderConfig:361)
```

커넥터 로그 파일에 비밀이 표시되지 않도록 하려면 플러그인 개발자는 Kafka Connect 열거형 상수 [`ConfigDef.Type.PASSWORD`]를 사용하여 민감한 속성을 정의해야 합니다. 속성이

`ConfigDef.Type.PASSWORD` 유형인 경우 값이 일반 텍스트로 전송되더라도 Kafka Connect는 해당 값을 커넥터 로그에서 제외합니다.

Amazon MSK Connect 모니터링

모니터링은 MSK Connect 및 기타 AWS 솔루션의 신뢰성, 가용성, 성능을 유지하는 데 있어 중요한 부분입니다. Amazon CloudWatch는 AWS에서 실행하는 AWS 리소스와 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch가 CPU 사용량이나 커넥터의 기타 지표를 추적하도록 설정하여 필요한 경우 용량을 늘릴 수 있습니다. 자세한 내용은 [Amazon CloudWatch 사용 설명서]를 참조하세요.

다음 API 작업을 사용할 수 있습니다.

- `DescribeConnectorOperation`: 커넥터 업데이트 작업의 상태를 모니터링합니다.
- `ListConnectorOperations`: 커넥터에서 실행된 이전 업데이트를 추적합니다.

다음 표는 MSK Connect가 `ConnectorName` 차원에 따라 CloudWatch에 전송하는 지표를 보여줍니다. MSK Connect는 이러한 지표를 추가 비용 없이 기본 제공합니다. CloudWatch는 이러한 지표를 15개월 동안 보관하므로 과거 정보에 액세스하고 커넥터의 성과를 더 잘 파악할 수 있습니다. 특정 임곗값을 주시하다가 해당 임곗값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 [Amazon CloudWatch 사용 설명서]를 참조하세요.

메트릭 이름	설명
`CpuUtilization`	시스템 및 사용자별 CPU 사용량 비율입니다.
`ErroredTaskCount`	오류가 발생한 작업의 수입니다.
`MemoryUtilization`	현재 사용 중인 Java 가상 머신(JVM) 힙 메모리뿐만 아니라 작업자 인스턴스의 전체 메모리에서 차지하는 비율입니다. JVM은 일반적으로 운영 체제로 메모리를 다시 릴리스하지 않습니다. 따라서 JVM 힙 크기(MemoryUtilization)는 일반적으로 최소 힙 크기로 시작하여 점진적으로 최대 약 80~90%까지 안정적으로 증가합니다. 커넥터의 실제 메모리 사용량이 변경됨에 따라

메트릭 이름	설명
	JVM 힙 사용량이 증가하거나 감소할 수 있습니다.
RebalanceCompletedTotal	해당 커넥터가 완료한 총 재조정 횟수입니다.
RebalanceTimeAvg	커넥터가 재조정에 소요한 평균 시간(밀리초)입니다.
RebalanceTimeMax	커넥터가 재조정에 소요한 최대 시간(밀리초)입니다.
RebalanceTimeSinceLast	해당 커넥터가 가장 최근에 재조정을 완료한 후의 시간(밀리초)입니다.
RunningTaskCount	커넥터에서 실행 중인 작업 수입니다.
SinkConsumerByteRate	변환이 데이터에 적용되기 전에 Kafka Connect 프레임워크의 싱크 컨슈머가 초당 컨슘한 평균 바이트 수입니다.
SinkRecordReadRate	Apache Kafka나 Amazon MSK 클러스터에서 읽은 초당 평균 레코드 수입니다.
SinkRecordSendRate	변환에서 출력되어 대상으로 전송되는 초당 평균 레코드 수입니다. 이 숫자에는 필터링된 레코드가 포함되지 않습니다.
SourceRecordPollRate	생성되거나 폴링된 초당 평균 레코드 수입니다.
SourceProducerByteRate	변환이 데이터에 적용된 후 Kafka Connect 프레임워크의 소스 프로듀서가 초당 생성하는 평균 바이트 수입니다.
SourceRecordWriteRate	변환에서 출력되어 Apache Kafka 또는 Amazon MSK 클러스터에 기록되는 초당 평균 레코드 수입니다.

메트릭 이름	설명
TaskStartupAttemptsTotal	커넥터가 시도한 총 작업 시작 횟수입니다. 이 지표를 사용하여 작업 시작 시도의 이상 징후를 식별할 수 있습니다.
TaskStartupSuccessPercentage	커넥터에 대한 성공적인 태스크 시작의 평균 비율입니다. 이 지표를 사용하여 작업 시작 시도의 이상 징후를 식별할 수 있습니다.
WorkerCount	커넥터에서 실행 중인 작업자 수입니다.
BytesInPerSec	작업자 간의 통신을 위해 Kafka Connect 프레임워크로 전송된 메타데이터 바이트입니다.
BytesOutPerSec	작업자 간의 통신을 위해 Kafka Connect 프레임워크에서 전송된 메타데이터 바이트입니다.

Amazon MSK Connect 리소스 설정 예제

이 섹션에는 일반적인 타사 커넥터 및 구성 공급자와 같은 Amazon MSK Connect 리소스를 설정하는 데 도움이 되는 예제가 포함되어 있습니다.

주제

Amazon S3 싱크 커넥터 설정

이 예제에서는 Confluent Amazon S3 싱크 커넥터와 AWS CLI를 사용하여 MSK Connect에서 Amazon S3 싱크 커넥터를 생성하는 방법을 보여줍니다.

1. 다음 JSON을 복사하여 새 파일에 붙여넣습니다. 자리 표시자 문자열을 Amazon MSK 클러스터의 부트스트랩 서버 연결 문자열 및 클러스터의 서브넷 및 보안 그룹 ID에 해당하는 값으로 변경합니다. 서비스 실행 역할을 설정하는 방법에 대한 자세한 내용은 the section called “IAM 역할 및 정책” 섹션을 참조하세요.

```
{
    "connectorConfiguration": {
        "connector.class": "io.confluent.connect.s3.S3SinkConnector",
        "s3.region": "us-east-1",
        "format.class": "io.confluent.connect.s3.format.json.JsonFormat",
        "flush.size": "1",
        "schema.compatibility": "NONE",
        "topics": "my-test-topic",
        "tasks.max": "2",
        "partitioner.class":
 "io.confluent.connect.storage.partitioner.DefaultPartitioner",
        "storage.class": "io.confluent.connect.s3.storage.S3Storage",
        "s3.bucket.name": "amzn-s3-demo-bucket"
    },
    "connectorName": "example-S3-sink-connector",
    "kafkaCluster": {
        "apacheKafkaCluster": {
            "bootstrapServers": "<cluster-bootstrap-servers-string>",
            "vpc": {
                "subnets": [
                    "<cluster-subnet-1>",
                    "<cluster-subnet-2>",
                    "<cluster-subnet-3>"
                ],
                "securityGroups": ["<cluster-security-group-id>"]
            }
        }
    },
    "capacity": {
        "provisionedCapacity": {
            "mcuCount": 2,
            "workerCount": 4
        }
    },
    "kafkaConnectVersion": "2.7.1",
    "serviceExecutionRoleArn": "<arn-of-a-role-that-msk-connect-can-assume>",
    "plugins": [
        {
            "customPlugin": {
                "customPluginArn": "<arn-of-custom-plugin-that-contains-connector-
code>",
                "revision": 1
            }
```

```
        }
    ],
    "kafkaClusterEncryptionInTransit": {"encryptionType": "PLAINTEXT"},
    "kafkaClusterClientAuthentication": {"authenticationType": "NONE"}
}
```

2. 이전 단계에서 JSON 파일을 저장한 폴더에서 다음 AWS CLI명령을 실행합니다.

```
aws kafkaconnect create-connector --cli-input-json file://connector-info.json
```

다음은 명령을 성공적으로 실행했을 때 표시되는 출력의 예제입니다.

```
{
    "ConnectorArn": "arn:aws:kafkaconnect:us-east-1:123450006789:connector/example-
S3-sink-connector/abc12345-abcd-4444-a8b9-123456f513ed-2",
    "ConnectorState": "CREATING",
    "ConnectorName": "example-S3-sink-connector"
}
```

MSK Connect용 EventBridge Kafka 싱크 커넥터 설정

이 주제에서는 MSK Connect용 EventBridge Kafka 싱크 커넥터를 설정하는 방법을 보여줍니다. 이 커넥터를 사용하면 MSK 클러스터에서 EventBridge 이벤트 버스로 이벤트를 보낼 수 있습니다. 이 주제에서는 필수 리소스를 생성하고 Kafka와 EventBridge 간의 원활한 데이터 흐름을 지원하도록 커넥터를 구성하는 프로세스에 대해 설명합니다.

주제

- 사전 조건
- MSK Connect에 필요한 리소스 설정
- 커넥터 생성
- Kafka로 메시지 전송

사전 조건

커넥터를 배포하기 전에 다음 리소스가 있는지 확인합니다.

- Amazon MSK 클러스터: Kafka 메시지를 생성하고 사용하기 위한 활성 MSK 클러스터입니다.

- Amazon EventBridge 이벤트 버스: Kafka 토픽에서 이벤트를 수신하는 EventBridge 이벤트 버스입니다.
- IAM 역할: MSK Connect 및 EventBridge 커넥터에 필요한 권한이 있는 IAM 역할을 생성합니다.
- MSK 클러스터의 VPC 및 서브넷에 생성된 EventBridge용 VPC 인터페이스 엔드포인트 또는 MSK Connect에서 퍼블릭 인터넷에 액세스합니다. 그러면 NAT 게이트웨이 없이 퍼블릭 인터넷을 통과하지 않아도 됩니다.
- 토픽을 생성하고 Kafka로 레코드를 전송하는 Amazon EC2 인스턴스 또는 AWS CloudShell과 같은 클라이언트 머신입니다.

MSK Connect에 필요한 리소스 설정

커넥터에 대한 IAM 역할을 생성한 후 커넥터를 생성합니다. 또한 EventBridge 규칙을 생성하여 EventBridge 이벤트 버스로 전송된 Kafka 이벤트를 필터링합니다.

주제

- 커넥터의 IAM 역할
- 수신 이벤트에 대한 EventBridge 규칙

커넥터의 IAM 역할

커넥터와 연결하는 IAM 역할에는 이벤트를 EventBridge로 전송할 수 있는 PutEvents 권한이 있어야 합니다. 다음 IAM 정책 예제에서는 `example-event-bus`라는 이벤트 버스로 이벤트를 전송할 수 있는 권한을 부여합니다. 다음 예제의 리소스 ARN을 이벤트 버스의 ARN으로 바꿔야 합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/example-event-bus"
    }
  ]
```

```
}
```

또한 커넥터의 IAM 역할에 다음 신뢰 정책이 포함되어 있는지 확인해야 합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kafkaconnect.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

수신 이벤트에 대한 EventBridge 규칙

수신 이벤트를 이벤트 패턴이라는 이벤트 데이터 기준과 일치시키는 규칙을 생성합니다. 이벤트 패턴을 사용하면 수신 이벤트를 필터링하는 기준을 정의하고 특정 규칙을 트리거한 다음 지정된 대상으로 라우팅해야 하는 이벤트를 결정할 수 있습니다. 이벤트 패턴의 다음 예제는 EventBridge 이벤트 버스로 전송된 Kafka 이벤트와 일치합니다.

```
{
  "detail": {
    "topic": ["msk-eventbridge-tutorial"]
  }
}
```

다음은 Kafka 싱크 커넥터를 사용하여 Kafka에서 EventBridge로 전송된 이벤트의 예입니다.

```
{
  "version": "0",
  "id": "dbc1c73a-c51d-0c0e-ca61-ab9278974c57",
  "account": "123456789012",
  "time": "2025-03-26T10:15:00Z",
```

```
  "region": "us-east-1",
  "detail-type": "msk-eventbridge-tutorial",
  "source": "kafka-connect.msk-eventbridge-tutorial",
  "resources": [],
  "detail": {
    "topic": "msk-eventbridge-tutorial",
    "partition": 0,
    "offset": 0,
    "timestamp": 1742984100000,
    "timestampType": "CreateTime",
    "headers": [],
    "key": "order-1",
    "value": {
      "orderItems": [
        "item-1",
        "item-2"
      ],
      "orderCreatedTime": "Wed Mar 26 10:15:00 UTC 2025"
    }
  }
}
```

EventBridge 콘솔에서 이 예제 패턴을 사용하여 이벤트 버스에 규칙을 생성하고 CloudWatch Logs 그룹과 같은 대상을 지정합니다. EventBridge 콘솔은 CloudWatch Logs 그룹에 필요한 액세스 정책을 자동으로 구성합니다.

커넥터 생성

다음 섹션에서는 AWS Management 콘솔을 사용하여 EventBridge Kafka 싱크 커넥터를 생성하고 배포합니다.

주제

1단계: 커넥터 다운로드

EventBridge Kafka 커넥터의 GitHub 릴리스 페이지에서 최신 EventBridge 커넥터 싱크 JAR을 다운로드합니다. 예를 들어 버전 v1.4.1을 다운로드하려면 JAR 파일 링크인 `kafka-eventbridge-sink-with-dependencies.jar`를 선택하여 커넥터를 다운로드합니다. 그런 다음 파일을 시스템의 기본 위치에 저장합니다.

2단계: Amazon S3 버킷 생성

1. MSK Connect에서 사용할 수 있도록 Amazon S3에 JAR 파일을 저장하려면를 연 AWS Management 콘솔다음 Amazon S3를 선택합니다.
2. Amazon S3 콘솔에서 버킷 생성을 선택하고 고유한 버킷 이름을 입력합니다. 예를 들어 **`amzn-s3-demo-bucket1-eb-connector`**입니다.
3. Amazon S3 버킷에 적합한 리전을 선택합니다. MSK 클러스터가 배포된 리전과 일치하는지 확인합니다.
4. 버킷 설정의 경우 기본 선택을 유지하거나 필요에 따라 조정합니다.
5. 버킷 만들기를 선택합니다.
6. JAR 파일을 Amazon S3 버킷에 업로드합니다.

3단계: MSK Connect에서 플러그인 생성

1. 를 열고 MSK Connect로 AWS Management 콘솔이동합니다.
2. 왼쪽 탐색 창에서 사용자 지정 플러그인을 선택합니다.
3. 플러그인 생성을 선택한 후 플러그인 이름을 입력합니다. 예를 들어 **`eventbridge-sink-plugin`**입니다.
4. 사용자 지정 플러그인 위치에 S3 객체 URL을 붙여 넣습니다.
5. 선택 사항으로 플러그인에 대한 설명을 추가합니다.
6. 플러그인 생성을 선택합니다.

플러그인이 생성된 후 이를 사용하여 MSK Connect에서 EventBridge Kafka 커넥터를 구성하고 배포할 수 있습니다.

4단계: 커넥터 생성

커넥터를 생성하기 전에 커넥터 오류를 방지하기 위해 필요한 Kafka 토픽을 생성하는 것이 좋습니다. 토픽을 생성하려면 클라이언트 머신을 사용합니다.

1. MSK 콘솔의 왼쪽 창에서 커넥터를 선택한 다음 커넥터 생성을 선택합니다.
2. 플러그인 목록에서 eventbridge-sink-plugin을 선택한 후 다음을 선택합니다.
3. 커넥터 이름에 **EventBridgeSink**를 입력합니다.
4. 클러스터 목록에서 MSK 클러스터를 선택합니다.
5. 커넥터에 대한 다음 구성을 복사하여 커넥터 구성 필드에 붙여넣습니다.

   필요에 따라 다음 구성의 자리 표시자를 바꿉니다.

   - MSK 클러스터에 퍼블릭 인터넷 액세스 권한이 있는 경우 `aws.eventbridge.endpoint.uri`를 제거합니다.
   - PrivateLink를 사용하여 MSK에서 EventBridge로 안전하게 연결하는 경우 `https://` 뒤에 있는 DNS 부분을 이전에 생성한 EventBridge에 대한 (선택 사항) VPC 인터페이스 엔드포인트의 올바른 프라이빗 DNS 이름으로 바꿉니다.
   - 다음 구성의 EventBridge 이벤트 버스 ARN을 이벤트 버스의 ARN으로 바꿉니다.
   - 리전별 값을 업데이트합니다.

```
{
  "connector.class":
 "software.amazon.event.kafkaconnector.EventBridgeSinkConnector",
  "aws.eventbridge.connector.id": "msk-eventbridge-tutorial",
  "topics": "msk-eventbridge-tutorial",
  "tasks.max": "1",
  "aws.eventbridge.endpoint.uri": "https://events.us-east-1.amazonaws.com",
  "aws.eventbridge.eventbus.arn": "arn:aws:events:us-east-1:123456789012:event-bus/
example-event-bus",
  "value.converter.schemas.enable": "false",
  "value.converter": "org.apache.kafka.connect.json.JsonConverter",
  "aws.eventbridge.region": "us-east-1",
  "auto.offset.reset": "earliest",
  "key.converter": "org.apache.kafka.connect.storage.StringConverter"
}
```

   커넥터 구성에 대한 자세한 내용은 eventbridge-kafka-connector를 참조하세요.

   필요한 경우 작업자 및 자동 규모 조정에 대한 설정을 변경합니다. 또한 드롭다운에서 사용 가능한 최신(권장) Apache Kafka Connect 버전을 사용하는 것이 좋습니다. 액세스 권한에서 이전에 생성한 역할을 사용합니다. 또한 관찰성 및 문제 해결을 위해 CloudWatch에 대한 로깅을 활성화하는

것이 좋습니다. 필요에 따라 태그와 같은 기타 설정 옵션을 조정합니다. 그런 다음 커넥터를 배포하고 상태가 실행 중으로 변할 때까지 기다립니다.

Kafka로 메시지 전송

`value.converter` 및 선택적으로 Kafka Connect에서 사용할 수 있는 `key.converter` 설정을 통해 다른 변환기를 지정하여 Apache Avro 및 JSON과 같은 메시지 인코딩을 구성할 수 있습니다.

이 토픽의 connector example 섹션은 `value converter`용 `org.apache.kafka.connect.json.JsonConverter` 사용에서 알 수 있듯이 JSON 인코딩 메시지와 함께 작동하도록 구성됩니다. 커넥터가 실행 중 상태이면 클라이언트 시스템에서 `msk-eventbridge-tutorial` Kafka 토픽으로 레코드를 전송합니다.

구성 공급자가 있는 Debezium 소스 커넥터 사용

이 예제는 MySQL 호환 Amazon Aurora 데이터베이스를 소스로 하여 Debezium MySQL 커넥터 플러그인을 사용하는 방법을 보여줍니다. 이 예제에서는 AWS Secrets Manager에서 데이터베이스 보안 인증 정보를 외부화하기 위해 오픈 소스 AWS Secrets Manager 구성 공급자도 설정했습니다. 구성 공급자에 대해 자세히 알아보려면 튜토리얼: 구성 공급자를 사용하여 민감한 정보 외부화 섹션을 참조하세요.

> **Important**
>
> Debezium MySQL 커넥터 플러그인은 하나의 태스크만 지원하며 Amazon MSK Connect의 자동 규모 조정 용량 모드에서는 작동하지 않습니다. 대신 프로비저닝된 용량 모드를 사용하고 커넥터 구성에서 `workerCount`를 1로 설정해야 합니다. MSK Connect의 용량 모드에 대한 자세한 내용은 커넥터 용량 이해 섹션을 참조하세요.

Debezium 소스 커넥터를 사용하기 위한 사전 조건 완료

커넥터가 외부에 AWS Secrets Manager있는와 같은 서비스와 상호 작용할 수 있도록 인터넷에 액세스할 수 있어야 합니다Amazon Virtual Private Cloud. 이 섹션의 단계에 따라 다음 작업을 완료하여 인터넷 액세스를 활성화할 수 있습니다.

- NAT 게이트웨이를 호스팅하고 VPC의 인터넷 게이트웨이로 트래픽을 라우팅하는 퍼블릭 서브넷을 설정합니다.
- 프라이빗 서브넷 트래픽을 NAT 게이트웨이로 보내는 기본 경로를 생성합니다.

자세한 내용은 Amazon MSK Connect에 대한 인터넷 액세스 활성화 단원을 참조하십시오.

사전 조건

인터넷 액세스를 활성화하기 전에 다음 항목이 필요합니다.

- 클러스터와 연결된 Amazon Virtual Private Cloud(VPC)의 ID입니다. 예를 들어 vpc-123456ab입니다.
- VPC에 있는 프라이빗 서브넷 ID. 예를 들어 subnet-a1b2c3de, subnet-f4g5h6ij 등입니다. 커넥터를 프라이빗 서브넷으로 구성해야 합니다.

커넥터에 인터넷 액세스를 활성화하려면 다음을 수행합니다.

1. https://console.aws.amazon.com/vpc/ Amazon Virtual Private Cloud콘솔을 엽니다.
2. 설명이 포함된 이름으로 NAT 게이트웨이의 퍼블릭 서브넷을 생성하고 서브넷 ID를 기록합니다. 자세한 지침은 VPC에서 서브넷 생성을 참조하세요.
3. VPC가 인터넷과 통신할 수 있도록 인터넷 게이트웨이를 생성하고 게이트웨이 ID를 기록해 두세요. 인터넷 게이트웨이를 VPC에 연결합니다. 지침은 인터넷 게이트웨이 생성 및 연결을 참조하세요.
4. 프라이빗 서브넷의 호스트가 퍼블릭 서브넷에 연결될 수 있도록 퍼블릭 NAT 게이트웨이를 프로비저닝합니다. NAT 게이트웨이를 생성하는 경우 이전에 생성한 퍼블릭 서브넷을 선택합니다. 지침은 NAT 게이트웨이 생성을 참조하세요.
5. 라우팅 테이블을 구성합니다. 이 설정을 완료하려면 총 2개의 라우팅 테이블이 있어야 합니다. 이미 VPC와 동시에 자동으로 생성된 기본 라우팅 테이블이 있어야 합니다. 이 단계에서는 퍼블릭 서브넷에 대한 추가 라우팅 테이블을 생성합니다.

   a. 다음 설정을 사용하여 프라이빗 서브넷이 NAT 게이트웨이로 트래픽을 라우팅하도록 VPC의 기본 라우팅 테이블을 수정합니다. 지침은 Amazon Virtual Private Cloud 사용 설명서에서 라우팅 테이블로 작업을 참조하세요.

      프라이빗 MSKC 라우팅 테이블

속성	값
[Name tag]	이 라우팅 테이블을 식별하는 데 도움이 되도록 설명이 포함된 이름 태그 지정을 권장합니다. 프라이빗 MSKC를 예로 들 수 있습니다.

속성	값
관련 서브넷	프라이빗 서브넷
MSK Connect에 대한 인터넷 액세스를 활성화하는 경로	• 대상 주소: 0.0.0.0/0 • 대상: 사용자의 NAT 게이트웨이 ID. nat-12a345bc6789efg1h를 예로 들 수 있습니다.
내부 트래픽을 위한 로컬 경로	• 대상 주소: 10.0.0.0/16. 이 값은 VPC의 CIDR 블록에 따라 다를 수 있습니다. • 대상: 로컬

b. 사용자 지정 라우팅 테이블 생성의 지침에 따라 퍼블릭 서브넷에 대한 라우팅 테이블을 생성합니다. 테이블을 생성할 때 테이블이 연결된 서브넷을 식별할 수 있도록 이름 태그 필드에 설명이 포함된 이름을 입력합니다. 퍼블릭 MSKC를 예로 들 수 있습니다.

c. 다음 설정을 사용하여 퍼블릭 MSKC 라우팅 테이블을 구성합니다.

속성	값
[Name tag]	퍼블릭 MSKC 또는 사용자가 선택한 설명이 포함된 다른 이름
관련 서브넷	NAT 게이트웨이가 포함된 퍼블릭 서브넷
MSK Connect에 대한 인터넷 액세스를 활성화하는 경로	• 대상 주소: 0.0.0.0/0 • 대상: 사용자의 인터넷 게이트웨이 ID. igw-1a234bc5를 예로 들 수 있습니다.
내부 트래픽을 위한 로컬 경로	• 대상 주소: 10.0.0.0/16. 이 값은 VPC의 CIDR 블록에 따라 다를 수 있습니다. • 대상: 로컬

이제 Amazon MSK Connect에 대한 인터넷 액세스를 활성화했으므로 커넥터를 생성할 준비가 되었습니다.

Debezium 소스 커넥터 생성

이 절차에서는 Debezium 소스 커넥터를 생성하는 방법을 설명합니다.

1. 사용자 지정 플러그인 생성

   a. Debezium 사이트에서 안정적인 최신 릴리스를 위한 MySQL 커넥터 플러그인을 다운로드합니다. 다운로드한 Debezium 릴리스 버전(버전 2.x 또는 이전 시리즈 1.x)을 기록해 둡니다. 이 절차의 뒷부분에서 사용 중인 Debezium 버전에 따라 커넥터를 생성합니다.

   b. AWS Secrets Manager 구성 공급자를 다운로드하여 압축을 풉니다.

   c. 다음 아카이브를 같은 디렉터리에 배치합니다.

      - `debezium-connector-mysql` 폴더
      - `jcusten-border-kafka-config-provider-aws-0.1.1` 폴더

   d. 이전 단계에서 생성한 디렉터리를 ZIP 파일로 압축한 다음 해당 ZIP 파일을 S3 버킷에 업로드합니다. 지침은 Amazon S3 사용 설명서에서 객체 업로드를 참조하세요.

   e. 다음 JSON을 복사하여 파일에 붙여넣습니다. 예를 들어 `debezium-source-custom-plugin.json`입니다. *<example-custom-plugin-name>*을 플러그인에 사용할 이름으로 변경하고, *<amzn-s3-demo-bucket-arn>*을 ZIP 파일을 업로드한 Amazon S3 버킷의 ARN으로 변경하고, *<file-key-of-ZIP-object>*를 S3에 업로드한 ZIP 객체의 파일 키로 변경합니다.

```
{
    "name": "<example-custom-plugin-name>",
    "contentType": "ZIP",
    "location": {
        "s3Location": {
            "bucketArn": "<amzn-s3-demo-bucket-arn>",
            "fileKey": "<file-key-of-ZIP-object>"
        }
    }
}
```

   f. JSON 파일을 저장한 폴더에서 다음 AWS CLI명령을 실행하여 플러그인을 생성합니다.

```
aws kafkaconnect create-custom-plugin --cli-input-json file://<debezium-source-custom-plugin.json>
```

   다음 예제와 유사한 출력이 표시됩니다.

```
{
    "CustomPluginArn": "arn:aws:kafkaconnect:us-east-1:012345678901:custom-plugin/example-custom-plugin-name/abcd1234-a0b0-1234-c1-12345678abcd-1",
    "CustomPluginState": "CREATING",
    "Name": "example-custom-plugin-name",
    "Revision": 1
}
```

g. 다음 명령을 실행하여 플러그인 상태를 확인합니다. 상태가 `CREATING`에서 `ACTIVE`로 변경됩니다. ARN 자리 표시자를 이전 명령의 출력에서 가져온 ARN으로 변경합니다.

```
aws kafkaconnect describe-custom-plugin --custom-plugin-arn "<arn-of-your-custom-plugin>"
```

2. 데이터베이스 자격 증명에 대한 보안 암호 구성 AWS Secrets Manager및 생성

   a. https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.

   b. 데이터베이스 로그인 보안 인증 정보를 저장할 새로운 비밀번호를 생성합니다. 지침은 AWS Secrets Manager 사용 설명서에서 보안 암호 생성을 참조하세요.

   c. 보안 암호의 ARN을 복사합니다.

   d. 다음 예제 정책의 Secrets Manager 권한을 서비스 실행 역할 이해에 추가합니다. *<arn:aws:secretsmanager:us-east-1:123456789000:secret:MySecret-1234>*를 보안 암호의 ARN으로 변경합니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
      ],
      "Resource": [
```

```
            "arn:aws:secretsmanager:us-
east-1:123456789012:secret:MySecret-1234"
            ]
        }
    ]
}
```

IAM 권한을 추가하는 방법에 대한 지침은 IAM 사용 설명서에서 IAM ID 권한 추가 및 제거를 참조하세요.

3. 구성 제공자에 대한 정보를 사용하여 사용자 지정 작업자 구성 생성

   a. 다음 작업자 구성 속성을 파일에 복사하여 자리 표시자 문자열을 시나리오에 해당하는 값으로 변경합니다. AWS Secrets Manager 구성 공급자의 구성 속성에 대한 자세한 내용은 플러그인 설명서에서 SecretsManagerConfigProvider를 참조하세요.

```
key.converter=<org.apache.kafka.connect.storage.StringConverter>
value.converter=<org.apache.kafka.connect.storage.StringConverter>
config.providers.secretManager.class=com.github.jcustenborder.kafka.config.aws.SecretsM
config.providers=secretManager
config.providers.secretManager.param.aws.region=<us-east-1>
```

   b. 다음 AWS CLI명령을 실행하여 사용자 지정 작업자 구성을 생성합니다.

   다음 값을 교체합니다.

   - *<my-worker-config-name>* - 사용자 지정 작업자 구성을 설명하는 이름
   - *<encoded-properties-file-content-string>* - 이전 단계에서 복사한 일반 텍스트 속성의 base64 인코딩 버전

```
aws kafkaconnect create-worker-configuration --name <my-worker-config-name> --
properties-file-content <encoded-properties-file-content-string>
```

4. 커넥터 생성

   a. 사용 중인 Debezium 버전(2.x 또는 1.x)에 해당하는 다음 JSON을 복사하여 새 파일에 붙여넣습니다. *<placeholder>* 문자열을 시나리오에 해당하는 값으로 변경합니다. 서비스 실행 역할을 설정하는 방법에 대한 자세한 내용은 the section called “IAM 역할 및 정책” 섹션을 참조하세요.

이 구성에서는 데이터베이스 보안 인증 정보를 지정할 때 일반 텍스트 대신 `${secretManager:MySecret-1234:dbusername}`과 같은 변수를 사용한다는 점에 유의하세요. *MySecret-1234*를 보안 암호의 이름으로 변경하고 검색하려는 키의 이름을 입력합니다. 또한 *<arn-of-config-provider-worker-configuration>*을 사용자 지정 작업자 구성의 ARN으로 변경해야 합니다.

Debezium 2.x

Debezium 2.x 버전의 경우 다음 JSON을 복사하여 새로운 파일에 붙여넣습니다. *<placeholder>* 문자열을 시나리오에 해당하는 값으로 변경합니다.

```
{
 "connectorConfiguration": {
  "connector.class": "io.debezium.connector.mysql.MySqlConnector",
  "tasks.max": "1",
  "database.hostname": "<aurora-database-writer-instance-endpoint>",
  "database.port": "3306",
  "database.user": "<${secretManager:MySecret-1234:dbusername}>",
  "database.password": "<${secretManager:MySecret-1234:dbpassword}>",
  "database.server.id": "123456",
  "database.include.list": "<list-of-databases-hosted-by-specified-server>",
  "topic.prefix": "<logical-name-of-database-server>",
  "schema.history.internal.kafka.topic": "<kafka-topic-used-by-debezium-to-
track-schema-changes>",
  "schema.history.internal.kafka.bootstrap.servers": "<cluster-bootstrap-
servers-string>",
  "schema.history.internal.consumer.security.protocol": "SASL_SSL",
  "schema.history.internal.consumer.sasl.mechanism": "AWS_MSK_IAM",
  "schema.history.internal.consumer.sasl.jaas.config":
 "software.amazon.msk.auth.iam.IAMLoginModule required;",
  "schema.history.internal.consumer.sasl.client.callback.handler.class":
 "software.amazon.msk.auth.iam.IAMClientCallbackHandler",
  "schema.history.internal.producer.security.protocol": "SASL_SSL",
  "schema.history.internal.producer.sasl.mechanism": "AWS_MSK_IAM",
  "schema.history.internal.producer.sasl.jaas.config":
 "software.amazon.msk.auth.iam.IAMLoginModule required;",
  "schema.history.internal.producer.sasl.client.callback.handler.class":
 "software.amazon.msk.auth.iam.IAMClientCallbackHandler",
  "include.schema.changes": "true"
 },
 "connectorName": "example-Debezium-source-connector",
 "kafkaCluster": {
```

```
  "apacheKafkaCluster": {
   "bootstrapServers": "<cluster-bootstrap-servers-string>",
   "vpc": {
    "subnets": [
     "<cluster-subnet-1>",
     "<cluster-subnet-2>",
     "<cluster-subnet-3>"
    ],
    "securityGroups": ["<id-of-cluster-security-group>"]
   }
  }
 },
 "capacity": {
  "provisionedCapacity": {
   "mcuCount": 2,
   "workerCount": 1
  }
 },
 "kafkaConnectVersion": "2.7.1",
 "serviceExecutionRoleArn": "<arn-of-service-execution-role-that-msk-
connect-can-assume>",
 "plugins": [{
  "customPlugin": {
   "customPluginArn": "<arn-of-msk-connect-plugin-that-contains-connector-
code>",
   "revision": 1
  }
 }],
 "kafkaClusterEncryptionInTransit": {
  "encryptionType": "TLS"
 },
 "kafkaClusterClientAuthentication": {
  "authenticationType": "IAM"
 },
 "workerConfiguration": {
  "workerConfigurationArn": "<arn-of-config-provider-worker-configuration>",
  "revision": 1
 }
}
```

Debezium 1.x

Debezium 1.x 버전의 경우 다음 JSON을 복사하여 새로운 파일에 붙여넣습니다. *<placeholder>* 문자열을 시나리오에 해당하는 값으로 변경합니다.

```
{
 "connectorConfiguration": {
  "connector.class": "io.debezium.connector.mysql.MySqlConnector",
  "tasks.max": "1",
  "database.hostname": "<aurora-database-writer-instance-endpoint>",
  "database.port": "3306",
  "database.user": "<${secretManager:MySecret-1234:dbusername}>",
  "database.password": "<${secretManager:MySecret-1234:dbpassword}>",
  "database.server.id": "123456",
  "database.server.name": "<logical-name-of-database-server>",
  "database.include.list": "<list-of-databases-hosted-by-specified-server>",
  "database.history.kafka.topic": "<kafka-topic-used-by-debezium-to-track-
schema-changes>",
  "database.history.kafka.bootstrap.servers": "<cluster-bootstrap-servers-
string>",
  "database.history.consumer.security.protocol": "SASL_SSL",
  "database.history.consumer.sasl.mechanism": "AWS_MSK_IAM",
  "database.history.consumer.sasl.jaas.config":
 "software.amazon.msk.auth.iam.IAMLoginModule required;",
  "database.history.consumer.sasl.client.callback.handler.class":
 "software.amazon.msk.auth.iam.IAMClientCallbackHandler",
  "database.history.producer.security.protocol": "SASL_SSL",
  "database.history.producer.sasl.mechanism": "AWS_MSK_IAM",
  "database.history.producer.sasl.jaas.config":
 "software.amazon.msk.auth.iam.IAMLoginModule required;",
  "database.history.producer.sasl.client.callback.handler.class":
 "software.amazon.msk.auth.iam.IAMClientCallbackHandler",
  "include.schema.changes": "true"
 },
 "connectorName": "example-Debezium-source-connector",
 "kafkaCluster": {
  "apacheKafkaCluster": {
   "bootstrapServers": "<cluster-bootstrap-servers-string>",
   "vpc": {
    "subnets": [
     "<cluster-subnet-1>",
     "<cluster-subnet-2>",
     "<cluster-subnet-3>"
```

```
     ],
     "securityGroups": ["<id-of-cluster-security-group>"]
    }
   }
  },
  "capacity": {
   "provisionedCapacity": {
    "mcuCount": 2,
    "workerCount": 1
   }
  },
  "kafkaConnectVersion": "2.7.1",
  "serviceExecutionRoleArn": "<arn-of-service-execution-role-that-msk-
connect-can-assume>",
  "plugins": [{
   "customPlugin": {
    "customPluginArn": "<arn-of-msk-connect-plugin-that-contains-connector-
code>",
    "revision": 1
   }
  }],
  "kafkaClusterEncryptionInTransit": {
   "encryptionType": "TLS"
  },
  "kafkaClusterClientAuthentication": {
   "authenticationType": "IAM"
  },
  "workerConfiguration": {
   "workerConfigurationArn": "<arn-of-config-provider-worker-configuration>",
   "revision": 1
  }
}
```

b. 이전 단계에서 JSON 파일을 저장한 폴더에서 다음 AWS CLI명령을 실행합니다.

```
aws kafkaconnect create-connector --cli-input-json file://connector-info.json
```

다음은 명령을 성공적으로 실행했을 때 표시되는 출력의 예제입니다.

```
{
    "ConnectorArn": "arn:aws:kafkaconnect:us-east-1:123450006789:connector/
example-Debezium-source-connector/abc12345-abcd-4444-a8b9-123456f513ed-2",
    "ConnectorState": "CREATING",
```

```
    "ConnectorName": "example-Debezium-source-connector"
}
```

Debezium 커넥터 구성 업데이트

Debezium 커넥터의 구성을 업데이트하려면 다음 단계를 따르세요.

1. 다음 JSON을 복사하여 새 파일에 붙여넣습니다. `<placeholder>` 문자열을 시나리오에 해당하는 값으로 변경합니다.

   ```
   {
      "connectorArn": <connector_arn>,
      "connectorConfiguration": <new_configuration_in_json>,
      "currentVersion": <current_version>
   }
   ```

2. 이전 단계에서 JSON 파일을 저장한 폴더에서 다음 AWS CLI명령을 실행합니다.

   ```
   aws kafkaconnect update-connector --cli-input-json file://connector-info.json
   ```

   다음은 명령을 성공적으로 실행했을 때 표시되는 출력의 예제입니다.

   ```
   {
       "connectorArn": "arn:aws:kafkaconnect:us-east-1:123450006789:connector/example-Debezium-source-connector/abc12345-abcd-4444-a8b9-123456f513ed-2",
       "connectorOperationArn": "arn:aws:kafkaconnect:us-east-1:123450006789:connector-operation/example-Debezium-source-connector/abc12345-abcd-4444-a8b9-123456f513ed-2/41b6ad56-3184-479b-850a-a8bedd5a02f3",
       "connectorState": "UPDATING"
   }
   ```

3. 이제 다음 명령을 실행하여 작업의 현재 상태를 모니터링할 수 있습니다.

   ```
   aws kafkaconnect describe-connector-operation --connector-operation-arn
    <operation_arn>
   ```

자세한 단계가 포함된 Debezium 커넥터 예제는 Amazon MSK Connect 소개 - 관리형 커넥터를 사용하여 Apache Kafka 클러스터와 데이터 스트리밍을 참조하세요.

Amazon MSK Connect로 마이그레이션

이 섹션에서는 Apache Kafka 커넥터 애플리케이션을 Amazon Managed Streaming for Apache Kafka Connect(Amazon MSK Connect)로 마이그레이션하는 방법을 설명합니다. Amazon MSK Connect로 마이그레이션할 경우 얻게 되는 이점에 대한 자세한 내용은 ??? 단원을 참조하세요.

또한 이 섹션에서는 Kafka Connect 및 Amazon MSK Connect에서 사용하는 상태 관리 주제를 설명하고 소스 및 싱크 커넥터 마이그레이션의 절차를 다룹니다.

Kafka Connect에 사용되는 내부 주제 이해

분산 모드에서 실행되는 Apache Kafka Connect 애플리케이션은 Kafka 클러스터 및 그룹 멤버십의 내부 주제를 사용하여 상태를 저장합니다. 다음은 Kafka Connect 애플리케이션에 사용되는 내부 주제에 해당하는 구성 값입니다.

- 구성 주제, `config.storage.topic`을 통해 지정

  구성 주제에서 Kafka Connect는 사용자가 시작한 모든 커넥터 및 작업의 구성을 저장합니다. 사용자가 커넥터의 구성을 업데이트할 때마다 또는 커넥터가 재구성을 요청할 때(예: 커넥터가 더 많은 작업을 시작할 수 있음을 감지하는 경우) 레코드가 이 주제로 전송됩니다. 이 주제는 압축이 활성화되어 있으므로 항상 각 엔터티의 마지막 상태를 유지합니다.

- 오프셋 주제, `offset.storage.topic`을 통해 지정

  오프셋 주제에서 Kafka Connect는 소스 커넥터의 오프셋을 저장합니다. 구성 주제와 마찬가지로 오프셋 주제에서도 압축이 활성화됩니다. 이 주제는 외부 시스템에서 Kafka에 데이터를 생성하는 소스 커넥터에 대해서만 소스 위치를 작성하는 데 사용됩니다. Kafka에서 데이터를 읽고 외부 시스템으로 전송하는 싱크 커넥터는 일반 Kafka 소비자 그룹을 사용하여 소비자 오프셋을 저장합니다.

- 상태 주제, `status.storage.topic`을 통해 지정

  상태 주제에서 Kafka Connect는 커넥터 및 작업의 현재 상태를 저장합니다. 이 주제는 REST API 사용자가 쿼리하는 데이터의 중앙 위치로 사용됩니다. 이 주제에서는 사용자가 모든 작업자를 쿼리하고 실행 중인 모든 플러그인의 상태를 가져올 수 있습니다. 구성 및 오프셋 주제와 마찬가지로 상태 주제에서도 압축이 활성화됩니다.

이러한 주제 외에도 Kafka Connect는 Kafka의 그룹 멤버십 API를 광범위하게 사용합니다. 그룹 이름은 커넥터 이름에 따라 지정됩니다. 예를 들어 file-sink라는 커넥터의 경우 그룹 이름은 connect-file-sink입니다. 그룹의 각 소비자는 단일 작업에 레코드를 제공합니다. 이러한 그룹과 해당 오프셋은

`Kafka-consumer-group.sh`와 같은 일반 소비자 그룹 도구를 사용하여 검색할 수 있습니다. 각 싱크 커넥터에 대해 Connect 런타임은 Kafka에서 레코드를 추출하는 일반 소비자 그룹을 실행합니다.

Amazon MSK Connect 애플리케이션의 상태 관리

기본적으로 Amazon MSK Connect는 각 Amazon MSK 커넥터에 대해 Kafka 클러스터에 세 가지 별도의 주제를 생성하여 커넥터의 구성, 오프셋 및 상태를 저장합니다. 기본 주제 이름은 다음과 같이 구성됩니다.

- __msk_connect_configs_*connector-name*_*connector-id*
- __msk_connect_status_*connector-name*_*connector-id*
- __msk_connect_offsets_*connector-name*_*connector-id*

> **Note**
>
> 소스 커넥터 간에 오프셋 연속성을 제공하려면 기본 주제 대신 원하는 오프셋 스토리지 주제를 사용하면 됩니다. 오프셋 스토리지 주제를 지정하면 이전 커넥터의 마지막 오프셋에서 읽기를 다시 시작하는 소스 커넥터를 생성하는 것과 같은 작업을 수행하는 데 도움이 됩니다. 오프셋 스토리지 주제를 지정하려면 커넥터를 생성하기 전에 Amazon MSK Connect 작업자 구성에서 offset.storage.topic 속성 값을 제공해야 합니다.

소스 커넥터를 Amazon MSK Connect로 마이그레이션

소스 커넥터는 외부 시스템에서 Kafka로 레코드를 가져오는 Apache Kafka Connect 애플리케이션입니다. 이 섹션에서는 온프레미스로 또는 AWS 기반의 자체 관리형 Kafka Connect 클러스터에서 실행 중인 Apache Kafka Connect 소스 커넥터 애플리케이션을 Amazon MSK Connect로 마이그레이션하는 프로세스를 설명합니다.

Kafka Connect 소스 커넥터 애플리케이션은 구성 속성 `offset.storage.topic`에 대해 설정된 값으로 이름이 지정된 주제에 오프셋을 저장합니다. 다음은 `movies` 및 `shows`라는 두 개의 테이블에서 데이터를 가져오는 두 개의 작업을 실행하는 JDBC 커넥터에 대한 샘플 오프셋 메시지입니다. 테이블 영화에서 가져온 가장 최근 행의 기본 ID는 18343입니다. 쇼 테이블에서 가져온 가장 최근 행의 기본 ID는 732입니다.

```
["jdbcsource",{"protocol":"1","table":"sample.movies"}] {"incrementing":18343}
["jdbcsource",{"protocol":"1","table":"sample.shows"}] {"incrementing":732}
```

소스 커넥터를 Amazon MSK Connect로 마이그레이션하려면 다음을 수행하세요.

1. 온프레미스 또는 자체 관리형 Kafka Connect 클러스터에서 커넥터 라이브러리를 가져와 Amazon MSK Connect 사용자 지정 플러그인을 생성합니다.
2. Amazon MSK Connect 작업자 속성을 생성하고 `value.converter`, `key.converter` 및 `offset.storage.topic` 속성을 기존 Kafka Connect 클러스터에서 실행 중인 Kafka 커넥터에 대해 설정된 것과 동일한 값으로 설정합니다.
3. 기존 Kafka Connect 클러스터에서 `PUT /connectors/connector-name/pause` 요청을 하여 기존 클러스터의 커넥터 애플리케이션을 일시 중지합니다.
4. 커넥터 애플리케이션의 모든 작업이 완전히 중지되었는지 확인합니다. 기존 Kafka Connect 클러스터에서 `GET /connectors/connector-name/status` 요청을 하거나 `status.storage.topic` 속성에 대해 설정된 주제 이름의 메시지를 사용하여 작업을 중지할 수 있습니다.
5. 기존 클러스터에서 커넥터 구성을 가져옵니다. 기존 클러스터에서 `GET /connectors/connector-name/config/` 요청을 하거나 `config.storage.topic` 속성에 대해 설정된 주제 이름의 메시지를 사용하여 커넥터 구성을 가져올 수 있습니다.
6. 기존 클러스터와 이름이 동일한 새 Amazon MSK 커넥터를 생성합니다. 1단계에서 생성한 커넥터 사용자 지정 플러그인, 2단계에서 생성한 작업자 속성, 5단계에서 추출한 커넥터 구성을 사용하여 이 커넥터를 생성합니다.
7. Amazon MSK 커넥터 상태가 `active`이면 로그를 보고, 커넥터가 소스 시스템에서 데이터 가져오기를 시작했는지 확인합니다.
8. `DELETE /connectors/connector-name` 요청을 하여 기존 클러스터의 커넥터를 삭제합니다.

싱크 커넥터를 Amazon MSK Connect로 마이그레이션

싱크 커넥터는 Kafka에서 외부 시스템으로 데이터를 내보내는 Apache Kafka Connect 애플리케이션입니다. 이 섹션에서는 온프레미스로 또는 AWS 기반의 자체 관리형 Kafka Connect 클러스터에서 실행 중인 Apache Kafka Connect 싱크 커넥터 애플리케이션을 Amazon MSK Connect로 마이그레이션하는 프로세스를 설명합니다.

Kafka Connect 싱크 커넥터는 Kafka 그룹 멤버십 API를 사용하고 일반적인 소비자 애플리케이션과 동일한 `__consumer_offset` 주제에 오프셋을 저장합니다. 이 동작으로 자체 관리형 클러스터에서 Amazon MSK Connect로 싱크 커넥터를 마이그레이션하는 작업이 간소화됩니다.

싱크 커넥터를 Amazon MSK Connect로 마이그레이션하려면 다음을 수행하세요.

1. 온프레미스 또는 자체 관리형 Kafka Connect 클러스터에서 커넥터 라이브러리를 가져와 Amazon MSK Connect 사용자 지정 플러그인을 생성합니다.
2. Amazon MSK Connect 작업자 속성을 생성하고 기존 Kafka Connect 클러스터에서 실행 중인 Kafka 커넥터에 대해 설정된 것과 동일한 값으로 `key.converter` 및 `value.converter` 속성을 설정합니다.
3. 기존 Kafka Connect 클러스터에서 `PUT /connectors/connector-name/pause` 요청을 하여 기존 클러스터의 커넥터 애플리케이션을 일시 중지합니다.
4. 커넥터 애플리케이션의 모든 작업이 완전히 중지되었는지 확인합니다. 기존 Kafka Connect 클러스터에서 `GET /connectors/connector-name/status` 요청을 하거나 `status.storage.topic` 속성에 대해 설정된 주제 이름의 메시지를 사용하여 작업을 중지할 수 있습니다.
5. 기존 클러스터에서 커넥터 구성을 가져옵니다. 기존 클러스터에서 `GET /connectors/connector-name/config` 요청을 하거나 `config.storage.topic` 속성에 대해 설정된 주제 이름의 메시지를 사용하여 커넥터 구성을 가져올 수 있습니다.
6. 기존 클러스터와 이름이 동일한 새 Amazon MSK 커넥터를 생성합니다. 1단계에서 생성한 커넥터 사용자 지정 플러그인, 2단계에서 생성한 작업자 속성, 5단계에서 추출한 커넥터 구성을 사용하여 이 커넥터를 생성합니다.
7. Amazon MSK 커넥터 상태가 `active`이면 로그를 보고, 커넥터가 소스 시스템에서 데이터 가져오기를 시작했는지 확인합니다.
8. `DELETE /connectors/connector-name` 요청을 하여 기존 클러스터의 커넥터를 삭제합니다.

Amazon MSK Connect의 문제 해결

다음 정보는 MSK Connect를 사용하는 동안 발생할 수 있는 문제를 해결하는 데 도움이 될 수 있습니다. AWS re:Post에 문제를 게시할 수도 있습니다.

커넥터가 퍼블릭 인터넷에서 호스팅되는 리소스에 액세스할 수 없습니다.

Amazon MSK Connect를 위한 인터넷 액세스 활성화를 참조하세요.

커넥터에서 실행 중인 작업 수가 tasks.max에 지정된 작업 수와 동일하지 않습니다.

다음은 커넥터가 지정된 최대 작업 수보다 적은 수의 작업을 사용하는 몇 가지 이유입니다.

- 일부 커넥터 구현은 사용할 수 있는 작업 수를 제한합니다. 예를 들어 MySQL을 위한 Debezium 커넥터는 단일 태스크 사용으로 제한됩니다.

- 자동 규모 조정 용량 모드를 사용하는 경우 Amazon MSK Connect는 커넥터에서 실행 중인 작업자 수와 작업자당 MCU 수에 비례하는 값으로 커넥터의 tasks.max 속성을 재정의합니다.
- 싱크 커넥터의 경우 병렬 처리 수준(작업 수)은 주제 파티션 수를 초과할 수 없습니다. 작업 최대값을 이보다 크게 설정할 수는 있지만, 단일 파티션은 한 번에 두 개 이상의 작업으로 처리되지 않습니다.
- Kafka Connect 2.7.x에서 기본 소비자 파티션 할당자는 `RangeAssignor`입니다. 이 할당자의 동작은 모든 주제의 첫 번째 파티션을 단일 소비자에게, 모든 주제의 두 번째 파티션을 단일 소비자에게 제공하는 등의 작업을 수행합니다. 즉, `RangeAssignor`를 사용하는 싱크 커넥터의 최대 활성 작업 수는 사용 중인 단일 주제의 최대 파티션 수와 동일합니다. 이 방법이 사용 사례에 적합하지 않은 경우 `consumer.partition.assignment.strategy` 속성을 더 적합한 소비자 파티션 할당자로 설정하는 작업자 구성을 생성해야 합니다. Kafka 2.7 인터페이스 ConsumerPartitionAssignor: 알려진 모든 구현 클래스를 참조하세요.

Amazon MSK Replicator란 무엇인가요?

Amazon MSK Replicator는 서로 다른 리전 또는 동일한 AWS 리전의 Amazon MSK 클러스터 전반에서 데이터를 안정적으로 복제할 수 있게 해주는 Amazon MSK의 기능입니다. 단, 소스 및 대상 클러스터가 모두 동일한 AWS 계정에 있어야 합니다. MSK Replicator를 사용하면 가용성과 비즈니스 연속성을 높이기 위해 지역적으로 탄력적인 스트리밍 애플리케이션을 쉽게 구축할 수 있습니다. MSK Replicator는 MSK 클러스터 전반에 걸쳐 자동 비동기 복제 기능을 제공하므로 사용자 지정 코드를 작성하거나 인프라를 관리하거나 리전 간 네트워킹을 설정할 필요가 없습니다.

MSK Replicator는 기본 리소스를 자동으로 확장하므로 용량을 모니터링하거나 확장할 필요 없이 온디맨드 방식으로 데이터를 복제할 수 있습니다. 또한 MSK Replicator는 주제 구성, 액세스 제어 목록(ACL), 소비자 그룹 오프셋 등 필요한 Kafka 메타데이터를 복제합니다. 리전에서 예기치 않은 이벤트가 발생하면 다른 AWS리전으로 장애 조치하고 처리를 원활하게 재개할 수 있습니다.

MSK Replicator는 리전 간 복제(CRR)와 동일 리전 복제(SRR)를 모두 지원합니다. 교차 리전 복제에서 소스 및 대상 MSK 클러스터는 서로 다른 AWS리전에 있습니다. 동일 리전 복제에서는 소스 MSK 클러스터와 대상 MSK 클러스터가 모두 동일한 AWS리전에 있습니다. 소스 및 대상 MSK 클러스터를 생성한 후 MSK Replicator에서 사용해야 합니다.

> **Note**
>
> MSK Replicator는 다음 AWS리전을 지원합니다. 미국 동부(us-east-1, 버지니아 북부) 미국 동부(us-east-2, 오하이오) 미국 서부(us-west-2, 오레곤) 유럽(eu-west-1, 아일랜드) 유럽(eu-central-1, 프랑크푸르트) 아시아 태평양(ap-southeast-1, 싱가포르) 아시아 태평양(ap-southeast-2, 시드니) 유럽(eu-north-1, 스톡홀름) 아시아 태평양(ap-south-1, 뭄바이) 유럽(eu-west-3, 파리) 남아메리카(sa-east-1, 상파울루) 아시아 태평양(ap-northeast-2, 서울) 유럽(eu-west-2, 런던) 아시아 태평양(ap-northeast-1, 도쿄) 미국 서부(us-west-1, 캘리포니아 북부) 캐나다(ca-central-1, 중앙) 중국(베이징)(cn-north-1), 중국(닝샤)(cn-northwest-1) 아시아 태평양(하이데라바드)(ap-south-2), 아시아 태평양(말레이시아)(ap-southeast-5), 아시아 태평양(태국)(ap-southeast-7), 멕시코(중부)(mx-central-1), 아시아 태평양(타이베이)(ap-east-2), 캐나다 서부(캘거리)(ca-west-1), 유럽(스페인)(eu-south-2), 중동(바레인)(me-south-1), 아시아 태평양(자카르타)(ap-southeast-3), 아프리카(케이프타운)(af-south-1), 중동(UAE)(me-central-1), 아시아 태평양(홍콩)(ap-east-1), 아시아 태평양(오사카)(ap-northeast-3), 아시아 태평양(멜버른)(ap-southeast-4), 유럽(밀라노)(eu-south-1), 이스라엘(텔아비브)(il-central-1) 및 유럽(취리히)(eu-central-2).

다음은 Amazon MSK Replicator의 몇 가지 일반적인 용도입니다.

- 다중 리전 스트리밍 애플리케이션 빌드: 사용자 지정 솔루션을 설정하지 않고도 복원력을 높이기 위해 가용성이 높고 내결함성이 뛰어난 스트리밍 애플리케이션을 빌드합니다.
- 데이터 액세스 지연 시간 단축: 다양한 지리적 리전의 소비자에게 지연 시간이 더 짧은 데이터 액세스를 제공합니다.
- 파트너에게 데이터 배포: 하나의 Apache Kafka 클러스터에서 여러 Apache Kafka 클러스터로 데이터를 복사하여 여러 팀/파트너가 각자의 데이터 복사본을 가질 수 있도록 합니다.
- 분석을 위한 데이터 집계: 여러 Apache Kafka 클러스터의 데이터를 하나의 클러스터로 복사하여 집계된 실시간 데이터에 대한 인사이트를 간편하게 생성할 수 있습니다.
- 로컬에서 쓰기, 전역적으로 데이터에 액세스: 한 AWS리전에서 수행된 쓰기를 다른 리전으로 자동 전파하도록 다중 활성 복제를 설정하여 더 짧은 지연 시간과 비용으로 데이터를 제공합니다.

Amazon MSK Replicator 작동 방식

MSK Replicator를 시작하려면 대상 클러스터의 AWS리전에 새 Replicator를 생성해야 합니다. MSK Replicator는 소스라는 기본 AWS리전의 클러스터에서 대상이라는 대상 리전의 클러스터로 모든 데이터를 자동으로 복사합니다. 소스 클러스터와 대상 클러스터는 동일하거나 다른 AWS리전에 있을 수 있습니다. 대상 클러스터가 아직 존재하지 않는 경우 이를 생성해야 합니다.

복제기를 생성하면 MSK Replicator는 대상 클러스터의 AWS리전에 필요한 모든 리소스를 배포하여 데이터 복제 지연 시간을 최적화합니다. 복제 지연 시간은 MSK 클러스터의 AWS리전 간 네트워크 거리, 소스 및 대상 클러스터의 처리량 용량, 소스 및 대상 클러스터의 파티션 수 등 여러 요인에 따라 달라집니다. MSK Replicator는 기본 리소스를 자동으로 확장하므로 용량을 모니터링하거나 확장할 필요 없이 온디맨드 방식으로 데이터를 복제할 수 있습니다.

데이터 복제

기본적으로 MSK Replicator는 소스 클러스터 주제 파티션의 최신 오프셋에서 모든 데이터를 대상 클러스터로 비동기적으로 복사합니다. '새 주제 감지 및 복사' 설정이 켜져 있으면 MSK Replicator에서 새로운 주제 또는 주제 파티션을 자동으로 감지하여 대상 클러스터에 복사합니다. 하지만 Replicator가 대상 클러스터에서 새로운 주제 또는 주제 파티션을 감지하고 생성하는 데 최대 30초가 걸릴 수 있습니다. 대상 클러스터에서 주제가 생성되기 전까지는 소스 주제에 생성된 메시지는 복제되지 않습니다. 또는 주제의 기존 메시지를 대상 클러스터에 복제하려는 경우 소스 클러스터 주제 파티션의 가장 빠른 오프셋에서 복제를 시작하도록 생성 중 Replicator를 구성할 수 있습니다.

MSK Replicator는 데이터를 저장하지 않습니다. 데이터는 소스 클러스터에서 소비되고, 인메모리로 버퍼링되고, 대상 클러스터에 작성됩니다. 버퍼는 데이터가 성공적으로 작성되거나 재시도 후 실패할 때 자동으로 지워집니다. MSK Replicator와 클러스터 간의 모든 통신 및 데이터는 항상 전송 중에 암호화됩니다. `DescribeClusterV2`, `CreateTopic`와 같은 모든 MSK Replicator API 호출`DescribeTopicDynamicConfiguration`은AWSCloudTrail에서 캡처됩니다. MSK 브로커 로그도 동일하게 반영합니다.

MSK Replicator는 Replicator Factor가 3인 대상 클러스터에 주제를 생성합니다. 필요한 경우 대상 클러스터에서 직접 복제 인수를 수정할 수 있습니다.

메타데이터 복제

MSK Replicator는 소스 클러스터에서 대상 클러스터로의 메타데이터 복사도 지원합니다. 메타데이터에는 주제 구성, 액세스 제어 목록(ACL), 소비자 그룹 오프셋이 포함되어 있습니다. 데이터 복제와 마찬가지로 메타데이터 복제도 비동기적으로 이루어집니다. 성능을 높이기 위해 MSK Replicator는 메타데이터 복제보다 데이터 복제에 우선순위를 둡니다.

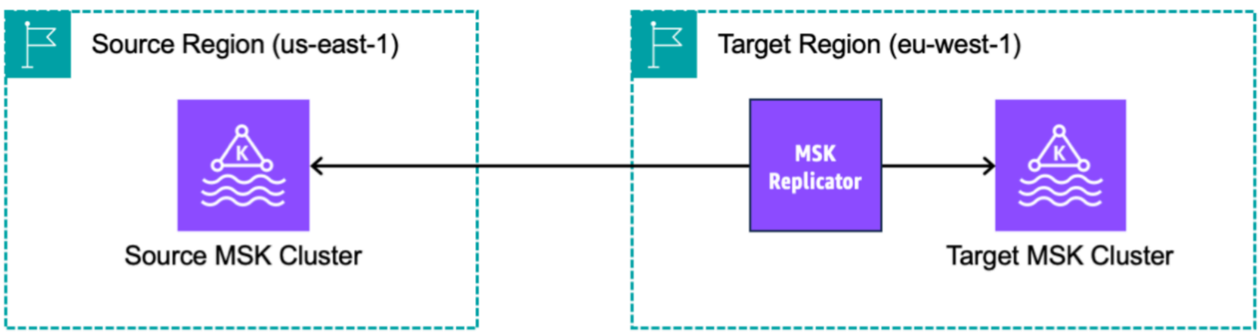
다음 표는 MSK Replicator에서 복사하는 액세스 제어 목록(ACL) 목록입니다.

연산	연구	허용되는 API
Alter	Topic	CreatePartitions
AlterConfigs	Topic	AlterConfigs
생성	Topic	CreateTopics, Metadata
Delete	Topic	DeleteRecords, DeleteTopics
설명	Topic	ListOffsets, Metadata, OffsetFetch, OffsetFor LeaderEpoch
DescribeConfigs	Topic	DescribeConfigs
읽기	Topic	Fetch, OffsetCommit, TxnOffsetCommit
쓰기(거부만 해당)	Topic	Produce, AddPartitionsToTxn

MSK Replicator는 리소스 유형 주제에 대해서만 LITERAL 패턴 유형 ACL을 복사합니다. PREFIXED 패턴 유형 ACL 및 기타 리소스 유형 ACL은 복사되지 않습니다. 또한 MSK Replicator는 대상 클러스터에서 ACL을 삭제하지 않습니다. 소스 클러스터에서 ACL을 삭제하는 경우 대상 클러스터에서도 동시에 삭제해야 합니다. Kafka ACL 리소스, 패턴 및 작업에 대한 자세한 내용은 https://kafka.apache.org/documentation/#security_authz_cli를 참조하세요.

MSK Replicator는 IAM 액세스 제어에서 사용하지 않는 Kafka ACL만 복제합니다. 클라이언트가 IAM 액세스 제어를 사용하여 MSK 클러스터를 읽고 쓰는 경우 원활한 장애 조치를 위해 대상 클러스터에서 관련 IAM 정책을 구성해야 합니다. 이는 접두사 및 동일한 주제 이름 복제 구성 모두에 대해서도 마찬가지입니다.

소비자 그룹 오프셋 동기화의 일환으로 MSK Replicator는 스트림의 팁(주제 파티션의 끝)에 가까운 위치에서 읽고 있는 소스 클러스터의 소비자를 위해 최적화합니다. 소비자 그룹이 소스 클러스터에서 지연되는 경우 소스에 비해 대상의 해당 소비자 그룹 지연이 더 높을 수 있습니다. 즉, 대상 클러스터로의 장애 조치 후 소비자는 더 많은 중복 메시지를 재처리합니다. 이 지연을 줄이려면 소스 클러스터의 소비자가 스트림의 팁(주제 파티션의 끝)을 파악하여 이 팁에서 소비를 시작해야 합니다. 소비자가 소비를 시작하면 MSK Replicator가 지연을 자동으로 줄입니다.



주제 이름 구성

MSK Replicator에는 접두사(기본값) 또는 동일한 주제 이름 복제라는 두 가지 주제 이름 구성 모드가 있습니다.

접두사 주제 이름 복제

기본적으로 MSK Replicator는 소스 클러스터 주제 이름에 자동 생성된 접두사(예: `<sourceKafkaClusterAlias>.topic`)를 추가하여 대상 클러스터에 새로운 주제를 생성합니다.

이는 대상 클러스터의 복제된 주제를 다른 주제와 구분하고 클러스터 간에 데이터가 순환 복제되는 것을 방지하기 위한 것입니다.

예를 들어 MSK Replicator는 소스 클러스터에서 'topic' 이름이 들어간 주제의 데이터를 <sourceKafkaClusterAlias >.topic이라는 대상 클러스터의 새로운 주제로 복제합니다. `DescribeReplicator` API 또는 MSK 콘솔의 Replicator 세부 정보 페이지를 사용하여 sourceKafkaClusterAlias 필드 아래에서 대상 클러스터의 주제 이름에 추가되는 접두사를 찾을 수 있습니다. 대상 클러스터의 접두사는 <sourceKafkaClusterAlias>입니다.

소비자가 대기 클러스터에서 처리를 안정적으로 다시 시작할 수 있도록 하려면 와일드카드 연산자 `.*`를 사용하여 주제의 데이터를 읽도록 소비자를 구성해야 합니다. 예를 들어 소비자는 두 AWS리전 모두에서 `*topic1`를 사용하여를 사용해야 합니다. 이 예에는 `footopic1`과 같은 주제도 포함되므로 필요에 따라 와일드카드 연산자를 조정하세요.

활성-활성 클러스터 설정과 같이 대상 클러스터의 별도의 주제에 Replicator 데이터를 유지하려는 경우 접두사를 추가하는 MSK Replicator를 사용해야 합니다.

동일한 주제 이름 복제

기본 설정의 대안으로 Amazon MSK Replicator를 사용하면 주제 복제가 동일한 주제 이름 복제로 설정된 Replicator를 생성할 수 있습니다(콘솔에 동일한 주제 이름 유지). 대상 MSK 클러스터가 있는 AWS 리전에서 새 Replicator를 생성할 수 있습니다. 이름이 동일한 복제된 주제를 사용하면 복제된 주제를 읽도록 클라이언트를 재구성하지 않아도 됩니다.

동일한 주제 이름을 복제(콘솔에서 동일한 주제 이름 유지)하면 다음과 같은 이점이 있습니다.

- 복제 프로세스 중에 동일한 주제 이름을 유지하면서 무한 복제 반복의 위험을 자동으로 방지할 수 있습니다.
- 복제된 주제에서 읽도록 클라이언트를 재구성하지 않아도 되므로 다중 클러스터 스트리밍 아키텍처를 더 간단하게 설정하고 운영할 수 있습니다.
- 액티브-패시브 클러스터 아키텍처의 경우 동일한 주제 이름 복제 기능은 장애 조치 프로세스도 간소화하므로 주제 이름 변경이나 클라이언트 재구성 없이 애플리케이션이 대기 클러스터로의 장애 조치를 원활하게 수행할 수 있습니다.
- 데이터 집계 또는 중앙 집중식 분석을 위해 여러 MSK 클러스터의 데이터를 단일 클러스터로 더 쉽게 통합하는 데 사용할 수 있습니다. 이렇게 하려면 각 소스 클러스터와 동일한 대상 클러스터에 대해 별도의 Replicator를 생성해야 합니다.
- 대상 클러스터에서 동일한 이름의 주제로 데이터를 복제하여 한 MSK 클러스터에서 다른 MSK 클러스터로의 데이터 마이그레이션을 간소화할 수 있습니다.

Amazon MSK Replicator는 Kafka 헤더를 사용하여 데이터가 시작된 주제로 다시 복제되는 것을 자동으로 방지하므로 복제 중에 무한 주기의 위험이 없습니다. 헤더는 각 Kafka 메시지에 키, 값 및 타임스탬프와 함께 포함될 수 있는 키-값 페어입니다. MSK Replicator는 복제되는 각 레코드의 헤더에 소스 클러스터 및 주제에 대한 식별자를 포함합니다. MSK Replicator는 헤더 정보를 사용하여 무한 복제 반복을 방지합니다. 클라이언트가 복제된 데이터를 예상대로 읽을 수 있는지 확인해야 합니다.

튜토리얼: Amazon MSK Replicator에 대한 소스 및 대상 클러스터 설정

이 자습서에서는 동일한 AWS리전 또는 다른 AWS리전에서 소스 클러스터와 대상 클러스터를 설정하는 방법을 보여줍니다. 그런 다음 해당 클러스터를 사용하여 Amazon MSK Replicator를 생성합니다.

Amazon MSK 소스 클러스터 준비

MSK Replicator를 위해 이미 생성된 MSK 소스 클러스터가 있는 경우 이 섹션에 설명된 요구 사항을 충족하는지 확인하세요. 그렇지 않으면 다음 단계에 따라 MSK 프로비저닝 또는 서버리스 소스 클러스터를 생성합니다.

리전 간 및 동일 리전의 MSK Replicator 소스 클러스터를 생성하는 프로세스도 유사합니다. 다음 절차에서 차이점을 설명합니다.

1. 소스 리전에서 IAM 액세스 제어가 켜져 있는 MSK 프로비저닝 클러스터 또는 서버리스 클러스터를 생성합니다. 소스 클러스터에는 최소 3개의 브로커가 있어야 합니다.
2. 리전 간 MSK Replicator의 경우 소스가 프로비저닝 클러스터일 때 IAM 액세스 제어 체계에 대해 다중 VPC 프라이빗 연결을 사용하도록 설정하여 구성합니다. 다중 VPC가 활성화된 경우에는 인증되지 않은 인증 유형이 지원되지 않는다는 점에 유의하세요. 다른 인증 체계(mTLS 또는 SASL/SCRAM)를 위해 다중 VPC 프라이빗 연결을 설정할 필요는 없습니다. MSK 클러스터에 연결하는 다른 클라이언트에 대해 mTLS 또는 SASL/SCRAM 인증 체계를 동시에 사용할 수 있습니다. 콘솔 클러스터 세부 정보 네트워크 설정에서 또는 `UpdateConnectivity` API를 사용하여 다중 VPC 프라이빗 연결을 구성할 수 있습니다. 클러스터 소유자가 다중 VPC 활성화를 참조하세요. 소스 클러스터가 MSK 서버리스 클러스터인 경우 다중 VPC 프라이빗 연결을 켤 필요는 없습니다.

   동일 지역 MSK Replicator의 경우 MSK 소스 클러스터는 다중 VPC 프라이빗 연결이 필요하지 않으며 다른 클라이언트가 인증되지 않은 인증 유형을 사용하여 클러스터에 계속 액세스할 수 있습니다.
3. 리전 간 MSK Replicator의 경우 리소스 기반 권한 정책을 소스 클러스터에 연결해야 합니다. 이렇게 하면 MSK가 해당 클러스터에 연결하여 데이터를 복제할 수 있습니다. 아래 CLI 또는 AWS콘솔 절

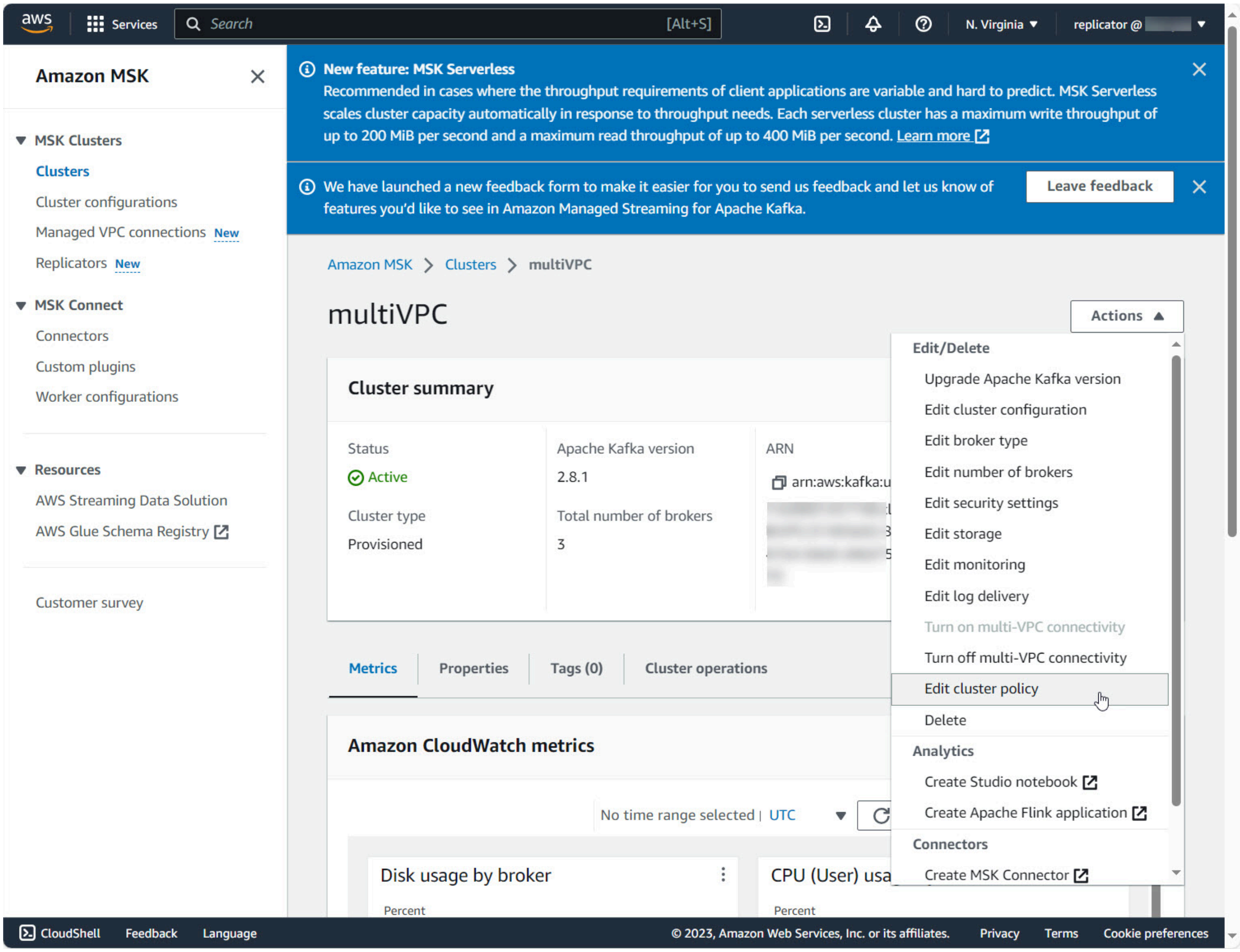
차를 사용하여이 작업을 수행할 수 있습니다. Amazon MSK 리소스 기반 정책도 참조하세요. 동일 리전의 MSK Replicator의 경우 이 단계를 수행할 필요가 없습니다.

Console: create resource policy

다음 JSON으로 소스 클러스터 정책을 업데이트합니다. 자리 표시자를 소스 클러스터의 ARN으로 변경합니다.

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": [
                "kafka.amazonaws.com"
            ]
        },
        "Action": [
            "kafka:CreateVpcConnection",
            "kafka:GetBootstrapBrokers",
            "kafka:DescribeClusterV2"
        ],
        "Resource": "arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-5678-90ab-cdef-1234567890ab-1"
    }
  ]
}
```

클러스터 세부 정보 페이지의 작업 메뉴에서 클러스터 정책 편집 옵션을 사용합니다.



CLI: create resource policy

참고: AWS콘솔을 사용하여 소스 클러스터를 생성하고 새 IAM 역할을 생성하는 옵션을 선택하면 가 필요한 신뢰 정책을 역할에 AWS연결합니다. MSK가 기존 IAM 역할을 사용하도록 하거나 직접 역할을 생성하는 경우 다음 신뢰 정책을 해당 역할에 연결하여 MSK Replicator가 해당 역할을 맡을 수 있도록 하세요. 역할의 신뢰 관계를 수정하는 방법에 대한 자세한 내용은 역할 수정을 참조하세요.

1. 이 명령을 사용하여 MSK 클러스터 정책의 현재 버전을 가져옵니다. 자리 표시자를 실제 클러스터 ARN으로 변경합니다.

```
aws kafka get-cluster-policy —cluster-arn <Cluster ARN>
{
"CurrentVersion": "K1PA6795UKM GR7",
"Policy": "..."
```

```
}
```

2. 리소스 기반 정책을 생성하여 MSK Replicator가 소스 클러스터에 액세스할 수 있도록 허용합니다. 다음 구문을 템플릿으로 사용하여 자리 표시자를 실제 소스 클러스터 ARN으로 변경합니다.

```
aws kafka put-cluster-policy --cluster-arn "<sourceClusterARN>" --policy '{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Principal": {
"Service": [
"kafka.amazonaws.com"
]
},
"Action": [
"kafka:CreateVpcConnection",
"kafka:GetBootstrapBrokers",
"kafka:DescribeClusterV2"
],
"Resource": "<sourceClusterARN>"
}
]
```

Amazon MSK 대상 클러스터 준비

IAM 액세스 제어가 켜져 있는 MSK 대상 클러스터(프로비저닝 또는 서버리스)를 생성합니다. 대상 클러스터는 다중 VPC 프라이빗 연결이 켜져 있지 않아도 됩니다. 대상 클러스터는 소스 클러스터와 동일한 AWS리전 또는 다른 리전에 있을 수 있습니다. 소스 클러스터와 대상 클러스터는 모두 동일한 AWS 계정에 있어야 합니다. 대상 클러스터에는 최소 3개의 브로커가 있어야 합니다.

튜토리얼: Amazon MSK Replicator 생성

소스 및 대상 클러스터를 설정한 후 해당 클러스터를 사용하여 Amazon MSK Replicator를 생성할 수 있습니다. Amazon MSK Replicator를 생성하기 전에 먼저 권한(MSK Replicator를 생성하는 데 필요한 IAM 권한)이 있는지 확인합니다.

주제

- Amazon MSK Replicator를 생성하기 위한 고려 사항

Amazon MSK Replicator를 생성하기 위한 고려 사항

다음 섹션에서는 MSK Replicator 기능을 사용하기 위한 사전 조건, 지원되는 구성 및 모범 사례를 간략히 소개합니다. 필요한 권한, 클러스터 호환성 및 서버리스별 요구 사항과 생성 후 Replicator 관리에 대한 지침을 다룹니다.

MSK Replicator를 생성하는 데 필요한 IAM 권한

다음은 MSK Replicator를 생성하는 데 필요한 IAM 정책의 예입니다. 이 `kafka:TagResource` 작업은 MSK Replicator를 생성할 때 태그가 제공된 경우에만 필요합니다. Replicator IAM 정책은 클라이언트에 해당하는 IAM 역할에 연결되어야 합니다. 권한 부여 정책 생성에 대한 자세한 내용은 권한 부여 정책 생성을 참조하세요.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "MSKReplicatorIAMPassRole",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/MSKReplicationRole",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "kafka.amazonaws.com"
        }
      }
    },
```

```
    {
      "Sid": "MSKReplicatorServiceLinkedRole",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::123456789012:role/aws-service-role/
kafka.amazonaws.com/AWSServiceRoleForKafka*"
    },
    {
      "Sid": "MSKReplicatorEC2Actions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs",
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:123456789012:subnet/subnet-0abcd1234ef56789",
        "arn:aws:ec2:us-east-1:123456789012:security-group/sg-0123abcd4567ef89",
        "arn:aws:ec2:us-east-1:123456789012:network-
interface/eni-0a1b2c3d4e5f67890",
        "arn:aws:ec2:us-east-1:123456789012:vpc/vpc-0a1b2c3d4e5f67890"
      ]
    },
    {
      "Sid": "MSKReplicatorActions",
      "Effect": "Allow",
      "Action": [
        "kafka:CreateReplicator",
        "kafka:TagResource"
      ],
      "Resource": [
        "arn:aws:kafka:us-
east-1:123456789012:cluster/myCluster/abcd1234-56ef-78gh-90ij-klmnopqrstuv",
        "arn:aws:kafka:us-
east-1:123456789012:replicator/myReplicator/wxyz9876-54vu-32ts-10rq-ponmlkjihgfe"
      ]
    }
  ]
}
```

다음은 복제기를 설명하는 IAM 정책의 예입니다. `kafka:DescribeReplicator` 작업과 `kafka:ListTagsForResource` 작업 중 하나가 필요하지만 둘 다 필요한 것은 아닙니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "kafka:DescribeReplicator",
                "kafka:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

MSK Replicator에 대해 지원되는 클러스터 유형 및 버전

지원되는 인스턴스 유형, Kafka 버전, 네트워크 구성에 대한 요구 사항입니다.

- MSK Replicator는 소스 및 대상 클러스터로 MSK 프로비저닝된 클러스터와 MSK 서버리스 클러스터를 모두 조합하여 지원합니다. 현재 MSK Replicator에서는 다른 유형의 Kafka 클러스터를 지원하지 않습니다.
- MSK 서버리스 클러스터는 IAM 액세스 제어가 필요하고, Apache Kafka ACL 복제를 지원하지 않으며 주제별 구성 복제를 제한적으로 지원합니다. [MSK Serverless란 무엇인가요?](#)을(를) 참조하세요.
- MSK Replicator는 소스 클러스터와 대상 클러스터가 같은 리전에 있는지 또는 다른 AWS 리전에 있는지와 관계없이 Apache Kafka 2.7.0 이상을 실행하는 클러스터에서만 지원됩니다.
- MSK Replicator는 인스턴스 유형이 m5.large 이상인 클러스터를 지원합니다. t3.small 클러스터는 지원되지 않습니다.
- MSK 프로비저닝 클러스터와 함께 MSK Replicator를 사용하는 경우 소스 및 대상 클러스터 모두에 최소 3개의 브로커가 필요합니다. 두 개의 가용성 영역에 있는 클러스터 간에 데이터를 복제할 수 있지만 해당 클러스터에는 최소 4개의 브로커가 필요합니다.

- 소스 및 대상 MSK 클러스터는 모두 동일한 AWS계정에 있어야 합니다. 다른 계정의 클러스터 간 복제는 지원되지 않습니다.
- 소스 및 대상 MSK 클러스터가 서로 다른 AWS리전(교차 리전)에 있는 경우 MSK Replicator를 사용하려면 소스 클러스터에 IAM 액세스 제어 방법에 대해 다중 VPC 프라이빗 연결이 켜져 있어야 합니다.

  교차 AWS 리전 MSK 복제에 대한 소스 클러스터의 다른 인증 방법에는 다중 VPC가 필요하지 않습니다.

  동일한 AWS 리전에 있는 클러스터 간에 데이터를 복제하는 경우에도 다중 VPC가 필요하지 않습니다. the section called “단일 리전에서의 다중 VPC 프라이빗 연결”을(를) 참조하세요.
- 동일한 주제 이름 복제(콘솔에서 동일한 주제 이름 유지)를 위해서는 Kafka 버전 2.8.1 이상을 실행하는 MSK 클러스터가 필요합니다.
- 동일한 주제 이름 복제(콘솔에서 동일한 주제 이름 유지) 구성의 경우 주기적 복제 위험을 방지하기 위해 MSK Replicator에서 생성하는 헤더를 변경하지 마세요(`__mskmr`).

지원되는 MSK Serverless 클러스터 구성

- MSK 서버리스는 주제 생성 중에 MSK 서버리스 대상 클러스터에 대한 주제 구성(`cleanup.policy`, `compression.type`, `max.message.bytes`, `retention.bytes`, `retention.ms`)의 복제를 지원합니다.
- MSK 서버리스는 주제 구성 동기화 중에 `compression.type`, `max.message.bytes`, `retention.bytes`, `retention.ms` 주제 구성만 지원합니다.
- 복제기는 대상 MSK 서버리스 클러스터에서 83개의 압축된 파티션을 사용합니다. 대상 MSK 서버리스 클러스터에 충분한 수의 압축된 파티션이 있는지 확인합니다. MSK 서버 규모 조정을(를) 참조하세요.

클러스터 구성 변경 사항

- MSK Replicator를 생성한 후에는 계층형 스토리지를 켜거나 끄지 않는 것이 좋습니다. 대상 클러스터가 계층화되지 않은 경우 소스 클러스터가 계층화되어 있는지 여부에 관계없이 MSK는 계층화된 스토리지 구성을 복사하지 않습니다. 복제기가 생성된 후 대상 클러스터에서 계층형 스토리지를 켜는 경우 복제기를 다시 생성해야 합니다. 계층화되지 않은 클러스터에서 계층화된 클러스터로 데이터를 복사하려는 경우 주제 구성을 복사해서는 안 됩니다. 기존 주제에서 계층형 스토리지 활성화 및 비활성화를 참조하세요.

- MSK Replicator를 생성한 후에는 클러스터 구성 설정을 변경하지 마세요. 클러스터 구성 설정은 MSK Replicator를 생성하는 동안 검증됩니다. MSK Replicator와 관련된 문제를 방지하려면 MSK Replicator가 생성된 후에는 다음 설정을 변경하지 마세요.
  - MSK 클러스터를 t3 인스턴스 유형으로 변경합니다.
  - 서비스 실행 역할 권한을 변경합니다.
  - MSK 다중 VPC 프라이빗 연결을 비활성화합니다.
  - 연결된 클러스터 리소스 기반 정책을 변경합니다.
  - 클러스터 보안 그룹 규칙을 변경합니다.

대상 클러스터 리전에서 AWS 콘솔을 사용하여 복제기 생성

다음 섹션에서는 Replicator를 생성하기 위한 단계별 콘솔 워크플로를 설명합니다.

Replicator 세부 정보

1. 대상 MSK 클러스터가 위치한 AWS리전에서 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 계정의 복제기 목록을 표시하려면 복제기를 선택합니다.
3. 복제기 생성을 선택합니다.
4. 복제기 세부 정보 창에서 새로운 복제기에 고유한 이름을 지정합니다.

소스 클러스터 선택

소스 클러스터에는 대상 MSK 클러스터로 복사하려는 데이터가 포함되어 있습니다.

1. 소스 클러스터 창에서 소스 클러스터가 있는 AWS 리전을 선택합니다.

   MSK 클러스터로 이동하여 클러스터 세부 정보 ARN을 확인하여 클러스터의 리전을 조회할 수 있습니다. 리전 이름은 ARN 문자열에 포함되어 있습니다. 다음 예제 ARN에서 `ap-southeast-2`는 클러스터 리전입니다.

   ```
   arn:aws:kafka:ap-southeast-2:123456789012:cluster/cluster-11/
   eec93c7f-4e8b-4baf-89fb-95de01ee639c-s1
   ```
2. 소스 클러스터의 ARN을 입력하거나 소스 클러스터를 찾아 선택합니다.
3. 소스 클러스터에 대한 서브넷을 선택합니다.

콘솔에 소스 클러스터의 리전에서 사용할 수 있는 서브넷이 표시되어 선택할 수 있습니다. 최소 2개의 서브넷을 선택해야 합니다. 동일 리전 MSK Replicator의 경우 소스 클러스터에 액세스하도록 설정한 서브넷과 대상 클러스터에 액세스하도록 설정한 서브넷은 동일한 가용 영역에 있어야 합니다.

4. MSK Replicator가 소스 클러스터에 액세스할 수 있는 보안 그룹을 선택합니다.

   - 크로스 리전 복제(CRR)의 경우 소스 클러스터의 보안 그룹을 제공할 필요가 없습니다.
   - 동일 리전 복제(SRR)의 경우 Amazon EC2 콘솔(https://console.aws.amazon.com/ec2/)로 이동하여 소스 클러스터의 보안 그룹으로의 트래픽을 허용하는 아웃바운드 규칙이 Replicator에 제공할 보안 그룹에 있는지 확인합니다. 또한 소스에 제공된 Replicator 보안 그룹으로부터의 트래픽을 허용하는 인바운드 규칙이 있는지 소스 클러스터의 보안 그룹에 확인합니다.

   소스 클러스터의 보안 그룹에 인바운드 규칙을 추가하려면

   1. AWS콘솔에서 클러스터 이름을 선택하여 소스 클러스터의 세부 정보로 이동합니다.
   2. 속성 탭을 선택한 다음 네트워크 설정 창까지 아래로 스크롤하여 적용된 보안 그룹의 이름을 선택합니다.
   3. 인바운드 규칙으로 이동하여 인바운드 규칙 편집을 선택합니다.
   4. 규칙 추가를 선택합니다.
   5. 새 규칙의 유형 열에서 사용자 지정 TCP를 선택합니다.
   6. 포트 범위 열에 `9098`을 입력합니다. MSK Replicator는 IAM 액세스 제어를 사용하여 포트 9098을 사용하는 클러스터에 연결합니다.
   7. 소스 열에 소스 클러스터에 대한 Replicator 생성 시 제공할 보안 그룹의 이름(MSK 소스 클러스터의 보안 그룹과 동일할 수 있음)을 입력한 다음, 규칙 저장을 선택합니다.

   소스에 제공된 Replicator의 보안 그룹에 아웃바운드 규칙을 추가하려면:

   1. Amazon EC2용 AWS콘솔에서 소스에 대한 복제기 생성 중에 제공할 보안 그룹으로 이동합니다.
   2. 아웃바운드 규칙으로 이동하여 아웃바운드 규칙 편집을 선택합니다.
   3. 규칙 추가를 선택합니다.
   4. 새 규칙의 유형 열에서 사용자 지정 TCP를 선택합니다.

5. 포트 범위 열에 `9098`을 입력합니다. MSK Replicator는 IAM 액세스 제어를 사용하여 포트 9098을 사용하는 클러스터에 연결합니다.
6. 소스 열에 MSK 소스 클러스터의 보안 그룹 이름을 입력한 다음, 규칙 저장을 선택합니다.

> **Note**
>
> 또는 보안 그룹을 사용하여 트래픽을 제한하지 않으려면 모든 트래픽을 허용하는 인바운드 및 아웃바운드 규칙을 추가할 수 있습니다.
> 1. 규칙 추가를 선택합니다.
> 2. 유형 열에서 모든 트래픽을 선택합니다.
> 3. 소스 열에 `0.0.0.0/0`을 입력한 다음 규칙 저장을 선택합니다.

대상 클러스터 선택

대상 클러스터는 소스 데이터가 복사되는 MSK 프로비저닝 클러스터 또는 서버리스 클러스터입니다.

> **Note**
>
> MSK Replicator는 주제 이름에 자동 생성된 접두사를 추가하여 대상 클러스터에 새 주제를 생성합니다. 예를 들어 MSK Replicator는 소스 클러스터의 “`topic`”에 있는 데이터를 `<sourceKafkaClusterAlias>.topic`이라는 대상 클러스터의 새 주제로 복제합니다. 이는 소스 클러스터에서 복제된 데이터가 포함된 주제를 대상 클러스터의 다른 주제와 구분하고 클러스터 간에 데이터가 순환 복제되는 것을 방지하기 위한 것입니다. 대상 클러스터의 주제 이름에 추가될 접두사는 `DescribeReplicator` API 또는 MSK 콘솔의 복제기 세부 정보 페이지를 사용하여 sourceKafkaClusterAlias 필드 아래에서 찾을 수 있습니다. 대상 클러스터의 접두사는 `<sourceKafkaClusterAlias>`입니다.

1. 대상 클러스터 창에서 대상 클러스터가 위치한 AWS리전을 선택합니다.
2. 대상 클러스터의 ARN을 입력하거나 대상 클러스터를 찾아 선택합니다.
3. 대상 클러스터에 대한 서브넷을 선택합니다.

   콘솔에 대상 클러스터의 리전에서 사용할 수 있는 서브넷이 표시되어 선택할 수 있습니다. 최소 2개의 서브넷을 선택합니다.
4. MSK Replicator가 대상 클러스터에 액세스할 수 있는 보안 그룹을 선택합니다.

대상 클러스터의 리전에서 사용할 수 있는 보안 그룹이 표시되어 선택할 수 있습니다. 선택한 보안 그룹이 각 연결과 관련이 있습니다. 보안 그룹 사용에 대한 자세한 내용은 Amazon VPC 사용 설명서의 [보안 그룹을 사용하여 AWS리소스에 대한 트래픽 제어를 참조하세요](#).

- 크로스 리전 복제(CRR)의 경우 Amazon EC2 콘솔(https://console.aws.amazon.com/ec2/)로 이동하여 대상 클러스터의 보안 그룹으로의 트래픽을 허용하는 아웃바운드 규칙이 Replicator에 제공할 보안 그룹에 있는지 확인합니다. 또한 대상 클러스터의 보안 그룹에 대상에 제공된 복제기 보안 그룹으로부터의 트래픽을 허용하는 인바운드 규칙이 있는지 확인합니다.

대상 클러스터의 보안 그룹에 대해 인바운드 규칙을 추가하려면

1. AWS콘솔에서 클러스터 이름을 선택하여 대상 클러스터의 세부 정보로 이동합니다.
2. 속성 탭을 선택한 다음, 네트워크 설정 창까지 아래로 스크롤하여 적용된 보안 그룹의 이름을 선택합니다.
3. 인바운드 규칙으로 이동하여 인바운드 규칙 편집을 선택합니다.
4. 규칙 추가를 선택합니다.
5. 새 규칙의 유형 열에서 사용자 지정 TCP를 선택합니다.
6. 포트 범위 열에 9098을 입력합니다. MSK Replicator는 IAM 액세스 제어를 사용하여 포트 9098을 사용하는 클러스터에 연결합니다.
7. 소스 열에 대상 클러스터에 대한 Replicator를 생성하는 동안 제공할 보안 그룹의 이름(MSK 대상 클러스터의 보안 그룹과 동일할 수 있음)을 입력한 다음 규칙 저장을 선택합니다.

대상에 대해 제공된 Replicator의 보안 그룹에 아웃바운드 규칙을 추가하려면:

1. AWS콘솔에서 대상에 대한 복제기 생성 중에 제공할 보안 그룹으로 이동합니다.
2. 속성 탭을 선택한 다음, 네트워크 설정 창까지 아래로 스크롤하여 적용된 보안 그룹의 이름을 선택합니다.
3. 아웃바운드 규칙으로 이동하여 아웃바운드 규칙 편집을 선택합니다.
4. 규칙 추가를 선택합니다.
5. 새 규칙의 유형 열에서 사용자 지정 TCP를 선택합니다.
6. 포트 범위 열에 9098을 입력합니다. MSK Replicator는 IAM 액세스 제어를 사용하여 포트 9098을 사용하는 클러스터에 연결합니다.
7. 소스 열에 MSK 대상 클러스터의 보안 그룹 이름을 입력한 다음, 규칙 저장을 선택합니다.

> **Note**
>
> 또는 보안 그룹을 사용하여 트래픽을 제한하지 않으려면 모든 트래픽을 허용하는 인바운드 및 아웃바운드 규칙을 추가할 수 있습니다.
>
> 1. 규칙 추가를 선택합니다.
> 2. 유형 열에서 모든 트래픽을 선택합니다.
> 3. 소스 열에 `0.0.0.0/0`을 입력한 다음 규칙 저장을 선택합니다.

복제기 설정 및 권한 구성

1. 복제기 설정 창에서 허용 및 거부 목록의 정규 표현식을 사용하여 복제할 주제를 지정합니다. 기본적으로 모든 주제가 복제됩니다.

    > **Note**
    >
    > MSK Replicator는 정렬된 순서로 최대 750개의 주제만 복제합니다. 더 많은 주제를 복제해야 하는 경우 별도의 Replicator를 생성하는 것이 좋습니다. 복제기당 750개 이상의 주제에 대한 지원이 필요한 경우 AWS콘솔 지원 센터로 이동하여 [지원 사례를 생성합니다](). 'TopicCount' 지표를 사용하여 복제되는 주제 수를 모니터링할 수 있습니다. [Amazon MSK Standard 브로커 할당량]()을(를) 참조하세요.

2. 기본적으로 MSK Replicator는 선택한 주제의 최신(가장 최근) 오프셋에서 복제를 시작합니다. 또는 주제에 기존 데이터를 복제하려는 경우 선택한 주제의 가장 빠른(가장 오래된) 오프셋에서 복제를 시작할 수 있습니다. Replicator가 생성된 후에는 이 설정을 변경할 수 없습니다. 이 설정은 [`CreateReplicator`]() 요청 및 [`DescribeReplicator`]() 응답 API의 [`startingPosition`]() 필드에 해당합니다.

3. 다음과 같은 주제 이름 구성을 선택합니다.

    - `PREFIXED` 주제 이름 복제(콘솔의 주제 이름에 접두사 추가): 기본 설정입니다. MSK Replicator는 소스 클러스터에서 이름이 `<sourceKafkaClusterAlias>.topic1`인 대상 클러스터의 새 주제로 'topic1'을 복제합니다.
    - 동일한 주제 이름 복제(콘솔에 동일한 주제 이름 유지): 소스 클러스터의 주제는 대상 클러스터에 동일한 주제 이름으로 복제됩니다.

이 설정은 `CreateReplicator` 요청 및 `DescribeReplicator` 응답 API의 `TopicNameConfiguration` 필드에 해당합니다. Amazon MSK Replicator 작동 방식을(를) 참조하세요.

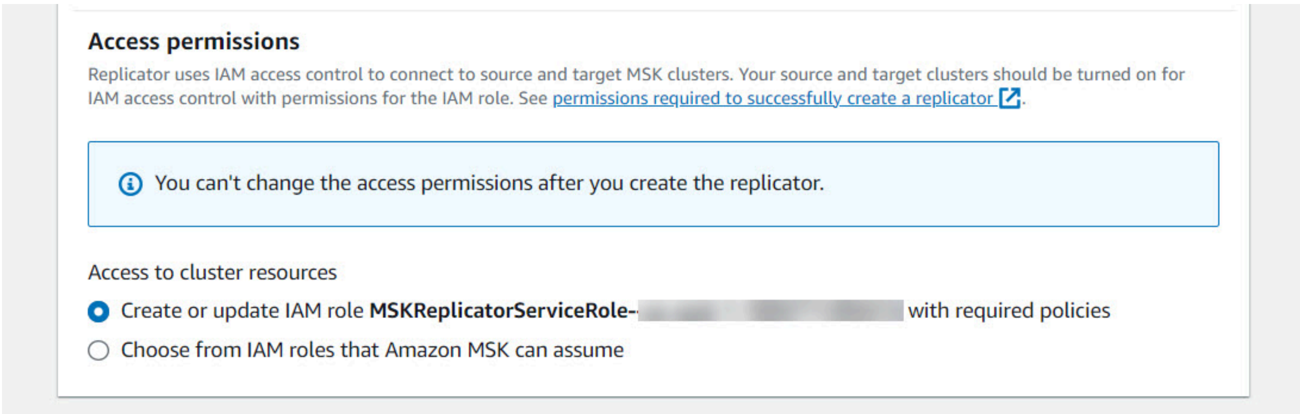
> **Note**
>
> MSK Replicator는 기본적으로 주제 이름에 자동 생성된 접두사를 추가하여 대상 클러스터에 새 주제를 생성합니다. 이는 소스 클러스터에서 복제된 데이터가 포함된 주제를 대상 클러스터의 다른 주제와 구분하고 클러스터 간에 데이터가 순환 복제되는 것을 방지하기 위한 것입니다. 또는 복제 중에 주제 이름이 유지되도록 동일한 주제 이름 복제(콘솔에 동일한 주제 이름 유지)를 사용하여 MSK Replicator를 생성할 수 있습니다. 이 구성을 사용하면 설정 중에 클라이언트 애플리케이션을 재구성할 필요가 줄어들고 다중 클러스터 스트리밍 아키텍처를 더 쉽게 운영할 수 있습니다.

4. 기본적으로 MSK Replicator는 원활한 장애 조치를 위해 주제 구성, 액세스 제어 목록(ACL), 소비자 그룹 오프셋을 포함한 모든 메타데이터를 복사합니다. 장애 조치를 위해 복제기를 생성하지 않는 경우 추가 설정 섹션에서 사용 가능한 설정 중 하나 이상을 선택적으로 비활성화할 수 있습니다.

> **Note**
>
> 생산자가 대상 클러스터의 복제된 주제에 직접 쓰면 안 되므로 MSK Replicator는 쓰기 ACL을 복제하지 않습니다. 생산자는 장애 조치 후 대상 클러스터의 로컬 주제에 써야 합니다. 세부 정보는 보조 AWS리전으로 계획된 장애 조치 수행 섹션을 참조하세요.

5. 소비자 그룹 복제 창에서 허용 및 거부 목록의 정규 표현식을 사용하여 복제할 소비자 그룹을 지정합니다. 기본적으로 모든 소비자 그룹이 복제됩니다.
6. 압축 창에서 대상 클러스터에 쓰여진 데이터를 압축하도록 선택할 수 있습니다(선택 사항). 압축을 사용하려는 경우 원본 클러스터의 데이터와 동일한 압축 방법을 사용하는 것을 권장합니다.
7. 액세스 권한 패널에서 다음 중 하나를 수행합니다.

    a. 필요한 정책으로 IAM 역할 생성 또는 업데이트를 선택합니다. MSK 콘솔은 소스 및 대상 MSK 클러스터를 읽고 쓰는 데 필요한 서비스 실행 역할에 필요한 권한과 신뢰 정책을 자동으로 연결합니다.



b. Amazon MSK가 맡을 수 있는 IAM 역할 중에서 선택을 선택하여 자체 IAM 역할을 제공합니다. 자체 IAM 정책을 작성하는 대신 `AWSMSKReplicatorExecutionRole` 관리형 IAM 정책을 서비스 실행 역할에 연결하는 것이 좋습니다.

- Replicator에서 신뢰 정책의 일부로 아래 JSON과 역할에 연결된 `AWSMSKReplicatorExecutionRole`을 사용하여 소스 및 대상 MSK 클러스터를 읽고 쓰는 데 사용할 IAM 역할을 생성합니다. 신뢰 정책에서 자리 표시자 <yourAccountID>를 실제 계정 ID로 변경합니다.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "kafka.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<yourAccountID>"
                }
            }
        }
    ]
}
```

8. 복제기 태그 창에서 MSK Replicator 리소스에 태그를 할당할 수 있습니다(선택 사항). 자세한 내용은 Amazon MSK 클러스터에 태그 지정 단원을 참조하십시오. 리전 간 MSK Replicator의 경우 복제기가 생성되면 태그가 원격 리전으로 자동 동기화됩니다. 복제기를 생성한 후 태그를 변경하면 변경 사항이 원격 리전에 자동으로 동기화되지 않으므로 로컬 복제기와 원격 복제기 참조를 수동으로 동기화해야 합니다.
9. 생성을 선택합니다.

`kafka-cluster:WriteData` 권한을 제한하려면 Amazon MSK에 대한 IAM 액세스 제어 작동 방식 의 권한 부여 정책 생성 섹션을 참조하세요. 소스 클러스터와 대상 클러스터 모두에 `kafka-cluster:WriteDataIdempotently` 권한을 추가해야 합니다.

MSK Replicator가 성공적으로 생성되고 실행 중 상태로 전환되는 데 약 30분이 소요됩니다.

삭제한 복제기를 대체할 새 MSK Replicator를 생성하면 새 복제기가 최신 오프셋부터 복제를 시작합니다.

MSK Replicator가 실패 상태로 전환된 경우 문제 해결 섹션 MSK Replicator 문제 해결을 참조하세요.

MSK Replicator 설정 편집

MSK Replicator가 생성된 후에는 소스 클러스터, 대상 클러스터, Replicator 시작 위치 또는 주제 이름 복제 구성을 변경할 수 없습니다. 동일한 주제 이름 복제 구성을 사용하려면 새로운 Replicator를 생성해야 합니다. 그러나 복제할 주제 및 소비자 그룹과 같은 다른 Replicator 설정을 편집할 수 있습니다.

1. 에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Replicator를 선택하여 계정의 Replicator 목록을 표시하고 편집하려는 MSK Replicator를 선택합니다.
3. 속성 탭을 선택합니다.
4. 복제기 설정 섹션에서 복제기 편집을 선택합니다.
5. 이러한 설정을 변경하여 MSK Replicator 설정을 편집할 수 있습니다.
   - 허용 및 거부 목록에서 정규 표현식을 사용하여 복제할 주제를 지정합니다. 기본적으로 MSK Replicator는 원활한 장애 조치를 위해 주제 구성, 액세스 제어 목록(ACL), 소비자 그룹 오프셋을 포함한 모든 메타데이터를 복사합니다. 장애 조치를 위해 복제기를 생성하지 않는 경우 추가 설정 섹션에서 사용 가능한 설정 중 하나 이상을 선택적으로 비활성화할 수 있습니다.

> **Note**
>
> 생산자가 대상 클러스터의 복제된 주제에 직접 쓰면 안 되므로 MSK Replicator는 쓰기 ACL을 복제하지 않습니다. 생산자는 장애 조치 후 대상 클러스터의 로컬 주제에 써야 합니다. 세부 정보는 보조 AWS리전으로 계획된 장애 조치 수행 섹션을 참조하세요.

- 소비자 그룹 복제의 경우 허용 및 거부 목록에서 정규 표현식을 사용하여 복제할 소비자 그룹을 지정할 수 있습니다. 기본적으로 모든 소비자 그룹이 복제됩니다. 허용 및 거부 목록이 비어 있으면 소비자 그룹 복제가 비활성화됩니다.
- 대상 압축 유형에서 대상 클러스터에 기록되는 데이터를 압축할지 여부를 선택할 수 있습니다. 압축을 사용하려는 경우 원본 클러스터의 데이터와 동일한 압축 방법을 사용하는 것을 권장합니다.

6. 변경 내용을 저장합니다.

   MSK Replicator를 성공적으로 생성하고 실행 상태로 전환하는 데는 약 30분이 소요됩니다. MSK Replicator가 실패 상태로 전환된 경우 문제 해결 섹션 ???을 참조하세요.

MSK Replicator 삭제

MSK Replicator가 생성에 실패한 경우 삭제해야 할 수 있습니다(실패 상태). MSK Replicator가 생성된 후에는 MSK Replicator에 할당된 소스 및 타겟 클러스터를 변경할 수 없습니다. 기존 MSK Replicator를 삭제하고 새로 생성할 수 있습니다. 삭제된 복제기를 대체할 새 MSK Replicator를 생성하면 새 복제기가 최신 오프셋부터 복제를 시작합니다.

1. 소스 클러스터가 위치한 AWS리전에서에 로그인AWS Management 콘솔하고 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ Amazon MSK 콘솔을 엽니다.
2. 탐색 창에서 복제기를 선택합니다.
3. MSK Replicator 목록에서 삭제하려는 복제기를 선택하고 삭제를 선택합니다.

복제 모니터링

대상 클러스터 리전에서 https://console.aws.amazon.com/cloudwatch/를 사용하여 각 Amazon MSK Replicator의 주제 및 집계 수준에서 `ReplicationLatency`, `MessageLag`, `ReplicatorThroughput`에 대한 지표를 볼 수 있습니다. 지표는 "AWS/Kafka" 네임스페이스의

ReplicatorName 아래에 표시됩니다. `ReplicatorFailure`, `AuthError`, `ThrottleTime` 지표를 확인하여 문제를 확인할 수도 있습니다.

MSK 콘솔은 각 MSK Replicator에 대한 CloudWatch 지표의 하위 세트를 표시합니다. 콘솔 복제기 목록에서 복제기의 이름을 선택하고 모니터링 탭을 선택합니다.

MSK Replicator 지표

다음 지표는 MSK Replicator의 성능 또는 연결 지표에 대해 설명합니다.

AuthError 지표는 주제 수준의 인증 오류를 다루지 않습니다. MSK Replicator의 주제 수준 인증 오류를 모니터링하려면 복제기의 ReplicationLatency 지표와 소스 클러스터의 주제 수준 지표인 MessagesInPerSec를 모니터링하세요. 주제의 ReplicationLatency가 0으로 떨어졌지만 주제에 여전히 생성되는 데이터가 있는 경우 이는 복제기에 해당 주제에 대한 인증 문제가 있음을 나타냅니다. 복제기의 서비스 실행 IAM 역할에 주제에 액세스할 수 있는 충분한 권한이 있는지 확인합니다.

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계
성능	ReplicationLatency	소스에서 대상 클러스터로 레코드를 복제하는 데 걸리는 시간과 소스에서 레코드 생성 시간과 대상에 복제되는 시간 사이의 기간입니다. ReplicationLatency가 증가하면 클러스터에 복제를 지원할 만큼 충분한 파티션이 있는지 확인합니다. 높은 처리량에 비해 파티션 수가 너무 적	ReplicatorName	밀리초	Partition	최대
			ReplicatorName, Topic	밀리초	Partition	최대

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계	
		으면 복제 대기 시간이 길어질 수 있습니다.					

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계
성능	MessageLag	MSK Replicator와 소스 클러스터 간 동기화를 모니터링합니다. MessageLag는 소스 클러스터에 생성된 메시지와 Replicator에서 사용한 메시지 간의 지연을 나타냅니다. 소스 클러스터와 대상 클러스터 간의 지연은 아닙니다. 소스 클러스터를 사용할 수 없거나 중단하더라도 Replicator는 대상 클러스터에 사용한 메시지 작성을 완료합니다. 중단 후 MessageLag는 증가 표시를 통해 복제기가 소스 클러스터보다 뒤처져 있음을 나타내는 메시지 수를 알리며, 메시지 수가 0이 될 때까지, 즉 복제기	ReplicatorName	개수	Partition	Sum
			ReplicatorName, Topic	개수	Partition	Sum

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계
		가 소스 클러스터를 따라잡을 때까지 이 표시를 모니터링할 수 있습니다.				

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계
성능	ReplicatorBytesInPerSec	초당 평균 Replicator에서 처리하는 바이트 수입니다. MSK Replicator에서 처리하는 데이터는 MSK Replicator가 수신하는 모든 데이터로 구성되며, 여기에는 대상 클러스터에 복제된 데이터와 MSK Replicator에서 필터링한 데이터(Replicator가 동일한 주제 이름 구성으로 구성된 경우에만 해당)가 포함되어 데이터가 시작된 동일한 주제로 다시 복사되는 것을 방지합니다. Replicator가 '접두사 지정된' 주제 이름 구성에 따라 구성된 경우 `ReplicatorBytesInPerSec` 및 `Replicato`	ReplicatorName	BytesPerSecond	ReplicatorName	Sum

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계
		rThroughput 지표 모두 MSK Replicator 에서 필터링하는 데이터가 없으므로 값이 동일합니다.				
성능	ReplicatorThroughput	초당 평균 복제되는 바이트 수입니다. 주제에 대해 ReplicatorThroughput이 떨어지면 KafkaClusterPingSuccessCount와 AuthError 지표를 확인하여 복제기가 클러스터와 통신할 수 있는지 확인한 다음 클러스터 지표를 확인하여 클러스터가 다운되지 않았는지 확인합니다.	ReplicatorName	BytesPerSecond	Partition	Sum
			ReplicatorName, Topic	BytesPerSecond	Partition	Sum

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계
디버그	AuthError	초당 인증에 실패한 연결 수입니다. 이 지표가 0보다 크면 복제기에 대한 서비스 실행 역할 정책이 유효한지 확인하고 클러스터 권한에 대해 거부 권한이 설정되어 있지 않은지 확인할 수 있습니다. clusterAlias 차원을 기반으로 소스 또는 대상 클러스터에서 인증 오류가 발생하고 있는지 확인할 수 있습니다.	ReplicatorName, ClusterAlias	개수	작업자	Sum

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계
디버그	ThrottleTime	클러스터의 브로커가 요청을 제한한 평균 시간(ms)입니다. MSK Replicator가 클러스터를 과도하게 사용하지 않도록 제한을 설정합니다. 해당 지표가 0이면 replicationLatency가 높지 않고 replicatorThroughput이 예상과 같으면 제한이 예상대로 작동하는 것입니다. 이 지표가 0보다 크면 그에 따라 제한을 조정할 수 있습니다.	ReplicatorName, ClusterAlias	밀리초	작업자	최대
디버그	ReplicatorFailure	복제기에서 발생한 장애 횟수입니다.	ReplicatorName	개수		합계

지표 유형	지표	설명	측정 기준	단위	원시 지표 세부 수준	원시 지표 집계 통계
디버그	KafkaClusterPingSuccessCount	Kafka 클러스터에 대한 복제기 연결의 상태를 나타냅니다. 이 값이 1이면 연결 상태는 정상입니다. 값이 0이거나 데이터 포인트가 없으면 연결이 정상적이지 않은 것입니다. 값이 0이면 Kafka 클러스터에 대한 네트워크 또는 IAM 권한 설정을 확인할 수 있습니다. ClusterAlias 차원을 기반으로 해당 지표가 소스 클러스터에 대한 것인지 대상 클러스터에 대한 것인지 식별할 수 있습니다.	ReplicatorName, ClusterAlias	개수		합계

복제를 사용하여 여러 리전에 걸쳐 Kafka 스트리밍 애플리케이션의 복원력 향상

MSK Replicator를 사용하여 액티브-액티브 또는 액티브-패시브 클러스터 토폴로지를 설정하여 AWS 리전 간 Apache Kafka 애플리케이션의 복원력을 높일 수 있습니다. 액티브-액티브 설정에서는 두

MSK 클러스터가 모두 읽기 및 쓰기를 적극적으로 제공합니다. 액티브-패시브 설정에서는 한 번에 하나의 MSK 클러스터만 스트리밍 데이터를 적극적으로 서비스하고 다른 클러스터는 대기 상태로 유지됩니다.

다중 리전 Apache Kafka 애플리케이션 빌드의 고려 사항

소비자는 다운스트림에 영향을 주지 않으면서 중복된 메시지를 재처리할 수 있어야 합니다. MSK Replicator는 대기 클러스터에 중복이 발생할 수 있는 데이터를 최소 한 번 복제합니다. 보조 AWS리전으로 전환하면 소비자가 동일한 데이터를 두 번 이상 처리할 수 있습니다. MSK Replicator는 더 나은 성능을 위해 소비자 오프셋보다 데이터 복사에 우선 순위를 둡니다. 장애 조치 후 소비자는 이전 오프셋부터 읽기를 시작하므로 중복 처리가 발생할 수 있습니다.

생산자와 소비자는 최소한의 데이터 손실도 감수해야 합니다. MSK Replicator는 데이터를 비동기식으로 복제하므로 기본 AWS리전에서 오류가 발생하기 시작해도 모든 데이터가 보조 리전에 복제된다는 보장은 없습니다. 복제 대기 시간을 사용하여 보조 리전으로 복사되지 않은 최대 데이터를 결정할 수 있습니다.

액티브-액티브 대 액티브-패시브 클러스터 토폴로지 사용

액티브-액티브 클러스터 토폴로지는 거의 0에 가까운 복구 시간과 스트리밍 애플리케이션이 여러 AWS 리전에서 동시에 작동할 수 있도록 하는 기능을 제공합니다. 한 리전의 클러스터가 손상되더라도 다른 리전의 클러스터에 연결된 애플리케이션이 데이터를 계속 처리합니다.

액티브-패시브 설정은 한 번에 하나의 AWS 리전에서만 실행할 수 있는 애플리케이션이나 데이터 처리 순서를 세부적으로 제어해야 하는 경우에 적합합니다. 액티브-패시브 설정은 장애 조치 후 스트리밍 데이터를 다시 시작하려면 보조 리전에서 생산자와 소비자를 포함한 전체 액티브-패시브 설정을 시작해야 하므로 액티브-액티브 설정보다 복구 시간이 더 오래 걸립니다.

권장 주제 이름 지정 구성을 사용하여 액티브-패시브 Kafka 클러스터 설정 생성

액티브-패시브 설정의 경우 서로 다른 두 AWS리전에서 생산자, MSK 클러스터 및 소비자(동일한 소비자 그룹 이름)의 유사한 설정을 운영하는 것이 좋습니다. 안정적인 데이터 복제를 보장하려면 두 MSK 클러스터의 읽기 및 쓰기 용량이 동일해야 합니다. 기본 클러스터에서 대기 클러스터로 데이터를 지속적으로 복사하려면 MSK Replicator를 생성해야 합니다. 또한 동일한 AWS리전의 클러스터에 있는 주제에 데이터를 쓰도록 생산자를 구성해야 합니다.

액티브-패시브 설정의 경우 주제 이름이 동일한 새로운 Replicator를 생성하여(콘솔에서 동일한 주제 이름 유지) 기본 리전의 MSK 클러스터에서 보조 리전의 클러스터로 데이터 복제를 시작합니다. 부트

스트랩 문자열을 사용하여 각각 자체 AWS리전의 클러스터에 연결하는 두 리전에서 중복된 생산자 및 소비자 집합을 운영하는 것이 좋습니다. 이렇게 하면 부트스트랩 문자열을 변경할 필요가 없으므로 장애 조치 프로세스가 간소화됩니다. 소비자가 중단한 위치 근처에서 읽도록 하려면 소스 클러스터와 대상 클러스터의 소비자는 동일한 소비자 그룹 ID를 가져야 합니다.

MSK Replicator에 대해 동일한 주제 이름 복제(콘솔에서 동일한 주제 이름 유지)를 사용하는 경우 해당 소스 주제와 동일한 이름으로 주제가 복제됩니다.

대상 클러스터에서 클라이언트에 대해 클러스터 수준 설정 및 권한을 구성하는 것이 좋습니다. 액세스 제어 목록을 복사하는 옵션을 선택한 경우 MSK Replicator에서 자동으로 복사하므로 주제 수준 설정과 리터럴 읽기 ACL을 구성할 필요가 없습니다. 메타데이터 복제을(를) 참조하세요.

보조 AWS 리전으로의 장애 조치

Amazon CloudWatch를 사용하여 보조 AWS리전의 복제 지연 시간을 모니터링하는 것이 좋습니다. 기본 AWS리전의 서비스 이벤트 중에 복제 지연 시간이 갑자기 증가할 수 있습니다. 지연 시간이 계속 증가하는 경우 AWS서비스 상태 대시보드를 사용하여 기본 AWS리전의 서비스 이벤트를 확인합니다. 이벤트가 있는 경우 보조 AWS리전으로 장애 조치를 수행할 수 있습니다.

보조 AWS리전으로 계획된 장애 조치 수행

계획된 장애 조치를 수행하여 소스 MSK 클러스터가 있는 기본 AWS리전의 예상치 못한 이벤트에 대해 애플리케이션의 복원력을 테스트할 수 있습니다. 계획된 장애 조치로 인해 데이터가 손실되어서는 안 됩니다.

동일한 주제 이름 복제 구성을 사용하는 경우 다음 단계를 따르세요.

1. 소스 클러스터에 연결하는 모든 생산자와 소비자를 종료합니다.
2. 새 MSK Replicator를 생성하여 보조 리전의 MSK 클러스터에서 기본 리전의 MSK 클러스터로 동일한 주제 이름 복제를 통해 데이터를 복제합니다(콘솔에서 동일한 주제 이름 유지). 예기치 않은 이벤트가 종료된 후 기본 리전으로 페일백할 수 있도록 보조 리전에 기록할 데이터를 다시 기본 리전으로 복사하는 데 필요합니다.
3. 보조 AWS리전의 대상 클러스터에 연결된 생산자 및 소비자를 시작합니다.

접두사 지정된 주제 이름 구성을 사용하는 경우 다음 단계를 따라 장애 조치를 수행합니다.

1. 소스 클러스터에 연결하는 모든 생산자와 소비자를 종료합니다.

2. 새 MSK Replicator를 생성하여 보조 리전의 MSK 클러스터에서 기본 리전의 MSK 클러스터로 데이터를 복제합니다. 예기치 않은 이벤트가 종료된 후 기본 리전으로 페일백할 수 있도록 보조 리전에 기록할 데이터를 다시 기본 리전으로 복사하는 데 필요합니다.
3. 보조 AWS리전의 대상 클러스터에서 생산자를 시작합니다.
4. 애플리케이션의 메시지 정렬 요구 사항에 따라 다음 탭 중 하나의 단계를 따릅니다.

    No message ordering

    애플리케이션에 메시지 순서 지정이 필요하지 않은 경우 와일드카드 연산자(예:)를 사용하여 로컬(예: 주제) 및 복제된 주제(예: `<sourceKafkaClusterAlias>.topic`) 모두에서 읽는 보조 AWS리전에서 소비자를 시작합니다`.*topic`.

    Message ordering

    애플리케이션에 메시지 정렬이 필요한 경우 대상 클러스터의 복제된 토픽(예: `<sourceKafkaClusterAlias>.topic`)에 대해서만 소비자를 시작하고 로컬 토픽(예: `topic`)에 대해서는 시작하지 마세요.
5. 대상 MSK 클러스터에 복제된 주제의 모든 소비자가 모든 데이터 처리를 완료하여 소비자 지연이 0이 되고 처리된 레코드 수 또한 0이 될 때까지 기다립니다. 그런 다음 대상 클러스터에서 복제된 주제에 대한 소비자를 중지합니다. 이 시점에서 소스 MSK 클러스터에서 대상 MSK 클러스터로 복제된 모든 레코드가 소비되었습니다.
6. 대상 MSK 클러스터에서 로컬 주제(예: `topic`)에 대한 소비자를 시작합니다.

보조 AWS리전에 대한 계획되지 않은 장애 조치 수행

소스 MSK 클러스터가 있는 기본 AWS리전에 서비스 이벤트가 있고 트래픽을 대상 MSK 클러스터가 있는 보조 리전으로 일시적으로 리디렉션하려는 경우 계획되지 않은 장애 조치를 수행할 수 있습니다. MSK Replicator가 데이터를 비동기적으로 복제하므로 계획되지 않은 장애 조치로 인해 일부 데이터가 손실될 수 있습니다. ???의 지표를 사용하여 메시지 지연을 추적할 수 있습니다.

동일한 주제 이름 복제 구성(콘솔에서 동일한 주제 이름 유지)을 사용하는 경우 다음 단계를 따르세요.

1. 기본 리전의 소스 MSK 클러스터에 연결하는 모든 생산자와 소비자를 종료하려고 시도합니다. 해당 리전의 장애로 인해 이 작업이 성공하지 못할 수 있습니다.
2. 보조 AWS리전의 대상 MSK 클러스터에 연결하는 생산자와 소비자를 시작하여 장애 조치를 완료합니다. MSK Replicator는 읽기 ACL 및 소비자 그룹 오프셋을 비롯한 메타데이터도 복제하므로 생산자와 소비자는 장애 조치 전에 중단한 위치 근처에서 처리를 원활하게 재개합니다.

PREFIX 주제 이름 구성을 사용하는 경우 다음 단계에 따라 장애 조치를 수행하세요.

1. 기본 리전의 소스 MSK 클러스터에 연결하는 모든 생산자와 소비자를 종료하려고 시도합니다. 해당 리전의 장애로 인해 이 작업이 성공하지 못할 수 있습니다.
2. 보조 AWS리전의 대상 MSK 클러스터에 연결하는 생산자와 소비자를 시작하여 장애 조치를 완료합니다. MSK Replicator는 읽기 ACL 및 소비자 그룹 오프셋을 비롯한 메타데이터도 복제하므로 생산자와 소비자는 장애 조치 전에 중단한 위치 근처에서 처리를 원활하게 재개합니다.
3. 애플리케이션의 메시지 정렬 요구 사항에 따라 다음 탭 중 하나의 단계를 따릅니다.

No message ordering

애플리케이션에 메시지 순서 지정이 필요하지 않은 경우 와일드카드 연산자(예: `topic`)를 사용하여 로컬(예:) 및 복제된 주제(예: `<sourceKafkaClusterAlias>.topic`) 모두에서 읽는 대상 AWS리전의 소비자를 시작합니다`.*topic`.

Message ordering

1. 대상 클러스터의 복제된 주제(예: `<sourceKafkaClusterAlias>.topic`)에 대해서만 소비자를 시작하고 로컬 주제(예: `topic`)는 시작하지 않습니다.
2. 대상 MSK 클러스터에 복제된 주제의 모든 소비자가 모든 데이터 처리를 완료하여 오프셋 지연이 0이 되고 처리된 레코드 수 또한 0이 될 때까지 기다립니다. 그런 다음 대상 클러스터에서 복제된 주제에 대한 소비자를 중지합니다. 이 시점에서 소스 MSK 클러스터에서 대상 MSK 클러스터로 복제된 모든 레코드가 소비되었습니다.
3. 대상 MSK 클러스터에서 로컬 주제(예: `topic`)에 대한 소비자를 시작합니다.

4. 기본 리전에서 서비스 이벤트가 종료되면 새 MSK Replicator를 생성하여 보조 리전의 MSK 클러스터에서 기본 리전의 MSK 클러스터로 데이터를 복제합니다. 이때 Replicator 시작 위치는 가장 빠른 위치로 설정됩니다. 이는 서비스 이벤트가 종료된 후 기본 리전으로 페일백할 수 있도록 보조 리전에 쓸 데이터를 다시 기본 리전으로 복사하는 데 필요합니다. Replicator 시작 위치를 가장 빠른 위치로 설정하지 않으면 기본 리전의 서비스 이벤트 중에 보조 리전의 클러스터에 생성한 모든 데이터는 기본 리전의 클러스터로 다시 복사되지 않습니다.

기본 AWS 리전으로 장애 복구 수행

해당 AWS리전의 서비스 이벤트가 종료된 후 기본 리전으로 페일백할 수 있습니다.

동일한 주제 이름 복제 구성을 사용하는 경우 다음 단계를 따르세요.

1. 보조 클러스터를 소스로, 기본 클러스터를 대상으로 하여 새 MSK Replicator를 생성합니다. 시작 위치는 가장 빠른 동일한 주제 이름 복제로 설정됩니다(콘솔에서 동일한 주제 이름 유지).

   이렇게 하면 장애 조치 후 보조 클러스터에 작성된 모든 데이터를 기본 리전으로 다시 복사하는 프로세스가 시작됩니다.
2. Amazon CloudWatch에서 새로운 Replicator의 `MessageLag` 지표가 `0`에 도달할 때까지 모니터링합니다. 이 도달은 모든 데이터가 보조에서 기본으로 복제되었음을 나타냅니다.
3. 모든 데이터가 복제되면 보조 클러스터에 연결하는 모든 생산자를 중지하고 기본 클러스터에 연결하는 생산자를 시작합니다.
4. 소비자가 보조 클러스터에 연결할 때까지 `MaxOffsetLag` 지표가 `0`이 될 때까지 기다렸다가 모든 데이터를 처리되었는지 확인합니다. [소비자 지연 모니터링](#)을(를) 참조하세요.
5. 모든 데이터가 처리되면 보조 리전의 소비자를 중지하고 소비자를 기본 클러스터에 연결하여 장애 복구를 완료합니다.
6. 보조 클러스터에서 기본 클러스터로 데이터를 복제하는 첫 번째 단계에서 생성한 Replicator를 삭제합니다.
7. 기본 클러스터에서 보조 클러스터로 데이터를 복사하는 기존 Replicator의 상태가 'RUNNING'이며 Amazon CloudWatch의 `ReplicatorThroughput` 지표가 `0`인지 확인합니다.

   시작 위치가 가장 빠른 장애 복구인 새로운 Replicator를 생성하면 보조 클러스터의 주제에 있는 모든 데이터를 읽기 시작합니다. 데이터 보존 설정에 따라 소스 클러스터에서 가져온 데이터가 주제에 있을 수 있습니다. MSK Replicator에서 이러한 메시지를 자동으로 필터링하지만 보조 클러스터의 모든 데이터에 대한 데이터 처리 및 전송 요금이 계속 발생합니다. `ReplicatorBytesInPerSec`를 사용하여 Replicator에서 처리한 총 데이터를 추적할 수 있습니다. [MSK Replicator 지표](#)을(를) 참조하세요.

접두사 지정된 주제 이름 구성을 사용하는 경우 다음 단계를 따르세요.

보조 리전의 클러스터에서 기본 리전의 클러스터로 복제가 완료되고 Amazon CloudWatch의 MessageLag 지표가 0에 가까워진 후에만 장애 복구 단계를 시작해야 합니다. 계획된 페일백으로 인해 데이터가 손실되어서는 안 됩니다.

1. 보조 리전의 MSK 클러스터에 연결하는 모든 생산자와 소비자를 종료합니다.
2. 액티브-패시브 토폴로지의 경우 보조 리전의 클러스터에서 기본 리전으로 데이터를 복제하는 복제기를 삭제합니다. 액티브-액티브 토폴로지의 경우에는 복제기를 삭제할 필요가 없습니다.
3. 기본 리전의 MSK 클러스터에 연결하는 생산자를 시작합니다.
4. 애플리케이션의 메시지 정렬 요구 사항에 따라 다음 탭 중 하나의 단계를 따릅니다.

No message ordering

애플리케이션에 메시지 순서 지정이 필요하지 않은 경우 와일드카드 연산자(예: `topic`)를 사용하여 로컬(예:) 및 복제된 주제(예: `<sourceKafkaClusterAlias>.topic`) 모두에서 읽는 기본 AWS리전에서 소비자를 시작합니다`.*topic`. 로컬 주제(예: 주제)의 소비자는 장애 조치 전에 소비한 마지막 오프셋부터 다시 시작합니다. 장애 조치 이전부터 처리되지 않은 데이터가 있는 경우 이제 처리됩니다. 계획된 장애 조치의 경우 이러한 레코드가 없어야 합니다.

Message ordering

1. 기본 리전(예: `<sourceKafkaClusterAlias>.topic`)에서 복제된 주제에 대해서만 소비자를 시작하고 로컬 주제(예: `topic`)는 시작하지 않습니다.
2. 기본 리전의 클러스터에서 복제된 주제의 모든 소비자가 모든 데이터 처리를 완료하여 오프셋 지연이 0이 되고 처리된 레코드 수 또한 0이 될 때까지 기다립니다. 그런 다음 기본 리전의 클러스터에서 복제된 토픽에 대한 소비자를 중지합니다. 이 시점에서 장애 조치 후 보조 리전에서 생성된 모든 레코드는 기본 리전에서 소비됩니다.
3. 기본 지역의 클러스터에서 로컬 주제(예: `topic`)에 대한 소비자를 시작합니다.

5. 기본 리전의 클러스터에서 보조 리전의 클러스터로 기존 Replicator가 RUNNING 상태이고 `ReplicatorThroughput` 및 대기 시간 지표를 사용하여 예상대로 작동하는지 확인합니다.

MSK Replicator를 사용하여 액티브-액티브 설정 생성

두 MSK 클러스터가 모두 읽기 및 쓰기를 적극적으로 제공하는 액티브-액티브 설정을 생성하려면 접두사 지정된 주제 이름 복제(콘솔에서 주제 이름에 접두사 추가)가 포함된 MSK Replicator를 사용하는 것이 좋습니다. 하지만 이렇게 하려면 복제된 주제를 읽도록 소비자를 재구성해야 합니다.

다음 단계에 따라 소스 MSK 클러스터 A와 대상 MSK 클러스터 B 간에 액티브-액티브 토폴로지를 설정합니다.

1. MSK 클러스터 A를 소스로, MSK 클러스터 B를 대상으로 하는 MSK Replicator를 생성합니다.
2. 위의 MSK Replicator가 성공적으로 생성되면 클러스터 B를 소스로, 클러스터 A를 대상으로 복제기를 생성합니다.
3. 생산자와 같은 리전에 있는 클러스터의 로컬 주제(예: “주제”)에 각각 데이터를 동시에 기록하는 두 세트의 생산자를 생성합니다.
4. 소비자와 동일한 AWS리전에 있는 MSK 클러스터에서 와일드카드 구독(예: “.*topic”)을 사용하여 각각 데이터를 읽는 두 개의 소비자 세트를 생성합니다. 이렇게 하면 소비자가 로컬 주제(예: `topic`)에서 해당 리전에서 로컬로 생성된 데이터와 접두사(`<sourceKafkaClusterAlias>.topic`)가

있는 주제의 다른 리전에서 복제된 데이터를 자동으로 읽습니다. 이 두 소비자 세트는 서로 다른 소비자 그룹 ID를 가져야 MSK Replicator가 다른 클러스터로 복사할 때 소비자 그룹 오프셋이 덮어쓰지 않습니다.

클라이언트를 재구성하지 않으려면 접두사 지정된 주제 이름 복제(콘솔에서 주제 이름에 접두사 추가) 대신 동일한 주제 이름 복제(콘솔에서 동일한 주제 이름 유지)를 사용하여 MSK Replicator를 생성하여 액티브-액티브 설정을 생성할 수 있습니다. 하지만 각 Replicator에 대해 추가 데이터 처리 및 데이터 전송 요금을 지불하게 됩니다. 이는 각 Replicator가 일반적인 데이터 양을 두 번(복제를 위해 한 번, 무한 루프를 방지하기 위해 다시 한 번) 처리해야 하기 때문입니다. `ReplicatorBytesInPerSec` 지표를 사용하여 각 Replicator가 처리하는 총 데이터 양을 추적할 수 있습니다. 복제 모니터링을(를) 참조하세요. 이 지표에는 대상 클러스터에 복제된 데이터와 MSK Replicator에서 필터링한 데이터가 포함되어 데이터가 시작된 동일한 주제로 다시 복사되는 것을 방지합니다.

> **Note**
>
> 동일한 주제 이름 복제(콘솔에서 동일한 주제 이름 유지)를 사용하여 액티브-액티브 토폴로지를 설정하는 경우 주제를 삭제한 후 30초 이상 기다린 다음, 동일한 이름으로 주제를 다시 생성합니다. 이 대기 기간은 중복된 메시지가 소스 클러스터로 다시 복제되는 것을 방지하는 데 도움이 됩니다. 소비자는 다운스트림에 영향을 주지 않으면서 중복된 메시지를 재처리할 수 있어야 합니다. 다중 리전 Apache Kafka 애플리케이션 빌드의 고려 사항을(를) 참조하세요.

MSK Replicator를 사용하여 Amazon MSK 클러스터 간의 마이그레이션

클러스터 마이그레이션에 동일한 주제 이름 복제를 사용할 수 있지만 소비자는 다운스트림 영향 없이 중복 메시지를 처리할 수 있어야 합니다. MSK Replicator가 최소 한 번 복제를 제공하므로 드문 상황에서 중복 메시지가 발생할 수 있기 때문입니다. 소비자가 이 요구 사항을 충족하는 경우 다음 단계를 따르세요.

1. Replicator의 시작 위치를 가장 빠른 위치로 설정하고 동일한 주제 이름 복제를 사용하여 이전 클러스터의 데이터를 새 클러스터로 복제하는 Replicator를 생성합니다(콘솔에서 동일한 주제 이름 유지).
2. 새로운 클러스터에 대한 클러스터 수준 설정 및 권한을 구성합니다. MSK Replicator가 자동으로 복사하므로 주제 수준 설정과 '리터럴' 읽기 ACL을 구성할 필요가 없습니다.

3. Amazon CloudWatch의 `MessageLag` 지표가 0에 도달할 때까지 이 지표를 모니터링합니다. 0은 모든 데이터가 복제되었음을 나타냅니다.
4. 모든 데이터가 복제되면 생산자에서 이전 클러스터에 데이터를 쓰지 못하도록 합니다.
5. 새로운 클러스터에 연결하고 시작하도록 해당 생산자를 재구성합니다.
6. 이전 클러스터에서 데이터를 읽는 소비자의 `MaxOffsetLag` 지표가 `0`이 될 때까지 지표를 모니터링합니다. 이 값은 기존 데이터가 모두 처리되었음을 나타냅니다.
7. 이전 클러스터에 연결하는 소비자를 중지합니다.
8. 새로운 클러스터에 연결하고 시작하도록 소비자를 재구성합니다.

자체 관리형 MirrorMaker2에서 MSK Replicator로 마이그레이션

MirrorMaker(MM2)에서 MSK Replicator로 마이그레이션하려면 다음 단계를 따르세요.

1. 소스 Amazon MSK 클러스터에 쓰는 생산자를 중지합니다.
2. MM2가 소스 클러스터의 주제에 있는 모든 메시지를 복제하도록 허용합니다. 소스 MSK 클러스터에서 MM2 소비자의 소비자 지연을 모니터링하여 모든 데이터가 복제된 시기를 확인할 수 있습니다.
3. 시작 위치가 최신으로, 주제 이름 구성이 `IDENTICAL`(콘솔에서 동일한 주제 이름 복제)로 설정된 새로운 Replicator를 생성합니다.
4. Replicator가 RUNNING 상태가 되면 생산자가 소스 클러스터에 쓰기를 다시 시작할 수 있습니다.

MSK Replicator 문제 해결

다음 정보는 MSK Replicator에서 발생할 수 있는 문제를 해결하는 데 도움이 될 수 있습니다. 다른 Amazon MSK 기능에 대한 문제 해결 정보는 [Amazon MSK 클러스터 문제 해결](#) 단원을 참조하세요. 또한 [AWS re:Post](#)에 문제를 게시할 수 있습니다.

MSK Replicator 상태가 생성 중에서 실패로 변경됩니다.

다음은 MSK Replicator 생성 실패의 몇 가지 일반적인 원인입니다.

1. 대상 클러스터 섹션에서 복제기 생성을 위해 제공한 보안 그룹에 대상 클러스터의 보안 그룹에 대한 트래픽을 허용하는 아웃바운드 규칙이 있는지 확인합니다. 또한 대상 클러스터 섹션에서 복제기 생성을 위해 제공한 보안 그룹의 트래픽을 허용하는 인바운드 규칙이 대상 클러스터의 보안 그룹에 있는지 확인합니다. [대상 클러스터 선택](#)을(를) 참조하세요.

2. 리전 간 복제를 위해 복제기를 생성하는 경우 소스 클러스터에 IAM 액세스 제어 인증 방법에 대한 다중 VPC 연결이 활성화되어 있는지 확인합니다. 단일 리전에서의 Amazon MSK 다중 VPC 프라이빗 연결을(를) 참조하세요. 또한 소스 클러스터에 클러스터 정책이 설정되어 있는지 확인하여 MSK Replicator가 소스 클러스터에 연결할 수 있도록 합니다. Amazon MSK 소스 클러스터 준비을(를) 참조하세요.
3. MSK Replicator 생성 시 제공한 IAM 역할에 소스 및 대상 클러스터에 읽고 쓰는 데 필요한 권한이 있는지 확인합니다. 또한 IAM 역할에 주제에 대한 쓰기 권한이 있는지 확인합니다. 복제기 설정 및 권한 구성 섹션을 참조하세요
4. 네트워크 ACL이 MSK Replicator와 소스 및 대상 클러스터 간의 연결을 차단하지 않는지 확인합니다.
5. MSK Replicator가 연결을 시도할 때 소스 또는 대상 클러스터를 완전히 사용할 수 없을 수도 있습니다. 이는 과도한 부하, 디스크 사용량 또는 CPU 사용량으로 인해 복제기가 브로커에 연결할 수 없기 때문일 수 있습니다. 브로커 관련 문제를 해결하고 복제기 생성을 다시 시도합니다.

위의 검증을 수행한 후 MSK Replicator를 다시 생성합니다.

MSK Replicator가 생성 중 상태로 멈춤

때때로 MSK Replicator 생성에 최대 30분이 소요될 수 있습니다. 30분간 기다렸다가 복제기의 상태를 다시 확인합니다.

MSK Replicator가 데이터를 복제하지 않거나 일부 데이터만 복제합니다.

다음 단계에 따라 데이터 복제 문제를 해결합니다.

1. Amazon CloudWatch의 MSK Replicator에서 제공하는 AuthError 지표를 사용하여 Replicator에 인증 오류가 발생하지 않는지 확인합니다. 이 지표가 0을 초과하는 경우 복제기에 대해 제공한 IAM 역할의 정책이 유효하고 클러스터 권한에 설정된 거부 권한이 없는지 확인합니다. clusterAlias 차원을 기반으로 소스 또는 대상 클러스터에서 인증 오류가 발생하고 있는지 확인할 수 있습니다.
2. 원본 및 대상 클러스터에 문제가 없는지 확인합니다. 복제기가 소스 또는 대상 클러스터에 연결하지 못할 수 있습니다. 너무 많은 연결, 디스크의 최대 용량 또는 높은 CPU 사용량으로 인해 이러한 문제가 발생할 수 있습니다.
3. Amazon CloudWatch의 KafkaClusterPingSuccessCount 지표를 사용하여 소스 및 대상 클러스터가 MSK Replicator에서 연결 가능한지 확인합니다. clusterAlias 차원을 기반으로 소스 또는 대상 클러스터에서 인증 오류가 발생하고 있는지 확인할 수 있습니다. 이 값이 0이거나 데이터 포인트가 없으면 연결이 정상적이지 않은 것입니다. MSK Replicator가 클러스터 연결에 사용하는 네트워크 및 IAM 역할 권한을 확인해야 합니다.

4. Amazon CloudWatch의 ReplicatorFailure 지표를 사용하여 주제 수준 권한이 누락되어 Replicator에 장애가 발생하지 않는지 확인합니다. 이 지표가 0보다 크면 제공한 IAM 역할에서 주제 수준 권한을 확인합니다.
5. 복제기를 생성할 때 허용 목록에 제공한 정규 표현식이 복제하려는 주제의 이름과 일치하는지 확인합니다. 또한 거부 목록의 정규 표현식으로 인해 해당 주제가 복제에서 제외되고 있지 않은지 확인하세요.
6. Replicator가 대상 클러스터에서 새로운 주제 또는 주제 파티션을 감지하고 생성하는 데 최대 30초가 걸릴 수 있습니다. 대상 클러스터에서 주제가 생성되기 전에 소스 주제에 생성된 메시지는 Replicator 시작 위치가 최신(기본값)인 경우 복제되지 않습니다. 또는 대상 클러스터의 주제에 기존 메시지를 복제하려는 경우 소스 클러스터 주제 파티션의 가장 빠른 오프셋에서 복제를 시작할 수 있습니다. 복제기 설정 및 권한 구성을(를) 참조하세요.

대상 클러스터의 메시지 오프셋이 소스 클러스터와 다릅니다.

MSK Replicator는 데이터 복제의 일환으로 소스 클러스터의 메시지를 사용하여 대상 클러스터에 메시지를 생성합니다. 이로 인해 소스 클러스터와 대상 클러스터에 서로 다른 오프셋의 메시지가 발생할 수 있습니다. 하지만 Replicator 생성 중에 소비자 그룹 오프셋 동기화를 켠 경우 MSK Replicator는 메타데이터를 복사하는 동안 오프셋을 자동으로 변환하므로 대상 클러스터로 장애 조치한 후 소비자는 소스 클러스터에서 중단한 부분 근처에서 처리를 다시 시작할 수 있습니다.

MSK Replicator가 소비자 그룹 오프셋을 동기화하지 않고 있거나 대상 클러스터에 소비자 그룹이 없습니다.

다음 단계에 따라 메타데이터 복제 문제를 해결하세요.

1. 데이터 복제가 예상대로 작동하는지 확인합니다. 그렇지 않은 경우 MSK Replicator가 데이터를 복제하지 않거나 일부 데이터만 복제합니다.를 참조하세요.
2. Replicator를 생성할 때 허용 목록에 제공한 정규 표현식이 복제하려는 소비자 그룹의 이름과 일치하는지 확인합니다. 또한 거부 목록의 정규 표현식으로 인해 해당 주제가 복제에서 제외되고 있지 않은지 확인합니다.
3. MSK Replicator가 대상 클러스터에서 주제를 생성했는지 확인합니다. Replicator가 대상 클러스터에서 새로운 주제 또는 주제 파티션을 감지하고 생성하는 데 최대 30초가 걸릴 수 있습니다. 대상 클러스터에서 주제가 생성되기 전에 소스 주제에 생성된 메시지는 Replicator 시작 위치가 최신(기본값)인 경우 복제되지 않습니다. 소스 클러스터의 소비자 그룹이 MSK Replicator에서 복제하지 않은 메시지만 사용한 경우 소비자 그룹은 대상 클러스터에 복제되지 않습니다. 대상 클러스터에서 주제가 성공적으로 생성되면 MSK Replicator에서 소스 클러스터에 새로 작성된 메시지를 대상으로 복제

하기 시작합니다. 소비자 그룹이 소스에서 이러한 메시지를 읽기 시작하면 MSK Replicator에서 소비자 그룹을 대상 클러스터에 자동으로 복제합니다. 또는 대상 클러스터의 주제에 기존 메시지를 복제하려는 경우 소스 클러스터 주제 파티션의 가장 빠른 오프셋에서 복제를 시작할 수 있습니다. 복제기 설정 및 권한 구성을(를) 참조하세요.

> **Note**
>
> MSK Replicator는 주제 파티션 끝에 가까운 위치에서 읽는 소스 클러스터의 소비자에 대한 소비자 그룹 오프셋 동기화를 최적화합니다. 소비자 그룹이 소스 클러스터에서 지연되는 경우 소스에 비해 대상의 해당 소비자 그룹 지연이 더 높을 수 있습니다. 즉, 대상 클러스터로의 장애 조치 후 소비자는 더 많은 중복 메시지를 재처리합니다. 이 지연을 줄이려면 소스 클러스터의 소비자가 스트림의 팁(주제 파티션의 끝)을 파악하여 이 팁에서 소비를 시작해야 합니다. 소비자가 소비를 시작하면 MSK Replicator가 지연을 자동으로 줄입니다.

복제 지연 시간이 길거나 계속 증가하는 경우

다음은 복제 지연 시간이 길어지는 몇 가지 일반적인 원인입니다.

1. 소스 및 대상 MSK 클러스터에 적절한 수의 파티션이 있는지 확인합니다. 파티션이 너무 적거나 많으면 성능에 영향을 미칠 수 있습니다. 파티션 수 선택에 대한 지침은 MSK Replicator 사용 모범 사례 섹션을 참조하세요. 다음 표는 MSK Replicator로 원하는 처리량을 얻기 위해 권장되는 최소 파티션 수를 보여줍니다.

   처리량 및 권장되는 최소 파티션 수

처리량(MB/초)	필요한 최소 파티션 수
50	167
100	334
250	833
500	1666
1000	3333

2. 소스 및 대상 MSK 클러스터에 복제 트래픽을 지원할 수 있는 충분한 읽기 및 쓰기 용량이 있는지 확인합니다. MSK Replicator는 소스 클러스터(송신)의 소비자 역할과 대상 클러스터(수신)의 생산자 역할을 합니다. 따라서 클러스터의 다른 트래픽과 더불어 복제 트래픽을 지원할 수 있도록 클러스터 용량을 프로비저닝해야 합니다. MSK 클러스터 크기 조정에 대한 지침은 ??? 섹션을 참조하세요.

3. 복제 지연 시간은 클러스터가 서로 지리적으로 얼마나 멀리 떨어져 있는지에 따라 서로 다른 소스 및 대상 AWS리전 페어의 MSK 클러스터에 따라 다를 수 있습니다. 예를 들어 유럽(아일랜드) 및 유럽(런던) 리전의 클러스터 간 복제 시 일반적으로 유럽(아일랜드) 및 아시아 태평양(시드니) 리전의 클러스터 간 복제에 비해 복제 지연 시간이 더 짧습니다.

4. 소스 또는 대상 클러스터에 설정된 지나치게 공격적인 할당량으로 인해 복제기가 제한되고 있지는 않은지 확인합니다. Amazon CloudWatch에서 MSK Replicator가 제공하는 ThrottleTime 지표를 사용하여 소스/대상 클러스터의 브로커가 요청을 제한한 평균 시간(밀리초)을 확인할 수 있습니다. 이 지표가 0보다 크면 Kafka 쿼터를 조정하여 제한을 줄여 복제기가 따라잡을 수 있도록 해야 합니다. 복제기의 Kafka 할당량 관리에 대한 자세한 내용은 Kafka 할당량을 사용하여 MSK Replicator 처리량 관리 섹션을 참조하세요.

5. AWS리전 성능이 저하되면 ReplicationLatency 및 MessageLag가 증가할 수 있습니다. AWS 서비스 상태 대시보드를 사용하여 기본 MSK 클러스터가 있는 리전에서 MSK 서비스 이벤트를 확인합니다. 서비스 이벤트가 있는 경우 애플리케이션 읽기 및 쓰기를 다른 리전으로 일시적으로 리디렉션할 수 있습니다.

ReplicatorFailure 지표를 사용하여 MSK Replicator 실패 문제 해결

ReplicatorFailure 지표는 MSK Replicator에서 복제 문제를 모니터링하고 감지하는 데 도움이 됩니다. 이 지표 값이 0이 아닌 경우 일반적으로 복제 실패 문제를 나타내며, 이는 다음 요인으로 인해 발생할 수 있습니다.

- 메시지 크기 제한
- 타임스탬프 범위 위반
- 배치 크기 문제 기록

ReplicatorFailure 지표가 0이 아닌 값을 보고하는 경우 다음 단계에 따라 문제를 해결합니다.

> **Note**
> 이 지표에 대한 자세한 정보는 MSK Replicator 지표 섹션을 참조하세요.

1. 대상 MSK 클러스터에 연결 가능하고 Apache Kafka CLI 도구 설정이 있는 클라이언트를 구성합니다. 클라이언트 및 Kafka CLI 도구 설정에 대한 자세한 내용은 Amazon MSK 프로비저닝된 클러스터에 연결 섹션을 참조하세요.
2. https://console.aws.amazon.com/msk/home?region=us-east-1#/home/에서 Amazon MSK 콘솔을 엽니다.

   뒤이어 다음과 같이 하세요.

   a. MSK Replicator 및 대상 MSK 클러스터의 ARN을 가져옵니다.
   b. 대상 MSK 클러스터의 브로커 엔드포인트를 가져옵니다. 다음 단계에서 해당 엔드포인트를 사용합니다.
3. 다음 명령을 실행하여 이전 단계에서 얻은 MSK Replicator ARN 및 브로커 엔드포인트를 내보냅니다.

   다음 예제에서 사용되는 <*ReplicatorARN*>, <*BootstrapServerString*> 및 <*ConsumerConfigFile*>의 자리 표시자 값을 실제 값으로 바꿔야 합니다.

   ```
   export TARGET_CLUSTER_SERVER_STRING=<BootstrapServerString>
   ```

   ```
   export REPLICATOR_ARN=<ReplicatorARN>
   ```

   ```
   export CONSUMER_CONFIG_FILE=<ConsumerConfigFile>
   ```
4. <*path-to-your-kafka-installation*>/bin 디렉터리에서 다음을 수행합니다.

   a. 다음 스크립트를 저장하고 이름을 **query-replicator-failure-message.sh**로 지정합니다.

   ```
   #!/bin/bash

   # Script: Query MSK Replicator Failure Message
   # Description: This script queries exceptions fromAWSMSK Replicator status
    topics
   # It takes a replicator ARN and bootstrap server as input and searches for
    replicator exceptions
   # in the replicator's status topic, formatting and displaying them in a
    readable manner
   #
   # Required Arguments:
   ```

```
#   --replicator-arn: The ARN of the AWS MSK Replicator
#   --bootstrap-server: The Kafka bootstrap server to connect to
#   --consumer.config: Consumer config properties file
# Usage Example:
#   ./query-replicator-failure-message.sh ./query-replicator-failure-message.sh
 --replicator-arn <replicator-arn> --bootstrap-server <bootstrap-server> --
consumer.config <consumer.config>

print_usage() {
  echo "USAGE: $0 ./query-replicator-failure-message.sh --replicator-arn
 <replicator-arn> --bootstrap-server <bootstrap-server> --consumer.config
 <consumer.config>"
  echo "--replicator-arn <String: MSK Replicator ARN>      REQUIRED: The ARN of
 AWS MSK Replicator."
  echo "--bootstrap-server <String: server to connect to>  REQUIRED: The Kafka
 server to connect to."
  echo "--consumer.config <String: config file>            REQUIRED: Consumer
 config properties file."
  exit 1
}

# Initialize variables
replicator_arn=""
bootstrap_server=""
consumer_config=""

# Parse arguments
while [[ $# -gt 0 ]]; do
  case "$1" in
    --replicator-arn)
      if [ -z "$2" ]; then
        echo "Error: --replicator-arn requires an argument."
        print_usage
      fi
      replicator_arn="$2"; shift 2 ;;
    --bootstrap-server)
      if [ -z "$2" ]; then
        echo "Error: --bootstrap-server requires an argument."
        print_usage
      fi
      bootstrap_server="$2"; shift 2 ;;
    --consumer.config)
      if [ -z "$2" ]; then
        echo "Error: --consumer.config requires an argument."
```

```
        print_usage
      fi
      consumer_config="$2"; shift 2 ;;
    *) echo "Unknown option: $1"; print_usage ;;
  esac
done

# Check for required arguments
if [ -z "$replicator_arn" ] || [ -z "$bootstrap_server" ] || [ -z
 "$consumer_config" ]; then
  echo "Error: --replicator-arn, --bootstrap-server, and --consumer.config are
 required."
  print_usage
fi

# Extract replicator name and suffix from ARN
replicator_arn_suffix=$(echo "$replicator_arn" | awk -F'/' '{print $NF}')
replicator_name=$(echo "$replicator_arn" | awk -F'/' '{print $(NF-1)}')
echo "Replicator name: $replicator_name"

# List topics and find the status topic
topics=$(./kafka-topics.sh --command-config client.properties --list --
bootstrap-server "$bootstrap_server")
status_topic_name="__amazon_msk_replicator_status_${replicator_name}_
${replicator_arn_suffix}"

# Check if the status topic exists
if echo "$topics" | grep -Fq "$status_topic_name"; then
  echo "Found replicator status topic: '$status_topic_name'"
  ./kafka-console-consumer.sh --bootstrap-server "$bootstrap_server" --
consumer.config "$consumer_config" --topic "$status_topic_name" --from-
beginning | stdbuf -oL grep "Exception" | stdbuf -oL sed -n 's/.*Exception:\(.*
\) Topic: \([^,]*\), Partition: \([^\]*\).*/ReplicatorException:\1 Topic: \2,
 Partition: \3/p'
else
  echo "No topic matching the pattern '$status_topic_name' found."
fi
```

b. 이 스크립트를 실행하여 MSK Replicator 실패 메시지를 쿼리합니다.

```
<path-to-your-kafka-installation>/bin/query-replicator-failure-message.sh --
replicator-arn $REPLICATOR_ARN --bootstrap-server $TARGET_CLUSTER_SERVER_STRING
 --consumer.config $CONSUMER_CONFIG_FILE
```

이 스크립트는 예외 메시지 및 영향을 받는 토픽 파티션과 함께 모든 오류를 출력합니다. 일반적인 MSK Replicator 장애 및 해당 솔루션에 설명된 대로 해당 예외 정보를 사용하여 실패를 완화할 수 있습니다. 토픽에 모든 과거 실패 메시지가 포함되어 있으므로 마지막 메시지를 사용하여 조사를 시작합니다. 다음은 실패 메시지의 예입니다.

```
ReplicatorException: The request included a message larger than the max message
 size the server will accept. Topic: test, Partition: 1
```

일반적인 MSK Replicator 장애 및 해당 솔루션

다음 목록은 발생할 수 있는 일부 MSK Replicator 장애와 이를 완화하는 방법을 설명합니다.

max.request.size보다 큰 메시지 크기

원인

이 실패는 개별 메시지 크기가 10MB를 초과하기 때문에 MSK Replicator가 데이터를 복제하지 못할 때 발생합니다. 기본적으로 MSK Replicator는 최대 10MB 크기의 메시지를 복제합니다.

다음은 오류 메시지 유형의 예입니다.

```
ReplicatorException: The message is 20635370 bytes when serialized which is larger
 than 10485760, which is the value of the max.request.size configuration. Topic:
 test, Partition: 1
```

Solution

토픽의 개별 메시지 크기를 줄입니다. 이렇게 할 수 없는 경우 다음 지침에 따라 한도 증가를 요청하세요.

서버가 수락할 최대 메시지 크기보다 큰 메시지 크기

원인

이 실패는 메시지 크기가 대상 클러스터의 최대 메시지 크기를 초과할 때 발생합니다.

다음은 오류 메시지 유형의 예입니다.

```
ReplicatorException: The request included a message larger than the max message size
 the server will accept. Topic: test, Partition: 1
```

Solution

대상 클러스터 또는 해당 대상 클러스터 토픽의 `max.message.bytes` 구성을 증가시킵니다. 압축되지 않은 최대 메시지 크기와 일치하도록 대상 클러스터의 `max.message.bytes` 구성을 설정합니다. 이에 대한 자세한 내용은 max.message.bytes를 참조하세요.

범위를 벗어난 타임스탬프

원인

이 실패는 개별 메시지 타임스탬프가 대상 클러스터의 허용 범위를 벗어나기 때문에 발생합니다.

다음은 오류 메시지 유형의 예입니다.

```
ReplicatorException: Timestamp 1730137653724 of message with offset 0 is out of
 range. The timestamp should be within [1730137892239, 1731347492239] Topic: test,
 Partition: 1
```

Solution

이전 타임스탬프가 있는 메시지를 허용하도록 대상 클러스터의 `message.timestamp.before.max.ms` 구성을 업데이트합니다. 이에 대한 자세한 내용은 message.timestamp.before.max.ms를 참조하세요.

레코드 배치가 너무 큼

원인

이 실패는 레코드 배치 크기가 대상 클러스터의 토픽에 대해 설정된 세그먼트 크기를 초과하기 때문에 발생합니다. MSK Replicator는 최대 배치 크기 1MB를 지원합니다.

다음은 오류 메시지 유형의 예입니다.

```
ReplicatorException: The request included message batch larger than the configured
 segment size on the server. Topic: test, Partition: 1
```

Solution

Replicator가 오류 없이 진행되려면 대상 클러스터의 segment.bytes 구성이 배치 크기(1MB) 이상이어야 합니다. 대상 클러스터의 segment.bytes를 1048576(1MB) 이상으로 업데이트합니다. 이에 대한 자세한 내용은 segment.bytes를 참조하세요.

> **Note**
>
> 해당 솔루션을 적용한 후에도 ReplicatorFailure 지표가 0이 아닌 값을 계속 내보내는 경우 지표가 0의 값을 내보낼 때까지 문제 해결 프로세스를 반복합니다.

MSK Replicator 사용 모범 사례

이 섹션에서는 Amazon MSK Replicator 사용에 대한 일반적인 모범 사례와 구현 전략을 다룹니다.

주제

- Kafka 할당량을 사용하여 MSK Replicator 처리량 관리
- 클러스터 보존 기간 설정

Kafka 할당량을 사용하여 MSK Replicator 처리량 관리

MSK Replicator는 소스 클러스터의 소비자 역할을 하므로 복제로 인해 소스 클러스터에서 다른 소비자가 제한될 수 있습니다. 제한의 양은 원본 클러스터의 읽기 용량과 복제 중인 데이터의 처리량에 따라 달라집니다. 소스 클러스터와 대상 클러스터에 동일한 용량을 프로비저닝하고 필요한 용량을 계산할 때 복제 처리량을 고려하는 것이 좋습니다.

또한 소스 및 대상 클러스터에서 복제기에 대한 Kafka 할당량을 설정하여 MSK Replicator가 사용할 수 있는 용량을 제어할 수 있습니다. 네트워크 대역폭 할당량을 사용하는 것을 권장합니다. 네트워크 대역폭 할당량은 할당량을 공유하는 하나 이상의 클라이언트에 대해 초당 바이트로 정의되는 바이트 속도 임계값을 정의합니다. 이 할당량은 브로커별로 정의됩니다.

할당량을 적용하려면 다음 단계를 수행합니다.

1. 소스 클러스터의 부트스트랩 서버 문자열을 검색합니다. Amazon MSK 클러스터를 위한 부트스트랩 브로커 가져오기을(를) 참조하세요.
2. MSK Replicator에서 사용하는 서비스 실행 역할(SER)을 검색합니다. `CreateReplicator` 요청에 사용한 SER입니다. 기존 복제기의 DescribeReplicator 응답에서 SER을 가져올 수도 있습니다.
3. Kafka CLI 도구를 사용하여 소스 클러스터에 대해 다음 명령을 실행합니다.

```
./kafka-configs.sh --bootstrap-server <source-cluster-bootstrap-server> --alter --
add-config 'consumer_byte_
```

```
rate=<quota_in_bytes_per_second>' --entity-type users --entity-name
 arn:aws:sts::<customer-account-id>:assumed-role/<ser-role-name>/<customer-account-
id> --command-config <client-properties-for-iam-auth></programlisting>
```

4. 위 명령을 실행한 후 `ReplicatorThroughput` 지표가 설정한 할당량을 초과하지 않는지 확인합니다.

여러 MSK Replicator 간에 서비스 실행 역할을 다시 사용하는 경우 모두 이 할당량이 적용된다는 점에 유의하세요. 복제기별로 별도의 할당량을 유지하려면 별도의 서비스 실행 역할을 사용하세요.

할당량과 함께 MSK IAM 인증을 사용하는 방법에 대한 자세한 내용은 [IAM 액세스 제어 및 Kafka 할당량이 있는 Amazon MSK의 멀티 테넌시 Apache Kafka 클러스터 - 1부]를 참조하세요.

> **Warning**
>
> Consumer_byte_rate를 매우 낮게 설정하면 MSK Replicator가 예상치 못한 방식으로 작동할 수 있습니다.

클러스터 보존 기간 설정

MSK 프로비저닝 및 서버리스 클러스터에 대한 로그 보존 기간을 설정할 수 있습니다. 권장 보존 기간은 7일입니다. [클러스터 구성 변경 사항] 또는 [지원되는 MSK Serverless 클러스터 구성]을 참조하세요.

MSK 통합

이 섹션에서는 Amazon MSK와 통합되는 AWS기능에 대한 참조를 제공합니다.

주제

Amazon MSK용 Amazon Athena 커넥터

Amazon MSK를 위한 Amazon Athena 커넥터를 사용하면 Amazon Athena가 Apache Kafka 주제에서 SQL 쿼리를 실행할 수 있습니다. 이 커넥터를 사용하여 Athena에서 Apache Kafka 주제를 테이블로, 메시지를 행으로 볼 수 있습니다.

자세한 내용은 Amazon Athena 사용 설명서에서 Amazon Athena MSK 커넥터를 참조하세요.

Amazon MSK를 위한 Amazon Redshift 스트리밍 데이터 수집

Amazon Redshift는 Amazon MSK의 스트리밍 모으기를 지원합니다. Amazon Redshift 스트리밍 모으기 기능은 Amazon MSK의 스트리밍 데이터를 Amazon Redshift 구체화된 뷰로 짧은 지연 시간과 고속 수집을 제공합니다. Amazon S3에서 데이터를 스테이징할 필요가 없기 때문에 Amazon Redshift는 지연 시간을 줄이고 스토리지 비용을 절감하면서 스트리밍 데이터를 모을 수 있습니다. SQL 문을 사용하여 Amazon MSK 토픽을 인증하고 연결하기 위해 Amazon Redshift 클러스터에서 Amazon Redshift 스트리밍 모으기를 구성할 수 있습니다.

자세한 내용은 Amazon Redshift 데이터베이스 개발자 설명서에서 스트리밍 모으기를 참조하세요.

Amazon MSK를 위한 Firehose 통합

Amazon MSK는 Firehose와 통합하여 서버리스, 코드 없는 솔루션을 제공하여 Apache Kafka 클러스터에서 Amazon S3 데이터 레이크로 스트림을 전송합니다. Firehose는 스트리밍 추출, 전환, 적재(ETL)

서비스로, Amazon MSK Kafka 주제에서 데이터를 읽고, Parquet으로 변환하는 등의 전환을 수행하며, 데이터를 집계하여 Amazon S3에 작성합니다. 콘솔에서 몇 번의 클릭만으로 Kafka 주제에서 읽고 S3 위치로 전송하도록 Firehose 스트림을 설정할 수 있습니다. 작성할 코드, 커넥터 애플리케이션, 프로비저닝할 리소스가 없습니다. Firehose는 Kafka 주제에 게시된 데이터의 양에 따라 자동으로 확장되며 사용자는 Kafka에서 수집한 바이트에 대해서만 비용을 지불합니다.

이 기능에 대한 자세한 내용은 다음을 참조하세요.

- Amazon Data Firehose 개발자 안내서의 Amazon MSK를 사용하여 Kinesis Data Firehose에 쓰기 - Amazon Kinesis Data Firehose
- 블로그: Amazon MSK, Apache Kafka에서 데이터 레이크까지 관리형 데이터 전송 도입
- 랩: Firehose를 사용하여 Amazon S3로 전달

AWS LambdaAmazon MSK와 통합

Lambda 통합은 Event Poller라는 리소스를 사용하여 주제의 메시지를 지속적으로 폴링하는 이벤트 소스 매핑(ESM)을 사용하여 Amazon MSK 클러스터를 선택한 Lambda 함수에 연결합니다. ESM은 주제의 모든 파티션에 대해 OffsetLag 지표를 사용하여 메시지 백로그를 평가하고 이벤트 폴러를 자동 조정하여 메시지를 효율적으로 처리합니다.

자세한 내용은 AWS Lambda개발자 안내서의 Amazon MSK에서 Lambda 사용을 참조하세요.

Amazon MSK 콘솔을 통해 Amazon EventBridge 파이프에 액세스

Amazon EventBridge 파이프는 소스를 대상에 연결합니다. 파이프는 지원되는 소스와 대상 간의 지점 간 통합을 위한 것으로 고급 변환 및 강화를 지원합니다. EventBridge 파이프는 Amazon MSK 클러스터를 Step Functions, Amazon SQS, API 게이트웨이와 같은 AWS 서비스뿐만 아니라 Salesforce와 같은 타사 서비스형 소프트웨어(SaaS) 애플리케이션에 연결할 수 있는 확장성이 뛰어난 방법을 제공합니다.

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

Amazon MSK 클러스터의 세부 정보 페이지에서 해당 클러스터를 소스로 사용하는 파이프를 볼 수 있습니다. 다음을 수행할 수도 있습니다.

- EventBridge 콘솔을 실행하여 파이프 세부 정보를 확인합니다.

- EventBridge 콘솔을 실행하여 클러스터가 소스인 새 파이프를 생성합니다.

Amazon MSK 클러스터를 파이프 소스로 구성하는 방법에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 소스로서의 Amazon Managed Streaming for Apache Kafka 클러스터를 참조하세요. 일반적인 EventBridge 파이프에 대한 자세한 내용은 EventBridge 파이프를 참조하세요.

지정된 Amazon MSK 클러스터에 대한 EventBridge 파이프에 액세스하려면 다음을 수행합니다.

1. Amazon MSK 콘솔을 열고 클러스터를 선택합니다.
2. 클러스터를 선택합니다.
3. 클러스터 세부 정보 페이지에서 통합 탭을 선택합니다.

   통합 탭에는 선택한 클러스터를 소스로 사용하도록 현재 구성된 모든 파이프 목록이 포함되어 있습니다.

   - 파이프 이름
   - 현재 상태
   - 파이프 대상
   - 파이프를 마지막으로 수정한 시점
4. 원하는 대로 Amazon MSK 클러스터의 파이프를 관리합니다.

   파이프에 대한 추가 세부 정보에 액세스하려면 다음을 수행합니다.

   - 파이프를 선택합니다.

     그러면 EventBridge 콘솔의 파이프 세부 정보 페이지가 시작됩니다.

   새 파이프를 생성하려면 다음을 수행합니다.

   - Amazon MSK 클러스터를 파이프에 연결을 선택합니다.

     그러면 Amazon MSK 클러스터가 파이프 소스로 지정된 EventBridge 콘솔의 파이프 생성 페이지가 시작됩니다. 자세한 내용은 Amazon EventBridge 사용 설명서에서 EventBridge 파이프 생성을 참조하세요.

   - 클러스터 페이지에서 클러스터에 대한 파이프를 생성할 수도 있습니다. 클러스터를 선택하고 작업 메뉴에서 EventBridge 파이프 생성을 선택합니다.

MSK Express 브로커 및 MSK Serverless를 통해 Kafka Streams 사용

Kafka Streams는 상태 비저장 및 상태 저장 변환을 지원합니다. 개수, 집계 또는 조인과 같은 상태 저장 변환은 내부 Kafka 주제에 상태를 저장하는 연산자를 사용합니다. 또한 groupBy 또는 재분할과 같은 일부 상태 비저장 변환은 결과를 내부 Kafka 주제에 저장합니다. 기본적으로 Kafka Streams는 해당 연산자를 기반으로 이러한 내부 주제의 이름을 지정합니다. 해당하는 주제가 없는 경우 Kafka Streams는 내부 Kafka 주제를 생성합니다. 내부 주제 생성을 위해 Kafka Streams는 segment.bytes 구성을 하드코딩하고 이를 50MB로 설정합니다. Express 브로커 및 MSK Serverless를 사용하는 MSK Provisioned는 주제 생성 중에 segment.size를 포함한 일부 주제 구성을 보호합니다. 따라서 상태 저장 변환을 지원하는 Kafka Streams 애플리케이션은 MSK Express 브로커 또는 MSK Serverless를 사용하여 내부 주제를 생성하지 못합니다.

MSK Express 브로커 또는 MSK Serverless에서 이러한 Kafka Streams 애플리케이션을 실행하려면 내부 주제를 직접 생성해야 합니다. 이를 수행하려면 먼저 주제가 필요한 Kafka Streams 연산자를 식별하고 이름을 지정합니다. 그런 다음 해당 내부 Kafka 주제를 생성합니다.

> **Note**
>
> - Kafka Streams에서 연산자, 특히 내부 주제를 사용하는 연산자의 이름을 수동으로 지정하는 것이 가장 좋습니다. 이름 지정 연산자에 대한 자세한 내용은 Kafka Streams 설명서의 Kafka Streams DSL 애플리케이션의 이름 지정 연산자를 참조하세요.
> - 상태 저장 변환의 내부 주제 이름은 Kafka Streams 애플리케이션의 `application.id` 및 상태 저장 연산자 `application.id-statefuloperator_name`의 이름에 따라 달라집니다.

주제

- MSK Express 브로커 또는 MSK Serverless를 사용하여 Kafka Streams 애플리케이션 생성

MSK Express 브로커 또는 MSK Serverless를 사용하여 Kafka Streams 애플리케이션 생성

Kafka Streams 애플리케이션의 `application.id`가 `msk-streams-processing`으로 설정된 경우 MSK Express 브로커 또는 MSK Serverless를 사용하여 Kafka Streams 애플리케이션을 생성할 수 있

습니다. 이를 수행하려면 이름이 있는 내부 주제가 필요한 `count()` 연산자를 사용합니다. 예를 들어 `msk-streams-processing-count-store`입니다.

Kafka Streams 애플리케이션을 생성하려면 다음을 수행합니다.

주제

- 연산자 식별 및 이름 지정
- 내부 주제 생성
- (선택 사항) 주제 이름 확인
- 이름 지정 연산자의 예제

연산자 식별 및 이름 지정

1. Kafka Streams 설명서의 상태 저장 변환을 사용하여 상태 저장 프로세서를 식별합니다.

   상태 저장 프로세서의 몇 가지 예로는 `count`, `aggregate` 또는 `join`이 있습니다.
2. 재분할을 위한 주제를 생성하는 프로세서를 식별합니다.

   다음 예제에는 상태가 필요한 `count()` 연산자가 포함되어 있습니다.

   ```
   var stream =
       paragraphStream
           .groupByKey()
           .count()
           .toStream();
   ```
3. 주제 이름을 지정하려면 각 상태 저장 프로세서의 이름을 추가합니다. 프로세서 유형에 따라 서로 다른 이름 지정 클래스에서 이름이 지정됩니다. 예를 들어 `count()` 연산자는 집계 작업입니다. 따라서 `Materialized` 클래스가 필요합니다.

   상태 저장 작업의 이름 지정 클래스에 대한 자세한 내용은 Kafka Streams 설명서의 결론을 참조하세요.

   다음 예제에서는 `Materialized` 클래스를 사용하여 `count()` 연산자의 이름을 `count-store`로 설정합니다.

   ```
   var stream =
       paragraphStream
           .groupByKey()
   ```

```
        .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("count-
store")  // descriptive name for the store
            .withKeySerde(Serdes.String())
            .withValueSerde(Serdes.Long()))
        .toStream();
```

내부 주제 생성

Kafka Streams는 내부 주제 이름에 `application.id` 접두사를 붙입니다. 여기서 `application.id`는 사용자 정의됩니다. 예를 들어 `application.id-internal_topic_name`입니다. 내부 주제는 일반적인 Kafka 주제이며, Kafka API의 [Apache Kafka 주제 생성](#) 또는 `AdminClient`에서 제공되는 정보를 사용하여 주제를 생성할 수 있습니다.

사용 사례에 따라 Kafka Streams의 기본 정리 및 보존 정책을 사용하거나 해당 값을 사용자 지정할 수 있습니다. 해당 내용은 `cleanup.policy` 및 `retention.ms`에서 정의합니다.

다음 예제에서는 `AdminClient` API를 사용하여 주제를 생성하고 `application.id`를 **`msk-streams-processing`**으로 설정합니다.

```
try (AdminClient client = AdminClient.create(configs.kafkaProps())) {
    Collection<NewTopic> topics = new HashSet<>();
    topics.add(new NewTopic("msk-streams-processing-count-store", 3, (short) 3));
    client.createTopics(topics);
}
```

클러스터에 주제가 생성되면 Kafka Streams 애플리케이션이 `count()` 연산자에 `msk-streams-processing-count-store` 주제를 사용할 수 있습니다.

(선택 사항) 주제 이름 확인

토폴로지 설명자를 사용하여 스트림의 토폴로지를 설명하고 내부 주제의 이름을 볼 수 있습니다. 다음 예제에서는 토폴로지 설명자를 실행하는 방법을 보여줍니다.

```
final StreamsBuilder builder = new StreamsBuilder();
Topology topology = builder.build();
System.out.println(topology.describe());
```

다음 출력은 이전 예제에 대한 스트림의 토폴로지를 보여줍니다.

```
Topology Description:
```

```
Topologies:
   Sub-topology: 0
    Source: KSTREAM-SOURCE-0000000000 (topics: [input_topic])
      --> KSTREAM-AGGREGATE-0000000001
    Processor: KSTREAM-AGGREGATE-0000000001 (stores: [count-store])
      --> KTABLE-TOSTREAM-0000000002
      <-- KSTREAM-SOURCE-0000000000
    Processor: KTABLE-TOSTREAM-0000000002 (stores: [])
      --> KSTREAM-SINK-0000000003
      <-- KSTREAM-AGGREGATE-0000000001
    Sink: KSTREAM-SINK-0000000003 (topic: output_topic)
      <-- KTABLE-TOSTREAM-0000000002
```

토폴로지 설명자를 사용하는 방법에 대한 자세한 내용은 Kafka Streams 설명서의 Kafka Streams DSL 애플리케이션의 이름 지정 연산자를 참조하세요.

이름 지정 연산자의 예제

이 섹션에서는 이러한 이름 지정 연산자의 예제를 제공합니다.

groupByKey()에 대한 이름 지정 연산자의 예제

```
groupByKey() -> groupByKey(Grouped.as("kafka-stream-groupby"))
```

normal count()에 대한 이름 지정 연산자의 예제

```
normal count() -> .count(Materialized.<String, Long, KeyValueStore<Bytes,
 byte[]>>as("kafka-streams-window")  // descriptive name for the store
    .withKeySerde(Serdes.String())
    .withValueSerde(Serdes.Long()))
```

windowed count()에 대한 이름 지정 연산자의 예제

```
windowed count() -> .count(Materialized.<String, Long, WindowStore<Bytes,
 byte[]>>as("kafka-streams-window")  // descriptive name for the store
    .withKeySerde(Serdes.String())
    .withValueSerde(Serdes.Long()))
```

windowed suppressed에 대한 이름 지정 연산자의 예제

```
windowed suppressed() ->
Suppressed<Windowed> suppressed = Suppressed
```

```
        .untilWindowCloses(Suppressed.BufferConfig.unbounded())
        .withName("kafka-suppressed");
    .suppress(suppressed)
```

실시간 벡터 임베딩 블루프린트

Amazon Managed Streaming for Apache Kafka(Amazon MSK)는 Amazon Managed Service for Apache Flink 블루프린트를 지원하여 Amazon Bedrock을 사용하여 벡터 임베딩을 생성함으로써 최신 상태 및 컨텍스트 데이터로 구동되는 실시간 AI 애플리케이션을 구축하는 프로세스를 간소화합니다. MSF 블루프린트는 Amazon MSK 스트리밍 파이프라인의 최신 데이터를 생성형 AI 모델에 통합하는 프로세스를 간소화하므로 실시간 데이터 스트림, 벡터 데이터베이스 및 대규모 언어 모델을 통합하기 위해 사용자 지정 코드를 작성할 필요가 없습니다.

Bedrock의 임베딩 모델을 사용하여 벡터 임베딩을 지속적으로 생성하도록 MSF 블루프린트를 구성한 다음 Amazon MSK 데이터 스트림에 대해 OpenSearch Service에서 해당 임베딩을 인덱싱할 수 있습니다. 이를 통해 실시간 데이터의 컨텍스트를 Bedrock의 강력한 대규모 언어 모델과 결합하여 사용자 지정 코드를 작성하지 않고도 정확한 최신 상태의 AI 응답을 생성할 수 있습니다. 또한 모델 수집을 위한 고품질 입력을 지원하는 오픈 소스 라이브러리인 LangChain의 데이터 청킹 기술에 대한 기본 지원을 사용하여 데이터 검색의 효율성을 개선하도록 선택할 수 있습니다. 블루프린트는 MSK, 선택한 임베딩 모델 및 OpenSearch 벡터 스토어 간의 데이터 통합 및 처리를 관리하므로 기본 통합을 관리하는 대신 AI 애플리케이션을 구축하는 데 집중할 수 있습니다.

실시간 벡터 임베딩 블루프린트는 다음 AWS리전에서 사용할 수 있습니다.

- 버지니아 북부 - us-east-1
- 오하이오 - us-east-2
- 오리건 - us-west-2
- 뭄바이 - ap-south-1
- 서울 - ap-northeast-2
- 싱가포르 - ap-southeast-1
- 시드니 - ap-southeast-2
- 도쿄 - ap-northeast-1
- 캐나다 중부 - ca-central-1
- 프랑크푸르트 eu - central-1
- 아일랜드 - eu-west-1
- 런던 - eu-west-2

- 파리 - eu-west-3
- 상파울루 - sa-east-1

주제

로깅 및 관찰성

CloudWatch 로그를 사용하여 실시간 벡터 임베딩 블루프린트에 대한 모든 로그 및 지표를 활성화할 수 있습니다.

일반 MSF 애플리케이션 및 Amazon Bedrock에 사용할 수 있는 모든 지표는 애플리케이션 및 Bedrock 지표를 모니터링할 수 있습니다.

임베딩 생성 성능을 모니터링하기 위한 두 가지 추가 지표가 있습니다. 이러한 지표는 CloudWatch의 EmbeddingGeneration 작업 이름의 일부입니다.

- BedrockTitanEmbeddingTokenCount: Bedrock에 대한 단일 요청에 있는 토큰 수를 모니터링합니다.
- BedrockEmbeddingGenerationLatencyMs: Bedrock에서 임베딩을 생성하기 위한 응답을 보내고 받는 데 걸린 시간을 밀리초 단위로 보고합니다.

OpenSearch 서비스의 경우 다음 지표를 사용할 수 있습니다.

- OpenSearch Serverless 컬렉션 지표: Amazon OpenSearch Service 개발자 안내서의 Amazon CloudWatch를 사용하여 OpenSearch Serverless 모니터링을 참조하세요.
- OpenSearch 프로비저닝된 지표: Amazon OpenSearch Service 개발자 안내서의 Amazon CloudWatch를 사용하여 OpenSearch 클러스터 지표 모니터링을 참조하세요.

실시간 벡터 임베딩 블루프린트 활성화 전 참고 사항

Managed Service for Apache Flink 애플리케이션은 입력 스트림의 비정형 텍스트 또는 JSON 데이터만 지원합니다.

두 종류의 입력 처리 모드가 지원됩니다.

- 입력 데이터가 비정형 텍스트인 경우 전체 텍스트 메시지가 포함됩니다. 벡터 DB에는 원본 텍스트와 생성된 임베딩이 포함됩니다.
- 입력 데이터가 JSON 형식인 경우 애플리케이션은 임베딩 프로세스에 사용할 JSON 객체 값 내에서 하나 이상의 키를 구성하고 지정할 수 있는 기능을 제공합니다. 키가 두 개 이상인 경우 모든 키가 함께 벡터화되고 벡터 DB에서 인덱싱됩니다. 벡터 DB에는 원본 메시지와 생성된 임베딩이 포함됩니다.

임베딩 생성: 애플리케이션이 Bedrock에서 독점적으로 제공하는 모든 텍스트 임베딩 모델을 지원합니다.

벡터 DB 스토어에 유지: 애플리케이션이 고객 계정의 기존 OpenSearch 클러스터(프로비저닝 또는 서버리스)를 임베디드 데이터를 유지하기 위한 대상으로 사용합니다. Opensearch Serverless를 사용하여 벡터 인덱스를 생성할 때는 항상 벡터 필드 이름 `embedded_data`를 사용합니다.

MSF 블루프린트와 마찬가지로 인프라를 관리하여 실시간 벡터 임베딩 블루프린트와 연결된 코드를 실행해야 합니다.

MSF 블루프린트와 마찬가지로 MSF 애플리케이션이 생성되면 콘솔 또는 CLI를 사용하여 AWS계정에서만 시작해야 합니다. AWS는 MSF 애플리케이션을 시작하지 않습니다. StartApplication API(CLI 또는 콘솔을 통해)를 직접 호출하여 애플리케이션을 실행해야 합니다.

데이터의 교차 계정 이동: 애플리케이션은 입력 스트림과 다른 AWS계정에 있는 벡터 대상 간에 데이터를 이동하도록 허용하지 않습니다.

스트리밍 데이터 벡터화 블루프린트 배포

이 주제에서는 스트리밍 데이터 벡터화 블루프린트를 배포하는 방법을 설명합니다.

스트리밍 데이터 벡터화 블루프린트 배포

1. 다음 리소스가 올바르게 설정되었는지 확인합니다.
   - 데이터가 포함된 하나 이상의 토픽으로 프로비저닝된 또는 서버리스 MSK 클러스터입니다.
2. Bedrock Setup: [원하는 Bedrock 모델에 액세스합니다](). 현재 지원되는 Bedrock 모델은 다음과 같습니다.
   - Amazon Titan Embeddings G1 - Text
   - Amazon Titan Text Embeddings V2
   - Amazon Titan Multimodal Embeddings G1

- Cohere Embed English
- Cohere Embed Multilingual

3. AWSOpenSearch 컬렉션:

   - 프로비저닝된 또는 Serverless OpenSearch Service 컬렉션을 사용할 수 있습니다.
   - OpenSearch Service 컬렉션에는 인덱스가 하나 이상 있어야 합니다.
   - OpenSearch Serverless 컬렉션을 사용하려는 경우 벡터 검색 컬렉션을 생성해야 합니다. 벡터 인덱스를 설정하는 방법에 대한 자세한 내용은 지식 기반에 대한 자체 벡터 저장소의 사전 조건을 참조하세요. 벡터화에 대한 자세한 내용은 설명된 Amazon OpenSearch Service의 벡터 데이터베이스 기능을 참조하세요.

     > Note
     >
     > 벡터 인덱스를 생성할 때 벡터 필드 이름 `embedded_data`를 사용해야 합니다.

   - OpenSearch 프로비저닝된 컬렉션을 사용하려는 경우 블루프린트에서 생성된 MSF 애플리케이션 역할(Opensearch 액세스 정책 포함)을 OpenSearch 컬렉션에 마스터 사용자로 추가해야 합니다. 또한 OpenSearch의 액세스 정책이 '허용' 작업으로 설정되어 있는지 확인합니다. 이는 세분화된 액세스 제어를 활성화하는 데 필요합니다.
   - 선택적으로 OpenSearch 대시보드에 대한 액세스를 활성화하여 결과를 볼 수 있습니다. 세분화된 액세스 제어를 참조하세요

4. aws:CreateStack 권한을 허용하는 역할을 사용하여 로그인합니다.
5. MSF 콘솔 대시보드로 이동하여 스트리밍 애플리케이션 생성을 선택합니다.
6. 스트림 처리 애플리케이션을 설정하는 방법 선택에서 블루프린트 사용을 선택합니다.
7. 블루프린트 드롭다운 메뉴에서 실시간 AI 애플리케이션 블루프린트를 선택합니다.
8. 원하는 구성을 제공합니다. 페이지 구성 생성을 참조하세요.
9. 블루프린트 배포를 선택하여 CloudFormation 배포를 시작합니다.
10. CloudFormation 배포가 완료되면 배포된 Flink 애플리케이션으로 이동합니다. 애플리케이션의 런타임 속성을 확인합니다.
11. 애플리케이션에 런타임 속성을 변경/추가하도록 선택할 수 있습니다. 이러한 속성을 구성하는 방법에 대한 자세한 내용은 런타임 속성 구성을 참조하세요.

    > Note
    >
    > 참고:

프로비저닝된 OpenSearch를 사용하는 경우 세분화된 액세스 제어를 활성화했는지 확인하세요.
프로비저닝된 클러스터가 프라이빗인 경우 OpenSearch 프로비저닝된 VPC 엔드포인트 URL에 `https://`를 추가하고 이 엔드포인트를 가리키도록 `sink.os.endpoint`를 변경합니다.
프로비저닝된 클러스터가 퍼블릭인 경우 MSF 애플리케이션이 인터넷에 액세스할 수 있는지 확인합니다. 자세한 내용은 >>>>>> express-brokers-publication-merge type="documentation" url="managed-flink/latest/java/vpc-internet.html" >VPC에 연결된 Managed Service for Apache Flink 애플리케이션에 대한 인터넷 및 서비스 액세스를 참조하세요.

12. 모든 구성이 만족스러우면 Run을 선택합니다. 애플리케이션 실행이 시작됩니다.
13. MSK 클러스터에 메시지를 펌프합니다.
14. Opensearch 클러스터로 이동하여 OpenSearch 대시보드로 이동합니다.
15. 대시보드의 왼쪽 메뉴에서 검색을 선택합니다. 벡터 임베딩과 함께 지속된 문서가 보일 것입니다.
16. 인덱스에 저장된 벡터를 사용하는 방법을 알아보려면 벡터 검색 컬렉션 작업을 참조하세요.

페이지 구성 생성

이 주제에서는 실시간 AI 애플리케이션 블루프린트에 대한 구성을 지정할 때 참조할 페이지 구성 생성에 대해 설명합니다.

애플리케이션 이름

MSF의 기존 필드에서 애플리케이션에 원하는 이름을 지정합니다.

MSK 클러스터

드롭다운 목록에서 설정 과정에서 생성한 MSK 클러스터를 선택합니다.

주제

설정에서 생성한 토픽(들)의 이름을 추가합니다.

입력 스트림 데이터 유형

MSK 스트림에 문자열 입력을 제공할 경우 문자열을 선택합니다.

MSK 스트림의 입력이 JSON인 경우 JSON을 선택합니다. 임베딩된 JSON 키에서 임베딩 생성을 위해 Bedrock에 값을 보내려는 입력 JSON의 필드 이름을 작성합니다.

Bedrock 임베딩 모델

목록에서 하나를 선택합니다. 선택한 모델에 대한 모델 액세스 권한이 있는지 확인합니다. 그렇지 않으면 스택이 실패할 수 있습니다. Amazon Bedrock 파운데이션 모델에 대한 액세스 추가 또는 제거를 참조하세요.

OpenSearch 클러스터

드롭다운에서 생성한 클러스터를 선택합니다.

OpenSearch 벡터 인덱스 이름

위 단계에서 생성한 벡터 인덱스를 선택합니다.

Amazon MSK 할당량

AWS 계정에는 Amazon MSK에 대한 기본 할당량이 있습니다. 달리 명시되지 않는 한 각 계정당 할당량은 내에서 리전별로 다릅니다AWS 계정.

주제

Amazon MSK에서 할당량 증가 요청

Service Quotas 콘솔 AWS CLI또는 지원 사례를 사용하여 각 리전에 대한 할당량 증가를 요청할 수 있습니다. Service Quotas 콘솔에서 조정 가능한 할당량을 사용할 수 없는 경우 AWS Support Center Console를 사용하여 서비스 할당량 증가 사례를 생성합니다.

지원은 할당량 증가 요청을 승인 또는 거부하거나 부분적으로 승인할 수 있습니다. 증가분은 즉시 부여되지 않으며 적용되는 데 며칠이 걸릴 수 있습니다.

Service Quotas 콘솔을 사용하여 증가 요청

1. https://console.aws.amazon.com/servicequotas/에서 Service Quotas 콘솔을 엽니다.
2. 화면 상단의 탐색 모음에서 리전을 선택합니다.
3. 왼쪽 탐색 창에서 AWS 서비스를 선택합니다.
4. 서비스 찾기 상자에 **msk**를 입력한 다음 Amazon Managed Streaming for Apache Kafka(MSK)를 선택합니다.
5. 서비스 할당량에서 증가를 요청할 할당량 이름을 선택합니다. 예를 들어 **Number of brokers per account**입니다.
6. 계정 수준에서 증가 요청을 선택합니다.
7. 할당량 값 증가에 새 할당량 값을 입력합니다.

8. 요청을 선택합니다.
9. (선택 사항) 콘솔에서 보류 중이거나 최근에 해결된 요청을 보려면 왼쪽 탐색 창에서 대시보드를 선택합니다. 보류 중인 요청의 경우, 요청 상태를 선택하여 요청 접수증을 엽니다. 요청의 초기 상태는 Pending(보류 중)입니다. 상태가 할당량 요청됨으로 변경되면 지원의 케이스 번호가 표시됩니다. 이 사례 번호를 선택하여 요청의 티켓을 엽니다.

AWS CLI또는 SDKs를 사용하여 할당량 증가를 요청하는 방법을 비롯한 자세한 내용은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

Amazon MSK Standard 브로커 할당량

다음 표에서는 Standard 브로커의 할당량을 설명합니다.

측정 기준	할당량	참고
브로커당 서버	90	더 높은 할당량을 요청하려면 Service Quotas 콘솔로 이동합니다.
브로커당 클러스터	ZooKeeper 기반 클러스터의 경우 30, KRaft 기반 클러스터의 경우 60	더 높은 할당량을 요청하려면 Service Quotas 콘솔로 이동합니다.
브로커당 최소 스토리지	1GiB	
브로커당 최대 스토리지	16384GiB	
브로커당 최대 TCP 연결 수 (IAM 액세스 제어)	3000	이 제한을 늘리려면 Kafka `AlterConfig` API 또는 `kafka-configs.sh` 도구를 사용하여 `listener.name.client_iam.max.connections` 또는 `listener.name.client_iam_public.max.connections` 구성 속성을 조정하면 됩니다. 두 속성 중 하

측정 기준	할당량	참고
		나를 높은 값으로 높이면 사용할 수 없게 될 수 있다는 점에 유의하세요.
브로커당 최대 TCP 연결 속도(IAM)	초당 100(M5 및 M7g 인스턴스 크기) 초당 4(t3 인스턴스 크기)	실패한 연결에 대한 재시도를 처리하려면 클라이언트 측에서 `reconnect.backoff.ms` 구성 파라미터를 설정하면 됩니다. 예를 들어 클라이언트가 1초 후에 연결을 다시 시도하도록 하려면 `reconnect.backoff.ms` 를 `1000`으로 설정합니다. 자세한 내용은 Apache Kafka 설명서의 [reconnect.backoff.ms](https://kafka.apache.org/documentation/#consumerconfigs_reconnect.backoff.ms)를 참조하세요.
브로커당 최대 TCP 연결 수(비 IAM)	해당 사항 없음	MSK는 비 IAM 인증에 연결 한도를 강제로 적용하지 않습니다. 과도한 연결로 인해 클러스터에 과부하가 걸리지 않도록 CPU 및 메모리 사용량과 같은 다른 지표를 모니터링해야 합니다.
계정당 구성	100	더 높은 할당량을 요청하려면 Service Quotas 콘솔로 이동합니다. MSK 클러스터의 구성 또는 Apache Kafka 버전을 업데이트하려면 먼저 브로커당 파티션 수가 클러스터 크기를 적절하게 조정: Standard 브로커당 파티션 수에 설명된 제한 이하인지 확인합니다.

측정 기준	할당량	참고
계정당 개정 구성	50	

Amazon MSK Express 브로커 할당량

다음 표에서는 Express 브로커의 할당량을 설명합니다.

측정 기준	할당량	참고
브로커당 서버	90	더 높은 할당량을 요청하려면 [Service Quotas 콘솔](#)로 이동합니다.
브로커당 클러스터	ZooKeeper 기반 클러스터의 경우 30, KRaft 기반 클러스터의 경우 60	더 높은 할당량을 요청하려면 [Service Quotas 콘솔](#)로 이동합니다.
최대 스토리지	무제한	
브로커당 최대 TCP 연결 수 (IAM 액세스 제어)	3000	연결 한도를 높이려면 Kafka AlterConfig API 또는 kafka-configs.sh 도구를 사용하여 다음 구성 속성 중 하나를 조정합니다. • `listener.name.client_iam.max.connections` • `listener.name.client_iam_public.max.connections` 이러한 속성을 높은 값으로 설정하면 클러스터를 사용할 수 없게 될 수 있습니다.

측정 기준	할당량	참고
브로커당 최대 TCP 연결 속도 (IAM)	100/초	실패한 연결에 대한 재시도를 처리하려면 클라이언트 측에서 `reconnect.backoff.ms` 구성 파라미터를 설정하면 됩니다. 예를 들어 클라이언트가 1초 후에 연결을 다시 시도하도록 하려면 `reconnect.backoff.ms` 를 `1000`으로 설정합니다. 자세한 내용은 Apache Kafka 설명서의 reconnect.backoff.ms를 참조하세요.
브로커당 최대 TCP 연결 수(비 IAM)	해당 사항 없음	MSK는 비 IAM 인증에 연결 한도를 강제로 적용하지 않습니다. 그러나 CPU 및 메모리 사용량과 같은 다른 지표를 모니터링하여 과도한 연결로 인해 클러스터에 과부하가 걸리지 않도록 해야 합니다.
계정당 구성	100	더 높은 할당량을 요청하려면 Service Quotas 콘솔로 이동합니다. MSK 클러스터의 구성 또는 Apache Kafka 버전을 업데이트하려면 먼저 브로커당 파티션 수가 클러스터 크기 조정: Express 브로커당 파티션 수에 설명된 제한 이하인지 확인합니다.
계정당 개정 구성	50	
브로커당 최대 수신	권장: 15.6~500.0MBps	인스턴스 크기에 따라 다릅니다.

측정 기준	할당량	참고
브로커당 최대 송신	권장: 31.2~1000.0MBps	인스턴스 크기에 따라 다릅니다.

주제

- 브로커 크기별 Express 브로커 처리량 스로틀 제한
- Express 브로커 파티션 할당량

브로커 크기별 Express 브로커 처리량 스로틀 제한

다음 표에는 다양한 브로커 크기에 대한 수신 및 송신과 관련된 권장 및 최대 처리량 스로틀 제한이 나열되어 있습니다. 이 표에서 권장 처리량은 애플리케이션에 성능 저하가 발생하지 않는 임계값인 지속적인 성능으로 표시됩니다. 두 측정 기준 중 하나에서 이러한 제한을 초과하여 작업하는 경우 처리량이 증가할 수 있지만 성능 저하가 발생할 수도 있습니다. 최대 할당량은 클러스터가 읽기/쓰기 트래픽을 제한하는 임계값입니다. 애플리케이션이 이 임계값을 초과하여 작동할 수 없습니다.

인스턴스 크기	수신을 위한 지속적 성능(MBps)	수신을 위한 최대 할당량(MBps)	송신을 위한 지속적 성능(MBps)	송신을 위한 최대 할당량(MBps)
Express.m7g.large	15.6	23.4	31.2	58.5
Express.m7g.xlarge	31.2	46.8	62.5	117
Express.m7g.2xlarge	62.5	93.7	125	234.2
Express.m7g.4xlarge	124.9	187.5	249.8	468.7
Express.m7g.8xlarge	250	375	500	937.5

인스턴스 크기	수신을 위한 지속적 성능(MBps)	수신을 위한 최대 할당량(MBps)	송신을 위한 지속적 성능(MBps)	송신을 위한 최대 할당량(MBps)
Express.m7g.12xlarge	375	562.5	750	1406.2
Express.m7g.16xlarge	500	750	1000	1875

Express 브로커 파티션 할당량

다음 표는 각 Express 브로커에 권장되는 파티션 수(리더 및 팔로어 복제본 포함)를 보여줍니다. 각 Express 브로커에 대해 다음 표에 언급된 최대 파티션 수를 초과할 수 없습니다.

Express 브로커에 파티션을 할당할 때 고려해야 할 모범 사례에 대한 자세한 내용은 클러스터 크기 조정: Express 브로커당 파티션 수 섹션을 참조하세요.

브로커 크기	브로커당 권장 파티션 수(리더 및 팔로어 복제본 포함)	브로커당 최대 파티션 수
`express.m7g.large`	1000	1500
`express.m7g.xlarge`	1000	2000
`express.m7g.2xlarge`	2500	4000
`express.m7g.4xlarge`	6000	8000
`express.m7g.8xlarge`	12000	16000
`express.m7g.12xlarge`	16000	24000
`express.m7g.16xlarge`	20000	32000

MSK Replicator 할당량

- 계정당 최대 15개의 MSK Replicator.

- MSK Replicator는 정렬된 순서로 최대 750개의 주제만 복제합니다. 더 많은 주제를 복제해야 하는 경우 별도의 Replicator를 생성하는 것이 좋습니다. Replicator당 750개가 넘는 주제에 대한 지원이 필요한 경우 Service Quotas 콘솔로 이동합니다. 'TopicCount' 지표를 사용하여 복제되는 주제 수를 모니터링할 수 있습니다.
- MSK Replicator당 최대 수신 처리량은 초당 1GB입니다. Service Quotas 콘솔을 통해 더 높은 할당량을 요청합니다.
- MSK Replicator 레코드 크기 - 최대 10MB 레코드 크기(message.max.bytes). Service Quotas 콘솔을 통해 더 높은 할당량을 요청합니다.

MSK 서버 규모 조정

다음 표에 지정된 할당량은 달리 명시되지 않는 한 클러스터당입니다.

> **Note**
> 서비스 할당량 제한에 문제가 있는 경우 사용 사례와 요청된 제한으로 지원 사례를 생성합니다.

측정 기준	할당량	할당량 위반 결과
최대 수신 처리량	200MBps	위반에 따른 제한 지속 시간으로 인한 속도 저하
최대 송신 처리량	400MBps	위반에 따른 제한 지속 시간으로 인한 속도 저하
최대 보존 기간	무제한	해당 사항 없음
최대 클라이언트 연결 횟수	3000	연결 종료
최대 연결 시도 횟수	100회/초	연결 종료
최대 메시지 크기	8MiB	오류 코드: INVALID_REQUEST로 요청이 실패함

측정 기준	할당량	할당량 위반 결과
최대 요청 빈도	15,000회/초	위반에 따른 제한 지속 시간으로 인한 속도 저하
최대 주제 관리 API 요청 빈도	2회/초	위반에 따른 제한 지속 시간으로 인한 속도 저하
요청당 최대 가져오기 바이트	55MB	오류 코드: INVALID_REQUEST로 요청이 실패함
최대 사용자 그룹 수	500	JoinGroup 요청 실패
최대 파티션 수(리더)	압축되지 않은 주제의 경우 2400. 압축된 주제의 경우 120. 서비스 할당량 조정을 요청하려면 사용 사례와 요청된 제한이 포함된 지원 사례를 생성합니다.	오류 코드: INVALID_REQUEST로 요청이 실패함
최대 파티션 생성 및 삭제 속도	5분에 250회	오류 코드: THROUGHPUT_QUOTA_EXCEEDED로 요청이 실패함
파티션당 최대 수신 처리량	5MBps	위반에 따른 제한 지속 시간으로 인한 속도 저하
파티션당 최대 송신 처리량	10MBps	위반에 따른 제한 지속 시간으로 인한 속도 저하
최대 파티션 크기(압축된 주제의 경우)	250GB	오류 코드: THROUGHPUT_QUOTA_EXCEEDED로 요청이 실패함
서버리스 클러스터당 최대 클라이언트 VPC 수	5	

측정 기준	할당량	할당량 위반 결과
계정당 최대 서버리스 클러스터 수	10. 서비스 할당량 조정을 요청하려면 사용 사례와 요청된 제한이 포함된 지원 사례를 생성합니다.	

MSK Connect 할당량

- 최대 100개의 사용자 지정 플러그인.
- 최대 100개의 작업자 구성.
- 최대 60개의 연결 작업자. 커넥터가 자동 규모 조정 용량을 가지도록 설정된 경우 커넥터가 보유하도록 설정된 최대 작업자 수는 MSK Connect가 계정에 대한 할당량을 계산하는 데 사용하는 수입니다.
- 커넥터당 최대 10개의 작업자.

MSK Connect에 더 높은 할당량을 요청하려면 Service Quotas 콘솔로 이동합니다.

Amazon MSK 개발자 안내서의 문서 기록

다음 표에서는 Amazon MSK 개발자 안내서의 중요한 변경 사항에 대해 설명합니다.

최종 설명서 업데이트: 2024년 6월 25일

변경	설명	Date
Apache Kafka 버전 4.1.x에 대한 지원	이제 Amazon MSK는 Apache Kafka 버전 4.1.x를 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전을 참조하세요.	2025-10-15
Kafka 워크로드를 Amazon MSK 클러스터로 마이그레이션하는 주제 개선	마이그레이션에 MSK Replicator 또는 Apache MirrorMaker 2.0을 사용하는 방법에 대한 지침을 개선하여 Apache Kafka 워크로드 마이그레이션 주제를 개선했습니다. 또한 콘텐츠 조직을 개선하고 마이그레이션에 대한 단계별 지침을 제공하는 리소스에 대한 관련 링크가 포함되어 있습니다. 자세한 내용은 Kafka 워크로드를 Amazon MSK 클러스터로 마이그레이션을 참조하세요.	2025-10-12
중국 리전에서 mTLS 출시	이제 Amazon MSK는 mTLS를 지원하여 중국(베이징) 및 중국(닝샤) 리전의 클라이언트를 인증합니다. 자세한 내용은 Apache Kafka API의 인증 및 권한 부여를 참조하세요.	2025-09-22

변경	설명	Date
Apache Kafka 버전 주제 개선	• 추가 지원 세부 정보와 ZooKeeper 및 KRaft 메타데이터 관리 시스템 모두를 지원하는 마지막 버전인 Apache Kafka 버전 3.9 설명서가 개선되었습니다. • 자세한 지침으로 Kafka 버전 업그레이드 절차가 개선되었습니다. 자세한 내용은 [지원되는 Apache Kafka 버전](#) 및 [Apache Kafka 버전 업그레이드](#)를 참조하세요.	2025-09-19
Express 브로커에 대한 StorageUsed 지표	이제 Amazon MSK에는 복제본을 제외한 총 스토리지 사용량에 대한 클러스터 수준의 가시성을 제공하는 Express 브로커에 대한 새로운 DEFAULT 수준 지표인 `StorageUsed` 가 포함되어 있습니다. 자세한 내용은 [Express 브로커에 대한 DEFAULT 수준 모니터링](#)을 참조하세요.	2025-07-24
Amazon MSK Express 브로커의 높은 파티션 수	Amazon MSK는 Express 브로커에 대해 높은 파티션 수를 지원하기 시작했습니다. 자세한 내용은 [Express 브로커 파티션 할당량](#)을 참조하세요.	2025-07-21

변경	설명	Date
새로운 Amazon MSK Connect 지표	MSK Connect는 커넥터 데이터 처리량을 측정하기 위해 `SinkConsumerByteRate` 및 `SourceProducerByteRate` 라는 두 가지 새로운 지표를 추가했습니다. 자세한 내용은 [MSK Connect 모니터링](#)을 참조하세요.	2025-06-30
MSK Provisioned 시작하기 주제 점검	사용자 경험, 콘텐츠 흐름 및 가독성을 개선하기 위해 MSK Provisioned 시작하기 주제를 완전히 재구성했습니다. 또한 수정된 주제에는 명확한 결정 지침이 포함된 스토리지 모드, 인증 방법 및 모니터링 수준을 포함한 모든 콘솔 옵션에 대한 자세한 설명서가 포함되어 있습니다. 자세한 내용은 [Amazon MSK 사용 시작](#)을 참조하세요.	2025-06-28
Apache Kafka 버전 3.8.x에서 Express 브로커 지원	이제 Amazon MSK는 Apache Kafka 버전 3.8.x에서 Express 브로커를 지원합니다. 자세한 내용은 [Amazon MSK Express 브로커](#)를 참조하세요.	2025-06-05

변경	설명	Date
새로운 Amazon MSK Replicator 문제 해결 정보	이제 Amazon Managed Streaming for Apache Kafka 개발자 안내서에는 진단 스크립트와 자세한 오류 완화 절차가 포함된 포괄적인 MSK Replicator 문제 해결 설명서가 포함되어 있습니다. 자세한 내용은 ReplicatorFailure 지표를 사용하여 MSK Replicator 실패 문제 해결을 참조하세요.	2025-05-09
중단 없는 인증서 갱신	Amazon MSK는 필수 13개월 인증서 업데이트 중에 유지 관리 가동 중지 시간을 제거하기 위해 Express 브로커에 대한 중단 없는 인증서 갱신을 시작했습니다. 자세한 내용은 Amazon MSK 암호화를 참조하세요.	2025-05-05
Apache Kafka 버전 4.0.x에 대한 지원	이제 Amazon MSK는 Apache Kafka 버전 4.0.x를 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전을 참조하세요.	2025-05-02
중국 리전에서 Amazon MSK Connect 출시	이제 MSK Connect를 모든 중국 리전(중국(베이징) 및 중국(닝샤))에서 사용할 수 있습니다.	2025-04-10

변경	설명	Date
Apache Kafka 버전 3.9.x에 대한 지원	이제 Amazon MSK는 Apache Kafka 버전 3.9.x를 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전을 참조하세요.	2025-04-21
MSK Connect용 Amazon EventBridge Kafka 싱크 커넥터	이제 Amazon Managed Streaming for Apache Kafka 개발자 안내서에는 EventBridge Kafka 싱크 커넥터를 MSK Connect와 함께 사용하는 방법을 설명하는 포괄적인 주제가 포함되어 있습니다. 자세한 내용은 MSK Connect용 EventBridge Kafka 싱크 커넥터 설정을 참조하세요.	2025-03-28
Express 브로커 할당량 업데이트	이제 Amazon Managed Streaming for Apache Kafka 개발자 안내서에는 Express 브로커의 수신 및 송신 처리량 제한에 대한 정보가 포함되어 있습니다. 자세한 내용은 Amazon MSK Express 브로커 할당량을 참조하세요.	2025-03-06
Apache Kafka 버전 3.8.x에 대한 지원	이제 Amazon MSK는 Apache Kafka 버전 3.8.x를 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전을 참조하세요.	2025-02-20

변경	설명	Date
Amazon MSK 버전 3.4.0 지원 종료일 개정	Apache Kafka 버전 3.4.0의 지원 종료일이 2025년 8월 4일로 개정되었습니다. 자세한 내용은 지원되는 Apache Kafka 버전을 참조하세요.	2025-02-18
기존 MSK Connect 커넥터 구성을 수정하기 위한 UpdateConnector API 시작	이제 Amazon MSK에 기존 MSK Connect 커넥터 구성을 수정하는 UpdateConnector API가 포함되어 새 커넥터를 생성할 필요가 없습니다. 또한 커넥터 업데이트 작업을 추적하고 구성 변경에 대한 감사 추적 기록을 유지하기 위해 DescribeConnectorOperation 및 ListConnectorOperations API에 대한 설명서가 추가되었습니다.	2025-01-12
AWS PrivateLink인터페이스 VPC 엔드포인트 설명서	Amazon Managed Streaming for Apache Kafka 개발자 안내서에 이제 AWS PrivateLink인터페이스 VPC 엔드포인트 설명서가 포함되어 있습니다. 자세한 내용은 인터페이스 VPC 엔드포인트와 함께 Amazon MSK API 사용을 참조하세요.	2024-12-18
MSK Connect 3.7.x 버전 출시	이제 MSK Connect는 버전 3.7.x를 지원합니다. 자세한 내용은 MSK Connect 이해를 참조하세요.	2024-12-18

변경	설명	Date
Express 브로커 기능이 추가되었습니다. 개발자 안내서 주제가 재구성되었습니다.	MSK는 Standard 및 새로운 Express 브로커를 지원합니다.	2024-11-6
적절한 Graviton 업그레이드 기능을 추가했습니다.	클러스터 브로커 크기를 M5 또는 T3에서 M7g로, 또는 M7g에서 M5로 업데이트할 수 있습니다.	2024-6-25
3.4.0 지원 종료일이 발표되었습니다.	Apache Kafka 버전 3.4.0의 지원 종료일은 2025년 6월 17일입니다.	2024-6-24
브로커 제거 기능을 추가했습니다.	가용성 영향, 데이터 내구성 위험 또는 데이터 스트리밍 애플리케이션의 중단 없이 브로커 세트를 제거하여 프로비저닝된 클러스터의 스토리지 및 컴퓨팅 용량을 줄일 수 있습니다.	2024-5-16
AWSMSKReplicatorExecutionRole에 `WriteDataIdempotently` 추가됨	WriteDataIdempotently 권한이 AWSMSKReplicatorExecutionRole 정책에 추가되어 MSK 클러스터 간의 데이터 복제를 지원합니다.	2024-5-16
브라질과 바레인에서 Graviton M7g 브로커가 출시되었습니다.	Amazon MSK는 이제 AWSGraviton 프로세서(Amazon Web Services에서 구축한 사용자 지정 Arm 기반 프로세서)를 사용하여 M7g 브로커의 남아메리카(sa-east-1, 상파울루) 및 중동(me-south-1, 바레인) 리전 가용성을 지원합니다.	2024-2-07

변경	설명	Date
Graviton M7g 브로커를 중국 리전으로 릴리스	이제 Amazon MSK는 AWSGraviton 프로세서(Amazon Web Services에서 구축한 사용자 지정 Arm 기반 프로세서)를 사용하여 M7g 브로커의 중국 리전 가용성을 지원합니다.	2024-01-11
Amazon MSK Kafka 버전 지원 정책	Amazon MSK 지원 Kafka 버전 지원 정책에 대한 설명을 추가했습니다. 자세한 내용은 [Apache Kafka 버전]을 참조하세요.	2023-12-08
Amazon MSK Replicator를 지원하는 새로운 서비스 실행 역할 정책입니다.	Amazon MSK가 Amazon MSK Replicator를 지원하기 위해 새로운 `AWSMSKReplicatorExecutionRole` 정책을 추가했습니다. 자세한 내용은 [AWS 관리형 정책AWSMSKReplicatorExecutionRole] 단원을 참조하십시오.	2023-12-06
M7g Graviton 지원	이제 Amazon MSK는 AWSGraviton 프로세서(Amazon Web Services에서 구축한 사용자 지정 Arm 기반 프로세서)를 사용하는 M7g 브로커를 지원합니다.	2023-11-27

변경	설명	Date
Amazon MSK Replicator	Amazon MSK Replicator는 Amazon MSK 클러스터 간에 데이터를 복제하는 데 사용할 수 있는 새로운 기능입니다. Amazon MSK Replicator에는 AmazonMSKFullAccess 정책에 대한 업데이트가 포함되어 있습니다. 자세한 내용은 AWS 관리형 정책AmazonMSKFullAccess 단원을 참조하십시오.	2023-09-28
IAM 모범 사례 업데이트.	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하세요.	2023-03-08
다중 VPC 프라이빗 연결을 지원하는 서비스 연결 역할 업데이트	이제 Amazon MSK에는 계정에서 네트워크 인터페이스와 VPC 엔드포인트를 관리하여 클러스터 브로커가 VPC의 클라이언트에 액세스할 수 있도록 하는 AWSServiceRoleForKafka 서비스 연결 역할 업데이트가 포함됩니다. Amazon MSK는 `DescribeVpcEndpoints` , `ModifyVpcEndpoint` , `DeleteVpcEndpoints` 에 대한 권한을 사용합니다. 자세한 내용은 Amazon MSK의 서비스 연결 역할 단원을 참조하십시오.	2023-03-08

변경	설명	Date
Apache Kafka 2.7.2에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.7.2를 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전 단원을 참조하십시오.	2021-12-21
Apache Kafka 2.6.3에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.6.3을 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전 단원을 참조하십시오.	2021-12-21
MSK 서버리스 사전 릴리스	MSK 서버리스는 서버리스 클러스터를 생성하는 데 사용할 수 있는 새로운 기능입니다. 자세한 내용은 MSK 서버리스 단원을 참조하십시오.	2021-11-29
Apache Kafka 2.8.1에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.8.1을 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전 단원을 참조하십시오.	2021-09-30
MSK Connect	MSK Connect는 Apache Kafka 커넥터를 만들고 관리하는 데 사용할 수 있는 새로운 기능입니다. 자세한 내용은 MSK Connect 이해 단원을 참조하십시오.	2021-09-16
Apache Kafka 2.7.1에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.7.1을 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전 단원을 참조하십시오.	2021-05-25

변경	설명	Date
Apache Kafka 2.8.0에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.8.0을 지원합니다. 자세한 내용은 [지원되는 Apache Kafka 버전](#) 단원을 참조하십시오.	2021-04-28
Apache Kafka 2.6.2에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.6.2를 지원합니다. 자세한 내용은 [지원되는 Apache Kafka 버전](#) 단원을 참조하십시오.	2021-04-28
브로커 유형 업데이트 지원	이제 기존 클러스터의 브로커 유형을 변경할 수 있습니다. 자세한 내용은 [Amazon MSK 클러스터 브로커 크기 업데이트](#) 단원을 참조하십시오.	2021-01-21
Apache Kafka 2.6.1에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.6.1을 지원합니다. 자세한 내용은 [지원되는 Apache Kafka 버전](#) 단원을 참조하십시오.	2021-01-19
Apache Kafka 2.7.0에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.7.0을 지원합니다. 자세한 내용은 [지원되는 Apache Kafka 버전](#) 단원을 참조하십시오.	2020-12-29

변경	설명	Date
Apache Kafka 버전 1.1.1을 사용하는 새로운 클러스터가 없음	더 이상 Apache Kafka 버전 1.1.1로 새 Amazon MSK 클러스터를 생성할 수 없습니다. 그러나 Apache Kafka 버전 1.1.1을 실행 중인 기존 MSK 클러스터가 있는 경우 해당 기존 클러스터에서 현재 지원되는 모든 기능을 계속 사용할 수 있습니다. 자세한 내용은 Apache Kafka 버전 단원을 참조하십시오.	2020-11-24
소비자 지연 지표	이제 Amazon MSK에서 소비자 지연을 모니터링하는 데 사용할 수 있는 지표를 제공합니다. 자세한 내용은 Amazon MSK 프로비저닝된 클러스터 모니터링 단원을 참조하십시오.	2020-11-23
크루즈 컨트롤 지원	이제 Amazon MSK가 LinkedIn의 Cruise Control을 지원합니다. 자세한 내용은 Amazon MSK에서 Apache Kafka용 LinkedIn의 Cruise Control 사용 단원을 참조하십시오.	2020-11-17
Apache Kafka 2.6.0에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.6.0을 지원합니다. 자세한 내용은 지원되는 Apache Kafka 버전 단원을 참조하십시오.	2020-10-21

변경	설명	Date
Apache Kafka 2.5.1에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.5.1을 지원합니다. Apache Kafka 버전 2.5.1에서 Amazon MSK는 클라이언트와 ZooKeeper 엔드포인트 간 전송 시 암호화를 지원합니다. 자세한 내용은 [지원되는 Apache Kafka 버전](#) 단원을 참조하십시오.	2020-09-30
애플리케이션 자동 확장	사용량 증가에 대응하여 클러스터의 스토리지를 자동으로 확장하도록 Amazon Managed Streaming for Apache Kafka를 구성할 수 있습니다. 자세한 내용은 [클러스터의 자동 규모 조정](#) 단원을 참조하십시오.	2020-09-30
사용자 이름과 암호 보안 지원	이제 Amazon MSK에서 사용자 이름과 암호를 사용하여 클러스터에 로그인하는 기능을 지원합니다. Amazon MSK는 AWSSecrets Manager에 자격 증명을 저장합니다. 자세한 내용은 [SASL/SCRAM 인증](#) 단원을 참조하십시오.	2020-09-17
Amazon MSK 클러스터의 Apache Kafka 버전 업그레이드 지원	이제 기존 MSK 클러스터의 Apache Kafka 버전을 업그레이드할 수 있습니다.	2020년 5월 28일
T3.small 브로커 노드 지원	이제 Amazon MSK는 Amazon EC2 유형 T3.small의 브로커를 사용한 클러스터 생성을 지원합니다.	2020년 4월 8일

변경	설명	Date
Apache Kafka 2.4.1에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.4.1을 지원합니다.	2020년 4월 2일
스트리밍 브로커 로그 지원	이제 Amazon MSK는 브로커 로그를 CloudWatch Logs, Amazon S3 및 Amazon Data Firehose로 스트리밍할 수 있습니다. Firehose는 이러한 로그를 OpenSearch Service와 같이 지원하는 대상에 전송할 수 있습니다.	2020년 2월 25일
Apache Kafka 2.3.1에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.3.1을 지원합니다.	2019년 12월 19일
오픈 모니터링	이제 Amazon MSK는 Prometheus를 사용한 개방형 모니터링을 지원합니다.	2019년 12월 4일
Apache Kafka 2.2.1에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.2.1을 지원합니다.	2019년 7월 31일
정식 출시	새로운 기능에는 태그 지정 지원, 인증, TLS 암호화, 구성 및 브로커 스토리지 업데이트 기능이 포함됩니다.	2019년 5월 30일
Apache Kafka 2.1.0에 대한 지원	이제 Amazon MSK가 Apache Kafka 버전 2.1.0을 지원합니다.	2019년 2월 5일

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