aws

사용자 가이드

Amazon CloudWatch

Amazon CloudWatch: 사용자 가이드

Table of Contents

Amazon CloudWatch란 무엇인가요?

Amazon CloudWatch는 Amazon Web Services(AWS) 리소스 및 AWS에서 실행되는 애플리케이션을 실시간으로 모니터링합니다. CloudWatch를 사용하여 리소스 및 애플리케이션에 대해 측정할 수 있는 변수인 지표를 수집하고 추적할 수 있습니다.

CloudWatch 홈페이지에는 사용 중인 모든 AWS 서비스에 관한 지표가 자동으로 표시됩니다. 사용자 지정 대시보드를 추가로 생성해 사용자 지정 애플리케이션에 대한 지표를 표시하고, 선택한 지표의 사용자 지정 집합을 표시할 수 있습니다.

지표를 감시해 알림을 보내거나 임계값을 위반한 경우 모니터링 중인 리소스를 자동으로 변경하는 경보를 생성할 수 있습니다. 예를 들면, Amazon EC2 인스턴스의 CPU 사용량과 디스크 읽기 및 쓰기를 모니터링한 다음에 증가한 로드를 처리하려면 추가 인스턴스를 시작해야 하는지 여부를 해당 데이터로 결정할 수 있습니다. 또한 이러한 데이터를 사용하여 잘 사용되지 않는 인스턴스를 중지할 수도 있습니다.

CloudWatch를 사용하면 시스템 전체의 리소스 사용률, 애플리케이션 성능, 운영 상태를 파악할 수 있습니다.

CloudWatch 액세스

다음 방법 중 하나를 사용하여 CloudWatch에 액세스할 수 있습니다.

- Amazon CloudWatch 콘솔 – https://console.aws.amazon.com/cloudwatch/
- AWS CLI – 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface 설정 단원을 참조하세요.
- CloudWatch API – 자세한 내용은 Amazon CloudWatch API 참조 단원을 참조하세요.
- AWS SDK – 자세한 내용은 Amazon Web Services용 도구를 참조하세요.

관련 AWS 서비스

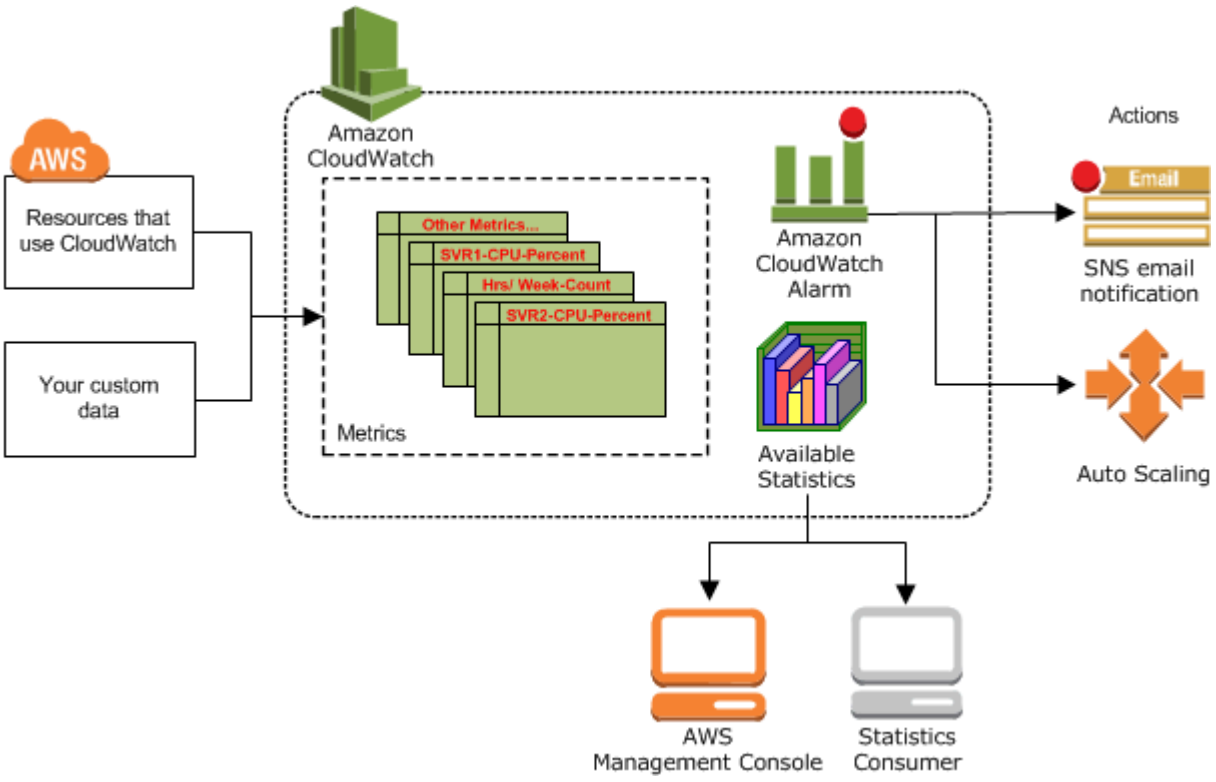
다음 서비스가 Amazon CloudWatch와 함께 사용됩니다.

- Amazon Simple Notification Service(Amazon SNS)는 구독 중인 엔드포인트 또는 클라이언트에 메시지를 전달 또는 전송하는 것을 조정하고 관리합니다. CloudWatch와 함께 Amazon SNS를 사용하여 경보 임계값에 도달한 경우 메시지를 전송합니다. 자세한 내용은 Amazon SNS 알림 설정 단원을 참조하세요.

- Amazon EC2 Auto Scaling을 사용하면 사용자 정의 정책, 상태 확인, 일정에 따라 Amazon EC2 인스턴스를 자동으로 시작하거나 종료할 수 있습니다. Amazon EC2 Auto Scaling과 함께 CloudWatch 경보를 사용하여 수요에 따라 EC2 인스턴스의 크기를 조정할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 동적 크기 조정 단원을 참조하세요.
- AWS CloudTrail을 사용하면 AWS Management Console, AWS CLI 및 기타 서비스에서 수행한 호출을 포함하여 계정의 Amazon CloudWatch API에 대한 호출을 모니터링할 수 있습니다. CloudTrail 로깅을 활성화한 경우 CloudWatch는 CloudTrail을 구성할 때 지정한 Amazon S3 버킷에 로그 파일을 씁니다. 자세한 내용은 AWS CloudTrail으로 Amazon CloudWatch API 호출 로그하기 단원을 참조하세요.
- AWS Identity and Access Management(IAM)는 사용자의 AWS 리소스에 대한 액세스를 안전하게 제어하도록 지원하는 웹 서비스입니다. IAM을 사용하여 AWS 리소스를 사용할 수 있는 사용자를 제어(인증)하고 해당 사용자가 사용할 수 있는 리소스 및 사용 방법을 제어(권한 부여)할 수 있습니다. 자세한 내용은 Amazon CloudWatch의 Identity and Access Management 단원을 참조하세요.

Amazon CloudWatch 작동 방식

Amazon CloudWatch는 기본적으로 지표 리포지토리입니다. Amazon EC2와 같은 AWS 서비스는 지표를 리포지토리에 저장하므로 이러한 지표를 기반으로 통계를 검색할 수 있습니다. 사용자 지정 지표를 리포지토리에 저장하면 해당 지표에 대한 통계도 검색할 수 있습니다.



지표를 사용하여 통계를 계산한 다음, CloudWatch 콘솔에서 데이터를 그래픽으로 나타낼 수 있습니다. 지표를 생성하여 CloudWatch에 전송하는 다른 AWS 리소스에 대한 자세한 내용은 CloudWatch 지표를 게시하는 AWS 서비스 단원을 참조하세요.

특정 기준이 충족되었을 때 Amazon EC2 인스턴스를 중지, 시작 또는 종료하도록 경보 작업을 구성할 수 있습니다. 또한 사용자를 대신하여 Amazon EC2 Auto Scaling 및 Amazon Simple Notification Service(Amazon SNS) 작업을 시작하는 경보를 생성할 수 있습니다. CloudWatch 경보 생성에 대한 자세한 내용은 경보 단원을 참조하세요.

AWS 클라우드 컴퓨팅 리소스는 가용성이 매우 높은 데이터 설비에 있습니다. 확장성 및 안정성을 더욱 높이기 위해 각 데이터 센터 설비는 지역이라고 하는 특정 지리적 영역에 있습니다. 장애 격리 및 안정성을 최대한 높이기 위해 리전이 서로 완전히 분리되도록 설계되었습니다. 지표는 리전에 별도로 저장되지만, CloudWatch 교차 리전 기능을 사용하여 다른 리전의 통계를 집계할 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 교차 계정 교차 리전 CloudWatch 콘솔 및 Regions and Endpoints를 참조하세요.

Amazon CloudWatch 개념

다음 용어 및 개념은 Amazon CloudWatch를 이해하고 사용하는 데 있어 매우 중요합니다.

- 네임스페이스
- 지표
- 차원
- 해결 방법
- Statistics
- 백분위수
- 경보

CloudWatch 지표, 경보, API 요청, 경보 이메일 알림에 대한 Service Quotas에 대한 정보는 CloudWatch Service Quotas를 참조하세요.

네임스페이스

네임스페이스는 CloudWatch 지표의 컨테이너입니다. 다른 네임스페이스의 지표는 서로 격리되어 있으므로 다른 애플리케이션의 지표가 실수로 동일한 통계로 집계되는 일은 없습니다.

기본 네임스페이스는 없습니다. CloudWatch에 게시하는 각 데이터 포인트의 네임스페이스를 지정해야 합니다. 사용자는 지표를 생성할 때 네임스페이스 이름을 지정할 수 있습니다. 이러한 이름은 유효한 ASCII 문자를 포함해야 하며 255자 이하여야 합니다. 가능한 문자로는 영숫자 문자(0-9A-Za-z), 마침표(.), 하이픈(-), 밑줄(_), 슬래시(/), 해시(#), 콜론(:) 및 스페이스 문자가 있습니다. 네임스페이스에는 공백이 아닌 문자가 하나 이상 포함되어야 합니다.

AWS 네임스페이스는 일반적으로 `AWS/`*`service`*라는 명명 규칙을 사용합니다. 예를 들어 Amazon EC2는 `AWS/EC2` 네임스페이스를 사용합니다. AWS 네임스페이스 목록은 CloudWatch 지표를 게시하는 AWS 서비스 단원을 참조하십시오.

지표

지표는 CloudWatch의 기본 개념입니다. 지표는 CloudWatch에 게시된 시간 순서별 데이터 요소 집합을 나타냅니다. 지표는 모니터링할 변수로, 데이터 요소는 시간에 따른 변수의 값을 나타내는 것으로 간주합니다. 예를 들어 특정 EC2 인스턴스의 CPU 사용량은 Amazon EC2가 제공하는 하나의 지표입니다. 데이터 요소 그 자체는 데이터를 수집하는 애플리케이션이나 비즈니스 활동에서 나올 수 있습니다.

기본적으로 많은 AWS 서비스에서 리소스(예: Amazon EC2 인스턴스, Amazon EBS 볼륨, Amazon RDS DB 인스턴스)에 대한 지표를 무료로 제공합니다. 또한 유료로 Amazon EC2 인스턴스와 같은 일부 리소스에 대한 세부 모니터링을 사용하거나 자체 애플리케이션 지표를 게시할 수도 있습니다. 사용자 지정 지표의 경우 원하는 순서와 속도로 데이터 요소를 추가할 수 있습니다. 이러한 데이터 요소에 대한 통계를 정렬된 시계열 집합으로 검색할 수 있습니다.

지표는 생성된 리전에만 존재합니다. 지표는 삭제가 불가능하지만, 지표에 새 데이터가 게시되지 않을 경우 15개월 후에 자동으로 만료됩니다. 15개월이 지난 데이터 요소는 순서대로 만료됩니다. 새로운 데이터 요소가 들어오면 15개월이 지난 데이터가 삭제됩니다.

지표는 이름, 네임스페이스 및 0개 이상의 측정기준으로 고유하게 정의됩니다. 지표의 각 데이터 요소에는 타임스탬프와 측정 단위(선택 사항)가 있습니다. CloudWatch에서 어떤 지표의 통계든 검색할 수 있습니다.

자세한 내용은 사용 가능한 지표 보기 및 사용자 지정 지표 게시 단원을 참조하세요.

타임스탬프

각 지표 데이터 요소에는 타임스탬프가 연결되어 있어야 합니다. 타임스탬프는 최대 2주 전이고 최대 2시간 빠를 수 있습니다. 타임스탬프를 제공하지 않으면 CloudWatch는 데이터 요소를 수신한 시간에 따라 자동으로 타임스탬프를 생성합니다.

타임스탬프는 시간, 분, 초(예: 2016-10-31T23:59:59Z)와 더불어 완전한 날짜가 있는 dateTime 개체입니다. 자세한 내용은 dateTime을 참조하십시오. 필수 요건은 아니지만 UTC(협정 세계시)를 사용하는 것이 좋습니다. CloudWatch에서 통계를 검색할 때 모든 시간은 UTC 기준입니다.

CloudWatch 경보는 UTC의 현재 시간을 기반으로 지표를 확인합니다. 현재 UTC 시간이 아닌 타임스탬프와 함께 사용자 지정 지표를 CloudWatch에 전송하면 경보에 [데이터 부족(Insufficient Data)] 상태가 표시되거나 경보가 지연될 수 있습니다.

지표 보존 기간

CloudWatch는 지표 데이터를 다음과 같이 유지합니다.

- 기간이 60초 미만으로 설정된 데이터 요소들은 3시간 동안 사용이 가능합니다. 이러한 데이터 요소는 고분해능 사용자 지정 지표입니다.
- 기간이 60초(1분)로 설정된 데이터 요소들은 15일 동안 사용이 가능
- 300초(5분) 주기의 데이터 포인트는 63일 동안 사용할 수 있습니다.
- 기간이 3600초(1시간)로 설정된 데이터 요소들은 455일(15개월) 동안 사용이 가능

원래 더 짧은 기간으로 게시된 데이터 요소는 장기 보관을 위해 집계됩니다. 예를 들어 데이터를 1분 기간으로 수집할 경우 15일 동안 1분 분해능으로 데이터를 사용할 수 있습니다. 15일 이후에는 이 데이터를 계속 사용할 수 있지만 데이터가 5분 분해능으로 집계됩니다. 63일 이후에는 이 데이터가 추가로 집계되어 1시간 분해능으로 제공됩니다.

> **Note**
>
> 지난 2주 동안 새로운 데이터 요소가 없는 지표는 콘솔에 나타나지 않습니다. 콘솔의 모든 지표 탭에 있는 검색 상자에 지표 이름이나 측정기준 이름을 입력할 때도 나타나지 않으며 list-metrics 명령의 결과에도 반환되지 않습니다. 이러한 지표를 검색하는 가장 좋은 방법은 AWS CLI에서 get-metric-data 또는 get-metric-statistics 명령을 사용하는 것입니다.

차원

차원은 지표의 보안 인증에 속하는 명칭/값 쌍입니다. 각 지표에 측정기준을 최대 30개까지 할당할 수 있습니다.

모든 지표에는 자신을 설명하는 고유한 특징이 있고 측정기준을 이러한 특징에 대한 범주로 생각할 수 있습니다. 측정기준을 사용하면 통계 계획을 위한 구조를 설계할 수 있습니다. 측정기준은 지표에 대한

고유한 식별자의 일부이므로 지표 중 하나에 이름/값 쌍을 추가할 때마다 해당 지표의 새로운 변형이 생성되는 것입니다.

CloudWatch에 데이터를 전송하는 AWS 서비스는 각 지표에 측정기준을 연결합니다. 측정기준을 사용하여 CloudWatch가 반환하는 결과를 필터링할 수 있습니다. 예를 들어 지표를 검색할 때 `InstanceId` 측정기준을 지정하여 특정 EC2 인스턴스에 대한 통계를 얻을 수 있습니다.

Amazon EC2와 같은 특정 AWS 서비스에서 생성한 지표의 경우 CloudWatch는 측정기준 전반의 데이터를 집계할 수 있습니다. 예를 들어 `AWS/EC2` 네임스페이스의 지표를 검색하는데 어떤 측정기준도 지정하지 않으면 CloudWatch는 지정된 지표에 대한 모든 데이터를 집계하여 요청된 통계를 생성합니다. 사용자 지정 지표의 경우 CloudWatch는 측정기준 전반에서 집계하지 않습니다.

측정기준 조합

CloudWatch는 지표에 동일한 지표 이름이 있는 경우에도 각각의 고유한 측정기준 조합을 별도의 지표로 처리합니다. 사용자가 게시한 측정기준의 조합만 사용해서 통계를 검색할 수 있습니다. 통계를 검색할 때 지표 생성 시 사용했던 네임스페이스, 지표 이름 및 측정기준 파라미터에 동일한 값을 지정합니다. 또한 CloudWatch에서 집계에 사용할 시작 및 종료 시간을 지정할 수도 있습니다.

예를 들어 다음 속성을 가진 DataCenterMetric 네임스페이스에 ServerStats라는 이름을 가진 서로 다른 4개의 지표를 게시한다고 가정합니다.

```
Dimensions: Server=Prod, Domain=Frankfurt, Unit: Count, Timestamp:
 2016-10-31T12:30:00Z, Value: 105
Dimensions: Server=Beta, Domain=Frankfurt, Unit: Count, Timestamp:
 2016-10-31T12:31:00Z, Value: 115
Dimensions: Server=Prod, Domain=Rio,       Unit: Count, Timestamp:
 2016-10-31T12:32:00Z, Value: 95
Dimensions: Server=Beta, Domain=Rio,       Unit: Count, Timestamp:
 2016-10-31T12:33:00Z, Value: 97
```

이러한 4개의 지표만 게시할 경우, 측정기준의 조합에 대한 통계를 검색할 수 있습니다.

- `Server=Prod,Domain=Frankfurt`
- `Server=Prod,Domain=Rio`
- `Server=Beta,Domain=Frankfurt`
- `Server=Beta,Domain=Rio`

다음 측정기준이 사용되거나 측정기준을 지정하지 않은 경우에는 통계를 검색할 수 없습니다. (단, 여러 지표에 대한 통계를 검색할 수 있는 지표 수학 SEARCH 함수를 사용하는 경우는 예외입니다. 자세한 내용은 그래프에서 검색 표현식 사용 단원을 참조하세요.)

- `Server=Prod`
- `Server=Beta`
- `Domain=Frankfurt`
- `Domain=Rio`

해결 방법

각 지표는 다음 중 하나입니다.

- 표준 분해능 - 1분 세분화 데이터
- 고분해능 - 1초 세분화 데이터

AWS 서비스에 의해 생성되는 지표는 기본적으로 표준 분해능입니다. 사용자 지정 지표를 게시할 때는 지표를 표준 분해능 또는 고분해능으로 정의할 수 있습니다. 고분해능 지표를 게시할 경우 CloudWatch는 이 지표를 1초 분해능으로 저장합니다. 그러면 사용자는 1초, 5초, 10초, 30초 또는 60초의 배수 기간으로 지표를 읽고 검색할 수 있습니다.

고분해능 지표는 애플리케이션의 단기(1분 미만) 활동을 보다 즉각적으로 관찰할 수 있게 해줍니다. 사용자 지정 지표에 대해 `PutMetricData`를 호출할 때마다 요금이 부과되며, 따라서 고분해능 지표에 대해 `PutMetricData`를 자주 호출할수록 요금이 증가할 수 있다는 점에 유의하세요. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

고분해능 지표에 대해 경보를 설정할 경우 고분해능 경보를 10초 또는 30초 기간으로 지정하거나 60초의 배수 기간으로 정기 경보를 설정할 수 있습니다. 10초 또는 30초 기간의 고분해능 경보는 요금이 더 비쌉니다.

Statistics

통계는 지정한 기간에 걸친 지표 데이터 집계입니다. CloudWatch는 사용자 지정 데이터가 제공하거나 다른 AWS 서비스가 CloudWatch에 제공한 지표 데이터 요소를 기반으로 통계를 제공합니다. 집계는 네임스페이스, 지표 이름, 측정기준 및 데이터 요소 측정 단위를 사용하여 지정한 기간에 대해 수행됩니다.

CloudWatch에서 지원하는 통계에 대한 자세한 정의는 CloudWatch 통계 정의 단원을 참조하세요.

단위

각각의 통계는 측정 단위를 가지고 있습니다. 단위로는 Bytes, Seconds, Count 및 Percent가 있습니다. CloudWatch에서 지원하는 단위의 전체 목록은 Amazon CloudWatch API 참조의 MetricDatum 데이터 유형을 참조하세요.

사용자 지정 지표를 만들 때 단위를 지정할 수도 있습니다. 단위를 지정하지 않을 경우 CloudWatch는 None을 단위로 사용합니다. 단위를 사용하면 데이터에 개념적 의미를 더할 수 있습니다. CloudWatch는 내부적으로 단위에 의미를 부여하지 않지만 다른 애플리케이션은 단위를 기반으로 의미 정보를 도출할 수 있습니다.

측정 단위를 지정하는 지표 데이터 요소들은 개별적으로 집계됩니다. 단위를 지정하지 않고 통계를 구하는 경우 CloudWatch는 동일한 단위의 데이터 요소를 모두 함께 집계합니다. 단위만 다른 동일한 지표가 두 개 있는 경우에는 각 단위에 대해 하나씩, 개별 데이터 스트림 두 개가 반환됩니다.

기간

'기간'은 특정 Amazon CloudWatch 통계와 연관된 시간의 길이입니다. 각 통계는 지정한 기간에 대해 수집된 지표 데이터의 집계를 나타냅니다. 기간은 초 단위로 정의되며, 유효한 기간 값은 1, 5, 10, 30 또는 60의 배수입니다. 예를 들어 6분의 기간을 지정하려면 기간 값으로 360을 사용합니다. 기간을 변경하여 데이터가 집계되는 방식을 조정할 수 있습니다. 기간 기본값은 60초입니다. 기간은 최소 1초이고 기본값인 60초 이상이면 60의 배수여야 합니다.

저장 분해능 1초를 사용하여 정의한 사용자 지정 지표만 1분 미만 기간을 지원합니다. 콘솔에서는 항상 60 미만의 기간을 설정할 수 있지만 지표가 저장되는 방식과 일치하도록 기간을 선택해야 합니다. 1분 미만 기간을 지원하는 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하십시오.

통계를 검색할 때 기간, 시작 시간, 종료 시간을 지정할 수 있습니다. 이들 파라미터는 통계와 연관된 전체 기간을 결정합니다. 시작 시간과 종료 시간은 지난 1시간 동안의 통계를 얻을 수 있도록 기본 설정되어 있습니다. 시작 시간 및 종료 시간에 지정하는 값에 따라 CloudWatch가 반환하는 기간이 결정됩니다. 예를 들어 기간, 시작 시간 및 종료 시간에 대한 기본값을 사용해 통계를 검색하면 전 시간 동안 1분마다 집계된 통계값들이 반환됩니다. 10분 단위로 집계된 통계를 선호할 경우에는 기간을 600으로 지정합니다. 전체 시간 동안 통계를 집계하고 싶은 경우에는 기간을 3600으로 지정합니다.

특정 시간 동안 통계가 집계되는 경우, 통계가 그 기간이 시작하는 시간으로 타임 스탬프가 추가됩니다. 예를 들어, 7:00pm에서 8:00pm에 집계된 데이터는 타임 스탬프가 7:00pm로 추가됩니다. 또한, 7:00pm와 8:00pm 사이에 집계된 데이터는 7:00pm에 표시되기 시작하며, 그렇게 집계된 데이터의 값은 CloudWatch가 해당 기간 동안 더 많은 샘플을 수집하면서 변경될 수 있습니다.

기간은 CloudWatch 경보에도 중요합니다. 특정 지표를 모니터링하도록 경보를 생성하면 CloudWatch가 해당 지표를 지정된 임계값과 비교하게 됩니다. CloudWatch가 이러한 비교를 수행하는 방식을 광범위하게 제어할 수 있습니다. 비교 작업이 수행되는 기간을 지정할 수 있을 뿐 아니라, 결론에 도달하기까지 사용되는 평가 기간의 수를 지정할 수 있습니다. 예를 들어 세 평가 기간을 지정하는 경우 CloudWatch는 세 데이터 요소의 기간을 비교합니다. CloudWatch는 가장 오래된 데이터 요소가 위반이고 다른 데이터 요소가 위반 또는 누락인 경우에만 이를 알려 줍니다.

집계

Amazon CloudWatch는 통계 검색 시 지정한 기간에 따라 통계를 집계합니다. 동일하거나 유사한 타임스탬프를 사용하여 데이터 요소를 원하는 만큼 게시할 수 있습니다. CloudWatch는 지정된 기간에 따라 데이터 요소를 집계합니다. CloudWatch는 리전 전체에서 데이터를 자동으로 집계하지 않습니다. 그러나 지표 수학을 사용하여 다양한 리전의 지표를 집계할 수 있습니다.

동일한 타임스탬프뿐만 아니라 동일한 네임스페이스 및 측정기준을 공유하는 지표에 대한 데이터 요소를 게시할 수 있습니다. CloudWatch는 이러한 데이터 요소에 대해 통계를 집계해 반환합니다. 또한 타임스탬프에 상관 없이 동일하거나 다른 지표에 대한 여러 데이터 요소를 게시할 수도 있습니다.

대량의 데이터 세트에서는 통계 세트라는 사전 집계된 데이터 세트를 삽입할 수 있습니다. 통계 집합을 사용하면 CloudWatch에 여러 데이터 요소의 Min, Max, Sum, SampleCount를 제공할 수 있습니다. 통계 세트는 1분에 여러 번 데이터를 수집해야 하는 경우 일반적으로 사용됩니다. 예를 들어 웹 페이지의 요청 지연 시간에 대한 지표가 있다고 가정해 보겠습니다. 웹 페이지 방문 시 데이터를 게시하는 것은 적절하지 않습니다. 해당 웹 페이지에 대한 모든 방문의 대기 시간을 수집하고 1분에 한 번씩 이러한 데이터를 집계하여 해당 통계 집합을 CloudWatch에 전송하는 것이 좋습니다.

Amazon CloudWatch는 지표의 소스를 구별하지 않습니다. 소스가 다르지만 네임스페이스 및 측정기준이 동일한 지표를 게시하는 경우 CloudWatch는 이 지표를 단일 지표로 처리합니다. 이는 확장된 분산형 시스템의 서비스 지표에 유용할 수 있습니다. 예를 들어 웹 서버 애플리케이션의 모든 호스트가 처리 중인 요청의 대기 시간을 나타내는 동일한 지표를 게시할 수 있습니다. CloudWatch는 이러한 지표를 단일 지표로 처리하므로 애플리케이션 전체의 모든 요청에 대한 최솟값, 최댓값, 평균, 합계 등의 통계를 얻을 수 있습니다.

백분위수

백분위수는 데이터 세트에서 값의 상대적 위치를 나타냅니다. 예를 들어 95 백분위는 데이터의 95%가 이 값보다 아래에 있고 5%가 이 값보다 위에 있다는 것을 의미합니다. 백분위수는 지표 데이터의 분포를 정확하게 이해하는 데 도움이 됩니다.

백분위 수는 종종 이상치를 격리하는 데 사용됩니다. 일반적인 분포에서 데이터의 95%는 평균값으로부터 2 표준 편차 내에 있으며, 데이터의 99.7%는 평균값으로부터 표준 편차 3 이내에 있습니다. 3 표

준 편차 밖에 있는 데이터는 평균값에서 크게 벗어나 있다는 점에서 종종 이상치로 간주됩니다. 예를 들어 뛰어난 고객 경험을 보장하기 위해 EC2 인스턴스의 CPU 사용률을 모니터링하고 있다고 가정합니다. 평균값을 모니터링하면 이상치가 감춰질 수 있습니다. 최대값을 모니터링하면 단 하나의 이상치로도 결과가 잘못될 수 있습니다. 백분위수를 사용하면 CPU 사용률에 대한 95 백분위를 모니터링하여 비정상적으로 부하가 많은 인스턴스를 확인할 수 있습니다.

일부 CloudWatch 지표는 백분위수를 통계로 지원합니다. 이러한 지표의 경우 다른 CloudWatch 통계(평균, 최솟값, 최댓값, 합계)를 사용할 때와 마찬가지로 백분위수를 사용하여 시스템 및 애플리케이션을 모니터링할 수 있습니다. 예를 들어 경보를 생성할 때 통계 함수로 백분위수를 사용할 수 있습니다. 백분위수를 소수점 이하 10자리까지 지정할 수 있습니다(예: p95.0123456789).

사용자 지정 지표에 대해 요약되지 않은 원시 데이터 요소를 게시하는 경우 사용자 지정 지표에 백분위수 통계를 사용할 수 있습니다. 지표 값에 음수 값이 포함된 지표에서는 백분위수 통계를 사용할 수 없습니다.

CloudWatch가 백분위수를 계산하려면 원시 데이터 요소가 필요합니다. 대신 통계 세트를 사용해 데이터를 게시하면 아래 조건 중 하나가 true인 경우에만 이 데이터에 대한 백분위수 통계를 검색할 수 있습니다.

- 통계 세트의 SampleCount 값은 1이고 Min, Max 및 Sum은 모두 같습니다.
- Min과 Max는 같고 Sum은 Min에 SampleCount를 곱한 값과 같습니다.

다음 AWS 서비스에는 백분위수 통계를 지원하는 지표가 포함되어 있습니다.

- API Gateway
- Application Load Balancer
- Amazon EC2
- Elastic Load Balancing
- Kinesis
- Amazon RDS

또한 CloudWatch는 백분위수와 비슷한 용도로 사용할 수 있는 절사 평균 및 기타 성능 통계도 지원합니다. 자세한 내용은 CloudWatch 통계 정의 단원을 참조하십시오.

경보

경보를 사용하여 작업을 자동으로 시작할 수 있습니다. 경보는 지정한 기간에 단일 지표를 감시하고 시간에 따른 임계값에 대한 지표 값을 기준으로 지정된 작업을 하나 이상 수행합니다. 이 작업은 Amazon SNS 주제 또는 Auto Scaling 정책에 전송되는 알림입니다. 대시보드에 경보를 추가할 수도 있습니다.

경보는 지속적인 상태 변경에 대해서만 작업을 호출합니다. CloudWatch 경보는 단순히 특정 상태에 있다고 해서 작업을 호출하지 않습니다. 상태가 변경되어 지정된 기간 수 동안 유지되어야 합니다.

경보를 생성할 때 지표의 분해능 이상인 경보 모니터링 기간을 선택합니다. 예를 들어 Amazon EC2에 대한 기본 모니터링은 5분마다 인스턴스에 대한 지표를 제공합니다. 기본 모니터링 지표에 대한 경보 설정 시 기간을 300초(5분) 이상으로 선택합니다. Amazon EC2에 대한 세부 모니터링은 1분의 분해능으로 인스턴스에 대한 지표를 제공합니다. 세부 모니터링 지표에 대한 경보 설정 시 기간을 60초(1분) 이상으로 선택합니다.

고분해능 지표에 대해 경보를 설정할 경우 고분해능 경보를 10초 또는 30초 기간으로 지정하거나 60초의 배수 기간으로 정기 경보를 설정할 수 있습니다. 고분해능 경보는 요금이 더 비쌉니다. 고분해능 지표에 대한 자세한 내용은 사용자 지정 지표 게시 단원을 참조하세요.

자세한 내용은 Amazon CloudWatch 경보 사용 및 그래프의 지표에서 경보 생성 섹션을 참조하세요.

결제 및 비용

CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

청구서를 분석하고 비용을 최적화하고 절감하는 데 도움이 될 수 있는 정보는 CloudWatch 결제 및 비용를 참조하세요.

Amazon CloudWatch 리소스

다음의 관련 리소스는 이 서비스를 이용할 때 도움이 될 수 있습니다.

Resource	설명
Amazon CloudWatch FAQ	FAQ는 이 제품에 대해 개발자들이 가장 많이 질문한 상위 개 내용을 소개합니다.

Resource	설명
AWS 개발자 센터	설명서, 코드 예제, 릴리스 정보 및 AWS를 사용하여 혁신적인 응용 프로그램을 구현하는 데 도움이 되는 기타 정보를 이 한 곳에서 찾을 수 있습니다.
AWS Management Console	이 콘솔을 사용하면 프로그래밍 없이 Amazon CloudWatch 및 기타 다양한 AWS 서비스의 기능 대부분을 수행할 수 있습니다.
Amazon CloudWatch 토론 포럼	개발자가 Amazon CloudWatch와 관련된 기술적 문제에 관해 토론할 수 있는 커뮤니티 기반 포럼입니다.
AWS Support	AWS Support 사례를 생성 및 관리하는 곳입니다. 또한 포럼, 기술 FAQ, 서비스 상태 및 AWS Trusted Advisor 등의 기타 유용한 자료에 대한 링크가 있습니다.
Amazon CloudWatch 제품 정보	Amazon CloudWatch에 관한 정보를 제공하는 기본 웹 페이지입니다.
문의처	AWS 결제, 계정, 이벤트, 침해 등에 대해 문의할 수 있는 중앙 연락 지점입니다.

설정

Amazon CloudWatch를 사용하려면 AWS 계정이 있어야 합니다. AWS 계정이 있어야 서비스(예: Amazon EC2)를 사용해 포인트 앤 클릭 방식의 웹 기반 인터페이스인 CloudWatch 콘솔에서 확인 가능한 지표를 생성할 수 있습니다. 뿐만 아니라 AWS 명령줄 인터페이스(CLI)를 설치 및 구성할 수 있습니다.

AWS 계정에 등록

AWS 계정이 없는 경우 다음 절차에 따라 계정을 생성합니다.

AWS 계정에 등록하려면

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

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

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

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

관리 액세스 권한이 있는 사용자 생성

AWS 계정에 가입하고 AWS 계정 루트 사용자에 보안 조치를 한 다음, AWS IAM Identity Center를 활성화하고 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 생성합니다.

귀하의 AWS 계정 루트 사용자 보호

1. 루트 사용자를 선택하고 AWS 계정이메일 주소를 입력하여 AWS Management Console에 계정 소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.

2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정루트 사용자용 가상 MFA 디바이스 활성화(콘솔)를 참조하세요.

관리 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center 설정을 참조하세요.

2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   IAM Identity Center 디렉터리를 ID 소스로 사용하는 방법에 대한 자습서는 AWS IAM Identity Center 사용 설명서의 기본 IAM Identity Center 디렉터리로 사용자 액세스 구성을 참조하세요.

관리 액세스 권한을 가진 사용자로 로그인

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

  IAM Identity Center 사용자로 로그인하는 데 도움이 필요한 경우 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

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

   지침은 AWS IAM Identity Center 사용 설명서의 권한 세트 생성을 참조하세요.

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

   지침은 AWS IAM Identity Center 사용 설명서의 그룹 추가를 참조하세요.

Amazon CloudWatch 콘솔에 로그인합니다.

Amazon CloudWatch 콘솔에 로그인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 필요한 경우 탐색 모음을 사용하여 AWS 리소스가 있는 리전으로 리전을 변경합니다.
3. 이번이 CloudWatch 콘솔을 처음 사용하는 경우더라도 [지표(Your Metrics)]에 이미 지표가 보고되었을 수 있습니다. 이는 지표를 Amazon CloudWatch에 자동 푸시하는 AWS 제품을 무료로 사용했기 때문입니다. 다른 서비스에서는 지표를 사용 설정해야 합니다.

   경보가 없으면 Your Alarms(사용자 경보) 섹션에 경보 생성 버튼이 표시됩니다.

AWS CLI 설정

AWS CLI 또는 Amazon CloudWatch CLI를 사용하여 CloudWatch 명령을 실행할 수 있습니다. AWS CLI로 CloudWatch CLI를 대체합니다. AWS CLI에만 새로운 CloudWatch 기능이 포함됩니다.

AWS CLI를 설치 및 구성하는 방법에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 설정 단원을 참조하세요.

Amazon CloudWatch CLI를 설치 및 구성하는 방법에 대한 자세한 내용은 Amazon CloudWatch CLI 참조의 명령줄 인터페이스 설정 단원을 참조하세요.

Amazon CloudWatch 시작하기

https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

CloudWatch 개요 홈페이지가 표시됩니다.

개요에는 다음 항목이 표시되며, 이러한 항목은 자동으로 새로고침됩니다.

- AWS 서비스별 경보는 계정에서 사용하는 AWS 서비스 목록과 해당 서비스의 경보 상태를 표시합니다. 그 옆에는 계정에 있는 둘 또는 네 경보가 표시됩니다. 숫자는 사용하는 AWS 서비스 수에 따라 달라집니다. 표시된 경보는 ALARM 상태의 경보 또는 가장 최근에 상태가 변경된 경보입니다.

  이러한 상단 영역에서는 모든 서비스의 경보 상태와 최근에 상태가 변경된 경보를 보고 AWS 서비스의 상태를 빠르게 평가할 수 있습니다. 따라서 모니터링해 문제를 빠르게 진단할 수 있습니다.
- 이 영역 아래에는 기본 대시보드가 있습니다(있는 경우). 기본 대시보드는 사용자가 생성하여 CloudWatch-Default라고 이름을 지정한 사용자 지정 대시보드입니다. 대시보드에서는 편리하게, 사용자 지정 서비스 또는 애플리케이션에 대한 지표를 개요 페이지에 추가하거나 가장 모니터링하고 싶은 AWS 서비스에서 주요 지표를 추가로 가져와 표시할 수 있습니다.

> **Note**
>
> 계정이 CloudWatch 교차 계정 관찰성을 위해 설정된 모니터링 계정인 경우에도 CloudWatch 홈 페이지의 자동 대시보드에는 현재 계정의 정보만 표시됩니다. 사용자 지정 교차 계정 대시

보드 생성에 대한 자세한 내용은 CloudWatch 크로스 계정 관측성 대시보드 섹션을 참조하세요.

이 개요에서 여러 AWS 서비스의 지표로 구성된 교차 서비스 대시보드를 보거나 특정 리소스 그룹 또는 특정 AWS 서비스에 초점을 맞출 수 있습니다. 따라서 관심 있는 리소스의 일부분으로 시야를 좁힐 수 있습니다. 자세한 내용은 다음 섹션을 참조하십시오.

단일 서비스를 위해 사전 구축된 자동 대시보드 참조

단일 서비스를 위해 사전 구축된 자동 대시보드를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

   홈 페이지가 나타납니다.
2. 왼쪽 탐색 창에서 대시보드를 선택합니다.
3. 자동 대시보드 탭을 선택한 다음 살펴볼 서비스를 선택합니다.
4. 이 서비스에 대한 경보 보기로 전환하려면 현재 서비스 이름이 표시된 화면 상단 근처에서 경보 내, 데이터 부족 또는 정상 확인란을 선택합니다.
5. 지표를 볼 때 다음과 같은 여러 가지 방법으로 특정 지표를 집중적으로 살펴볼 수 있습니다.

   a. 임의의 그래프에서 지표를 보다 자세히 살펴보려면 그래프 위에 마우스를 놓고 작업 아이콘 지표에서 보기(View in metrics)를 선택합니다.

      새 탭에 그래프가 나타나는데, 해당 그래프 아래에는 관련 지표가 나열되어 있습니다. 그래프 보기를 사용자 지정해 표시되는 지표 및 리소스, 통계, 기간 및 기타 요소를 변경하여 현재 상황을 더욱 정확하게 파악하도록 할 수 있습니다.

   b. 그래프에 표시된 시간 범위에 발생한 로그 이벤트를 볼 수 있습니다. 이렇게 하면 인프라에서 발생해 지표에 예기치 않은 변화를 가져온 이벤트를 찾을 수 있습니다.

      로그 이벤트를 보려면 그래프 위에 마우스를 올려 놓고 작업 아이콘, 로그에서 보기(View in logs)를 선택합니다.

      CloudWatch Logs 보기가 새 탭에 나타나서 로그 그룹 목록을 표시합니다. 로그 그룹 중 하나에서 원래 그래프에 표시된 시간 범위에서 발생한 로그 이벤트를 보려면 로그 그룹을 선택합니다.
6. 경보를 볼 때 다음과 같은 여러 가지 방법으로 특정 경보를 집중적으로 살펴볼 수 있습니다.

- 경보를 보다 자세히 살펴보려면 경보 위에 마우스를 놓고 작업 아이콘 경보에서 보기(View in alarms)를 선택합니다.

  새 탭에 경보 보기가 나타나는데, 여기에는 선택한 경보에 대한 세부 정보와 함께 경보 목록이 표시됩니다. 경보 기록을 보려면 기록(History) 탭을 선택합니다.

7. 경보는 항상 1분마다 한 번씩 새로 고침됩니다. 보기를 새로 고치려면 화면 오른쪽 상단에서 새로 고침 아이콘(구부러진 화살표 2개)을 선택합니다. 화면에서 경보 이외의 항목에 대한 자동 새로 고침 속도를 변경하려면 새로 고침 아이콘 옆에 있는 아래쪽 화살표를 선택하고 새로 고침 속도를 선택합니다. 또한 자동 새로 고침을 끄도록 선택할 수도 있습니다.
8. 현재 표시된 모든 그래프 및 경고에 나타나는 시간 범위를 변경하려면 화면 상단에서 시간 범위(Time range) 옆에서 범위를 선택합니다. 기본적으로 표시된 시간 범위보다 더 많은 시간 범위 옵션 중에서 선택하려면 사용자 지정(custom)을 선택합니다.
9. 교차 서비스 대시보드로 돌아가려면 현재 집중적으로 살펴보고 있는 서비스가 표시된 화면 상단의 목록에서 개요(Overview)를 선택합니다.

   또는 아무 보기에서나 화면 상단에서 CloudWatch를 선택하여 필터를 모두 지우고 개요 페이지로 돌아갈 수 있습니다.

사전 구축된 교차 서비스 대시보드 보기

교차 서비스 대시보드 화면으로 전환하고 사용 중인 모든 AWS 서비스의 대시보드와 상호 작용할 수 있습니다. CloudWatch 콘솔은 대시보드를 알파벳순으로 표시하고 각 대시보드에 하나 또는 두 개의 주요 지표를 표시합니다.

> **Note**
>
> 5개 이상의 AWS 서비스를 사용하는 경우 CloudWatch 콘솔은 개요 화면에 교차 서비스 대시보드를 표시하지 않습니다.

교차 서비스 대시보드 열기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

   개요 화면으로 이동합니다.
2. 개요 화면에서 개요(Overview)가 표시된 드롭다운을 선택한 다음 교차 서비스 대시보드(Cross service dashboard)를 선택합니다.

교차 서비스 대시보드 화면으로 이동합니다.

3. (선택 사항) 원래 인터페이스를 사용하는 경우 교차 서비스 대시보드(Cross-service dashboard) 섹션을 스크롤하여 교차 서비스 대시보드 보기(View Cross-service dashboard)를 선택합니다.

   교차 서비스 대시보드 화면으로 이동합니다.

4. 다음 두 가지 방식으로 특정 서비스를 집중적으로 살펴볼 수 있습니다.

   a. 서비스에 대한 주요 지표를 추가로 보려면 화면 상단에 있는 목록에서 지표의 이름을 선택합니다. 현재 화면 상단에는 교차 서비스 대시보드(Cross service dashboard)가 표시되어 있습니다. 또는 서비스 이름 옆에서 서비스 대시보드 보기(Service dashboard)를 선택할 수 있습니다.

      해당 서비스에 대한 자동 대시보드가 나타나 이 서비스에 대한 지표를 추가로 보여줍니다. 또한 일부 서비스의 경우 서비스 대시보드 하단에 해당 서비스와 관련된 리소스가 표시됩니다. 서비스 콘솔에 대한 리소스 중 하나를 선택해 해당 리소스를 집중적으로 살펴볼 수 있습니다.

   b. 서비스와 관련된 경보를 모두 보려면 화면 오른쪽에서 해당 서비스 이름 옆에 있는 버튼을 선택합니다. 이러한 버튼에 표시된 텍스트는 해당 서비스에서 생성한 경보 수와 ALARM 상태인 경보가 있는지 여부를 나타냅니다.

      경보가 표시되면 설정(예: 측정기준, 임계값 또는 기간)이 유사한 여러 경보는 단일 그래프에 표시될 수 있습니다.

      그런 다음 경보에 대한 세부 정보를 보고 경보 기록을 볼 수 있습니다. 이렇게 하려면 경보 그래프 위에 마우스를 올려 놓고 작업 아이콘, 경보에서 보기(View in alarms)를 선택합니다.

      새 브라우저 탭에 경보 보기가 나타나는데, 여기에는 선택한 경보에 대한 세부 정보와 함께 경보 목록이 표시됩니다. 경보 기록을 보려면 기록(History) 탭을 선택합니다.

5. 특정 리소스 그룹의 리소스를 집중적으로 살펴볼 수 있습니다. 이렇게 하려면 모든 리소스(All resources)가 표시된 페이지 상단의 목록에서 리소스 그룹을 선택합니다.

   자세한 내용은 리소스 그룹을 위해 사전 구축된 대시보드 보기 단원을 참조하십시오.

6. 현재 표시된 모든 그래프 및 경고에 나타나는 시간 범위를 변경하려면 화면 상단에서 시간 범위(Time range) 옆에서 원하는 범위를 선택합니다. 사용자 지정(custom)을 선택하여 기본적으로 표시된 시간 범위보다 더 많은 시간 범위 옵션 중에서 선택합니다.

7. 경보는 항상 1분마다 새로 고침됩니다. 보기를 새로 고치려면 화면 오른쪽 상단에서 새로고침 아이콘(구부러진 화살표 2개)을 선택합니다. 화면에서 경보 이외의 항목에 대한 자동 새로 고침 속도

를 변경하려면 새로 고침 아이콘 옆에 있는 아래쪽 화살표를 선택하고 원하는 새로 고침 속도를 선택합니다. 또한 자동 새로 고침을 끄도록 선택할 수도 있습니다.

교차 서비스 대시보드에서 서비스 제거

서비스의 지표가 교차 서비스 대시보드에 나타나지 않도록 지정할 수 있습니다. 이렇게 하면 교차 서비스 대시보드에서 최우선으로 모니터링하려는 서비스를 집중적으로 살펴볼 수 있습니다.

교차 서비스 대시보드에서 서비스를 제거하더라도 해당 서비스에 대한 경보는 경보 보기에 그대로 나타납니다.

교차 서비스 대시보드에 나타나지 않도록 서비스의 지표를 제거하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

   홈 페이지가 나타납니다.
2. 페이지 상단의 개요(Overview)에서 제거하려는 서비스를 선택합니다.

   해당 서비스에 대한 지표만 표시하도록 보기가 바뀝니다.
3. 작업(Actions)을 선택하고 교차 서비스 대시보드에 표시(Show on cross service dashboard) 옆에 있는 확인란을 선택 취소합니다.

리소스 그룹을 위해 사전 구축된 대시보드 보기

보기에 집중해 단일 리소스 그룹의 지표 및 경보를 표시할 수 있습니다. 리소스 그룹을 사용하면 태그를 사용해 프로젝트를 정리하거나, 아키텍처의 일부분을 집중적으로 살펴보거나, 프로덕션 환경과 개발 환경을 구분할 수 있습니다. 또한 CloudWatch 개요에서 이러한 각 리소스 그룹을 집중적으로 살펴볼 수 있습니다. 자세한 내용은 AWS Resource Groups이란 무엇입니까?단원을 참조하세요.

리소스 그룹을 집중적으로 살펴볼 때 해당 리소스 그룹의 일부로 리소스에 태그를 지정한 서비스만 표시하도록 바뀝니다. 최근 경보 영역에는 리소스 그룹에 속하는 리소스와 연결된 경보만 표시됩니다. 또는 CloudWatch-Default-ResourceGroupName이라는 대시보드를 생성한 경우 해당 대시보드가 기본 대시보드(Default dashboard) 영역에 표시됩니다.

단일 AWS 서비스와 리소스 그룹 둘 다를 동시에 집중적으로 살펴보면 좀 더 자세히 드릴다운할 수 있습니다. 다음 절차는 리소스 그룹을 집중적으로 살펴보는 방법에 대한 설명만을 제공합니다.

단일 리소스 그룹을 집중적으로 살펴보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 모든 리소스(All resources)가 표시된 페이지 상단에서 리소스 그룹을 선택합니다.
3. 해당 리소스 그룹과 관련된 지표를 추가로 보려면 화면 하단 근처에서 교차 서비스 대시보드 보기(View cross service dashboard)를 선택합니다.

   교차 서비스 대시보드가 나타나면 리소스 그룹과 관련된 서비스만 표시됩니다. 서비스마다 주요 지표가 한두 개 표시되어 있습니다.
4. 현재 표시된 모든 그래프 및 경고에 나타나는 시간 범위를 변경하려면 화면 상단의 시간 범위(Time range)에서 범위를 선택합니다. 기본적으로 표시된 시간 범위보다 더 많은 시간 범위 옵션 중에서 선택하려면 사용자 지정(custom)을 선택합니다.
5. 경보는 항상 1분마다 한 번씩 새로 고침됩니다. 보기를 새로 고치려면 화면 오른쪽 상단에서 새로 고침 아이콘(구부러진 화살표 2개)을 선택합니다. 화면에서 경보 이외의 항목에 대한 자동 새로 고침 속도를 변경하려면 새로 고침 아이콘 옆에 있는 아래쪽 화살표를 선택하고 새로 고침 속도를 선택합니다. 또한 자동 새로 고침을 끄도록 선택할 수도 있습니다.
6. 계정의 모든 리소스에 대한 정보를 표시하는 화면으로 되돌아가려면 현재 리소스 그룹의 이름이 표시되어 있는 화면 상단 근처에서 모든 리소스(All resources)를 선택합니다.

사전 구축된 교차 서비스 대시보드 보기

교차 서비스 대시보드 화면으로 전환하고 사용 중인 모든 AWS 서비스의 대시보드와 상호 작용할 수 있습니다. CloudWatch 콘솔은 대시보드를 알파벳 순서로 표시하고 각 서비스의 주요 지표를 하나 또는 두 개 표시합니다.

> **Note**
>
> 5개 이상의 AWS 서비스를 사용하는 경우 CloudWatch 콘솔은 개요 화면에 교차 서비스 대시보드를 표시하지 않습니다.

교차 서비스 대시보드 열기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

   개요 화면으로 이동합니다.

2. 개요 화면에서 개요(Overview)가 표시된 드롭다운을 선택한 다음 교차 서비스 대시보드(Cross service dashboard)를 선택합니다.

   교차 서비스 대시보드 화면으로 이동합니다.

3. (선택 사항) 원래 인터페이스를 사용하는 경우 교차 서비스 대시보드(Cross-service dashboard) 섹션을 스크롤하여 교차 서비스 대시보드 보기(View Cross-service dashboard)를 선택합니다.

   교차 서비스 대시보드 화면으로 이동합니다.

4. 다음 두 가지 방식으로 특정 서비스를 집중적으로 살펴볼 수 있습니다.

   a. 서비스에 대한 주요 지표를 추가로 보려면 화면 상단에 있는 목록에서 지표의 이름을 선택합니다. 현재 화면 상단에는 교차 서비스 대시보드(Cross service dashboard)가 표시되어 있습니다. 또는 서비스 이름 옆에서 서비스 대시보드 보기(Service dashboard)를 선택할 수 있습니다.

      해당 서비스에 대한 자동 대시보드가 나타나 이 서비스에 대한 지표를 추가로 보여줍니다. 또한 일부 서비스의 경우 서비스 대시보드 하단에 해당 서비스와 관련된 리소스가 표시됩니다. 서비스 콘솔에 대한 리소스 중 하나를 선택해 해당 리소스를 집중적으로 살펴볼 수 있습니다.

   b. 서비스와 관련된 경보를 모두 보려면 화면 오른쪽에서 해당 서비스 이름 옆에 있는 버튼을 선택합니다. 이러한 버튼에 표시된 텍스트는 해당 서비스에서 생성한 경보 수와 ALARM 상태인 경보가 있는지 여부를 나타냅니다.

      경보가 표시되면 설정(예: 측정기준, 임계값 또는 기간)이 유사한 여러 경보는 단일 그래프에 표시될 수 있습니다.

      그런 다음 경보에 대한 세부 정보를 보고 경보 기록을 볼 수 있습니다. 이렇게 하려면 경보 그래프 위에 마우스를 올려 놓고 작업 아이콘, 경보에서 보기(View in alarms)를 선택합니다.

      새 브라우저 탭에 경보 보기가 나타나는데, 여기에는 선택한 경보에 대한 세부 정보와 함께 경보 목록이 표시됩니다. 경보 기록을 보려면 기록(History) 탭을 선택합니다.

5. 특정 리소스 그룹의 리소스를 집중적으로 살펴볼 수 있습니다. 이렇게 하려면 모든 리소스(All resources)가 표시된 페이지 상단의 목록에서 리소스 그룹을 선택합니다.

   자세한 내용은 리소스 그룹을 위해 사전 구축된 대시보드 보기 단원을 참조하십시오.

6. 현재 표시된 모든 그래프 및 경고에 나타나는 시간 범위를 변경하려면 화면 상단에서 시간 범위(Time range) 옆에서 원하는 범위를 선택합니다. 사용자 지정(custom)을 선택하여 기본적으로 표시된 시간 범위보다 더 많은 시간 범위 옵션 중에서 선택합니다.

7. 경보는 항상 1분마다 새로 고침됩니다. 보기를 새로 고치려면 화면 오른쪽 상단에서 새로고침 아이콘(구부러진 화살표 2개)을 선택합니다. 화면에서 경보 이외의 항목에 대한 자동 새로 고침 속도를 변경하려면 새로 고침 아이콘 옆에 있는 아래쪽 화살표를 선택하고 원하는 새로 고침 속도를 선택합니다. 또한 자동 새로 고침을 끄도록 선택할 수도 있습니다.

교차 서비스 대시보드에 나타나지 않도록 서비스 제거

서비스의 지표가 교차 서비스 대시보드에 나타나지 않도록 지정할 수 있습니다. 이렇게 하면 교차 서비스 대시보드에서 최우선으로 모니터링하려는 서비스를 집중적으로 살펴볼 수 있습니다.

교차 서비스 대시보드에서 서비스를 제거하더라도 해당 서비스에 대한 경보는 경보 보기에 그대로 나타납니다.

교차 서비스 대시보드에 나타나지 않도록 서비스의 지표를 제거하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

   홈 페이지가 나타납니다.
2. 페이지 상단의 개요(Overview)에서 제거하려는 서비스를 선택합니다.

   해당 서비스에 대한 지표만 표시하도록 보기가 바뀝니다.
3. 작업(Actions)을 선택하고 교차 서비스 대시보드에 표시(Show on cross service dashboard) 옆에 있는 확인란을 선택 취소합니다.

단일 AWS 서비스를 위한 사전 구축된 대시보드 보기

CloudWatch 홈페이지에서 단일 AWS 서비스를 집중적으로 살펴볼 수 있습니다. 단일 AWS 서비스와 리소스 그룹 둘 다를 동시에 집중적으로 살펴보면 좀 더 자세히 드릴다운할 수 있습니다. 다음 절차는 AWS 서비스를 집중적으로 살펴보는 방법만 보여줍니다.

단일 서비스를 집중적으로 살펴보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

   홈 페이지가 나타납니다.
2. 드롭다운 메뉴에 현재 개요가 표시되어 있는 경우 개요에서 서비스 대시보드를 선택합니다.
3. 주목하려는 서비스를 선택합니다.

선택한 서비스의 주요 지표 그래프를 표시하도록 보기가 바뀝니다.

4. 이 서비스에 대한 경보 보기로 전환하려면 현재 서비스 이름이 표시된 화면 상단 근처에서 경보 내, 데이터 부족 또는 정상 확인란을 선택합니다.
5. 지표를 볼 때 다음과 같은 여러 가지 방법으로 특정 지표를 집중적으로 살펴볼 수 있습니다.

   a. 임의의 그래프에서 지표를 보다 자세히 살펴보려면 그래프 위에 마우스를 놓고 작업 아이콘 지표에서 보기(View in metrics)를 선택합니다.

      새 탭에 그래프가 나타나는데, 해당 그래프 아래에는 관련 지표가 나열되어 있습니다. 그래프 보기를 사용자 지정해 표시되는 지표 및 리소스, 통계, 기간 및 기타 요소를 변경하여 현재 상황을 더욱 정확하게 파악하도록 할 수 있습니다.

   b. 그래프에 표시된 시간 범위에 발생한 로그 이벤트를 볼 수 있습니다. 이렇게 하면 인프라에서 발생해 지표에 예기치 않은 변화를 가져온 이벤트를 찾을 수 있습니다.

      로그 이벤트를 보려면 그래프 위에 마우스를 올려 놓고 작업 아이콘, 로그에서 보기(View in logs)를 선택합니다.

      CloudWatch Logs 보기가 새 탭에 나타나서 로그 그룹 목록을 표시합니다. 로그 그룹 중 하나에서 원래 그래프에 표시된 시간 범위에서 발생한 로그 이벤트를 보려면 로그 그룹을 선택합니다.

6. 경보를 볼 때 다음과 같은 여러 가지 방법으로 특정 경보를 집중적으로 살펴볼 수 있습니다.

   - 경보를 보다 자세히 살펴보려면 경보 위에 마우스를 놓고 작업 아이콘 경보에서 보기(View in alarms)를 선택합니다.

     새 탭에 경보 보기가 나타나는데, 여기에는 선택한 경보에 대한 세부 정보와 함께 경보 목록이 표시됩니다. 경보 기록을 보려면 기록(History) 탭을 선택합니다.

7. 경보는 항상 1분마다 한 번씩 새로 고침됩니다. 보기를 새로 고치려면 화면 오른쪽 상단에서 새로 고침 아이콘(구부러진 화살표 2개)을 선택합니다. 화면에서 경보 이외의 항목에 대한 자동 새로 고침 속도를 변경하려면 새로 고침 아이콘 옆에 있는 아래쪽 화살표를 선택하고 새로 고침 속도를 선택합니다. 또한 자동 새로 고침을 끄도록 선택할 수도 있습니다.
8. 현재 표시된 모든 그래프 및 경고에 나타나는 시간 범위를 변경하려면 화면 상단에서 시간 범위(Time range) 옆에서 범위를 선택합니다. 기본적으로 표시된 시간 범위보다 더 많은 시간 범위 옵션 중에서 선택하려면 사용자 지정(custom)을 선택합니다.
9. 교차 서비스 대시보드로 돌아가려면 현재 집중적으로 살펴보고 있는 서비스가 표시된 화면 상단의 목록에서 개요(Overview)를 선택합니다.

또는 아무 보기에서나 화면 상단에서 CloudWatch를 선택하여 필터를 모두 지우고 개요 페이지로 돌아갈 수 있습니다.

리소스 그룹을 위해 사전 구축된 대시보드 보기

보기에 집중해 단일 리소스 그룹의 지표 및 경보를 표시할 수 있습니다. 리소스 그룹을 사용하면 태그를 사용해 프로젝트를 정리하거나, 아키텍처의 일부분을 집중적으로 살펴보거나, 프로덕션 환경과 개발 환경을 구분할 수 있습니다. 또한 CloudWatch 개요에서 이러한 각 리소스 그룹을 집중적으로 살펴볼 수 있습니다. 자세한 내용은 AWS Resource Groups이란 무엇입니까?단원을 참조하세요.

리소스 그룹을 집중적으로 살펴볼 때 해당 리소스 그룹의 일부로 리소스에 태그를 지정한 서비스만 표시하도록 바뀝니다. 최근 경보 영역에는 리소스 그룹에 속하는 리소스와 연결된 경보만 표시됩니다. 또는 CloudWatch-Default-ResourceGroupName이라는 대시보드를 생성한 경우 해당 대시보드가 기본 대시보드(Default dashboard) 영역에 표시됩니다.

단일 AWS 서비스와 리소스 그룹 둘 다를 동시에 집중적으로 살펴보면 좀 더 자세히 드릴다운할 수 있습니다. 다음 절차는 리소스 그룹을 집중적으로 살펴보는 방법을 보여줍니다.

단일 리소스 그룹을 집중적으로 살펴보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 모든 리소스(All resources)가 표시된 페이지 상단에서 리소스 그룹을 선택합니다.
3. 해당 리소스 그룹과 관련된 지표를 추가로 보려면 화면 하단 근처에서 교차 서비스 대시보드 보기(View cross service dashboard)를 선택합니다.

   교차 서비스 대시보드가 나타나면 리소스 그룹과 관련된 서비스만 표시됩니다. 서비스마다 주요 지표가 한두 개 표시되어 있습니다.
4. 현재 표시된 모든 그래프 및 경고에 나타나는 시간 범위를 변경하려면 화면 상단의 시간 범위(Time range)에서 범위를 선택합니다. 기본적으로 표시된 시간 범위보다 더 많은 시간 범위 옵션 중에서 선택하려면 사용자 지정(custom)을 선택합니다.
5. 경보는 항상 1분마다 한 번씩 새로 고침됩니다. 보기를 새로 고치려면 화면 오른쪽 상단에서 새로 고침 아이콘(구부러진 화살표 2개)을 선택합니다. 화면에서 경보 이외의 항목에 대한 자동 새로 고침 속도를 변경하려면 새로 고침 아이콘 옆에 있는 아래쪽 화살표를 선택하고 새로 고침 속도를 선택합니다. 또한 자동 새로 고침을 끄도록 선택할 수도 있습니다.
6. 계정의 모든 리소스에 대한 정보를 표시하는 화면으로 되돌아가려면 현재 리소스 그룹의 이름이 표시되어 있는 화면 상단 근처에서 모든 리소스(All resources)를 선택합니다.

CloudWatch 결제 및 비용

이 섹션에서는 Amazon CloudWatch 기능 사용 시 발생하는 비용을 설명합니다. 또한 CloudWatch 비용을 분석하고 최적화하고 줄이는 데 도움이 되는 방법도 제시합니다. 이 섹션에서 CloudWatch 기능을 설명할 때 요금을 언급하기도 합니다. 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

주제

- Cost Explorer로 CloudWatch 비용 및 사용량 데이터 분석
- AWS Cost and Usage Report와 Athena로 CloudWatch 비용 및 사용량 데이터 분석
- 비용 최적화 및 절감 모범 사례

Cost Explorer로 CloudWatch 비용 및 사용량 데이터 분석

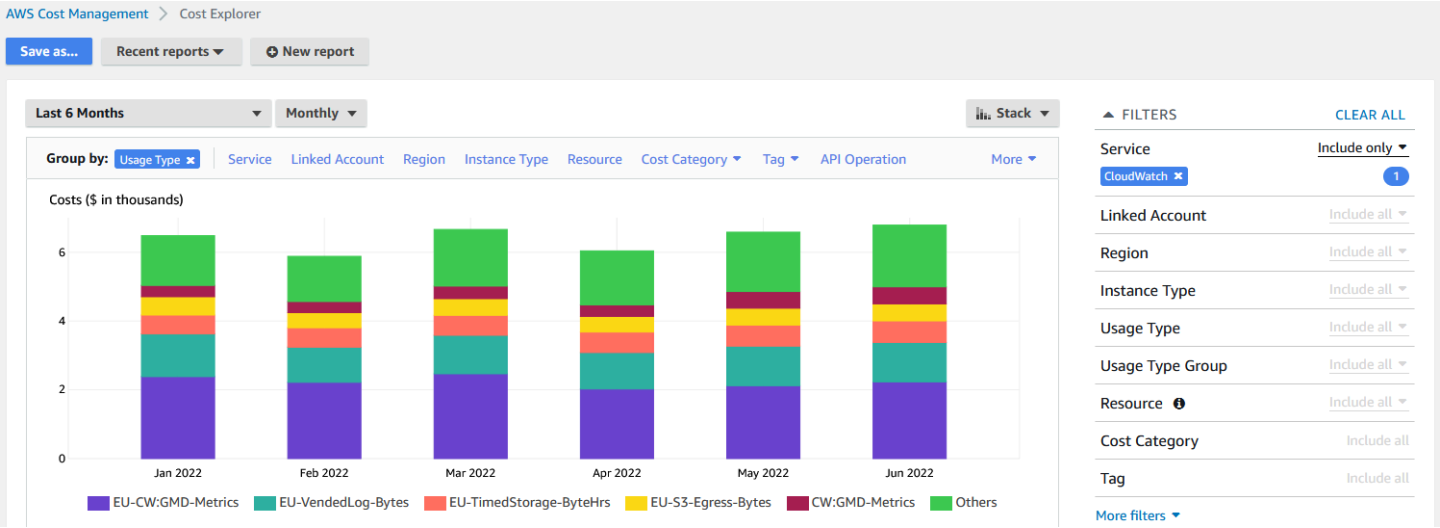
AWS Cost Explorer를 사용하면 CloudWatch를 비롯한 AWS 서비스의 시간 경과에 따른 비용 및 사용량 데이터를 시각화하고 분석할 수 있습니다. 자세한 내용은 AWS Cost Explorer 시작하기를 참조하십시오.

다음 절차에서는 Cost Explorer를 사용하여 CloudWatch 비용 및 사용량 데이터를 시각화하고 분석하는 방법을 설명합니다.

CloudWatch 비용 및 사용량 데이터를 시각화하고 분석하려면

1. https://console.aws.amazon.com/cost-management/home#/custom에서 Cost Explorer 콘솔에 로그인합니다.
2. FILTERS(필터) 아래의 Service(서비스)에서 CloudWatch를 선택합니다.
3. Group by(그룹화 기준)에서 Usage Type(사용 유형)을 선택합니다. 다음과 같은 다른 범주를 기준으로 결과를 그룹화할 수도 있습니다.
   - API Operation(API 작업) - 가장 많은 비용을 유발한 API 작업을 확인할 수 있습니다.
   - Region(리전) - 비용이 가장 많이 발생한 리전을 확인할 수 있습니다.

다음 이미지는 6개월 동안 CloudWatch 기능 사용으로 발생한 비용의 예를 보여줍니다.



어떤 CloudWatch 기능이 가장 많은 비용을 유발했는지 확인하려면 UsageType의 값을 살펴봅니다. 예를 들어 EU-CW:GMD-Metrics는 CloudWatch 대량 API 요청으로 발생한 비용을 나타냅니다.

> **Note**
>
> UsageType의 문자열은 특정 기능과 리전을 나타냅니다. 예를 들어 EU-CW:GMD-Metrics(EU)는 유럽(아일랜드) 리전을 나타내고 EU-CW:GMD-Metrics(GMD-Metrics)는 CloudWatch 대량 API 요청을 나타냅니다.
> UsageType의 전체 문자열 형식은 <Region>-CW:<Feature> 또는 <Region>-<Feature>입니다.
> 가독성을 높이기 위해 이 문서에 있는 표에서 UsageType의 문자열은 문자열 접미사로 축약되었습니다. 예를 들어 EU-CW:GMD-Metrics는 GMD-Metrics로 축약됩니다.

다음 표에는 각 CloudWatch 기능의 이름, 각 하위 기능의 이름 및 UsageType의 문자열이 나열되어 있습니다.

CloudWatch 기능	CloudWatch 하위 기능	**UsageType**
CloudWatch 지표	사용자 지정 지표	MetricMonitorUsage
	세부 모니터링	MetricMonitorUsage
	임베디드 지표	

CloudWatch 기능	CloudWatch 하위 기능	**UsageType**
		MetricMonitorUsage
CloudWatch API 요청	API 요청	Requests
	대량(Get)	GMD-Metrics
	Contributor Insights	GIRR-Metrics
	비트맵 이미지 스냅샷	GMWI-Metrics
CloudWatch 지표 스트림	지표 스트림	MetricStreamUsage
CloudWatch 대시보드	지표가 50개 이하인 대시보드	DashboardsUsageHour-Basic
	50개를 초과하는 지표가 있는 대시보드	DashboardsUsageHour
CloudWatch 경보	표준(지표 경보)	AlarmMonitorUsage
	고분해능(지표 경보)	HighResAlarmMonitorUsage
	Metrics Insights query alarm(Metrics Insights 쿼리 경보)	MetricInsightAlarmUsage
	Composite (aggregated alarm) (복합(집계된 경보))	CompositeAlarmMonitorUsage

CloudWatch 기능	CloudWatch 하위 기능	**UsageType**
CloudWatch Application Signals	Application Signals	Application-Signals
CloudWatch 사용자 지정 로그	수집(ingest)	DataProcessing-Bytes
	저장(archive)	TimedStorage-ByteHrs
	분석(query)	DataScanned-Bytes
CloudWatch Infrequent Access 로그	수집(ingest)	DataProcessingIA-Bytes
CloudWatch 벤딩 로그	전송(Amazon CloudWatch Logs)	VendedLog-Bytes
	전송(CloudWatch Logs Infrequent Access 로그)	VendedLogIA-Bytes
	전송(Amazon Simple Storage Service)	S3-Egress-ComprBytes S3-Egress-Bytes
	전송(Amazon Data Firehose)	FH-Egress-Bytes
Contributor Insights	CloudWatch Logs(규칙)	ContributorInsightRules
	CloudWatch Logs(이벤트)	ContributorInsightEvents

CloudWatch 기능	CloudWatch 하위 기능	**UsageType**
	Amazon DynamoDB(규칙)	ContributorRulesManaged
	DynamoDB(이벤트)	ContributorEventsManaged
Canary(Synthetics)	실행	Canary-runs
Evidently	이벤트	Evidently-event
	분석 단위	Evidently-eau
RUM	이벤트	RUM-event

AWS Cost and Usage Report와 Athena로 CloudWatch 비용 및 사용량 데이터 분석

CloudWatch 비용 및 사용량 데이터를 분석하는 또 다른 방법은 AWS Cost and Usage Report를 Amazon Athena와 함께 사용하는 것입니다. AWS Cost and Usage Report에는 다양한 비용 및 사용량 데이터 세트가 포함되어 있습니다. 비용 및 사용량을 추적하는 보고서를 생성하고, 생성한 보고서를 선택한 S3 버킷에 게시할 수 있습니다. S3 버킷에서 보고서를 다운로드하고 삭제할 수도 있습니다. 자세한 내용은 AWS Cost and Usage Report 사용 설명서에서 AWS Cost and Usage Report란 무엇입니까?를 참조하세요.

> **Note**
>
> AWS Cost and Usage Report 사용에 대해 부과되는 요금은 없습니다. Amazon Simple Storage Service(S3)에 보고서를 게시할 때만 스토리지에 대해 비용을 지불합니다. 자세한 내용은 AWS Cost and Usage Report 사용 설명서에서 할당량 및 제한을 참조하세요.

Athena는 AWS Cost and Usage Report와 함께 사용하여 비용 및 사용량 데이터를 분석할 수 있는 쿼리 서비스입니다. 보고서를 먼저 다운로드하지 않고도 S3 버킷에서 보고서를 쿼리할 수 있습니다. 자세한 내용은 Amazon Athena 사용 설명서에서 Amazon Athena란 무엇입니까?를 참조하세요. 자세한 내용은 Amazon Athena 사용 설명서에서 Amazon Athena란 무엇입니까?를 참조하세요. 요금에 대한 자세한 내용은 Amazon Athena 요금을 참조하세요.

다음 절차에서는 AWS Cost and Usage Report를 활성화하고 이 서비스를 Athena와 통합하는 프로세스를 설명합니다. 이 절차에는 CloudWatch 비용 및 사용량 데이터를 분석하는 데 사용할 수 있는 두 가지 예제 쿼리가 포함되어 있습니다.

> **Note**
>
> 이 문서의 예제 쿼리 중 어느 것이든 사용할 수 있습니다. 이 문서의 모든 예제 쿼리는 costandusagereport라는 데이터베이스에 해당하며, 4월 한 달과 2022년 한 해 동안의 결과를 보여줍니다. 이 정보를 변경할 수 있습니다. 단, 쿼리를 실행하기 전에 데이터베이스 이름이 쿼리의 데이터베이스 이름과 일치하는지 확인하세요.

AWS Cost and Usage Report 및 Athena로 CloudWatch 비용 및 사용량 데이터를 분석하려면

1. AWS Cost and Usage Report를 활성화합니다. 자세한 내용은 AWS Cost and Usage Report 사용 설명서에서 비용 및 사용 보고서 생성을 참조하세요.

   > **Tip**
   >
   > 보고서를 생성할 때 Include resource IDs(리소스 ID 포함)를 선택해야 합니다. 그렇지 않으면 보고서에 `line_item_resource_id` 열이 포함되지 않습니다. 이 줄은 비용 및 사용량 데이터를 분석할 때 비용을 추가로 식별하는 데 도움이 됩니다.

2. AWS Cost and Usage Report와 Athena 통합 자세한 내용은 AWS Cost and Usage Report 사용 설명서에서 AWS CloudFormation 템플릿을 사용하여 Athena 설정을 참조하세요.
3. 비용 및 사용 보고서를 쿼리합니다.

예: Athena 쿼리

다음 쿼리를 사용하여 특정 월에 가장 많은 비용을 유발한 CloudWatch 기능을 표시할 수 있습니다.

```
SELECT
CASE
-- Metrics
WHEN line_item_usage_type LIKE '%%MetricMonitorUsage%%' THEN 'Metrics (Custom, Detailed
 monitoring management portal EMF)'
WHEN line_item_usage_type LIKE '%%Requests%%' THEN 'Metrics (API Requests)'
WHEN line_item_usage_type LIKE '%%GMD-Metrics%%' THEN 'Metrics (Bulk API Requests)'
WHEN line_item_usage_type LIKE '%%MetricStreamUsage%%' THEN 'Metric Streams'
-- Dashboard
WHEN line_item_usage_type LIKE '%%DashboardsUsageHour%%' THEN 'Dashboards'
-- Alarms
WHEN line_item_usage_type LIKE '%%AlarmMonitorUsage%%' THEN 'Alarms (Standard)'
WHEN line_item_usage_type LIKE '%%HighResAlarmMonitorUsage%%' THEN 'Alarms (High
 Resolution)'
WHEN line_item_usage_type LIKE '%%MetricInsightAlarmUsage%%' THEN 'Alarms (Metrics
 Insights)'
WHEN line_item_usage_type LIKE '%%CompositeAlarmMonitorUsage%%' THEN 'Alarms
 (Composite)'
-- Logs
WHEN line_item_usage_type LIKE '%%DataProcessing-Bytes%%' THEN 'Logs (Collect - Data
 Ingestion)'
-- Logs
WHEN line_item_usage_type LIKE '%%DataProcessingIA-Bytes%%' THEN 'Infrequent Access
 Logs (Collect - Data Ingestion)'
WHEN line_item_usage_type LIKE '%%TimedStorage-ByteHrs%%' THEN 'Logs (Storage -
 Archival)'
WHEN line_item_usage_type LIKE '%%DataScanned-Bytes%%' THEN 'Logs (Analyze - Logs
 Insights queries)'
-- Vended Logs
WHEN line_item_usage_type LIKE '%%VendedLog-Bytes%%' THEN 'Vended Logs (Delivered to
 CW)'
WHEN line_item_usage_type LIKE '%%VendedLogIA-Bytes%%' THEN 'Vended Infrequent Access
 Logs (Delivered to CW)'
WHEN line_item_usage_type LIKE '%%FH-Egress-Bytes%%' THEN 'Vended Logs (Delivered to
 Kinesis FH)'
WHEN (line_item_usage_type LIKE '%%S3-Egress-Bytes%%') OR (line_item_usage_type LIKE '%
%S3-Egress-
ComprBytes%%') THEN 'Vended Logs (Delivered to S3)'
-- Other
WHEN line_item_usage_type LIKE '%%Application-Signals%%' THEN 'Application Signals'
WHEN line_item_usage_type LIKE '%%Canary-runs%%' THEN 'Synthetics'
WHEN line_item_usage_type LIKE '%%Evidently%%' THEN 'Evidently'
WHEN line_item_usage_type LIKE '%%RUM-event%%' THEN 'RUM'
```

```
ELSE 'Others'
END AS UsageType,
-- REGEXP_EXTRACT(line_item_resource_id,'^(?:.+?:){5}(.+)$',1) as ResourceID,
-- SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
-- AND line_item_usage_account_id = '123456789012' - If you want to filter on a
 specific account, you can
remove this comment at the beginning of the line and specify an AWS account.
GROUP BY
1
ORDER BY
TotalSpend DESC,
UsageType;
```

예: Athena 쿼리

다음 쿼리를 사용하여 UsageType과 Operation의 결과를 표시할 수 있습니다. 이 쿼리는 CloudWatch 기능 사용에 따른 비용 발생 내역을 보여줍니다. 결과에는 UsageQuantity와 TotalSpend의 값도 표시되므로 총 사용 비용을 확인할 수 있습니다.

> **Tip**
>
> UsageType에 대한 추가 정보를 표시하려면 이 쿼리에 다음 줄을 추가합니다.
> `line_item_line_item_description`
> 이 줄은 Description(설명)이라는 열을 생성합니다.

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_usage_type AS UsageType,
line_item_operation AS Operation,
line_item_resource_id AS ResourceID,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
```

```
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation
```

비용 최적화 및 절감 모범 사례

CloudWatch 지표

Amazon Elastic Compute Cloud(Amazon EC2), Amazon S3, Amazon Data Firehose 등 다수의 AWS 서비스는 무료로 CloudWatch에 지표를 자동 전송합니다. 하지만 다음 범주로 분류된 지표의 경우 추가 비용이 발생할 수 있습니다.

- 사용자 지정 지표, 세부 모니터링 및 임베디드 지표
- API 요청
- 지표 스트림

자세한 내용은 Amazon CloudWatch 지표 사용을 참조하세요.

사용자 지정 지표, 세부 모니터링 및 임베디드 지표

사용자 지정 지표

사용자 지정 지표를 만들어 원하는 순서와 속도로 데이터 포인트를 구성할 수 있습니다.

모든 사용자 지정 지표는 시간을 기준으로 비례 배분됩니다. 이러한 지표는 CloudWatch로 전송된 경우에만 과금됩니다. CloudWatch 지표 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

다음 표에는 CloudWatch 지표와 관련한 하위 기능의 이름이 나와 있습니다. 이 표에는 지표 관련 비용을 분석하고 식별하는 데 유용한 `UsageType` 및 `Operation`의 문자열이 포함되어 있습니다.

> **Note**
>
> Athena로 비용 및 사용량 데이터를 쿼리할 때 다음 표에 나열되는 지표에 대한 자세한 내용을 보려면 `Operation`의 문자열을 표시되는 `line_item_operation`의 결과와 매칭합니다.

CloudWatch 하위 기능	**`UsageType`**	**`Operation`**	용도
사용자 지정 지표	`MetricMonitorUsage`	`MetricStorage`	사용자 지정 지표
세부 모니터링	`MetricMonitorUsage`	`MetricStorage:AWS/`*`{Service}`*	세부 모니터링
임베디드 지표	`MetricMonitorUsage`	`MetricStorage:AWS/Logs-EMF`	임베디드 지표 로깅
로그 필터	`MetricMonitorUsage`	`MetricStorage:AWS/CloudWatchLogs`	로그 그룹 지표 필터

세부 모니터링

CloudWatch에는 두 가지 유형의 모니터링 기능이 있습니다.

- 기본 모니터링

  기본 모니터링은 무료이며 이 기능을 지원하는 모든 AWS 서비스에서 자동으로 활성화됩니다.
- 세부 모니터링

세부 모니터링은 비용이 부과되며 AWS 서비스에 따라 향상된 다양한 기능을 추가로 제공합니다. 세부 모니터링을 지원하는 각 AWS 서비스별로 이 기능을 활성화할지 여부를 선택할 수 있습니다. 자세한 내용은 기본 모니터링과 세부 모니터링을 참조하세요.

> **Note**
>
> 세부 모니터링을 지원하는 다른 AWS 서비스에서는 이 기능의 이름이 다를 수 있습니다. 예를 들어 Amazon S3의 경우 세부 모니터링 기능을 요청 지표라고 합니다.

사용자 지정 지표와 마찬가지로 세부 모니터링은 시간을 기준으로 비례 배분되고 데이터가 CloudWatch로 전송되는 경우에만 과금됩니다. 세부 모니터링은 CloudWatch로 전송된 지표 수에 따라 비용이 발생합니다. 비용을 줄이려면 필요한 경우에만 세부 모니터링을 활성화하세요. 세부 모니터링의 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

예: Athena 쿼리

다음 쿼리를 사용하여 세부 모니터링이 활성화된 EC2 인스턴스를 표시할 수 있습니다.

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_usage_type AS UsageType,
line_item_operation AS Operation,
line_item_resource_id AS ResourceID,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_operation='MetricStorage:AWS/EC2'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
```

```
line_item_resource_id,
line_item_operation,
line_item_line_item_description
ORDER BY line_item_operation
```

임베디드 지표

CloudWatch 임베디드 지표 형식을 사용하면 애플리케이션 데이터를 로그 데이터로 수집하여 유용한 지표를 생성할 수 있습니다. 자세한 내용은 CloudWatch 임베디드 지표 형식을 사용하여 높은 카디널리티 로그 수집 및 지표 생성을 참조하세요.

임베디드 지표는 수집된 로그 수, 보관된 로그 수 및 생성된 사용자 지정 지표 수에 따라 비용이 발생합니다.

다음 표에는 CloudWatch 임베디드 지표 형식과 관련한 하위 기능의 이름이 나와 있습니다. 이 표에는 비용을 분석하고 식별하는 데 유용한 `UsageType` 및 `Operation`의 문자열이 포함되어 있습니다.

CloudWatch 하위 기능	**`UsageType`**	**`Operation`**	용도
사용자 지정 지표	`MetricMonitorUsage`	`MetricStorage:AWS/Logs-EMF`	임베디드 지표 로깅
로그 수집	`DataProcessing-Bytes`	`PutLogEvents`	지정된 로그 그룹 또는 로그 스트림에 로그 이벤트의 배치를 업로드합니다.
로그 보관	`TimedStorage-ByteHrs`	`HourlyStorageMetering`	CloudWatch Logs에 시간당 로그와 바이트당 로그를 저장합니다.

비용을 분석하려면 AWS Cost and Usage Report를 Athena와 함께 사용하여 비용이 발생하는 지표를 식별하고 비용이 어떻게 발생하는지 확인합니다.

CloudWatch 임베디드 지표 형식으로 인해 발생하는 비용을 최적화하려면 높은 카디널리티의 차원을 기반으로 지표를 생성하지 마세요. 그렇게 해야 CloudWatch가 고유한 각 차원 조합에 대해 사용자 지정 지표를 생성하지 않습니다. 자세한 내용은 차원 단원을 참조하세요.

CloudWatch Container Insights를 사용하여 임베디드 지표 형식을 활용하는 경우, AWS Distro for Open Telemetry를 대신 사용하여 지표 관련 비용을 최적화할 수 있습니다. Container Insights를 사용하면 컨테이너식 애플리케이션 및 마이크로서비스의 지표 및 로그를 수집하고 집계하며 요약할 수 있습니다. Container Insights를 활성화하면 CloudWatch 에이전트가 CloudWatch에 로그를 전송하므로 로그를 사용하여 임베디드 지표를 생성할 수 있습니다. 하지만 CloudWatch 에이전트는 CloudWatch에 일정 수의 지표만 전송하는데, 사용하지 않는 지표를 포함하여 제공되는 모든 지표에 대해 요금이 부과됩니다. AWS Distro for Open Telemetry를 사용하면 CloudWatch로 전송할 지표 및 차원을 구성하고 사용자 지정할 수 있습니다. 이를 통해 Container Insights가 생성하는 데이터 양과 비용을 줄일 수 있습니다. 자세한 정보는 다음 자료를 참조하십시오.

- Container Insights 사용
- AWS Distro for Open Telemetry

API 요청

CloudWatch에는 다음과 같은 유형의 API 요청이 있습니다.

- API 요청
- 대량(Get)
- Contributor Insights
- 비트맵 이미지 스냅샷

API 요청은 요청 유형과 요청된 지표 수에 따라 비용이 발생합니다.

다음 표에는 API 요청 유형이 나열되어 있으며, API 관련 비용을 분석하고 식별하는 데 유용한 `UsageType` 및 `Operation`의 문자열이 포함되어 있습니다.

API 요청 유형	**`UsageType`**	**`Operation`**	용도
API 요청	`Requests`	`GetMetric Statistics`	지정된 지표에 대한 통계를 가져옵니다.

API 요청 유형	**UsageType**	**Operation**	용도
	Requests	ListMetrics	지정된 지표를 나열합니다.
	Requests	PutMetricData	CloudWatch에 지표 데이터 포인트를 게시합니다.
	Requests	GetDashboard	지정된 대시보드에 대한 세부 정보를 표시합니다.
	Requests	ListDashboards	계정의 대시보드를 나열합니다.
	Requests	PutDashboard	대시보드를 생성 또는 업데이트합니다.
	Requests	DeleteDashboards	지정된 대시보드를 모두 삭제합니다.
대량(Get)	GMD-Metrics	GetMetricData	CloudWatch 지표 값을 가져옵니다.
Contributor Insights	GIRR-Metrics	GetInsightRuleReport	Contributor Insights에 의해 수집된 시계열 데이터를 반환합니다.
비트맵 이미지 스냅샷	GMWI-Metrics	GetMetricWidgetImage	CloudWatch 지표 1개 이상의 스냅샷을 비트맵 이미지로 가져옵니다.

비용을 분석하려면 Cost Explorer를 사용하고 API Operation(API 작업)별로 결과를 그룹화합니다.

API 요청별로 비용이 다르며, AWS 프리 티어 한도에 따라 제공된 API 호출 수를 초과하면 비용이 발생합니다.

> **Note**
>
> GetMetricData와 GetMetricWidgetImage는 AWS 프리 티어 한도에 포함되지 않습니다. 자세한 내용은 AWS Billing 사용 설명서에서 AWS 프리 티어 사용을 참조하세요.

일반적으로 비용이 많이 발생하는 API 요청은 Put 요청과 Get 요청입니다.

PutMetricData

PutMetricData는 호출할 때마다 비용이 발생하며 사용 사례에 따라 상당한 비용이 발생할 수 있습니다. 자세한 내용은 Amazon CloudWatch API 참조에서 PutMetricData를 참조하세요.

PutMetricData로 인해 발생하는 비용을 최적화하려면 API 호출에서 데이터를 배치 처리하세요. 사용 사례에 따라, CloudWatch Logs 또는 CloudWatch 임베디드 지표 형식을 사용하여 지표 데이터를 주입하는 것이 좋습니다. 자세한 정보는 다음 자료를 참조하십시오.

- Amazon CloudWatch Logs 사용 설명서의 Amazon CloudWatch Logs란 무엇입니까?
- CloudWatch 임베디드 지표 형식을 사용하여 높은 카디널리티 로그 수집 및 지표 생성
- Amazon CloudWatch 임베디드 사용자 지정 지표로 비용을 절감하고 고객에게 집중

GetMetricData

GetMetricData도 상당한 비용을 유발할 수 있습니다. 비용을 유발하는 일반적인 사용 사례 중 하나로, 데이터를 가져와 인사이트를 생성하는 서드 파티 모니터링 도구가 있습니다. 자세한 내용은 Amazon CloudWatch API 참조에서 GetMetricData를 참조하세요.

GetMetricData로 인해 발생하는 비용을 줄이려면, 모니터링되고 사용되는 데이터만 가져오거나 데이터 가져오기 빈도를 줄이는 것이 좋습니다. 사용 사례에 따라, GetMetricData 대신 지표 스트림을 사용하여 저렴한 비용으로 데이터를 서드 파티에게 거의 실시간으로 푸시할 수 있습니다. 자세한 정보는 다음 자료를 참조하십시오.

- 지표 스트림 사용
- CloudWatch 지표 스트림 - 파트너 및 앱에 실시간으로 AWS 지표 전송

GetMetricStatistics

사용 사례에 따라 GetMetricData 대신 GetMetricStatistics를 사용하는 것이 효율적일 수 있습니다. GetMetricData를 사용하면 데이터를 대규모로 신속하게 가져올 수 있습니다. 단,

`GetMetricStatistics`는 최대 API 요청 100만 건이라는 AWS 프리 티어 한도에 포함되므로, 호출당 많은 수의 지표와 데이터 포인트를 가져올 필요가 없는 경우 비용을 줄이는 데 도움이 될 수 있습니다. 자세한 정보는 다음 자료를 참조하십시오.

- Amazon CloudWatch API 참조의 GetMetricStatistics
- GetMetricData 또는 GetMetricStatistics를 사용해야 하나요?

> **Note**
>
> 외부 호출자들은 API를 통해 호출합니다. 현재 이러한 호출자를 식별하는 유일한 방법은 CloudWatch 팀에 대한 기술 지원 요청을 개설하고 관련 정보를 요청하는 것입니다. 기술 지원 요청 생성에 대한 자세한 내용은 AWS에서 기술 지원을 받으려면 어떻게 해야 하나요?를 참조하세요.

CloudWatch 지표 스트림

CloudWatch 지표 스트림을 사용하면 AWS 대상과 서드 파티 제공업체 대상에 지표를 지속적으로 전송할 수 있습니다.

지표 스트림은 지표 업데이트 수를 기준으로 비용이 발생합니다. 지표 업데이트에는 항상 다음 통계의 값이 포함됩니다.

- `Minimum`
- `Maximum`
- `Sample Count`
- `Sum`

자세한 내용은 스트리밍할 수 있는 통계를 참조하세요.

CloudWatch 지표 스트림에서 발생하는 비용을 분석하려면 AWS Cost and Usage Report를 Athena와 함께 사용합니다. 이렇게 하면 비용이 발생하는 지표 스트림을 식별하고 비용이 어떻게 발생하는지 확인할 수 있습니다.

예: Athena 쿼리

다음 쿼리를 사용하여 비용이 발생하는 지표 스트림을 Amazon 리소스 이름(ARN)으로 추적할 수 있습니다.

```
SELECT
SPLIT_PART(line_item_resource_id,'/',2) AS "Stream Name",
line_item_resource_id as ARN,
SUM(CAST(line_item_unblended_cost AS decimal(16,2))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
-- AND line_item_usage_account_id = '123456789012' - If you want to filter on a
 specific account, you can
remove this comment at the beginning of the line and specify an AWS account.
AND line_item_usage_type LIKE '%%MetricStreamUsage%%'
GROUP BY line_item_resource_id
ORDER BY TotalSpend DESC
```

CloudWatch 지표 스트림에서 발생하는 비용을 줄이려면 비즈니스 가치를 제공하는 지표만 스트리밍하세요. 사용하지 않는 지표 스트림을 중지하거나 일시 중지할 수도 있습니다.

CloudWatch 경보

CloudWatch 경보를 사용하면 단일 지표 기반의 경보, Metrics Insights 쿼리 기반의 경보 및 다른 경보를 관찰하는 복합 경보를 생성할 수 있습니다.

> **Note**
>
> 미터법 및 복합 경보에 대한 비용은 시간별로 비례 배분됩니다. 알람이 있는 동안에만 알람에 대한 비용이 발생합니다. 비용을 최적화하려면 잘못 구성되거나 가치가 낮은 경보를 남기지 않도록 해야 합니다. 이를 위해 더 이상 필요하지 않은 CloudWatch 경보를 자동으로 정리할 수 있습니다. 자세한 내용은 Amazon CloudWatch 경보 대규모 정리를 참조하십시오.

지표 경보

지표 경보의 분해능 설정은 다음과 같습니다.

- Standard(표준)(60초마다 평가)

- High resolution(고분해능)(10초마다 평가)

지표 경보를 생성할 때 비용은 경보의 해상도 설정과 경보가 참조하는 지표 수를 기반으로 합니다. 예를 들어, 하나의 지표를 참조하는 지표 경보는 시간당 하나의 경보-지표 비용을 발생시킵니다. 자세한 내용은 Amazon CloudWatch 경보 사용을 참조하세요.

여러 지표를 참조하는 지표 수학 식을 포함하는 지표 경보를 만드는 경우 지표 수학 표현식에서 참조되는 각 경보-지표에 대해 비용이 발생합니다. 지표 수학 표현식이 포함된 지표 경보를 생성하는 방법은 지표 수학 표현식을 기반으로 CloudWatch 경보 생성을 참조하세요.

경보가 과거 지표 데이터를 분석하여 예상 값 모델을 생성하는 이상 탐지 경보를 생성하는 경우 경보에서 참조되는 각 경보-지표에 대한 비용과 예외 탐지 모델이 생성하는 상위 및 하위 대역 지표에 대한 두 개의 추가 지표에 대한 비용이 발생합니다. 이상 탐지 경보를 생성하는 방법은 이상 탐지를 기반으로 CloudWatch 경보 생성을 참조하세요.

Metrics Insights query alarms(Metrics Insights 쿼리 경보)

Metric Insights 쿼리 경보는 특정 유형의 지표 경보로, 표준 해상도(60초마다 평가됨)에서만 사용할 수 있습니다.

Metric Insights 쿼리 경보를 생성할 때 비용은 경보가 참조하는 쿼리에서 분석한 지표 수를 기반으로 합니다. 예를 들어, 필터가 10개의 지표와 일치하는 쿼리를 참조하는 Metric Insights 쿼리 경보는 시간당 10개의 지표 분석 비용을 발생시킵니다. 자세한 내용은 Amazon CloudWatch Pricing(Amazon CloudWatch 요금)의 요금 예를 참조하세요.

Metrics Insights 쿼리와 지표 수학 표현식이 모두 포함된 경보를 생성하면 Metrics Insights 쿼리 경보로 보고됩니다. 경보에 Metrics Insights 쿼리에서 분석한 지표 외에 다른 지표를 참조하는 지표 수학 표현식이 포함된 경우 지표 수학 표현식에서 참조하는 각 경보 지표에 대해 추가 비용이 발생합니다. 지표 수학 표현식이 포함된 지표 경보를 생성하는 방법은 지표 수학 표현식을 기반으로 CloudWatch 경보 생성을 참조하세요.

복합 경보

복합 경보에는 자체 상태를 확인하기 위해 다른 경보의 상태를 평가하는 방법을 지정하는 규칙 표현식이 포함되어 있습니다. 복합 알람은 평가하는 다른 알람 수에 관계없이 시간당 표준 비용이 발생합니다. 규칙 표현식에서 복합 경보가 참조하는 경보는 별도의 비용이 발생합니다. 자세한 내용은 복합 경보 생성을 참조하세요.

경보 사용 유형

다음 표에는 CloudWatch 경보와 관련한 하위 기능의 이름이 나와 있습니다. 이 표에는 경보 관련 비용을 분석하고 식별하는 데 유용한 UsageType의 문자열이 포함되어 있습니다.

CloudWatch 하위 기능	**UsageType**
표준 지표 경보	AlarmMonitorUsage
고분해능 지표 경보	HighResAlarmMonitorUsage
Metrics Insights query alarm(Metrics Insights 쿼리 경보)	MetricInsightAlarmUsage
복합 경보	CompositeAlarmMonitorUsage

경보 비용 절감

4개 이상의 지표를 집계하는 지표 수식 경보에서 발생하는 비용을 최적화하기 위해 데이터가 CloudWatch로 전송되기 전에 데이터를 집계할 수 있습니다. 이렇게 하면 여러 지표에 대한 데이터를 집계하는 경보 대신, 단일 지표에 대한 경보를 만들 수 있습니다. 자세한 내용은 사용자 지정 지표 게시를 참조하세요.

Metrics Insights 쿼리 경보에서 생성된 비용을 최적화하기 위해 쿼리에 사용된 필터가 모니터링하려는 지표와만 일치하는지 확인할 수 있습니다.

비용을 절감하는 가장 좋은 방법은 불필요하거나 사용하지 않는 경보를 모두 제거하는 것입니다. 예를 들어, 더 이상 존재하지 않는 AWS 리소스가 내보낸 지표를 평가하는 경보를 삭제할 수 있습니다.

예: ***DescribeAlarms***이(가) 있는 INSUFFICIENT_DATA 상태의 경보 확인

리소스는 삭제하지만 리소스에서 내보내는 지표 경보는 삭제하지 않은 경우 경보가 계속 존재하며 일반적으로 INSUFFICIENT_DATA 상태가 됩니다. INSUFFICIENT_DATA 상태의 경보를 확인하려면 다음 AWS Command Line Interface(AWS CLI) 명령을 사용합니다.

```
$ aws cloudwatch describe-alarms –state-value INSUFFICIENT_DATA
```

다음 방법으로도 비용을 절감할 수 있습니다.

- 올바른 지표에 대한 경보를 생성합니다.
- 작업하지 않는 리전에 경보가 활성화되어 있지 않은지 확인합니다.

- 복합 알람은 소음을 줄여주지만 추가 비용도 발생합니다.
- 표준 경보를 생성할지 아니면 고분해능 경보를 생성할지 결정할 때는 사용 사례와 각 경보 유형이 제공하는 가치를 고려하세요.

CloudWatch Logs

Amazon CloudWatch Logs의 로그 유형은 다음과 같습니다.

- 사용자 지정 로그(애플리케이션용으로 생성한 로그)
- 벤딩 로그(Amazon Virtual Private Cloud(VPC), Amazon Route 53 등의 다른 AWS 서비스에서 자동으로 생성되는 로그)

벤딩 로그에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 특정 AWS 서비스에서 로깅 활성화를 참조하세요.

사용자 지정 로그와 벤딩 로그는 수집되고, 저장되고, 분석되는 로그 수에 따라 비용이 발생합니다. 반면, 벤딩 로그의 경우 Amazon S3와 Firehose로 전송하는 데 대한 비용이 발생합니다.

다음 표에는 CloudWatch Logs 기능의 이름과 관련 하위 기능의 이름이 나와 있습니다. 이 표에는 로그 관련 비용을 분석하고 식별하는 데 유용한 `UsageType` 및 `Operation`의 문자열이 포함되어 있습니다.

CloudWatch Logs 기능	CloudWatch Logs 하위 기능	**`UsageType`**	**`Operation`**	용도
사용자 지정 로그	수집(ingest)	`DataProcessing-Bytes`	`PutLogEvents`	특정 로그 스트림에 로그의 배치를 업로드합니다.
	저장(archive)	`TimedStorage-ByteHrs`	`HourlyStorageMetering`	CloudWatch Logs에 시간당 로그와 바이트당 로그를 저장합니다.
	분석(Logs Insights 쿼리)	`DataScanned-Bytes`	`StartQuery`	CloudWatch Logs Insights 쿼

CloudWatch Logs 기능	CloudWatch Logs 하위 기능	**UsageType**	**Operation**	용도
				리에 의해 스캔된 데이터를 로깅합니다.
벤딩 로그	전송(CloudWatch Logs)	VendedLog-Bytes	PutLogEvents	특정 로그 스트림에 로그의 배치를 업로드합니다.
	전송(Amazon S3)	S3-Egress-ComprBytes S3-Egress-Bytes	LogDelivery	벤딩 로그 전송(CloudWatch, Amazon S3 또는 Firehose)
	전송(Firehose)	FH-Egress-Bytes	LogDelivery	벤딩 로그 전송(CloudWatch, Amazon S3 또는 Firehose)

비용을 분석하려면 AWS Cost and Usage Report를 Athena와 함께 사용하여 비용이 발생하는 로그를 식별하고 비용이 어떻게 발생하는지 확인합니다.

예: Athena 쿼리

다음 쿼리를 사용하여 비용이 발생하는 로그를 리소스 ID로 추적할 수 있습니다.

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_resource_id AS ResourceID,
line_item_usage_type AS UsageType,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity
FROM
costandusagereport
```

```
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_operation IN
 ('PutLogEvents','HourlyStorageMetering','StartQuery','LogDelivery')
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation
ORDER BY
TotalSpend DESC
```

CloudWatch Logs에서 발생하는 비용을 최적화하려면 다음을 고려하세요.

- 비즈니스 가치를 창출하는 이벤트만 로깅합니다. 이렇게 하면 수집 비용을 줄일 수 있습니다.
- 로그 보존 설정을 변경하여 스토리지 비용을 절감할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs User Guide의 CloudWatch에서 로그 데이터 보존 기간을 변경을 참조하세요.
- CloudWatch Logs Insights가 기록에 자동으로 저장하는 쿼리를 실행합니다. 이렇게 하면 분석 비용을 줄일 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 실행 중인 쿼리 또는 쿼리 기록 보기를 참조하세요.
- CloudWatch 에이전트를 사용하여 시스템 및 애플리케이션 로그를 수집하여 CloudWatch로 전송합니다. 이렇게 하면 기준에 맞는 로그 이벤트만 수집할 수 있습니다. 자세한 내용은 Amazon CloudWatch Agent adds Support for Log Filter Expressions(Amazon CloudWatch 에이전트, 로그 필터 표현식에 대한 지원 추가)를 참조하세요.

벤딩 로그의 비용을 줄이려면, 사용 사례를 고려하여 로그를 CloudWatch로 전송할지 Amazon S3 S3로 전송할지 결정하세요. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 Amazon S3로 전송된 로그를 참조하세요.

> **Tip**
>
> 지표 필터, 구독 필터, CloudWatch Logs Insights 및 Contributor Insights를 사용하려면 벤딩 로그를 CloudWatch로 전송하세요.

또는 VPC 흐름 로그를 감사 및 규정 준수 목적으로 사용하는 경우, 벤딩 로그를 Amazon S3로 전송하세요.

VPC 흐름 로그를 S3 버킷에 게시할 때 발생하는 요금을 추적하는 방법은 Amazon S3에서 AWS Cost and Usage Report 및 비용 할당 태그를 사용하여 VPC 흐름 로그 데이터 모으기 파악(Using s and cost allocation tags to understand VPC FLow Logs data ingestion in Amazon S3)을 참조하세요.

CloudWatch Logs에서 발생하는 비용을 최적화하는 방법은 CloudWatch Logs 요금이 갑자기 증가하는 것은 어떤 로그 그룹 때문인가요?(Which log group is causing a sudden increase in my CloudWatch Logs bill?)를 참조하세요.

Amazon CloudWatch 대시보드 사용

Amazon CloudWatch 대시보드는 CloudWatch 콘솔에서 사용자 지정이 가능한 홈페이지로, 단일 보기에서 리소스(다양한 리전에 분산되어 있는 리소스 포함)를 모니터링하는 데 사용할 수 있습니다. CloudWatch 대시보드를 사용하면 AWS 리소스에 대한 지표 및 경보의 사용자 지정 보기를 생성할 수 있습니다.

대시보드에서 다음을 생성할 수 있습니다.

- 선택한 지표 및 경보에 대한 단일 보기를 생성하여 하나 이상의 리전에 있는 리소스 및 애플리케이션의 상태를 평가할 수 있습니다. 여러 그래프에서 동일한 지표를 손쉽게 추적할 수 있도록 각 그래프에서 지표 각각에 사용되는 색상을 선택할 수 있습니다.
- 작동 지침서를 생성하여 운영 이벤트 동안 팀원들에게 특정 사고에 대한 대응 방법에 관해 지침을 제공할 수 있습니다.
- 중요한 리소스 및 애플리케이션 지표에 대한 공통 뷰를 생성하여 운영 이벤트 동안 신속하고 원활한 의사 소통을 위해 팀원들이 이를 공유하도록 할 수 있습니다.

여러 AWS 계정이 있는 경우 CloudWatch 크로스 계정 관측성을 설정한 다음 모니터링 계정에서 풍부한 크로스 계정 대시보드를 생성할 수 있습니다. 이러한 대시보드에는 소스 계정의 지표 그래프와 소스 계정의 로그 그룹 쿼리가 포함된 CloudWatch Logs Insights 위젯이 포함될 수 있습니다. 또한 모니터링 계정에서 생성하는 경보는 소스 계정의 지표를 관찰할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

콘솔에서 또는 AWS CLI나`PutDashboard` API 작업을 사용하여 대시보드를 생성할 수 있습니다. 즐겨찾기 목록에 대시보드를 추가하여 즐겨찾는 대시보드뿐만 아니라 최근에 방문한 대시보드에도 액세스할 수 있습니다. 자세한 내용은 즐겨찾기 목록에 대시보드 추가를 참조하세요.

CloudWatch 대시보드에 액세스하려면 다음 중 하나가 필요합니다.

- `AdministratorAccess` 정책
- `CloudWatchFullAccess` 정책
- 다음과 같은 특정 권한 중 하나 이상을 포함하는 사용자 지정 정책:
  - 대시보드를 볼 수 있는 `cloudwatch:GetDashboard` 및 `cloudwatch:ListDashboards`
  - 대시보드를 생성하거나 수정할 수 있는 `cloudwatch:PutDashboard`
  - 대시보드를 삭제할 수 있는 `cloudwatch:DeleteDashboards`

내용

CloudWatch 대시보드 생성

시작하려면 CloudWatch 대시보드를 생성합니다. 여러 대시보드를 생성하고 즐겨찾기 목록에 대시보드를 추가할 수 있습니다. AWS 계정의 CloudWatch 대시보드 수에는 제한이 없습니다. 모든 대시보드는 전역 대시보드이며 리전별로 구분되지 않습니다.

다음 절차는 CloudWatch 콘솔에서 대시보드를 생성하는 방법을 보여줍니다. `PutDashboard` API 작업을 사용하여 명령줄 인터페이스에서 대시보드를 생성할 수 있습니다. API 작업에는 대시보드 콘텐츠를 정의하는 JSON 문자열이 포함되어 있습니다. `PutDashboard` API 작업을 사용한 대시보드 생성에 대한 자세한 내용은 Amazon CloudWatch API 참조의 PutDashboard를 참조하세요.

> **Tip**
>
> `PutDashboard` API 작업을 사용하여 새 대시보드를 생성하는 경우 이미 있는 대시보드에서 JSON 문자열을 사용할 수 있습니다.

콘솔에서 대시보드 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards), 대시보드 생성(Create dashboard)을 차례로 선택합니다.

3. 새 대시보드 생성(Create new dashboard) 대화 상자에서 대시보드 이름을 입력하고 대시보드 생성(Create dashboard)을 선택합니다.

   CloudWatch-Default 또는 CloudWatch-Default-***ResourceGroupName***이라는 이름을 사용하는 경우 대시보드가 기본 대시보드(Default Dashboard) 아래의 CloudWatch 홈 페이지의 개요에 표시됩니다. 자세한 내용은 Amazon CloudWatch 시작하기 단원을 참조하십시오.

4. 이 대시보드에 추가(Add to this dashboard) 대화 상자에서 다음 중 하나를 수행합니다.

   - 대시보드에 그래프를 추가하려면 선(Line) 또는 누적 영역(Stacked area)을 선택한 다음 구성(Configure)을 선택합니다. 지표 그래프 추가(Add metric graph) 대화 상자에서 그래프로 표시할 지표를 선택하고 위젯 생성(Create widget)을 선택합니다. 지표가 14일 이상 데이터를 게시하지 않아 대화 상자에 표시되지 않는 경우 수동으로 추가할 수 있습니다. 자세한 내용은 CloudWatch 대시보드에서 수동으로 지표 그래프 생성 단원을 참조하십시오.
   - 대시보드에 지표를 표시하는 숫자를 추가하려면 번호(Number), 구성(Configure)을 차례로 선택합니다. 지표 그래프 추가(Add metric graph) 대화 상자에서 그래프로 표시할 지표를 선택하고 위젯 생성(Create widget)을 선택합니다.
   - 대시보드에 텍스트 블록을 추가하려면 텍스트(Text), 구성(Configure)을 차례로 선택합니다. 새 텍스트 위젯(New text widget) 대화 상자의 마크다운(Markdown)에서 마크다운(Markdown)을 사용하여 텍스트를 포맷하고 위젯 생성(Create widget)을 선택합니다.

5. (선택 사항) 위젯 추가(Add widget)를 선택하고 4단계를 반복하여 대시보드에 다른 위젯을 추가합니다. 이 단계를 여러 번 반복할 수 있습니다.

   대시보드의 각 그래프에는 오른쪽 상단에 정보 아이콘이 있습니다. 이 아이콘을 선택하면 그래프에서 지표에 대한 설명을 볼 수 있습니다.

6. 대시보드 저장을 선택합니다.

CloudWatch 크로스 계정 관측성 대시보드

여러 AWS 계정이 있는 경우 CloudWatch 크로스 계정 관측성을 설정한 다음 모니터링 계정에서 풍부한 크로스 계정 대시보드를 생성할 수 있습니다. 계정 경계 없이 지표, 로그 및 추적을 원활하게 검색, 시각화 및 분석할 수 있습니다.

CloudWatch 크로스 계정 관측성 설정에 대한 자세한 내용은 CloudWatch 크로스 계정 관측성 섹션을 참조하세요.

CloudWatch 크로스 계정 관측성을 사용하면 모니터링 계정의 대시보드에서 다음 작업을 수행할 수 있습니다.

- 소스 계정에 있는 지표의 그래프를 검색, 확인 및 생성합니다. 단일 그래프에 여러 계정의 지표가 포함될 수 있습니다.
- 모니터링 계정에 소스 계정의 지표를 관찰하는 경보를 생성합니다.
- 소스 계정에 있는 로그 그룹의 로그 이벤트를 보고 소스 계정에 있는 로그 그룹의 CloudWatch Logs Insights 쿼리를 실행합니다. 모니터링 계정의 단일 CloudWatch Logs Insights 쿼리는 여러 소스 계정의 여러 로그 그룹을 한 번에 쿼리할 수 있습니다.
- X-Ray의 트레이스 맵에서 소스 계정의 노드를 봅니다. 그런 다음 맵을 특정 소스 계정으로 필터링할 수 있습니다.

모니터링 계정에 로그인하면 CloudWatch 크로스 계정 관측성을 지원하는 모든 페이지의 오른쪽 상단에 파란색 Monitoring account(모니터링 계정) 배지가 나타납니다.

교차 계정 교차 리전 대시보드

여러 AWS 계정 및 여러 리전의 CloudWatch 데이터를 하나의 대시보드에 요약하는 '교차 계정 교차 리전 대시보드'를 생성할 수 있습니다. 이 개략적인 대시보드에서 애플리케이션 전체를 확인할 수 있으며 계정에 로그인 및 로그아웃하거나 리전을 전환하지 않고도 더욱 구체적인 대시보드로 드릴다운할 수 있습니다.

AWS Management Console에서 프로그래밍 방식으로 교차 계정 교차 리전 대시보드를 생성할 수 있습니다.

사전 조건

교차 계정 교차 리전 대시보드를 생성하려면 먼저 하나 이상의 공유 계정과 하나 이상의 모니터링 계정을 활성화해야 합니다. 또한 CloudWatch 콘솔을 사용하여 교차 계정 대시보드를 생성할 수 있으려면 교차 계정 기능에 대해 콘솔을 사용 설정해야 합니다. 자세한 내용은 교차 계정 교차 리전 CloudWatch 콘솔 단원을 참조하세요.

AWS Management Console을 사용하여 교차 계정 교차 리전 대시보드 생성 및 사용

AWS Management Console을 사용하여 교차 계정 교차 리전 대시보드를 생성할 수 있습니다.

교차 계정 교차 리전 대시보드를 생성하려면

1. 모니터링 계정에 로그인합니다.

2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
4. 대시보드를 선택하거나 새 대시보드를 생성합니다.
5. 화면 상단에서 계정 및 리전 간에 전환할 수 있습니다. 대시보드를 생성할 때 여러 계정 및 리전의 위젯을 포함할 수 있습니다. 위젯에는 그래프, 경보 및 CloudWatch Logs Insights 위젯이 포함됩니다.

다른 계정 및 리전의 지표로 그래프 생성

1. 모니터링 계정에 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 탐색 창에서 지표를 선택한 다음 모든 지표를 선택합니다.
4. 지표를 추가할 계정과 리전을 선택합니다. 화면 오른쪽 상단에 있는 계정 및 리전 드롭다운 메뉴에서 계정과 리전을 선택할 수 있습니다.
5. 그래프에 원하는 지표를 추가합니다. 자세한 내용은 지표 그래프 작성 단원을 참조하세요.
6. 4-5단계를 반복하여 다른 계정 및 리전의 지표를 추가합니다.
7. (선택 사항) 그래프로 표시된 지표 탭을 선택하고 선택한 지표를 사용하는 지표 수학 함수를 추가합니다. 자세한 내용은 지표 수학 사용 단원을 참조하세요.

   여러 SEARCH 함수를 포함하도록 단일 그래프를 설정할 수도 있습니다. 검색마다 서로 다른 계정 또는 리전을 참조할 수 있습니다.
8. 그래프 사용을 마치면 작업, 대시보드에 추를를 선택합니다.

   교차 계정 대시보드를 선택하고 대시보드에 추가를 선택합니다.

교차 계정 대시보드에 다른 계정의 경보 추가

1. 모니터링 계정에 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 페이지 상단에서 경보가 있는 계정을 선택합니다.
4. 탐색 창에서 경보(Alarms)를 선택합니다.
5. 추가할 경보 옆의 확인란을 선택하고 대시보드에 추가를 선택합니다.
6. 추가할 교차 계정 대시보드를 선택하고 대시보드에 추가를 선택합니다.

프로그래밍 방식으로 교차 계정 교차 리전 대시보드 생성

AWS API 및 SDK를 사용하여 프로그래밍 방식으로 대시보드를 생성할 수 있습니다. 자세한 내용은 PutDashboard를 참조하세요.

교차 계정 교차 리전 대시보드를 활성화하기 위해 다음 표와 예제와 같이 대시보드 본문 구조에 새 파라미터를 추가했습니다. 전체 대시보드 본문 구조에 대한 자세한 내용은 대시보드 본문 구조 및 구문을 참조하세요.

파라미터	사용	범위	기본값
accountId	위젯 또는 지표가 있는 계정의 ID를 지정합니다.	위젯 또는 지표	현재 로그인되어 있는 계정
region	지표의 리전을 지정합니다.	위젯 또는 지표	콘솔에서 선택된 현재 리전

다음 예제에서는 교차 계정 교차 리전 대시보드에서 위젯의 JSON 소스를 보여줍니다.

이 예제에서는 accountId 필드를 위젯 수준에서 공유 계정의 ID로 설정합니다. 이 위젯의 모든 지표를 해당 공유 계정 및 리전에서 가져오도록 지정합니다.

```
{
    "widgets": [
        {
            ...
            "properties": {
              "metrics": [
                     …..
                     ],
                    "accountId": "111122223333",
                    "region": "us-east-1"
           }
       }
  ]
}
```

이 예제에서는 각 지표 레벨에서 accountId 필드를 다르게 설정합니다. 이 예제에서는 이 지표 수학 표현식의 다른 지표를 서로 다른 공유 계정과 서로 다른 리전에서 가져옵니다.

```
{
"widgets": [
    {
        ...
        "properties": {
            "metrics": [
                [ { "expression": "SUM(METRICS())", "label": "[avg: ${AVG}]
 Expression1", "id": "e1", "stat": "Sum" } ],
                [ "AWS/EC2", "CPUUtilization", { "id": "m2", "accountId":
 "5555666677778888", "region": "us-east-1", "label": "[avg: ${AVG}] ApplicationALabel
 " } ],
                [ ".", ".", { "id": "m1", "accountId": "9999000011112222", "region":
 "eu-west-1", "label": "[avg: ${AVG}] ApplicationBLabel" } ]
            ],
            "view": "timeSeries",
            "region":"us-east-1", ---> home region of the metric. Not present in above
 example
            "stacked": false,
            "stat": "Sum",
            "period": 300,
            "title": "Cross account example"
        }
    }
]
}
```

경보 위젯을 보여주는 예제입니다.

```
{
    "type": "metric",
    "x": 6,
    "y": 0,
    "width": 6,
    "height": 6,
    "properties": {
        "accountID": "111122223333",
        "title": "over50",
        "annotations": {
            "alarms": [
                "arn:aws:cloudwatch:us-east-1:379642911888:alarm:over50"
            ]
        },
        "view": "timeSeries",
```

```
        "stacked": false
    }
}
```

이 예는 CloudWatch Logs Insights 위젯에 대한 것입니다.

```
{
    "type": "log",
    "x": 0,
    "y": 6,
    "width": 24,
    "height": 6,
    "properties": {
        "query": "SOURCE 'route53test' | fields @timestamp, @message\n| sort @timestamp
 desc\n| limit 20",
        "accountId": "111122223333",
        "region": "us-east-1",
        "stacked": false,
        "view": "table"
    }
}
```

프로그래밍 방식으로 대시보드를 생성하는 또 다른 방법은 먼저 AWS Management Console에서 대시보드를 생성한 다음 이 대시보드의 JSON 소스를 복사하는 것입니다. 이렇게 하려면 대시보드를 로드하고 작업, 소스 보기/편집을 선택합니다. 그런 다음 이 대시보드 JSON을 복사해 템플릿으로 사용하여 유사한 대시보드를 생성할 수 있습니다.

대시보드 변수를 사용하여 유연한 대시보드 생성

대시보드 변수를 사용하여 대시보드 내의 입력 필드 값에 따라 여러 위젯에 서로 다른 콘텐츠를 빠르게 표시하는 유연한 대시보드를 생성할 수 있습니다. 예를 들어 서로 다른 Lambda 함수 또는 Amazon EC2 인스턴스 ID 사이를 빠르게 전환하는 대시보드 또는 서로 다른 AWS 리전으로 전환하는 대시보드를 만들 수 있습니다.

변수를 사용하는 대시보드를 생성한 후에는 동일한 변수 패턴을 기존의 다른 대시보드에 복사할 수 있습니다.

대시보드 변수를 사용하면 대시보드를 사용하는 사람의 운영 워크플로가 개선됩니다. 또한 유사한 대시보드를 여러 개 만드는 대신 하나의 대시보드에서 대시보드 변수를 사용하기 때문에 비용을 절감할 수 있습니다.

> **Note**
>
> 대시보드 변수가 포함된 대시보드를 공유하는 경우 공유한 사람은 변수 값을 변경할 수 없습니다.

대시보드 변수 유형

대시보드 변수는 속성 변수 또는 패턴 변수일 수 있습니다.

- 속성 변수는 대시보드의 모든 위젯에서 속성의 모든 인스턴스를 변경합니다. 이 속성은 대시보드의 JSON 소스에 있는 모든 JSON 속성(예: `region`)이 될 수 있습니다. 또는 `InstanceID` 또는 `FunctionName`과 같은 지표의 측정기준 이름일 수도 있습니다.

  속성 변수를 사용하는 자습서는 자습서: 함수 이름을 변수로 사용하여 Lambda 대시보드 생성 섹션을 참조하세요.

  대시보드의 JSON 소스에 대한 자세한 내용은 Dashboard Body Structure and Syntax를 참조하세요. CloudWatch 콘솔에서 작업, 소스 보기/편집을 선택하면 사용자 지정 대시보드의 JSON 소스를 볼 수 있습니다.
- 패턴 변수는 정규 표현식 패턴을 사용하여 JSON 속성의 전체 또는 특정 부분만 변경합니다.

  패턴 변수를 사용하는 자습서는 자습서: 정규 표현식 패턴을 사용하여 리전 간을 전환하는 대시보드 생성하기 섹션을 참조하세요.

속성 변수는 대부분의 사용 사례에 적용되며 설정하기가 덜 복잡합니다.

주제

- 자습서: 함수 이름을 변수로 사용하여 Lambda 대시보드 생성
- 자습서: 정규 표현식 패턴을 사용하여 리전 간을 전환하는 대시보드 생성하기
- 변수를 다른 대시보드로 복사

자습서: 함수 이름을 변수로 사용하여 Lambda 대시보드 생성

이 절차의 단계에서는 속성 변수를 사용하여 다양한 지표 그래프를 표시하는 유연한 대시보드를 생성하는 방법을 설명합니다. 여기에는 대시보드의 드롭다운 선택 상자가 포함되어 있어 모든 그래프의 지표를 서로 다른 Lambda 함수 간에 전환하는 데 사용할 수 있습니다.

이 유형의 대시보드에 대한 다른 사용 사례 예로는 `InstanceId`를 변수로 사용하여 인스턴스 ID에 대한 드롭다운이 있는 지표 대시보드를 만드는 것이 있습니다. 또는 `region`을 변수로 사용하여 다른 지역의 동일한 지표 세트를 표시하는 대시보드를 생성할 수 있습니다.

대시보드 속성 변수를 사용하여 유연한 Lambda 대시보드를 만들려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드, 대시보드 생성을 선택합니다.
3. 대시보드의 이름을 입력하고 대시보드 생성을 선택합니다.
4. Lambda 함수에 대한 지표를 표시하는 위젯을 대시보드에 추가합니다. 이러한 위젯을 생성할 때 위젯 지표에 대해 Lambda, 함수 이름별을 지정합니다. 함수의 경우 이 대시보드에 포함하려는 Lambda 함수 중 하나를 지정합니다.

   대시보드에 위젯을 추가하는 방법에 대한 자세한 내용은 CloudWatch 대시보드에서 위젯 생성 및 작업 섹션을 참조하세요.
5. 위젯을 추가한 후 대시보드를 보면서 작업, 변수, 변수 생성하기를 선택합니다.
6. 속성 변수를 선택합니다.
7. 변수가 변경되는 속성에 대해 FunctionName을 선택합니다.
8. 입력 유형의 경우 해당 사용 사례에서는 메뉴 선택(드롭다운)을 선택하는 것이 좋습니다. 이렇게 하면 대시보드에 드롭다운 메뉴가 생성되어 지표를 표시할 Lambda 함수 이름을 선택할 수 있습니다.

   변수에 대해 두세 가지 값만 전환하는 대시보드의 경우 라디오 버튼을 선택하는 것이 좋습니다.

   변수 값을 입력하거나 붙여넣으려면 텍스트 입력을 선택합니다. 이 옵션에는 드롭다운 목록이나 라디오 버튼이 포함되어 있지 않습니다.
9. 메뉴 선택(드롭다운)을 선택하면 값을 입력하여 메뉴를 채울지, 아니면 지표 검색을 사용하여 메뉴를 채울지 선택해야 합니다. 이 사용 사례에서는 많은 수의 Lambda 함수가 있고 모든 함수를 수동으로 입력하고 싶지 않다고 가정해 보겠습니다. 지표 검색 결과 사용을 선택한 후 다음을 수행합니다.

   a. 사전 빌드된 쿼리, Lambda, 오류를 선택합니다.

      (오류를 선택해도 대시보드에 오류 지표가 추가되지는 않지만 FunctionName 변수 선택 상자가 빠르게 채워집니다.)

   b. 함수 이름별을 선택한 다음 검색을 선택합니다.

검색 버튼 아래에 FunctionName이 선택된 것을 볼 수 있습니다. 또한 입력 상자를 채우기 위해 발견된 FunctionName 측정기준 값의 수에 대한 메시지도 표시됩니다.

10. (선택 사항) 추가 설정을 보려면 보조 설정을 선택하고 다음 중 하나 이상을 수행하세요.
    - 변수 이름을 사용자 지정하려면 사용자 지정 변수 이름에 이름을 입력합니다.
    - 변수 입력 필드의 레이블을 사용자 지정하려면 입력 레이블에 레이블을 입력합니다.
    - 대시보드를 처음 열 때 이 변수의 기본값을 설정하려면 기본값에 기본값을 입력합니다.
11. 변수 추가를 선택합니다.

    대시보드 상단 근처에 FunctionName 드롭다운 선택 상자가 나타납니다. 이 상자에서 Lambda 함수를 선택하면 해당 변수를 사용하는 모든 위젯에 선택한 함수에 대한 정보가 표시됩니다.

    나중에 FunctionName 측정기준을 사용하여 Lambda 지표를 감시하는 위젯을 대시보드에 더 추가하면 해당 위젯이 자동으로 변수를 사용합니다.

자습서: 정규 표현식 패턴을 사용하여 리전 간을 전환하는 대시보드 생성하기

이 절차의 단계는 리전 간에 전환할 수 있는 유연한 대시보드를 만드는 방법을 설명합니다. 이 자습서에서는 속성 변수 대신 정규 표현식 패턴 변수를 사용합니다. 속성 변수를 사용하는 자습서는 자습서: 함수 이름을 변수로 사용하여 Lambda 대시보드 생성 섹션을 참조하세요.

많은 사용 사례에서 속성 변수를 사용하여 리전 간을 전환하는 대시보드를 생성할 수 있습니다. 하지만 리전 이름이 포함된 Amazon 리소스 이름(ARN)을 사용하는 위젯의 경우 패턴 변수를 사용하여 ARN 내에서 리전 이름을 변경해야 합니다.

대시보드 패턴 변수를 사용하여 유연한 다중 지역 대시보드를 만들려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드, 대시보드 생성을 선택합니다.
3. 대시보드의 이름을 입력하고 대시보드 생성을 선택합니다.
4. 대시보드에 위젯을 추가합니다. 리전별 데이터를 표시할 위젯을 추가할 때는 하나의 리전에만 표시되는 값으로 측정기준을 지정하지 마세요. 예를 들어 Amazon EC2 지표의 경우 InstanceID를 측정기준으로 사용하는 지표 대신 집계된 지표를 지정합니다.

대시보드에 위젯을 추가하는 방법에 대한 자세한 내용은 CloudWatch 대시보드에서 위젯 생성 및 작업 섹션을 참조하세요.

5. 위젯을 추가한 후 대시보드를 보면서 작업, 변수, 변수 생성하기를 선택합니다.
6. 패턴 변수를 선택합니다.
7. 변수가 변경되는 속성에 현재 대시보드 리전의 이름(예: **us-east-2**)을 입력합니다.

   해당 상자 아래의 레이블에 변수의 영향을 받는 위젯이 표시되어 있으면 올바른 리전을 입력한 것입니다.
8. 이 사용 사례의 경우 입력 유형에서 라디오 버튼을 선택합니다.
9. 입력이 채워지는 방식 정의에서 사용자 지정 값 목록 생성을 선택합니다.
10. 사용자 지정 값 생성의 경우 전환하려는 리전을 각 줄에 하나씩 입력합니다. 각 지역 뒤에 쉼표를 입력한 다음 해당 라디오 버튼에 표시할 레이블을 입력합니다. 예:

    us-east-1, N. Virginia

    us-east-2, Ohio

    eu-west-3, Paris

    사용자 지정 값을 입력하면 미리 보기 창이 업데이트되어 라디오 버튼의 모양이 표시됩니다.
11. (선택 사항) 추가 설정을 보려면 보조 설정을 선택하고 다음 중 하나 이상을 수행하세요.

    - 변수 이름을 사용자 지정하려면 사용자 지정 변수 이름에 이름을 입력합니다.
    - 변수 입력 필드의 레이블을 사용자 지정하려면 입력 레이블에 레이블을 입력합니다. 이 자습서에서는 **Region:**을 입력합니다.

      여기에 값을 입력하면 미리 보기 창이 업데이트되어 라디오 버튼의 모양이 표시됩니다.
    - 대시보드를 처음 열 때 이 변수의 기본값을 설정하려면 기본값에 기본값을 입력합니다.
12. 변수 추가를 선택합니다.

    대시보드가 나타나고 상단의 리전 라디오 버튼 옆에 리전: 레이블이 표시됩니다. 리전 간에 전환하면 변수를 사용하는 모든 위젯에 선택한 리전에 대한 정보가 표시됩니다.

변수를 다른 대시보드로 복사

유용한 변수가 있는 대시보드를 만든 후에는 이러한 변수를 기존의 다른 대시보드에 복사할 수 있습니다. 대시보드 변수에 대한 자세한 내용은 대시보드 변수를 사용하여 유연한 대시보드 생성 섹션을 참조하세요.

대시보드 변수를 다른 대시보드에 복사하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드를 선택한 다음 복사하려는 변수가 있는 대시보드의 이름을 선택합니다. 필요한 경우 문자열을 입력하여 이름이 일치하는 대시보드를 찾습니다.
3. 작업, 변수, 변수 관리를 선택합니다.
4. 복사하려는 변수 옆의 라디오 단추를 선택하고 다른 대시보드로 복사를 선택합니다.
5. 선택 상자를 선택하고 변수를 복사할 대시보드 이름을 입력합니다.
6. 대시보드 이름을 선택하고 변수 복사를 선택합니다.

CloudWatch 대시보드에서 위젯 생성 및 작업

이 단원의 주제를 참고하여 대시보드에서 그래프, 경보 및 텍스트 위젯을 생성하고 사용할 수 있습니다.

내용

CloudWatch 대시보드에서 그래프 추가 또는 제거

하나 이상의 지표를 포함하는 그래프를 CloudWatch 대시보드에 추가할 수 있습니다. 대시보드에는 선(Line), 누적 영역(Stacked area), 숫자(Number), 게이지(Gauge), 막대(Bar) 및 파이(Pie) 유형의 그래프를 추가할 수 있습니다. 그래프가 더 이상 필요 없는 경우 대시보드에서 그래프를 제거할 수 있습니다. 이 섹션의 절차에서는 대시보드에서 그래프를 추가 및 제거하는 방법을 설명합니다. 대시보드에서 그래프를 편집하는 방법에 대한 자세한 내용은 CloudWatch 대시보드에서 그래프 편집을 참조하세요.

대시보드에 그래프 추가

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. + 기호를 선택한 다음 대시보드에 추가할 그래프 유형을 선택한 후 다음을 선택합니다.

   - 선(Line), 누적 영역(Stacked area), 막대(Bar) 또는 파이(Pie)를 선택한 경우 지표(Metrics)를 선택합니다.
4. 찾아보기 탭에서 그래프로 표시할 지표를 검색하거나 찾아보고 원하는 지표를 선택합니다.
5. (선택 사항) 그래프의 시간 범위를 변경하려면 화면 상단에서 미리 정의된 시간 범위 중 하나를 선택합니다. 이 범위는 1시간(1h), 3시간(3h), 12시간(12h), 1일(1d), 3일(3d), 1주(1w) 중에서 선택할 수 있습니다.

   고유한 시간 범위를 설정하려면 사용자 지정을 선택합니다.

   - (선택 사항) 나중에 대시보드의 나머지 시간 범위가 변경되더라도 이 위젯에는 선택한 이 시간 범위가 계속 사용되게 하려면 시간 범위 유지를 선택합니다.
6. (선택 사항) 그래프의 위젯 유형을 변경하려면 미리 정의된 시간 범위 옆에 있는 드롭다운을 사용합니다.
7. (선택 사항) 그래프로 표시된 지표(Graphed metrics)에서 지표에 동적 레이블을 추가하고 지표의 레이블, 레이블 색상, 통계 및 기간을 변경할 수 있습니다. Y축에서 레이블의 위치를 왼쪽에서 오른쪽으로 결정할 수도 있습니다.

   a. 동적 레이블을 추가하려면 그래프로 표시된 지표(Graphed metrics)를 선택한 다음 동적 레이블 추가(Add dynamic labels)를 선택합니다. 동적 레이블은 그래프 범례에 지표에 대한 통계를 표시합니다. 동적 레이블은 대시보드나 그래프가 새로 고쳐지는 경우 자동으로 업데이트됩니다. 기본적으로 레이블에 추가하는 동적 값은 레이블 시작 부분에 표시됩니다. 자세한 내용은 동적 레이블 사용 단원을 참조하십시오.

   b. 지표의 색상을 변경하려면 지표 옆에 있는 색상 사각형을 선택합니다.

c. 통계를 변경하려면 통계(Statistic)에서 드롭다운을 선택한 다음 새 값을 선택합니다. 자세한 내용은 통계를 참조하세요.

d. 기간을 변경하려면 기간(Period) 열에서 드롭다운을 선택한 다음 새 값을 선택합니다.

8. 게이지 위젯을 만드는 경우 옵션 탭을 선택하고 게이지의 양쪽 끝에 사용할 최소값과 최대값을 지정해야 합니다.

9. (선택 사항) Y축을 사용자 지정하려면 옵션(Options)을 선택합니다. 레이블 필드의 왼쪽 Y축(Left Y-axis) 아래에서 사용자 지정 레이블을 추가할 수 있습니다. 또한 그래프가 Y축의 오른쪽에 값을 표시하는 경우 해당 레이블도 사용자 지정할 수 있습니다. 그래프에 지정한 값의 범위만 표시되도록 Y축 값에 최솟값과 최댓값을 설정할 수도 있습니다.

10. (선택 사항) 선형 또는 누적 영역 그래프에 가로 주석을 추가 또는 편집하거나 게이지 위젯에 임곗값을 추가하려면 옵션을 선택합니다.

    a. 가로 주석 또는 임곗값을 추가하려면 가로 주석 추가 또는 임곗값 추가를 선택합니다.

    b. 레이블에 주석의 레이블을 입력한 다음 확인 표시 아이콘을 선택합니다.

    c. 레이블(Label)에서 현재 값 옆에 있는 펜과 종이 아이콘을 선택하고 새 값을 입력합니다. 값을 입력한 후 확인 표시 아이콘을 선택합니다.

    d. 채우기(Fill)에서 드롭다운을 선택하고 주석에서 음영을 사용하는 방법을 지정합니다. 없음(None) , 위(Above), 사이(Between) 또는 아래(Below)를 선택할 수 있습니다. 채우기 색상을 변경하려면 주석 옆의 색상 정사각형을 선택합니다.

    e. 축(Axis)에서 주석을 Y축의 왼쪽 또는 오른쪽에 표시할지 여부를 지정합니다.

    f. 주석을 숨기려면 숨기려는 주석 옆의 확인란을 선택 취소합니다.

    g. 주석을 삭제하려면 작업(Actions)에서 X를 선택합니다.

> **Note**
>
> 이 단계를 반복하여 동일한 그래프 또는 게이지에 여러 개의 가로 주석 또는 임곗값을 추가할 수 있습니다.

11. (선택 사항) 세로 주석을 추가 또는 편집하려면 옵션(Options)을 선택합니다.

    a. 세로 주석을 추가하려면 세로 주석 추가(Add vertical annotation)를 선택합니다.

    b. 레이블(Label)에서 현재 주석 옆에 있는 펜과 종이 아이콘을 선택하고 새 주석을 입력합니다. 날짜와 시간만 표시하려면 레이블 필드를 비워둡니다.

    c. 날짜(Date)에서 현재 날짜 및 시간을 선택한 후 새 날짜 및 시간을 입력합니다.

d. 채우기(Fill)에서 드롭다운을 선택하고 주석에서 음영을 사용하는 방법을 지정합니다. 없음(None) , 위(Above), 사이(Between) 또는 아래(Below)를 선택할 수 있습니다. 채우기 색상을 변경하려면 주석 옆의 색상 정사각형을 선택합니다.

e. 주석을 숨기려면 숨기려는 주석 옆의 확인란을 선택 취소합니다.

f. 주석을 삭제하려면 작업(Actions)에서 X를 선택합니다.

> **Note**
>
> 이 단계를 반복하여 동일한 그래프에 여러 세로 주석을 추가할 수 있습니다.

12. 위젯 생성을 선택합니다.
13. 대시보드 저장을 선택합니다.

대시보드에서 그래프를 제거하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 제거하려는 그래프 오른쪽 상단에서 위젯 작업(Widget actions)을 선택한 다음 삭제(Delete)를 선택합니다.
4. 대시보드 저장을 선택합니다.

CloudWatch 대시보드에서 수동으로 지표 그래프 생성

지표가 지난 14일간 데이터를 게시하지 않은 경우 CloudWatch 대시보드에서 그래프에 추가할 지표를 검색할 때 이 지표를 찾을 수 없습니다. 다음 단계에 따라 기존 그래프에 지표를 수동으로 추가할 수 있습니다.

그래프에 검색에서 찾을 수 없는 지표를 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 대시보드에 지표를 추가하고자 하는 그래프가 이미 있어야 합니다. 아직 없는 경우 그래프를 생성한 다음 지표를 추가합니다. 자세한 내용은 CloudWatch 대시보드에서 그래프 추가 또는 제거 단원을 참조하세요.
4. 작업, 소스 보기/편집을 선택합니다.

JSON 블록이 표시됩니다. 블록은 대시보드 및 해당 콘텐츠의 위젯을 지정합니다. 다음은 그래프 하나를 정의하는 이 블록 일부의 예제입니다.

```
{
            "type": "metric",
            "x": 0,
            "y": 0,
            "width": 6,
            "height": 3,
            "properties": {
                "view": "singleValue",
                "metrics": [
                    [ "AWS/EBS", "VolumeReadOps", "VolumeId",
 "vol-1234567890abcdef0" ]
                ],
                "region": "us-west-1"
            }
        },
```

이 예제에서 다음 섹션은 이 그래프에 표시되는 지표를 정의합니다.

```
[ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ]
```

5. 아직 없는 경우 닫는 괄호 뒤에 쉼표를 추가한 후, 쉼표 뒤에 비슷한 괄호로 묶은 섹션을 추가합니다. 이 새 섹션에서 네임스페이스, 지표 이름 및 그래프에 추가하려는 지표의 필요한 모든 측정기준을 지정합니다. 다음은 예입니다.

```
[ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ],
[ "MyNamespace", "MyMetricName", "DimensionName", "DimensionValue" ]
```

JSON으로 지표 형식 지정에 대한 자세한 정보는 지표 위젯 객체의 속성을 참조하세요.

6. 업데이트를 선택합니다.

기존 그래프 작업

이 단원의 절차에 따라 기존 대시보드 그래프 위젯을 편집하고 수정할 수 있습니다.

주제

- CloudWatch 대시보드에서 그래프 편집

- CloudWatch 대시보드에서 그래프 이동 또는 크기 조정
- CloudWatch 대시보드에서 그래프 이름 변경

CloudWatch 대시보드에서 그래프 편집

CloudWatch 대시보드에 추가하는 그래프를 편집할 수 있습니다. 그래프의 제목, 통계 또는 기간을 변경할 수 있습니다. 그래프에서 지표를 추가, 업데이트 및 제거할 수 있습니다. 그래프에 두 개 이상의 지표가 포함되어 있는 경우, 사용하지 않는 지표를 숨기면 혼란을 줄일 수 있습니다. 이 섹션의 절차에서는 대시보드에서 그래프를 편집하는 방법에 대해 설명합니다. 그래프 생성에 대한 자세한 내용은 CloudWatch 대시보드에서 그래프 추가 또는 제거를 참조하세요.

New interface

대시보드에서 그래프를 편집하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 편집하려는 그래프의 오른쪽 상단 모서리에서 위젯 작업(Widget actions)을 선택한 다음 편집(Edit)을 선택합니다.
4. 그래프 제목을 변경하려면 현재 제목 옆의 펜과 종이 아이콘을 선택합니다. 새 제목을 입력한 다음 적용(Apply)을 선택합니다.
5. (선택 사항) 그래프의 시간 범위를 변경하려면 그래프 상단에서 미리 정의된 시간 범위 중 하나를 선택합니다. 이 범위는 1시간(1h), 3시간(3h), 12시간(12h), 1일(1d), 3일(3d), 1주(1w) 중에서 선택할 수 있습니다.

   고유한 시간 범위를 설정하려면 사용자 지정을 선택합니다.

   - (선택 사항) 나중에 대시보드의 나머지 시간 범위가 변경되더라도 이 위젯에는 선택한 이 시간 범위가 계속 사용되게 하려면 시간 범위 유지를 선택합니다.
6. 그래프의 위젯 유형을 변경하려면 미리 정의된 시간 범위 옆에 있는 드롭다운을 사용합니다.
7. 그래프로 표시된 지표(Graphed metrics)에서 지표에 동적 레이블을 추가하고 지표의 레이블, 레이블 색상, 통계 및 기간을 변경할 수 있습니다. Y축에서 레이블의 위치를 왼쪽에서 오른쪽으로 결정할 수도 있습니다.

   a. 지표에 대한 동적 레이블을 추가하려면 동적 레이블(Dynamic labels)을 선택합니다. 동적 레이블은 그래프 범례에 지표에 대한 통계를 표시합니다. 동적 레이블은 대시보드나 그래프가 새로 고쳐지는 경우 자동으로 업데이트됩니다. 기본적으로 레이블에 추가하는 동적

값은 레이블 시작 부분에 표시됩니다. 자세한 내용은 동적 레이블 사용 단원을 참조하십시오.

b. 지표의 색상을 변경하려면 지표 옆에 있는 색상 사각형을 선택합니다.

c. 통계를 변경하려면 통계(Statistic) 열에서 통계 값을 선택한 다음 새 값을 선택합니다. 자세한 내용은 Statistics 단원을 참조하십시오.

d. 기간을 변경하려면 기간(Period) 열에서 기간 값을 선택한 다음 새 값을 선택합니다.

8. 가로 주석을 추가 또는 편집하려면 옵션(Options)을 선택합니다.

a. 가로 주석을 추가하려면 가로 주석 추가(Add horizontal annotation)를 선택합니다.

b. 레이블(Label)에서 현재 주석 옆에 있는 펜 앤 페이퍼 아이콘을 선택합니다. 그런 다음 새 주석을 입력합니다. 주석을 입력한 후 확인 표시 아이콘을 선택합니다.

c. 값(Value)에서 현재 메트릭 값 옆에 있는 펜 앤 페이퍼 아이콘을 선택합니다. 그런 다음 새 지표 값을 입력합니다. 값을 입력한 후 확인 표시 아이콘을 선택합니다.

d. 채우기(Fill)에서 열 아래의 드롭다운을 선택한 다음 주석에서 음영을 사용하는 방법을 지정합니다. 없음(None) , 위(Above), 사이(Between) 또는 아래(Below)를 선택할 수 있습니다. 사이(Between)를 선택한 경우 다른 새 레이블 및 값 필드가 나타납니다.

> **Tip**
> 주석 옆의 유색 정사각형을 선택하여 채우기 색상을 변경할 수 있습니다.

e. 축(Axis)에서 주석을 Y축의 왼쪽 또는 오른쪽에 표시할지 여부를 지정합니다.

f. 주석을 숨기려면 그래프에서 숨기려는 주석 옆의 확인란을 선택 취소합니다.

g. 주석을 삭제하려면 작업(Actions) 열에서 x를 선택합니다.

> **Note**
> 이 단계를 반복하여 동일한 그래프에 여러 가로 주석을 추가할 수 있습니다.

9. 세로 주석을 추가 또는 편집하려면 옵션(Options)을 선택합니다.

a. 세로 주석을 추가하려면 세로 주석 추가(Add vertical annotation)를 선택합니다.

b. 레이블(Label)에서 현재 주석 옆에 있는 펜 앤 페이퍼 아이콘을 선택합니다. 그런 다음 새 주석을 입력합니다. 주석을 입력한 후 확인 표시 아이콘을 선택합니다.

> **Tip**
>
> 날짜와 시간만 표시하려면 레이블 필드를 비워둡니다.

c. 날짜(Date)에서 현재 날짜 및 시간을 선택합니다. 그런 다음 새 날짜 및 시간을 입력합니다.

d. 채우기(Fill)에서 열 아래의 드롭다운을 선택한 다음 주석에서 음영을 사용하는 방법을 지정합니다. 없음(None) , 위(Above), 사이(Between) 또는 아래(Below)를 선택할 수 있습니다. 사이(Between)를 선택한 경우 새 레이블 및 값 필드가 나타납니다.

> **Tip**
>
> 주석 옆의 색상 정사각형을 선택하여 채우기 색상을 변경할 수 있습니다.

> **Note**
>
> 이 단계를 반복하여 동일한 그래프에 여러 세로 주석을 추가할 수 있습니다.

e. 주석을 숨기려면 그래프에서 숨기려는 주석 옆의 확인란을 선택 취소합니다.

f. 주석을 삭제하려면 작업(Actions) 열에서 x를 선택합니다.

10. Y축을 사용자 지정하려면 옵션(Options)을 선택합니다. 왼쪽 Y축(Left Y-axis)에 레이블(Label)에 대한 사용자 지정 레이블을 입력할 수 있습니다. 그래프가 오른쪽 Y축의 값을 표시하는 경우 해당 레이블도 사용자 지정할 수 있습니다. 그래프에 지정한 값의 범위만 표시되도록 Y축 값에 최솟값과 최댓값을 설정할 수도 있습니다.

11. 변경을 마치면 위젯 업데이트(Update widget)를 선택합니다.

그래프 범례의 위치를 숨기거나 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 편집하려는 그래프 오른쪽 상단 모서리에서 위젯 작업(Widget actions)을 선택합니다. 범례(Legend)를 선택하고 숨김(Hidden), 하단(Bottom) 또는 오른쪽(Right)을 선택합니다.

대시보드의 그래프에서 지표를 일시적으로 숨기려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 그래프의 바닥글에서 숨기려는 지표의 색상 정사각형을 선택합니다. 색상 사각형에 마우스를 가져가면 X가 표시되고 정사각형을 선택하면 회색으로 바뀝니다.
4. 숨겨진 지표를 복원하려면 회색 사각형의 X를 지웁니다.

Original interface

대시보드에서 그래프를 편집하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 편집하려는 그래프의 오른쪽 상단 모서리에 마우스를 가져갑니다. 위젯 작업(Widget Actions)을 선택한 다음, 편집(Edit)을 선택합니다.
4. 그래프 제목을 변경하려면 현재 제목 옆의 펜과 종이 아이콘을 선택한 다음 새 제목을 입력합니다.
5. 그래프의 시간 범위를 변경하려면 그래프 상단 영역에서 미리 정의된 시간 범위 중 하나를 선택합니다. 이 범위는 1시간에서 1주까지 다양합니다(1시간, 3시간, 12시간, 1일, 3일, 1주).
   - 고유한 시간 범위를 설정하려면 사용자 지정(Custom)을 선택합니다.
6. 그래프의 위젯 유형을 변경하려면 그래프 옵션(Graph options) 탭을 선택합니다. 선(Line), 누적 영역(Stacked area), 숫자(Number), 막대(Bar), 파이(Pie)를 선택할 수 있습니다.

   > **Tip**
   > 미리 정의된 시간 범위 옆에 있는 드롭다운을 선택하여 그래프의 위젯 유형을 변경할 수도 있습니다.

7. 그래프로 표시된 지표(Graphed metrics)에서 지표에 동적 레이블을 추가하고 지표의 레이블, 레이블 색상, 통계 및 기간을 변경할 수 있습니다. Y축에서 레이블의 위치를 왼쪽에서 오른쪽으로 결정할 수도 있습니다.

   a. 지표에 대한 동적 레이블을 추가하려면 동적 레이블(Dynamic labels)을 선택합니다. 동적 레이블은 그래프 범례에 지표에 대한 통계를 표시합니다. 동적 레이블은 대시보드나 그래프가 새로 고쳐지는 경우 자동으로 업데이트됩니다. 기본적으로 레이블에 추가하는 동적

값은 레이블 시작 부분에 표시됩니다. 자세한 내용은 동적 레이블 사용 단원을 참조하십시오.

b. 지표의 색상을 변경하려면 지표 옆에 있는 색상 사각형을 선택합니다.

c. 통계를 변경하려면 통계(Statistic) 열에서 통계 값을 선택한 다음 새 값을 선택합니다. 자세한 내용은 Statistics 단원을 참조하십시오.

d. 기간을 변경하려면 기간(Period) 열에서 기간 값을 선택한 다음 새 값을 선택합니다.

8. 가로 주석을 추가 또는 편집하려면 그래프 옵션(Graph options)을 선택합니다.

a. 가로 주석을 추가하려면 가로 주석 추가(Add horizontal annotation)를 선택합니다.

b. 레이블(Label)에서 현재 주석 옆에 있는 연필 아이콘을 선택합니다. 그런 다음 새 주석을 입력합니다. 주석을 입력한 후 확인 표시 아이콘을 선택합니다.

c. 값(Value)에서 현재 메트릭 값 옆에 있는 연필 아이콘을 선택합니다. 그런 다음 새 지표 값을 입력합니다. 값을 입력한 후 확인 표시 아이콘을 선택합니다.

d. 채우기(Fill)에서 열 아래의 드롭다운을 선택한 다음 주석에서 음영을 사용하는 방법을 지정합니다. 없음(None) , 위(Above), 사이(Between) 또는 아래(Below)를 선택할 수 있습니다. 사이(Between)를 선택한 경우 새 레이블 및 값 필드가 나타납니다.

> **Tip**
>
> 주석 옆의 색상 정사각형을 선택하여 채우기 색상을 변경할 수 있습니다.

e. 축(Axis)에서 주석을 Y축의 왼쪽 또는 오른쪽에 표시할지 여부를 지정합니다.

f. 주석을 숨기려면 그래프에서 숨기려는 주석 옆의 확인란을 선택 취소합니다.

g. 주석을 삭제하려면 작업(Actions) 열에서 x를 선택합니다.

> **Note**
>
> 이 단계를 반복하여 동일한 그래프에 여러 가로 주석을 추가할 수 있습니다.

9. 세로 주석을 추가 또는 편집하려면 그래프 옵션(Graph options)을 선택합니다.

a. 세로 주석을 추가하려면 세로 주석 추가(Add vertical annotation)를 선택합니다.

b. 레이블(Label)에서 현재 주석 옆에 있는 연필 아이콘을 선택합니다. 그런 다음 새 주석을 입력합니다. 주석을 입력한 후 확인 표시 아이콘을 선택합니다.

> **Tip**
>
> 날짜와 시간만 표시하려면 레이블 필드를 비워둡니다.

c. 날짜(Date)에서 현재 날짜 및 시간 옆에 있는 연필 아이콘을 선택합니다. 그런 다음 새 날짜 및 시간을 입력합니다.

d. 채우기(Fill)에서 열 아래의 드롭다운을 선택한 다음 주석에서 음영을 사용하는 방법을 지정합니다. 없음(None) , 위(Above), 사이(Between) 또는 아래(Below)를 선택할 수 있습니다. 사이(Between)를 선택한 경우 새 레이블 및 값 필드가 나타납니다.

> **Tip**
>
> 주석 옆의 색상 정사각형을 선택하여 채우기 색상을 변경할 수 있습니다.

> **Note**
>
> 이 단계를 반복하여 동일한 그래프에 여러 세로 주석을 추가할 수 있습니다.

e. 주석을 숨기려면 그래프에서 숨기려는 주석 옆의 확인란을 선택 취소합니다.

f. 주석을 삭제하려면 작업(Actions) 열에서 x를 선택합니다.

10. Y축을 사용자 지정하려면 그래프 옵션(Graph options)을 선택합니다. 왼쪽 Y축(Left Y-axis)에 레이블(Label)에 대한 사용자 지정 레이블을 입력할 수 있습니다. 그래프가 오른쪽 Y축의 값을 표시하는 경우 해당 레이블도 사용자 지정할 수 있습니다. 그래프에 지정한 값의 범위만 표시되도록 Y축 값에 최솟값과 최댓값을 설정할 수도 있습니다.

11. 변경을 마치면 위젯 업데이트(Update widget)를 선택합니다.

그래프 범례의 위치를 숨기거나 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 편집하려는 그래프 오른쪽 상단 모서리에 마우스를 가져간 다음 위젯 작업(Widget actions)을 선택합니다. 범례(Legend)를 선택하고 숨김(Hidden), 하단(Bottom) 또는 오른쪽(Right)을 선택합니다.

대시보드의 그래프에서 지표를 일시적으로 숨기려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 그래프의 바닥글에서 숨기려는 지표의 색상 정사각형을 선택합니다. 색상 사각형에 마우스를 가져가면 X가 표시되고 정사각형을 선택하면 회색으로 바뀝니다.
4. 숨겨진 지표를 복원하려면 회색 사각형의 X를 지웁니다.

CloudWatch 대시보드에서 그래프 이동 또는 크기 조정

CloudWatch 대시보드에서 그래프를 정렬하고 그래프 크기를 조정할 수 있습니다.

대시보드에서 그래프를 이동시키려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 다음 중 하나를 수행합니다.
   - 선택 아이콘이 나타날 때까지 그래프 제목에 마우스 포인터를 둡니다. 원하는 그래프를 선택하고 대시보드의 새로운 위치로 드래그합니다.
   - 위젯을 대시보드의 왼쪽 상단 또는 왼쪽 하단으로 이동하려면 위젯의 오른쪽 상단에 있는 세로 줄임표를 선택하여 위젯 작업 메뉴를 엽니다. 그런 다음 이동을 선택하고 위젯을 이동할 위치를 선택합니다.
4. 대시보드 저장을 선택합니다.

그래프 크기를 조정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 크기를 늘리거나 줄이려면 그래프에 마우스 포인터를 두고 그래프의 오른쪽 하단 모서리를 드래그합니다. 원하는 크기가 되면 모서리 드래그를 중지합니다.
4. 대시보드 저장(Save dashboard)을 선택합니다.

그래프를 일시적으로 확대하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 그래프를 선택합니다. 또는 그래프 제목에 마우스 포인터를 놓고 위젯 작업(Widget actions)과 확대(Enlarge)를 선택합니다.

CloudWatch 대시보드에서 그래프 이름 변경

CloudWatch가 대시보드에서 그래프에 할당한 기본 이름을 변경할 수 있습니다.

대시보드에서 그래프 이름을 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 그래프 제목 위로 마우스를 이동하고 위젯 작업(Widget actions), 편집(Edit)을 선택합니다.
4. 그래프 편집(Edit graph) 화면의 위쪽에서 그래프의 제목을 선택합니다.
5. 제목(Title)에 새 이름을 입력하고 확인(Ok)(체크 표시)을 선택합니다. 그래프 편집(Edit graph) 화면의 우측 하단 모서리에서 위젯 업데이트(Update widget)를 선택합니다.

CloudWatch 대시보드에 지표 탐색기 위젯 추가

지표 탐색기 위젯에는 동일한 태그가 있거나 인스턴스 유형과 같은 동일한 리소스 속성을 공유하는 여러 리소스의 그래프가 포함됩니다. 이러한 위젯은 일치하는 리소스가 생성되거나 삭제될 때 최신 상태로 유지됩니다. 대시보드에 지표 탐색기 위젯을 추가하면 환경 관련 문제를 더욱 효율적으로 해결할 수 있습니다.

예를 들어 프로덕션 또는 테스트와 같은 환경을 나타내는 태그를 할당하여 EC2 인스턴스 플릿을 모니터링할 수 있습니다. 그런 다음, 이러한 태그를 사용하여 `CPUUtilization`과 같은 운영 지표를 필터링하고 집계함으로써 각 태그와 연결된 EC2 인스턴스의 상태 및 성능을 파악할 수 있습니다.

다음 단계에서는 콘솔을 사용하여 대시보드에 지표 탐색기 위젯을 추가하는 방법을 설명합니다. 프로그래밍 방식으로 또는 AWS CloudFormation을 사용하여 위젯을 추가할 수도 있습니다. 자세한 내용은 지표 탐색기 위젯 객체 정의 및 AWS::CloudWatch::Dashboard 단원을 참조하세요.

대시보드에 지표 탐색기 위젯을 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 지표 탐색기 위젯을 추가하려는 대시보드의 이름을 선택합니다.

4. + 기호를 선택합니다.
5. 탐색기(Explorer)를 선택한 후 다음(Next)을 선택합니다.

> **Note**
>
> 지표 탐색기 위젯을 추가하려면 새 대시보드 보기를 옵트인해야 합니다. 옵트인하려면 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 페이지 상단의 배너에서 새 인터페이스 사용해 보기(try out the new interface)를 선택합니다.

6. 다음 중 하나를 수행합니다.
   - 템플릿을 사용하려면 미리 채워진 탐색기 위젯(Pre-filled Explorer widget)을 선택한 다음, 사용할 템플릿을 선택합니다.
   - 사용자 지정 시각화를 생성하려면 빈 탐색기 위젯(Empty Explorer widget)을 선택합니다.
7. 생성(Create)을 선택합니다.

   템플릿을 사용한 경우 위젯이 선택한 지표와 함께 대시보드에 나타납니다. 탐색기 위젯 및 대시보드가 마음에 들면 대시보드 저장을 선택합니다.

   템플릿을 사용하지 않은 경우 다음 단계를 계속 진행합니다.
8. 탐색기(Explorer) 아래의 새 위젯에 있는 지표(Metrics) 상자에서 단일 지표 또는 서비스에서 사용 가능한 모든 지표를 선택합니다.

   지표를 선택한 후 필요에 따라 이 단계를 반복하여 더 많은 지표를 추가할 수 있습니다.
9. 선택한 각 지표에 대해 CloudWatch는 지표 이름 바로 뒤에 사용할 통계를 표시합니다. 이를 변경하려면 통계 이름을 선택한 다음, 원하는 통계를 선택합니다.
10. 다음에서(From)에서 결과를 필터링할 태그 또는 리소스 속성을 선택합니다.

    이 작업을 수행한 후 필요에 따라 이 단계를 반복하여 더 많은 태그 또는 리소스 속성을 선택할 수 있습니다.

    EC2 인스턴스 유형 두 개와 같이 속성이 동일한 값을 여러 개 선택하는 경우 탐색기에는 선택한 속성 중 하나와 일치하는 리소스가 모두 표시됩니다. 이는 OR 연산으로 처리됩니다.

    `Production` 태그 및 M5 인스턴스 유형과 같이 서로 다른 속성 또는 태그를 선택하는 경우 이러한 모든 선택 사항과 일치하는 리소스만 표시됩니다. 이는 AND 연산으로 처리됩니다.

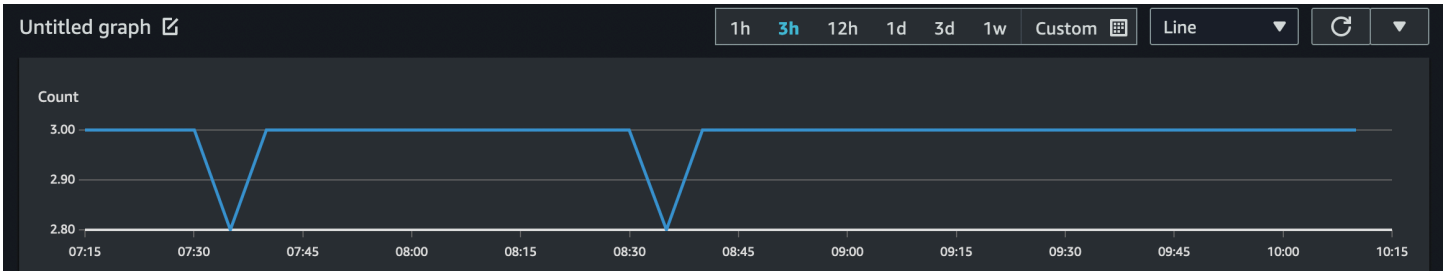
11. (선택 사항) 집계 기준(Aggregate by)에서 지표 집계에 사용할 통계를 선택합니다. 그런 다음, 대상(for) 옆에 있는 목록에서 지표를 집계하는 방법을 선택합니다. 현재 표시된 모든 리소스를 함께 집계하거나 단일 태그 또는 리소스 속성을 기준으로 집계할 수 있습니다.

    선택하는 집계 방법에 따라 결과가 단일 시계열 또는 다중 시계열이 될 수 있습니다.

12. 분할 기준(Split by)에서 다중 시계열의 단일 그래프를 여러 그래프로 분할하도록 선택할 수 있습니다. 분할 기준(Split by)에서 선택하는 다양한 기준에 따라 분할을 수행할 수 있습니다.
13. 그래프 옵션(Graph options)에서 기간, 그래프 유형, 범례 배치, 레이아웃을 변경하여 그래프를 구체화할 수 있습니다.
14. 탐색기 위젯 및 대시보드가 마음에 들면 대시보드 저장을 선택합니다.

CloudWatch 대시보드에서 선 위젯 추가 또는 제거

선 위젯을 사용하면 기간 동안의 지표를 비교할 수 있습니다. 위젯의 미니 맵 확대/축소 기능을 사용하여 확대/축소된 뷰 간에 변경하지 않고 선 그래프의 섹션을 검사할 수도 있습니다. 이 섹션의 절차에서는 CloudWatch 대시보드에서 선 위젯을 추가 및 제거하는 방법에 대해 설명합니다. 선 그래프와 함께 위젯의 미니 맵 확대/축소 기능을 사용하는 방법에 대한 자세한 내용은 선 또는 누적 영역 그래프 확대를 참조하세요.



대시보드에 선 위젯 추가

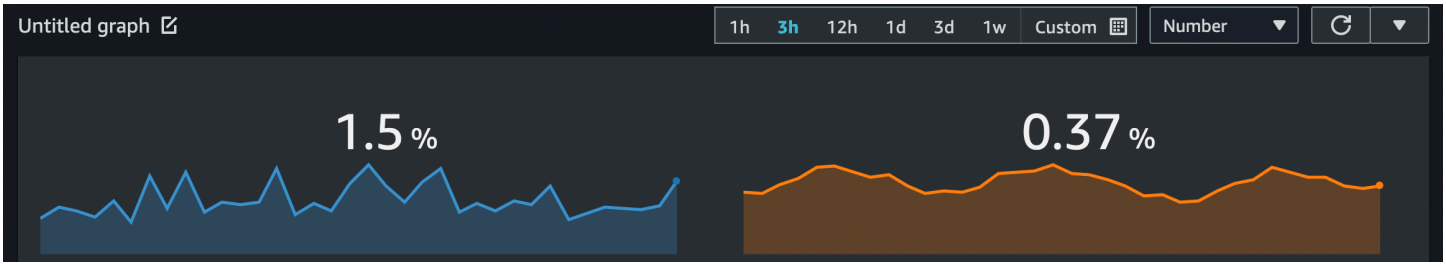
1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. + 기호를 선택하고 행(Line)을 선택합니다.
4. [지표(Metrics)]를 선택합니다.
5. 검색(Browse)을 선택한 다음 그래프로 표시할 지표를 선택합니다.
6. 위젯 생성(Create widget)을 선택한 다음 대시보드 저장(Save dashboard)을 선택합니다.

대시보드에서 선 위젯 제거

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 제거하려는 선 위젯의 오른쪽 상단에서 위젯 작업(Widget actions)을 선택한 다음 삭제(Delete)를 선택합니다.
4. 대시보드 저장을 선택합니다.

CloudWatch 대시보드에서 숫자 위젯 추가 또는 제거

숫자 위젯을 사용하면 최신 지표 값과 추세가 나타나는 즉시 확인할 수 있습니다. 숫자 위젯에는 스파크라인 기능이 포함되어 있으므로 단일 그래프로 지표 추세의 상단 및 하단 절반을 시각화할 수 있습니다. 이 섹션의 절차에서는 CloudWatch 대시보드에서 숫자 위젯을 추가 및 제거하는 방법에 대해 설명합니다.



> **Note**
>
> 스파크라인 기능은 새 인터페이스에서만 지원됩니다. 숫자 위젯을 생성하면 스파크라인 기능이 자동으로 포함됩니다.

대시보드에 숫자 위젯을 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. + 기호를 선택하고 숫자(Number)를 선택합니다.
4. 찾아보기 탭에서 표시하려는 지표를 검색하거나 찾아봅니다.
5. (선택 사항) 스파크라인 기능의 색상을 변경하려면 그래프로 표시된 지표(Graphed metrics)를 선택하고 지표 레이블 옆에 있는 색상 상자를 선택합니다. 다른 색상을 선택하거나 6자리 16진수 색상 코드를 입력하면 색상을 지정할 수 있는 메뉴가 표시됩니다.

6. (선택 사항) 스파크라인 기능을 끄려면 옵션(Options)을 선택합니다. 스파크라인(Sparkline) 아래에서 확인란을 선택합니다.
7. (선택 사항) 숫자 위젯의 시간 범위를 변경하려면 위젯의 상단 영역에서 미리 정의된 시간 범위 중 하나를 선택합니다. 이 범위는 1시간(1h), 3시간(3h), 12시간(12h), 1일(1d), 3일(3d), 1주(1w) 중에서 선택할 수 있습니다.

   고유한 시간 범위를 설정하려면 사용자 지정을 선택합니다.

   - (선택 사항) 나중에 대시보드의 나머지 시간 범위가 변경되더라도 이 위젯에는 선택한 이 시간 범위가 계속 사용되게 하려면 시간 범위 유지를 선택합니다.
8. (선택 사항) 숫자 위젯에서 집계 표시(1시간, 3시간, 12시간, 1일, 3일 또는 1주).

   고유한 시간 범위를 설정하려면 사용자 지정을 선택합니다.

   - (선택 사항) 이 위젯에서 최근 값이 아닌 전체 시간 범위의 평균 지표 값을 표시하려면 옵션, 시간 범위 값에 전체 시간 범위의 값 표시를 차례로 선택합니다.
9. 위젯 생성(Create widget)을 선택한 다음 대시보드 저장(Save dashboard)을 선택합니다.

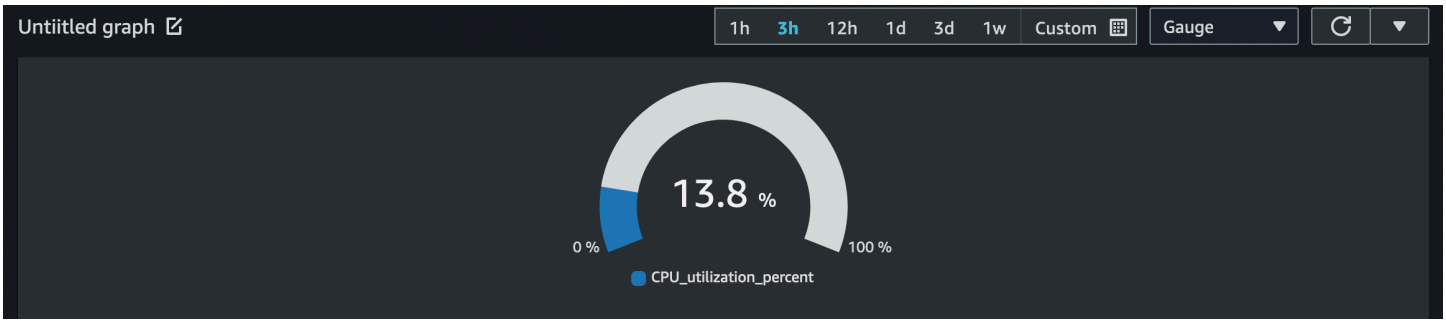
> **Tip**
>
> 대시보드 화면의 숫자 위젯에서 스파크라인 기능을 끌 수 있습니다. 수정하려는 숫자 위젯의 오른쪽 상단 모서리에서 위젯 작업(Widget actions)을 선택합니다. 스파크라인(Sparkline)을 선택한 다음 스파크라인 숨기기(Hide sparkline)를 선택합니다.

대시보드에서 숫자 위젯 제거

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음 삭제하려는 숫자 위젯이 포함된 대시보드를 선택합니다.
3. 제거하려는 숫자 위젯의 오른쪽 상단에서 위젯 작업(Widget actions)을 선택한 다음 삭제(Delete)를 선택합니다.
4. 대시보드 저장을 선택합니다.

CloudWatch 대시보드에서 게이지 위젯 추가 또는 제거

게이지 위젯을 사용하면 범위 사이에 있는 지표 값을 시각화할 수 있습니다. 예를 들어 게이지 위젯을 사용하여 백분율과 CPU 사용률을 그래프로 표시하여 발생하는 성능 문제를 관찰하고 진단할 수 있습니다. 이 섹션의 절차에서는 CloudWatch 대시보드에서 게이지 위젯을 추가 및 제거하는 방법에 대해 설명합니다.



> **Note**
>
> CloudWatch 콘솔의 새 인터페이스만 게이지 위젯 생성을 지원합니다. 이 위젯을 생성하는 경우 게이지 범위를 설정해야 합니다.

대시보드에 게이지 위젯 추가

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 대시보드 화면에서 + 기호를 선택한 다음 게이지(Gauge)를 선택합니다.
4. 검색(Browse)을 선택한 다음 그래프로 표시할 지표를 선택합니다.
5. 옵션(Options)을 선택합니다. 게이지 범위(Gauge range)에서 최솟값(Min)과 최댓값(Max)을 설정합니다. CPU 사용률과 같은 백분율의 경우 `Min` 값을 `0`으로 `Max` 값을 `100`으로 설정하는 것이 좋습니다.
6. (선택 사항) 게이지 위젯의 색상을 변경하려면 그래프로 표시된 지표(Graphed metrics)를 선택하고 지표 레이블 옆에 있는 색상 상자를 선택합니다. 다른 색상을 선택하거나 6자리 16진수 색상 코드를 입력하면 색상을 지정할 수 있는 메뉴가 표시됩니다.
7. 위젯 생성(Create widget)을 선택한 다음 대시보드 저장(Save dashboard)을 선택합니다.

대시보드에서 게이지 위젯 제거

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음 삭제하려는 게이지 위젯이 포함된 대시보드를 선택합니다.
3. 삭제하려는 게이지 위젯의 오른쪽 상단에서 위젯 작업(Widget actions)을 선택한 다음 삭제(Delete)를 선택합니다.
4. 대시보드 저장을 선택합니다.

CloudWatch 대시보드에 사용자 지정 위젯 추가

사용자 지정 위젯은 사용자 지정 파라미터를 사용하여 AWS Lambda 함수를 호출할 수 있는 CloudWatch 대시보드 위젯입니다. 이 위젯은 함수 호출 후 반환된 HTML 또는 JSON을 표시합니다. 사용자 지정 위젯은 대시보드에서 사용자 지정 데이터 보기를 구축하는 간단한 방법입니다. Lambda 코드를 작성하고 HTML을 생성할 수 있다면 유용한 사용자 지정 위젯을 생성할 수 있습니다. 또한 Amazon은 코드 없이도 생성할 수 있는 사전 구축된 사용자 지정 위젯을 몇 가지 제공합니다.

사용자 지정 위젯으로 사용할 Lambda 함수를 생성할 경우에는 함수 이름에 customWidget 접두사를 포함하는 것이 좋습니다. 이렇게 하면 대시보드에 사용자 지정 위젯을 추가할 때 어떤 Lambda 함수를 사용하는 것이 안전한지 알 수 있습니다.

사용자 지정 위젯은 대시보드의 다른 위젯처럼 동작합니다. 새로 고침 및 자동 새로 고침, 크기 조정, 이동할 수 있습니다. 사용자 지정 위젯은 대시보드의 시간 범위에 반응합니다.

CloudWatch 콘솔 교차 계정 기능을 설정한 경우 한 계정에서 생성한 사용자 지정 위젯을 다른 계정의 대시보드에 추가할 수 있습니다. 자세한 내용은 교차 계정 교차 리전 CloudWatch 콘솔 단원을 참조하세요.

또한 CloudWatch 대시보드 공유 기능을 사용하여 자체 웹 사이트에서 사용자 지정 위젯을 사용할 수도 있습니다. 자세한 내용은 CloudWatch 대시보드 공유 단원을 참조하세요.

주제

- 사용자 지정 위젯에 관한 세부 정보
- 보안 및 JavaScript
- 사용자 지정 위젯의 상호 작용
- 사용자 지정 위젯 생성
- 샘플 사용자 지정 위젯

사용자 지정 위젯에 관한 세부 정보

사용자 지정 위젯은 다음과 같이 작동합니다.

1. CloudWatch 대시보드는 위젯 코드가 포함된 Lambda 함수를 호출합니다. 그리고 위젯에 정의된 사용자 지정 파라미터를 전달합니다.
2. Lambda 함수는 HTML, JSON 또는 Markdown 문자열을 반환합니다. Markdown은 다음 형식의 JSON으로 반환됩니다.

   ```
   {"markdown":"markdown content"}
   ```

3. 대시보드는 반환된 HTML 또는 JSON을 표시합니다.

함수가 HTML을 반환하는 경우 대부분의 HTML 태그가 지원됩니다. Cascading Style Sheets(CSS) 스타일 및 Scalable Vector Graphics(SVG)를 사용하여 정교한 보기를 구축할 수 있습니다.

링크 및 테이블과 같은 HTML 요소의 기본 스타일은 CloudWatch 대시보드의 스타일 지정을 준수합니다. `<style>` 태그를 활용하여 인라인 스타일을 사용함으로써 이 스타일을 사용자 지정할 수 있습니다. 또한 `cwdb-no-default-styles` 클래스와 함께 단일 HTML 요소를 포함하여 기본 스타일을 비활성화할 수도 있습니다. 예를 들어 `<div class="cwdb-no-default-styles"></div>`는 기본 스타일을 비활성화합니다.

Lambda에 대한 사용자 지정 위젯의 모든 호출에는 Lambda 함수 개발자에게 유용한 컨텍스트 정보를 제공하는 다음 내용이 들어 있는 `widgetContext` 요소가 포함됩니다.

```
{
    "widgetContext": {
        "dashboardName": "Name-of-current-dashboard",
        "widgetId": "widget-16",
        "accountId": "012345678901",
        "locale": "en",
        "timezone": {
            "label": "UTC",
            "offsetISO": "+00:00",
            "offsetInMinutes": 0
        },
        "period": 300,
        "isAutoPeriod": true,
        "timeRange": {
            "mode": "relative",
            "start": 1627236199729,
```

```
            "end": 1627322599729,
            "relativeStart": 86400012,
            "zoom": {
                "start": 1627276030434,
                "end": 1627282956521
             }
        },
        "theme": "light",
        "linkCharts": true,
        "title": "Tweets for Amazon website problem",
        "forms": {
            "all": {}
        },
        "params": {
            "original": "param-to-widget"
        },
        "width": 588,
        "height": 369
    }
}
```

기본 CSS 스타일 지정

사용자 지정 위젯은 다음과 같은 기본 CSS 스타일 지정 요소를 제공합니다.

- CSS 클래스 [btn]을 사용하여 버튼을 추가할 수 있습니다. 다음 예와 같이 앵커(<a>)를 버튼으로 바꿉니다.

  ```
  <a class="btn" href=https://amazon.com”>Open Amazon</a>
  ```

- CSS 클래스 [btn btn-primary]를 사용하여 기본 버튼을 추가할 수 있습니다.
- [table], [select], [headers(h1, h2, h3)], [preformatted text(pre)], [input], [text area]와 같은 요소는 기본적으로 스타일이 지정됩니다.

describe 파라미터 사용

빈 문자열만 반환하는 경우에도 함수에서 [describe] 파라미터를 지원하는 것이 좋습니다. 이 파라미터를 지원하지 않고 사용자 지정 위젯에서 호출하면 위젯 콘텐츠가 문서인 것처럼 표시됩니다.

[describe] 파라미터를 포함하는 경우 Lambda 함수는 문서를 Markdown 형식으로 반환하고 그 밖에 아무것도 하지 않습니다.

콘솔에서 사용자 지정 위젯을 생성할 때 Lambda 함수를 선택하면 [문서 가져오기(Get documentation)] 버튼이 표시됩니다. 이 버튼을 선택하면 함수가 [describe] 파라미터와 함께 호출되고 함수의 문서가 반환됩니다. 문서의 형식이 올바른 Markdown인 경우 CloudWatch는 YAML에서 세 개의 단일 백틱 문자(```)로 둘러싸인 문서의 첫 번째 항목을 구문 분석합니다. 그런 다음, 파라미터의 문서를 자동으로 채웁니다. 다음은 이러한 형식이 잘 지정된 문서의 예입니다.

```
``` yaml
echo: <h1>Hello world</h1>
```
```

보안 및 JavaScript

보안상의 이유로 반환된 HTML에는 JavaScript가 허용되지 않습니다. JavaScript를 제거하면 Lambda 함수 작성자가 대시보드에서 위젯을 보는 사용자보다 더 높은 권한으로 실행할 수 있는 코드를 삽입하는 권한 에스컬레이션 문제를 방지할 수 있습니다.

반환된 HTML에 JavaScript 코드 또는 기타 알려진 보안 취약성이 포함되어 있다면 해당 코드 또는 취약성은 대시보드에서 렌더링되기 전에 HTML에서 제거됩니다. 예를 들어 <iframe> 및 <use> 태그는 허용되지 않으며 제거됩니다.

사용자 정의 위젯은 기본적으로 대시보드에서 실행되지 않습니다. 대신 호출하는 Lambda 함수를 신뢰하는 경우 사용자 정의 위젯이 실행되도록 명시적으로 허용해야 합니다. 개별 위젯과 전체 대시보드 모두에 대해 한 번 허용하거나 항상 허용하도록 선택할 수 있습니다. 개별 위젯 및 전체 대시보드에 대한 권한을 거부할 수도 있습니다.

사용자 지정 위젯의 상호 작용

JavaScript가 허용되지는 않지만 반환된 HTML과의 상호 작용을 허용하는 다른 방법이 있습니다.

- 팝업에 정보를 표시하고 클릭 시 확인을 요청하며 해당 요소가 선택되었을 때 Lambda 함수를 호출할 수 있는 <cwdb-action> 태그의 특수 구성으로 반환된 HTML의 요소에 태그를 지정할 수 있습니다. 예를 들어 Lambda 함수를 사용하여 AWS API를 호출하는 버튼을 정의할 수 있습니다. 기존 Lambda 위젯의 콘텐츠를 대체하거나 모달 내부에 표시되도록 반환된 HTML을 설정할 수 있습니다.
- 반환된 HTML에 새 콘솔을 열거나 다른 고객 페이지를 열거나 다른 대시보드를 로드하는 링크를 포함할 수 있습니다.
- HTML에는 사용자가 해당 요소 위로 마우스를 가져가면 추가 정보를 제공하는 요소에 대한 `title` 속성이 포함될 수 있습니다.
- 요소에 애니메이션 또는 기타 CSS 효과를 호출할 수 있는 `:hover`와 같은 CSS 선택기를 포함할 수 있습니다. 또한 페이지에서 요소를 표시하거나 숨길 수도 있습니다.

<cwdb-action> 정의 및 사용법

<cwdb-action> 요소는 바로 이전 요소에 대한 동작을 정의합니다. <cwdb-action>의 내용은 표시할 HTML이거나 Lambda 함수에 전달할 파라미터의 JSON 블록입니다.

다음은 <cwdb-action> 요소의 예입니다.

```
<cwdb-action
     action="call|html"
     confirmation="message"
     display="popup|widget"
     endpoint="<lambda ARN>"
     event="click|dblclick|mouseenter">

     html | params in JSON
</cwdb-action>
```

- action - 유효한 값은 Lambda 함수를 호출하는 call이거나 <cwdb-action> 내에 포함된 HTML을 표시하는 html입니다. 기본값은 html입니다.
- confirmation - 작업을 수행하기 전에 확인해야 하는 확인 메시지를 표시하며, 이를 통해 고객이 취소할 수 있습니다.
- display - 유효한 값은 popup 및 widget으로, 위젯 자체의 콘텐츠를 대체합니다. 기본값은 widget입니다.
- endpoint - 호출할 Lambda 함수의 Amazon 리소스 이름(ARN)입니다. action이 call인 경우 이 값은 필수입니다.
- event - 작업을 호출하는 이전 요소의 이벤트를 정의합니다. 유효한 값은 click, dblclick, mouseenter입니다. mouseenter 이벤트는 html 작업과 함께 사용해야만 합니다. 기본값은 click입니다.

예제

다음은 <cwdb-action> 태그를 사용하여 Lambda 함수 호출을 사용함으로써 Amazon EC2 인스턴스를 재부팅하는 버튼을 생성하는 방법의 예입니다. 팝업에 호출 성공 또는 실패를 표시합니다.

```
<a class="btn">Reboot Instance</a>
<cwdb-action action="call" endpoint="arn:aws:lambda:us-
east-1:123456:function:rebootInstance" display="popup">
       { "instanceId": "i-342389adbfef" }
```

```
</cwdb-action>
```

다음 예는 팝업에 추가 정보를 표시합니다.

```
<a>Click me for more info in popup</a>
<cwdb-action display="popup">
   <h1>Big title</h1>
   More info about <b>something important</b>.
</cwdb-action>
```

이 예는 위젯의 콘텐츠를 Lambda 함수에 대한 호출로 바꾸는 [다음(Next)] 버튼입니다.

```
<a class="btn btn-primary">Next</a>
<cwdb-action action="call" endpoint="arn:aws:lambda:us-
east-1:123456:function:nextPage">
   { "pageNum": 2 }
</cwdb-action>
```

사용자 지정 위젯 생성

사용자 지정 위젯을 생성하려면 AWS에서 제공하는 샘플 중 하나를 사용하거나 직접 생성하면 됩니다. AWS 샘플은 JavaScript와 Python 샘플을 모두 포함하며 AWS CloudFormation 스택에 의해 생성됩니다. 샘플 목록은 샘플 사용자 지정 위젯 단원을 참조하세요.

CloudWatch 대시보드에서 사용자 지정 위젯을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. + 기호를 선택합니다.
4. [사용자 지정 위젯(Custom widget)]을 선택합니다.
5. 다음 방법 중 한 가지를 선택하세요.
   - AWS에서 제공하는 샘플 사용자 지정 위젯을 사용하려면 다음을 수행합니다.
     a. 드롭다운 상자에서 샘플을 선택합니다.

        AWS CloudFormation 콘솔이 새 브라우저에서 시작됩니다. AWS CloudFormation 콘솔에서 다음을 수행합니다.
     b. (선택 사항) AWS CloudFormation 스택 이름을 사용자 지정합니다.
     c. 샘플에 사용되는 파라미터를 선택합니다.

d. I acknowledge that AWS CloudFormation might create IAM resources(에서 IAM 리소스를 생성할 수 있음을 인정합니다)를 선택하고 Create stack(스택 생성)을 선택합니다.

- AWS에서 제공하는 자체 사용자 지정 위젯을 생성하려면 다음을 수행합니다.

a. 다음을 선택합니다.

b. 목록에서 Lambda 함수 선택 또는 Amazon 리소스 이름(ARN) 입력을 선택합니다. 목록에서 선택하는 경우 함수가 있는 리전 및 사용할 버전도 지정합니다.

c. [파라미터(Parameters)]에서 함수에 사용되는 파라미터를 선택합니다.

d. 위젯의 제목을 입력합니다.

e. [업데이트 시점(Update on)]에서 위젯을 업데이트해야 하는 시점(Lambda 함수를 다시 호출해야 하는 시점)을 구성합니다. 대시보드가 자동으로 새로 고쳐질 때 업데이트하려면 [새로 고침(Refresh)], 위젯 크기가 조정될 때마다 업데이트하려면 [크기 조정(Resize)] 또는 그래프를 확대했을 때를 포함하여 대시보드의 시간 범위가 조정될 때마다 업데이트하려면 [시간 범위(Time Range)]를 선택하되 하나 이상을 선택할 수 있습니다.

f. 미리 보기가 마음에 들면 [위젯 생성(Create widget)]을 선택합니다.

샘플 사용자 지정 위젯

AWS는 JavaScript 및 Python 샘플 사용자 지정 위젯을 모두 제공합니다. 아래에 있는 목록의 각 위젯에 대한 링크를 사용하여 이러한 샘플 위젯을 생성할 수 있습니다. 또는 CloudWatch 콘솔을 사용하여 위젯을 생성하고 사용자 지정할 수 있습니다. 아래에 있는 목록의 링크는 AWS CloudFormation 콘솔을 열고 AWS CloudFormation 빠른 생성 링크를 사용하여 사용자 지정 위젯을 생성합니다.

또한 GitHub에서 사용자 지정 위젯 샘플에 액세스할 수도 있습니다.

이 목록 다음에는 각 언어에 대한 Echo 위젯의 전체 예가 나와 있습니다.

JavaScript

JavaScript 샘플 사용자 지정 위젯

- Echo - 새 위젯을 작성할 필요 없이 HTML이 사용자 지정 위젯에 어떻게 표시되는지 테스트하는 데 사용할 수 있는 기본 Echo 샘플입니다.
- Hello world – 매우 기본적인 시작 위젯입니다.
- 사용자 위젯 디버거 – Lambda 런타임 환경에 관한 유용한 정보를 표시하는 디버거 위젯입니다.
- CloudWatch Logs Insights 쿼리 – CloudWatch Logs Insights 쿼리를 실행하고 편집합니다.
- Amazon Athena 쿼리 실행 – Athena 쿼리를 실행하고 편집합니다.

- AWS API 호출 – 읽기 전용 AWS API를 호출하고 결과를 JSON 형식으로 표시합니다.
- 빠른 CloudWatch 비트맵 그래프 – 빠른 표시를 위해 서버 측에서 사용하는 CloudWatch 그래프를 렌더링합니다.
- CloudWatch 대시보드의 텍스트 위젯 – 지정된 CloudWatch 대시보드의 첫 번째 텍스트 위젯을 표시합니다.
- 테이블로 CloudWatch 지표 데이터 표시 – 원시 CloudWatch 지표 데이터를 테이블로 표시합니다.
- Amazon EC2 테이블 – CPU 사용률을 기준으로 상위 EC2 인스턴스를 표시합니다. 이 위젯에는 기본적으로 사용 중지된 재부팅 버튼도 포함되어 있습니다.
- AWS CodeDeploy 배포 – CodeDeploy 배포를 표시합니다.
- AWS Cost Explorer 보고서 – 선택한 시간 범위에 대한 각 AWS 서비스의 비용에 관한 보고서를 표시합니다.
- 외부 URL의 내용 표시 – 외부에서 액세스할 수 있는 URL의 내용을 표시합니다.
- Amazon S3 객체 표시 – 계정의 Amazon S3 버킷에 있는 객체를 표시합니다.
- 단순 SVG 파이 차트 – 그래픽 SVG 기반 위젯의 예입니다.

Python

Python 샘플 사용자 지정 위젯

- Echo - 새 위젯을 작성할 필요 없이 HTML이 사용자 지정 위젯에 어떻게 표시되는지 테스트하는 데 사용할 수 있는 기본 Echo 샘플입니다.
- Hello world – 매우 기본적인 시작 위젯입니다.
- 사용자 위젯 디버거 – Lambda 런타임 환경에 관한 유용한 정보를 표시하는 디버거 위젯입니다.
- AWS API 호출 – 읽기 전용 AWS API를 호출하고 결과를 JSON 형식으로 표시합니다.
- 빠른 CloudWatch 비트맵 그래프 – 빠른 표시를 위해 서버 측에서 사용하는 CloudWatch 그래프를 렌더링합니다.
- 이메일로 대시보드 스냅샷 전송 – 현재 대시보드의 스냅샷을 생성하여 이메일 수신자에게 전송합니다.
- Amazon S3에 대시보드 스냅샷 전송 – 현재 대시보드의 스냅샷을 생성하여 Amazon S3에 저장합니다.
- CloudWatch 대시보드의 텍스트 위젯 – 지정된 CloudWatch 대시보드의 첫 번째 텍스트 위젯을 표시합니다.
- 외부 URL의 내용 표시 – 외부에서 액세스할 수 있는 URL의 내용을 표시합니다.

- RSS 리더 - RSS 피드를 표시합니다.
- Amazon S3 객체 표시 – 계정의 Amazon S3 버킷에 있는 객체를 표시합니다.
- 단순 SVG 파이 차트 – 그래픽 SVG 기반 위젯의 예입니다.

JavaScript Echo 위젯

다음은 JavaScript Echo 샘플 위젯입니다.

```
const DOCS = `
## Echo
A basic echo script. Anything passed in the \`\`\`echo\`\`\` parameter is returned as
 the content of the custom widget.
### Widget parameters
Param | Description
---|---
**echo** | The content to echo back

### Example parameters
\`\`\` yaml
echo: <h1>Hello world</h1>
\`\`\`
`;

exports.handler = async (event) => {
    if (event.describe) {
        return DOCS;
    }

    let widgetContext = JSON.stringify(event.widgetContext, null, 4);
    widgetContext = widgetContext.replace(/</g, '&lt;');
    widgetContext = widgetContext.replace(/>/g, '&gt;');

    return `${event.echo || ''}<pre>${widgetContext}</pre>`;
};
```

Python Echo 위젯

다음은 Python Echo 샘플 위젯입니다.

```
import json
```

```
DOCS = """
## Echo
A basic echo script. Anything passed in the ```echo``` parameter is returned as the
 content of the custom widget.
### Widget parameters
Param | Description
---|---
**echo** | The content to echo back

### Example parameters
``` yaml
echo: <h1>Hello world</h1>
```"""

def lambda_handler(event, context):
    if 'describe' in event:
        return DOCS

    echo = event.get('echo', '')
    widgetContext = event.get('widgetContext')
    widgetContext = json.dumps(widgetContext, indent=4)
    widgetContext = widgetContext.replace('<', '<')
    widgetContext = widgetContext.replace('>', '>')

    return f'{echo}<pre>{widgetContext}</pre>'
```

Java Echo 위젯

다음은 Java Echo 샘플 위젯입니다.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class Handler implements RequestHandler<Event, String>{

  static String DOCS = ""
    + "## Echo\n"
    + "A basic echo script. Anything passed in the ```echo``` parameter is returned as
 the content of the custom widget.\n"
```

```
    + "### Widget parameters\n"
    + "Param | Description\n"
    + "---|---\n"
    + "**echo** | The content to echo back\n\n"
    + "### Example parameters\n"
    + "```yaml\n"
    + "echo: <h1>Hello world</h1>\n"
    + "```\n";

  Gson gson = new GsonBuilder().setPrettyPrinting().create();

  @Override
  public String handleRequest(Event event, Context context) {

    if (event.describe) {
      return DOCS;
    }

    return (event.echo != null ? event.echo : "") + "<pre>" +
 gson.toJson(event.widgetContext) + "</pre>";
  }
}

class Event {

    public boolean describe;
    public String echo;
    public Object widgetContext;

    public Event() {}

    public Event(String echo, boolean describe, Object widgetContext) {
        this.describe = describe;
        this.echo = echo;
        this.widgetContext = widgetContext;
    }
}
```

CloudWatch 대시보드에서 텍스트 위젯 추가 또는 제거

텍스트 위젯에는 마크다운 형식으로 된 텍스트 블록이 포함되어 있습니다. CloudWatch 대시보드에서 텍스트 위젯을 추가, 편집 또는 제거할 수 있습니다.

대시보드에 텍스트 위젯을 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. + 기호를 선택합니다.
4. 텍스트를 선택합니다.
5. 마크다운에서 텍스트를 추가하고 마크다운을 사용하여 서식을 지정하고 위젯 생성을 선택합니다.
6. 텍스트 위젯을 투명하게 만들려면 투명 배경을 선택합니다.
7. 대시보드 저장을 선택합니다.

대시보드에서 텍스트 위젯을 편집하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 텍스트 블록의 오른쪽 상단 모서리에 마우스를 가져가서 위젯 작업(Widget actions)을 선택합니다. 그런 다음 편집(Edit)을 선택합니다.
4. 필요에 따라 텍스트를 업데이트하고 위젯 업데이트를 선택합니다.
5. 대시보드 저장을 선택합니다.

대시보드에서 텍스트 위젯을 제거하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 텍스트 블록의 오른쪽 상단 모서리에 마우스를 가져가서 위젯 작업(Widget actions)을 선택합니다. 그런 다음 삭제를 선택합니다.
4. 대시보드 저장을 선택합니다.

CloudWatch 대시보드에서 경보 위젯 추가 또는 제거

대시보드에 경보 위젯을 추가하려면 다음 옵션 중 하나를 선택합니다.

- 경보 지표 그래프와 경보 상태를 모두 표시하는 위젯에 단일 경보를 추가합니다.

- 그리드에 여러 경보의 상태를 표시하는 경보 상태 위젯을 추가합니다. 이 위젯에는 경보 이름과 현재 상태만 표시되며, 그래프는 표시되지 않습니다. 경보 상태 위젯 하나에 최대 100개의 경보가 포함될 수 있습니다.

대시보드에 그래프를 포함하여 단일 경보를 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보를 선택하고 추가할 경보를 선택한 다음 대시보드에 추가를 선택합니다.
3. 대시보드를 선택하고, 위젯 유형(행, 누적 면적 또는 번호)을 선택한 다음 대시보드에 추가를 선택합니다.
4. 대시보드에 경보를 표시하려면 탐색 창에서 대시보드를 선택하고 해당 대시보드를 선택합니다.
5. (선택 사항) 경보 그래프를 일시적으로 확대하려면 해당 그래프를 선택합니다.
6. (선택 사항) 위젯 유형을 변경하려면 그래프 제목 위로 마우스를 가져가 위젯 작업을 선택한 다음 위젯 유형을 선택합니다.

대시보드에 경보 상태 위젯을 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. + 기호를 선택합니다.
4. 경보 상태(Alarm status)를 선택합니다.
5. 위젯에 추가하려는 경보 옆의 확인란을 선택한 다음, 위젯 생성(Create widget)을 선택합니다.
6. 대시보드에 추가(Add to dashboard)를 선택합니다.

대시보드에서 경보 위젯을 제거하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 위젯 위로 마우스를 가져가 위젯 작업을 선택한 다음 삭제를 선택합니다.
4. 대시보드 저장을 선택합니다. 변경 사항을 저장하기 전에 대시보드에서 다른 곳으로 이동하려고 시도하면 변경 사항을 저장하거나 삭제하라는 메시지가 나타납니다.

CloudWatch 대시보드에서 데이터 테이블 위젯 추가 또는 제거

데이터 테이블 위젯을 사용하면 지표의 원시 데이터포인트와 해당 원시 데이터에 대한 간략한 요약을 볼 수 있습니다. 데이터 테이블 위젯은 실제 데이터를 추상화하기 위한 차트가 아니므로 표시되는 데이터 포인트를 더 쉽게 이해할 수 있습니다. 이 섹션의 절차에서는 CloudWatch 대시보드에서 데이터 테이블 위젯을 추가 및 제거하는 방법에 대해 설명합니다.

Label	Min	Max	Sum	Average	11/20 06:00	11/20 00:00	11/19 18:00	11/19 12:00	11/ 06:(
TestMetric295	991	1,000	12k	998	996	1,000	997	999	
TestMetric296	995	1,000	12k	998	995	1,000	1,000	998	
TestMetric297	991	1,000	12k	998	998	1,000	999	997	
TestMetric298	994	1,000	12k	997	996	999	995	995	
TestMetric3	993	1,000	12k	998	1,000	999	999	1,000	
TestMetric299	995	999	12k	998	999	995	999	998	
TestMetric30	994	999	12k	998	999	998	999	999	
StackMetric2	99	99.9	1.2k	99.6	99.2	99.7	99.5	99.8	
StackMetric20	99	100	1.19k	99.5	100	99.1	99.4	99.4	
StackMetric21	97.5	100	1.19k	99.4	99.6	99.7	97.6	99.8	

테이블 속성

데이터 테이블에는 옵션이나 소스를 변경할 필요가 없는 기본 속성 세트가 있습니다. 이러한 속성에는 스티키 레이블 열, 활성화된 모든 요약 열, 반올림된 데이터 포인트, 변환된 단위 등이 포함됩니다.

각 데이터 테이블 위젯은 다음 속성을 가질 수 있습니다. 각 속성에 대한 정보에는 대시보드의 JSON 소스에서 속성을 구성하는 방법이 포함됩니다. 대시보드 JSON에 대한 자세한 내용은 Dashboard Body Structure and Syntax를 참조하세요.

요약

요약 열은 데이터 테이블 위젯에 도입된 새로운 속성입니다. 이러한 열은 현재 테이블 요약의 특정 하위 세트입니다. 예를 들어, 합계 요약은 해당 행에 표시된 모든 데이터 포인트의 합계입니다. 요약 열은 CloudWatch 통계와 다릅니다. 소스에서 다음과 같이 표시됩니다.

```
"table": {
       "summaryColumns": [
           "MIN",
           "MAX",
           "SUM",
           "AVG"
       ]
```

```
    },
```

임곗값

이를 사용하여 테이블에 임곗값을 적용합니다. 데이터 포인트가 임곗값 내에 속하면 해당 셀이 임곗값 색상으로 강조 표시됩니다. 소스에서 다음과 같이 표시됩니다.

```
"annotations": {
    "horizontal": [
        {
            "label": string,
            "value": int,
            "fill": "above" | "below"
        }
    ]
}
```

레이블 열의 단위

지표와 관련된 단위를 표시하려면 이 옵션을 활성화하여 레이블 옆의 레이블 열에 단위를 표시합니다. 소스에서 다음과 같이 표시됩니다.

```
"yAxis": {
    "left": {
        "showUnits": true | false
    }
}
```

행 및 열 반전

이렇게 하면 테이블이 변환되어 데이터 포인트가 열에서 행으로 바뀌고 지표가 열이 됩니다. 소스에서 다음과 같이 표시됩니다.

```
 "table": {
    "layout": "vertical" | "horizontal"
}
```

스티키 요약 열

이렇게 하면 요약 열이 고정되어 스크롤하는 동안 요약 열이 계속 표시됩니다. 레이블은 이미 고정되어 있습니다. 소스에서 다음과 같이 표시됩니다.

```
"table": {
    "stickySummary": true | false
}
```

요약 열만 표시

이렇게 하면 레이블 및 요약 열만 표시되도록 데이터 포인트의 열이 표시되지 않습니다. 소스에서 다음과 같이 표시됩니다.

```
 "table": {
    "showTimeSeriesData": false | true
}
```

라이브 데이터

아직 완전히 집계되지 않았더라도 가장 최근의 데이터 포인트를 표시합니다. 소스에서 다음과 같이 표시됩니다.

```
"liveData": true | false
```

숫자 위젯 형식

반올림하여 변환하기 전에 셀에 들어갈 수 있는 자릿수만큼 표시합니다. 소스에서 다음과 같이 표시됩니다.

```
"singleValueFullPrecision": true | false
```

대시보드에 데이터 테이블 위젯 추가

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드를 선택한 다음, 대시보드를 선택합니다.
3. + 버튼을 선택하고 데이터 테이블을 선택한 후 다음을 선택합니다.
4. 찾아보기 탭에서 테이블 위젯에 표시하려는 지표를 검색하거나 찾아봅니다. 그런 다음, 지표를 선택합니다.
5. (선택 사항) 테이블의 레이아웃을 변경하려면 옵션 탭을 선택하고 행 및 열 반전을 선택합니다.

   또한 옵션 탭을 사용하여 테이블에 나타나는 열을 변경하고 레이블 열에 사용 중인 단위를 표시할 수 있습니다.

> **Tip**
>
> 임곗값을 더 정확하게 표시하려면 반올림 전에 들어갈 수 있는 자릿수 표시를 선택합니다.

6. (선택 사항) 데이터 테이블 위젯의 시간 범위를 변경하려면 위젯의 상단 영역에서 미리 정의된 시간 범위 중 하나를 선택합니다. 시간 범위는 1시간에서 1주일입니다. 고유한 시간 범위를 설정하려면 사용자 지정을 선택합니다.
7. (선택 사항) 데이터 테이블 위젯의 시간 범위를 변경하려면 위젯의 상단 영역에서 미리 정의된 시간 범위 중 하나를 선택합니다. 시간 범위는 1시간에서 1주일입니다. 고유한 시간 범위를 설정하려면 사용자 지정을 선택합니다.
8. (선택 사항) 나중에 대시보드의 나머지 시간 범위가 변경되더라도 이 위젯에는 선택한 시간 범위가 계속 사용되게 하려면 시간 범위 유지를 선택합니다.
9. 위젯 생성을 선택한 다음 대시보드 저장을 선택합니다.

대시보드에서 테이블 위젯 제거

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 제거하려는 위젯의 오른쪽 상단에서 위젯 작업, 삭제를 선택합니다.
4. 대시보드 저장을 선택합니다.

CloudWatch 대시보드에서 그래프 링크 및 링크 해제

하나의 그래프를 확대 또는 축소하면 다른 그래프들도 동시에 확대 또는 축소가 되도록 대시보드의 그래프들을 하나로 링크할 수 있습니다. 그래프 링크를 해제하면 하나의 그래프로만 확대/축소를 제한할 수 있습니다.

대시보드에서 그래프들을 링크하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 작업을 선택한 후 그래프 연결을 선택합니다.

대시보드에서 그래프들의 링크를 해제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 작업과 그래프 연결의 선택을 차례로 취소합니다.

CloudWatch 대시보드 공유

AWS 계정에 직접 액세스할 수 없는 사용자와 CloudWatch 대시보드를 공유할 수 있습니다. 이렇게 하면 팀 간에 그리고 이해 관계자 및 조직 외부의 사람들과 대시보드를 공유할 수 있습니다. 대시보드를 팀 영역의 큰 화면에 표시하거나 Wiki 및 다른 웹 페이지에 포함할 수도 있습니다.

> **Warning**
>
> 대시보드를 공유받는 모든 사람에게는 계정의 대시보드를 공유받는 사용자에게 부여되는 권한 단원에 나열된 권한이 부여됩니다. 대시보드를 공개적으로 공유하면 대시보드에 대한 링크가 있는 모든 사람이 이러한 권한을 보유합니다.
>
> `cloudwatch:GetMetricData` 및 `ec2:DescribeTags` 권한의 경우 특정 지표 또는 EC2 인스턴스로 범위를 좁힐 수 없으므로 대시보드에 대한 액세스 권한이 있는 사용자는 모든 CloudWatch 지표와 계정에 있는 모든 EC2 인스턴스의 이름 및 태그를 쿼리할 수 있습니다.

대시보드를 공유할 때 다음 세 가지 방법으로 대시보드를 볼 수 있는 사용자를 지정할 수 있습니다.

- 단일 대시보드를 공유하고 대시보드를 볼 수 있는 사용자의 이메일 주소를 최대 5개 지정합니다. 이러한 각 사용자는 대시보드를 보기 위해 입력해야 하는 고유한 암호를 생성합니다.
- 링크가 있는 사람은 누구나 대시보드를 볼 수 있도록 단일 대시보드를 공개적으로 공유합니다.
- 계정의 모든 CloudWatch 대시보드를 공유하고 대시보드 액세스를 위한 서드 파티 통합 인증(SSO) 공급자를 지정합니다. 이 SSO 공급자 목록의 멤버인 모든 사용자는 계정의 모든 대시보드에 액세스할 수 있습니다. 이를 사용하려면 SSO 공급자를 Amazon Cognito와 통합합니다. SSO 공급자는 Security Assertion Markup Language(SAML)를 지원해야 합니다. Amazon Cognito에 대한 자세한 내용은 Amazon Cognito란? 단원을 참조하세요.

대시보드 공유에는우 요금이 발생하지 않지만 공유 대시보드 내의 위젯에는 표준 CloudWatch 요금이 부과됩니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

대시보드를 공유하면 Amazon Cognito 리소스가 미국 동부(버지니아 북부) 리전에 생성됩니다.

> **Important**
>
> 대시보드 공유 프로세스에서 생성된 리소스 이름 및 식별자를 수정하지 마세요. 여기에는 Amazon Cognito 및 IAM 리소스가 포함됩니다. 이러한 리소스를 수정하면 공유 대시보드의 예상치 못한 잘못된 기능이 발생할 수 있습니다.

> **Note**
>
> 경보 주석이 포함된 지표 위젯이 있는 대시보드를 공유하는 경우 대시보드가 공유된 사용자들에게는 해당 위젯이 표시되지 않습니다. 대신 위젯을 사용할 수 없다는 텍스트가 포함된 빈 위젯이 표시됩니다. 대시보드를 본인이 직접 확인하는 경우 경보 주석이 포함된 지표 위젯이 계속 표시됩니다.

대시보드 공유에 필요한 권한

다음 방법 중 하나를 사용하여 대시보드를 공유하고 이미 공유된 대시보드를 확인하려면 사용자로 로그인하거나 특정 권한이 있는 IAM 역할로 로그인해야 합니다.

대시보드를 공유할 수 있으려면 사용자 또는 IAM 역할에 다음 정책 설명에 포함된 권한이 포함되어 있어야 합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy",
        "iam:PassRole"
    ],
    "Resource": [
        "arn:aws:iam::*:role/service-role/CWDBSharing*",
        "arn:aws:iam::*:policy/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cognito-idp:*",
```

```
        "cognito-identity:*",
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cloudwatch:GetDashboard",
    ],
    "Resource": [
        "*"
        // or the ARNs of dashboards that you want to share
    ]
}
```

어떤 대시보드가 공유되는지 볼 수 있지만 대시보드를 공유할 수 없도록 하려면 사용자 또는 IAM 역할에 다음과 유사한 정책 설명을 포함하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "cognito-idp:*",
        "cognito-identity:*"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cloudwatch:ListDashboards",
    ],
    "Resource": [
        "*"
    ]
}
```

대시보드를 공유받는 사용자에게 부여되는 권한

대시보드를 공유할 때 CloudWatch는 대시보드를 공유받는 사용자에게 다음 권한을 부여하는 IAM 역할을 계정에 생성합니다.

- `cloudwatch:GetInsightRuleReport`
- `cloudwatch:GetMetricData`
- `cloudwatch:DescribeAlarms`
- `ec2:DescribeTags`

> **Warning**
>
> 대시보드를 공유받는 모든 사람에게는 계정의 이러한 권한이 부여됩니다. 대시보드를 공개적으로 공유하면 대시보드에 대한 링크가 있는 모든 사람이 이러한 권한을 보유합니다. `cloudwatch:GetMetricData` 및 `ec2:DescribeTags` 권한의 경우 특정 지표 또는 EC2 인스턴스로 범위를 좁힐 수 없으므로 대시보드에 대한 액세스 권한이 있는 사용자는 모든 CloudWatch 지표와 계정에 있는 모든 EC2 인스턴스의 이름 및 태그를 쿼리할 수 있습니다.

대시보드를 공유할 때 기본적으로 CloudWatch가 생성하는 권한은 대시보드가 공유될 때 대시보드에 있는 경보 및 Contributor Insights 규칙에 대한 액세스만 제한합니다. 대시보드에 새 경보 또는 Contributor Insights 규칙을 추가하고 대시보드를 공유받은 사용자도 해당 경보 또는 규칙을 볼 수 있도록 하려면 이러한 리소스를 허용하도록 정책을 업데이트해야 합니다.

특정 사용자와 단일 대시보드 공유

이 단원의 단계에 따라 선택한 최대 5개의 이메일 주소와 대시보드를 공유할 수 있습니다.

> **Note**
>
> 기본적으로 대시보드의 CloudWatch Logs 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 자세한 내용은 공유받는 사용자가 로그 테이블 위젯을 볼 수 있도록 허용 단원을 참조하세요.
>
> 기본적으로 대시보드의 복합 경보 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 자세한 내용은 공유받는 사용자가 복합 경보를 볼 수 있도록 허용 단원을 참조하세요.

특정 사용자와 대시보드를 공유하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 대시보드 이름을 선택합니다.
4. [작업(Actions)], [대시보드 공유(Share dashboard)]를 선택합니다.
5. [대시보드를 공유하며 사용자 이름 및 암호가 필요함(Share your dashboard and require a username and password)] 옆에 있는 [공유 시작(Start sharing)]을 선택합니다.
6. [이메일 주소 추가(Add email addresses)]에서 대시보드를 공유하려는 이메일 주소를 입력합니다. 최대 5개의 이메일 주소를 포함할 수 있습니다.
7. 이메일 주소를 모두 입력했으면 동의를 읽고 확인 상자를 선택합니다. 그런 다음, [정책 미리 보기(Preview policy)]를 선택합니다.
8. 공유할 리소스가 원하는 리소스인지 확인하고 [확인 및 공유 가능한 링크 생성(Confirm and generate shareable link)]을 선택합니다.
9. 다음 페이지에서 [클립보드에 링크 복사(Copy link to clipboard)]를 선택합니다. 그런 다음, 이 링크를 이메일에 붙여넣고 초대한 사용자에게 전송할 수 있습니다. 해당 사용자는 대시보드에 연결하는 데 사용할 사용자 이름 및 임시 암호가 포함된 별도의 이메일을 자동으로 받습니다.

공개적으로 단일 대시보드 공유

이 단원의 단계에 따라 대시보드를 공개적으로 공유할 수 있습니다. 이 방법은 대시보드를 팀 공간의 큰 화면에 표시하거나 Wiki 페이지에 포함하는 데 유용할 수 있습니다.

> **Important**
>
> 대시보드를 공개적으로 공유하면 링크가 있는 사람은 누구나 인증 없이 대시보드에 액세스할 수 있습니다. 따라서 민감한 정보가 포함되지 않은 대시보드에 대해서만 이 작업을 수행해야 합니다.

> **Note**
>
> 기본적으로 대시보드의 CloudWatch Logs 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 자세한 내용은 공유받는 사용자가 로그 테이블 위젯을 볼 수 있도록 허용 단원을 참조하세요.

기본적으로 대시보드의 복합 경보 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 자세한 내용은 공유받는 사용자가 복합 경보를 볼 수 있도록 허용 단원을 참조하세요.

대시보드를 공개적으로 공유하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 대시보드 이름을 선택합니다.
4. [작업(Actions)], [대시보드 공유(Share dashboard)]를 선택합니다.
5. [공개적으로 대시보드 공유(Share your dashboard publicly)] 옆에 있는 [공유 시작(Start sharing)]을 선택합니다.
6. 텍스트 상자에 **`Confirm`**를 입력합니다.
7. 동의를 읽고 확인 상자를 선택합니다. 그런 다음, [정책 미리 보기(Preview policy)]를 선택합니다.
8. 공유할 리소스가 원하는 리소스인지 확인하고 [확인 및 공유 가능한 링크 생성(Confirm and generate shareable link)]을 선택합니다.
9. 다음 페이지에서 [클립보드에 링크 복사(Copy link to clipboard)]를 선택합니다. 그런 다음, 이 링크를 공유할 수 있습니다. 링크를 공유받는 사람은 누구나 자격 증명을 제공하지 않고도 대시보드에 액세스할 수 있습니다.

SSO를 사용하여 계정의 모든 CloudWatch 대시보드 공유

이 단원의 단계에 따라 통합 인증(SSO)을 사용하여 계정의 모든 대시보드를 사용자와 공유할 수 있습니다.

> **Note**
>
> 기본적으로 대시보드의 CloudWatch Logs 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 자세한 내용은 공유받는 사용자가 로그 테이블 위젯을 볼 수 있도록 허용 단원을 참조하세요.
>
> 기본적으로 대시보드의 복합 경보 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 자세한 내용은 공유받는 사용자가 복합 경보를 볼 수 있도록 허용 단원을 참조하세요.

SSO 공급자 목록에 있는 사용자와 CloudWatch 대시보드를 공유하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 대시보드 이름을 선택합니다.
4. [작업(Actions)], [대시보드 공유(Share dashboard)]를 선택합니다.
5. [CloudWatch 설정으로 이동(Go to CloudWatch Settings)]을 선택합니다.
6. 원하는 SSO 공급자가 [사용 가능한 SSO 공급자(Available SSO providers)] 목록에 없다면 [SSO 공급자 관리(Manage SSO providers)]를 선택하고 CloudWatch 대시보드 공유를 위한 SSO 설정 단원의 지침을 따릅니다.

   그런 다음, CloudWatch 콘솔로 돌아와서 브라우저를 새로 고칩니다. 이제 사용 설정한 SSO 공급자가 목록에 표시됩니다.
7. [사용 가능한 SSO 공급자(Available SSO providers)] 목록에서 원하는 SSO 공급자를 선택합니다.
8. Save changes(변경 사항 저장)를 선택합니다.

CloudWatch 대시보드 공유를 위한 SSO 설정

SAML을 지원하는 서드 파티 통합 인증(SSO) 공급자를 통한 대시보드 공유를 설정하려면 다음 단계를 따릅니다.

> **Important**
>
> SAML SSO 공급자가 아닌 공급자를 사용하여 대시보드를 공유하지 않는 것이 좋습니다. 그렇게 하면 의도치 않게 서드 파티가 계정의 대시보드에 액세스하도록 허용할 위험이 발생할 수 있습니다.

대시보드 공유를 사용하기 위해 SSO 공급자를 설정하려면

1. SSO 공급자를 Amazon Cognito와 통합합니다. 자세한 내용은 서드 파티 SAML 자격 증명 공급자와 Amazon Cognito 사용자 풀 통합 단원을 참조하세요.
2. SSO 공급자에서 메타데이터 XML 파일을 다운로드합니다.
3. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
4. 탐색 창에서 설정을 선택합니다.
5. [대시보드 공유(Dashboard sharing)] 섹션에서 [구성(Configure)]을 선택합니다.

6. [SSO 공급자 관리(Manage SSO providers)]를 선택합니다.

   그러면 미국 동부(버지니아 북부) 리전(us-east-1)의 Amazon Cognito 콘솔이 열립니다. 어느 [사용자 풀(User Pools)]도 표시되지 않으면 다른 리전의 Amazon Cognito 콘솔이 열렸을 수 있습니다. 그렇다면 리전을 [미국 동부(버지니아 북부) us-east-1(US East (N. Virginia) us-east-1)]으로 변경하고 다음 단계를 진행합니다.
7. [CloudWatchDashboardSharing] 풀을 선택합니다.
8. 탐색 창에서 [자격 증명 공급자(Identity providers)]를 선택합니다.
9. SAML을 선택합니다.
10. [공급자 이름(Provider name)]에 SSO 공급자 이름을 입력합니다.
11. [파일 선택(Select file)]을 선택하고 1단계에서 다운로드한 메타데이터 XML 파일을 선택합니다.
12. 공급자 생성(Create provider)을 선택합니다.

공유되는 대시보드 수 확인

CloudWatch 콘솔을 사용하여 현재 다른 사용자와 공유되고 있는 CloudWatch 대시보드 수를 확인할 수 있습니다.

공유되고 있는 대시보드 수를 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. [대시보드 공유(Dashboard sharing)] 섹션에 공유된 대시보드 수가 표시됩니다.
4. 어떤 대시보드가 공유되는지 확인하려면 [사용자 이름 및 암호(Username and password)]와 [공개 대시보드(Public dashboards)]에서 공유된 대시보드 수(***number*** dashboards shared)]를 선택합니다.

공유되는 대시보드 확인

CloudWatch 콘솔을 사용하여 현재 다른 사용자와 공유되고 있는 대시보드를 확인할 수 있습니다.

공유되고 있는 대시보드를 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.

3. 대시보드 목록에서 [공유(Share)] 열을 확인합니다. 이 열의 아이콘이 채워진 대시보드는 현재 공유되고 있는 것입니다.
4. 대시보드가 어느 사용자와 공유되고 있는지 확인하려면 대시보드 이름을 선택한 다음, [작업(Actions)], [대시보드 공유(Share dashboard)]를 선택합니다.

   ['대시보드 이름' 대시보드 공유(Share dashboard ***dashboard name***)] 페이지에는 대시보드가 어떻게 공유되고 있는지 표시됩니다. 원하는 경우 [공유 중지(Stop sharing)]를 선택하여 대시보드 공유를 중지할 수 있습니다.

하나 이상의 대시보드 공유 중지

공유된 단일 대시보드의 공유를 중지하거나 공유된 모든 대시보드의 공유를 동시에 중지할 수 있습니다.

단일 대시보드 공유를 중지하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 공유된 대시보드의 이름을 선택합니다.
4. [작업(Actions)], [대시보드 공유(Share dashboard)]를 선택합니다.
5. [공유 중지(Stop sharing)]를 선택합니다.
6. 확인 상자에서 [공유 중지(Stop sharing)]를 선택합니다.

공유된 모든 대시보드의 공유를 중지하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. [대시보드 공유(Dashboard sharing)] 섹션에서 [모든 대시보드 공유 중지(Stop sharing all dashboards)]를 선택합니다.
4. 확인 상자에서 [모든 대시보드 공유 중지(Stop sharing all dashboards)]를 선택합니다.

공유된 대시보드 권한 검토 및 권한 범위 변경

공유된 대시보드 사용자의 권한을 검토하거나 공유된 대시보드의 권한 범위를 변경하려는 경우 이 단원의 단계에 따릅니다.

공유된 대시보드 권한을 검토하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 공유된 대시보드의 이름을 선택합니다.
4. [작업(Actions)], [대시보드 공유(Share dashboard)]를 선택합니다.
5. [리소스(Resources)]에서 [IAM 역할(IAM Role)]을 선택합니다.
6. IAM 콘솔에서 표시된 정책을 선택합니다.
7. (선택 사항) 공유된 대시보드 사용자가 볼 수 있는 경보를 제한하려면 [정책 편집(Edit policy)]을 선택하고 `cloudwatch:DescribeAlarms` 권한을 현재 위치에서 공유된 대시보드 사용자에게 표시하려는 경보의 ARN만 나열하는 새로운 `Allow` 문으로 이동합니다. 다음 예를 참조하세요.

   ```
   {
      "Effect": "Allow",
       "Action": "cloudwatch:DescribeAlarms",
       "Resource": [
           "AlarmARN1",
           "AlarmARN2"
       ]
   }
   ```

   이렇게 하는 경우 다음과 같은 현재 정책 섹션에서 `cloudwatch:DescribeAlarms` 권한을 제거해야 합니다.

   ```
   {
      "Effect": "Allow",
       "Action": [
           "cloudwatch:GetInsightRuleReport",
           "cloudwatch:GetMetricData",
           "cloudwatch:DescribeAlarms",
           "ec2:DescribeTags"
       ],
       "Resource": "*"
   }
   ```

8. (선택 사항) 공유된 대시보드 사용자가 볼 수 있는 Contributor Insights 규칙의 범위를 제한하려면 [정책 편집(Edit policy)]을 선택하고 `cloudwatch:GetInsightRuleReport`를 현재 위치에서 공유된 대시보드 사용자에게 표시하려는 Contributor Insights 규칙의 ARN만 나열하는 새로운 `Allow` 문으로 이동합니다. 다음 예를 참조하세요.

```
{
   "Effect": "Allow",
    "Action": "cloudwatch:GetInsightRuleReport",
    "Resource": [
        "PublicContributorInsightsRuleARN1",
        "PublicContributorInsightsRuleARN2"
    ]
}
```

이렇게 하는 경우 다음과 같은 현재 정책 섹션에서 `cloudwatch:GetInsightRuleReport`를 제거해야 합니다.

```
{
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetInsightRuleReport",
                "cloudwatch:GetMetricData",
                "cloudwatch:DescribeAlarms",
                "ec2:DescribeTags"
            ],
            "Resource": "*"
        }
```

공유받는 사용자가 복합 경보를 볼 수 있도록 허용

대시보드를 공유할 때 기본적으로 대시보드의 복합 경보 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 복합 경보 위젯을 표시하려면 대시보드 공유 정책에 `DescribeAlarms: *` 권한을 추가해야 합니다. 해당 권한은 다음과 같습니다.

```
{
    "Effect": "Allow",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": "*"
}
```

Warning

앞의 정책 문은 계정의 모든 경보에 대한 액세스 권한을 부여합니다. `cloudwatch:DescribeAlarms`의 범위를 줄이려면 `Deny` 문을 사용해야 합니다. 정책에

Deny 문을 추가하고 잠그려는 경보의 ARN을 지정할 수 있습니다. 해당 Deny 문은 다음과 비슷합니다.

```
{
    "Effect": "Allow",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": "*"
},
{
    "Effect": "Deny",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": [
        "SensitiveAlarm1ARN",
        "SensitiveAlarm1ARN"
    ]
}
```

공유받는 사용자가 로그 테이블 위젯을 볼 수 있도록 허용

대시보드를 공유할 때 기본적으로 대시보드에 있는 CloudWatch Logs Insights 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 이는 현재 존재하는 CloudWatch Logs Insights 위젯과 공유 이후에 대시보드에 추가되는 항목에 모두 적용됩니다.

이러한 사용자가 CloudWatch Logs 위젯을 볼 수 있도록 하려면 대시보드 공유를 위한 IAM 역할에 권한을 추가해야 합니다.

대시보드를 공유받는 사용자가 CloudWatch Logs 위젯을 볼 수 있도록 허용하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 공유된 대시보드의 이름을 선택합니다.
4. [작업(Actions)], [대시보드 공유(Share dashboard)]를 선택합니다.
5. [리소스(Resources)]에서 [IAM 역할(IAM Role)]을 선택합니다.
6. IAM 콘솔에서 표시된 정책을 선택합니다.
7. [정책 편집(Edit policy)]을 선택하고 다음 명령문을 추가합니다. 새 명령문에서, 공유하려는 로그 그룹의 ARN만 지정하는 것이 좋습니다. 다음 예를 참조하세요.

```
{
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:GetLogRecord",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "SharedLogGroup1ARN",
                "SharedLogGroup2ARN"
           ]
        },
```

8. 변경 사항 저장을 선택합니다.

대시보드 공유에 대한 IAM 정책에 *를 리소스로 사용하는 5가지 권한이 이미 포함되어 있는 경우 정책을 변경하고 공유하려는 로그 그룹의 ARN만 지정하는 것이 좋습니다. 예를 들어 이러한 권한의 Resource 섹션이 다음과 같은 경우

```
"Resource": "*"
```

다음 예와 같이 공유하려는 로그 그룹의 ARN만 지정하도록 정책을 변경합니다.

```
"Resource": [
  "SharedLogGroup1ARN",
  "SharedLogGroup2ARN"
]
```

공유받는 사용자가 사용자 지정 위젯을 볼 수 있도록 허용

대시보드를 공유할 때 기본적으로 대시보드에 있는 사용자 지정 위젯은 대시보드를 공유받는 사용자에게 표시되지 않습니다. 이는 현재 존재하는 사용자 지정 위젯과 공유 이후에 대시보드에 추가되는 항목에 모두 적용됩니다.

이러한 사용자가 사용자 지정 위젯을 볼 수 있도록 하려면 대시보드 공유를 위한 IAM 역할에 권한을 추가해야 합니다.

대시보드를 공유받는 사용자가 사용자 지정 위젯을 볼 수 있도록 허용하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 공유된 대시보드의 이름을 선택합니다.
4. [작업(Actions)], [대시보드 공유(Share dashboard)]를 선택합니다.
5. [리소스(Resources)]에서 [IAM 역할(IAM Role)]을 선택합니다.
6. IAM 콘솔에서 표시된 정책을 선택합니다.
7. [정책 편집(Edit policy)]을 선택하고 다음 명령문을 추가합니다. 새 명령문에서, 공유하려는 Lambda 함수의 ARN만 지정하는 것이 좋습니다. 다음 예를 참조하세요.

```
{
  "Sid": "Invoke",
  "Effect": "Allow",
  "Action": [
      "lambda:InvokeFunction"
  ],
  "Resource": [
    "LambdaFunction1ARN",
    "LambdaFunction2ARN"
  ]
 }
```

8. 변경 사항 저장을 선택합니다.

대시보드 공유를 위한 IAM 정책에 *를 리소스로 지정하는 해당 권한이 이미 포함되어 있는 경우 정책을 변경하고 공유하려는 Lambda 함수의 ARN만 지정하는 것이 좋습니다. 예를 들어 이러한 권한의 `Resource` 섹션이 다음과 같은 경우

```
"Resource": "*"
```

다음 예와 같이 공유하려는 사용자 지정 위젯의 ARN만 지정하도록 정책을 변경합니다.

```
"Resource": [
  "LambdaFunction1ARN",
  "LambdaFunction2ARN"
]
```

라이브 데이터 사용

지표 위젯에 라이브 데이터를 표시할지 여부를 선택할 수 있습니다. 라이브 데이터는 마지막 1분 이내에 게시된 완전히 집계되지 않은 데이터입니다.

- 라이브 데이터가 해제된 경우 집계 기간이 지난 1분 이상인 데이터 포인트만 표시됩니다. 예를 들어 5분의 기간을 사용하는 경우 12:35의 데이터 포인트는 12:35~12:40에 집계되고 12:41에 표시됩니다.
- 라이브 데이터가 설정되어 있으면 해당 집계 간격으로 데이터가 게시되는 즉시 최신 데이터 포인트가 표시됩니다. 화면을 새로 고칠 때마다 해당 집계 기간 내의 새 데이터가 게시됨에 따라 최신 데이터 포인트가 변경될 수 있습니다. 합계 또는 샘플 수와 같은 누적 통계에서 라이브 데이터를 사용하면 그래프 끝에 하락이 발생할 수 있습니다.

전체 대시보드 또는 대시보드의 개별 위젯에 대해 라이브 데이터를 사용하도록 선택할 수 있습니다.

전체 대시보드에서 라이브 데이터를 사용할지 여부를 선택하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 대시보드의 모든 위젯에서 라이브 데이터를 영구적으로 활성화하거나 비활성화하려면 다음을 수행합니다.
   a. 작업, 설정, Bulk update live data(라이브 데이터 대량 업데이트)를 선택합니다.
   b. Live Data on(라이브 데이터 설정) 또는 Live Data off(라이브 데이터 해제)를 선택하고 설정을 선택합니다.
4. 각 위젯의 라이브 데이터 설정을 일시적으로 재정의하려면 [작업(Actions)]을 선택합니다. 그런 다음, [재정의(Overrides)] 아래에 있는 [라이브 데이터(Live data)] 옆에서 다음 중 하나를 수행합니다.
   - 모든 위젯에서 라이브 데이터를 일시적으로 설정하려면 On(설정)을 선택합니다.
   - 모든 위젯에서 라이브 데이터를 일시적으로 해제하려면 Off(해제)를 선택합니다.
   - 각 위젯의 라이브 데이터 설정을 유지하려면 Do not override(재정의 안 함)를 선택합니다.

단일 위젯에서 라이브 데이터를 사용할지 여부를 선택하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 위젯을 선택하고 작업, 편집을 선택합니다.
4. 그래프 옵션 탭을 선택합니다.
5. Live Data(라이브 데이터) 아래의 확인란을 선택하거나 선택 취소합니다.

애니메이션 대시보드 보기

시간 경과에 따라 캡처된 CloudWatch 지표 데이터를 재생하는 애니메이션 대시보드를 볼 수 있습니다. 이렇게 하면 추세를 확인하며 프레젠테이션을 만들고 문제가 발생한 후 문제를 분석할 수 있습니다.

대시보드의 애니메이션 위젯에는 선 위젯, 누적 영역 위젯, 숫자 위젯, 지표 탐색기 위젯이 포함됩니다. 파이 그래프, 막대 차트, 텍스트 위젯, 로그 위젯은 대시보드에 표시되지만 애니메이션되지는 않습니다.

애니메이션 대시보드를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 대시보드 이름을 선택합니다.
4. [작업(Actions)], [대시보드 재생(Replay dashboard)]을 선택합니다.
5. (선택 사항) 기본적으로 애니메이션을 시작하면 애니메이션이 슬라이딩 창으로 표시됩니다. 대신 애니메이션을 요소별 애니메이션으로 표시하려면 애니메이션을 일시 중지한 상태에서 돋보기 아이콘을 선택하고 확대 및 축소를 재설정합니다.
6. 애니메이션을 시작하려면 재생 버튼을 선택합니다. 뒤로 및 앞으로 버튼을 선택하여 다른 시점으로 이동할 수도 있습니다.
7. (선택 사항) 애니메이션의 기간을 변경하려면 달력을 선택하고 기간을 선택합니다.
8. 애니메이션 속도를 변경하려면 [자동 속도(Auto speed)]를 선택하고 새 속도를 선택합니다.
9. 모두 마쳤으면 [애니메이션 종료(Exit animate)]를 선택합니다.

즐겨찾기 목록에 CloudWatch 대시보드 추가

CloudWatch 콘솔에서 대시보드, 경보 및 로그 그룹을 즐겨찾기 목록에 추가할 수 있습니다. 탐색 창에서 즐겨찾기 목록에 액세스할 수 있습니다. 다음 절차에서는 대시보드를 즐겨찾기 목록에 추가하는 방법에 대해 설명합니다.

즐겨찾기 목록에 대시보드 추가

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 대시보드 목록에서 즐겨찾기로 설정하려는 대시보드 이름 옆에 있는 별 기호를 선택합니다.
   - (선택 사항) 목록에서 대시보드를 선택하고 대시보드 이름 옆에 있는 별 기호를 선택하여 대시보드를 즐겨찾기로 설정할 수도 있습니다.
4. 즐겨찾기 목록에 액세스하려면 탐색 창에서 즐겨찾기 및 최근 항목(Favorites and recents)을 선택합니다. 메뉴에는 두 개의 열이 있습니다. 한 열에는 즐겨찾는 대시보드, 경보 및 로그 그룹이 포함되고, 다른 열에는 최근에 방문한 대시보드, 경보 및 로그 그룹이 포함됩니다.

> **Tip**
>
> CloudWatch 콘솔 탐색 창의 즐겨찾기 및 최근 항목(Favorites and recents) 메뉴에서 대시보드와 경보뿐만 아니라 로그 그룹을 즐겨찾기로 설정할 수 있습니다. 최근 방문(Recently visited) 열에서 즐겨찾기로 설정하려는 대시보드에 마우스를 가져가서 그 옆에 있는 별 기호를 선택합니다.

CloudWatch 대시보드에 대한 기간 재정의 설정 또는 새로 고침 간격 변경

이 대시보드에 추가된 그래프의 기간 설정을 유지하거나 수정하는 방법을 지정할 수 있습니다.

자동 기간 또는 지속 시간 범위를 위젯에 적용하면 그래프의 전체 시간 범위가 설정한 기간에 영향을 줄 수 있습니다.

- 시간 범위가 1일 이하인 경우 기간 설정은 변경되지 않습니다.
- 시간 범위가 1일에서 3일 사이인 경우 5분 미만으로 설정된 기간은 5분으로 변경됩니다.
- 시간 범위가 3일 이상인 경우 1시간 미만으로 설정된 기간은 1시간으로 변경됩니다.

다음 단계에서는 콘솔을 사용하여 기간 재정의 옵션을 변경하는 방법을 설명합니다. 대시보드의 JSON 구조에 있는 `periodOverride` 필드를 사용하여 변경할 수도 있습니다. 자세한 내용은 대시보드 본문 전체 구조를 참조하세요.

기간 재정의 옵션을 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 작업을 선택합니다.
3. 기간 재정의에서 다음 중 하나를 선택합니다.
   - 각 그래프에서 지표 기간을 대시보드의 시간 범위에 맞게 자동으로 조정하려면 자동을 선택합니다.
   - 각 그래프의 기간 설정을 항상 준수하려면 재정의하지 않습니다를 선택합니다.
   - 대시보드에 추가된 그래프의 기간 설정을, 선택된 해당 기간 설정으로 항상 조정하려면 다른 옵션 중 하나를 선택합니다.

   기간 재정의는 대시보드를 종료하거나 브라우저를 새로 고치면 항상 자동으로 재설정됩니다. 기간 재정의에 대해 설정을 여러 개 저장할 수는 없습니다.

CloudWatch 대시보드의 데이터가 새로 고쳐지는 빈도를 변경할 수 있습니다.

대시보드 새로 고침 간격을 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 새로 고침 옵션 메뉴(오른쪽 상단 모서리)에서 10초, 1분, 2분, 5분 또는 15분을 선택합니다.

CloudWatch 대시보드의 시간 범위 또는 시간대 형식 변경

대시보드 데이터가 표시되는 시간 범위를 분, 시간, 일 또는 주 단위로 변경할 수 있습니다. 대시보드 데이터가 표시되는 시간대 형식도 UTC 또는 현지 시간으로 변경할 수 있습니다. 현지 시간은 컴퓨터의 운영 체제에 지정된 시간대입니다.

> **Note**
>
> 100개 이상의 고분해능 지표를 포함하는 그래프를 사용하여 대시보드를 생성하는 경우 시간 범위를 1시간 이상으로 설정하지 않는 것이 좋습니다. 자세한 내용은 고분해능 지표 단원을 참조하십시오.

> **Note**
>
> 대시보드의 시간 범위가 대시보드의 위젯에 사용된 기간보다 짧으면 다음과 같은 상황이 발생합니다.
>
> - 대시보드의 시간 범위보다 길더라도 해당 위젯의 전체 기간 1개에 해당하는 데이터의 양을 표시하도록 위젯이 수정됩니다. 이렇게 하면 그래프에 데이터 포인트가 1개 이상 있을 수 있습니다.
> - 이 데이터 포인트의 기간에서 시작 시간은 적어도 하나의 데이터 포인트가 표시될 수 있도록 역방향으로 조정됩니다.

New console

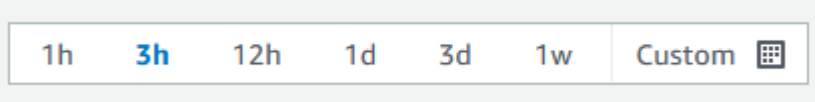
대시보드 시간 범위를 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 대시보드 화면에서 다음 중 하나를 수행합니다.
   - 대시보드의 상단 영역에서 미리 정의된 시간 범위 중 하나를 선택합니다. 이 범위는 1시간에서 1주까지 다양합니다(1시간, 3시간, 12시간, 1일, 1주).
   - 또는 다음과 같은 사용자 지정 시간 범위 옵션 중 하나를 선택할 수도 있습니다.
     - 사용자 지정(Custom)을 선택한 다음 상대(Relative) 탭을 선택합니다. 1분부터 15개월까지의 시간 범위를 선택합니다.
     - 사용자 지정(Custom)을 선택한 다음 절대(Absolute)를 선택합니다. 캘린더나 텍스트 필드를 사용하여 시간 범위를 지정합니다.

> **Tip**
>
> 그래프의 시간 범위를 변경할 때 집계 기간이 자동(Auto)으로 설정된 경우 CloudWatch가 기간을 변경할 수 있습니다. 기간을 수동으로 설정하려면 작업(Actions) 드롭다운을 선택한 다음 기간 재정의(Period override)를 선택합니다.

대시보드 시간대 형식을 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 대시보드 상단 영역에서 사용자 지정을 선택합니다.



4. 표시되는 상자 오른쪽 상단 모서리에서 UTC 또는 현지 시간(Local time)을 선택합니다.
5. 적용(Apply)을 선택합니다.

Old console

대시보드 시간 범위를 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 대시보드 화면에서 다음 중 하나를 수행합니다.
   - 대시보드의 상단 영역에서 미리 정의된 시간 범위 중 하나를 선택합니다. 이 범위는 1시간에서 1주까지 다양합니다(1시간, 3시간, 12시간, 1일, 3일, 1주).
   - 또는 다음과 같은 사용자 지정 시간 범위 옵션 중 하나를 선택할 수도 있습니다.
     - 사용자 지정(Custom) 드롭다운을 선택한 다음 상대(Relative) 탭을 선택합니다. 1분부터 15개월까지 사전 정의된 범위 중 하나를 선택합니다.
     - 사용자 지정(Custom) 드롭다운을 선택한 다음 절대(Absolute) 탭을 선택합니다. 캘린더나 텍스트 필드를 사용하여 시간 범위를 지정합니다.

> **Tip**
>
> 그래프의 시간 범위를 변경할 때 집계 기간이 자동(Auto)으로 설정된 경우 CloudWatch가 기간을 변경할 수 있습니다. 기간을 수동으로 설정하려면 작업(Actions) 드롭다운을 선택한 다음 기간 재정의(Period override)를 선택합니다.

대시보드 시간대 형식을 변경하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택한 다음, 대시보드를 선택합니다.
3. 대시보드 화면의 오른쪽 상단 모서리에서 사용자 지정(Custom) 드롭다운을 선택합니다.
4. 표시되는 상자의 오른쪽 상단 모서리에서 UTC 또는 현지 시간대(Local timezone)를 선택합니다.

Amazon CloudWatch 지표 사용

지표는 시스템 성능에 대한 데이터입니다. 기본적으로 많은 서비스에서 리소스(예: Amazon EC2 인스턴스, Amazon EBS 볼륨, Amazon RDS DB 인스턴스)에 대한 무료 지표를 제공합니다. Amazon EC2 인스턴스와 같은 일부 리소스에 대한 세부 모니터링을 사용하거나 자체 애플리케이션 지표를 게시할 수도 있습니다. Amazon CloudWatch는 검색, 그래프 작성, 경보를 위해 계정의 모든 지표(AWS 리소스 지표와 직접 제공한 애플리케이션 지표 모두)를 로드할 수 있습니다.

지표 데이터는 15개월 동안 보관되기 때문에 최신 데이터와 이력 데이터를 모두 볼 수 있습니다.

콘솔에서 지표를 그래프로 표시하려면 고성능 SQL 쿼리 엔진인 CloudWatch Metrics Insights를 사용하여 모든 지표 내의 추세와 패턴을 실시간으로 식별할 수 있습니다.

내용

기본 모니터링 및 세부 모니터링

CloudWatch는 기본 모니터링과 세부 모니터링이라는 두 가지 모니터링 범주를 제공합니다.

다수의 AWS 서비스는 고객에게 무료로 CloudWatch에 기본 지표 세트를 게시하여 기본 모니터링을 제공합니다. 기본적으로 AWS 서비스 중 하나를 사용하기 시작할 때 기본 모니터링이 자동으로 활성화됩니다. 기본 모니터링을 제공하는 서비스 목록은 CloudWatch 지표를 게시하는 AWS 서비스 단원을 참조하세요.

세부 모니터링은 일부 서비스에서만 제공됩니다. 또한 요금이 부과됩니다. AWS 서비스에 사용하려면 활성화하기를 선택해야 합니다. 요금에 대한 자세한 정보는 Amazon CloudWatch 비용을 참조하세요.

자세한 모니터링 옵션은 제공하는 서비스에 따라 다릅니다. 예를 들어, Amazon EC2 세부 모니터링은 Amazon EC2 기본 모니터링에 사용되는 5분 간격 대신 1분 간격으로 게시되는 더 빈번한 지표를 제공합니다. Apache Kafka에 대한 Amazon S3 및 Amazon 관리형 스트리밍에 대한 세부 모니터링은 더 세밀한 지표를 의미합니다.

다른 AWS 서비스, 상세한 모니터링도 다른 이름을 가지고 있습니다. 예를 들어 Amazon EC2에서는 세부 모니터링이라고 합니다. AWS Elastic Beanstalk에서는 이를 향상된 모니터링이라고 하며, Amazon S3에서는 이를 요청 지표라고 합니다.

Amazon EC2에 대한 세부 모니터링을 사용하면 Amazon EC2 리소스를 보다 효율적으로 관리할 수 있으므로 추세를 파악하고 조치를 더 빠르게 수행할 수 있습니다. 운영 문제를 신속하게 확인하여 조치하기 위해 Amazon S3 요청 지표를 1분 간격으로 사용할 수 있습니다. Amazon MSK에서 다음을 활성화할 때 `PER_BROKER`, `PER_TOPIC_PER_BROKER` 또는 `PER_TOPIC_PER_PARTITION` 레벨 모니터링을 통해 더 많은 가시성을 제공하는 추가 지표를 얻을 수 있습니다.

다음 표에는 세부 모니터링을 제공하는 서비스가 나와 있습니다. 또한 세부 모니터링에 대해 자세히 설명하고 활성화 방법에 대한 지침을 제공하는 서비스의 설명서 링크도 포함되어 있습니다.

Service	설명서
Amazon API Gateway	API Gateway 지표 측정기준
Amazon CloudFront	추가 CloudFront 배포 지표 보기
Amazon EC2	인스턴스에 대한 세부 모니터링 활성화 또는 비활성화
Elastic Beanstalk	향상된 상태 보고 및 모니터링
Amazon Kinesis Data Streams	향상된 샤드 수준 지표

Service	설명서
Amazon MSK	CloudWatch 지표를 사용한 Amazon MSK 모니터링
Amazon S3	CloudWatch의 Amazon S3 요청 지표
Amazon SES	Amazon SES 이벤트 게시를 사용하여 CloudWatch 세부 모니터링 지표를 수집합니다.

또한 CloudWatch는 다음 표에 표시된 바와 같이 일부 AWS 서비스에 대해 보다 자세한 지표와 사전 생성된 대시보드가 포함된 즉시 사용 가능한 모니터링 솔루션을 제공합니다.

Service	기능 설명서
Lambda	Lambda Insights
Amazon ECS	Amazon ECS용 컨테이너 인사이트
Amazon EKS	Amazon EKS 및 Kubernetes용 컨테이너 인사이트

CloudWatch Metrics Insights를 사용하는 지표 쿼리

CloudWatch Metrics Insights는 대규모 지표를 쿼리하는 데 사용할 수 있는 강력한 고성능 SQL 쿼리 엔진입니다. 모든 CloudWatch 지표 내에서 트렌드와 패턴을 실시간으로 식별할 수 있습니다.

단일 시계열을 반환하는 Metrics Insights 쿼리에 경보를 설정할 수도 있습니다. 이는 인프라 또는 애플리케이션 플릿에서 집계된 지표를 관찰하는 경보를 생성하는 데 특히 유용할 수 있습니다. 경보를 한 번 생성하면 플릿에서 리소스가 추가되거나 제거될 때 경보가 동적으로 조정됩니다.

CloudWatch Metrics Insights 쿼리 편집기를 사용하여 콘솔에서 CloudWatch Metrics Insights 쿼리를 수행할 수 있습니다. `GetMetricData` 또는 `PutDashboard`를 실행하여 AWS CLI 또는 AWS SDK로 CloudWatch Metrics Insights 쿼리를 수행할 수도 있습니다. CloudWatch Metrics Insights 쿼리 편집기로 실행하는 쿼리에는 요금이 부과되지 않습니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

CloudWatch Metrics Insights 쿼리 편집기를 사용하면 미리 구축된 다양한 샘플 쿼리 중에서 선택할 수 있고 자체 쿼리도 생성할 수도 있습니다. 쿼리를 생성할 때 작성기 뷰를 사용하여 기존 지표와 측정기준을 찾아볼 수 있습니다. 또는 편집기 뷰를 사용하여 수동으로 쿼리를 작성할 수도 있습니다.

또한 자연어를 사용하여 CloudWatch Metrics Insights 쿼리를 생성할 수 있습니다. 그러려면 찾고 있는 데이터에 대해 질문하거나 설명합니다. 이 AI 지원 기능은 프롬프트를 기반으로 쿼리를 생성하고 쿼리 작동 방식에 대한 라인별 설명을 제공합니다. 자세한 내용은 Use natural language to generate and update CloudWatch Metrics Insights queries를 참조하세요.

Metrics Insights를 사용하면 대규모로 쿼리를 실행할 수 있습니다. GROUP BY 절을 사용하면 특정 측정기준 값별로 지표를 실시간으로 개별 시계열로 그룹화할 수 있습니다. Metrics Insights 쿼리에는 ORDER BY 기능이 포함되므로 Metrics Insights를 사용하여 '상위 N' 유형의 쿼리를 만들 수 있습니다. 예를 들어, '상위 N' 유형의 쿼리는 계정에 있는 수백만 개의 지표를 검사하고 CPU를 가장 많이 사용하는 인스턴스 10개를 반환할 수 있습니다. 이를 통해 애플리케이션의 지연 문제를 정확히 찾아내 해결할 수 있습니다.

주제

- SQL 추론
- Metrics Insights 샘플 쿼리
- Metrics Insights 제한
- Metrics Insights 용어집
- Metrics Insights 문제 해결

쿼리 구축

CloudWatch 콘솔인 AWS CLI 또는 AWS SDK를 사용하여 CloudWatch Metrics Insights 쿼리를 실행할 수 있습니다. 콘솔에서 실행되는 쿼리는 무료입니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

Metrics Insights 쿼리 수행을 위해 AWS SDK를 사용하는 방법에 대한 자세한 내용은 GetMetricData를 참조하세요.

CloudWatch 콘솔을 사용하여 쿼리를 실행하려면 다음 단계를 따르세요.

Metrics Insights를 사용하는 지표 쿼리

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 모든 지표를 선택합니다.
3. 쿼리(Query) 탭을 선택합니다.
4. (선택 사항) 미리 구축된 샘플 쿼리를 실행하려면 쿼리 추가(Add query)를 선택하고 실행할 쿼리를 선택합니다. 이 쿼리에 만족하는 경우 절차의 나머지 부분은 건너뛸 수 있습니다. 또는 편집기(Editor)를 선택하여 샘플 쿼리를 편집한 다음 실행(Run)을 선택하여 수정된 쿼리를 실행합니다.
5. 자체 쿼리를 만들려면 빌더(Builder) 보기, 편집기(Editor) 보기를 사용하거나 두 가지를 결합해서 사용할 수도 있습니다. 두 보기 간에 전환할 수 있으며 두 보기 모두에서 진행 중인 작업을 확인할 수 있습니다.

   빌더(Builder) 보기에서 지표 네임스페이스, 지표 이름, 필터, 그룹 및 순서 옵션을 찾아 선택할 수 있습니다. 이러한 각 옵션에 대해 쿼리 빌더는 사용자 환경에서 선택할 수 있는 선택 목록을 제공합니다.

   Editor(편집기) 보기에서 쿼리 작성을 시작할 수 있습니다. 입력할 때 편집기는 지금까지 입력한 문자를 기반으로 제안 사항을 제공합니다.
6. 쿼리에 만족하는 경우 실행(Run)을 선택합니다.

7. (선택 사항) 그래프로 작성한 쿼리를 편집하는 또 다른 방법은 그래프로 표시된 지표(Graphed metrics) 탭을 선택한 다음 세부 정보(Details) 열의 쿼리 공식 옆에 있는 편집 아이콘을 선택합니다.
8. (선택 사항) 그래프에서 쿼리를 제거하려면 그래프로 표시된 지표(Graphed metrics)을 선택한 다음 쿼리를 표시하는 행의 오른쪽에 있는 X 아이콘을 선택합니다.

Metrics Insights 쿼리 구성 요소 및 구문

CloudWatch Metrics Insights 구문은 다음과 같습니다.

```
SELECT FUNCTION(metricName)
FROM namespace | SCHEMA(...)
[ WHERE labelKey OPERATOR labelValue [AND ... ] ]
[ GROUP BY labelKey [ , ... ] ]
[ ORDER BY FUNCTION() [ DESC | ASC ] ]
[ LIMIT number ]
```

Metrics Insights 쿼리에서 가능한 절은 다음과 같습니다. 키워드는 대소문자를 구분하지 않지만 지표 이름, 네임스페이스 및 측정기준 등의 식별자는 대소문자를 구분합니다.

SELECT

필수 사항입니다. 각 시간 버킷에서 관측치를 집계하는 데 사용할 함수를 지정합니다(제공된 기간까지 결정). 쿼리할 지표 이름도 지정합니다.

FUNCTION에 유효한 값은 AVG, COUNT, MAX, MIN 및SUM입니다.

- AVG는 쿼리와 일치하는 관측치의 평균을 계산합니다.
- COUNT는 쿼리와 일치하는 관측치의 개수를 반환합니다.
- MAX는 쿼리와 일치하는 관측치의 최대값을 반환합니다.
- MIN은 쿼리와 일치하는 관측치의 최소값을 반환합니다.
- SUM은 쿼리와 일치하는 관측치의 합계를 계산합니다.

FROM

필수 사항입니다. 지표의 소스을 지정합니다. 쿼리할 지표가 포함된 지표 네임스페이스 또는 SCHEMA 테이블 함수를 지정할 수 있습니다. 지표 네임스페이스의 예는 "AWS/EC2", "AWS/Lambda" 및 사용자 지정 지표에 대해 생성한 지표 네임스페이스입니다.

/ 또는 문자, 숫자 또는 밑줄이 아닌 다른 문자가 포함된 지표 네임스페이스는 큰따옴표로 묶어야 합니다. 자세한 내용은 따옴표 또는 이스케이프 문자는 무엇에 필요한가요? 단원을 참조하십시오.

스키마

FROM 절에서 사용할 수 있는 테이블 함수(선택 사항). SCHEMA를 사용하여 쿼리 결과를 측정기준 목록과 정확히 일치하는 지표 또는 측정기준이 없는 지표로 범위를 축소합니다.

SCHEMA 절을 사용하는 경우 적어도 하나의 인수를 포함해야 하며 이 첫 번째 인수는 쿼리되는 지표 네임스페이스여야 합니다. SCHEMA를 네임스페이스 인수만 사용해서 지정한 경우 결과 범위가 측정기준이 없는 지표로만 축소됩니다.

SCHEMA를 추가 인수와 함께 지정한 경우 네임스페이스 인수 뒤에 있는 추가 인수는 레이블 키여야 합니다. 레이블 키는 측정기준 이름이어야 합니다. 이러한 레이블 키 중 하나 이상을 지정하면 정확한 측정기준 집합이 있는 지표로만 결과 범위가 축소됩니다. 이러한 레이블 키의 순서는 중요하지 않습니다.

예:

- SELECT AVG(CPUUtilization) FROM "AWS/EC2"는 측정기준과 관계없이 `CPUUtilization` 네임스페이스에 있는 `AWS/EC2` 지표와 일치하면 단일 집계된 시계열을 반환합니다.
- SELECT AVG(CPUUtilization) FROM SCHEMA("AWS/EC2")는 정의되지 않은 측정기준이 없는 `AWS/EC2` 네임스페이스에서 `CPUUtilization` 지표와만 일치합니다.
- SELECT AVG(CPUUtilization) FROM SCHEMA("AWS/EC2", InstanceId)는 정확히 하나의 `InstanceId` 측정기준으로 CloudWatch에 보고된 `CPUUtilization` 지표와만 일치합니다.
- SELECT SUM(RequestCount) FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)는 정확히 `LoadBalancer` 및 `AvailabilityZone` 측정기준을 사용한 `AWS/ApplicationELB`에서 CloudWatch로 보고된 `RequestCount` 지표와만 일치합니다.

WHERE

선택 사항입니다. 1개 이상의 레이블 키에 대해 특정 레이블 값을 사용하여 지정된 표현식과 일치하는 지표로만 결과를 필터링합니다. 예를 들어 WHERE InstanceType = 'c3.4xlarge'는 결과를 `c3.4xlarge` 인스턴스 유형으로만 필터링하며 WHERE InstanceType != 'c3.4xlarge'는 `c3.4xlarge`를 제외한 모든 인스턴스 유형으로 결과를 필터링합니다.

모니터링 계정에서 쿼리를 실행하는 경우 `WHERE AWS.AccountId`를 사용하여 지정된 계정으로만 결과를 제한할 수 있습니다. 예를 들어, `WHERE AWS.AccountId=444455556666`은 계정 `444455556666`의 지표만 쿼리합니다. 모니터링 계정 자체의 지표로만 쿼리를 제한하려면 `WHERE AWS.AccountId=CURRENT_ACCOUNT_ID()`를 사용합니다.

레이블 값은 항상 작은따옴표로 묶어야 합니다.

지원되는 연산자

WHERE 절은 다음과 같은 연산자를 지원합니다.

- = 레이블 값은 지정된 문자열과 일치해야 합니다.
- != 레이블 값은 지정된 문자열과 일치하지 않아야 합니다.
- AND. 지정된 두 조건과 모두 일치하려면 true여야 합니다. 여러 AND 키워드를 사용하여 2개 이상의 조건을 지정할 수 있습니다.

GROUP BY

선택 사항입니다. 쿼리 결과를 여러 시계열로 그룹화합니다. 각 시계열은 지정된 레이블 키 또는 키에 대한 여러 값에 해당합니다. 예를 들어 `GROUP BY InstanceId`를 사용하면 `InstanceId`의 각 값에 대한 여러 시계열을 반환합니다. `GROUP BY ServiceName, Operation`을 사용하면 가능한 각 `ServiceName` 및 `Operation` 값에 대해 서로 다른 시계열을 생성합니다.

GROUP BY 절을 사용하면 기본적으로 결과는 GROUP BY 절에 지정된 레이블 시퀀스를 사용하여 알파벳 오름차순으로 정렬됩니다. 결과 순서를 변경하려면 ORDER BY 절을 쿼리에 추가합니다.

모니터링 계정에서 쿼리를 실행할 때 `GROUP BY AWS.AccountId`를 사용하여 결과를 가져온 계정을 기준으로 결과를 그룹화할 수 있습니다.

> **Note**
>
> 일치하는 지표 일부에 GROUP BY 절로 지정된 특정 레이블 키가 포함되어 있지 않은 경우 `Other`로 이름이 지정된 null 그룹이 반환됩니다. 예를 들어 `GROUP BY ServiceName, Operation`을 지정하고 반환된 지표 일부에 측정기준인 `ServiceName`이 포함되지 않은 경우 해당 지표는 `ServiceName` 값으로 `Other`를 표시합니다.

ORDER BY

선택 사항입니다. 쿼리가 2개 이상의 시계열을 반환하는 경우 반환된 시계열에 사용할 순서를 지정합니다. 순서는 ORDER BY 절에서 지정한 FUNCTION으로 찾은 값을 기준으로 합니다. FUNCTION은 반환된 각 시계열에서 단일 스칼라 값을 계산하는 데 사용되며 해당 값은 순서를 결정하는 데 사용됩니다.

ASC로 오름차순 사용 여부 또는 DESC로 내림차순 사용 여부를 지정할 수도 있습니다. 생략하는 경우 기본값은 ASC로 오름차순입니다.

예를 들어, `ORDER BY MAX() DESC` 절을 추가하면 시간 범위 내에서 관찰된 최대 데이터 포인트별로 결과를 내림차순으로 정렬합니다. 즉, 최대 데이터 포인트가 가장 높은 시계열이 먼저 반환됩니다.

ORDER BY 절에서 사용할 유효 함수는 `AVG()`, `COUNT()`, `MAX()`, `MIN()` 및 `SUM()`입니다.

ORDER BY 절을 LIMIT 절과 함께 사용하는 경우 결과 쿼리는 "상위 N" 쿼리입니다. ORDER BY로 각 쿼리는 500개 이하의 시계열을 반환할 수 있으므로 많은 수의 지표를 반환할 수 있는 쿼리에도 유용합니다. 쿼리가 500개 이상의 시계열과 일치하고 ORDER BY 절을 사용하는 경우 시계열이 정렬된 후 정렬 순서에서 먼저 나오는 500개 시계열이 반환되는 시계열입니다.

LIMIT

선택 사항입니다. 쿼리에 의해 반환되는 시계열 수를 지정한 값으로 제한합니다. 지정할 수 있는 최대값은 500이며 LIMIT을 지정하지 않은 쿼리도 500개 이하의 시계열을 반환할 수 있습니다.

ORDER BY가 있는 LIMIT 절을 사용하면 "상위 N" 쿼리를 제공합니다.

따옴표 또는 이스케이프 문자는 무엇에 필요한가요?

쿼리에서 레이블 값은 항상 작은따옴표로 묶어야 합니다. 예: SELECT MAX(CPUUtilization) FROM "AWS/EC2" WHERE AutoScalingGroupName = 'my-production-fleet'.

문자, 숫자 및 밑줄(_) 이외의 문자가 포함된 지표 네임스페이스, 지표 이름 및 레이블 키는 큰따옴표로 묶어야 합니다. 예: SELECT MAX("My.Metric").

이들 중 하나에 큰따옴표나 작은따옴표가 포함된 경우(예:`Bytes"Input"`) SELECT AVG("Bytes \"Input\"")와 같이 백슬래시를 사용하여 각 따옴표를 이스케이프 처리해야 합니다.

지표 네임스페이스, 지표 이름 또는 레이블 키에 Metrics Insights의 예약된 키워드 단어가 포함되어 있는 경우 이러한 단어도 큰따옴표로 묶어야 합니다. 예를 들어 `LIMIT`으로 이름 붙인 지표가 있는 경우 `SELECT AVG("LIMIT")`를 사용합니다. 예약어가 포함되어 있지 않더라도 네임스페이스, 지표 이름 또는 레이블을 큰따옴표로 묶는 것도 유효합니다.

전체 예약어 목록은 예약어 섹션을 참조하세요.

리치 쿼리 작성 단계별 방법

이 섹션에서는 가능한 모든 절을 단계별로 사용하는 전체 예제를 작성하는 방법을 설명합니다.

측정기준 `LoadBalancer` 및 `AvailabilityZone`을 사용하여 수집된 모든 Application Load Balancer `RequestCount` 지표를 집계하는 다음 쿼리로 시작합니다.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
```

이제 특정 로드 밸런서의 지표만 보고 싶다면 WHERE 절을 추가하여 LoadBalancer 측정기준 값이 app/load-balancer-1인 해당 지표만 반환되는 지표로 제한할 수 있습니다.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
```

앞의 쿼리는 이 로드 밸런서에 대한 모든 가용 영역의 RequestCount 지표를 하나의 시계열로 집계합니다. 각 가용 영역에 대해 서로 다른 시계열을 보려면 GROUP BY 절을 추가할 수 있습니다.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
```

다음으로 가장 높은 값을 먼저 확인하기 위해 이러한 결과를 정렬할 수 있습니다. 다음 ORDER BY 절은 쿼리 시간 범위 동안 각 시계열에 의해 보고된 최대값만큼 내림차순으로 시계열을 정렬합니다.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
ORDER BY MAX() DESC
```

마지막으로 "상위 N" 유형의 쿼리에 주로 관심이 있다면 LIMIT 절을 사용할 수 있습니다. 마지막 예에서는 결과를 상위 5개의 MAX 값을 가진 시계열로만 제한합니다.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
ORDER BY MAX() DESC
LIMIT 5
```

크로스 계정 쿼리 예제

이러한 예제는 CloudWatch 크로스 계정 관찰성에서 모니터링 계정으로 설정된 계정에서 실행할 때 유효합니다.

다음 예제에서는 소스 계정 123456789012에서 모든 Amazon EC2 인스턴스를 검색하고 평균을 반환합니다.

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
WHERE AWS.AccountId ='123456789012'
```

다음 예제는 연결된 모든 소스 계정에서 AWS/EC2 의 CPUUtilization 지표를 쿼리하고 결과를 계정 ID 및 인스턴스 유형별로 그룹화합니다.

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
GROUP BY AWS.AccountId, InstanceType
```

다음 예제는 모니터링 계정 자체에서 CPUUtilization을 쿼리합니다.

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
WHERE AWS.AccountId = CURRENT_ACCOUNT_ID()
```

예약어

다음은 CloudWatch 지표 인사이트의 예약된 키워드입니다. 이러한 단어가 쿼리의 네임스페이스, 지표 이름 또는 레이블 키에 있는 경우 큰따옴표로 묶어야 합니다. 예약어는 대소문자를 구분하지 않습니다.

```
"ABORT" "ABORTSESSION" "ABS" "ABSOLUTE" "ACCESS" "ACCESSIBLE" "ACCESS_LOCK" "ACCOUNT"
 "ACOS" "ACOSH" "ACTION" "ADD" "ADD_MONTHS"
"ADMIN" "AFTER" "AGGREGATE" "ALIAS" "ALL" "ALLOCATE" "ALLOW" "ALTER" "ALTERAND" "AMP"
 "ANALYSE" "ANALYZE" "AND" "ANSIDATE" "ANY" "ARE" "ARRAY",
"ARRAY_AGG" "ARRAY_EXISTS" "ARRAY_MAX_CARDINALITY" "AS" "ASC" "ASENSITIVE" "ASIN"
 "ASINH" "ASSERTION" "ASSOCIATE" "ASUTIME" "ASYMMETRIC" "AT",
"ATAN" "ATAN2" "ATANH" "ATOMIC" "AUDIT" "AUTHORIZATION" "AUX" "AUXILIARY" "AVE"
 "AVERAGE" "AVG" "BACKUP" "BEFORE" "BEGIN" "BEGIN_FRAME" "BEGIN_PARTITION",
"BETWEEN" "BIGINT" "BINARY" "BIT" "BLOB" "BOOLEAN" "BOTH" "BREADTH" "BREAK" "BROWSE"
 "BT" "BUFFERPOOL" "BULK" "BUT" "BY" "BYTE" "BYTEINT" "BYTES" "CALL",
```

```
"CALLED" "CAPTURE" "CARDINALITY" "CASCADE" "CASCADED" "CASE" "CASESPECIFIC" "CASE_N"
 "CAST" "CATALOG" "CCSID" "CD" "CEIL" "CEILING" "CHANGE" "CHAR",
"CHAR2HEXINT" "CHARACTER" "CHARACTERS" "CHARACTER_LENGTH" "CHARS" "CHAR_LENGTH" "CHECK"
 "CHECKPOINT" "CLASS" "CLASSIFIER" "CLOB" "CLONE" "CLOSE" "CLUSTER",
"CLUSTERED" "CM" "COALESCE" "COLLATE" "COLLATION" "COLLECT" "COLLECTION" "COLLID"
 "COLUMN" "COLUMN_VALUE" "COMMENT" "COMMIT" "COMPLETION" "COMPRESS" "COMPUTE",
"CONCAT" "CONCURRENTLY" "CONDITION" "CONNECT" "CONNECTION" "CONSTRAINT" "CONSTRAINTS"
 "CONSTRUCTOR" "CONTAINS" "CONTAINSTABLE" "CONTENT" "CONTINUE" "CONVERT",
"CONVERT_TABLE_HEADER" "COPY" "CORR" "CORRESPONDING" "COS" "COSH" "COUNT" "COVAR_POP"
 "COVAR_SAMP" "CREATE" "CROSS" "CS" "CSUM" "CT" "CUBE" "CUME_DIST",
"CURRENT" "CURRENT_CATALOG" "CURRENT_DATE" "CURRENT_DEFAULT_TRANSFORM_GROUP"
 "CURRENT_LC_CTYPE" "CURRENT_PATH" "CURRENT_ROLE" "CURRENT_ROW" "CURRENT_SCHEMA",
"CURRENT_SERVER" "CURRENT_TIME" "CURRENT_TIMESTAMP" "CURRENT_TIMEZONE"
 "CURRENT_TRANSFORM_GROUP_FOR_TYPE" "CURRENT_USER" "CURRVAL" "CURSOR" "CV" "CYCLE"
 "DATA",
"DATABASE" "DATABASES" "DATABLOCKSIZE" "DATE" "DATEFORM" "DAY" "DAYS" "DAY_HOUR"
 "DAY_MICROSECOND" "DAY_MINUTE" "DAY_SECOND" "DBCC" "DBINFO" "DEALLOCATE" "DEC",
"DECFLOAT" "DECIMAL" "DECLARE" "DEFAULT" "DEFERRABLE" "DEFERRED" "DEFINE" "DEGREES"
 "DEL" "DELAYED" "DELETE" "DENSE_RANK" "DENY" "DEPTH" "DEREF" "DESC" "DESCRIBE",
"DESCRIPTOR" "DESTROY" "DESTRUCTOR" "DETERMINISTIC" "DIAGNOSTIC" "DIAGNOSTICS"
 "DICTIONARY" "DISABLE" "DISABLED" "DISALLOW" "DISCONNECT" "DISK" "DISTINCT",
"DISTINCTROW" "DISTRIBUTED" "DIV" "DO" "DOCUMENT" "DOMAIN" "DOUBLE" "DROP" "DSSIZE"
 "DUAL" "DUMP" "DYNAMIC" "EACH" "ECHO" "EDITPROC" "ELEMENT" "ELSE" "ELSEIF",
"EMPTY" "ENABLED" "ENCLOSED" "ENCODING" "ENCRYPTION" "END" "END-EXEC" "ENDING"
 "END_FRAME" "END_PARTITION" "EQ" "EQUALS" "ERASE" "ERRLVL" "ERROR" "ERRORFILES",
"ERRORTABLES" "ESCAPE" "ESCAPED" "ET" "EVERY" "EXCEPT" "EXCEPTION" "EXCLUSIVE" "EXEC"
 "EXECUTE" "EXISTS" "EXIT" "EXP" "EXPLAIN" "EXTERNAL" "EXTRACT" "FALLBACK
"FALSE" "FASTEXPORT" "FENCED" "FETCH" "FIELDPROC" "FILE" "FILLFACTOR" "FILTER" "FINAL"
 "FIRST" "FIRST_VALUE" "FLOAT" "FLOAT4" "FLOAT8" "FLOOR"
"FOR" "FORCE" "FOREIGN" "FORMAT" "FOUND" "FRAME_ROW" "FREE" "FREESPACE" "FREETEXT"
 "FREETEXTTABLE" "FREEZE" "FROM" "FULL" "FULLTEXT" "FUNCTION"
"FUSION" "GE" "GENERAL" "GENERATED" "GET" "GIVE" "GLOBAL" "GO" "GOTO" "GRANT" "GRAPHIC"
 "GROUP" "GROUPING" "GROUPS" "GT" "HANDLER" "HASH"
"HASHAMP" "HASHBAKAMP" "HASHBUCKET" "HASHROW" "HAVING" "HELP" "HIGH_PRIORITY" "HOLD"
 "HOLDLOCK" "HOUR" "HOURS" "HOUR_MICROSECOND" "HOUR_MINUTE"
"HOUR_SECOND" "IDENTIFIED" "IDENTITY" "IDENTITYCOL" "IDENTITY_INSERT" "IF" "IGNORE"
 "ILIKE" "IMMEDIATE" "IN" "INCLUSIVE" "INCONSISTENT" "INCREMENT"
"INDEX" "INDICATOR" "INFILE" "INHERIT" "INITIAL" "INITIALIZE" "INITIALLY" "INITIATE"
 "INNER" "INOUT" "INPUT" "INS" "INSENSITIVE" "INSERT" "INSTEAD"
"INT" "INT1" "INT2" "INT3" "INT4" "INT8" "INTEGER" "INTEGERDATE" "INTERSECT"
 "INTERSECTION" "INTERVAL" "INTO" "IO_AFTER_GTIDS" "IO_BEFORE_GTIDS"
"IS" "ISNULL" "ISOBID" "ISOLATION" "ITERATE" "JAR" "JOIN" "JOURNAL" "JSON_ARRAY"
 "JSON_ARRAYAGG" "JSON_EXISTS" "JSON_OBJECT" "JSON_OBJECTAGG"
```

```
"JSON_QUERY" "JSON_TABLE" "JSON_TABLE_PRIMITIVE" "JSON_VALUE" "KEEP" "KEY" "KEYS"
 "KILL" "KURTOSIS" "LABEL" "LAG" "LANGUAGE" "LARGE" "LAST"
"LAST_VALUE" "LATERAL" "LC_CTYPE" "LE" "LEAD" "LEADING" "LEAVE" "LEFT" "LESS" "LEVEL"
 "LIKE" "LIKE_REGEX" "LIMIT" "LINEAR" "LINENO" "LINES"
"LISTAGG" "LN" "LOAD" "LOADING" "LOCAL" "LOCALE" "LOCALTIME" "LOCALTIMESTAMP" "LOCATOR"
 "LOCATORS" "LOCK" "LOCKING" "LOCKMAX" "LOCKSIZE" "LOG"
"LOG10" "LOGGING" "LOGON" "LONG" "LONGBLOB" "LONGTEXT" "LOOP" "LOWER" "LOW_PRIORITY"
 "LT" "MACRO" "MAINTAINED" "MAP" "MASTER_BIND"
"MASTER_SSL_VERIFY_SERVER_CERT" "MATCH" "MATCHES" "MATCH_NUMBER" "MATCH_RECOGNIZE"
 "MATERIALIZED" "MAVG" "MAX" "MAXEXTENTS" "MAXIMUM" "MAXVALUE"
"MCHARACTERS" "MDIFF" "MEDIUMBLOB" "MEDIUMINT" "MEDIUMTEXT" "MEMBER" "MERGE" "METHOD"
 "MICROSECOND" "MICROSECONDS" "MIDDLEINT" "MIN" "MINDEX"
"MINIMUM" "MINUS" "MINUTE" "MINUTES" "MINUTE_MICROSECOND" "MINUTE_SECOND" "MLINREG"
 "MLOAD" "MLSLABEL" "MOD" "MODE" "MODIFIES" "MODIFY"
"MODULE" "MONITOR" "MONRESOURCE" "MONSESSION" "MONTH" "MONTHS" "MSUBSTR" "MSUM"
 "MULTISET" "NAMED" "NAMES" "NATIONAL" "NATURAL" "NCHAR" "NCLOB"
"NE" "NESTED_TABLE_ID" "NEW" "NEW_TABLE" "NEXT" "NEXTVAL" "NO" "NOAUDIT" "NOCHECK"
 "NOCOMPRESS" "NONCLUSTERED" "NONE" "NORMALIZE" "NOT" "NOTNULL"
"NOWAIT" "NO_WRITE_TO_BINLOG" "NTH_VALUE" "NTILE" "NULL" "NULLIF" "NULLIFZERO" "NULLS"
 "NUMBER" "NUMERIC" "NUMPARTS" "OBID" "OBJECT" "OBJECTS"
"OCCURRENCES_REGEX" "OCTET_LENGTH" "OF" "OFF" "OFFLINE" "OFFSET" "OFFSETS" "OLD"
 "OLD_TABLE" "OMIT" "ON" "ONE" "ONLINE" "ONLY" "OPEN" "OPENDATASOURCE"
"OPENQUERY" "OPENROWSET" "OPENXML" "OPERATION" "OPTIMIZATION" "OPTIMIZE"
 "OPTIMIZER_COSTS" "OPTION" "OPTIONALLY" "OR" "ORDER" "ORDINALITY" "ORGANIZATION"
"OUT" "OUTER" "OUTFILE" "OUTPUT" "OVER" "OVERLAPS" "OVERLAY" "OVERRIDE" "PACKAGE" "PAD"
 "PADDED" "PARAMETER" "PARAMETERS" "PART" "PARTIAL" "PARTITION"
"PARTITIONED" "PARTITIONING" "PASSWORD" "PATH" "PATTERN" "PCTFREE" "PER" "PERCENT"
 "PERCENTILE" "PERCENTILE_CONT" "PERCENTILE_DISC" "PERCENT_RANK" "PERIOD" "PERM"
"PERMANENT" "PIECESIZE" "PIVOT" "PLACING" "PLAN" "PORTION" "POSITION" "POSITION_REGEX"
 "POSTFIX" "POWER" "PRECEDES" "PRECISION" "PREFIX" "PREORDER"
"PREPARE" "PRESERVE" "PREVVAL" "PRIMARY" "PRINT" "PRIOR" "PRIQTY" "PRIVATE"
 "PRIVILEGES" "PROC" "PROCEDURE" "PROFILE" "PROGRAM" "PROPORTIONAL"
"PROTECTION" "PSID" "PTF" "PUBLIC" "PURGE" "QUALIFIED" "QUALIFY" "QUANTILE" "QUERY"
 "QUERYNO" "RADIANS" "RAISERROR" "RANDOM" "RANGE" "RANGE_N" "RANK"
"RAW" "READ" "READS" "READTEXT" "READ_WRITE" "REAL" "RECONFIGURE" "RECURSIVE" "REF"
 "REFERENCES" "REFERENCING" "REFRESH" "REGEXP" "REGR_AVGX" "REGR_AVGY"
"REGR_COUNT" "REGR_INTERCEPT" "REGR_R2" "REGR_SLOPE" "REGR_SXX" "REGR_SXY" "REGR_SYY"
 "RELATIVE" "RELEASE" "RENAME" "REPEAT" "REPLACE" "REPLICATION"
"REPOVERRIDE" "REQUEST" "REQUIRE" "RESIGNAL" "RESOURCE" "RESTART" "RESTORE" "RESTRICT"
 "RESULT" "RESULT_SET_LOCATOR" "RESUME" "RET" "RETRIEVE" "RETURN"
"RETURNING" "RETURNS" "REVALIDATE" "REVERT" "REVOKE" "RIGHT" "RIGHTS" "RLIKE" "ROLE"
 "ROLLBACK" "ROLLFORWARD" "ROLLUP" "ROUND_CEILING" "ROUND_DOWN"
"ROUND_FLOOR" "ROUND_HALF_DOWN" "ROUND_HALF_EVEN" "ROUND_HALF_UP" "ROUND_UP" "ROUTINE"
 "ROW" "ROWCOUNT" "ROWGUIDCOL" "ROWID" "ROWNUM" "ROWS" "ROWSET"
```

```
"ROW_NUMBER" "RULE" "RUN" "RUNNING" "SAMPLE" "SAMPLEID" "SAVE" "SAVEPOINT" "SCHEMA"
 "SCHEMAS" "SCOPE" "SCRATCHPAD" "SCROLL" "SEARCH" "SECOND" "SECONDS"
"SECOND_MICROSECOND" "SECQTY" "SECTION" "SECURITY" "SECURITYAUDIT" "SEEK" "SEL"
 "SELECT" "SEMANTICKEYPHRASETABLE" "SEMANTICSIMILARITYDETAILSTABLE"
"SEMANTICSIMILARITYTABLE" "SENSITIVE" "SEPARATOR" "SEQUENCE" "SESSION" "SESSION_USER"
 "SET" "SETRESRATE" "SETS" "SETSESSRATE" "SETUSER" "SHARE" "SHOW"
"SHUTDOWN" "SIGNAL" "SIMILAR" "SIMPLE" "SIN" "SINH" "SIZE" "SKEW" "SKIP" "SMALLINT"
 "SOME" "SOUNDEX" "SOURCE" "SPACE" "SPATIAL" "SPECIFIC" "SPECIFICTYPE"
"SPOOL" "SQL" "SQLEXCEPTION" "SQLSTATE" "SQLTEXT" "SQLWARNING" "SQL_BIG_RESULT"
 "SQL_CALC_FOUND_ROWS" "SQL_SMALL_RESULT" "SQRT" "SS" "SSL" "STANDARD"
"START" "STARTING" "STARTUP" "STAT" "STATE" "STATEMENT" "STATIC" "STATISTICS" "STAY"
 "STDDEV_POP" "STDDEV_SAMP" "STEPINFO" "STOGROUP" "STORED" "STORES"
"STRAIGHT_JOIN" "STRING_CS" "STRUCTURE" "STYLE" "SUBMULTISET" "SUBSCRIBER" "SUBSET"
 "SUBSTR" "SUBSTRING" "SUBSTRING_REGEX" "SUCCEEDS" "SUCCESSFUL"
"SUM" "SUMMARY" "SUSPEND" "SYMMETRIC" "SYNONYM" "SYSDATE" "SYSTEM" "SYSTEM_TIME"
 "SYSTEM_USER" "SYSTIMESTAMP" "TABLE" "TABLESAMPLE" "TABLESPACE" "TAN"
"TANH" "TBL_CS" "TEMPORARY" "TERMINATE" "TERMINATED" "TEXTSIZE" "THAN" "THEN"
 "THRESHOLD" "TIME" "TIMESTAMP" "TIMEZONE_HOUR" "TIMEZONE_MINUTE" "TINYBLOB"
"TINYINT" "TINYTEXT" "TITLE" "TO" "TOP" "TRACE" "TRAILING" "TRAN" "TRANSACTION"
 "TRANSLATE" "TRANSLATE_CHK" "TRANSLATE_REGEX" "TRANSLATION" "TREAT"
"TRIGGER" "TRIM" "TRIM_ARRAY" "TRUE" "TRUNCATE" "TRY_CONVERT" "TSEQUAL" "TYPE" "UC"
 "UESCAPE" "UID" "UNDEFINED" "UNDER" "UNDO" "UNION" "UNIQUE"
"UNKNOWN" "UNLOCK" "UNNEST" "UNPIVOT" "UNSIGNED" "UNTIL" "UPD" "UPDATE" "UPDATETEXT"
 "UPPER" "UPPERCASE" "USAGE" "USE" "USER" "USING" "UTC_DATE"
"UTC_TIME" "UTC_TIMESTAMP" "VALIDATE" "VALIDPROC" "VALUE" "VALUES" "VALUE_OF"
 "VARBINARY" "VARBYTE" "VARCHAR" "VARCHAR2" "VARCHARACTER" "VARGRAPHIC"
"VARIABLE" "VARIADIC" "VARIANT" "VARYING" "VAR_POP" "VAR_SAMP" "VCAT" "VERBOSE"
 "VERSIONING" "VIEW" "VIRTUAL" "VOLATILE" "VOLUMES" "WAIT" "WAITFOR"
"WHEN" "WHENEVER" "WHERE" "WHILE" "WIDTH_BUCKET" "WINDOW" "WITH" "WITHIN"
 "WITHIN_GROUP" "WITHOUT" "WLM" "WORK" "WRITE" "WRITETEXT" "XMLCAST" "XMLEXISTS"
"XMLNAMESPACES" "XOR" "YEAR" "YEARS" "YEAR_MONTH" "ZEROFILL" "ZEROIFNULL" "ZONE"
```

Metrics Insights 쿼리에 대한 경보 생성

Metrics Insights 쿼리에 대한 경보를 생성할 수 있습니다. 이를 통해 나중에 업데이트할 필요 없이 여러 리소스를 추적하는 경보를 생성할 수 있습니다. 쿼리는 새 리소스와 변경되는 리소스를 캡처합니다. 예를 들어 플릿의 CPU 사용률을 관찰하는 경보를 생성할 수 있으며 경보는 경보 생성 후 시작하는 새 인스턴스를 자동으로 평가합니다.

CloudWatch 크로스 계정 관찰성을 위해 설정된 모니터링 계정에서 Metrics Insights 경보는 소스 계정과 모니터링 계정 자체의 리소스를 관찰할 수 있습니다. 경보 쿼리를 특정 계정으로 제한하거나 결과를

계정 ID별로 그룹화하는 방법에 대한 자세한 내용은 Metrics Insights 쿼리 구성 요소 및 구문의 WHERE 및 GROUP BY 섹션을 참조하세요.

목차

Metrics Insights 경보 생성

콘솔을 사용하여 Metrics Insights 쿼리에 대한 경보를 생성하려면 다음을 수행하세요.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 모든 지표를 선택합니다.
3. 쿼리(Query) 탭을 선택합니다.
4. (선택 사항) 미리 구축된 샘플 쿼리를 실행하려면 쿼리 추가(Add query)를 선택하고 실행할 쿼리를 선택합니다. 또는 편집기(Editor)를 선택하여 샘플 쿼리를 편집한 다음 실행(Run)을 선택하여 수정된 쿼리를 실행합니다.
5. 자체 쿼리를 생성하려면 Builder(빌더) 보기 또는 Editor(편집기) 보기를 사용하거나 두 가지를 결합해서 사용합니다. 두 보기 간에 전환할 수 있으며 두 보기 모두에서 진행 중인 작업을 확인할 수 있습니다.

   빌더(Builder) 보기에서 지표 네임스페이스, 지표 이름, 필터, 그룹 및 순서 옵션을 찾아 선택할 수 있습니다. 이러한 각 옵션에 대해 쿼리 빌더는 사용자 환경에서 선택할 수 있는 선택 목록을 제공합니다.

   Editor(편집기) 보기에서 쿼리 작성을 시작할 수 있습니다. 입력할 때 편집기는 지금까지 입력한 문자를 기반으로 제안 사항을 제공합니다.

   > **Important**
   >
   > Metrics Insights 쿼리에 대한 경보를 설정하려면 쿼리가 단일 시계열을 반환해야 합니다. GROUP BY 문이 포함된 경우 표현식의 최종 결과로 하나의 시계열만 반환하는 지표 수학 표현식 내에 GROUP BY 문을 래핑해야 합니다.

6. 쿼리에 만족하는 경우 실행(Run)을 선택합니다.
7. 경보 생성(Create alarm)을 선택하십시오.

8. 조건에서 다음을 지정합니다.

   a. 지표가 **### ##** 항상에서 지표가 임곗값보다 크거나, 작거나, 같아야 하는지 여부를 지정합니다. than...에서 임곗값을 지정합니다.

   b. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

      N 중 M 경보를 생성하려면 두 번째 값에 지정한 값보다 낮은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가 단원을 참조하세요.

   c. 누락 데이터 처리에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하세요.

9. 다음(Next)을 선택합니다.

10. 알림(Notification)에서 경보가 ALARM 상태, OK 상태 또는 INSUFFICIENT_DATA 상태일 때 알릴 SNS 주제를 선택합니다.

    경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

    경보에서 알림을 보내지 않게 하려면 제거를 선택합니다.

11. 경보가 Auto Scaling, EC2 또는 Systems Manager 작업을 수행하도록 하려면 해당 버튼을 선택하고 경보 상태 및 수행할 작업을 선택합니다. 경보는 ALARM 상태가 될 때만 Systems Manager 작업을 수행할 수 있습니다. Systems Manager 작업에 대한 자세한 내용은 경보에서 OpsItem을 생성하도록 CloudWatch 구성 및 인시던트 생성을 참조하세요.

    > **Note**
    >
    > SSM Incident Manager 작업을 수행하는 경보를 생성하려면 특정 권한이 있어야 합니다. 자세한 내용은 AWS Systems Manager Incident Manager의 자격 증명 기반 정책 예 단원을 참조하세요.

12. 마친 후에는 다음을 선택합니다.

13. 경보 이름 및 설명을 입력합니다. 이름은 ASCII 문자만 포함해야 합니다. 그리고 다음(Next)을 선택합니다.

14. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

AWS CLI를 사용하여 Metrics Insights 쿼리에 대한 경보를 생성하려면 다음을 수행하세요.

- `put-metric-alarm` 명령을 사용하고 `metrics` 파라미터에 Metrics Insights 쿼리를 지정합니다. 예를 들어, 다음 명령은 인스턴스의 CPU 사용률이 50%를 초과할 경우 ALARM 상태가 되는 경보를 설정합니다.

```
aws cloudwatch put-metric-alarm --alarm-name Metrics-Insights-alarm --
evaluation-periods 1 --comparison-operator GreaterThanThreshold --metrics
 '[{"Id":"m1","Expression":"SELECT MAX(CPUUtilization) FROM SCHEMA(\"AWS/EC2\",
 InstanceId)", "Period":60}]' --threshold 50
```

부분적 데이터 사례

경보에 사용된 Metrics Insights 쿼리가 10,000여 개의 지표와 일치하는 경우 쿼리가 찾은 처음 10,000개의 지표를 기반으로 경보가 평가됩니다. 이는 부분 데이터에 대해 경보가 평가되고 있음을 의미합니다.

다음 방법을 사용하여 Metrics Insights 경보가 현재 부분 데이터를 기반으로 경보 상태를 평가하고 있는지 여부를 확인할 수 있습니다.

- 콘솔에서 Details(세부 정보) 페이지를 보기 위해 경보를 선택하면 해당 페이지에 Evaluation warning: Not evaluating all data(평가 경고: 모든 데이터를 평가하지 않음) 메시지가 나타납니다.
- describe-alarms AWS CLI 명령 또는 DescribeAlarms API를 사용하면 `EvaluationState` 필드에 `PARTIAL_DATA` 값이 표시됩니다.

경보는 부분 데이터 상태가 될 때 Amazon EventBridge에 이벤트를 게시하므로 EventBridge 규칙을 생성하여 이러한 이벤트를 관찰할 수 있습니다. 이러한 이벤트에서 `evaluationState` 필드의 값은 `PARTIAL_DATA`입니다. 다음은 예입니다.

```
{
    "version": "0",
    "id": "12345678-3bf9-6a09-dc46-12345EXAMPLE",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-11-08T11:26:05Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:my-alarm-name"
```

```
    ],
    "detail": {
        "alarmName": "my-alarm-name",
        "state": {
            "value": "ALARM",
            "reason": "Threshold Crossed: 3 out of the last 3 datapoints [20000.0
 (08/11/22 11:25:00), 20000.0 (08/11/22 11:24:00), 20000.0 (08/11/22 11:23:00)] were
 greater than the threshold (0.0) (minimum 1 datapoint for OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2022-11-08T11:26:05.399+0000\",\"startDate\":\"2022-11-08T11:23:00.000+0000\",
\"period\":60,\"recentDatapoints\":[20000.0,20000.0,20000.0],\"threshold\":0.0,
\"evaluatedDatapoints\":[{\"timestamp\":\"2022-11-08T11:25:00.000+0000\",\"value
\":20000.0}]}",
            "timestamp": "2022-11-08T11:26:05.401+0000",
            "evaluationState": "PARTIAL_DATA"
        },
        "previousState": {
            "value": "INSUFFICIENT_DATA",
            "reason": "Unchecked: Initial alarm creation",
            "timestamp": "2022-11-08T11:25:51.227+0000"
        },
        "configuration": {
            "metrics": [
                {
                    "id": "m2",
                    "expression": "SELECT SUM(PartialDataTestMetric) FROM
 partial_data_test",
                    "returnData": true,
                    "period": 60
                }
            ]
        }
    }
}
```

경보에 대한 쿼리에 초기에 500여 개의 시계열을 반환하는 GROUP BY 문이 포함된 경우 쿼리가 찾은 처음 500개의 시계열을 기준으로 경보가 평가됩니다. 그러나 ORDER BY 절을 사용하면 쿼리가 찾은 모든 시계열이 정렬되고 ORDER BY 절에 따라 가장 높거나 가장 낮은 값을 가진 500개를 사용하여 경보가 평가됩니다.

지표 수식과 함께 Metrics Insights 쿼리 사용

Metrics Insights 쿼리를 지표 수학 함수에 대한 입력으로 사용할 수 있습니다. 지표 수학에 대한 자세한 정보는 지표 수학 사용 섹션을 참조하세요.

GROUP BY 절을 포함하지 않는 Metrics Insights 쿼리는 단일 시계열을 반환합니다. 따라서 반환된 결과는 단일 시계열을 입력값으로 사용하는 모든 지표 수학 함수와 함께 사용할 수 있습니다.

GROUP BY 절을 포함하는 Metrics Insights 쿼리는 여러 시계열을 반환합니다. 따라서 반환된 결과는 시계열 배열을 입력값으로 사용하는 모든 지표 수학 함수와 함께 사용할 수 있습니다.

예를 들어 다음 쿼리는 리전의 각 버킷에 대해 다운로드된 총 바이트 수를 시계열 배열로 반환합니다.

```
SELECT SUM(BytesDownloaded)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
```

콘솔의 그래프 또는 GetMetricData 작업에서 이 쿼리의 결과는 q1과 같습니다. 이 쿼리는 결과를 바이트 단위로 반환하므로 대신 MB로 표시하려는 경우 다음 수학 함수를 사용할 수 있습니다.

```
q1/1024/1024
```

자연어를 사용하여 CloudWatch Metrics Insights 쿼리를 생성하고 업데이트합니다.

이 기능은 CloudWatch용으로 미국 동부(버지니아 북부), 미국 서부(오레곤) 및 아시아 태평양(도쿄)에서 미리 출시되었으며 변경될 수 있습니다.

CloudWatch는 자연어 쿼리 기능을 지원하므로 CloudWatch Metrics Insights 및 CloudWatch Logs Insights에 대한 쿼리를 생성하고 업데이트할 수 있습니다.

이 기능을 사용하면 찾고 있는 CloudWatch 데이터에 대해 질문하거나 일반 영어로 설명할 수 있습니다. 자연어 기능은 사용자가 입력한 프롬프트에 따라 쿼리를 생성하고 쿼리 작동 방식을 한 줄씩 설명합니다. 쿼리를 업데이트하여 데이터를 더 자세히 조사할 수도 있습니다.

환경에 따라 '네트워크 출력이 가장 높은 Amazon Elastic Compute Cloud 인스턴스는 뭐야?' 및 '읽은 횟수별로 Amazon DynamoDB 테이블 상위 10개를 보여줘.'와 같은 프롬프트를 입력할 수 있습니다.

이 기능을 사용하여 CloudWatch Metrics Insights 쿼리를 생성하려면 작성기 또는 편집기 뷰에서 CloudWatch Metrics Insights 쿼리 편집기를 열고 쿼리 생성을 선택합니다.

> **Important**
>
> 자연어 쿼리 기능을 사용하려면 CloudWatchFullAccess, CloudWatchReadOnlyAccess, CloudWatchFullAccessV2, AdministratorAccess 또는 ReadOnlyAccess 정책을 사용해야 합니다.
> 신규 또는 기존 고객 관리형 또는 인라인 정책에 `cloudwatch:GenerateQuery` 작업을 포함시킬 수도 있습니다.

쿼리 예제

이 섹션의 예제에서는 자연어 기능을 사용하여 쿼리를 생성하고 업데이트하는 방법을 설명합니다.

> **Note**
>
> CloudWatch Metrics Insights 쿼리 편집기 및 구문에 대한 자세한 내용은 CloudWatch Metrics Insights 쿼리 구성 요소 및 구문을 참조하세요.

예제: 자연어 쿼리 생성

자연어를 사용하여 쿼리를 생성하려면 프롬프트를 입력하고 새 쿼리 생성을 선택합니다. 이 예제는 기본 검색을 수행하는 쿼리를 보여줍니다.

프롬프트

다음은 읽기 용량을 가장 많이 소비하는 상위 10개의 DynamoDB 테이블을 검색하도록 지시하는 프롬프트의 예입니다.

```
Show top 10 DynamoDB Tables by consumed reads
```

Query

다음은 프롬프트를 기반으로 자연 언어 기능이 생성하는 쿼리의 예입니다. 쿼리 전에 댓글에 프롬프트가 어떻게 나타나는지 확인합니다. 쿼리가 끝나면 쿼리 작동 방식을 설명하는 설명을 읽을 수 있습니다.

```
# Show top 10 DynamoDB Tables by consumed reads
SELECT SUM("ConsumedReadCapacityUnits")
FROM "AWS/DynamoDB"
GROUP BY TableName
ORDER BY SUM() DESC
LIMIT 10
# This query selects the sum of consumed read capacity units for each DynamoDB table,
 groups the results by table name, orders the results from highest to lowest read
 capacity consumption, and limits the results to the top 10 tables.
```

> **Note**
>
> 프롬프트 표시 및 쿼리 작동 방식에 대한 설명을 끄려면 편집기의 톱니바퀴 아이콘을 사용합니다.

예제: 자연어 쿼리 업데이트

초기 프롬프트를 편집한 다음, 쿼리 업데이트를 선택하여 쿼리를 업데이트할 수 있습니다.

업데이트된 프롬프트

다음 예제는 이전 프롬프트의 업데이트된 버전을 보여줍니다. 이 프롬프트는 읽기 용량을 가장 많이 소비하는 상위 10개의 DynamoDB 테이블을 검색하는 메시지 대신 이제 반환된 바이트 수를 기준으로 결과를 정렬하도록 지시합니다.

```
Sort by bytes returned instead
```

업데이트된 쿼리

다음은 업데이트된 쿼리의 예제입니다. 업데이트된 쿼리가 나오기 전에 댓글에 업데이트된 프롬프트가 어떻게 나타나는지 확인합니다. 쿼리가 끝나면 원래 쿼리가 어떻게 업데이트되었는지에 대한 설명을 읽을 수 있습니다.

```
# Sort by bytes returned instead
SELECT SUM("ReturnedBytes")
FROM "AWS/DynamoDB"
GROUP BY TableName
ORDER BY SUM() DESC
LIMIT 10
```

```
# This query modifies the original query to select the sum of returned bytes instead
 of consumed read capacity units, and orders the results from highest to lowest sum of
 returned bytes, limiting the results to the top 10 tables.
```

서비스 개선을 위한 데이터 사용 거부

AI 모델을 학습시키고 관련 쿼리를 생성하기 위해 제공하는 자연어 프롬프트 데이터는 서비스를 제공하고 유지하는 용도로만 사용됩니다. 이 데이터는 CloudWatch Metrics Insights의 품질을 개선하는 데 사용할 수 있습니다. 고객의 신뢰와 개인 정보 보호는 물론 콘텐츠 보안도 당사의 최우선 과제입니다. 자세한 내용은 AWS 서비스 약관과 AWS 책임감 있는 AI 정책을 참조하세요.

AI 서비스 옵트아웃 정책을 생성하여 콘텐츠가 자연어 쿼리의 개발 또는 품질 향상에 사용되는 것을 거부할 수 있습니다. 쿼리 생성 기능을 포함하여 모든 CloudWatch AI 기능에 대한 데이터 수집을 거부하려면 CloudWatch에 대한 옵트아웃 정책을 생성해야 합니다. 자세한 내용은 AWS Organizations 사용 설명서의 AI 서비스 옵트아웃 정책을 참조하세요.

SQL 추론

CloudWatch Metrics Insights는 여러 메커니즘을 사용하여 지정된 SQL 쿼리의 의도를 추론합니다.

주제

- 시간 버킷팅
- 필드 프로젝션
- ORDER BY를 사용한 글로벌 집계

시간 버킷팅

쿼리로 인한 시계열 데이터 포인트는 요청된 기간을 기준으로 시간 버킷으로 롤업됩니다. 표준 SQL에서 값을 집계하려면 지정된 기간의 모든 관측치를 함께 수집하도록 명시적으로 GROUP BY 절을 정의해야 합니다. 이것이 시계열 데이터를 쿼리하는 표준 방법이기 때문에 CloudWatch Metrics Insights는 명시적인 GROUP BY 절을 표현할 필요 없이 시간 버킷팅을 추론합니다.

예를 들어 쿼리가 1분 간격으로 수행되면 다음(제외)까지 해당 분에 속하는 모든 관측치가 시간 버킷의 시작 시간까지 롤업됩니다. 이렇게 하면 Metrics Insights SQL 문이 더욱 간결해집니다.

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

이전 쿼리는 모든 Amazon EC2 인스턴스의 평균 CPU 사용률을 나타내는 단일 시계열(타임스탬프 값 페어)을 반환합니다. 요청 기간이 1분이라고 가정하면 반환되는 각 데이터 포인트는 특정 1분 간격(시작 시간 포함, 종료 시간 제외) 내에서 측정된 모든 관측치의 평균을 나타냅니다. 특정 데이터 포인트와 관련된 타임스탬프는 버킷의 시작 시간입니다.

필드 프로젝션

Metrics Insights 쿼리는 항상 타임스탬프 프로젝션을 반환합니다. 각 해당 데이터 포인트 값의 타임 스탬프를 가져오기 위해 SELECT 절에서 타임스탬프 열을 지정할 필요가 없습니다. 타임스탬프가 계산되는 방법에 대한 자세한 내용은 시간 버킷팅 섹션을 참조하세요.

GROUP BY을 사용하는 경우 각 그룹 이름도 결과에 추론되고 프로젝션되므로 반환된 시계열을 그룹화할 수 있습니다.

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
```

이전 쿼리는 각 Amazon EC2 인스턴스에 대한 시계열을 반환합니다. 각 시계열은 인스턴스 ID 값 뒤에 레이블 지정됩니다.

ORDER BY를 사용한 글로벌 집계

ORDER BY를 사용하는 경우 FUNCTION()은 정렬할 집계 함수(쿼리된 지표의 데이터 포인트 값)를 추론합니다. 집계 작업은 쿼리된 시간 기간에서 각 시계열의 일치하는 모든 데이터 포인트에 대해 수행됩니다.

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
ORDER BY MAX()
LIMIT 10
```

이전 쿼리는 각 Amazon EC2 인스턴스에 대한 CPU 사용률을 반환하여 결과 집합을 10개 항목으로 제한합니다. 결과는 요청된 시간 기간 내에 있는 개별 시계열의 최대값을 기준으로 정렬됩니다. ORDER BY 절은 LIMIT 이전에 적용되므로 10개 이상의 시계열에 대해 순서가 계산됩니다.

Metrics Insights 샘플 쿼리

이 섹션에는 쿼리 편집기에서 직접 복사 및 사용하거나 복사 및 수정할 수 있는 유용한 CloudWatch Metrics Insights 쿼리의 예제가 포함되어 있습니다. 이러한 예제 중 일부는 콘솔에서 이미 사용할 수 있으며 지표(Metrics) 보기의 쿼리 추가(Add query)를 선택하여 액세스할 수 있습니다.

Application Load Balancer 예제

모든 로드 밸런서의 총 요청 수

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer)
```

상위 10개 활성 로드 밸런서

```
SELECT MAX(ActiveConnectionCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer)
GROUP BY LoadBalancer
ORDER BY SUM() DESC
LIMIT 10
```

AWS API 사용 예제

계정의 호출 수를 기준으로 한 상위 20개 AWS API

```
SELECT COUNT(CallCount)
FROM SCHEMA("AWS/Usage", Class, Resource, Service, Type)
WHERE Type = 'API'
GROUP BY Service, Resource
ORDER BY COUNT() DESC
LIMIT 20
```

호출별로 정렬된 CloudWatch API

```
SELECT COUNT(CallCount)
FROM SCHEMA("AWS/Usage", Class, Resource, Service, Type)
WHERE Type = 'API' AND Service = 'CloudWatch'
GROUP BY Resource
ORDER BY COUNT() DESC
```

DynamoDB 예제

사용된 읽기별 상위 10개 테이블

```
SELECT SUM(ProvisionedWriteCapacityUnits)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

반환된 바이트별 상위 10개 테이블

```
SELECT SUM(ReturnedBytes)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

사용자 오류별 상위 10개 테이블

```
SELECT SUM(UserErrors)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

Amazon Elastic Block Store 이벤트

기록된 바이트별 상위 10개 Amazon EBS 볼륨

```
SELECT SUM(VolumeWriteBytes)
FROM SCHEMA("AWS/EBS", VolumeId)
GROUP BY VolumeId
ORDER BY SUM() DESC
LIMIT 10
```

Amazon EBS 볼륨 쓰기 시간 평균

```
SELECT AVG(VolumeTotalWriteTime)
FROM SCHEMA("AWS/EBS", VolumeId)
```

Amazon EC2 예제

EC2 인스턴스의 CPU 사용률을 높은순으로 정렬

```
SELECT AVG(CPUUtilization)
  FROM SCHEMA("AWS/EC2", InstanceId)
  GROUP BY InstanceId
  ORDER BY AVG() DESC
```

전체 플릿의 평균 CPU 사용률

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

CPU 사용률이 높은 상위 10개 인스턴스

```
SELECT MAX(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
ORDER BY MAX() DESC
LIMIT 10
```

이 경우 CloudWatch 에이전트가 애플리케이션당 **CPUUtilization** 지표를 수집합니다. 이 쿼리는 특정 애플리케이션 이름에 대해 해당 지표의 평균을 필터링합니다.

```
SELECT AVG(CPUUtilization)
FROM "AWS/CWAgent"
WHERE ApplicationName = 'eCommerce'
```

Amazon Elastic Container Service 예제

모든 ECS 클러스터의 평균 CPU 사용률

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/ECS", ClusterName)
```

메모리 사용률별 상위 10개 클러스터

```
SELECT AVG(MemoryUtilization)
FROM SCHEMA("AWS/ECS", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC
LIMIT 10
```

CPU 사용률별 상위 10개 서비스

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/ECS", ClusterName, ServiceName)
GROUP BY ClusterName, ServiceName
ORDER BY AVG() DESC
LIMIT 10
```

태스크 실행별 상위 10개 서비스(Container Insights)

```
SELECT AVG(RunningTaskCount)
FROM SCHEMA("ECS/ContainerInsights", ClusterName, ServiceName)
GROUP BY ClusterName, ServiceName
ORDER BY AVG() DESC
LIMIT 10
```

Amazon Elastic Kubernetes Service Container Insights 예제

모든 EKS 클러스터의 평균 CPU 사용률

```
SELECT AVG(pod_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
```

노드 CPU 사용률별 상위 10개 클러스터

```
SELECT AVG(node_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC LIMIT 10
```

포드 메모리 사용률별 상위 10개 클러스터

```
SELECT AVG(pop_memory_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC LIMIT 10
```

CPU 사용률별 상위 10개 노드

```
SELECT AVG(node_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName, NodeName)
```

```
GROUP BY ClusterName, NodeName
ORDER BY AVG() DESC LIMIT 10
```

메모리 사용률별 상위 10개 포드

```
SELECT AVG(pod_memory_utilization)
FROM SCHEMA("ContainerInsights", ClusterName, PodName)
GROUP BY ClusterName, PodName
ORDER BY AVG() DESC LIMIT 10
```

EventBridge 예제

호출별 상위 10개 규칙

```
SELECT SUM(Invocations)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

실패한 호출별 상위 10개 규칙

```
SELECT SUM(FailedInvocations)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

일치하는 규칙별 상위 10개 규칙

```
SELECT SUM(MatchedEvents)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

Kinesis 예제

작성된 바이트별 상위 10개 스트림

```
SELECT SUM("PutRecords.Bytes")
FROM SCHEMA("AWS/Kinesis", StreamName)
GROUP BY StreamName
ORDER BY SUM() DESC LIMIT 10
```

스트림에서 가장 빠른 항목별 상위 10개 스트림

```
SELECT MAX("GetRecords.IteratorAgeMilliseconds")
FROM SCHEMA("AWS/Kinesis", StreamName)
GROUP BY StreamName
ORDER BY MAX() DESC LIMIT 10
```

Lambda 예제

호출 수를 기준으로 정렬된 Lambda 함수

```
SELECT SUM(Invocations)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY SUM() DESC
```

가장 긴 런타임별 상위 10가지 Lambda 함수

```
SELECT AVG(Duration)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY MAX() DESC
LIMIT 10
```

오류 개수에 따른 상위 10가지 Lambda 함수

```
SELECT SUM(Errors)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY SUM() DESC
LIMIT 10
```

CloudWatch Logs 예제

수신 이벤트별 상위 10개 로그 그룹

```
SELECT SUM(IncomingLogEvents)
FROM SCHEMA("AWS/Logs", LogGroupName)
GROUP BY LogGroupName
ORDER BY SUM() DESC LIMIT 10
```

기록된 바이트별 상위 10개 로그 그룹

```
SELECT SUM(IncomingBytes)
FROM SCHEMA("AWS/Logs", LogGroupName)
GROUP BY LogGroupName
ORDER BY SUM() DESC LIMIT 10
```

Amazon RDS 예제

CPU 사용률이 높은 상위 10개 Amazon RDS 인스턴스

```
SELECT MAX(CPUUtilization)
FROM SCHEMA("AWS/RDS", DBInstanceIdentifier)
GROUP BY DBInstanceIdentifier
ORDER BY MAX() DESC
LIMIT 10
```

쓰기별 상위 10개 Amazon RDS 클러스터

```
SELECT SUM(WriteIOPS)
FROM SCHEMA("AWS/RDS", DBClusterIdentifier)
GROUP BY DBClusterIdentifier
ORDER BY MAX() DESC
LIMIT 10
```

Amazon Simple Storage Service 예제

버킷별 평균 대기 시간

```
SELECT AVG(TotalRequestLatency)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
ORDER BY AVG() DESC
```

다운로드한 바이트별 상위 10개 버킷

```
SELECT SUM(BytesDownloaded)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
ORDER BY SUM() DESC
```

```
LIMIT 10
```

Amazon Simple Notification Service 예제

SNS 주제가 게시한 총 메시지

```
SELECT SUM(NumberOfMessagesPublished)
FROM SCHEMA("AWS/SNS", TopicName)
```

게시된 메시지별 상위 10가지 주제

```
SELECT SUM(NumberOfMessagesPublished)
FROM SCHEMA("AWS/SNS", TopicName)
GROUP BY TopicName
ORDER BY SUM() DESC
LIMIT 10
```

메시지 전달 실패별 상위 10가지 주제

```
SELECT SUM(NumberOfNotificationsFailed)
FROM SCHEMA("AWS/SNS", TopicName)
GROUP BY TopicName
ORDER BY SUM() DESC
LIMIT 10
```

Amazon SQS 예제

표시되는 메시지 수 기준 상위 10개 대기열

```
SELECT AVG(ApproximateNumberOfMessagesVisible)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY AVG() DESC
LIMIT 10
```

가장 활발한 상위 10개 대기열

```
SELECT SUM(NumberOfMessagesSent)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY SUM() DESC
```

```
LIMIT 10
```

가장 빠른 메시지 연령별 상위 10개 대기열

```
SELECT AVG(ApproximateAgeOfOldestMessage)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY AVG() DESC
LIMIT 10
```

Metrics Insights 제한

CloudWatch Metrics Insights에는 현재 다음과 같은 제한이 있습니다.

- 현재 가장 최근 3시간의 데이터만 쿼리할 수 있습니다.
- 단일 쿼리는 10,000개 이하의 지표를 처리할 수 있습니다. 즉, SELECT, FROM, WHERE 절이 10,000개 이상의 지표와 일치하면 쿼리는 찾은 해당 지표 중 처음 10,000개만 처리합니다.
- 단일 쿼리는 500개 이하의 시계열을 반환할 수 있습니다. 즉, 쿼리가 500개 이상의 지표를 반환하는 경우 일부 지표만 쿼리 결과에 반환됩니다. ORDER BY 절을 사용하면 처리되는 모든 지표가 정렬되고 ORDER BY 절에 따라 내림차순 또는 오름차순으로 500개가 반환됩니다.

  ORDER BY 절을 포함하지 않는 경우 반환되는 500개의 일치하는 지표을 제어할 수 없습니다.
- 리전당 최대 200개의 Metrics Insights 경보를 가질 수 있습니다.
- Metrics Insights는 1분 미만의 세부 수준으로 보고되는 지표 데이터인 고해상도 데이터를 지원하지 않습니다. 고해상도 데이터를 요청하는 경우 요청은 실패하지 않지만 출력은 1분 단위로 집계됩니다.
- 각 GetMetricData 작업에는 쿼리가 하나만 있을 수 있지만 대시보드에는 각 쿼리가 포함된 위젯이 여러 개 있을 수 있습니다.

Metrics Insights 용어집

레이블

Metrics Insights에서 레이블은 쿼리의 범위를 지정하여 특정 데이터 집합을 반환하거나 쿼리 결과를 별도의 시계열로 구분하는 기준을 정의하는 데 사용되는 키 값 페어입니다. 레이블 키는 SQL의 열 이름과 유사합니다. 현재 레이블은 CloudWatch 지표 측정기준이어야 합니다.

관찰

관측치는 지정된 시간에 지정된 지표에 대해 기록된 값입니다.

Metrics Insights 문제 해결

결과에는 "기타"가 포함되지만, 측정기준으로 지정하지 않습니다.

즉, 쿼리는 쿼리에 의해 반환되는 일부 지표에 사용되지 않는 레이블 키를 지정하는 GROUP BY 절을 포함합니다. 이 경우 이름이 `Other`인 null 그룹이 반환됩니다. 해당 레이블 키를 포함하지 않는 지표는 해당 레이블 키의 모든 값에 걸쳐 집계된 값을 반환하는 지표를 집계할 수 있습니다.

예를 들어 다음과 같은 쿼리가 있다고 가정합니다.

```
SELECT AVG(Faults)
FROM MyCustomNamespace
GROUP BY Operation, ServiceName
```

반환된 지표 일부에 측정기준인 `ServiceName`이 포함되지 않은 경우 해당 지표는 `ServiceName` 값으로 `Other`를 표시합니다.

결과에 "Other"가 표시되지 않도록 하려면 다음 예제와 같이 FROM 절에서 SCHEMA를 사용합니다.

```
SELECT AVG(Faults)
FROM SCHEMA(MyCustomNamespace, Operation)
GROUP BY Operation, ServiceName
```

이렇게 하면 반환된 결과가 `Operation` 및 `ServiceName` 측정기준 모두가 포함된 지표로만 제한됩니다.

그래프에서 가장 오래된 타임스탬프는 다른 지표보다 지표 값이 낮습니다.

CloudWatch Metrics Insights는 현재 최근 3시간의 데이터만 지원합니다. 1분보다 긴 기간으로 그래프를 만들 때 가장 오래된 데이터 포인트가 예상 값과 다른 경우가 있을 수 있습니다. 이는 Metrics Insights 쿼리가 가장 최근 3시간 분량의 데이터만 반환하기 때문입니다. 이 경우 쿼리에서 가장 오래된 데이터 포인트는 해당 데이터 포인트의 기간 내에 있는 모든 관측값을 반환하는 대신 지난 3시간 경계 내에서 측정된 관측값만 반환합니다.

지표 탐색기를 사용하여 태그 및 속성별 리소스 모니터링

지표 탐색기는 태그 및 리소스 속성별로 지표를 필터링하고 집계하며 시각화하여 서비스에 대한 관찰 가능성을 향상할 수 있도록 지원하는 태그 기반 도구입니다. 이 도구를 사용하면 유연하고 동적인 문제 해결 환경을 통해 한 번에 여러 그래프를 생성할 수 있으며 이러한 그래프를 사용하여 애플리케이션 상태 대시보드를 구축할 수 있습니다.

지표 탐색기 시각화는 동적입니다. 따라서 지표 탐색기 위젯을 생성하여 CloudWatch 대시보드에 추가한 후에 일치하는 리소스를 생성하면 새 리소스가 탐색기 위젯에 자동으로 나타납니다.

예를 들어 모든 EC2 프로덕션 인스턴스에 **`production`** 태그가 있는 경우 지표 탐색기를 사용함으로써 이러한 모든 인스턴스의 지표를 필터링하고 집계하여 상태 및 성능을 파악할 수 있습니다. 일치하는 태그가 있는 새 인스턴스를 나중에 생성하면 해당 인스턴스가 지표 탐색기 위젯에 자동으로 추가됩니다.

> **Note**
>
> 지표 탐색기는 특정 시점에 대한 경험을 제공합니다. 종료되었거나 지정한 속성 또는 태그가 더 이상 존재하지 않는 리소스는 시각화에 표시되지 않습니다. 그러나 이러한 리소스에 대한 지표는 CloudWatch 지표 보기에서 계속 찾을 수 있습니다.

지표 탐색기를 사용하면 기준과 일치하는 리소스에서 지표를 집계하는 방법과 해당 지표를 단일 그래프에 모두 표시할지 아니면 하나의 지표 탐색기 위젯 내 다른 그래프에 표시할지 여부를 선택할 수 있습니다.

지표 탐색기에는 클릭 한 번으로 유용한 시각화 그래프를 보는 데 사용할 수 있는 템플릿이 포함되어 있습니다. 이러한 템플릿을 확장하여 완전히 사용자 지정한 지표 탐색기 위젯을 생성할 수도 있습니다.

지표 탐색기는 메모리, 디스크 및 CPU 지표를 포함하여 CloudWatch 에이전트가 게시한 EC2 지표와 AWS에서 내보낸 지표를 지원합니다. 지표 탐색기를 사용하여 CloudWatch 에이전트가 게시한 지표를 확인하려면 CloudWatch 에이전트 구성 파일을 업데이트해야 할 수 있습니다. 자세한 내용은 지표 탐색기에 대한 CloudWatch 에이전트 구성 단원을 참조하세요.

지표 탐색기를 사용하여 시각화를 생성하고 필요에 따라 대시보드에 추가하려면 다음 단계를 따릅니다.

지표 탐색기를 사용하여 시각화를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 탐색 창에서 [탐색기(Explorer)]를 선택합니다.
3. 다음 중 하나를 수행하세요.
   - 템플릿을 사용하려면 현재 [빈 탐색기(Empty Explorer)]가 표시된 상자에서 템플릿을 선택합니다.

     템플릿에 따라 탐색기에 지표 그래프가 즉시 표시될 수 있습니다. 즉시 표시되지 않은 경우 [다음에서(From)] 상자에서 태그 또는 속성을 하나 이상 선택하면 데이터가 나타납니다. 데이터가 나타나지 않은 경우 페이지 상단의 옵션을 사용하여 그래프에 더 긴 시간 범위가 표시되도록 합니다.
   - 사용자 지정 시각화를 생성하려면 [지표(Metrics)]에서 단일 지표 또는 서비스에서 사용 가능한 모든 지표를 선택합니다.

     지표를 선택한 후 필요에 따라 이 단계를 반복하여 더 많은 지표를 추가할 수 있습니다.
4. 선택한 각 지표에 대해 CloudWatch는 지표 이름 바로 뒤에 사용할 통계를 표시합니다. 이를 변경하려면 통계 이름을 선택한 다음, 원하는 통계를 선택합니다.
5. [다음에서(From)]에서 결과를 필터링할 태그 또는 리소스 속성을 선택합니다.

   이 작업을 수행한 후 필요에 따라 이 단계를 반복하여 더 많은 태그 또는 리소스 속성을 선택할 수 있습니다.

   EC2 인스턴스 유형 두 개와 같이 속성이 동일한 값을 여러 개 선택하는 경우 탐색기에는 선택한 속성 중 하나와 일치하는 리소스가 모두 표시됩니다. 이는 OR 연산으로 처리됩니다.

   `Production` 태그 및 M5 인스턴스 유형과 같이 서로 다른 속성 또는 태그를 선택하는 경우 이러한 모든 선택 사항과 일치하는 리소스만 표시됩니다. 이는 AND 연산으로 처리됩니다.
6. (선택 사항) [집계 기준(Aggregate by)]에서 지표 집계에 사용할 통계를 선택합니다. 그런 다음, 대상(for) 옆에 있는 목록에서 지표를 집계하는 방법을 선택합니다. 현재 표시된 모든 리소스를 함께 집계하거나 단일 태그 또는 리소스 속성을 기준으로 집계할 수 있습니다.

   선택하는 집계 방법에 따라 결과가 단일 시계열 또는 다중 시계열이 될 수 있습니다.
7. 분할 기준(Split by)에서 다중 시계열의 단일 그래프를 여러 그래프로 분할하도록 선택할 수 있습니다. 분할 기준(Split by)에서 선택하는 다양한 기준에 따라 분할을 수행할 수 있습니다.
8. [그래프 옵션(Graph options)]에서 기간, 그래프 유형, 범례 배치, 레이아웃을 변경하여 그래프를 구체화할 수 있습니다.
9. 이 시각화를 CloudWatch 대시보드에 위젯으로 추가하려면 [대시보드에 추가(Add to dashboard)]를 선택합니다.

지표 탐색기에 대한 CloudWatch 에이전트 구성

CloudWatch 에이전트가 게시한 EC2 지표를 지표 탐색기가 검색할 수 있도록 하려면 CloudWatch 에이전트 구성 파일에 다음 값이 포함되어 있는지 확인합니다.

- `metrics` 섹션에서 `aggregation_dimensions` 파라미터에 `["InstanceId"]`가 포함되어 있는지 확인합니다. 이 파라미터에는 다른 측정기준도 포함될 수 있습니다.
- `metrics` 섹션에서 `append_dimensions` 파라미터에 `{"InstanceId":"${aws:InstanceId}"}` 줄이 포함되어 있는지 확인합니다. 이 파라미터에는 다른 줄도 포함될 수 있습니다.
- `metrics` 섹션의 `metrics_collected` 섹션 내에서 지표 탐색기가 검색할 각 리소스 유형에 대한 섹션(예: `cpu`, `disk` 및 `memory` 섹션)을 확인합니다. 이러한 각 섹션에 `"resources": [ "*"]` `line.`이 있는지 확인합니다.
- `metrics_collected` 섹션의 `cpu` 섹션에서 `"totalcpu": true` 줄이 있는지 확인합니다.
- 사용자 지정 네임스페이스 대신 CloudWatch 에이전트가 수집한 지표의 기본 `CWAgent` 네임스페이스를 사용해야 합니다.

위 목록의 설정으로 인해 CloudWatch 에이전트는 디스크, CPU 및 이를 사용하는 모든 인스턴스에 대해 지표 탐색기에 표시될 수 있는 기타 리소스의 집계 지표를 게시합니다.

이러한 설정은 이전에 여러 측정기준과 함께 게시되도록 설정한 지표를 다시 게시하며 이는 지표 비용에 추가됩니다.

CloudWatch 에이전트 구성 파일 편집에 대한 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하세요.

지표 스트림 사용

'지표 스트림'을 사용하여 원하는 대상에 CloudWatch 지표를 지속적으로 스트리밍할 수 있으며, 짧은 대기 시간으로 실시간에 가깝게 전달합니다. 지원되는 대상에는 Amazon Simple Storage Service와 같은 AWS 대상 및 여러 서드 파티 서비스 공급자 대상이 포함됩니다.

CloudWatch 지표 스트림의 세 가지 주요 사용 시나리오는 다음과 같습니다.

- Firehose 사용자 지정 설정 - 지표 스트림을 생성하고, CloudWatch 지표를 원하는 위치로 전달하는 Amazon Data Firehose 전송 스트림에 보냅니다. Amazon S3와 같은 데이터 레이크로 스트리밍하거나 타사 공급자를 포함하여 Firehose에서 지원하는 모든 대상 또는 엔드포인트로 스트리밍할

수 있습니다. JSON, OpenTelemetry 1.0.0 및 OpenTelemetry 0.7.0 형식이 기본적으로 지원되거나 Firehose 전송 스트림에서 변환을 구성하여 데이터를 Parquet과 같은 다른 형식으로 변환할 수 있습니다. 지표 스트림을 사용하여 모니터링 데이터를 지속적으로 업데이트하거나 이 CloudWatch 지표 데이터를 청구 및 성능 데이터와 결합하여 풍부한 데이터 집합을 생성할 수 있습니다. 그런 다음, Amazon Athena와 같은 도구를 사용하여 비용 최적화, 리소스 성능, 리소스 사용률에 대한 인사이트를 얻을 수 있습니다.

- 빠른 S3 설정 - 빠른 설정 프로세스를 통해 Amazon Simple Storage Service로 스트리밍할 수 있습니다. 기본적으로 CloudWatch는 스트림에 필요한 리소스를 생성합니다. JSON, OpenTelemetry 1.0.0 및 OpenTelemetry 0.7.0 형식이 지원됩니다.
- 빠른 AWS 파트너 설정 — CloudWatch는 일부 타사 파트너에게 빠른 설정 환경을 제공합니다. 타사 서비스 공급자를 통해 스트리밍된 CloudWatch 데이터를 활용하여 애플리케이션을 모니터링하고 문제를 해결하며 분석할 수 있습니다. 빠른 파트너 설정 워크플로를 사용하는 경우 목적지의 대상 URL과 API 키만 제공하면 CloudWatch가 나머지 설정을 처리합니다. 다음 타사 공급자가 빠른 파트너 설정을 사용할 수 있습니다.

  - Datadog
  - Dynatrace
  - New Relic
  - Splunk Observability Cloud
  - SumoLogic

모든 CloudWatch 지표를 스트리밍하거나 필터를 사용하여 지정된 지표만 스트리밍할 수 있습니다. 각 지표 스트림은 지표 네임스페이스를 또는 특정 지표를 포함하거나 제외하는 필터를 최대 1,000개까지 포함할 수 있습니다. 단일 지표 스트림에는 포함 또는 제외 필터만 있을 수 있으며 두 필터가 모두 있을 수는 없습니다.

지표 스트림을 생성하면 필터와 일치하는 새 지표가 생성되어 준비된 경우 새 지표가 스트림에 자동으로 포함됩니다.

계정당 또는 리전당 지표 스트림 수에는 제한이 없으며, 스트리밍되는 지표 업데이트 수에도 제한이 없습니다.

각 스트림은 JSON 형식, OpenTelemetry 1.0.0 또는 OpenTelemetry 0.7.0 형식을 사용할 수 있습니다. OpenTelemetry 0.7.0에서 OpenTelemetry 1.0.0으로 업그레이드하는 경우와 같이 언제든지 지표 스트림의 출력 형식을 편집할 수 있습니다. 출력 형식에 대한 자세한 내용은 지표 스트림 출력 형식 섹션을 참조하세요.

모니터링 계정의 지표 스트림의 경우 해당 모니터링 계정에 연결된 소스 계정의 지표를 포함할지 여부를 선택할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

지표 스트림은 항상 `Minimum`, `Maximum`, `SampleCount` 및 `Sum` 통계를 포함합니다. 또한 추가 비용으로 추가 통계를 포함하도록 선택할 수도 있습니다. 자세한 내용은 스트림 가능한 통계 단원을 참조하십시오.

지표 스트림 요금은 지표 업데이트 수를 기반으로 합니다. 또한 지표 스트림에 사용되는 전송 스트림에 대해 Firehose 요금이 발생합니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

주제

지표 스트림 설정

다음 단원의 단계에 따라 CloudWatch 지표 스트림을 설정할 수 있습니다.

지표 스트림을 생성한 후 지표 데이터가 대상에 나타나는 데 걸리는 시간은 Firehose 전송 스트림에 구성된 버퍼링 설정에 따라 다릅니다. 버퍼링은 최대 페이로드 크기 또는 최대 대기 시간 중 먼저 도달하는 것으로 표시됩니다. 버퍼링을 최솟값(60초, 1MB)으로 설정했다면 선택한 CloudWatch 네임스페이스에 활성 지표 업데이트가 있는 경우 예상 대기 시간은 3분 이내입니다.

CloudWatch 지표 스트림에서 데이터는 1분마다 전송됩니다. 데이터는 순서와 상관없이 최종 대상에 도착할 수 있습니다. 지정된 네임스페이스의 모든 지정된 지표는 2일보다 오래된 타임스탬프가 있는 지표를 제외하고 지표 스트림으로 전송됩니다.

스트리밍하는 지표 이름과 네임스페이스의 각 조합에 대해 해당 지표 이름과 네임스페이스의 모든 차원 조합이 스트리밍됩니다.

모니터링 계정의 지표 스트림의 경우 해당 모니터링 계정에 연결된 소스 계정의 지표를 포함할지 여부를 선택할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

지표 스트림을 생성하고 관리하려면 CloudWatchFullAccess 정책 및 `iam:PassRole` 권한이 있는 계정 또는 다음 권한 목록이 있는 계정에 로그온해야 합니다.

- `iam:PassRole`
- `cloudwatch:PutMetricStream`
- `cloudwatch:DeleteMetricStream`
- `cloudwatch:GetMetricStream`
- `cloudwatch:ListMetricStreams`
- `cloudwatch:StartMetricStreams`
- `cloudwatch:StopMetricStreams`

CloudWatch에 대해 지표 스트림에 필요한 IAM 역할을 설정하려면 `iam:CreateRole` 및 `iam:PutRolePolicy` 권한도 있어야 합니다.

> **Important**
>
> `cloudwatch:PutMetricStream`이 있는 사용자는 `cloudwatch:GetMetricData` 권한이 없더라도 스트리밍 중인 CloudWatch 지표 데이터에 액세스할 수 있습니다.

주제

- Firehose 사용자 지정 설정
- 빠른 Amazon S3 설정 사용
- 빠른 파트너 설정

Firehose 사용자 지정 설정

이 방법을 사용하여 지표 스트림을 생성하고, CloudWatch 지표를 원하는 위치로 전달하는 Amazon Data Firehose 전송 스트림에 보냅니다. Amazon S3와 같은 데이터 레이크로 스트리밍하거나 타사 공급자를 포함하여 Firehose에서 지원하는 모든 대상 또는 엔드포인트로 스트리밍할 수 있습니다.

JSON, OpenTelemetry 1.0.0 및 OpenTelemetry 0.7.0 형식이 기본적으로 지원되거나 Firehose 전송 스트림에서 변환을 구성하여 데이터를 Parquet과 같은 다른 형식으로 변환할 수 있습니다. 지표 스트림을 사용하여 모니터링 데이터를 지속적으로 업데이트하거나 이 CloudWatch 지표 데이터를 청구 및

성능 데이터와 결합하여 풍부한 데이터 집합을 생성할 수 있습니다. 그런 다음, Amazon Athena와 같은 도구를 사용하여 비용 최적화, 리소스 성능, 리소스 사용률에 대한 인사이트를 얻을 수 있습니다.

CloudWatch 콘솔, AWS CLI, AWS CloudFormation 또는 AWS Cloud Development Kit (AWS CDK)를 사용하여 지표 스트림을 설정할 수 있습니다.

지표 스트림에 사용하는 Firehose 전송 스트림은 지표 스트림을 설정한 동일한 계정 및 동일한 리전에 있어야 합니다. 교차 리전 기능을 구현하려면 다른 계정이나 다른 리전에 있는 최종 대상에 스트리밍하도록 Firehose 전송 스트림을 구성하면 됩니다.

CloudWatch 콘솔

이 섹션에서는 CloudWatch 콘솔을 사용하여 Firehose를 사용한 지표 스트림을 설정하는 방법을 설명합니다.

Firehose를 사용한 사용자 지정 지표 스트림 설정 방법

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 스트리밍을 선택합니다. 지표 스트림 생성을 선택합니다.
3. (선택 사항) CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정에 로그인한 경우 연결된 소스 계정의 지표를 이 지표 스트림에 포함할지 선택할 수 있습니다. 소스 계정의 지표를 포함하려면 Include source account metrics(소스 계정 지표 포함)를 선택합니다.
4. Firehose를 사용한 사용자 지정 설정을 선택합니다.
5. Kinesis Data Firehose 스트림 선택에서 사용할 Firehose 전송 스트림을 선택합니다. 이 스트림은 동일한 계정에 있어야 합니다. 이 옵션의 기본 형식은 OpenTelemetry 0.7.0이지만, 이 절차의 뒷부분에서 형식을 변경할 수 있습니다.

   그런 다음 Firehose 전송 스트림 선택에서 사용할 Firehose 전송 스트림을 선택합니다.
6. (선택 사항)기존 서비스 역할 선택을 선택하면 CloudWatch가 자동으로 새 역할을 생성하도록 하는 대신 기존 IAM 역할을 사용할 수 있습니다.
7. (선택 사항) 시나리오의 기본 형식에서 출력 형식을 변경하려면 [출력 형식 변경(Change output format)]을 선택합니다. 지원되는 형식은 JSON, OpenTelemetry 1.0.0 및 OpenTelemetry 0.7.0입니다.
8. 스트리밍할 지표에서 모든 지표 또는 지표 선택을 선택합니다.

   모든 지표를 선택하는 경우, 이 계정의 모든 지표가 스트림에 포함됩니다.

   스트리밍하는 지표가 많을수록 지표 스트림 요금이 올라가므로 모든 지표를 스트리밍할지 여부를 신중하게 고려해야 합니다.

지표 선택을 선택하는 경우, 다음 중 하나를 수행합니다.

- 대부분의 지표 네임스페이스를 스트리밍하려면 제외를 선택하고 제외할 네임스페이스나 지표를 선택합니다. 제외에 네임스페이스를 지정하는 경우 해당 네임스페이스에서 제외할 특정 지표를 선택적으로 지정할 수 있습니다. 네임스페이스를 제외하도록 선택했지만 해당 네임스페이스에서 지표를 선택하지 않으면 해당 네임스페이스의 모든 지표가 제외됩니다.
- 지표 스트림에 몇 개의 지표 네임스페이스 또는 지표만 포함하려면 포함을 선택한 다음, 포함할 네임스페이스 또는 지표를 선택합니다. 네임스페이스를 포함하도록 선택했지만 해당 네임스페이스에서 지표를 선택하지 않으면 해당 네임스페이스의 모든 지표가 포함됩니다.

9. (선택 사항) 최소, 최대값, 샘플 수, 합계 이외의 일부 지표에 대한 추가 통계를 스트리밍하려면 통계 추가를 선택합니다. 권장 지표 추가를 선택하여 자주 사용되는 통계를 추가하거나, 추가 통계를 스트리밍할 네임스페이스 및 지표 이름을 수동으로 선택합니다. 다음으로 스트리밍할 추가 통계를 선택합니다.

   그런 다음 다른 추가 통계 집합을 스트리밍할 다른 지표 그룹을 선택하려면 통계 추가를 선택합니다. 각 지표에는 최대 20개의 추가 통계가 포함될 수 있으며 지표 스트림 내에 100개까지 추가 통계가 포함될 수 있습니다.

   추가 통계를 스트리밍하면 더 많은 비용이 발생합니다. 자세한 내용은 스트림 가능한 통계 단원을 참조하십시오.

   추가 통계에 대한 정의는 CloudWatch 통계 정의 단원을 참조하세요.

10. (선택 사항) [지표 스트림 이름(Metric stream name)]에서 새 지표 스트림 이름을 사용자 지정합니다.
11. [지표 스트림 생성(Create metric stream)]을 선택합니다.

AWS CLI 또는 AWS API

다음 단계에 따라 CloudWatch 지표 스트림을 생성할 수 있습니다.

AWS CLI 또는 AWS API를 사용하여 지표 스트림을 생성하려면

1. Amazon S3에 스트리밍하는 경우 먼저, 버킷을 생성합니다. 자세한 내용은 버킷 생성 단원을 참조하세요.
2. Firehose 전송 스트림을 생성합니다. 자세한 내용은 Firehose 스트림 생성을 참조하세요.
3. CloudWatch가 Firehose 전송 스트림에 쓸 수 있게 하는 IAM 역할을 생성합니다. 이 역할의 내용에 대해 자세히 알아보려면 CloudWatch와 Firehose 간의 신뢰 단원을 참조하세요.

4. `aws cloudwatch put-metric-stream` CLI 명령 또는 `PutMetricStream` API를 사용하여 CloudWatch 지표 스트림을 생성합니다.

AWS CloudFormation

AWS CloudFormation을 사용하여 지표 스트림을 설정할 수 있습니다. 자세한 내용은 AWS::CloudWatch::MetricStream 단원을 참조하세요.

AWS CloudFormation을 사용하여 지표 스트림을 생성하려면

1. Amazon S3에 스트리밍하는 경우 먼저, 버킷을 생성합니다. 자세한 내용은 버킷 생성 단원을 참조하세요.
2. Firehose 전송 스트림을 생성합니다. 자세한 내용은 Firehose 스트림 생성을 참조하세요.
3. CloudWatch가 Firehose 전송 스트림에 쓸 수 있게 하는 IAM 역할을 생성합니다. 이 역할의 내용에 대해 자세히 알아보려면 CloudWatch와 Firehose 간의 신뢰 단원을 참조하세요.
4. AWS CloudFormation에서 스트림을 생성합니다. 자세한 내용은 AWS::CloudWatch::MetricStream 단원을 참조하세요.

AWS Cloud Development Kit (AWS CDK)

AWS Cloud Development Kit (AWS CDK)를 사용하여 지표 스트림을 설정할 수 있습니다.

AWS CDK를 사용하여 지표 스트림을 생성하려면

1. Amazon S3에 스트리밍하는 경우 먼저, 버킷을 생성합니다. 자세한 내용은 버킷 생성 단원을 참조하세요.
2. Firehose 전송 스트림을 생성합니다. 자세한 내용은 Creating an Amazon Data Firehose Delivery Stream 섹션을 참조하세요.
3. CloudWatch가 Firehose 전송 스트림에 쓸 수 있게 하는 IAM 역할을 생성합니다. 이 역할의 내용에 대해 자세히 알아보려면 CloudWatch와 Firehose 간의 신뢰 단원을 참조하세요.
4. 지표 스트림을 생성합니다. 지표 스트림 리소스는 AWS CDK에서 `CfnMetricStream`이라는 레벨 1(L1) 구문으로 사용할 수 있습니다. 자세한 내용은 L1 구문 사용을 참조하세요.

빠른 Amazon S3 설정 사용

빠른 S3 설정 방법은 Amazon S3로의 스트림을 빠르게 설정하길 원하는 경우에 유용하며, 지원되는 JSON, OpenTelemetry 1.0.0 및 OpenTelemetry 0.7.0 형식 이외의 형식 변환이 필요하지 않습니다.

CloudWatch가 Firehose 전송 스트림 및 필요한 IAM 역할을 비롯하여 필요한 모든 리소스를 생성합니다. 이 옵션의 기본 형식은 JSON이지만 스트림을 설정하는 동안 형식을 변경할 수 있습니다.

또는 최종 형식이 Parquet 형식 또는 ORC(Optimized Row Columnar)가 되길 원하는 경우 Firehose 사용자 지정 설정의 단계를 수행해야 합니다.

CloudWatch 콘솔

이 섹션에서는 CloudWatch 콘솔을 사용하여 빠른 S3 설정을 사용한 지표 스트림 Amazon S3을 설정하는 방법을 설명합니다.

Quick S3 설정을 사용하여 지표 스트림 설정

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 스트리밍을 선택합니다. 지표 스트림 생성을 선택합니다.
3. (선택 사항) CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정에 로그인한 경우 연결된 소스 계정의 지표를 이 지표 스트림에 포함할지 선택할 수 있습니다. 소스 계정의 지표를 포함하려면 Include source account metrics(소스 계정 지표 포함)를 선택합니다.
4. 빠른 S3 설정을 선택합니다. CloudWatch가 Firehose 전송 스트림 및 필요한 IAM 역할을 비롯하여 필요한 모든 리소스를 생성합니다. 이 옵션의 기본 형식은 JSON이지만, 이 절차의 뒷부분에서 형식을 변경할 수 있습니다.
5. (선택 사항) 기존 리소스 선택을 선택하면 CloudWatch가 새 리소스를 생성하도록 하는 대신 기존 S3 버킷 또는 기존 IAM 역할을 사용할 수 있습니다.
6. (선택 사항) 시나리오의 기본 형식에서 출력 형식을 변경하려면 [출력 형식 변경(Change output format)]을 선택합니다. 지원되는 형식은 JSON, OpenTelemetry 1.0.0 및 OpenTelemetry 0.7.0입니다.
7. 스트리밍할 지표에서 모든 지표 또는 지표 선택을 선택합니다.

   모든 지표를 선택하는 경우, 이 계정의 모든 지표가 스트림에 포함됩니다.

   스트리밍하는 지표가 많을수록 지표 스트림 요금이 올라가므로 모든 지표를 스트리밍할지 여부를 신중하게 고려해야 합니다.

   지표 선택을 선택하는 경우, 다음 중 하나를 수행합니다.

   - 대부분의 지표 네임스페이스를 스트리밍하려면 제외를 선택하고 제외할 네임스페이스나 지표를 선택합니다. 제외에 네임스페이스를 지정하는 경우 해당 네임스페이스에서 제외할 특정 지표를 선택적으로 지정할 수 있습니다. 네임스페이스를 제외하도록 선택했지만 해당 네임스페이스에서 지표를 선택하지 않으면 해당 네임스페이스의 모든 지표가 제외됩니다.

- 지표 스트림에 몇 개의 지표 네임스페이스 또는 지표만 포함하려면 포함을 선택한 다음, 포함할 네임스페이스 또는 지표를 선택합니다. 네임스페이스를 포함하도록 선택했지만 해당 네임스페이스에서 지표를 선택하지 않으면 해당 네임스페이스의 모든 지표가 포함됩니다.

8. (선택 사항) 최소, 최대값, 샘플 수, 합계 이외의 일부 지표에 대한 추가 통계를 스트리밍하려면 통계 추가를 선택합니다. 권장 지표 추가를 선택하여 자주 사용되는 통계를 추가하거나, 추가 통계를 스트리밍할 네임스페이스 및 지표 이름을 수동으로 선택합니다. 다음으로 스트리밍할 추가 통계를 선택합니다.

   그런 다음 다른 추가 통계 집합을 스트리밍할 다른 지표 그룹을 선택하려면 통계 추가를 선택합니다. 각 지표에는 최대 20개의 추가 통계가 포함될 수 있으며 지표 스트림 내에 100개까지 추가 통계가 포함될 수 있습니다.

   추가 통계를 스트리밍하면 더 많은 비용이 발생합니다. 자세한 내용은 스트림 가능한 통계 단원을 참조하십시오.

   추가 통계에 대한 정의는 CloudWatch 통계 정의 단원을 참조하세요.
9. (선택 사항) [지표 스트림 이름(Metric stream name)]에서 새 지표 스트림 이름을 사용자 지정합니다.
10. 지표 스트림 생성을 선택합니다.

빠른 파트너 설정

CloudWatch는 다음과 같은 타사 파트너에게 빠른 설정 환경을 제공합니다. 이 워크플로를 사용하려면 목적지의 대상 URL과 API 키만 제공하면 됩니다. CloudWatch가 Firehose 전송 스트림 및 필요한 IAM 역할의 생성을 비롯하여 설정에 필요한 나머지를 처리합니다.

> **Important**
>
> 빠른 파트너 설정을 사용하여 지표 스트림을 생성하기 전에 다음 목록에 링크된 해당 파트너의 설명서를 읽어 보는 것이 좋습니다.

- Datadog
- Dynatrace
- New Relic
- Splunk Observability Cloud

- SumoLogic

이러한 파트너 중 하나로 지표 스트림을 설정하면 스트림이 다음 섹션에 나열된 몇 가지 기본 설정으로 생성됩니다.

주제

- 빠른 파트너 설정을 사용하여 지표 스트림 설정
- Datadog 스트림 기본값
- Dynatrace 스트림 기본값
- New Relic 스트림 기본값
- Splunk Observability Cloud 스트림 기본값
- Sumo Logic 스트림 기본값

빠른 파트너 설정을 사용하여 지표 스트림 설정

CloudWatch는 일부 타사 파트너를 위한 빠른 설정 옵션을 제공합니다. 이 섹션의 단계를 시작하기 전에 먼저 파트너에 대한 특정 정보가 필요합니다. 이 정보는 대상 URL 및/또는 파트너 대상의 API 키가 포함될 수 있습니다. 또한 이전 섹션에 링크된 파트너 웹사이트의 설명서와 다음 섹션에 나열된 해당 파트너의 기본값을 읽어야 합니다.

빠른 설정이 지원되지 않는 타사를 대상으로 스트리밍하려면 Firehose 사용자 지정 설정의 지침에 따라 Firehose를 사용하여 스트림을 설정한 다음 지표를 Firehose에서 최종 대상으로 전송하면 됩니다.

빠른 파트너 설정을 사용하여 타사 공급자를 대상으로 지표 스트림 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 스트리밍을 선택합니다. 지표 스트림 생성을 선택합니다.
3. (선택 사항) CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정에 로그인한 경우 연결된 소스 계정의 지표를 이 지표 스트림에 포함할지 선택할 수 있습니다. 소스 계정의 지표를 포함하려면 Include source account metrics(소스 계정 지표 포함)를 선택합니다.
4. 빠른 Amazon Web Services 파트너 설정을 선택합니다.
5. 지표를 스트리밍할 대상 파트너의 이름을 선택합니다.
6. 엔드포인트 URL에 대상 URL을 입력합니다.
7. 액세스 키 또는 API 키에 파트너의 액세스 키를 입력합니다. 액세스 키가 필요하지 않은 파트너도 있습니다.

8. 스트리밍할 지표에서 모든 지표 또는 지표 선택을 선택합니다.

   모든 지표를 선택하는 경우, 이 계정의 모든 지표가 스트림에 포함됩니다.

   스트리밍하는 지표가 많을수록 지표 스트림 요금이 올라가므로 모든 지표를 스트리밍할지 여부를 신중하게 고려해야 합니다.

   지표 선택을 선택하는 경우, 다음 중 하나를 수행합니다.

   - 대부분의 지표 네임스페이스를 스트리밍하려면 제외를 선택하고 제외할 네임스페이스나 지표를 선택합니다. 제외에 네임스페이스를 지정하는 경우 해당 네임스페이스에서 제외할 특정 지표를 선택적으로 지정할 수 있습니다. 네임스페이스를 제외하도록 선택했지만 해당 네임스페이스에서 지표를 선택하지 않으면 해당 네임스페이스의 모든 지표가 제외됩니다.
   - 지표 스트림에 몇 개의 지표 네임스페이스 또는 지표만 포함하려면 포함을 선택한 다음, 포함할 네임스페이스 또는 지표를 선택합니다. 네임스페이스를 포함하도록 선택했지만 해당 네임스페이스에서 지표를 선택하지 않으면 해당 네임스페이스의 모든 지표가 포함됩니다.

9. (선택 사항) 최소, 최대값, 샘플 수, 합계 이외의 일부 지표에 대한 추가 통계를 스트리밍하려면 통계 추가를 선택합니다. 권장 지표 추가를 선택하여 자주 사용되는 통계를 추가하거나, 추가 통계를 스트리밍할 네임스페이스 및 지표 이름을 수동으로 선택합니다. 다음으로 스트리밍할 추가 통계를 선택합니다.

   그런 다음 다른 추가 통계 집합을 스트리밍할 다른 지표 그룹을 선택하려면 통계 추가를 선택합니다. 각 지표에는 최대 20개의 추가 통계가 포함될 수 있으며 지표 스트림 내에 100개까지 추가 통계가 포함될 수 있습니다.

   추가 통계를 스트리밍하면 더 많은 비용이 발생합니다. 자세한 내용은 스트림 가능한 통계 단원을 참조하십시오.

   추가 통계에 대한 정의는 CloudWatch 통계 정의 단원을 참조하세요.

10. (선택 사항) [지표 스트림 이름(Metric stream name)]에서 새 지표 스트림 이름을 사용자 지정합니다.
11. 지표 스트림 생성을 선택합니다.

Datadog 스트림 기본값

Datadog에 대한 빠른 파트너 설정 스트림은 다음 기본값을 사용합니다.

- 출력 형식: OpenTelemetry 0.7.0
- Firehose 스트림 콘텐츠 인코딩 GZIP

- Firehose 스트림 버퍼링 옵션 간격 60초, 크기 4MB
- Firehose 스트림 재시도 옵션 소요 시간 60초

빠른 파트너 설정을 사용하여 Datadog에 대한 지표 스트림을 생성하고 특정 지표를 스트리밍하는 경우, 기본적으로 해당 지표에는 몇 가지 추가 통계가 포함됩니다. 추가 통계를 스트리밍하면 추가 요금이 부과될 수 있습니다. 통계와 요금에 대한 자세한 내용은 스트림 가능한 통계 섹션을 참조하세요.

다음 목록은 추가 통계가 기본적으로 스트림되는 지표를 보여줍니다(해당 지표를 스트림하도록 선택한 경우). 스트림을 시작하기 전에 추가 통계를 선택 취소할 수 있습니다.

- **Duration**(**AWS/Lambda**): p50, p80, p95, p99, p99.9
- **PostRuntimeExtensionDuration**(**AWS/Lambda**): p50, p99
- **FirstByteLatency**, **TotalRequestLatency**(**AWS/S3**): p50, p90, p95, p99, p99.9
- **ResponseLatency**(**AWS/Polly**), **TargetResponseTime**(AWS/ApplicationELB): p50, p90, p95, p99
- **Latency**, **IntegrationLatency**(**AWS/ApiGateway**): p90, p95, p99
- **Latency**, **TargetResponseTime**(**AWS/ELB**): p95, p99
- **RequestLatency**(**AWS/AppRunner**): p50, p95, p99
- **ActivityTime**, **ExecutionTime**, **LambdaFunctionRunTime**, **LambdaFunctionScheduleTime**, **LambdaFunctionTime**, **ActivityRunTime**, **ActivityScheduleTime**(**AWS/States**): p95, p99
- **EncoderBitRate**, **ConfiguredBitRate**, **ConfiguredBitRateAvailable**(**AWS/MediaLive**): p90
- **Latency**(**AWS/AppSync**): p90

Dynatrace 스트림 기본값

Dynatrace에 대한 빠른 파트너 설정 스트림은 다음 기본값을 사용합니다.

- 출력 형식: OpenTelemetry 0.7.0
- Firehose 스트림 콘텐츠 인코딩 GZIP
- Firehose 스트림 버퍼링 옵션 간격 60초, 크기 5MB
- Firehose 스트림 재시도 옵션 소요 시간 600초

New Relic 스트림 기본값

New Relic에 대한 빠른 파트너 설정 스트림은 다음 기본값을 사용합니다.

- 출력 형식: OpenTelemetry 0.7.0
- Firehose 스트림 콘텐츠 인코딩 GZIP
- Firehose 스트림 버퍼링 옵션 간격 60초, 크기 1MB
- Firehose 스트림 재시도 옵션 소요 시간 60초

Splunk Observability Cloud 스트림 기본값

Splunk Observability Cloud에 대한 빠른 파트너 설정 스트림은 다음 기본값을 사용합니다.

- 출력 형식: OpenTelemetry 0.7.0
- Firehose 스트림 콘텐츠 인코딩 GZIP
- Firehose 스트림 버퍼링 옵션 간격 60초, 크기 1MB
- Firehose 스트림 재시도 옵션 소요 시간 300초

Sumo Logic 스트림 기본값

Sumo Logic에 대한 빠른 파트너 설정 스트림은 다음 기본값을 사용합니다.

- 출력 형식: OpenTelemetry 0.7.0
- Firehose 스트림 콘텐츠 인코딩 GZIP
- Firehose 스트림 버퍼링 옵션 간격 60초, 크기 1MB
- Firehose 스트림 재시도 옵션 소요 시간 60초

스트림 가능한 통계

지표 스트림에는 항상 `Minimum`, `Maximum`, `SampleCount`, 및 `Sum` 통계가 포함됩니다. 지표 스트림에 다음과 같은 추가 통계를 포함하도록 선택할 수도 있습니다. 이 선택은 지표별 기반으로 측정됩니다. CloudWatch 통계에 대한 자세한 내용은 CloudWatch 통계 정의 단원을 참조하세요.

- p95 또는 p99와 같은 백분위수 값(JSON 또는 OpenTelemetry 형식의 스트림의 경우)
- 트림된 평균(JSON 형식의 스트림에만 해당)
- 윈소화된 평균(JSON 형식의 스트림에만 해당)

- 트림된 개수(JSON 형식의 스트림에만 해당)
- 트림된 합계(JSON 형식의 스트림에만 해당)
- 백분위수 순위(JSON 형식의 스트림에만 해당)
- 사분위수 평균(JSON 형식의 스트림에만 해당)

추가 통계를 스트리밍하면 추가 요금이 부과됩니다. 특정 지표에 대한 이러한 추가 통계 중 1~5개 간의 스트리밍은 추가 지표 업데이트로 청구됩니다. 이후에는 이러한 통계 중 최대 5개의 추가 집합이 다른 지표 업데이트로 청구됩니다.

예를 들어, 하나의 지표에 대해 p95, p99, p99.9, 트리밍 평균, 윈소화된 평균 및 트리밍된 합계의 6가지 추가 통계를 스트리밍한다고 가정합니다. 이 지표의 각 업데이트는 기본 통계를 포함하는 지표 업데이트, 처음 다섯 개의 추가 통계에 대한 업데이트, 여섯 번째 추가 통계에 대한 업데이트로 세 가지 측정 단위 업데이트가 청구됩니다. 총 10개에 대한 통계를 최대 4개까지 추가해도 청구가 증가하지는 않지만 11번째 추가 통계는 증가할 수 있습니다.

추가 통계를 스트리밍하기 위해 지표 이름과 네임스페이스 조합을 지정하면 해당 지표 이름과 네임스페이스의 모든 차원 조합이 추가 통계와 함께 스트리밍됩니다.

CloudWatch 지표 스트림은 새로운 지표와 기본 지표 업데이트 수와 추가 통계 스트리밍으로 인해 발생하는 추가 지표 업데이트를 반영하는 `TotalMetricUpdate`을 게시합니다. 자세한 내용은 CloudWatch 지표를 사용하여 지표 스트림 모니터링 단원을 참조하십시오.

자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

> **Note**
>
> 일부 지표는 백분위수를 지원하지 않습니다. 이러한 지표에 대한 백분위수 통계는 스트림에서 제외되며 지표 스트림 요금이 발생하지 않습니다. 백분위수를 지원하지 않는 이러한 통계의 예는 `AWS/ECS` 네임스페이스에 있는 지표입니다.

구성하는 추가 통계는 스트림에 대한 필터와 일치하는 경우에만 스트리밍됩니다. 예를 들어 포함 필터에 `EC2`과 `RDS`만 있는 스트림을 만드는 경우, 통계 구성 목록은 `EC2`과`Lambda`만 포함됩니다. 추가 통계가 있는 `EC2` 지표, 기본 통계만 있는 `RDS` 지표는 포함되지 않으며, `Lambda` 통계는 전혀 포함되지 않습니다.

지표 스트림 작업 및 유지 관리

지표 스트림은 항상 [실행 중(Running)] 또는 [중지됨(Stopped)]의 두 가지 상태 중 하나입니다.

- 실행 중(Running) - 지표 스트림이 올바르게 실행되고 있습니다. 스트림의 필터로 인해 대상으로 스트리밍된 지표 데이터가 없을 수 있습니다.
- 중지됨(Stopped) - 지표 스트림이 오류 때문이 아니라 누군가에 의해 명시적으로 중지되었습니다. 스트림을 삭제하지 않고 데이터 스트리밍을 임시로 일시 중지하려면 스트림을 중지하는 것이 유용할 수 있습니다.

지표 스트림을 중지했다가 다시 시작하는 경우 지표 스트림이 중지된 동안 CloudWatch에 게시된 지표 데이터는 지표 스트림에 다시 채워지지 않습니다.

지표 스트림의 출력 형식을 변경하면 경우에 따라 대상에 이전 형식과 새 형식 둘 다로 작성된 지표 데이터가 약간 표시될 수 있습니다. 이러한 상황을 방지하려면 현재 구성과 동일한 구성으로 새로운 Firehose 전송 스트림을 생성한 다음, 새로운 Firehose 전송 스트림으로 변경하는 동시에 출력 형식을 변경하면 됩니다. 이렇게 하면 출력 형식이 다른 Kinesis 레코드는 Amazon S3의 별도 객체에 저장됩니다. 나중에 원래 Firehose 전송 스트림에 트래픽을 다시 보내고 두 번째 전송 스트림을 삭제할 수 있습니다.

지표 스트림을 보고 편집하며 중지하고 시작하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 스트리밍을 선택합니다.

   스트림 목록이 나타나고 [상태(Status)] 열에 각 스트림이 실행 중인지 아니면 중지되었는지 여부가 표시됩니다.
3. 지표 스트림을 중지하거나 시작하려면 스트림을 선택하고 [중지(Stop)] 또는 [시작(Start)]을 선택합니다.
4. 지표 스트림에 관한 세부 정보를 보려면 스트림을 선택하고 [세부 정보 보기(View details)]를 선택합니다.
5. 스트림의 출력 형식, 필터, 대상 Firehose 스트림 또는 역할을 변경하려면 편집을 선택하고 원하는 대로 변경합니다.

   필터를 변경하는 경우 전환 도중에 지표 데이터에 약간의 차이가 있을 수 있습니다.

CloudWatch 지표를 사용하여 지표 스트림 모니터링

지표 스트림은 `AWS/CloudWatch/MetricStreams` 네임스페이스의 상태 및 작업에 관한 CloudWatch 지표를 내보냅니다. 아래에 나와 있는 지표를 내보냅니다. 두 지표 모두 `MetricStreamName` 차원 및 차원 없이 내보내집니다. 차원 없이 지표를 사용하면 모든 지표 스트림

에 대해 집계된 지표를 확인할 수 있습니다. MetricStreamName 차원과 함께 지표를 사용하면 해당 지표 스트림에 관한 지표만 볼 수 있습니다.

이러한 두 지표 모두에서 값은 [실행 중(Running)] 상태에 있는 지표 스트림에 대해서만 내보내집니다.

지표	설명
MetricUpdate	지표 스트림으로 발송된 지표 업데이트 수입니다. 일정 기간 동안 지표 업데이트가 스트리밍되지 않으면 해당 기간 동안 이 지표가 내보내지지 않습니다. 지표 스트림을 중지하면 지표 스트림이 다시 시작될 때까지 이 지표의 내보내기가 중지됩니다. 유효한 통계: Sum 단위: 없음
TotalMetricUpdate	이 값은 MetricUpdate + 스트리밍 중인 추가 통계를 기반으로 한 숫자로 업데이트됩니다. 각 고유한 네임스페이스 및 지표 이름 조합에 대해 1~5의 추가 통계를 스트리밍하면 TotalMetricUpdate 에 1이 추가되며, 6~10개의 추가 통계를 스트리밍하면 TotalMetricUpdate 에 2가 추가되는 식입니다. 유효한 통계: Sum 단위: 없음
PublishErrorRate	Firehose 전송 스트림에 데이터를 넣을 때 발생하는 복구할 수 없는 오류의 수입니다. 특정 기간 동안 오류가 발생하지 않으면 해당 기간 동안 이 지표가 내보내지지 않습니다. 지표 스트림을 중지하면 지표 스트림이 다시 시작될 때까지 이 지표의 내보내기가 중지됩니다. 유효한 통계: Average - 쓰기가 불가능한 지표 업데이트의 비율을 확인할 수 있습니다. 이 값은 0.0 ~1.0입니다. 단위: 없음

CloudWatch와 Firehose 간의 신뢰

Firehose 전송 스트림은 Firehose에 대한 쓰기 권한이 있는 IAM 역할을 통해 CloudWatch를 신뢰해야 합니다. 이러한 권한은 CloudWatch 지표 스트림이 사용하는 단일 Firehose 전송 스트림으로 제한될 수 있습니다. IAM 역할은 `streams.metrics.cloudwatch.amazonaws.com` 서비스 보안 주체를 신뢰해야 합니다.

CloudWatch 콘솔을 사용하여 지표 스트림을 생성하는 경우 CloudWatch가 올바른 권한이 있는 역할을 생성하도록 할 수 있습니다. 다른 방법을 사용하여 지표 스트림을 생성하거나 IAM 역할 자체를 생성하려는 경우 다음 권한 정책 및 신뢰 정책을 포함해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:firehose:region:account-id:deliverystream/*"
        }
    ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "streams.metrics.cloudwatch.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

CloudWatch는 지표 스트림 리소스를 소유하는 소스를 대신하여 지표 데이터를 대상 Firehose 전송 스트림에 스트리밍합니다.

지표 스트림 출력 형식

CloudWatch 지표 스트림의 데이터는 JSON 형식 또는 OpenTelemetry 형식일 수 있습니다. 현재 OpenTelemetry 1.0.0과 0.7.0 형식이 모두 지원됩니다.

목차

JSON 형식

JSON 형식을 사용하는 CloudWatch 지표 스트림에서는 각 Firehose 레코드에 줄 바꿈 문자(\n)로 구분된 여러 JSON 객체가 포함됩니다. 각 객체에는 단일 지표의 단일 데이터 요소가 포함됩니다.

사용되는 JSON 형식은 AWS Glue 및 Amazon Athena와 완전히 호환됩니다. Firehose 전송 스트림 및 AWS Glue 테이블을 올바르게 형식 지정한 경우 형식은 S3에 저장되기 전에 Parquet 형식 또는 ORC(Optimized Row Columnar) 형식으로 자동 변환될 수 있습니다. 형식 변환에 대한 자세한 내용은 Converting Your Input Record Format in Firehose 섹션을 참조하세요. AWS Glue의 올바른 형식에 대한 자세한 내용은 JSON 출력 형식에 사용해야 하는 AWS Glue 스키마 단원을 참조하세요.

JSON 형식에서 유효한 `unit` 값은 `MetricDatum` API 구조의 `unit` 값과 동일합니다. 자세한 내용은 MetricDatum 단원을 참조하세요. `timestamp` 필드의 값은 `1616004674229`와 같은 Epoch 밀리초 단위입니다.

다음은 형식의 예입니다. 이 예에서 JSON은 읽기 쉽도록 형식이 지정되어 있지만 실제로는 전체 형식이 한 줄에 있습니다.

```
{
    "metric_stream_name": "MyMetricStream",
    "account_id": "1234567890",
    "region": "us-east-1",
```

```
    "namespace": "AWS/EC2",
    "metric_name": "DiskWriteOps",
    "dimensions": {
        "InstanceId": "i-123456789012"
    },
    "timestamp": 1611929698000,
    "value": {
        "count": 3.0,
        "sum": 20.0,
        "max": 18.0,
        "min": 0.0,
        "p99": 17.56,
        "p99.9": 17.8764,
        "TM(25%:75%)": 16.43
    },
    "unit": "Seconds"
}
```

JSON 출력 형식에 사용해야 하는 AWS Glue 스키마

다음은 Firehose에서 사용하게 될 AWS Glue 테이블의 `StorageDescriptor`에 대한 JSON 표현 예입니다. `StorageDescriptor`에 대한 자세한 내용은 StorageDescriptor 단원을 참조하세요.

```
{
  "Columns": [
    {
      "Name": "metric_stream_name",
      "Type": "string"
    },
    {
      "Name": "account_id",
      "Type": "string"
    },
    {
      "Name": "region",
      "Type": "string"
    },
    {
      "Name": "namespace",
      "Type": "string"
    },
    {
      "Name": "metric_name",
```

```
      "Type": "string"
    },
    {
      "Name": "timestamp",
      "Type": "timestamp"
    },
    {
      "Name": "dimensions",
      "Type": "map<string,string>"
    },
    {
      "Name": "value",
      "Type":
 "struct<min:double,max:double,count:double,sum:double,p99:double,p99.9:double>"
    },
    {
      "Name": "unit",
      "Type": "string"
    }
  ],
  "Location": "s3://my-s3-bucket/",
  "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
  "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
  "SerdeInfo": {
    "SerializationLibrary": "org.apache.hive.hcatalog.data.JsonSerDe"
  },
  "Parameters": {
    "classification": "json"
  }
}
```

앞의 예는 Amazon S3에 JSON 형식으로 작성된 데이터에 대한 것입니다. 데이터를 Parquet 형식 또는 ORC(Optimized Row Columnar) 형식으로 저장하려면 다음 필드의 값을 표시된 값으로 바꿉니다.

- Parquet:
  - inputFormat: org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat
  - outputFormat: org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat
  - SerDeInfo.serializationLib: org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe
  - parameters.classification: parquet
- ORC:
  - inputFormat: org.apache.hadoop.hive.ql.io.orc.OrcInputFormat

- outputFormat: org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
- SerDeInfo.serializationLib: org.apache.hadoop.hive.ql.io.orc.OrcSerde
- parameters.classification: orc

OpenTelemetry 1.0.0 형식

> **Note**
>
> OpenTelemetry 1.0.0 형식을 사용하면 지표 속성이 0.7.0 형식에서 사용되는 `StringKeyValue` 유형 대신 `KeyValue` 객체 목록으로 인코딩됩니다. 소비자의 입장에서 볼 때 0.7.0과 1.0.0 형식 사이의 주요 변경 사항은 이 정도뿐입니다. 0.7.0 proto 파일에서 생성된 구문 분석기는 1.0.0 형식으로 인코딩된 지표 속성을 구문 분석하지 않습니다. 반대의 경우도 마찬가지입니다. 1.0.0 proto 파일에서 생성된 구문 분석는 0.7.0 형식으로 인코딩된 지표 속성을 구문 분석하지 않습니다.

OpenTelemetry는 도구, API 및 SDK 모음입니다. OpenTelemetry를 사용하여 분석을 위한 원격 측정 데이터(지표, 로그, 추적)를 계측하고 생성하며 수집하고 내보낼 수 있습니다. OpenTelemetry는 Cloud Native Computing Foundation(CNCF)의 일부입니다. 자세한 내용은 OpenTelemetry를 참조하세요.

전체 OpenTelemetry 1.0.0 사양에 대한 내용은 릴리스 버전 1.0.0을 참조하세요.

Kinesis 레코드에는 하나 이상의 `ExportMetricsServiceRequest` OpenTelemetry 데이터 구조가 포함될 수 있습니다. 각 데이터 구조는 바이트 단위의 레코드 길이를 나타내는 `UnsignedVarInt32`가 있는 헤더로 시작합니다. 각 `ExportMetricsServiceRequest`에는 동시에 여러 지표의 데이터가 포함될 수 있습니다.

다음은 `ExportMetricsServiceRequest` OpenTelemetry 데이터 구조 메시지의 문자열 표현입니다. OpenTelemetry는 Google Protocol Buffers 바이너리 프로토콜을 직렬화하며 이것은 사람이 읽을 수 없습니다.

```
resource_metrics {
  resource {
    attributes {
      key: "cloud.provider"
      value {
        string_value: "aws"
      }
    }
```

```
    attributes {
      key: "cloud.account.id"
      value {
        string_value: "123456789012"
      }
    }
    attributes {
      key: "cloud.region"
      value {
        string_value: "us-east-1"
      }
    }
    attributes {
      key: "aws.exporter.arn"
      value {
        string_value: "arn:aws:cloudwatch:us-east-1:123456789012:metric-stream/
MyMetricStream"
      }
    }
  }
  scope_metrics {
    metrics {
      name: "amazonaws.com/AWS/DynamoDB/ConsumedReadCapacityUnits"
      unit: "NoneTranslated"
      summary {
        data_points {
          start_time_unix_nano: 60000000000
          time_unix_nano: 120000000000
          count: 1
          sum: 1.0
          quantile_values {
            value: 1.0
          }
          quantile_values {
            quantile: 0.95
            value: 1.0
          }
          quantile_values {
            quantile: 0.99
            value: 1.0
          }
          quantile_values {
            quantile: 1.0
            value: 1.0
```

```
          }
          attributes {
            key: "Namespace"
            value {
              string_value: "AWS/DynamoDB"
            }
          }
          attributes {
            key: "MetricName"
            value {
              string_value: "ConsumedReadCapacityUnits"
            }
          }
          attributes {
            key: "Dimensions"
            value {
              kvlist_value {
                values {
                  key: "TableName"
                  value {
                    string_value: "MyTable"
                  }
                }
              }
            }
          }
        }
        data_points {
          start_time_unix_nano: 70000000000
          time_unix_nano: 130000000000
          count: 2
          sum: 5.0
          quantile_values {
            value: 2.0
          }
          quantile_values {
            quantile: 1.0
            value: 3.0
          }
          attributes {
            key: "Namespace"
            value {
              string_value: "AWS/DynamoDB"
            }
```

```
          }
          attributes {
            key: "MetricName"
            value {
              string_value: "ConsumedReadCapacityUnits"
            }
          }
          attributes {
            key: "Dimensions"
            value {
              kvlist_value {
                values {
                  key: "TableName"
                  value {
                    string_value: "MyTable"
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

OpenTelemetry 지표 데이터를 직렬화하는 최상위 객체

`ExportMetricsServiceRequest`는 OpenTelemetry Exporter 페이로드를 직렬화하는 최상위 래퍼입니다. 여기에는 하나 이상의 `ResourceMetrics`가 포함됩니다.

```
message ExportMetricsServiceRequest {
  // An array of ResourceMetrics.
  // For data coming from a single resource this array will typically contain one
  // element. Intermediary nodes (such as OpenTelemetry Collector) that receive
  // data from multiple origins typically batch the data before forwarding further and
  // in that case this array will contain multiple elements.
  repeated opentelemetry.proto.metrics.v1.ResourceMetrics resource_metrics = 1;
}
```

`ResourceMetrics`는 MetricData 객체를 나타내는 최상위 객체입니다.

```
// A collection of ScopeMetrics from a Resource.
```

```
message ResourceMetrics {
  reserved 1000;

  // The resource for the metrics in this message.
  // If this field is not set then no resource info is known.
  opentelemetry.proto.resource.v1.Resource resource = 1;

  // A list of metrics that originate from a resource.
  repeated ScopeMetrics scope_metrics = 2;

  // This schema_url applies to the data in the "resource" field. It does not apply
  // to the data in the "scope_metrics" field which have their own schema_url field.
  string schema_url = 3;
}
```

Resource 객체

`Resource` 객체는 지표를 생성한 리소스에 관한 일부 정보를 포함하는 값 페어 객체입니다. AWS에서 생성한 지표의 경우 데이터 구조에 지표와 관련된 리소스(예: EC2 인스턴스 또는 S3 버킷)의 Amazon 리소스 이름(ARN)이 포함됩니다.

`Resource` 객체에는 키-값 페어 목록을 저장하는 `attributes`라는 속성이 포함되어 있습니다.

- `cloud.account.id`에는 계정 ID가 포함됩니다.
- `cloud.region`에는 리전이 포함됩니다.
- `aws.exporter.arn`에는 지표 스트림 ARN이 포함됩니다.
- `cloud.provider`은(는) 항상 `aws`입니다.

```
// Resource information.
message Resource {
  // Set of attributes that describe the resource.
  // Attribute keys MUST be unique (it is not allowed to have more than one
  // attribute with the same key).
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 1;

  // dropped_attributes_count is the number of dropped attributes. If the value is 0,
 then
  // no attributes were dropped.
  uint32 dropped_attributes_count = 2;
}
```

ScopeMetrics 객체

scope 필드는 채워지지 않습니다. 내보내는 지표 필드만 채워집니다.

```
// A collection of Metrics produced by an Scope.
message ScopeMetrics {
  // The instrumentation scope information for the metrics in this message.
  // Semantically when InstrumentationScope isn't set, it is equivalent with
  // an empty instrumentation scope name (unknown).
  opentelemetry.proto.common.v1.InstrumentationScope scope = 1;

  // A list of metrics that originate from an instrumentation library.
  repeated Metric metrics = 2;

  // This schema_url applies to all metrics in the "metrics" field.
  string schema_url = 3;
}
```

Metric 객체

Metric 객체에는 SummaryDataPoint 목록을 포함하는 Summary 데이터 필드와 일부 메타데이터가 포함되어 있습니다.

지표 스트림의 경우 메타데이터는 다음과 같습니다.

- name은 amazonaws.com/*metric_namespace*/*metric_name*이 됩니다.
- description은 비어 있게 됩니다.
- unit은 지표 데이터의 단위를 측정 단위에 대한 통합 코드의 대소문자를 구분하는 변형에 매핑함으로써 채워집니다. 자세한 내용은 OpenTelemetry 1.0.0 형식으로 변환 단원 및 측정 단위에 대한 통합 코드를 참조하세요.
- type은 SUMMARY이 됩니다.

```
message Metric {
  reserved 4, 6, 8;

  // name of the metric, including its DNS name prefix. It must be unique.
  string name = 1;

  // description of the metric, which can be used in documentation.
  string description = 2;
```

```
  // unit in which the metric value is reported. Follows the format
  // described by http://unitsofmeasure.org/ucum.html.
  string unit = 3;

  // Data determines the aggregation type (if any) of the metric, what is the
  // reported value type for the data points, as well as the relatationship to
  // the time interval over which they are reported.
  oneof data {
    Gauge gauge = 5;
    Sum sum = 7;
    Histogram histogram = 9;
    ExponentialHistogram exponential_histogram = 10;
    Summary summary = 11;
  }
}

message Summary {
  repeated SummaryDataPoint data_points = 1;
}
```

SummaryDataPoint 객체

SummaryDataPoint 객체에는 DoubleSummary 지표의 시계열에 있는 단일 데이터 포인트 값이 포함되어 있습니다.

```
// SummaryDataPoint is a single data point in a timeseries that describes the
// time-varying values of a Summary metric.
message SummaryDataPoint {
  reserved 1;

  // The set of key/value pairs that uniquely identify the timeseries from
  // where this point belongs. The list may be empty (may contain 0 elements).
  // Attribute keys MUST be unique (it is not allowed to have more than one
  // attribute with the same key).
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 7;

  // StartTimeUnixNano is optional but strongly encouraged, see the
  // the detailed comments above Metric.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  fixed64 start_time_unix_nano = 2;
```

```
  // TimeUnixNano is required, see the detailed comments above Metric.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  fixed64 time_unix_nano = 3;

  // count is the number of values in the population. Must be non-negative.
  fixed64 count = 4;

  // sum of the values in the population. If count is zero then this field
  // must be zero.
  //
  // Note: Sum should only be filled out when measuring non-negative discrete
  // events, and is assumed to be monotonic over the values of these events.
  // Negative events *can* be recorded, but sum should not be filled out when
  // doing so.  This is specifically to enforce compatibility w/ OpenMetrics,
  // see: https://github.com/OpenObservability/OpenMetrics/blob/main/specification/
OpenMetrics.md#summary
  double sum = 5;

  // Represents the value at a given quantile of a distribution.
  //
  // To record Min and Max values following conventions are used:
  // - The 1.0 quantile is equivalent to the maximum value observed.
  // - The 0.0 quantile is equivalent to the minimum value observed.
  //
  // See the following issue for more context:
  // https://github.com/open-telemetry/opentelemetry-proto/issues/125
  message ValueAtQuantile {
    // The quantile of a distribution. Must be in the interval
    // [0.0, 1.0].
    double quantile = 1;

    // The value at the given quantile of a distribution.
    //
    // Quantile values must NOT be negative.
    double value = 2;
  }

  // (Optional) list of values at different quantiles of the distribution calculated
  // from the current snapshot. The quantiles must be strictly increasing.
  repeated ValueAtQuantile quantile_values = 6;

  // Flags that apply to this specific data point.  See DataPointFlags
```

```
  // for the available flags and their meaning.
  uint32 flags = 8;
}
```

자세한 내용은 OpenTelemetry 1.0.0 형식으로 변환 단원을 참조하십시오.

OpenTelemetry 1.0.0 형식으로 변환

CloudWatch는 CloudWatch 데이터를 OpenTelemetry 형식으로 저장하기 위해 몇 가지 변환을 수행합니다.

네임스페이스, 지표 이름 및 차원 변환

이러한 속성은 매핑에 인코딩된 키-값 페어입니다.

- 한 속성에 `Namespace` 키가 있고 그 값은 지표의 네임스페이스입니다.
- 한 속성에 `MetricName` 키가 있고 그 값은 지표의 이름입니다.
- 한 페어에 `Dimensions` 키가 있고 그 값은 키-값 페어의 중첩 목록입니다. 이 목록의 각 페어는 CloudWatch 지표 측정기준에 매핑됩니다. 여기서 페어의 키는 측정기준 이름이고 해당 값은 측정기준 값입니다.

Average, Sum, SampleCount, Min 및 Max 변환

요약 데이터 요소를 사용하면 CloudWatch가 하나의 데이터 요소를 사용해 이러한 모든 통계를 내보낼 수 있습니다.

- `startTimeUnixNano`에는 CloudWatch `startTime`이 포함됩니다.
- `timeUnixNano`에는 CloudWatch `endTime`이 포함됩니다.
- `sum`에는 Sum 통계가 포함됩니다.
- `count`에는 SampleCount 통계가 포함됩니다.
- `quantile_values`에는 다음과 같이 두 개의 `valueAtQuantile.value` 객체가 포함됩니다.
  - `valueAtQuantile.value = Min value`가 있는 `valueAtQuantile.quantile = 0.0`
  - `valueAtQuantile.value = p99 value`가 있는 `valueAtQuantile.quantile = 0.99`
  - `valueAtQuantile.value = p99.9 value`가 있는 `valueAtQuantile.quantile = 0.999`
  - `valueAtQuantile.value = Max value`가 있는 `valueAtQuantile.quantile = 1.0`

지표 스트림을 사용하는 리소스에서는 Average 통계를 Sum/SampleCount로 계산할 수 있습니다.

단위 변환

CloudWatch 단위는 다음 표와 같이 측정 단위에 대한 통합 코드의 대소문자를 구분하는 변형에 매핑됩니다. 자세한 내용은 측정 단위에 대한 통합 코드를 참조하세요.

CloudWatch	OpenTelemetry
초	s
Second 또는 Seconds	s
마이크로초	us
밀리초	ms
바이트	By
KB	kBy
MB	MBy
GB	GBy
TB	TBy
비트	bit
Kbit	kbit
Mbit	MBit
Gbit	GBit
Tbit	Tbit
%	%
개수	{Count}
None	1

슬래시와 결합된 단위는 두 단위 모두의 OpenTelemetry 변환을 적용하여 매핑됩니다. 예를 들어 Bytes/Second는 By/s에 매핑됩니다.

OpenTelemetry 1.0.0 메시지를 구문 분석하는 방법

이 섹션에서는 OpenTelemetry 1.0.0의 구문 분석을 시작하는 데 도움이 되는 정보를 제공합니다.

먼저, 언어별 바인딩을 가져와야 합니다. 그러면 이 바인딩을 통해 원하는 언어로 OpenTelemetry 1.0.0 메시지를 구문 분석할 수 있습니다.

언어별 바인딩을 가져오려면

- 단계는 원하는 언어에 따라 다릅니다.
  - Java를 사용하려면 Java 프로젝트에 다음 Maven 종속 항목을 추가합니다. OpenTelemetry Java >> 0.14.1.
  - 다른 언어를 사용하려면 다음 단계를 따릅니다.

    a. 클래스 생성에서 목록을 확인하여 언어가 지원되는지 확인합니다.

    b. Protocol Buffers 다운로드의 단계에 따라 Protobuf 컴파일러를 설치합니다.

    c. 릴리스 버전 1.0.0에서 OpenTelemetry 0.7.0 ProtoBuf 정의를 다운로드합니다.

    d. 다운로드한 OpenTelemetry 0.7.0 ProtoBuf 정의의 루트 폴더에 있는지 확인합니다. 그런 다음에 `src` 폴더를 생성한 후 명령을 실행하여 언어별 바인딩을 생성합니다. 자세한 내용은 클래스 생성을 참조하세요.

    다음은 Javascript 바인딩을 생성하는 방법의 예입니다.

```
protoc --proto_path=./ --js_out=import_style=commonjs,binary:src \
opentelemetry/proto/common/v1/common.proto \
opentelemetry/proto/resource/v1/resource.proto \
opentelemetry/proto/metrics/v1/metrics.proto \
opentelemetry/proto/collector/metrics/v1/metrics_service.proto
```

다음 단원에는 이전 지침을 사용하여 구축할 수 있는 언어별 바인딩을 사용하는 예가 나와 있습니다.

Java

```
package com.example;

import io.opentelemetry.proto.collector.metrics.v1.ExportMetricsServiceRequest;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class MyOpenTelemetryParser {

    public List<ExportMetricsServiceRequest> parse(InputStream inputStream) throws
 IOException {
        List<ExportMetricsServiceRequest> result = new ArrayList<>();

        ExportMetricsServiceRequest request;
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        while ((request =
 ExportMetricsServiceRequest.parseDelimitedFrom(inputStream)) != null) {
            // Do whatever we want with the parsed message
            result.add(request);
        }

        return result;
    }
}
```

Javascript

이 예에서는 생성한 바인딩이 있는 루트 폴더가 `./`라고 가정합니다.

`parseRecord` 함수의 데이터 인수는 다음 유형 중 하나일 수 있습니다.

- `Uint8Array` - 이 유형이 최적입니다.
- `Buffer`- 이 유형은 노드에서 최적입니다.
- `Array.`*`number`* - 8비트 정수입니다.

```
const pb = require('google-protobuf')
```

```
const pbMetrics =
    require('./opentelemetry/proto/collector/metrics/v1/metrics_service_pb')

function parseRecord(data) {
    const result = []

    // Loop until we've read all the data from the buffer
    while (data.length) {
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        const reader = new pb.BinaryReader(data)
        const messageLength = reader.decoder_.readUnsignedVarint32()
        const messageFrom = reader.decoder_.cursor_
        const messageTo = messageFrom + messageLength

        // Extract the current `ExportMetricsServiceRequest` message to parse
        const message = data.subarray(messageFrom, messageTo)

        // Parse the current message using the ProtoBuf library
        const parsed =
            pbMetrics.ExportMetricsServiceRequest.deserializeBinary(message)

        // Do whatever we want with the parsed message
        result.push(parsed.toObject())

        // Shrink the remaining buffer, removing the already parsed data
        data = data.subarray(messageTo)
    }

    return result
}
```

Python

`var-int` 구분 기호를 직접 읽거나 내부 메서드 `_VarintBytes(size)` 및 `_DecodeVarint32(buffer, position)`를 사용해야 합니다. 이러한 메서드는 size 바이트 바로 뒤에 있는 버퍼의 위치를 반환합니다. 읽기 측에서는 메시지의 바이트만 읽는 것으로 제한되는 새 버퍼를 생성합니다.

```
size = my_metric.ByteSize()
f.write(_VarintBytes(size))
f.write(my_metric.SerializeToString())
msg_len, new_pos = _DecodeVarint32(buf, 0)
msg_buf = buf[new_pos:new_pos+msg_len]
request = metrics_service_pb.ExportMetricsServiceRequest()
request.ParseFromString(msg_buf)
```

Go

`Buffer.DecodeMessage()`를 사용합니다.

C#

`CodedInputStream`를 사용합니다. 이 클래스는 크기로 구분된 메시지를 읽을 수 있습니다.

C++

`google/protobuf/util/delimited_message_util.h`에 기술된 함수는 크기로 구분된 메시지를 읽을 수 있습니다.

기타 언어

기타 언어의 경우 Protocol Buffers 다운로드를 참조하세요.

구문 분석기를 구현할 때 Kinesis 레코드에 여러 `ExportMetricsServiceRequest` Protocol Buffers 메시지가 포함될 수 있다는 점을 고려하세요. 각 메시지는 바이트 단위의 레코드 길이를 나타내는 `UnsignedVarInt32`가 있는 헤더로 시작합니다.

OpenTelemetry 0.7.0 형식

OpenTelemetry는 도구, API 및 SDK 모음입니다. OpenTelemetry를 사용하여 분석을 위한 원격 측정 데이터(지표, 로그, 추적)를 계측하고 생성하며 수집하고 내보낼 수 있습니다. OpenTelemetry는 Cloud Native Computing Foundation(CNCF)의 일부입니다. 자세한 내용은 OpenTelemetry를 참조하세요.

전체 OpenTelemetry 0.7.0 사양에 대한 내용은 v0.7.0 릴리스를 참조하세요.

Kinesis 레코드에는 하나 이상의 `ExportMetricsServiceRequest` OpenTelemetry 데이터 구조가 포함될 수 있습니다. 각 데이터 구조는 바이트 단위의 레코드 길이를 나타내는 `UnsignedVarInt32`가 있는 헤더로 시작합니다. 각 `ExportMetricsServiceRequest`에는 동시에 여러 지표의 데이터가 포함될 수 있습니다.

다음은 ExportMetricsServiceRequest OpenTelemetry 데이터 구조 메시지의 문자열 표현입니다. OpenTelemetry는 Google Protocol Buffers 바이너리 프로토콜을 직렬화하며 이것은 사람이 읽을 수 없습니다.

```
resource_metrics {
  resource {
    attributes {
      key: "cloud.provider"
      value {
        string_value: "aws"
      }
    }
    attributes {
      key: "cloud.account.id"
      value {
        string_value: "2345678901"
      }
    }
    attributes {
      key: "cloud.region"
      value {
        string_value: "us-east-1"
      }
    }
    attributes {
      key: "aws.exporter.arn"
      value {
        string_value: "arn:aws:cloudwatch:us-east-1:123456789012:metric-stream/
MyMetricStream"
      }
    }
  }
  instrumentation_library_metrics {
    metrics {
      name: "amazonaws.com/AWS/DynamoDB/ConsumedReadCapacityUnits"
      unit: "1"
      double_summary {
        data_points {
          labels {
            key: "Namespace"
            value: "AWS/DynamoDB"
          }
          labels {
```

```
            key: "MetricName"
            value: "ConsumedReadCapacityUnits"
          }
          labels {
            key: "TableName"
            value: "MyTable"
          }
          start_time_unix_nano: 1604948400000000000
          time_unix_nano: 1604948460000000000
          count: 1
          sum: 1.0
          quantile_values {
            quantile: 0.0
            value: 1.0
          }
          quantile_values {
            quantile: 0.95
            value: 1.0
          }
          quantile_values {
            quantile: 0.99
            value: 1.0
          }
          quantile_values {
            quantile: 1.0
            value: 1.0
          }
        }
        data_points {
          labels {
            key: "Namespace"
            value: "AWS/DynamoDB"
          }
          labels {
            key: "MetricName"
            value: "ConsumedReadCapacityUnits"
          }
          labels {
            key: "TableName"
            value: "MyTable"
          }
          start_time_unix_nano: 1604948460000000000
          time_unix_nano: 1604948520000000000
          count: 2
```

```
          sum: 5.0
          quantile_values {
            quantile: 0.0
            value: 2.0
          }
          quantile_values {
            quantile: 1.0
            value: 3.0
          }
        }
      }
    }
  }
}
```

OpenTelemetry 지표 데이터를 직렬화하는 최상위 객체

`ExportMetricsServiceRequest`는 OpenTelemetry Exporter 페이로드를 직렬화하는 최상위 래퍼입니다. 여기에는 하나 이상의 `ResourceMetrics`가 포함됩니다.

```
message ExportMetricsServiceRequest {
  // An array of ResourceMetrics.
  // For data coming from a single resource this array will typically contain one
  // element. Intermediary nodes (such as OpenTelemetry Collector) that receive
  // data from multiple origins typically batch the data before forwarding further and
  // in that case this array will contain multiple elements.
  repeated opentelemetry.proto.metrics.v1.ResourceMetrics resource_metrics = 1;
}
```

`ResourceMetrics`는 MetricData 객체를 나타내는 최상위 객체입니다.

```
// A collection of InstrumentationLibraryMetrics from a Resource.
message ResourceMetrics {
  // The resource for the metrics in this message.
  // If this field is not set then no resource info is known.
  opentelemetry.proto.resource.v1.Resource resource = 1;

  // A list of metrics that originate from a resource.
  repeated InstrumentationLibraryMetrics instrumentation_library_metrics = 2;
}
```

Resource 객체

Resource 객체는 지표를 생성한 리소스에 관한 일부 정보를 포함하는 값 페어 객체입니다. AWS에서 생성한 지표의 경우 데이터 구조에 지표와 관련된 리소스(예: EC2 인스턴스 또는 S3 버킷)의 Amazon 리소스 이름(ARN)이 포함됩니다.

Resource 객체에는 키-값 페어 목록을 저장하는 attributes라는 속성이 포함되어 있습니다.

- cloud.account.id에는 계정 ID가 포함됩니다.
- cloud.region에는 리전이 포함됩니다.
- aws.exporter.arn에는 지표 스트림 ARN이 포함됩니다.
- cloud.provider은(는) 항상 aws입니다.

```
// Resource information.
message Resource {
  // Set of labels that describe the resource.
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 1;

  // dropped_attributes_count is the number of dropped attributes. If the value is 0,
  // no attributes were dropped.
  uint32 dropped_attributes_count = 2;
}
```

InstrumentationLibraryMetrics 객체

instrumentation_library 필드는 채워지지 않습니다. 내보내는 지표 필드만 채워집니다.

```
// A collection of Metrics produced by an InstrumentationLibrary.
message InstrumentationLibraryMetrics {
  // The instrumentation library information for the metrics in this message.
  // If this field is not set then no library info is known.
  opentelemetry.proto.common.v1.InstrumentationLibrary instrumentation_library = 1;
  // A list of metrics that originate from an instrumentation library.
  repeated Metric metrics = 2;
}
```

Metric 객체

Metric 객체에는 DoubleSummaryDataPoint 목록을 포함하는 DoubleSummary 데이터 필드가 포함되어 있습니다.

```
message Metric {
```

```
  // name of the metric, including its DNS name prefix. It must be unique.
  string name = 1;

  // description of the metric, which can be used in documentation.
  string description = 2;

  // unit in which the metric value is reported. Follows the format
  // described by http://unitsofmeasure.org/ucum.html.
  string unit = 3;

  oneof data {
    IntGauge int_gauge = 4;
    DoubleGauge double_gauge = 5;
    IntSum int_sum = 6;
    DoubleSum double_sum = 7;
    IntHistogram int_histogram = 8;
    DoubleHistogram double_histogram = 9;
    DoubleSummary double_summary = 11;
  }
}

message DoubleSummary {
  repeated DoubleSummaryDataPoint data_points = 1;
}
```

MetricDescriptor 객체

MetricDescriptor 객체에는 메타데이터가 포함되어 있습니다. 자세한 내용은 GitHub에서 metrics.proto를 참조하세요.

지표 스트림의 경우 MetricDescriptor의 내용은 다음과 같습니다.

- name은 amazonaws.com/*metric_namespace*/*metric_name*이 됩니다.
- description은 비어 있게 됩니다.
- unit은 지표 데이터의 단위를 측정 단위에 대한 통합 코드의 대소문자를 구분하는 변형에 매핑함으로써 채워집니다. 자세한 내용은 OpenTelemetry 0.7.0 형식으로 변환 단원 및 측정 단위에 대한 통합 코드를 참조하세요.
- type은 SUMMARY가 됩니다.

DoubleSummaryDataPoint 객체

DoubleSummaryDataPoint 객체에는 DoubleSummary 지표의 시계열에 있는 단일 데이터 요소 값이 포함되어 있습니다.

```
// DoubleSummaryDataPoint is a single data point in a timeseries that describes the
// time-varying values of a Summary metric.
message DoubleSummaryDataPoint {
  // The set of labels that uniquely identify this timeseries.
  repeated opentelemetry.proto.common.v1.StringKeyValue labels = 1;

  // start_time_unix_nano is the last time when the aggregation value was reset
  // to "zero". For some metric types this is ignored, see data types for more
  // details.
  //
  // The aggregation value is over the time interval (start_time_unix_nano,
  // time_unix_nano].
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  //
  // Value of 0 indicates that the timestamp is unspecified. In that case the
  // timestamp may be decided by the backend.
  fixed64 start_time_unix_nano = 2;

  // time_unix_nano is the moment when this aggregation value was reported.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  fixed64 time_unix_nano = 3;

  // count is the number of values in the population. Must be non-negative.
  fixed64 count = 4;

  // sum of the values in the population. If count is zero then this field
  // must be zero.
  double sum = 5;

  // Represents the value at a given quantile of a distribution.
  //
  // To record Min and Max values following conventions are used:
  // - The 1.0 quantile is equivalent to the maximum value observed.
  // - The 0.0 quantile is equivalent to the minimum value observed.
  message ValueAtQuantile {
    // The quantile of a distribution. Must be in the interval
```

```
    // [0.0, 1.0].
    double quantile = 1;

    // The value at the given quantile of a distribution.
    double value = 2;
  }

  // (Optional) list of values at different quantiles of the distribution calculated
  // from the current snapshot. The quantiles must be strictly increasing.
  repeated ValueAtQuantile quantile_values = 6;
}
```

자세한 정보는 OpenTelemetry 0.7.0 형식으로 변환을 참조하세요.

OpenTelemetry 0.7.0 형식으로 변환

CloudWatch는 CloudWatch 데이터를 OpenTelemetry 형식으로 저장하기 위해 몇 가지 변환을 수행합니다.

네임스페이스, 지표 이름 및 차원 변환

이러한 속성은 매핑에 인코딩된 키-값 페어입니다.

- 하나의 페어에는 지표의 네임스페이스가 포함됩니다.
- 하나의 페어에는 지표의 이름이 포함됩니다.
- 각 차원에 대해 CloudWatch는 `metricDatum.Dimensions[i].`*`Name`*, `metricDatum.Dimensions[i].`*`Value`* 페어를 저장합니다.

Average, Sum, SampleCount, Min 및 Max 변환

요약 데이터 요소를 사용하면 CloudWatch가 하나의 데이터 요소를 사용해 이러한 모든 통계를 내보낼 수 있습니다.

- `startTimeUnixNano`에는 CloudWatch `startTime`이 포함됩니다.
- `timeUnixNano`에는 CloudWatch `endTime`이 포함됩니다.
- `sum`에는 Sum 통계가 포함됩니다.
- `count`에는 SampleCount 통계가 포함됩니다.
- `quantile_values`에는 다음과 같이 두 개의 `valueAtQuantile.value` 객체가 포함됩니다.

- `valueAtQuantile.value = Min value`가 있는 `valueAtQuantile.quantile = 0.0`
- `valueAtQuantile.value = p99 value`가 있는 `valueAtQuantile.quantile = 0.99`
- `valueAtQuantile.value = p99.9 value`가 있는 `valueAtQuantile.quantile = 0.999`
- `valueAtQuantile.value = Max value`가 있는 `valueAtQuantile.quantile = 1.0`

지표 스트림을 사용하는 리소스에서는 Average 통계를 Sum/SampleCount로 계산할 수 있습니다.

단위 변환

CloudWatch 단위는 다음 표와 같이 측정 단위에 대한 통합 코드의 대소문자를 구분하는 변형에 매핑됩니다. 자세한 내용은 측정 단위에 대한 통합 코드를 참조하세요.

CloudWatch	OpenTelemetry
초	s
Second 또는 Seconds	s
Microsecond	us
밀리초	ms
바이트	By
KB	kBy
MB	MBy
GB	GBy
TB	TBy
비트	bit
Kbit	kbit
Mbit	MBit
Gbit	GBit

CloudWatch	OpenTelemetry
Tbit	Tbit
%	%
개수	{Count}
None	1

슬래시와 결합된 단위는 두 단위 모두의 OpenTelemetry 변환을 적용하여 매핑됩니다. 예를 들어 Bytes/Second는 By/s에 매핑됩니다.

OpenTelemetry 0.7.0 메시지를 구문 분석하는 방법

이 단원에서는 OpenTelemetry 0.7.0의 구문 분석을 시작하는 데 도움이 되는 정보를 제공합니다.

먼저, 언어별 바인딩을 가져와야 합니다. 그러면 이 바인딩을 통해 원하는 언어로 OpenTelemetry 0.7.0 메시지를 구문 분석할 수 있습니다.

언어별 바인딩을 가져오려면

- 단계는 원하는 언어에 따라 다릅니다.
  - Java를 사용하려면 Java 프로젝트에 다음 Maven 종속 항목을 추가합니다. OpenTelemetry Java >> 0.14.1.
  - 다른 언어를 사용하려면 다음 단계를 따릅니다.

    a. 클래스 생성에서 목록을 확인하여 언어가 지원되는지 확인합니다.

    b. Protocol Buffers 다운로드의 단계에 따라 Protobuf 컴파일러를 설치합니다.

    c. v0.7.0 릴리스에서 OpenTelemetry 0.7.0 ProtoBuf 정의를 다운로드합니다.

    d. 다운로드한 OpenTelemetry 0.7.0 ProtoBuf 정의의 루트 폴더에 있는지 확인합니다. 그런 다음에 `src` 폴더를 생성한 후 명령을 실행하여 언어별 바인딩을 생성합니다. 자세한 내용은 클래스 생성을 참조하세요.

    다음은 Javascript 바인딩을 생성하는 방법의 예입니다.

```
protoc --proto_path=./ --js_out=import_style=commonjs,binary:src \
opentelemetry/proto/common/v1/common.proto \
opentelemetry/proto/resource/v1/resource.proto \
```

```
opentelemetry/proto/metrics/v1/metrics.proto \
opentelemetry/proto/collector/metrics/v1/metrics_service.proto
```

다음 단원에는 이전 지침을 사용하여 구축할 수 있는 언어별 바인딩을 사용하는 예가 나와 있습니다.

Java

```
package com.example;

import io.opentelemetry.proto.collector.metrics.v1.ExportMetricsServiceRequest;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class MyOpenTelemetryParser {

    public List<ExportMetricsServiceRequest> parse(InputStream inputStream) throws
 IOException {
        List<ExportMetricsServiceRequest> result = new ArrayList<>();

        ExportMetricsServiceRequest request;
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        while ((request =
 ExportMetricsServiceRequest.parseDelimitedFrom(inputStream)) != null) {
            // Do whatever we want with the parsed message
            result.add(request);
        }

        return result;
    }
}
```

Javascript

이 예에서는 생성한 바인딩이 있는 루트 폴더가 `./`라고 가정합니다.

`parseRecord` 함수의 데이터 인수는 다음 유형 중 하나일 수 있습니다.

- `Uint8Array` - 이 유형이 최적입니다.
- `Buffer`- 이 유형은 노드에서 최적입니다.
- `Array.number` - 8비트 정수입니다.

```
const pb = require('google-protobuf')
const pbMetrics =
    require('./opentelemetry/proto/collector/metrics/v1/metrics_service_pb')

function parseRecord(data) {
    const result = []

    // Loop until we've read all the data from the buffer
    while (data.length) {
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ --------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ --------------------------- ------ ----------------------
         */
        const reader = new pb.BinaryReader(data)
        const messageLength = reader.decoder_.readUnsignedVarint32()
        const messageFrom = reader.decoder_.cursor_
        const messageTo = messageFrom + messageLength

        // Extract the current `ExportMetricsServiceRequest` message to parse
        const message = data.subarray(messageFrom, messageTo)

        // Parse the current message using the ProtoBuf library
        const parsed =
            pbMetrics.ExportMetricsServiceRequest.deserializeBinary(message)

        // Do whatever we want with the parsed message
        result.push(parsed.toObject())

        // Shrink the remaining buffer, removing the already parsed data
        data = data.subarray(messageTo)
    }
```

```
    return result
}
```

Python

var-int 구분 기호를 직접 읽거나 내부 메서드 _VarintBytes(size) 및 _DecodeVarint32(buffer, position)를 사용해야 합니다. 이러한 메서드는 size 바이트 바로 뒤에 있는 버퍼의 위치를 반환합니다. 읽기 측에서는 메시지의 바이트만 읽는 것으로 제한되는 새 버퍼를 생성합니다.

```
size = my_metric.ByteSize()
f.write(_VarintBytes(size))
f.write(my_metric.SerializeToString())
msg_len, new_pos = _DecodeVarint32(buf, 0)
msg_buf = buf[new_pos:new_pos+msg_len]
request = metrics_service_pb.ExportMetricsServiceRequest()
request.ParseFromString(msg_buf)
```

Go

Buffer.DecodeMessage()를 사용합니다.

C#

CodedInputStream를 사용합니다. 이 클래스는 크기로 구분된 메시지를 읽을 수 있습니다.

C++

google/protobuf/util/delimited_message_util.h에 기술된 함수는 크기로 구분된 메시지를 읽을 수 있습니다.

기타 언어

기타 언어의 경우 Protocol Buffers 다운로드를 참조하세요.

구문 분석기를 구현할 때 Kinesis 레코드에 여러 ExportMetricsServiceRequest Protocol Buffers 메시지가 포함될 수 있다는 점을 고려하세요. 각 메시지는 바이트 단위의 레코드 길이를 나타내는 UnsignedVarInt32가 있는 헤더로 시작합니다.

문제 해결

최종 대상에서 지표 데이터가 표시되지 않는 경우 다음을 확인하세요.

- 지표 스트림이 실행 중 상태인지 확인합니다. CloudWatch 콘솔을 사용하여 이를 수행하는 방법에 대한 절차는 지표 스트림 작업 및 유지 관리 단원을 참조하세요.
- 이전에 2일을 초과하여 게시된 지표는 스트리밍되지 않습니다. 특정 지표의 스트리밍 여부를 결정하려면 CloudWatch 콘솔에서 지표를 그래프로 표시하고 최종 표시되는 데이터 포인트가 얼마나 오래되었는지 확인합니다. 2일을 초과한 경우 지표 스트림에서 선택하지 않습니다.
- 지표 스트림에서 내보낸 지표를 확인합니다. CloudWatch 콘솔의 [지표(Metrics)]에서 [MetricUpdate], [TotalMetricUpdate] [PublishErrorRate] 지표의 [AWS/CloudWatch/MetricStreams] 네임스페이스를 검토합니다.
- PublishErrorRate 지표가 높은 경우 Firehose 전송 스트림이 사용하는 대상이 있는지 그리고 지표 스트림의 구성에 지정된 IAM 역할이 `CloudWatch` 서비스 보안 주체에 쓰기 권한을 부여하는지 확인합니다. 자세한 내용은 CloudWatch와 Firehose 간의 신뢰 단원을 참조하십시오.
- Firehose 전송 스트림에 최종 대상에 대한 쓰기 권한이 있는지 확인합니다.
- Firehose 콘솔에서 지표 스트림에 사용되는 Firehose 전송 스트림을 살펴보고 모니터링 탭을 확인하여 Firehose 전송 스트림이 데이터를 수신하고 있는지 확인합니다.
- Firehose 전송 스트림을 올바른 세부 정보로 구성했는지 확인합니다.
- Firehose 전송 스트림이 쓰는 최종 대상에 대해 사용 가능한 로그 또는 지표를 확인합니다.
- 더 자세한 정보를 얻으려면 Firehose 전송 스트림에서 CloudWatch Logs 오류 로깅을 사용 설정합니다. 자세한 내용은 CloudWatch Logs를 사용하여 Amazon Data Firehose 모니터링 섹션을 참조하세요.

사용 가능한 지표 보기

지표는 먼저 네임스페이스별로 그룹화된 다음, 각 네임스페이스 내에서 다양한 측정기준 조합별로 그룹화됩니다. 예를 들어 모든 EC2 지표, 인스턴스별로 그룹화된 EC2 지표 또는 Auto Scaling 그룹별로 그룹화된 EC2 지표를 볼 수 있습니다.

사용 중인 AWS 서비스만 Amazon CloudWatch에 지표를 전송합니다.

CloudWatch에 지표를 전송하는 AWS 서비스 목록은 CloudWatch 지표를 게시하는 AWS 서비스 단원을 참조하세요. 이 페이지에서는 각 서비스에 의해 게시된 지표 및 측정기준도 볼 수 있습니다.

> **Note**
>
> 지난 2주 동안 새로운 데이터 요소가 없는 지표는 콘솔에 나타나지 않습니다. 콘솔의 모든 지표 탭에 있는 검색 상자에 지표 이름이나 측정기준 이름을 입력할 때도 나타나지 않으며 list-

metrics 명령의 결과에도 반환되지 않습니다. 이러한 지표를 검색하는 가장 좋은 방법은 AWS CLI에서 get-metric-data 또는 get-metric-statistics 명령을 사용하는 것입니다.
확인할 이전 지표에 유사한 측정기준이 있는 현재 지표가 있는 경우, 현재 유사한 지표를 확인한 다음 소스 탭을 선택하고 지표 이름 및 측정기준 필드를 원하는 것으로 변경하고 시간 범위를 지표가 보고된 시간으로 변경합니다.

다음 단계에 따라 지표 네임스페이스를 탐색하여 지표를 찾고 볼 수 있습니다. 표적 검색어를 사용하여 지표를 검색할 수도 있습니다. 자세한 내용은 사용 가능한 지표 검색 단원을 참조하십시오.

CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정을 탐색하는 경우 이 모니터링 계정에 연결된 소스 계정의 지표를 볼 수 있습니다. 소스 계정의 지표가 표시되면 해당 계정의 ID 또는 레이블도 표시됩니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

콘솔을 사용하여 네임스페이스와 측정기준별로 사용 가능한 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 모든 지표를 선택합니다.
3. 지표 네임스페이스(예: EC2 또는 Lambda)를 선택합니다.
4. 지표 차원(예: Per-Instance Metrics(인스턴스별 지표) 또는 By Function Name(함수 이름별))을 선택합니다.
5. Browse(찾아보기) 탭에 네임스페이스의 해당 차원에 대한 모든 지표가 표시됩니다. 각 지표 이름에는 지표 정의가 포함된 팝업이 표시되도록 선택할 수 있는 정보 버튼이 있습니다.

   CloudWatch 크로스 계정 관측성의 모니터링 계정인 경우 이 모니터링 계정에 연결된 소스 계정의 지표도 볼 수 있습니다. 테이블의 Account label(계정 레이블) 및 Account id(계정 ID) 열에는 각 지표가 속한 계정이 표시됩니다.

   다음을 수행할 수 있습니다.

   a. 테이블을 정렬하려면 열 머리글을 사용합니다.
   b. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.
   c. 계정별로 필터링하려면 계정 레이블 또는 계정 ID를 선택한 다음 Add to search(검색에 추가)를 선택합니다.
   d. 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.
   e. 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.

6. (선택 사항) 이 그래프를 CloudWatch 대시보드에 추가하려면 [작업(Actions)], [대시보드에 추가(Add to dashboard)]를 선택합니다.

AWS CLI를 사용하여 계정 네임스페이스, 차원 또는 지표별로 사용 가능한 지표를 보려면 다음을 수행하세요.

list-metrics 명령을 사용하여 CloudWatch 지표를 나열합니다. 지표를 게시하는 모든 서비스의 네임스페이스, 지표 및 측정기준 목록을 보려면 CloudWatch 지표를 게시하는 AWS 서비스 단원을 참조하세요.

다음 예시 명령은 Amazon EC2에 대한 모든 지표를 나열합니다.

```
aws cloudwatch list-metrics --namespace AWS/EC2
```

출력의 예제는 다음과 같습니다.

```
{
  "Metrics" : [
    ...
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "NetworkOut"
    },
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "CPUUtilization"
    },
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
```

```
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "NetworkIn"
    },
    ...
  ]
}
```

지정된 리소스에서 사용 가능한 모든 지표를 나열하려면

다음 예제는 지정한 인스턴스의 결과만 보도록 AWS/EC2 네임스페이스와 InstanceId 측정기준을 지정합니다.

```
aws cloudwatch list-metrics --namespace AWS/EC2 --dimensions
 Name=InstanceId,Value=i-1234567890abcdef0
```

모든 리소스에 대한 지표를 나열하려면

다음 예제는 지정한 지표의 결과만 보도록 AWS/EC2 네임스페이스와 지표 이름을 지정합니다.

```
aws cloudwatch list-metrics --namespace AWS/EC2 --metric-name CPUUtilization
```

연결된 소스 계정에서 CloudWatch 크로스 계정 관측성의 지표를 검색하려면 다음을 수행하세요.

다음 예제에서는 모니터링 계정에서 실행되어 모니터링 계정과 연결된 모든 소스 계정 모두에서 지표를 검색합니다. --include-linked-accounts를 추가하지 않으면 명령이 모니터링 계정의 지표만 반환합니다.

```
aws cloudwatch list-metrics --include-linked-accounts
```

소스 계정에서 CloudWatch 크로스 계정 관측성의 지표를 검색하려면 다음을 수행하세요.

다음 예제에서는 모니터링 계정에서 실행되어 ID가 111122223333인 소스 계정에서 지표를 검색합니다.

```
aws cloudwatch list-metrics --include-linked-accounts --owning-account "111122223333"
```

사용 가능한 지표 검색

대상으로 지정된 검색어를 사용하여 계정의 모든 지표 중에서 검색할 수 있습니다. 네임스페이스, 지표 이름 또는 측정기준 내에서 결과가 일치하는 지표가 반환됩니다.

CloudWatch 크로스 계정 관측성의 모니터링 계정인 경우 이 모니터링 계정에 연결된 소스 계정의 지표를 검색할 수 있습니다.

> **Note**
>
> 지난 2주 동안 새로운 데이터 요소가 없는 지표는 콘솔에 나타나지 않습니다. 콘솔의 모든 지표 탭에 있는 검색 상자에 지표 이름이나 측정기준 이름을 입력할 때도 나타나지 않으며 list-metrics 명령의 결과에도 반환되지 않습니다. 이러한 지표를 검색하는 가장 좋은 방법은 AWS CLI에서 get-metric-data 또는 get-metric-statistics 명령을 사용하는 것입니다.

CloudWatch에서 사용 가능한 지표를 검색하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. All metrics(모든 지표) 탭의 검색 필드에 검색어(예: 지표 이름, 네임스페이스, 계정 ID, 계정 레이블, 차원 이름이나 값 또는 리소스 이름)를 입력합니다. 이렇게 하면 이 검색어가 포함된 지표가 있는 모든 네임스페이스가 표시됩니다.

   예를 들어 **`volume`**을 검색하면 이름에 이 단어가 있는 지표가 포함된 네임스페이스가 표시됩니다.

   검색에 대한 자세한 내용은 그래프에서 검색 표현식 사용 단원을 참조하세요.
4. 모든 검색 결과를 그래프로 작성하려면 Graph search(그래프 검색)를 선택합니다.

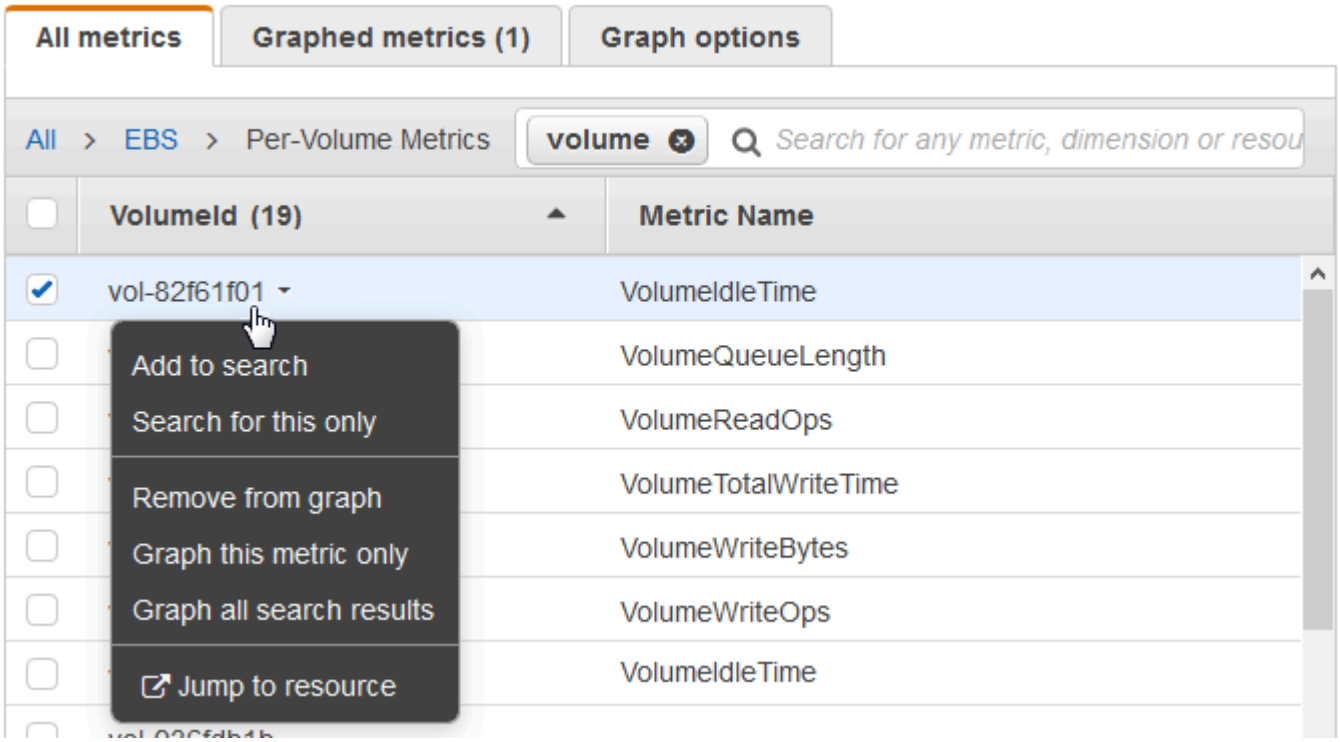
   또는

   해당 네임스페이스에서 지표를 볼 네임스페이스를 선택합니다. 그러면 다음 작업을 수행할 수 있습니다.

   a. 하나 이상의 지표를 그래프 처리하려면 각 지표 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.
   b. 검색을 구체화하려면 지표 이름 위로 마우스를 이동하고 검색에 추가 또는 이 항목만 검색을 선택합니다.
   c. 콘솔에서 리소스 중 하나를 보려면 리소스 ID를 선택하고 리소스로 이동을 선택합니다.

d. 지표에 대한 도움말을 보려면 지표 이름을 선택하고 이것은 무엇입니까?를 선택합니다.

선택한 지표가 그래프에 표시됩니다.



5. (선택 사항) 검색어의 해당 부분을 편집하려면 검색 표시줄에 있는 버튼 중 하나를 선택합니다.

지표 그래프 작성

CloudWatch 콘솔을 사용하여 다른 AWS 서비스에서 생성한 지표 데이터를 그래프로 표시할 수 있습니다. 이렇게 하면 서비스의 지표 활동을 보다 효율적으로 확인할 수 있습니다. 다음 절차에서는 CloudWatch에서 지표를 그래프로 표시하는 방법을 설명합니다.

내용

- 그래프의 지표에서 경보 생성

지표 그래프 작성

CloudWatch 콘솔을 사용하여 지표를 선택하고 지표 데이터의 그래프를 생성할 수 있습니다.

CloudWatch는 지표에 대해 `Average`, `Minimum`, `Maximum`, `Sum`, `SampleCount` 등의 통계를 지원합니다. 자세한 내용은 Statistics 단원을 참조하십시오.

다양한 세부 수준에서 데이터를 볼 수 있습니다. 예를 들어, 문제 해결 시 유용할 수 있는 1분 보기를 선택할 수 있습니다. 또는 세부적이지 않은 1시간 보기를 선택할 수 있습니다. 이 보기는 시간 경과에 따른 추세를 볼 수 있도록 넓은 시간 범위(예: 3일)를 볼 때 유용할 수 있습니다. 자세한 내용은 기간 단원을 참조하십시오.

CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정을 사용하는 경우 이 모니터링 계정에 연결된 소스 계정의 지표를 그래프로 만들 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

그래프 생성

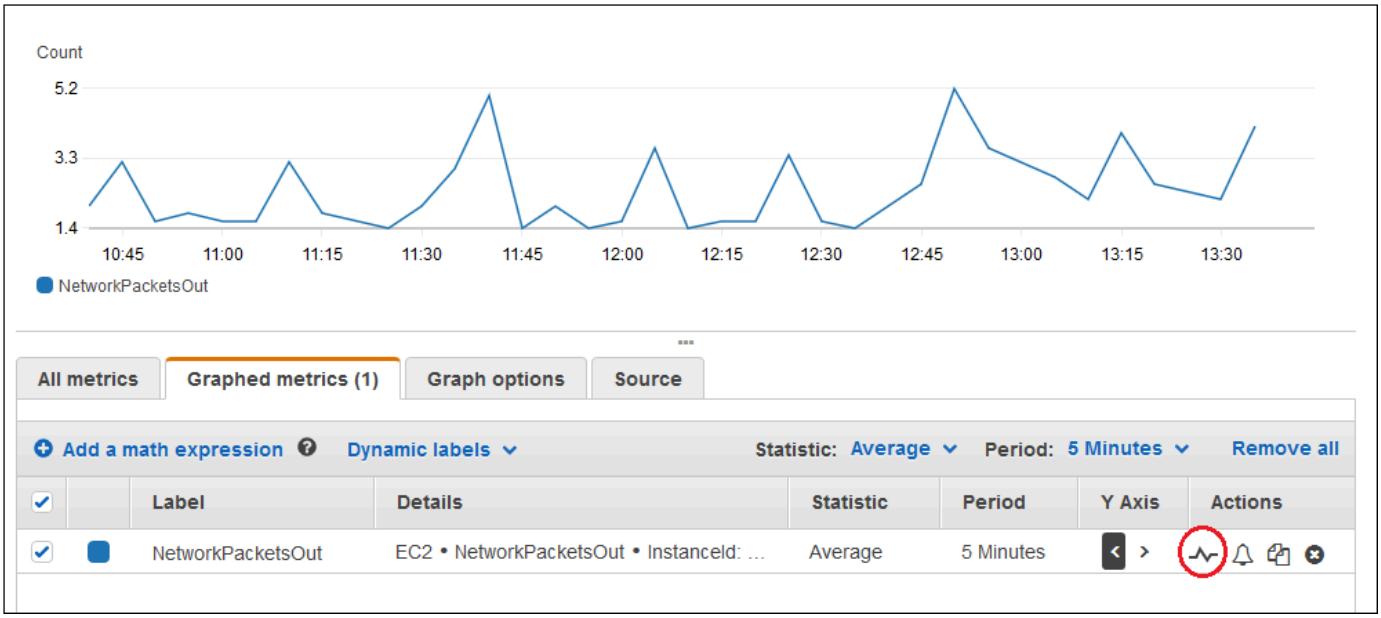
지표 그래프를 작성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 모든 지표를 선택합니다.
3. 찾아보기 탭의 검색 필드에 지표 이름, 계정 ID 또는 리소스 이름 등의 검색어를 입력합니다.

   예를 들어, `CPUUtilization` 지표를 검색하면 이 지표와 함께 네임스페이스와 측정기준이 표시됩니다.
4. 지표 검색을 위해 결과 중 하나를 선택합니다.
5. 하나 이상의 지표를 그래프 처리하려면 각 지표 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.
6. (선택 사항) 그래프 유형을 변경하려면 옵션 탭을 선택합니다. 그런 다음 선 그래프, 누적 영역 차트, 숫자 표시, 게이지, 막대형 차트 또는 파이 차트 중에서 선택할 수 있습니다.
7. 그래프로 표시된 지표 탭을 선택합니다.
8. (선택 사항) 그래프에 사용된 통계를 변경하려면 지표 이름 옆의 [통계(Statistic)] 열에서 새 통계를 선택합니다.

CloudWatch 통계에 대한 자세한 내용은 CloudWatch 통계 정의 단원을 참조하세요. pxx 백분위수 통계에 대한 자세한 내용은 백분위수 단원을 참조하세요.

9. (선택 사항) 지표에 대한 예상 값을 보여주는 이상 탐지 밴드를 추가하려면, 지표 옆에 있는 작업에서 이상 탐지 아이콘을 선택합니다.



CloudWatch는 지표의 최근(최대 2주 동안) 기록 데이터를 사용하여 예상 값에 대한 모델을 계산합니다. 그런 다음, 이 예상 값 범위를 그래프에 밴드로 표시합니다. CloudWatch는 지표에 [ANOMALY_DETECTION_BAND]라는 레이블이 지정된 새 행을 추가하여 이상 탐지 밴드 수학 표현식을 표시합니다. 최근 과거 데이터가 있으면 모델에서 생성된 이상 탐지 밴드의 근사치인 미리 보기 이상 탐지 밴드가 즉시 표시됩니다. 실제 이상 탐지 밴드가 표시되기까지 최대 15분이 걸립니다.

기본적으로 CloudWatch는 밴드 임계값의 기본값으로 2를 사용하여 예상 값 밴드의 상한 및 하한을 생성합니다. 이 숫자를 변경하려면, 밴드의 세부정보에서 표현식 끝에 있는 값을 변경합니다.

- (선택 사항) 이상 탐지 모델 계산 방법을 변경하려면 Edit model(모델 편집)을 선택합니다. 모델 계산을 위해 교육에서 과거 및 미래 기간을 사용하지 않도록 제외할 수 있습니다. 시스템 중단, 배포 및 휴일과 같은 비정상적인 이벤트 시스템을 교육 데이터에서 제외하는 것이 중요합니다. 일광 절약 시간제 변경을 위해 모델에 사용할 시간대를 지정할 수도 있습니다.

자세한 내용은 CloudWatch 이상 탐지 사용 단원을 참조하십시오.

그래프에서 모델을 숨기려면 ANOMALY_DETECTION_BAND 기능이 있는 선에서 체크 표시를 제거하거나 X 아이콘을 선택하세요. 모델을 완전히 삭제하려면 Edit model(모델 편집), Delete model(모델 삭제)을 선택합니다.

10. (선택 사항) 그래프로 표시할 지표를 선택할 때 각 지표의 그래프 범례에 표시할 동적 레이블을 지정합니다. 동적 레이블은 지표에 대한 통계를 표시하고 대시보드 또는 그래프를 새로 고칠 때 자동으로 업데이트됩니다. 동적 레이블을 추가하려면 그래프로 표시된 지표, 동적 레이블 추가를 선택합니다.

    기본적으로 레이블에 추가하는 동적 값은 레이블 시작 부분에 나타납니다. 그런 다음 지표에 대한 레이블 값을 선택하여 레이블을 편집할 수 있습니다. 자세한 내용은 동적 레이블 사용 단원을 참조하십시오.

11. 그래프로 표시하려는 지표에 대한 자세한 정보를 보려면 범례에 마우스 포인터를 둡니다.

12. 가로 주석을 사용하면 그래프 사용자가 지표가 특정 레벨까지 급상승하는지 아니면 지표가 사전 정의된 범위 내에 있는지 여부를 보다 효율적으로 확인할 수 있습니다. 가로 주석을 추가하려면 옵션 탭을 선택한 다음 가로 주석 추가를 선택합니다.

    a. 레이블에 주석의 레이블을 입력합니다.

    b. 값에 가로 주석이 표시될 지표 값을 입력합니다.

    c. Fill(채우기)에서 이 주석에 채우기 셰이딩을 사용할지 여부를 지정합니다. 예를 들어 채울 영역에 대해 Above 또는 Below를 선택합니다. Between을 지정할 경우 다른 Value 필드가 표시되며, 두 값 사이의 그래프 영역이 채워집니다.

    d. 그래프에 여러 지표가 포함된 경우 축에서 Value의 숫자가 왼쪽 Y축과 연결된 지표를 참조하는지 아니면 오른쪽 Y축과 연결된 지표를 참조하는지를 지정합니다.

    주석의 왼쪽 열에서 색상 정사각형을 선택하여 주석의 채우기 색상을 변경할 수 있습니다.

    동일한 그래프에 여러 가로 주석을 추가하려면 이들 단계를 반복합니다.

    주석을 숨기려면 해당 주석의 왼쪽 열에서 확인란을 선택 취소합니다.

    주석을 삭제하려면 [작업(Actions)] 열에서 [x]를 선택합니다.

13. 그래프에 대한 URL을 얻으려면 작업과 공유를 선택합니다. 저장하거나 공유할 URL을 복사합니다.

14. 대시보드에 그래프를 추가하려면 작업과 대시보드에 추가를 선택합니다.

다른 데이터 소스의 지표 그래프 생성

CloudWatch 이외의 데이터 소스의 리소스를 표시하는 그래프를 생성할 수 있습니다. 이러한 다른 데이터 소스에 대한 연결을 생성하는 방법에 대한 자세한 내용은 다른 데이터 소스의 쿼리 지표 섹션을 참조하세요.

그래프로 다른 데이터 소스의 지표 표시

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 모든 지표를 선택합니다.
3. 다중 소스 쿼리 탭을 선택합니다.
4. 데이터 소스에서 사용할 데이터 소스를 선택합니다.

   원하는 데이터 소스에 대한 연결을 아직 만들지 않은 경우 데이터 소스 생성 및 관리를 선택한 다음, 데이터 소스 생성 및 관리를 선택합니다. 이 데이터 소스 생성 프로세스의 나머지 부분에 대한 자세한 내용은 마법사로 사전 구축된 데이터 소스에 연결 섹션을 참조하세요.
5. 마법사나 쿼리 편집기는 쿼리에 필요한 정보를 입력하라는 메시지를 표시합니다. 워크플로는 각 데이터 소스마다 다르며 데이터 소스에 맞게 조정됩니다. 예를 들어, Amazon Managed Service for Prometheus 및 Prometheus 데이터 소스의 경우 쿼리 도우미가 있는 PromQL 쿼리 편집기 상자가 나타납니다.
6. 쿼리 구성을 마치면 쿼리 그래프로 표시를 선택합니다.

   그래프는 쿼리의 지표로 채워집니다.
7. (선택 사항) 가로 주석을 사용하면 그래프 사용자가 지표가 특정 레벨까지 급상승하는지 아니면 지표가 사전 정의된 범위 내에 있는지 여부를 보다 효율적으로 확인할 수 있습니다. 가로 주석을 추가하려면 옵션 탭을 선택한 다음 가로 주석 추가를 선택합니다.

   a. 레이블에 주석의 레이블을 입력합니다.

   b. 값에 가로 주석이 표시될 지표 값을 입력합니다.

   c. Fill(채우기)에서 이 주석에 채우기 셰이딩을 사용할지 여부를 지정합니다. 예를 들어 채울 영역에 대해 `Above` 또는 `Below`를 선택합니다. `Between`을 지정할 경우 다른 `Value` 필드가 표시되며, 두 값 사이의 그래프 영역이 채워집니다.

   d. 그래프에 여러 지표가 포함된 경우 축에서 `Value`의 숫자가 왼쪽 Y축과 연결된 지표를 참조하는지 아니면 오른쪽 Y축과 연결된 지표를 참조하는지를 지정합니다.

      주석의 왼쪽 열에서 색상 정사각형을 선택하여 주석의 채우기 색상을 변경할 수 있습니다.

동일한 그래프에 여러 가로 주석을 추가하려면 이들 단계를 반복합니다.

주석을 숨기려면 해당 주석의 왼쪽 열에서 확인란을 선택 취소합니다.

주석을 삭제하려면 [작업(Actions)] 열에서 [x]를 선택합니다.

8. (선택 사항) 이 그래프를 대시보드에 추가하려면 작업, 대시보드에 추가를 선택합니다.

그래프 업데이트

그래프를 업데이트하려면

1. 그래프 이름을 변경하려면 연필 아이콘을 선택합니다.
2. 시간 범위를 변경하려면 제공되는 값 중 하나를 선택하거나 맞춤을 선택합니다. 자세한 내용은 그래프의 시간 범위 또는 표준 시간대 형식 수정 단원을 참조하십시오.
3. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 통계 또는 사전 정의된 백분위수 중 하나를 선택하거나 사용자 지정 백분위수(예: **p95.45**)를 지정합니다.
4. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 다른 값을 선택합니다.
5. 가로 주석을 추가하려면 그래프 옵션을 선택한 후 가로 주석 추가를 선택합니다.

   a. 레이블에 주석의 레이블을 입력합니다.

   b. 값에 가로 주석이 표시될 지표 값을 입력합니다.

   c. Fill(채우기)에서 이 주석에 채우기 셰이딩을 사용할지 여부를 지정합니다. 예를 들어 채울 영역에 대해 `Above` 또는 `Below`를 선택합니다. `Between`을 지정할 경우 다른 `Value` 필드가 표시되며, 두 값 사이의 그래프 영역이 채워집니다.

   d. 그래프에 여러 지표가 포함된 경우 축에서 `Value`의 숫자가 왼쪽 Y축과 연결된 지표를 참조하는지 아니면 오른쪽 Y축과 연결된 지표를 참조하는지를 지정합니다.

      주석의 왼쪽 열에서 색상 정사각형을 선택하여 주석의 채우기 색상을 변경할 수 있습니다.

   동일한 그래프에 여러 가로 주석을 추가하려면 이들 단계를 반복합니다.

   주석을 숨기려면 해당 주석의 왼쪽 열에서 확인란을 선택 취소합니다.

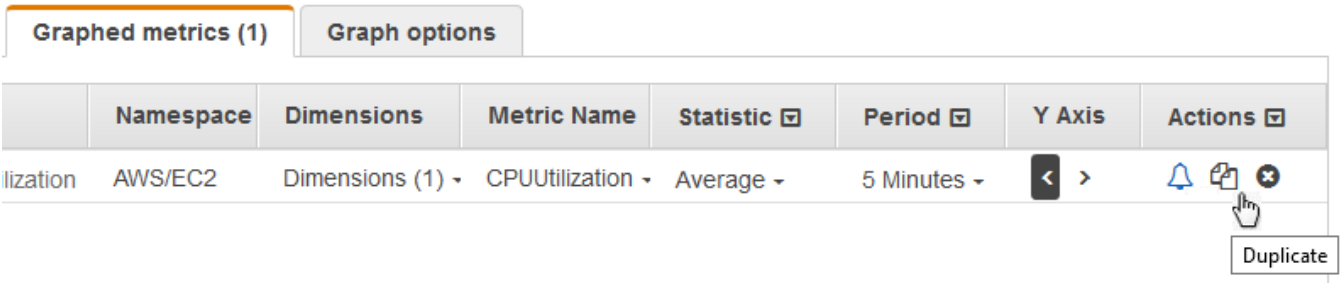
주석을 삭제하려면 [작업(Actions)] 열에서 [x]를 선택합니다.

6. 새로 고침 간격을 변경하려면 새로 고침 옵션을 선택한 후 자동 새로 고침을 선택하거나 1분, 2분, 5분 또는 15분을 선택합니다.

지표 복제

지표를 복제하려면

1. 그래프로 표시된 지표 탭을 선택합니다.
2. 작업에서 복제 아이콘을 선택합니다.



3. 필요에 따라 복제 지표를 업데이트합니다.

두 그래프를 하나로 병합

두 개의 서로 다른 그래프를 하나로 병합하면 결과 그래프에 두 지표가 모두 표시됩니다. 이미 다른 그래프에 서로 다른 지표가 표시되어 있는데 이를 결합하거나 다른 리전의 지표로 단일 그래프를 쉽게 만들려는 경우 유용할 수 있습니다.

그래프를 다른 그래프에 병합하려면 병합하려는 그래프의 URL 또는 JSON 소스를 사용합니다.

두 그래프를 하나로 병합하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 다른 그래프에 병합하려는 그래프를 엽니다. 이렇게 하려면 지표, 모든 지표를 선택한 다음 그래프로 표시할 지표를 선택하면 됩니다. 또는 대시보드를 연 다음 그래프를 선택하고 그래프 오른쪽 상단의 메뉴에서 지표에서 열기를 선택하여 대시보드의 그래프 중 하나를 열 수 있습니다.
3. 그래프를 연 후 다음 중 하나를 수행합니다.
   - 브라우저 표시줄에서 URL을 복사합니다.
   - 소스 탭을 선택한 다음 복사를 선택합니다.

4. 이전 그래프를 병합하려는 그래프를 엽니다.
5. 지표 보기에서 두 번째 그래프를 연 다음 작업, 그래프 병합을 선택합니다.
6. 이전에 복사한 URL 또는 JSON을 입력하고 병합을 선택합니다.
7. 병합된 그래프가 나타납니다. 왼쪽의 y축은 원래 그래프이고, 오른쪽의 y축은 병합한 그래프에 대한 것입니다.

> **Note**
>
> 병합한 그래프가 METRICS() 함수를 사용하는 경우 병합된 그래프의 지표는 병합된 그래프의 METRICS() 계산에 포함되지 않습니다.

8. 병합된 그래프를 대시보드에 저장하려면 작업, 대시보드에 추가를 선택합니다.

동적 레이블 사용

그래프에서 동적 레이블을 사용할 수 있습니다. 동적 레이블은 선택한 지표의 레이블에 동적으로 업데이트된 값을 추가합니다. 아래에 나와 있는 표와 같이 레이블에 다양한 값을 추가할 수 있습니다.

레이블에 표시된 동적 값은 현재 그래프에 표시된 시간 범위에서 파생됩니다. 레이블의 동적 부분은 대시보드 또는 그래프를 새로 고칠 때 자동으로 업데이트됩니다.

검색 표현식에서 동적 레이블을 사용하면 검색에서 반환된 모든 지표에 동적 레이블이 적용됩니다.

CloudWatch 콘솔을 사용하여 레이블에 동적 값을 추가하고 레이블을 편집하며 레이블 열 내에서 동적 값의 위치를 변경하고 다른 사용자 지정 작업을 수행할 수 있습니다.

동적 레이블

동적 레이블 내에서 다음과 같이 지표의 속성과 관련된 값을 사용할 수 있습니다.

동적 레이블 라이브 값	설명
${AVG}	현재 그래프에 표시된 시간 범위의 평균 값
${DATAPOINT_COUNT}	현재 그래프에 표시된 시간 범위의 데이터 요소 수입니다.
${FIRST}	현재 그래프에 표시된 시간 범위에서 가장 오래된 지표 값입니다.

동적 레이블 라이브 값	설명
${FIRST_LAST_RANGE}	현재 그래프에 표시된 가장 오래된 데이터 요소와 가장 최근 데이터 요소의 지표 값 간 차이입니다.
${FIRST_LAST_TIME_RANGE}	현재 그래프에 표시된 가장 오래된 데이터 요소와 가장 최근 데이터 요소 사이의 절대 시간 범위입니다.
${FIRST_TIME}	현재 그래프에 표시된 시간 범위에서 가장 오래된 데이터 요소의 타임스탬프입니다.
${FIRST_TIME_RELATIVE}	현재 그래프에 표시된 시간 범위에서 가장 오래된 데이터 요소의 타임스탬프와 현재 사이의 절대 시간 차이입니다.
${LABEL}	지표에 대한 기본 레이블의 표현입니다.
${LAST}	현재 그래프에 표시된 시간 범위에서 가장 최근 지표 값입니다.
${LAST_TIME}	현재 그래프에 표시된 시간 범위에서 가장 최근 데이터 요소의 타임스탬프입니다.
${LAST_TIME_RELATIVE}	현재 그래프에 표시된 시간 범위에서 가장 최근 데이터 요소의 타임스탬프와 현재 사이의 절대 시간 차이입니다.
${MAX}	현재 그래프에 표시된 시간 범위의 최대값
${MAX_TIME}	현재 그래프에 표시된 데이터 요소 중 지표 값이 가장 높은 데이터 요소의 타임스탬프입니다.
${MAX_TIME_RELATIVE}	현재 그래프에 표시된 해당 데이터 요소 중 값이 가장 높은 데이터 요소의 타임스탬프와 현재 사이의 절대 시간 차이입니다.
${MIN}	현재 그래프에 표시된 시간 범위의 최소값
${MIN_MAX_RANGE}	현재 그래프에 표시된 해당 데이터 요소 중 지표 값이 가장 높은 데이터 요소와 지표 값이 가장 낮은 데이터 요소 간 지표 값 차이입니다.

동적 레이블 라이브 값	설명
${MIN_MAX_TIME_RANGE}	현재 그래프에 표시된 해당 데이터 요소 중 지표 값이 가장 높은 데이터 요소와 지표 값이 가장 낮은 데이터 요소 사이의 절대 시간 범위입니다.
${MIN_TIME}	현재 그래프에 표시된 데이터 요소 중 지표 값이 가장 낮은 데이터 요소의 타임스탬프입니다.
${MIN_TIME_RELATIVE}	현재 그래프에 표시된 해당 데이터 요소 중 값이 가장 낮은 데이터 요소의 타임스탬프와 현재 사이의 절대 시간 차이입니다.
${PROP('AccountId')}	지표의 AWS 계정 ID입니다.
${PROP('AccountLabel')}	CloudWatch 크로스 계정 관측성에서 이 지표를 소유하는 소스 계정에 대해 지정된 레이블입니다.
${PROP('Dim.*dimension_name*')}	지정된 측정기준의 값입니다. *dimension_name* 을 차원의 대/소문자를 구분하는 이름으로 바꿉니다.
${PROP('MetricName')}	지표의 이름.
${PROP('Namespace')}	지표의 네임스페이스입니다.
${PROP('Period')}	지표의 기간(초)입니다.
${PROP('Region')}	지표가 게시된 AWS 리전입니다.
${PROP('Stat')}	그래프로 표시되고 있는 지표 통계입니다.
${SUM}	현재 그래프에 표시된 시간 범위 값의 합계

예를 들어 각 Lambda 함수의 `Errors`를 찾는 검색 표현식 **`SEARCH(' {AWS/Lambda, FunctionName} Errors ', 'Sum')`**가 있다고 가정합니다. 레이블을 `[max: ${MAX} Errors for Function Name ${LABEL}]`로 설정하면 각 지표의 레이블은 [max: ***number*** Errors for Function Name ***Name***]이 됩니다.

레이블에 최대 6개의 동적 값을 추가할 수 있습니다. 각 레이블 내에서 `${LABEL}` 자리 표시자를 한 번만 사용할 수 있습니다.

그래프의 시간 범위 또는 표준 시간대 형식 수정

이 단원에서는 CloudWatch 지표 그래프에서 날짜, 시간 및 시간대 형식을 수정하는 방법을 설명합니다. 그래프를 확대하여 특정 시간 범위를 적용하는 방법도 설명합니다. 그래프 생성에 대한 자세한 내용은 지표 그래프 작성 섹션을 참조하세요.

> **Note**
>
> 대시보드의 시간 범위가 대시보드의 그래프에 사용된 기간보다 짧으면 다음과 같은 상황이 발생합니다.
>
> - 대시보드의 시간 범위보다 길더라도 해당 위젯의 전체 기간 1개에 해당하는 데이터의 양을 표시하도록 그래프가 수정됩니다. 이렇게 하면 그래프에 데이터 포인트가 1개 이상 있을 수 있습니다.
> - 이 데이터 포인트의 기간에서 시작 시간은 적어도 하나의 데이터 포인트가 표시될 수 있도록 역방향으로 조정됩니다.

상대적 시간 범위 설정

New interface

그래프의 상대적 시간 범위를 설정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다. 화면 오른쪽 상단 모서리에서 1시간부터 1주까지 사전 정의된 시간 범위 중 하나를 선택할 수 있습니다(1시간, 3시간, 12시간, 1일, 3일 또는 1주). 또는 사용자 지정(Custom)을 선택하여 나만의 시간 범위를 설정할 수 있습니다.
3. 사용자 지정(Custom)을 선택한 다음 상자의 왼쪽 상단 모서리에 있는 상대(Relative) 탭을 선택합니다. 시간 범위를 분, 시간, 일, 주, 월로 지정할 수 있습니다.
4. 시간 범위를 지정한 후 적용(Apply)을 선택합니다.

Original interface

그래프의 상대적 시간 범위를 설정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다. 화면 오른쪽 상단 모서리에서 1시간부터 1주까지 사전 정의된 시간 범위 중 하나를 선택할 수 있습니다(1시간, 3시간, 12시간, 1일, 3일 또는 1주). 또는 사용자 지정(custom)을 선택하여 나만의 시간 범위를 설정할 수 있습니다.
3. 사용자 지정(custom)을 선택한 다음 상자의 왼쪽 상단 모서리에 있는 상대(Relative) 탭을 선택합니다. 시간 범위를 분, 시간, 일, 주, 월로 지정할 수 있습니다.

절대적 시간 범위 설정

New interface

그래프의 절대적 시간 범위를 설정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다. 화면 오른쪽 상단 모서리에서 1시간부터 1주까지 사전 정의된 시간 범위 중 하나를 선택할 수 있습니다(1시간, 3시간, 12시간, 1일, 3일 또는 1주). 또는 사용자 지정(Custom)을 선택하여 나만의 시간 범위를 설정할 수 있습니다.
3. 사용자 지정(Custom)을 선택한 다음 상자의 왼쪽 상단 모서리에 있는 절대(Absolute) 탭을 선택합니다. 캘린더 선택기나 텍스트 필드 상자를 사용하여 시간 범위를 지정합니다.
4. 시간 범위를 지정한 후 적용(Apply)을 선택합니다.

Original interface

그래프의 절대적 시간 범위를 설정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다. 화면 오른쪽 상단 모서리에서 1시간부터 1주까지 사전 정의된 시간 범위 중 하나를 선택할 수 있습니다(1시간, 3시간, 12시간, 1일, 3일 또는 1주). 또는 사용자 지정(custom)을 선택하여 나만의 시간 범위를 설정할 수 있습니다.
3. 사용자 지정(custom)을 선택한 다음 상자의 왼쪽 상단 모서리에 있는 절대(Absolute) 탭을 선택합니다. 캘린더 선택기나 텍스트 필드 상자를 사용하여 시간 범위를 지정합니다.
4. 시간 범위를 지정한 후 적용(Apply)을 선택합니다.

시간대 형식 설정

New interface

그래프의 시간대를 지정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다. 화면 오른쪽 상단 모서리에서 1시간부터 1주까지 사전 정의된 시간 범위 중 하나를 선택할 수 있습니다(1시간, 3시간, 12시간, 1일, 3일 또는 1주). 또는 사용자 지정(Custom)을 선택하여 나만의 시간 범위를 설정할 수 있습니다.
3. 사용자 지정(Custom)을 선택한 다음 상자의 오른쪽 상단 모서리에 있는 드롭다운을 선택합니다. 시간대를 UTC 또는 현지 시간대(Local time zone)로 변경할 수 있습니다.
4. 변경한 후 적용(Apply)을 선택합니다.

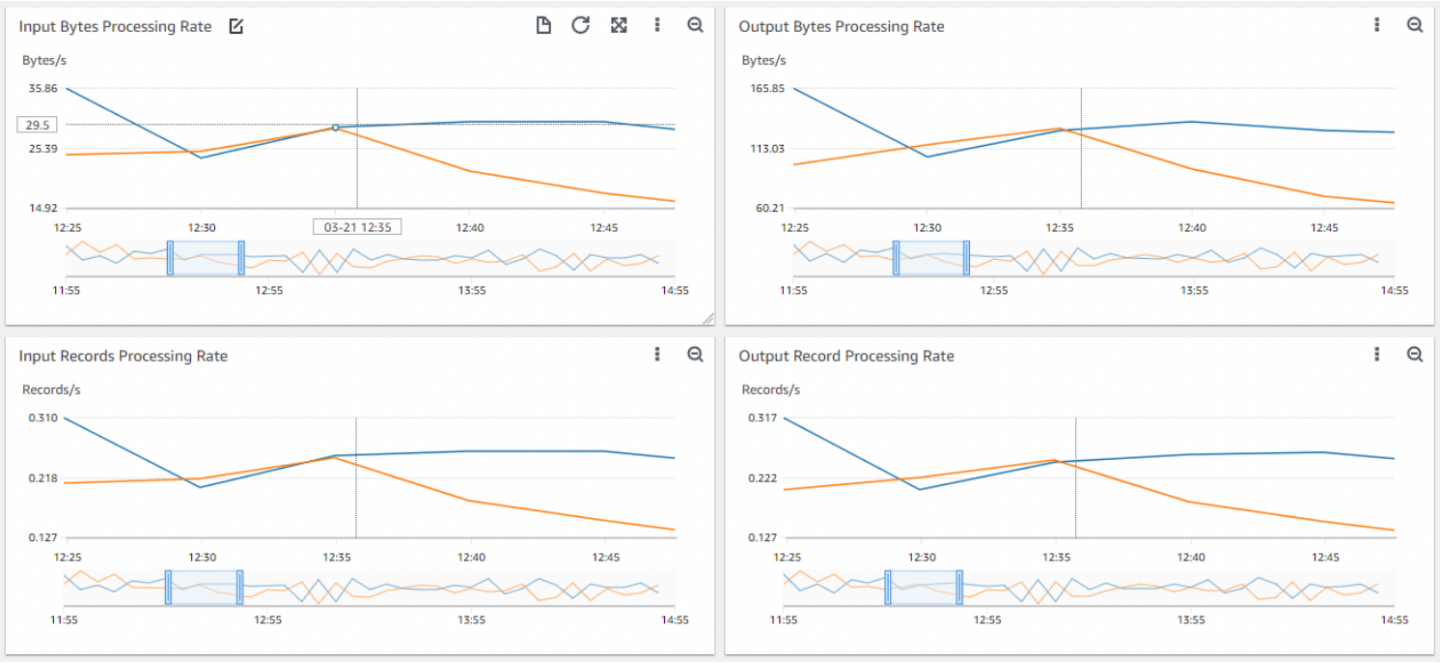
Original interface

그래프의 시간대를 지정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다. 화면 오른쪽 상단 모서리에서 1시간부터 1주까지 사전 정의된 시간 범위 중 하나를 선택할 수 있습니다(1시간, 3시간, 12시간, 1일, 3일 또는 1주). 또는 사용자 지정(custom)을 선택하여 나만의 시간 범위를 설정할 수 있습니다.
3. 사용자 지정(Custom)을 선택한 다음 상자의 오른쪽 상단 모서리에 있는 드롭다운을 선택합니다. 시간대를 UTC 또는 현지 시간대(Local time zone)로 변경할 수 있습니다.

선 그래프 또는 누적 영역 그래프 확대

CloudWatch 콘솔에서 미니 맵 확대/축소 기능을 사용하여 확대/축소 보기를 변경하지 않고도 선 그래프 및 누적 영역 그래프의 섹션에 초점을 맞출 수 있습니다. 예를 들어, 미니 맵 확대/축소 기능을 사용하여 선 그래프의 최고점에 초점을 맞출 수 있으므로 동일한 타임라인에서 대시보드의 다른 지표와 스파이크를 비교할 수 있습니다. 이 섹션의 절차에서는 확대/축소 기능을 사용하는 방법을 설명합니다.



위 이미지에서 확대/축소 기능은 입력 바이트 처리 속도와 관련된 선 그래프의 스파이크에 초점을 맞추면서 동일한 타임라인의 섹션에 초점을 맞춘 대시보드의 다른 선 그래프도 표시합니다.

New interface

그래프 확대

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택한 다음 모든 지표를 선택합니다.
3. 검색(Browse)을 선택한 다음 그래프로 표시할 지표를 선택합니다.
4. 옵션(Options)을 선택하고 위젯 유형(Widget type)에서 선(Line)을 선택합니다.
5. 초점을 맞추려는 그래프 영역을 선택하여 드래그한 다음 마우스 버튼을 놓습니다.
6. 확대/축소를 재설정하려면 안에 빼기(-) 기호가 있는 돋보기처럼 보이는 확대/축소 재설정 아이콘을 선택합니다.

Original interface

그래프 확대

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다.
3. 모든 지표(All metrics)를 선택한 다음 그래프로 표시할 지표를 선택합니다.

4. 그래프 옵션(Graph options)을 선택합니다. 위젯 유형(Widget type)에서 선(Line)을 선택합니다.
5. 초점을 맞추려는 그래프 영역을 선택하여 드래그한 다음 마우스 버튼을 놓습니다.
6. 확대/축소를 재설정하려면 안에 빼기(-) 기호가 있는 돋보기처럼 보이는 확대/축소 재설정 아이콘을 선택합니다.

> **Tip**
>
> 선 그래프 또는 누적 영역 그래프가 포함된 대시보드를 이미 생성한 경우 대시보드로 이동하여 확대/축소 기능을 사용할 수 있습니다.

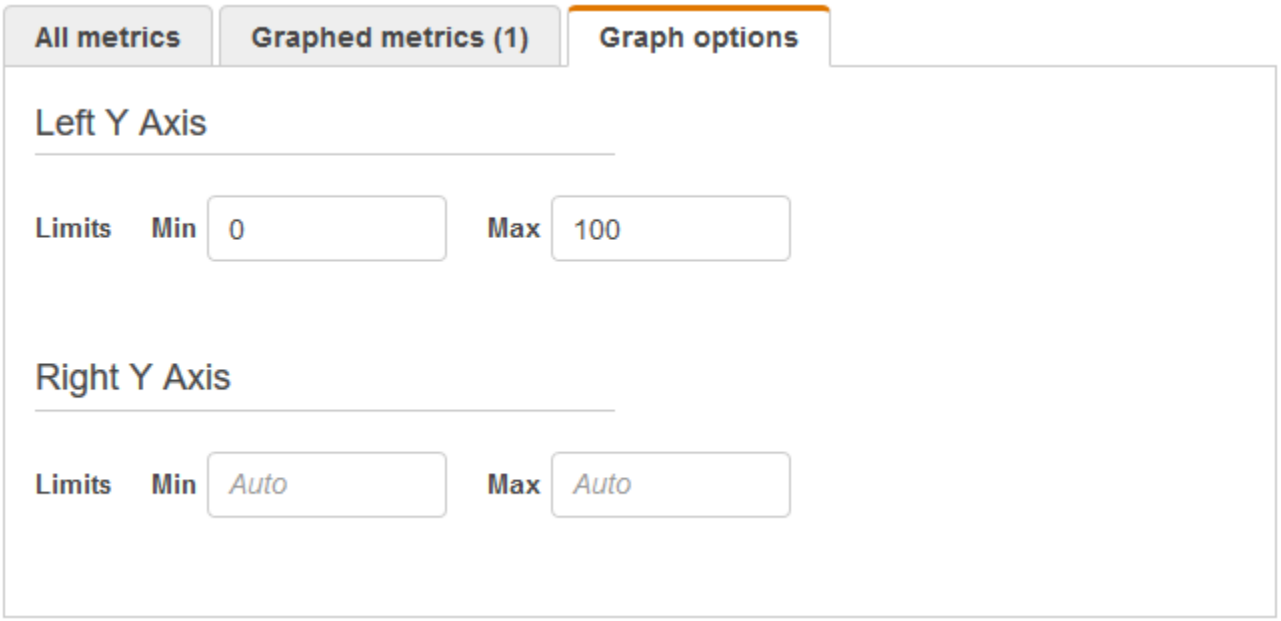
그래프의 y축 수정

데이터를 더 잘 볼 수 있도록 그래프에서 y축의 사용자 지정 경계를 설정할 수 있습니다. 예를 들어, CPU 사용률이 낮은지(그려지는 선이 그래프 하단 부근에 있음) 또는 높은지(그려지는 선이 그래프 상단 부근에 있음)를 쉽게 확인할 수 있도록 `CPUUtilization` 그래프의 경계를 100%로 변경할 수 있습니다.

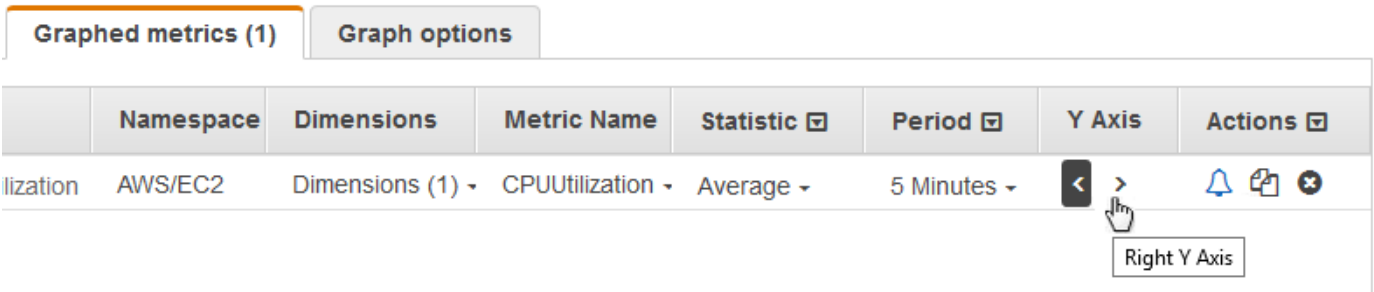
그래프에서 서로 다른 두 Y축 간을 전환할 수 있습니다. 이 기능은 그래프에 단위가 다르거나 값의 범위에 큰 차이가 있는 지표가 포함되어 있는 경우에 유용합니다.

그래프의 Y축을 수정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 지표 네임스페이스(예: EC2)를 선택한 후 지표 측정기준(예: 인스턴스별 지표)을 선택합니다.
4. 모든 지표 탭에 네임스페이스의 해당 측정기준에 대한 모든 지표가 표시됩니다. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다.
5. [그래프 옵션(Graph options)] 탭에서 [왼쪽 Y축(Left Y Axis)]의 [최소(Min)] 및 [최대(Max)] 값을 지정합니다. 최소 값은 최대 값보다 클 수 없습니다.



6. 두 번째 y축을 생성하려면 [오른쪽 Y축(Right Y Axis)]의 [최소(Min)] 및 [최대(Max)] 값을 지정합니다.
7. 두 Y축 간을 전환하려면 그래프로 표시된 지표 탭을 선택합니다. Y축에서 왼쪽 Y축 또는 오른쪽 Y축을 선택합니다.



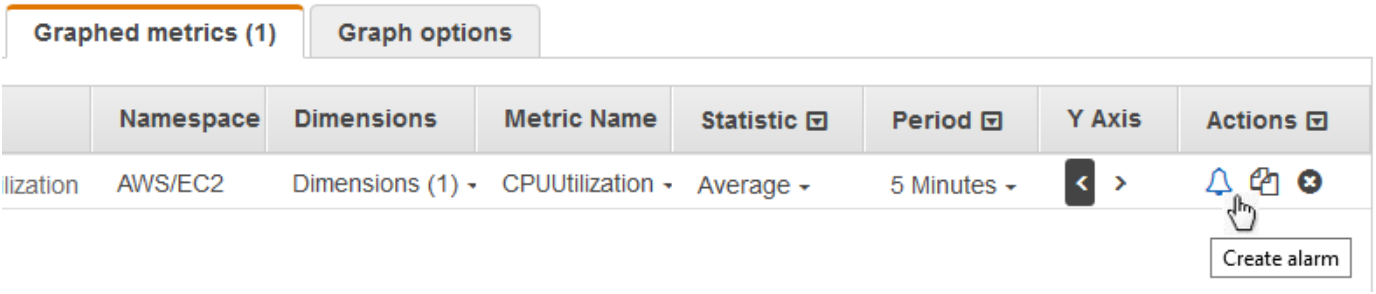
그래프의 지표에서 경보 생성

지표를 그래프 처리한 다음, 그래프의 지표에서 경보를 생성할 수 있습니다. 이 경우 다수의 경보 필드에서 값이 채워진다는 장점이 있습니다.

그래프의 지표에서 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 지표 네임스페이스(예: EC2)를 선택한 후 지표 측정기준(예: 인스턴스별 지표)을 선택합니다.

4. 모든 지표 탭에 네임스페이스의 해당 측정기준에 대한 모든 지표가 표시됩니다. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다.
5. 지표에 대한 경보를 생성하려면 그래프로 표시된 지표 탭을 선택합니다. 작업에서 경보 아이콘을 선택합니다.



6. 조건에서 정적 또는 이상 탐지를 선택하여 경보에 대해 정적 임계값을 사용할지 아니면 이상 탐지 모델을 사용할지를 지정합니다.

   선택한 사항에 따라 경보 조건에 대한 데이터의 나머지 부분을 입력합니다.
7. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

   N 중 M 경보를 생성하려면 두 번째 값에 지정한 값보다 낮은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가 단원을 참조하세요.
8. 누락 데이터 처리에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하세요.
9. 다음(Next)을 선택합니다.
10. 알림(Notification)에서 경보가 ALARM 상태, OK 상태 또는 INSUFFICIENT_DATA 상태일 때 알릴 SNS 주제를 선택합니다.

    경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

    경보에서 알림을 보내지 않게 하려면 제거를 선택합니다.
11. 경보가 Auto Scaling 또는 EC2 작업을 수행하도록 하려면 해당 버튼을 선택하고 경보 상태 및 수행할 작업을 선택합니다.
12. 마친 후에는 다음을 선택합니다.
13. 경보 이름 및 설명을 입력합니다. 이름은 ASCII 문자만 포함해야 합니다. 그리고 다음(Next)을 선택합니다.

14. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

CloudWatch 이상 탐지 사용

지표에 대해 '이상 탐지'를 사용 설정하면 CloudWatch는 통계 및 기계 학습 알고리즘을 적용합니다. 이러한 알고리즘은 시스템 및 애플리케이션의 지표를 지속적으로 분석하고, 정상 기준을 결정하며, 최소한의 사용자 개입으로 이상을 나타냅니다.

알고리즘은 이상 탐지 모델을 생성합니다. 모델은 정상 지표 동작을 나타내는 예상 값의 범위를 생성합니다.

AWS Management Console, AWS CLI, AWS CloudFormation 또는 AWS SDK를 사용하여 이상 탐지를 활성화할 수 있습니다. AWS에서 판매한 지표에 대해 및 사용자 정의 지표에 대해 이상 탐지를 활성화할 수 있습니다. CloudWatch 교차 계정 관찰성을 위해 모니터링 계정으로 설정된 계정에서 모니터링 계정의 지표뿐만 아니라 소스 계정의 지표에도 이상 탐지기를 만들 수 있습니다.

기댓값 모델을 두 가지 방법으로 사용할 수 있습니다.

- 지표의 예상 값을 기반으로 이상 탐지 경보를 생성합니다. 이러한 유형의 경보에는 경보 상태를 확인하기 위한 정적 임계값이 없습니다. 대신 이상 탐지 모델을 기반으로 지표의 값을 예상 값과 비교합니다.

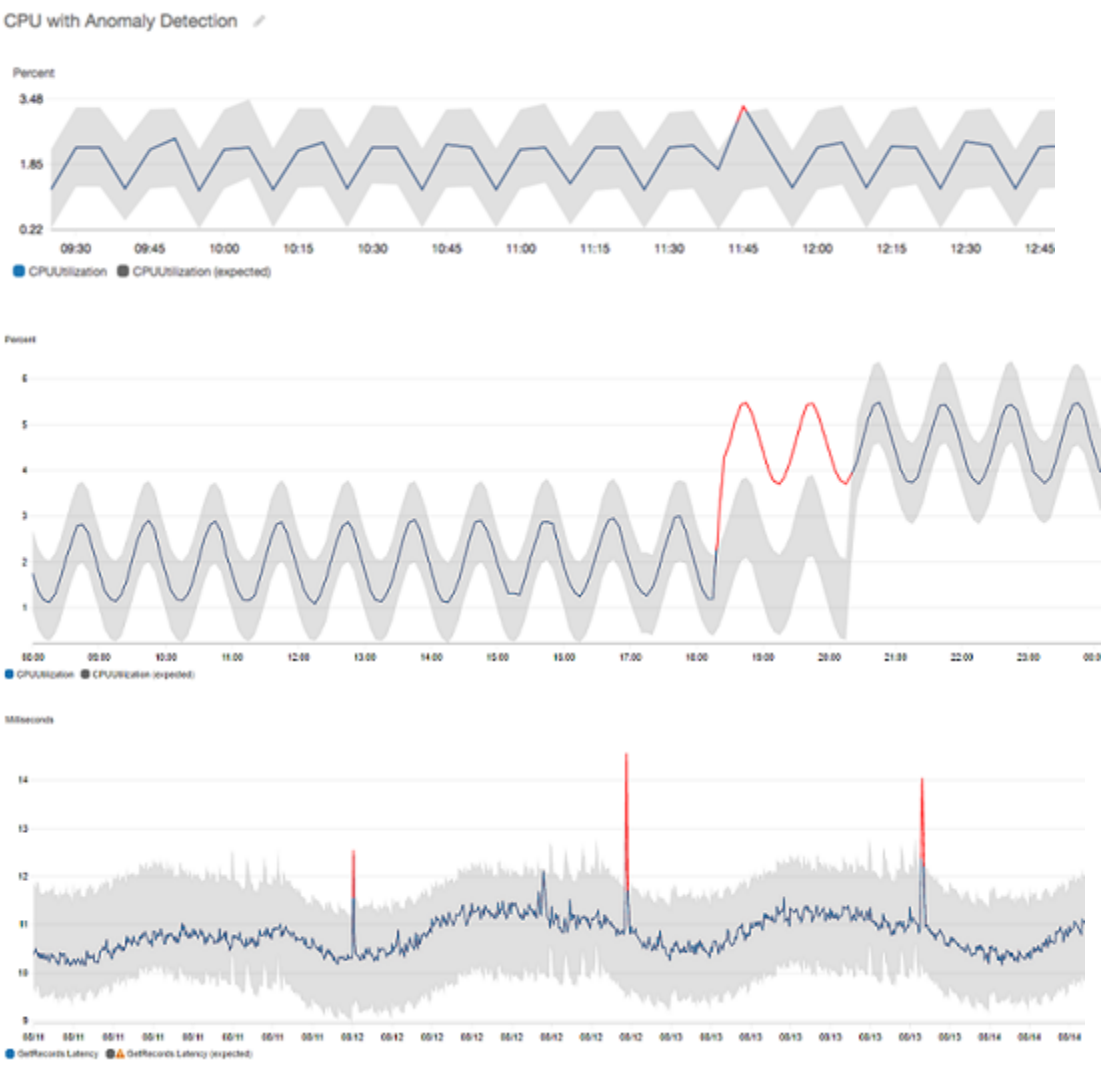
  지표 값이 예상 값의 밴드보다 높거나, 밴드보다 낮거나, 두 경우에 모두 해당할 때 경보를 트리거하도록 선택할 수 있습니다.

  자세한 내용은 이상 탐지를 기반으로 CloudWatch 경보 생성 단원을 참조하십시오.
- 지표 데이터의 그래프를 볼 때 예상 값을 그래프에 밴드로 오버레이합니다. 이렇게 하면 그래프의 어떤 값이 정상 범위를 벗어났는지 시각적으로 알 수 있습니다. 자세한 내용은 그래프 생성 단원을 참조하십시오.

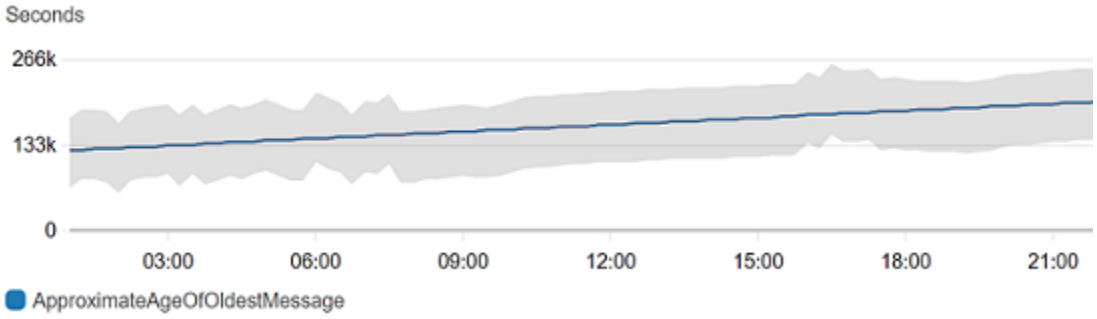
  또한, `ANOMALY_DETECTION_BAND` 지표 수학 함수가 있는 `GetMetricData` API 요청을 사용하여 모델 밴드의 상위 값과 하위 값을 검색할 수 있습니다. 자세한 내용은 GetMetricData 단원을 참조하세요.

이상 탐지가 있는 그래프에서 예상 값의 범위는 회색 밴드로 표시됩니다. 지표의 실제 값이 이 밴드를 초과하면 해당 시간 동안 빨간색으로 표시됩니다.

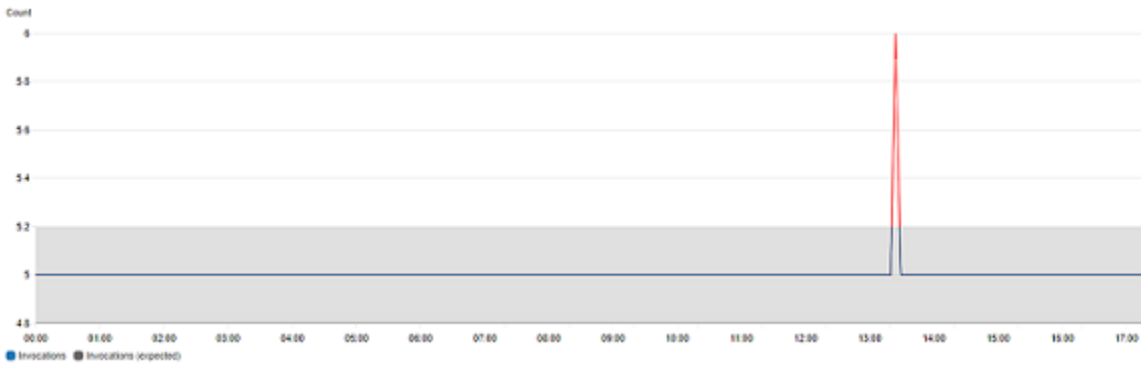
이상 탐지 알고리즘은 지표의 계절성 및 추세 변화를 설명합니다. 계절성 변화는 다음 예와 같이 시간별, 일별 또는 주별일 수 있습니다.



장거리 추세는 하향 또는 상향 추세가 될 수 있습니다.



이상 탐지는 플랫 패턴이 있는 지표에서도 잘 작동합니다.



CloudWatch 이상 탐지의 작동 방식

지표에 대해 이상 탐지를 사용 설정하면 CloudWatch는 기계 학습 알고리즘을 지표의 과거 데이터에 적용하여 지표의 예상 값 모델을 생성합니다. 모델은 지표의 추세와 시간별, 일별 및 주별 패턴을 모두 평가합니다. 이 알고리즘은 최대 2주의 지표 데이터를 교육할 수 있지만, 지표에 2주분의 데이터가 없더라도 지표에 이상 탐지를 활성화할 수 있습니다.

CloudWatch가 모델과 함께 지표 값의 '정상' 범위를 결정하는 데 사용하는 이상 탐지 임계값에 대한 값을 지정합니다. 이상 탐지 임계값의 값이 높을수록 "정상" 값의 밴드가 두꺼워집니다.

기계 학습 모델은 지표 및 통계에 고유합니다. 예를 들어, `AVG` 통계를 사용한 지표에 대한 이상 탐지를 활성화하면, 해당 모델은 `AVG` 통계에 고유합니다.

CloudWatch는 AWS 서비스의 많은 공통 지표에 대한 모델을 생성할 때 밴드가 논리적 값을 벗어나지 않도록 합니다. 예를 들어, EC2 인스턴스의 `MemoryUtilization` 대역은 0에서 100 사이로 유지되고, CloudFront `Requests`를 추적하는 대역(음수일 수 없음)은 0 이하로 확장되지 않습니다.

모델을 생성한 후 CloudWatch 이상 탐지는 지속적으로 모델을 평가하며 모델이 가능한 한 정확성을 유지하도록 모델을 조정합니다. 여기에는 지표 값이 시간 경과에 따라 점진적으로 변화하거나 갑작스럽게 변하는 경우에 조정하기 위해 모델을 다시 훈련하는 것이 포함되며, 계절성 또는 희소성이거나 급증하는 지표의 모델을 개선하기 위한 예측 변수도 포함됩니다.

지표에 이상 탐지를 활성화한 다음 모델을 교육할 때, 지표에 특정 기간을 사용하지 않도록 제외할 수 있습니다. 이렇게 하면, 모델 교육을 할 때 배포 또는 다른 이상 이벤트 등을 제외할 수 있어 가장 정확한 모델을 생성할 수 있습니다.

경보용 이상 탐지 모델을 사용하면 AWS 계정에 비용이 청구됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

지표 수학에 대한 이상 탐지

지표 수학에 대한 이상 탐지는 출력 지표 수학 표현식에 대한 이상 탐지 경보를 생성하는 데 사용할 수 있는 기능입니다. 이러한 표현식을 사용하여 이상 탐지 밴드를 시각화하는 그래프를 만들 수 있습니다. 이 기능은 기본 산술 함수, 비교 및 논리 연산자 및 대부분의 기타 함수를 지원합니다. 지원되지 않는 함수에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 지표 수학 사용을 참조하세요.

이상 탐지 모델을 생성한 방식과 유사하게 지표 수학 표현식을 기반으로 이상 탐지 모델을 생성할 수 있습니다. CloudWatch 콘솔에서 이상 탐지를 지표 수학 표현식에 적용해 이상 탐지를 이러한 표현식의 임계값 유형으로 선택할 수 있습니다.

> **Note**
>
> 지표 수학에 대한 이상 탐지는 최신 버전의 지표 사용자 인터페이스에서만 활성화 및 편집할 수 있습니다. 새 버전의 인터페이스에서 지표 수학 표현식을 기반으로 이상 탐지기를 만드는 경우 이전 버전에서 이상 탐지기를 볼 수는 있지만 편집할 수는 없습니다.

이상 탐지 및 지표 수학에 대한 경보 및 모델을 만드는 방법에 대한 자세한 내용은 다음 섹션을 참조하세요.

- 이상 탐지를 기반으로 CloudWatch 경보 생성
- 지표 수학 표현식을 기반으로 CloudWatch 경보 생성

`PutAnomalyDetector`, `DeleteAnomalyDetector`, `DescribeAnomalyDetectors`와 함께 CloudWatch API를 사용하여 지표 수학 표현식을 기반으로 이상 탐지 모델을 생성, 삭제 및 검색할 수도 있습니다. 이러한 API 작업에 대한 자세한 내용은 Amazon CloudWatch API 참조의 다음 섹션을 참조하세요.

- PutAnomalyDetector
- DeleteAnomalyDetector
- DescribeAnomalyDetectors

이상 탐지 경보의 요금이 책정되는 방법에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

지표 수학 사용

지표 수학을 사용하면 여러 CloudWatch 지표를 쿼리하고 수학 표현식을 활용함으로써 이러한 지표를 기반으로 새로운 시계열을 만들 수 있습니다. CloudWatch 콘솔에서 결과 시계열을 시각화하고 대시보드에 추가할 수 있습니다. 예를 들어 AWS Lambda 지표를 사용하여 `Errors` 지표를 `Invocations` 지표로 나누어 오류 발생률을 얻을 수 있습니다. 그런 다음, 결과 시계열을 CloudWatch 대시보드의 그래프에 추가합니다.

`GetMetricData` API 작업을 사용하여 지표 계산을 프로그래밍방식으로 실행할 수도 있습니다. 자세한 내용은 GetMetricData 단원을 참조하세요.

CloudWatch 그래프에 수학 표현식 추가

CloudWatch 대시보드의 그래프에 수학 표현식을 추가할 수 있습니다. 각 그래프의 지표와 표현식은 최대 500개까지 사용할 수 있도록 제한되므로 그래프의 지표가 499개 이하일 경우에만 수학 표현식을 추가할 수 있습니다. 이것은 그래프에 모든 지표가 표시되지 않는 경우에도 적용됩니다.

그래프에 수학 표현식을 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 그래프를 생성하거나 편집합니다. 그래프에 지표가 하나 이상 있어야 합니다.
3. 그래프로 표시된 지표(Graphed metrics)를 선택합니다.
4. 수학 표현식, 빈 표현식으로 시작을 선택합니다. 표현식의 새 줄이 나타납니다.
5. 새 행의 세부 정보 열에 수학 표현식을 입력합니다. 지표 수식 구문 및 함수 섹션의 테이블에는 표현식에 사용할 수 있는 함수가 나열됩니다.

   이 표현식을 위한 공식의 일부로 지표 또는 다른 표현식 결과를 사용하려면 Id 열에 표시된 값을 사용합니다(예: m1+m2 또는 e1-MIN(e1)).

   Id 값은 변경할 수 없습니다. 숫자, 문자, 밑줄이 포함될 수 있으며, 소문자로 시작해야 합니다. Id의 값을 좀 더 의미 있는 이름으로 변경하면 그래프를 더 쉽게 이해할 수 있습니다(예: m1 및 m2에서 errors 및 requests로 변경하는 경우).

   > **Tip**
   >
   > 수식 표현식 옆의 아래쪽 화살표를 선택하여 표현식을 만들 때 사용할 수 있는 지원되는 함수 목록을 확인합니다.

6. 표현식의 레이블 열에는 표현식으로 계산되는 사항을 설명하는 이름을 입력합니다.

   표현식의 결과가 시계열 배열인 경우, 그 각각의 시계열이 그래프에 각각의 행과 서로 다른 색상으로 표시됩니다. 그래프 바로 아래에 그래프 내 각 행의 범례가 표시됩니다. 하나의 표현식이 여러 개의 시계열을 생성하는 경우, 해당 시계열의 범례 캡션은 ***Expression-Label Metric-Label(###-### ##-###)*** 형식으로 표시됩니다. 예를 들어 그래프에 [오류(Errors)] 레이블이 있는 지표와 [0으로 채워짐:(Filled With 0:)] 레이블이 있는 [FILL(METRICS(), 0)] 표현식이 포함되어 있는 경우 범례의 한 줄은 [0으로 채워짐: 오류(Filled With 0: Errors)]가 됩니다. 범례에 원래의 지표 레이블만 표시하려면 *Expression-Label(###-###)*을 비워둡니다.

   한 표현식이 그래프에 시계열 배열을 생성하면 해당 시계열 각각에 사용된 색상을 변경할 수 없습니다.

7. 원하는 표현식을 추가한 후에는 원래 지표 일부를 숨겨 그래프를 간소화할 수 있습니다. 지표 또는 표현식을 숨기려면 Id 필드 좌측의 확인란 선택을 지웁니다.

지표 수학 구문 및 함수

아래 단원에서는 지표 수식에서 사용되는 함수를 설명합니다. 모든 함수는 대문자로 작성해야 하며(예: AVG), 모든 지표와 수학 표현식의 Id 필드는 소문자로 시작해야 합니다.

수학 표현식의 최종 결과는 단일 시계열이거나 시계열 배열이어야 합니다. 일부 함수는 스칼라 수를 생성합니다. 최종적으로 하나의 시계열을 생성하는 더 큰 함수 안에서 이러한 함수를 사용할 수 있습니다. 예를 들어, 단일 시계열에서 AVG를 빼면 스칼라 수가 생성되므로 최종 표현식 결과가 될 수 없습니다. 그러나 m1-AVG(m1) 함수에서 이를 사용하면 각 개별 데이터 요소와 시계열의 평균값 간 차이에 대한 시계열을 표시할 수 있습니다.

데이터 유형 약어

일부 함수는 특정 형식의 데이터에만 유효합니다. 다음 목록에 나와 있는 약어는 각 함수에 지원되는 데이터 형식을 나타내는 함수 표에서 사용됩니다.

- S는 2, -5 또는 50.25와 같은 스칼라 수를 나타냅니다.
- TS는 지난 3일 동안 `i-1234567890abcdef0` 인스턴스의 `CPUUtilization` 지표와 같은 시계열(시간 경과에 따른 일련의 단일 CloudWatch 지표 값)입니다.
- TS[]는 시계열 배열입니다(예: 여러 지표에 대한 시계열).
- String[]은 문자열 배열입니다.

METRICS() 함수

METRICS() 함수는 요청에 모든 지표를 반환합니다. 수학 표현식은 포함되지 않습니다.

단일 시계열이나 시계열 배열을 생성하는 더 큰 표현식 안에 METRICS() 를 사용할 수 있습니다. 예를 들어, 표현식 SUM(METRICS())은 모든 그래프 지표 값의 합인 시계열(TS)을 반환합니다. METRICS()/100 은 시계열 배열을 반환하며 그 각각은 지표 중 하나의 각 데이터 요소를 100으로 나눈 값을 표시하는 시계열입니다.

METRICS() 함수를 문자열과 함께 사용하여 그 Id 필드에 해당 문자열이 있는 그래프 지표만 반환할 수 있습니다. 예를 들어, 표현식 SUM(METRICS("errors"))은 그 Id 필드에 '오류'가 있는 모든 그래프 지표 값의 합인 시계열을 반환합니다. SUM([METRICS(“4xx”), METRICS(“5xx”)])을 사용하여 여러 문자열을 일치시킬 수도 있습니다.

기본 산술 함수

다음 표에는 지원되는 기본 산술 함수가 나와 있습니다. 시계열의 누락 값은 0으로 처리됩니다. 데이터 요소의 값 때문에 함수에서 0으로 나누려고 시도할 경우 해당 데이터 요소가 누락됩니다.

Operation	인수	예제
산술 연산자: + - * / ^	S, S	PERIOD(m1)/60
	S, TS	5 * m1
	TS, TS	m1 - m2
	S, TS[]	SUM(100/[m1, m2])
	TS, TS[]	AVG(METRICS())
		METRICS()*100
빼기 기호 -	S	-5*m1
	TS	-m1
	TS[]	SUM(-[m1, m2])

비교 및 논리 연산자

비교 및 논리 연산자를 시계열 쌍이나 단일 스칼라 값 쌍과 함께 사용할 수 있습니다. 비교 연산자를 시계열 쌍과 함께 사용하면 연산자는 각 데이터 요소가 0(false) 또는 1(true)인 시계열을 반환합니다. 스칼라 값 쌍에 비교 연산자를 사용하면 0 또는 1 중 하나의 스칼라 값이 반환됩니다.

비교 연산자가 두 시계열 사이에 사용되고 시계열 중 하나에만 특정 타임스탬프에 대한 값이 있는 경우, 이 함수는 다른 시계열의 누락 값을 0으로 처리합니다.

논리 연산자를 비교 연산자와 함께 사용하여 보다 복잡한 함수를 만들 수 있습니다.

다음 표에는 지원되는 연산자가 나와 있습니다.

연산자 유형	지원되는 연산자
비교 연산자	== != <= >= < >
논리 연산자	AND 및 && OR 또는 ||

이러한 연산자가 어떻게 사용되는지 알아보기 위해 2개의 시계열이 있다고 가정해 보겠습니다. metric1의 값은 `[30, 20, 0, 0]`이고 metric2의 값은 `[20, -, 20, -]`입니다. 여기서 - 기호는 해당 타임스탬프에 대한 값이 없음을 나타냅니다.

표현식	출력
(metric1 < metric2)	0, 0, 1, 0
(metric1 >= 30)	1, 0, 0, 0

표현식	출력
(metric1 > 15 AND metric2 > 15)	1, 0, 0, 0

지표 수학에 지원되는 함수

다음 표에서는 수학 표현식에서 사용할 수 있는 함수를 설명합니다. 모든 함수를 대문자로 입력합니다.

수학 표현식의 최종 결과는 단일 시계열이거나 시계열 배열이어야 합니다. 아래 단원에 나오는 표의 일부 함수는 스칼라 수를 생성합니다. 최종적으로 하나의 시계열을 생성하는 더 큰 함수 안에서 이러한 함수를 사용할 수 있습니다. 예를 들어, 단일 시계열에서 AVG를 빼면 스칼라 수가 생성되므로 최종 표현식 결과가 될 수 없습니다. 그렇지만 이것을 함수 m1-AVG(m1)에서 사용하여 각 개별 데이터 요소와 해당 데이터 요소의 평균값 차이인 시계열을 표시할 수 있습니다.

아래의 표에서 예제 열의 모든 예제는 단일 시계열이나 시계열 배열을 생성하는 표현식입니다. 스칼라 수를 반환하는 함수를 단일 시계열을 생성하는 유효한 표현식의 일부로 사용하는 방법을 보여주는 예제입니다.

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
ABS	TS TS[]	TS TS[]	각 데이터 요소의 절대값을 반환합니다.	ABS(m1-m2) MIN(ABS([m1, m2])) ABS(METRICS())	✓
ANOMALY_DETECTION_BAND	TS TS, S	TS[]	지정된 지표에 대한 이상 탐지 밴드를 반환합니다. 밴드는 두 시계열로 구성되어 있으며 하나는 지표의 "정상" 기댓값에 대한 상한 치수를 나타내며 다른 하나는 하한 치수를	ANOMALY_DETECTION_BAND(m1) ANOMALY_DETECTION_BAND(m1,4)	

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			나타냅니다. 함수는 두 개의 인수가 필요합니다. 첫 번째는 밴드를 생성할 지표의 ID입니다. 두 번째 인수는 밴드에 사용할 표준 편차의 수입니다. 이 인수를 지정하지 않으면, 두 개의 기본값이 사용됩니다. 자세한 내용은 CloudWatch 이상 탐지 사용 단원을 참조하십시오.		

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
AVG	TS TS[]	S TS	단일 시계열의 AVG는 지표의 모든 데이터 요소 평균을 나타내는 스칼라를 반환합니다. 시계열 배열의 AVG는 단일 시계열을 반환합니다. 누락된 값은 0로 처리됩니다. **Note** CloudWatch 경보에 이 함수를 사용하지 않는 것이 좋습니다. 예를 들면 `AVG(m2)`입니다. 경보가 상태 변경 여부를 평가할 때마다 CloudWatch는 평가 기간으로 지정된 수보다 더 높은 수의 데이터 요소를 검색하려고 합니다. 이 함수는 추가 데이터가 요청될 때 다르게 작동합니다.	SUM([m1,m2])/AVG(m2) AVG(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			이 기능을 경보, 특히 Auto Scaling 동작이 있는 경보에 사용하려면 경보가 N개의 데이터 포인트 중 M개를 사용하도록 설정하는 것이 좋습니다(여기서 M < N).		
CEIL	TS TS[]	TS TS[]	각 지표의 상한을 반환합니다. 상한은 각 값보다 크거나 같은 가장 작은 정수입니다.	CEIL(m1) CEIL(METRICS()) SUM(CEIL(METRICS()))	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
DATAPOINT_COUNT	TS TS[]	S TS	값을 보고한 데이터 포인트의 수를 반환합니다. 이는 희소 지표의 평균을 계산하는 데 유용합니다. **Note** CloudWatch 경보에 이 함수를 사용하지 않는 것이 좋습니다. 경보가 상태 변경 여부를 평가할 때마다 CloudWatch는 평가 기간으로 지정된 수보다 더 높은 수의 데이터 요소를 검색하려고 합니다. 이 함수는 추가 데이터가 요청될 때 다르게 작동합니다.	SUM(m1) / DATAPOINT_COUNT(m1) DATAPOINT_COUNT(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
DB_PERF_INSIGHTS	문자열, 문자열, 문자열 문자열, 문자열, 문자열[]	TS(단일 문자열이 주어진 경우) TS[](문자열 배열이 주어진 경우)	Amazon 관계형 데이터베이스 서비스 및 Amazon DocumentDB(MongoDB 호환)와 같은 데이터베이스에 대한 성능 개선 도우미 카운터 지표를 반환합니다. 이 함수는 Performance Insights API를 직접 쿼리하여 얻을 수 있는 것과 동일한 양의 데이터를 반환합니다. CloudWatch에서 이러한 지표를 사용하여 그래프를 작성하고 경보를 생성할 수 있습니다. **Important** 이 함수를 사용할 때는 데이터베이스의 고유 데이터베이스 리소스 ID를 지정해야 합니다. 이는 데이터베이스 식별자와는 다릅니다. Amazon RDS 콘솔에서 데이터베이스	DB_PERF_INSIGHTS('RDS', 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', 'os.cpuUtilization.user.avg') DB_PERF_INSIGHTS('DOCDB, 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', ['os.cpuUtilization.idle.avg', 'os.cpuUtilization.user.max'])	

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			리소스 ID를 찾으려면 DB 인스턴스를 선택하여 세부 정보를 확인합니다. 그런 다음 구성 탭을 선택합니다. 그러면 리소스 ID가 구성 섹션에 표시됩니다. 또한 DB_PERF_INSIGHTS는 1분 미만의 간격으로 DBLoad 지표를 가져옵니다. 이 함수로 검색된 성능 개선 도우미 지표는 CloudWatch에 저장되지 않습니다. 따라서 계정 간 통합 가시성, 이상 탐색, 지표 스트림, 지표 탐색기, Metric Insights와 같은 일부 CloudWatch 기능은 DB_PERF_INSIGHTS로 검색하는 성능 개선 도우미 지표에서는 작동하지 않습니다.		

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			DB_PERF_INSIGHTS 함수를 사용하는 단일 요청으로 다음과 같은 개수의 데이터 포인트를 검색할 수 있습니다. • 고해상도 기간(1초, 10초, 30초) 동안 1080개의 데이터 포인트 • 표준 해상도 기간(1분, 5분, 1시간, 1일) 동안 1440개의 데이터 포인트 DB_PERF_INSIGHTS 함수는 다음 기간 길이만 지원합니다. • 1초 • 10초 • 30초 • 1분 • 5분 • 1시간 • 1일 Amazon RDS 성능 개선 도우미 카운터 지표에 대		

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			한 자세한 내용은 성능 개선 도우미 카운터 섹션을 참조하세요. Amazon DocumentDB 성능 개선 도우미 카운터 지표에 대한 자세한 내용은 성능 개선 도우미 카운터 섹션을 참조하세요. **Note** DB_PERF_INSIGHTS가 검색하는 분 단위 이하의 고분해능 지표는 DBLoad 지표 또는 운영 체제 지표(고분해능에서 Enhanced Monitoring을 사용하도록 설정한 경우)에만 적용됩니다. Amazon RDS 확장 모니터링에 대한 자세한 내용은 Enhanced Monitoring을 사		

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			용하여 OS 지표 모니터링을 참조하세요. DB_PERF_INSIGHTS 함수를 사용하여 최대 3시간 범위의 고분해능 경보를 생성할 수 있습니다. CloudWatch 콘솔을 사용하여 DB_PERF_INSIGHTS 함수로 검색된 지표를 원하는 시간 범위에 대해 그래프로 표시할 수 있습니다.		
DIFF	TS TS[]	TS TS[]	시계열의 각 값과 해당 시계열의 이전 값 간 차이를 반환합니다.	DIFF(m1)	✓
DIFF_TIME	TS TS[]	TS TS[]	시계열에 있는 각 값의 타임스탬프와 해당 시계열에 있는 이전 값의 타임스탬프 간 차이(초)를 반환합니다.	DIFF_TIME(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
FILL	TS, [S | REPEA | LINEAR TS[], [TS | S | REPEA | LINEAR	TS TS[]	시계열의 누락된 값을 채웁니다. 누락된 값의 필러로 사용할 값과 관련하여 다음과 같은 몇 가지 옵션이 있습니다. • 필러 값으로 사용할 값을 지정할 수 있습니다. • 필러 값으로 사용할 지표를 지정할 수 있습니다. • REPEAT 키워드를 사용하여 누락된 값 이전 지표의 가장 최근 실제 값으로 누락된 값을 채울 수 있습니다. • LINEAR 키워드를 사용하여 결측의 시작 값과 끝 값 사이에 선형 보간을 생성하는 값으로 누락된 값을 채울 수 있습니다. Note 경보에 이 함수를 사용할 때 지표를 약간 지	FILL(m1,10) FILL(METRICS(), 0) FILL(METRICS(), m1) FILL(m1, MIN(m1)) FILL(m1, REPEAT) FILL(METRICS(), LINEAR)	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			체하여 게시하고 가장 최근 분에 데이터가 없는 경우 문제가 발생할 수 있습니다. 이 경우 FILL은 누락된 데이터 요소를 요청된 값으로 바꿉니다. 그러면 지표의 최신 데이터 요소가 항상 FILL 값이 되어 경보가 OK 상태 또는 ALARM 상태에서 멈추게 될 수 있습니다. 'M out of N' 경보를 사용하여 이 문제를 해결할 수 있습니다. 자세한 내용은 경보 평가 단원을 참조하십시오.		

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
FIRST LAST	TS[]	TS	시계열 배열에서 첫 번째 시계열 또는 마지막 시계열을 반환합니다. SORT 함수와 함께 사용할 때 유용합니다. 또한 ANOMALY_DETECTION_BAND 함수에서 높은 임계값과 낮은 임계값을 가져오는 데도 사용할 수 있습니다.	IF(FIRST(SORT(METRICS(), AVG, DESC))>100, 1, 0) 배열에서 AVG별로 정렬된 상위 지표를 확인합니다. 그런 다음, 해당 데이터 포인트 값이 100 이상인지 여부에 따라 각 데이터 포인트에 대해 1 또는 0을 반환합니다. LAST(ANOMALY_DETECTION_BAND(m1))은 이상 예측 대역의 상한값을 반환합니다.	✓
FLOOR	TS TS[]	TS TS[]	각 지표의 하한을 반환합니다. 하한은 각 값보다 작거나 같은 가장 큰 정수입니다.	FLOOR(m1) FLOOR(METRICS())	✓
IF	IF 표현식	TS	IF를 비교 연산자와 함께 사용하여 시계열에서 데이터 요소를 필터링하거나 수집된 여러 시계열로 구성된 혼합 시계열을 생성합니다. 자세한 내용은 IF 표현식 사용 단원을 참조하십시오.	예를 보려면 IF 표현식 사용을 참조하세요.	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
INSIGHT_RULE_METRIC	INSIGHT_RULE_METRIC(ruleName, metricName)	TS	INSIGHT_RULE_METRIC을 사용하여 Contributor Insights의 규칙에서 통계를 추출합니다. 자세한 내용은 규칙에 따라 생성된 지표 그래프 작성 단원을 참조하십시오.		
LAMBDA	LAMBDA(functionNam[, optionalarg]*)	TS TS{}	Lambda 함수를 직접적으로 호출하여 CloudWatch가 아닌 데이터 소스에서 지표를 쿼리합니다. 자세한 내용은 Lambda 함수에 인수를 전달하는 방법 단원을 참조하십시오.		
LOG	TS TS[]	TS TS[]	시계열의 LOG는 시계열에 있는 각 값의 자연 로그 값을 반환합니다.	LOG(METRICS())	✓
LOG10	TS TS[]	TS TS[]	시계열의 LOG10은 시계열에 있는 각 값의 base-10 로그 값을 반환합니다.	LOG10(m1)	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
MAX	TS TS[]	S TS	단일 시계열의 MAX는 지표의 모든 데이터 요소의 최대값을 나타내는 스칼라를 반환합니다. 시계열 배열을 입력하면 MAX 함수는 입력으로 사용된 시계열 중에서 각 데이터 포인트의 가장 큰 값으로 구성된 시계열을 만들어 반환합니다. **Note** CloudWatch 경보에 이 함수를 사용하지 않는 것이 좋습니다. 예를 들어 `MAX(m2)`의 경우 경보가 상태 변경 여부를 평가할 때마다 CloudWatch는 평가 기간으로 지정된 수보다 더 높은 수의 데이터 포인트를 검색하려고 합니다. 이 함수는 추	MAX(m1)/m1 MAX(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			가 데이터가 요청될 때 다르게 작동합니다.		
METRIC_COUNT	TS[]	S	시계열 어레이의 지표 수를 반환합니다.	m1/METRIC_COUNT(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
지표	null 문자열	TS[]	METRICS() 함수는 요청의 모든 CloudWatch 지표를 반환합니다. 수학 표현식은 포함되지 않습니다. 단일 시계열이나 시계열 배열을 생성하는 더 큰 표현식 안에 METRICS() 를 사용할 수 있습니다. METRICS() 함수를 문자열과 함께 사용하여 그 Id 필드에 해당 문자열이 있는 그래프 지표만 반환할 수 있습니다. 예를 들어, 표현식 SUM(METRICS("errors"))은 그 Id 필드에 '오류'가 있는 모든 그래프 지표 값의 합인 시계열을 반환합니다. SUM([METRICS(“4xx”), METRICS(“5xx”)])을 사용하여 여러 문자열을 일치시킬 수도 있습니다.	AVG(METRICS()) SUM(METRICS("errors"))	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
MIN	TS TS[]	S TS	단일 시계열의 MIN은 지표의 모든 데이터 요소의 최소값을 나타내는 스칼라를 반환합니다. 시계열 배열을 입력하면 MIN 함수는 입력으로 사용된 시계열 중에서 각 데이터 포인트의 가장 작은 값으로 구성된 시계열을 만들어 반환합니다. 시계열 배열을 입력하면 MIN 함수는 입력으로 사용된 시계열 중에서 각 데이터 포인트의 가장 큰 값으로 구성된 시계열을 만들어 반환합니다. **Note** CloudWatch 경보에 이 함수를 사용하지 않는 것이 좋습니다. 예를 들어 `MIN(m2)`의 경우 경보가 상태 변경 여부를 평가할 때마다	m1-MIN(m1) MIN(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			CloudWatch는 평가 기간으로 지정된 수보다 더 높은 수의 데이터 포인트를 검색하려고 합니다. 이 함수는 추가 데이터가 요청될 때 다르게 작동합니다.		

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?

분
시간
요일
날짜
달
년
EPOCH
	TS	TS	이러한 함수는 시계열의 기간 및 범위를 사용하여 각 값이 해당 타임스탬프를 기반으로 하는 새로운 비희소 시계열을 반환합니다.
			• MINUTE은 원래 시계열에 있는 각 타임스탬프의 UTC 분을 나타내는 정수(0~59)의 비희소 시계열을 반환합니다.
			• HOUR는 원래 시계열에 있는 각 타임스탬프의 UTC 시를 나타내는 정수(0~23)의 비희소 시계열을 반환합니다.
			• DAY는 원래 시계열에 있는 각 타임스탬프의 UTC 요일을 나타내는 정수(1~7)의 비희소 시계열을 반환합니다. 1은 월요일을 나타내고 7은 일요일을 나타냅니다.
			• DATE는 원래 시계열에 있는 각 타임스탬프의 UTC 날짜를 나타내
				MINUTE(m1)
				IF(DAY(m1)<6,m1)는 월요일부터 금요일까지 평일의 지표만 반환합니다.
				IF(MONTH(m1) == 4,m1)는 4월에 게시된 지표만 반환합니다.
					✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
			는 정수(1~31)의 비희소 시계열을 반환합니다. • MONTH는 원래 시계열에 있는 각 타임스탬프의 UTC 월을 나타내는 정수(1~12)의 비희소 시계열을 반환합니다. 1은 1월을 나타내고 12는 12월을 나타냅니다. • YEAR는 원래 시계열에 있는 각 타임스탬프의 UTC 연을 나타내는 정수의 비희소 시계열을 반환합니다. • EPOCH는 원래 시계열에 있는 각 타임스탬프의 Epoch 이후 UTC 시간(초)을 나타내는 정수의 비희소 시계열을 반환합니다. Epoch는 1970년 1월 1일입니다.		
PERIOD	TS	S	지표의 기간(초)을 반환합니다. 유효한 입력은 지표이지 다른 표현식의 결과가 아닙니다.	m1/PERIOD(m1)	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
속도	TS TS[]	TS TS[]	지표의 초당 변경 비율을 반환합니다. 이것은 마지막 데이터 요소 값과 그 이전의 데이터 요소 값의 차이를 두 값의 시간차(초)로 나눈 값으로 계산됩니다. **Important** 희소 데이터가 포함된 지표에 RATE 함수를 사용하는 표현식에 대한 경보를 설정하면 경보를 평가할 때 가져온 데이터 포인트의 범위가 데이터 포인트의 마지막 게시 시기에 따라 달라질 수 있기 때문에 예상치 못한 동작이 발생할 수 있습니다.	RATE(m1) RATE(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
REMOVE_EMPTY	TS[]	TS[]	시계열 배열에서 데이터 포인트가 없는 시계열을 제거합니다. 결과는 각 시계열에 적어도 하나의 데이터 포인트가 포함된 시계열 배열입니다. **Note** CloudWatch 경보에 이 함수를 사용하지 않는 것이 좋습니다. 경보가 상태 변경 여부를 평가할 때마다 CloudWatch는 평가 기간으로 지정된 수보다 더 높은 수의 데이터 요소를 검색하려고 합니다. 이 함수는 추가 데이터가 요청될 때 다르게 작동합니다.	REMOVE_EMPTY(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
RUNNING_SUM	TS TS[]	TS TS[]	원래 시계열에 값의 누적 합계가 있는 시계열을 반환합니다. **Note** CloudWatch 경보에 이 함수를 사용하지 않는 것이 좋습니다. 경보가 상태 변경 여부를 평가할 때마다 CloudWatch는 평가 기간으로 지정된 수보다 더 높은 수의 데이터 요소를 검색하려고 합니다. 이 함수는 추가 데이터가 요청될 때 다르게 작동합니다.	RUNNING_SUM([m1,m2])	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
SEARCH	검색 표현식	하나 이상의 TS	지정한 검색 기준과 일치하는 시계열을 하나 이상 반환합니다. SEARCH 함수를 사용하면 표현식 하나로 그래프에 관련된 시계열을 여러 개 추가할 수 있습니다. 나중에 추가되고 검색 기준과 일치하는 새 지표를 포함하도록 그래프가 동적으로 업데이트됩니다. 자세한 내용은 그래프에서 검색 표현식 사용 단원을 참조하십시오. SEARCH 표현식을 기반으로 경보를 생성할 수 없습니다. 검색 표현식은 여러 시계열을 반환하고 수학 표현식 기반 경보는 하나의 시계열만 관찰할 수 있기 때문입니다. CloudWatch 크로스 계정 관측성의 모니터링 계정에 로그인한 경우 SEARCH 함수는 소스 계정과 모니터링 계정에서 지표를 찾습니다.		✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
SERVICE_QUOTA	사용량 지표인 TS	TS	지정된 사용량 지표에 대한 서비스 할당량을 반환합니다. 이 함수를 사용하여 현재 사용량을 할당량과 비교하는 방법을 시각화하고 할당량에 접근할 때 경고하는 경보를 설정할 수 있습니다. 자세한 내용은 AWS 사용량 지표 단원을 참조하십시오.		✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
SLICE	(TS[], S, S) 또는 (TS[], S)	TS[] TS	시계열 배열의 일부를 검색합니다. SORT와 결합할 때 특히 유용합니다. 예를 들어 시계열 배열에서 상위 결과를 제외할 수 있습니다. 2개의 스칼라 인수를 사용하여 반환하려는 시계열 집합을 정의할 수 있습니다. 2개의 스칼라는 반환할 배열의 시작(포함)과 끝(제외)을 정의합니다. 배열은 0을 기반으로 인덱싱되므로 배열의 첫 번째 시계열은 시계열 0입니다. 또는 값을 하나만 지정할 수 있습니다. 그러면 CloudWatch는 해당 값으로 시작하는 모든 시계열을 반환합니다.	SLICE(SORT(METRICS(), SUM, DESC), 0, 10)는 요청의 지표 배열에서 SUM 값이 가장 높은 10개의 지표를 반환합니다. SLICE(SORT(METRICS(), AVG, ASC), 5)는 AVG 통계별로 지표 배열을 정렬한 다음 AVG가 가장 낮은 5를 제외한 모든 시계열을 반환합니다.	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
SORT	(TS[], FUNCT SORT_(R) (TS[], FUNCT SORT_(R, S)	TS[]	지정한 함수에 따라 시계열 배열을 정렬합니다. 사용하는 함수는 AVG, MIN, MAX 또는 SUM일 수 있습니다. 정렬 순서는 가장 낮은 값을 먼저 정렬하는 오름차순의 ASC 또는 더 높은 값을 먼저 정렬하는 DESC일 수 있습니다. 선택적으로 정렬 순서 뒤에 제한 역할을 하는 숫자를 지정할 수 있습니다. 예를 들어 제한을 5로 지정하면 정렬에서 상위 5개의 시계열만 반환됩니다. 이 수학 함수가 그래프에 표시되면 그래프의 각 지표에 대한 레이블도 정렬되고 번호가 매겨집니다.	SORT(METRICS(), AVG, DESC, 10)은 각 시계열의 평균값을 계산하고, 정렬의 시작 부분에서 가장 높은 값을 가진 시계열을 정렬하고, 평균이 가장 높은 10개의 시계열만 반환합니다. SORT(METRICS(), MAX, ASC)는 MAX 통계별로 지표 배열을 정렬한 다음 모두 오름차순으로 반환합니다.	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
STDDEV	TS TS[]	S TS	단일 시계열의 STDDEV는 지표의 모든 데이터 요소의 표준편차를 나타내는 스칼라를 반환합니다. 시계열 배열의 STDDEV 는 단일 시계열을 반환합니다. **Note** CloudWatch 경보에 이 함수를 사용하지 않는 것이 좋습니다. 예를 들어 `STDDEV(m2)` 의 경우 경보가 상태 변경 여부를 평가할 때마다 CloudWatch는 평가 기간으로 지정된 수보다 더 높은 수의 데이터 포인트를 검색하려고 합니다. 이 함수는 추가 데이터가 요청될 때 다르게 작동합니다.	m1/STDDEV(m1) STDDEV(METRICS())	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
SUM	TS TS[]	S TS	단일 시계열의 SUM은 지표의 모든 데이터 요소 값의 합을 나타내는 스칼라를 반환합니다. 시계열 배열의 SUM 은 단일 시계열을 반환합니다. **Note** CloudWatch 경보에 이 함수를 사용하지 않는 것이 좋습니다. 예를 들면 `SUM(m1)`입니다. 경보가 상태 변경 여부를 평가할 때마다 CloudWatch는 평가 기간으로 지정된 수보다 더 높은 수의 데이터 요소를 검색하려고 합니다. 이 함수는 추가 데이터가 요청될 때 다르게 작동합니다.	SUM(METRICS())/SUM(m1) SUM([m1,m2]) SUM(METRICS("errors"))/SUM(METRICS("requests"))*100	✓

함수	인수	반환 유형*	설명	예제	교차 계정에 대해 지원됩니까?
TIME_SERIES	S	TS	모든 값이 스칼라 인수로 설정된 비희소 시계열을 반환합니다.	TIME_SERIES(MAX(m1)) TIME_SERIES(5*AVG(m1)) TIME_SERIES(10)	✓

*스칼라 수만 반환하는 함수만 사용하는 것은 유효하지 않습니다. 표현식의 모든 최종 결과가 단일 시계열 또는 시계열 배열이어야 하기 때문입니다. 이러한 함수는 시계열을 반환하는 더 큰 표현식의 일부로 사용하세요.

IF 표현식 사용

IF를 비교 연산자와 함께 사용하여 시계열에서 데이터 요소를 필터링하거나 수집된 여러 시계열로 구성된 혼합 시계열을 생성합니다.

IF는 다음 인수를 사용합니다.

```
IF(condition, trueValue, falseValue)
```

조건은 조건 데이터 요소의 값이 0이면 FALSE로, 조건 값이 양수인지 음수인지 여부에 관계없이 다른 값이면 TRUE로 평가됩니다. 조건이 시계열이면 모든 타임스탬프에 대해 개별적으로 평가됩니다.

다음은 유효한 구문 목록입니다. 이러한 구문 각각에 대한 출력은 단일 시계열입니다.

- IF(TS ***Comparison Operator*** S, S | TS, S | TS)

> **Note**
>
> `TS comparison operator S`가 TRUE이지만 해당하는 데이터 포인트가 `metric2`에 없는 경우 출력은 0이 됩니다.

- IF(TS, TS, TS)
- IF(TS, S, TS)
- IF(TS, TS, S)
- IF(TS, S, S)
- IF(S, TS, TS)

다음 단원에서는 이러한 구문에 대한 자세한 내용과 예제를 제공합니다.

IF(TS ***Comparison Operator*** S, scalar2 | metric2, scalar3 | metric3)

해당 출력 시계열 값:

- TS ***## ###*** S가 TRUE인 경우 값은 scalar2 또는 metric2입니다.
- TS ***## ###*** S가 FALSE인 경우 값은 scalar3 또는 metric3입니다.
- TS ***## ###***가 TRUE이고 metric2의 해당 데이터 요소가 존재하지 않는 경우 값은 0입니다.
- TS ***## ###***가 FALSE이고 metric3의 해당 데이터 요소가 존재하지 않는 경우 값은 0입니다.
- 해당 데이터 요소가 metric3에 없거나 scalar3 /metric3이 이 표현식에서 생략된 경우 빈 시계열입니다.

IF(metric1, metric2, metric3)

metric1의 각 데이터 요소의 경우, 해당 출력 시계열 값:

- metric1의 해당 데이터 요소가 TRUE인 경우 값은 metric2입니다.
- metric1의 해당 데이터 요소가 FALSE인 경우 값은 metric3입니다.
- metric1의 해당 데이터 요소가 TRUE이고 해당 데이터 요소가 metric2에 없는 경우 값은 0입니다.
- metric1의 해당 데이터 요소가 FALSE이고 해당 데이터 요소가 metric3에 없는 경우 삭제됩니다.
- metric1의 해당 데이터 요소가 FALSE이고 metric3이 표현식에서 생략된 경우 삭제됩니다.
- metric1의 해당 데이터 요소가 없는 경우 삭제됩니다.

다음 표에서는 이 구문의 예제를 보여줍니다.

지표 또는 함수	값
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
(metric3)	[0, 0, 20, -, 20]
IF(metric1, metric2, metric3)	[30, 0, 20, 0, -]

IF(metric1, scalar2, metric3)

metric1의 각 데이터 요소의 경우, 해당 출력 시계열 값:

- metric1의 해당 데이터 요소가 TRUE인 경우 값은 scalar2입니다.
- metric1의 해당 데이터 요소가 FALSE인 경우 값은 metric3입니다.
- metric1의 해당 데이터 요소가 FALSE이고 해당 데이터 요소가 metric3에 없는 경우 또는 metric3이 표현력에서 생략된 경우 삭제됩니다.

지표 또는 함수	값
(metric1)	[1, 1, 0, 0, -]
scalar2	5
(metric3)	[0, 0, 20, -, 20]
IF(metric1, scalar2, metric3)	[5, 5, 20, -, -]

IF(metric1, metric2, scalar3)

metric1의 각 데이터 요소의 경우, 해당 출력 시계열 값:

- metric1의 해당 데이터 요소가 TRUE인 경우 값은 metric2입니다.
- metric1의 해당 데이터 요소가 FALSE인 경우 값은 scalar3입니다.

- metric1의 해당 데이터 요소가 TRUE이고 해당 데이터 요소가 metric2에 없는 경우 값은 0입니다.
- metric1의 해당 데이터 요소가 없으면 삭제됩니다.

지표 또는 함수	값
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
scalar3	5
IF(metric1, metric2, scalar3)	[30, 0, 5, 5, -]

IF(scalar1, metric2, metric3)

해당 출력 시계열 값:

- scalar1이 TRUE인 경우 값은 metric2입니다.
- scalar1이 FALSE인 경우 값은 metric3입니다.
- metric3이 표현식에서 생략된 경우 빈 시계열입니다.

IF 표현식의 사용 사례 예

다음 예제에서는 IF 함수의 가능한 용도를 보여줍니다.

- 지표의 하한값만 표시하려면:

  IF(metric1<400, metric1)
- 지표의 각 데이터 요소를 두 값 중 하나로 변경하고 원래 지표의 상대적 상한값 및 하한값을 표시하려면:

  IF(metric1<400, 10, 2)
- 지연 시간이 임계값을 초과하는 각 타임 스탬프에 대해 1을 표시하고 다른 모든 데이터 요소에 대해 0을 표시하려면:

  IF(latency>threshold, 1, 0)

GetMetricData API 작업에서 지표 수학 사용

`GetMetricData`를 사용하여 수학 표현식을 사용하는 계산을 수행할 수 있을 뿐만 아니라 하나의 API 호출에서 대규모 지표 데이터 배치를 검색할 수도 있습니다. 자세한 내용은 GetMetricData 단원을 참조하세요.

지표 수학에 대한 이상 탐지

지표 수학에 대한 이상 탐지는 단일 지표와 지표 수학 표현식의 출력에 대한 이상 탐지 경보를 만드는 데 사용할 수 있는 기능입니다. 이러한 표현식을 사용하여 이상 탐지 밴드를 시각화하는 그래프를 만들 수 있습니다. 이 기능은 기본 산술 함수, 비교 및 논리 연산자 및 대부분의 기타 함수를 지원합니다.

지표 수학에 대한 이상 탐지는 다음 함수를 지원하지 않습니다.

- 같은 줄에 둘 이상의 ANOMALY_DETECTION_BAND를 포함하는 표현식.
- 10개 이상의 지표 또는 수학 표현식을 포함하는 표현식.
- METRICS 표현식을 포함하는 표현식.
- SEARCH 함수를 포함하는 표현식.
- DP_PERF_INSIGHTS 함수를 사용하는 식입니다.
- 기간이 다른 지표를 사용하는 표현식.
- 고분해능 지표를 입력하는 지표 수학 이상 탐지기.

이 기능에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 이상 탐지 사용을 참조하세요.

그래프에서 검색 표현식 사용

검색 표현식은 CloudWatch 그래프에 추가할 수 있는 수학 표현식 유형입니다. 검색 표현식을 사용하면 관련된 여러 개의 지표를 그래프에 빠르게 추가할 수 있습니다. 또한 처음 그래프를 생성할 때 지표가 없는 경우에도 적절한 지표를 표시에 자동으로 추가하는 동적 그래프를 생성할 수 있습니다.

예를 들어, 리전의 모든 인스턴스에 대해 `AWS/EC2 CPUUtilization` 지표를 표시하는 검색 표현식을 생성할 수 있습니다. 나중에 새 인스턴스를 시작하면 새 인스턴스의 `CPUUtilization`이 그래프에 자동으로 추가됩니다.

그래프에 검색 표현식을 사용하면 검색은 지표 이름, 네임스페이스, 측정기준 이름 및 측정기준 값에서 검색 표현식을 찾습니다. 더 복잡하고 강력한 검색을 위해 부울 연산자를 사용할 수 있습니다. 검색 표현식은 지난 2주 이내에 데이터를 보고한 지표만 찾을 수 있습니다.

SEARCH 표현식을 기반으로 경보를 생성할 수 없습니다. 검색 표현식은 여러 시계열을 반환하고 수학 표현식 기반 경보는 하나의 시계열만 관찰할 수 있기 때문입니다.

CloudWatch 크로스 계정 관측성에서 모니터링 계정을 사용하는 경우 검색 표현식은 해당 모니터링 계정에 연결된 소스 계정에서 지표를 찾을 수 있습니다.

주제

CloudWatch 검색 표현식 구문

유효한 검색 표현식의 형식은 다음과 같습니다.

```
SEARCH(' {Namespace, DimensionName1, DimensionName2, ...} SearchTerm', 'Statistic')
```

예:

```
SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization"', 'Average')
```

- SEARCH 단어 뒤에 중괄호로 묶여 있는 쿼리의 첫 번째 부분은 검색할 지표 스키마입니다. 지표 스키마에는 하나의 지표 네임스페이스와 하나 이상의 측정기준 이름이 포함됩니다. 검색 쿼리에 지표 스키마를 포함하는 것은 선택 사항입니다. 지정된 경우 지표 스키마에는 네임스페이스가 있어야 하며, 해당 네임스페이스에서 유효한 하나 이상의 측정기준 이름을 선택적으로 포함할 수 있습니다.

  네임스페이스 또는 측정기준 이름에 공백이나 영숫자가 아닌 문자가 포함되지 않는 한 지표 스키마 내에 따옴표를 사용할 필요가 없습니다. 따옴표를 사용해야 하는 경우 해당 문자가 포함된 이름을 큰 따옴표로 묶어야 합니다.
- SearchTerm도 선택 사항이지만, 유효한 검색에는 지표 스키마, SearchTerm 또는 둘 다 포함되어야 합니다. 일반적으로 SearchTerm에는 하나 이상의 계정 ID, 지표 이름 또는 차원 값이 포함되어 있습니다. SearchTerm에는 부분 일치 및 정확한 일치로 검색할 용어가 여러 개 포함될 수 있습니다. 또한 부울 연산자도 포함될 수 있습니다.

  SearchTerm에서 계정 ID 사용은 CloudWatch 크로스 계정 관측성을 위한 모니터링 계정으로 설정된 계정에서만 가능합니다. SearchTerm의 계정 ID 구문은 :aws.AccountId =

"*444455556666*"입니다. `'LOCAL'`을 사용하여 `:aws.AccountId = 'LOCAL'`과 같이 모니터링 계정 자체를 지정할 수도 있습니다.

자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

`SearchTerm`에는 하나 이상의 지정자가 포함될 수 있지만(이 예에서는 `MetricName=`), 지정자를 사용하는 것은 필수 사항이 아닙니다.

지표 스키마 및 `SearchTerm`은 작은따옴표 쌍으로 함께 묶여야 합니다.

- `Statistic`은 유효한 CloudWatch 통계의 이름입니다. 이는 작은따옴표로 묶여야 합니다. 자세한 내용은 Statistics 단원을 참조하십시오.

위의 예제에서는 측정기준 이름이 `AWS/EC2`인 지표에 대한 `InstanceId` 네임스페이스를 검색합니다. 찾은 모든 `CPUUtilization` 지표를 반환하며 그래프에는 `Average` 통계가 표시됩니다.

검색 표현식은 지난 2주 이내에 데이터를 보고한 지표만 찾을 수 있습니다.

검색 표현식 제한

검색 표현식 쿼리의 최대 크기는 1024자입니다. 그래프 하나에 무려 100개의 검색 표현식이 있을 수 있습니다. 그래프에는 최대 500개의 시계열이 표시될 수 있습니다.

CloudWatch 검색 표현식: 토큰화

`SearchTerm`을 지정하면 검색 함수는 CloudWatch가 전체 지표 이름, 측정기준 이름, 측정기준 값, 네임스페이스에서 자동으로 생성하는 하위 문자열인 '토큰'을 검색합니다. CloudWatch는 원래 문자열에서 낙타 대문자로 구분되는 토큰을 생성합니다. 또한 숫자 문자는 새 토큰의 시작 부분으로 사용되며, 영숫자가 아닌 문자는 구분 기호로 사용되며, 영숫자가 아닌 문자 앞/뒤에는 토큰을 생성합니다.

동일한 유형의 토큰 구분 기호로 이루어진 연속 문자열이 한 개의 토큰이 됩니다.

생성된 토큰은 모두 소문자입니다. 다음 표에는 생성된 토큰의 일부 예가 나와 있습니다.

원래 문자열	생성된 토큰
CustomCount1	`customcount1` , `custom`, `count`, `1`
SDBFailure	`sdbfailure` , `sdb`, `failure`

원래 문자열	생성된 토큰
Project2-trial333	`project2trial333` , `project`, `2`, `trial`, `333`

CloudWatch 검색 표현식: 부분 일치

`SearchTerm`을 지정하면 검색어도 토큰화됩니다. CloudWatch는 검색어에서 생성된 단일 토큰이 지표 이름, 네임스페이스, 측정기준 이름 또는 측정기준 값에서 생성된 단일 토큰과 일치하는 부분 일치에 따라 지표를 찾습니다.

단일 토큰과 일치하는 부분 일치 검색은 대/소문자를 구별하지 않습니다. 예를 들어, 다음 검색어를 사용하면 `CustomCount1` 지표가 반환될 수 있습니다.

- `count`
- `Count`
- `COUNT`

그러나 `couNT`를 검색어로 사용하면 검색어 `couNT`의 대문자가 `cou` 및 `NT`로 토큰화되므로 `CustomCount1`을 찾지 않습니다.

검색 시 원래 이름에 연속으로 나타나는 여러 토큰인 복합 토큰과도 일치할 수 있습니다. 복합 토큰과 일치시키기 위해 검색 시 대/소문자를 구별합니다. 예를 들어, 원래 용어가 `CustomCount1`인 경우 `CustomCount` 또는 `Count1`에 대한 검색은 성공하지만 `customcount` 또는 `count1`에 대한 검색은 실패합니다.

CloudWatch 검색 표현식: 정확한 일치

정확하게 일치해야 하는 검색어 부분을 큰따옴표로 묶어 검색어와 정확하게 일치하는 용어만 찾도록 검색을 정의할 수 있습니다. 이러한 큰따옴표는 전체 검색어에 주위에 사용되는 작은따옴표로 묶여 있습니다. 예를 들어, **`SEARCH(' {MyNamespace}, "CustomCount1" ', 'Maximum')`**는 이름이 `CustomCount1`인 네임스페이스에 지표 이름, 측정기준 이름 또는 측정기준 값으로 있는 경우 정확한 문자열 `MyNamespace`을 찾습니다. 그러나 **`SEARCH(' {MyNamespace}, "customcount1" ', 'Maximum')`** 또는 **`SEARCH(' {MyNamespace}, "Custom" ', 'Maximum')`** 검색은 이 문자열을 찾지 않습니다.

단일 검색 표현식에서 부분 일치 용어와 정확한 일치 용어를 결합할 수 있습니다. 예를 들어 **`SEARCH(' {AWS/NetworkELB, LoadBalancer} "ConsumedLCUs" OR flow ',`**

'Maximum') 표현식은 ConsumedLCUs라는 Elastic Load Balancing 지표를 반환하는 것은 물론 flow 토큰이 포함된 모든 Elastic Load Balancing 지표 또는 측정기준을 반환합니다.

또한 다음 예제와 같이 영숫자가 아닌 문자 또는 공백 같은 특수 문자가 있는 이름을 찾을 때 정확한 일치를 사용하는 것이 좋습니다.

```
SEARCH(' {"My Namespace", "Dimension@Name"}, "Custom:Name[Special_Characters" ',
 'Maximum')
```

CloudWatch 검색 표현식: 지표 스키마 제외

지금까지 살펴본 모든 예제의 지표 스키마는 중괄호 안에 포함되어 있습니다. 지표 스키마를 생략하는 검색도 유효합니다.

예를 들어, **SEARCH(' "CPUUtilization" ', 'Average')**는 CPUUtilization 문자열과 정확하게 일치하는 모든 지표 이름, 측정기준 이름, 측정기준 값 및 네임스페이스를 반환합니다. AWS 지표 네임스페이스에서는 여기에 Amazon EC2, Amazon ECS, SageMaker 및 기타 서비스를 비롯한 여러 서비스의 지표가 포함될 수 있습니다.

이 검색 범위를 AWS 서비스 하나만으로 좁히려면 다음 예제와 같이 지표 스키마에 네임스페이스와 필요한 측정기준을 지정하는 것이 좋습니다. 이렇게 하면 검색 범위가 AWS/EC2 네임스페이스로 좁혀지지만 CPUUtilization을 해당 지표의 측정기준 값으로 정의한 경우 다른 지표의 결과가 계속 반환됩니다.

```
SEARCH(' {AWS/EC2, InstanceType} "CPUUtilization" ', 'Average')
```

또는 다음 예제와 같이 SearchTerm에 네임스페이스를 추가할 수 있습니다. 그러나 이 예제에서는 검색 시 사용자 지정 측정기준 이름 또는 값인 경우에도 AWS/EC2 문자열과 일치합니다.

```
SEARCH(' "AWS/EC2" MetricName="CPUUtilization" ', 'Average')
```

CloudWatch 검색 표현식: 검색에서 속성 이름 지정

"CustomCount1"에 대해 다음의 정확한 일치 검색을 수행하면 해당 이름과 정확하게 일치하는 모든 지표를 반환합니다.

```
SEARCH(' "CustomCount1" ', 'Maximum')
```

그러나 CustomCount1의 측정기준 이름, 측정기준 값 또는 네임스페이스가 있는 지표도 반환합니다. 검색을 자세히 구성하려면 검색에서 찾으려는 객체 유형의 속성 이름을 지정하면 됩니다. 다음 예제는 모든 네임스페이스를 검색하고 이름이 CustomCount1인 지표를 반환합니다.

```
SEARCH(' MetricName="CustomCount1" ', 'Maximum')
```

다음 예제와 같이 네임스페이스와 측정기준 이름/값 페어를 속성 이름으로 사용할 수도 있습니다. 이러한 예제의 첫 번째는 부분 일치 검색에도 속성 이름을 사용할 수 있음을 보여줍니다.

```
SEARCH(' InstanceType=micro ', 'Average')
```

```
SEARCH(' InstanceType="t2.micro" Namespace="AWS/EC2" ', 'Average')
```

CloudWatch 검색 표현식: 영숫자가 아닌 문자

영숫자가 아닌 문자는 구분 기호로 사용되며, 지표, 측정기준, 네임스페이스 및 검색어의 이름을 토큰으로 구분할 위치를 표시합니다. 용어가 토큰화되면 영숫자가 아닌 문자가 제거되며 토큰에 표시되지 않습니다. 예를 들어, Network-Errors_2는 network, errors 및 2 토큰을 생성합니다.

검색어에 영숫자가 아닌 문자가 포함될 수 있습니다. 검색어에 이러한 문자가 표시되면 해당 문자는 부분 일치에서 복합 토큰을 지정할 수 있습니다. 예를 들어, 다음 검색에서는 이름이 Network-Errors-2 또는 NetworkErrors2인 지표를 모두 찾습니다.

```
network/errors
network+errors
network-errors
Network_Errors
```

정확한 값 검색을 수행할 경우 정확한 검색에 사용된 영숫자가 아닌 문자는 검색할 문자열에 표시되는 정확한 문자여야 합니다. 예를 들어, Network-Errors-2를 찾으려는 경우 "Network-Errors-2"에 대한 검색은 성공하지만 "Network_Errors_2"에 대한 검색은 실패합니다.

정확한 일치 검색을 수행할 경우 다음 문자는 백슬래시로 이스케이프되어야 합니다.

```
" \ ( )
```

예를 들어, 정확한 일치로 이름이 Europe\France Traffic(Network)인 지표를 찾으려면 **"Europe\\France Traffic\(Network\)"** 검색어를 사용해야 합니다.

CloudWatch 검색 표현식: 부울 연산자

검색 시 SearchTerm 내에 부울 연산자 AND, OR 및 NOT을 사용할 수 있습니다. 부울 연산자는 전체 검색어를 묶을 때 사용하는 작은따옴표로 묶습니다. 부울 연산자는 대/소문자를 구별하므로 and, or 및 not은 부울 연산자로 사용할 수 없습니다.

SEARCH('{AWS/EC2,InstanceId} network AND packets', 'Average') 같이 검색에서 AND를 명시적으로 사용할 수 있습니다. 검색어 사이에 부울 연산자를 사용하지 않으면 암시적으로 AND 연산자가 있는 것처럼 검색합니다. 따라서 **SEARCH(' {AWS/EC2,InstanceId} network packets ', 'Average')**의 경우 동일한 검색 결과가 발생합니다.

결과에서 데이터의 하위 집합을 제외하려면 NOT을 사용합니다. 예를 들어, **SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT i-1234567890123456 ', 'Average')**는 i-1234567890123456 인스턴스를 제외한 모든 인스턴스에 대해 CPUUtilization을 반환합니다. NOT 절을 유일한 검색어로 사용할 수도 있습니다. 예를 들어, **SEARCH('NOT Namespace=AWS ', 'Maximum')**는 사용자 지정 지표(AWS가 포함되지 않은 네임스페이스가 있는 지표)를 모두 생성합니다.

쿼리에 여러 NOT 구문을 사용할 수 있습니다. 예를 들어, **SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT "ProjectA" NOT "ProjectB" ', 'Average')**는 리전에서 CPUUtilization 또는 ProjectA의 측정기준 값이 있는 것을 제외한 모든 인스턴스의 ProjectB을 반환합니다.

다음 예제와 같이 더욱 강력하고 상세한 검색을 위해 부울 연산자를 결합할 수 있습니다. 연산자를 그룹화하려면 괄호를 사용합니다.

다음의 두 예제는 EC2 및 EBS 네임스페이스 둘 다의 ReadOps이 포함된 지표 이름을 모두 반환합니다.

```
SEARCH(' (EC2 OR EBS) AND MetricName=ReadOps ', 'Maximum')
```

```
SEARCH(' (EC2 OR EBS) MetricName=ReadOps ', 'Maximum')
```

다음 예제는 이전 검색을 ProjectA(측정기준 값일 수 있음)만 포함된 결과로 범위를 좁힙니다.

```
SEARCH(' (EC2 OR EBS) AND ReadOps AND ProjectA ', 'Maximum')
```

다음 예제는 중첩 그룹화를 사용합니다. 모든 함수의 Errors 및 이름에 ProjectA 또는 ProjectB 문자열이 포함된 함수의 Invocations에 대한 Lambda 지표를 반환합니다.

```
SEARCH(' {AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND
 (ProjectA OR ProjectB)) ', 'Average')
```

CloudWatch 검색 표현식: 수학 표현식 사용

그래프의 수학 표현식 내에서 검색 표현식을 사용할 수 있습니다.

예를 들어 **SUM(SEARCH(' {AWS/Lambda, FunctionName} MetricName="Errors" ', 'Sum'))** 표현식은 모든 Lambda 함수의 Errors 지표에 대한 합계를 반환합니다.

검색 표현식과 수학 표현식에 별도의 줄을 사용하면 더 유용한 결과를 얻을 수 있습니다. 예를 들어, 그래프에서 다음의 두 표현식을 사용한다고 가정해 보겠습니다. 첫 번째 줄은 각 Lambda 함수에 대한 별도의 Errors 줄을 표시합니다. 이 표현식의 ID는 e1입니다. 두 번째 줄은 모든 함수의 오류 합계를 표시하는 다른 줄을 추가합니다.

```
SEARCH(' {AWS/Lambda, FunctionName}, MetricName="Errors" ', 'Sum')
SUM(e1)
```

CloudWatch 검색 표현식 예

다음 예제는 더 많은 검색 표현식 사용과 구문을 보여줍니다. 리전의 모든 인스턴스에 대한 CPUUtilization을 검색한 후 변형을 살펴보겠습니다.

이 예제는 리전의 각 인스턴스에 대한 줄 하나를 표시합니다. 이 줄은 AWS/EC2 네임스페이스의 CPUUtilization 지표를 보여줍니다.

```
SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" ', 'Average')
```

InstanceId를 InstanceType으로 변경하면 리전에서 사용되는 각 인스턴스 유형마다 한 줄씩 표시되도록 그래프가 변경됩니다. 각 유형의 모든 인스턴스에 있는 데이터는 해당 인스턴스 유형에 대해 한 줄로 집계됩니다.

```
SEARCH(' {AWS/EC2,InstanceType} MetricName="CPUUtilization" ', 'Average')
```

다음 예제는 인스턴스 유형별로 CPUUtilization을 집계하고 micro 문자열이 포함된 각 인스턴스 유형에 대해 한 줄을 표시합니다.

```
SEARCH('{AWS/EC2,InstanceType} InstanceType=micro MetricName="CPUUtilization" ',
 'Average')
```

이 예제는 이전 예제의 범위를 좁히며, InstanceType을 t2.micro 인스턴스에 대한 정확한 검색으로 변경합니다.

```
SEARCH('{AWS/EC2,InstanceType} InstanceType="t2.micro" MetricName="CPUUtilization" ',
 'Average')
```

다음 검색은 쿼리의 {metric schema} 부분을 제거하므로 모든 네임스페이스의 CPUUtilization 지표가 그래프에 표시됩니다. 여러 측정기준으로 집계된 각 CPUUtilization 서비스의 AWS 지표에 대해 여러 줄이 그래프에 포함되어 있으므로 상당히 많은 결과가 반환될 수 있습니다.

```
SEARCH('MetricName="CPUUtilization" ', 'Average')
```

이러한 결과를 약간 좁히려면 두 개의 특정 지표 네임스페이스를 지정하면 됩니다.

```
SEARCH('MetricName="CPUUtilization" AND ("AWS/ECS" OR "AWS/ES") ', 'Average')
```

각 쿼리에서 지표 스키마를 하나만 지정할 수 있으므로, 위의 예제는 검색 쿼리 하나로 여러 개의 특정 네임스페이스를 검색하는 유일한 방법입니다. 그러나 더 많은 구조를 추가하기 위해 다음 예제와 같이 그래프에서 쿼리 두 개를 사용할 수 있습니다. 또한 이 예에서는 Amazon ECS에 대한 데이터를 집계하는 데 사용할 측정기준을 지정하여 더 많은 구조를 추가합니다.

```
SEARCH('{AWS/ECS ClusterName}, MetricName="CPUUtilization" ', 'Average')
SEARCH(' {AWS/EBS} MetricName="CPUUtilization" ', 'Average')
```

다음 예에서는 ConsumedLCUs라는 Elastic Load Balancing 지표를 반환하는 것은 물론 flow 토큰이 포함된 모든 Elastic Load Balancing 지표 또는 측정기준을 반환합니다.

```
SEARCH('{AWS/NetworkELB, LoadBalancer} "ConsumedLCUs" OR flow ', 'Maximum')
```

다음 예제는 중첩 그룹화를 사용합니다. 모든 함수의 Errors 및 이름에 ProjectA 또는 ProjectB 문자열이 포함된 함수의 Invocations에 대한 Lambda 지표를 반환합니다.

```
SEARCH('{AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND
 (ProjectA OR ProjectB)) ', 'Average')
```

다음 예제는 AWS 서비스에서 생성된 지표를 제외한 모든 사용자 지정 지표를 표시합니다.

```
SEARCH('NOT Namespace=AWS ', 'Average')
```

다음 예제는 이름의 일부로 Errors 문자열을 포함하는 지표 이름, 네임스페이스, 측정기준 이름 및 측정기준 값이 있는 지표를 표시합니다.

```
SEARCH('Errors', 'Average')
```

다음 예제는 정확한 일치로 검색 범위를 좁힙니다. 예를 들어, 이 검색은 Errors 지표 이름을 찾지만 이름이 ConnectionErrors 또는 errors인 지표를 찾지 않습니다.

```
SEARCH(' "Errors" ', 'Average')
```

다음 예제는 검색어의 지표 스키마 부분에 공백이나 특수 문자가 포함된 이름을 지정하는 방법을 보여줍니다.

```
SEARCH('{"Custom-Namespace", "Dimension Name With Spaces"}, ErrorCount ', 'Maximum')
```

CloudWatch 크로스 계정 관측성 검색 표현식 예제

CloudWatch 크로스 계정 관측성 예제

CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정에 로그인한 경우 SEARCH 함수를 사용하여 지정된 소스 계정에서 지표를 반환할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

다음 예제에서는 계정 ID가 111122223333인 계정에서 모든 Lambda 지표를 검색합니다.

```
SEARCH(' AWS/Lambda :aws.AccountId = "111122223333" ', 'Average')
```

다음 예제에서는 111122223333 및 777788889999의 두 계정에서 모든 AWS/EC2 지표를 검색합니다.

```
SEARCH(' AWS/EC2 :aws.AccountId = ("111122223333" OR "777788889999") ', 'Average')
```

다음 예에서는 소스 계정 111122223333과 모니터링 계정 자체에서 모든 AWS/EC2 지표를 검색합니다.

```
SEARCH(' AWS/EC2 :aws.AccountId = ("111122223333" OR 'LOCAL') ', 'Average')
```

다음 예에서는 InstanceId 측정기준이 있는 계정 444455556666에서 MetaDataToken 지표 중 SUM을 검색합니다.

```
SEARCH('{AWS/EC2,InstanceId} :aws.AccountId=444455556666 MetricName=\"MetadataNoToken
\"','Sum')
```

검색 표현식을 사용하여 CloudWatch 그래프 생성

CloudWatch 콘솔에서 대시보드에 그래프를 추가할 때 또는 [지표(Metrics)] 보기를 사용하여 검색 기능에 액세스할 수 있습니다.

SEARCH 표현식을 기반으로 경보를 생성할 수 없습니다. 검색 표현식은 여러 시계열을 반환하고 수학 표현식 기반 경보는 하나의 시계열만 관찰할 수 있기 때문입니다.

기존 대시보드에 검색 표현식이 있는 그래프를 추가하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택하고 대시보드를 선택합니다.
3. 위젯 추가를 선택합니다.
4. 행 또는 누적 면적을 선택하고 구성을 선택합니다.
5. 그래프로 표시된 지표 탭에서 Add a math expression(수학 표현식 추가)을 선택합니다.
6. 세부 정보에 원하는 검색 표현식을 입력합니다. 예: **`SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization"', 'Average')`**
7. (선택 사항) 그래프에 다른 검색 표현식 또는 수학 표현식을 추가하려면 수학 표현식 추가(Add a math expression)를 선택합니다.
8. (선택 사항) 검색 표현식을 추가한 후 그래프 범례에서 각 지표에 대해 표시할 동적 레이블을 지정할 수 있습니다. 동적 레이블은 지표에 대한 통계를 표시하고 대시보드 또는 그래프를 새로 고칠 때 자동으로 업데이트됩니다. 동적 레이블을 추가하려면 그래프로 표시된 지표를 선택한 다음 동적 레이블(Dynamic labels)을 선택합니다.

   기본적으로 레이블에 추가하는 동적 값은 레이블 시작 부분에 나타납니다. 그런 다음 지표에 대한 레이블 값을 클릭하여 레이블을 편집할 수 있습니다. 자세한 내용은 동적 레이블 사용 단원을 참조하십시오.
9. (선택 사항) 그래프에 단일 지표를 추가하려면 모든 지표 탭을 선택하고 원하는 지표로 드릴다운합니다.
10. (선택 사항) 그래프에 표시된 시간 범위를 변경하려면 그래프 맨 위의 사용자 지정이나 사용자 지정 왼쪽에 있는 기간 중 하나를 선택합니다.

11. (선택 사항) 가로 주석을 사용하면 대시보드 사용자는 지표가 특정 레벨까지 급상승하는지 아니면 지표가 사전 정의된 범위 내에 있는지 여부를 빠르게 확인할 수 있습니다. 가로 주석을 추가하려면 그래프 옵션을 선택한 후 가로 주석 추가를 선택합니다.

    a. 레이블에 주석의 레이블을 입력합니다.

    b. 값에 가로 주석이 표시될 지표 값을 입력합니다.

    c. Fill(채우기)에서 이 주석에 채우기 셰이딩을 사용할지 여부를 지정합니다. 예를 들어 채울 영역에 대해 `Above` 또는 `Below`를 선택합니다. `Between`을 지정할 경우 다른 `Value` 필드가 표시되며, 두 값 사이의 그래프 영역이 채워집니다.

    d. 그래프에 여러 지표가 포함된 경우 축에서 `Value`의 숫자가 왼쪽 Y축과 연결된 지표를 참조하는지 아니면 오른쪽 Y축과 연결된 지표를 참조하는지를 지정합니다.

    주석의 왼쪽 열에서 색상 정사각형을 선택하여 주석의 채우기 색상을 변경할 수 있습니다.

    동일한 그래프에 여러 가로 주석을 추가하려면 이들 단계를 반복합니다.

    주석을 숨기려면 해당 주석의 왼쪽 열에서 확인란을 선택 취소합니다.

    주석을 삭제하려면 [작업(Actions)] 열에서 [x]를 선택합니다.

12. (선택 사항) 세로 주석을 사용하면 그래프에 운영 이벤트나 배포의 시작과 끝과 같은 마일스톤을 표시하는 데 도움이 됩니다. 세로 주석을 추가하려면 그래프 옵션을 선택한 후 Add vertical annotation(세로 주석 추가)을 선택합니다.

    a. 레이블에 주석의 레이블을 입력합니다. 주석에 날짜와 시간만 표시하려면 레이블 필드를 비워둡니다.

    b. 날짜에서, 세로 주석이 표시되는 날짜와 시간을 지정합니다.

    c. Fill(채우기)에서 채우기 셰이딩을 세로 주석 앞에 또는 뒤에 사용할지 아니면 두 개의 세로 주석 사이에 사용할지 여부를 지정합니다. 예를 들어 채울 영역에 대해 `Before` 또는 `After`를 선택합니다. `Between`을 지정할 경우 다른 `Date` 필드가 표시되며, 두 값 사이의 그래프 영역이 채워집니다.

    동일한 그래프에 여러 세로 주석을 추가하려면 이들 단계를 반복합니다.

    주석을 숨기려면 해당 주석의 왼쪽 열에서 확인란을 선택 취소합니다.

    주석을 삭제하려면 [작업(Actions)] 열에서 [x]를 선택합니다.

13. 위젯 생성을 선택합니다.

14. 대시보드 저장을 선택합니다.

지표 보기를 사용하여 검색한 지표를 그래프로 작성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 검색 필드에 검색할 토큰을 입력합니다(예: **`cpuutilization t2.small`**).

   검색과 일치하는 결과가 표시됩니다.
4. 검색과 일치하는 지표를 모두 그래프로 작성하려면 Graph search(그래프 검색)를 선택합니다.

   또는

   검색을 구체화하려면 검색 결과에 표시된 네임스페이스 중 하나를 선택합니다.
5. 네임스페이스를 선택하여 결과를 좁힌 경우 다음을 수행할 수 있습니다.

   a. 하나 이상의 지표를 그래프 처리하려면 각 지표 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.
   b. 검색을 구체화하려면 지표 이름 위로 마우스를 이동하고 검색에 추가 또는 이 항목만 검색을 선택합니다.
   c. 지표에 대한 도움말을 보려면 지표 이름을 선택하고 이것은 무엇입니까?를 선택합니다.

   선택한 지표가 그래프에 표시됩니다.
6. (선택 사항) 검색어의 해당 부분을 편집하려면 검색 표시줄에 있는 버튼 중 하나를 선택합니다.
7. (선택 사항) 대시보드에 그래프를 추가하려면 작업을 선택한 후 대시보드에 추가를 선택합니다.

지표에 대한 통계 얻기

CloudWatch 통계 정의

통계는 지정한 기간에 걸친 지표 데이터 집계입니다. 지표에 대한 통계를 그래프로 표시하거나 검색할 때 각 통계 값을 계산하는 데 사용할 '기간'(예: 5분)을 지정합니다. 예를 들어 [기간(Period)]이 5분이면 [합계(Sum)]는 5분의 기간 동안 수집된 모든 샘플 값의 합계이며 [최솟값(Minimum)]은 5분의 기간 동안 수집된 가장 낮은 값입니다.

CloudWatch는 지표에 대해 다음 통계를 지원합니다.

- SampleCount, 즉 샘플 수는 기간 동안의 데이터 요소 수입니다.
- Sum, 즉 합계는 해당 기간 동안 수집된 모든 데이터 요소 값의 합계입니다.
- Average, 즉 평균은 지정된 기간 동안의 `Sum/SampleCount` 값입니다.
- Minimum, 즉 최솟값은 지정된 기간 동안 관찰된 가장 낮은 값입니다.
- Maximum, 즉 최댓값은 지정된 기간 동안 관찰된 가장 높은 값입니다.
- Percentile(p), 즉 백분위수는 데이터 집합에서 값의 상대적인 위치를 나타냅니다. 예를 들어 p95는 95번째 백분위수로, 기간 내 데이터의 95%가 이 값보다 낮으며 데이터의 5%가 이 값보다 높음을 의미합니다. 백분위수는 지표 데이터의 분포를 정확하게 이해하는 데 도움이 됩니다.
- Trimmed mean(TM), 절사 평균은 지정된 두 경계 사이에 있는 모든 값의 평균입니다. 평균을 계산할 때 경계 외부의 값은 무시됩니다. 경계는 0에서 100 사이의 숫자 하나 또는 두 개(소수점 이하 10자리까지)로 정의됩니다. 숫자는 절댓값 또는 백분율일 수 있습니다. 예를 들어 tm90은 가장 높은 값을 가진 데이터 요소의 10%를 제거한 후 평균을 계산합니다. TM(2%:98%)은 2%의 가장 낮은 데이터 요소와 2%의 가장 높은 데이터 요소를 제거한 후 평균을 계산합니다. TM(150:1000)은 150 이하이거나 1000을 초과하는 모든 데이터 요소를 제거한 후 평균을 계산합니다.
- Interquartile mean(IQM), 즉 사분위수 평균은 '사분위수 범위'의 절사 평균 또는 값의 중간 50%입니다. 이 값은 TM(25%:75%)과 같습니다.
- Winsorized mean(WM), 즉 윈저화 평균은 절사 평균과 유사합니다. 그러나 윈저화 평균을 사용하면 경계 외부에 있는 값이 무시되지 않으며 대신 해당 경계의 가장자리에 있는 값과 동일한 것으로 간주됩니다. 그리고 이 정규화 후에 평균이 계산됩니다. 경계는 0에서 100 사이의 숫자 하나 또는 두 개(소수점 이하 10자리까지)로 정의됩니다. 예를 들어 wm98은 가장 높은 값의 2%를 98번째 백분위수 값과 동일하게 처리하면서 평균을 계산합니다. WM(10%:90%)은 가장 높은 10%의 데이터 요소를 90% 경계의 값으로 처리하고 가장 낮은 10%의 데이터 요소를 10% 경계의 값으로 처리하면서 평균을 계산합니다.
- Percentile rank(PR), 즉 백분위 점수는 고정 임계값을 충족하는 값의 백분율입니다. 예를 들어 PR(:300)은 값이 300 이하인 데이터 요소의 백분율을 반환합니다. PR(100:2000)은 값이 100에서 2000 사이인 데이터 요소의 백분율을 반환합니다.

  백분위 순위는 하한에서 제외되고 상한에서는 포함됩니다.
- Trimmed count(TC), 즉 절사 수는 절사 평균 통계에 대해 선택한 범위에 있는 데이터 요소의 수입니다. 예를 들어 tc90은 가장 높은 10%의 값에 속하는 데이터 요소를 포함하지 않는 데이터 요소 수를 반환합니다. TC(0.005:0.030)는 값이 0.005(제외)에서 0.030(포함) 사이인 데이터 요소 수를 반환합니다.
- Trimmed sum(TS), 즉 절사 합계는 절사 평균 통계에 대해 선택한 범위에 있는 데이터 요소 값의 합계입니다. 이 값은 (Trimmed Mean) * (Trimmed count)와 같습니다. 예를 들어 ts90은 가장 높은 10%

의 값에 속하는 데이터 요소를 포함하지 않는 데이터 요소의 합계를 반환합니다. TS(80%:)는 가장 낮은 80%의 값 범위에 값이 있는 데이터 요소를 포함하지 않는 데이터 요소 값의 합계를 반환합니다.

> **Note**
>
> Trimmed Mean, Trimmed Count, Trimmed Sum, Winsorized Mean의 경우 두 경계를 백분율 대신 고정 값으로 정의하면 계산에 상위 경계와 동일한 값이 포함되지만 하위 경계와 동일한 값은 포함되지 않습니다.

구문

Trimmed Mean, Trimmed Count, Trimmed Sum, Winsorized Mean의 경우 다음 구문 규칙이 적용됩니다.

- 백분율 기호가 있는 숫자 하나 또는 두 개와 함께 괄호를 사용함으로써 지정된 두 백분위수 사이에 속하는 데이터 집합의 값으로 사용할 경계를 정의합니다. 예를 들어 TM(10%:90%)은 10번째와 90번째 백분위수 사이의 값만 사용합니다. TM(:95%)은 데이터 집합의 가장 낮은 끝부터 95번째 백분위수까지의 값을 사용하며 가장 높은 값을 가진 데이터 요소의 5%를 무시합니다.
- 백분율 기호가 없는 숫자 하나 또는 두 개와 함께 괄호를 사용함으로써 지정된 명시적 값 사이에 속하는 데이터 집합의 값으로 사용할 경계를 정의합니다. 예를 들어 TC(80:500)는 80(제외)에서 500(포함) 사이에 있는 값만 사용합니다. TC(:0.5)는 0.5 이하의 값만 사용합니다.
- 괄호 없이 하나의 숫자를 사용하면 지정된 백분위수보다 높은 데이터 요소를 무시하고 백분율을 사용하여 계산합니다. 예를 들어 tm99는 가장 높은 값을 가진 데이터 요소의 1%를 무시하면서 평균을 계산합니다. 이 값은 TM(:99%)과 같습니다.
- Trimmed mean, Trimmed Count, Trimmed Sum, Winsorized Mean은 모두 TM(5%:95%), TM(100:200) 또는 TM(:95%)과 같이 범위를 지정할 때 대문자를 사용하여 축약할 수 있습니다. tm99와 같이 숫자를 하나만 지정할 때는 소문자만 사용하여 축약할 수 있습니다.

통계 사용 사례

- Trimmed mean은 웹 페이지 대기 시간과 같이 샘플 크기가 큰 지표에 가장 유용합니다. 예를 들어 tm99는 네트워크 문제 또는 인적 오류로 인해 발생할 수 있는 지나치게 높은 이상치를 무시함으로써 일반적인 요청의 평균 대기 시간에 대해 더욱 정확한 수치를 제공합니다. 마찬가지로 TM(10%:)은 캐시 적중으로 인해 발생하는 것과 같이 가장 낮은 10%의 대기 시간 값을 무시합니다. 그리고

TM(10%:99%)은 이러한 유형의 이상치를 둘 다 제외합니다. 지연 시간 모니터링에는 절사 평균을 사용하는 것이 좋습니다.

- Trimmed mean 계산에 사용되는 값의 수가 통계적으로 유의할 만큼 충분한지 확인하기 위해 Trimmed mean을 사용할 때마다 Trimmed Count를 계속 주시하는 것이 좋습니다.
- Percentile rank를 사용하면 값을 범위 '빈(bin)'에 넣을 수 있으며 이를 사용하여 히스토그램을 수동으로 생성할 수 있습니다. 이렇게 하려면 값을 PR(:1), PR(1:5), PR(5:10), PR(10:)과 같은 다양한 빈으로 구분합니다. 이러한 각 빈을 막대 차트로 시각화에 넣으면 히스토그램이 생깁니다.

  백분위 순위는 하한에서 제외되고 상한에서는 포함됩니다.

백분위수와 절사 평균 비교

p99 같은 백분위수와 tm99 같은 절사 평균은 유사하지만 동일하지는 않은 값입니다. p99와 tm99는 모두 이상치로 간주되는 가장 높은 값을 가진 데이터 요소의 1%를 무시합니다. 그다음에 p99는 나머지 99%의 최댓값이고 tm99는 나머지 99%의 평균입니다. 웹 요청 대기 시간을 검토하고 있다면 p99는 이상치를 무시하고 최악의 고객 경험을 알려 주며, tm99는 이상치를 무시하고 평균적인 고객 경험을 알려 줍니다.

절사 평균은 고객 경험을 최적화하려는 경우 살펴보기에 좋은 대기 시간 통계입니다.

백분위수, 절사 평균 및 기타 몇 가지 통계를 사용하기 위한 요구 사항

CloudWatch는 다음 통계를 계산하는데 원시 데이터 요소가 필요합니다.

- 백분위수
- Trimmed mean
- Interquartile mean
- Winsorized mean
- Trimmed sum
- Trimmed count
- Percentile rank

원시 데이터 대신 통계 집합을 사용하여 사용자 지정 통계의 데이터를 게시하는 경우 다음 조건 중 하나가 참일 때에만 이 데이터에 대해 이러한 유형의 통계를 검색할 수 있습니다.

- 통계 세트의 SampleCount 값은 1이고 Min, Max 및 Sum은 모두 같습니다.
- Min과 Max는 같고 Sum은 Min에 SampleCount를 곱한 값과 같습니다.

다음 AWS 서비스에는 이러한 유형의 통계를 지원하는 지표가 포함되어 있습니다.

- API Gateway
- Application Load Balancer
- Amazon EC2
- Elastic Load Balancing
- Kinesis
- Amazon RDS

또한 이러한 유형의 통계는 지표 값 중 어느 하나라도 음수인 경우 지표에 사용할 수 없습니다.

다음 예에서는 EC2 인스턴스와 같은 리소스에 대한 CloudWatch 지표의 통계를 얻는 방법을 보여 줍니다.

예제

- 특정 리소스에 대한 통계 얻기
- 여러 리소스에서 통계 집계
- Auto Scaling 그룹별 통계 집계
- Amazon Machine Image(AMI)별 통계 집계

특정 리소스에 대한 통계 얻기

다음 예제는 특정 EC2 인스턴스의 최대 CPU 사용률을 확인하는 방법을 보여 줍니다.

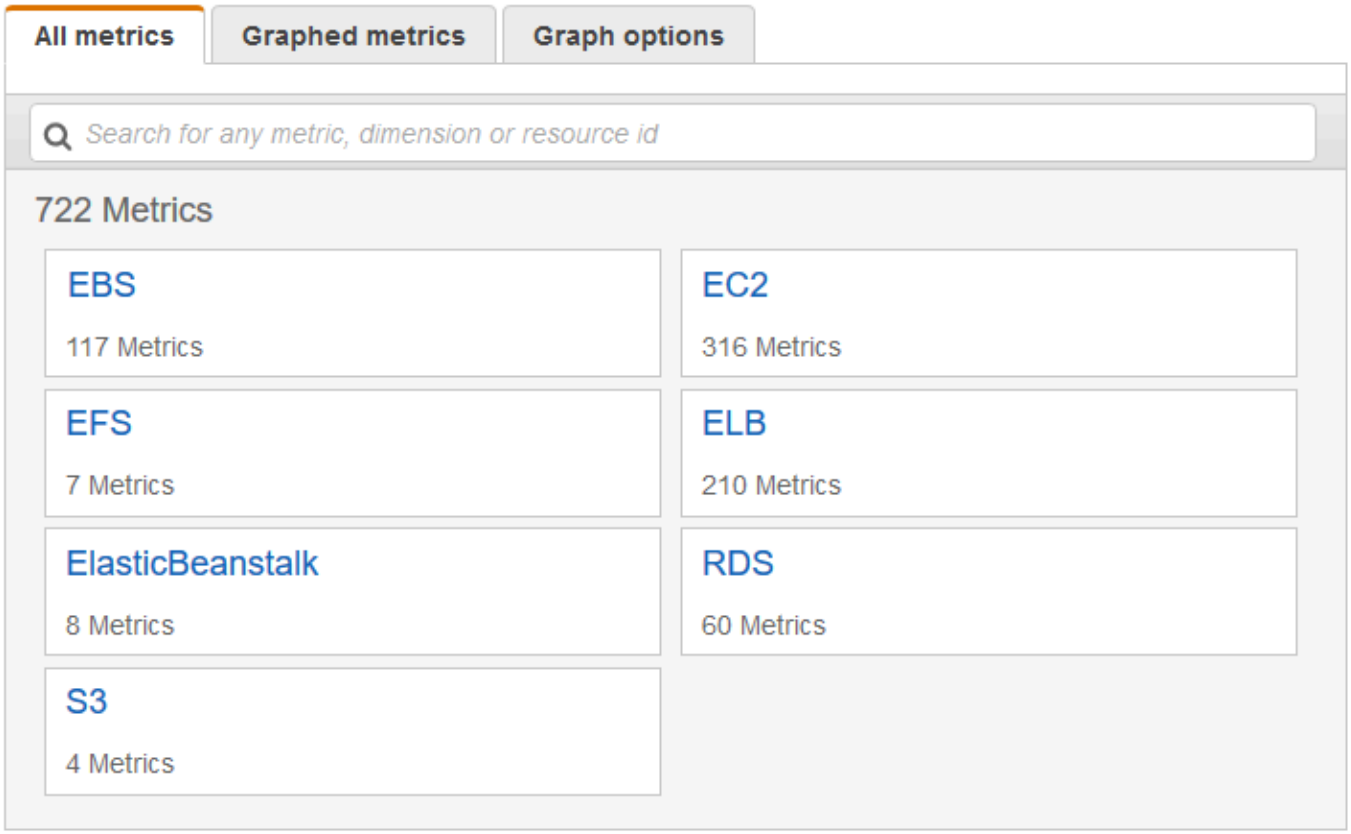
요구 사항

- 인스턴스의 ID가 필요합니다. 인스턴스 ID는 Amazon EC2 콘솔 또는 describe-instances 명령을 사용하여 확인할 수 있습니다.
- 기본적으로 기본 모니터링이 사용되지만 세부 모니터링을 사용하도록 설정할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에 대한 세부 모니터링 사용 또는 사용 중지 단원을 참조하세요.

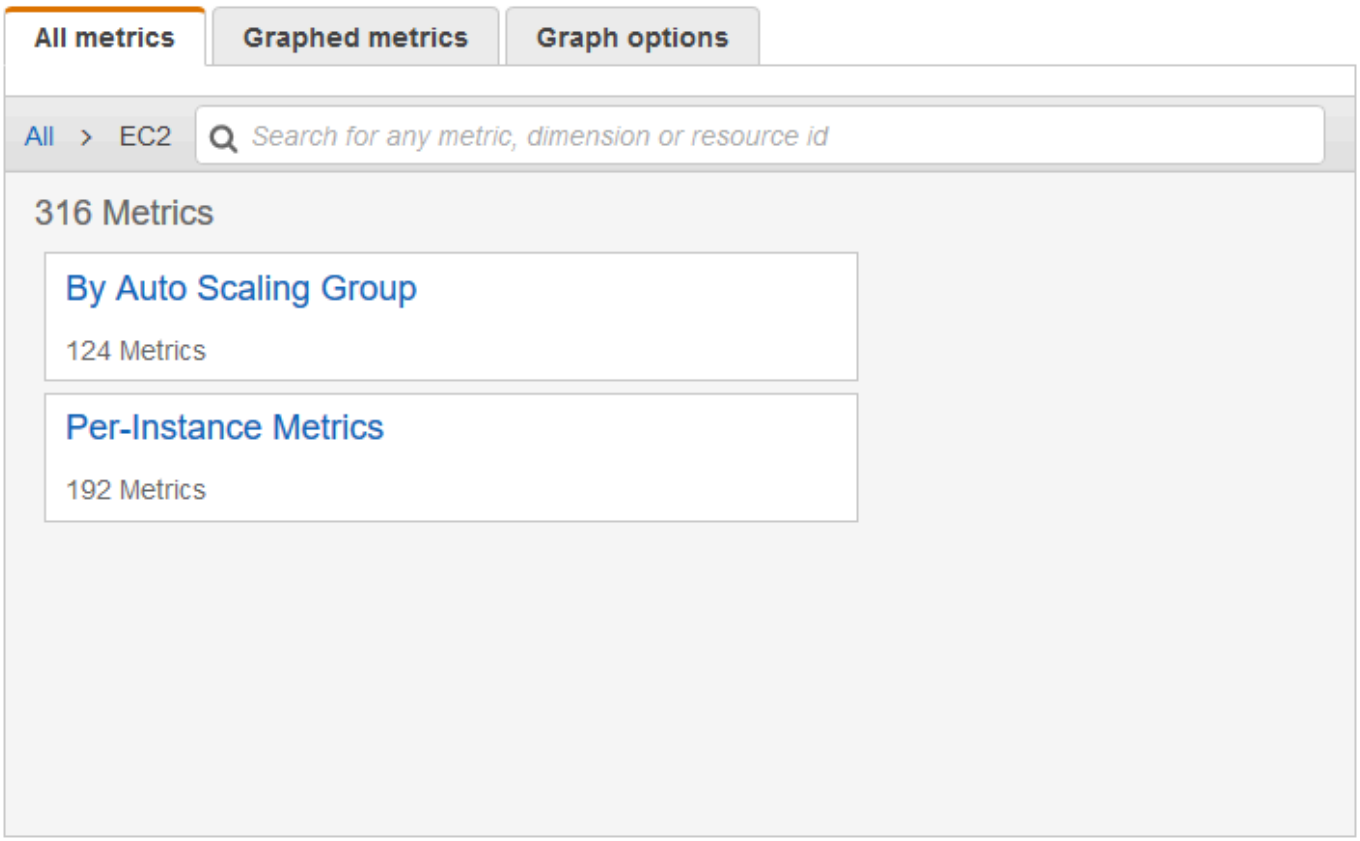
콘솔을 사용하여 특정 인스턴스에 대한 평균 CPU 사용률을 표시하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

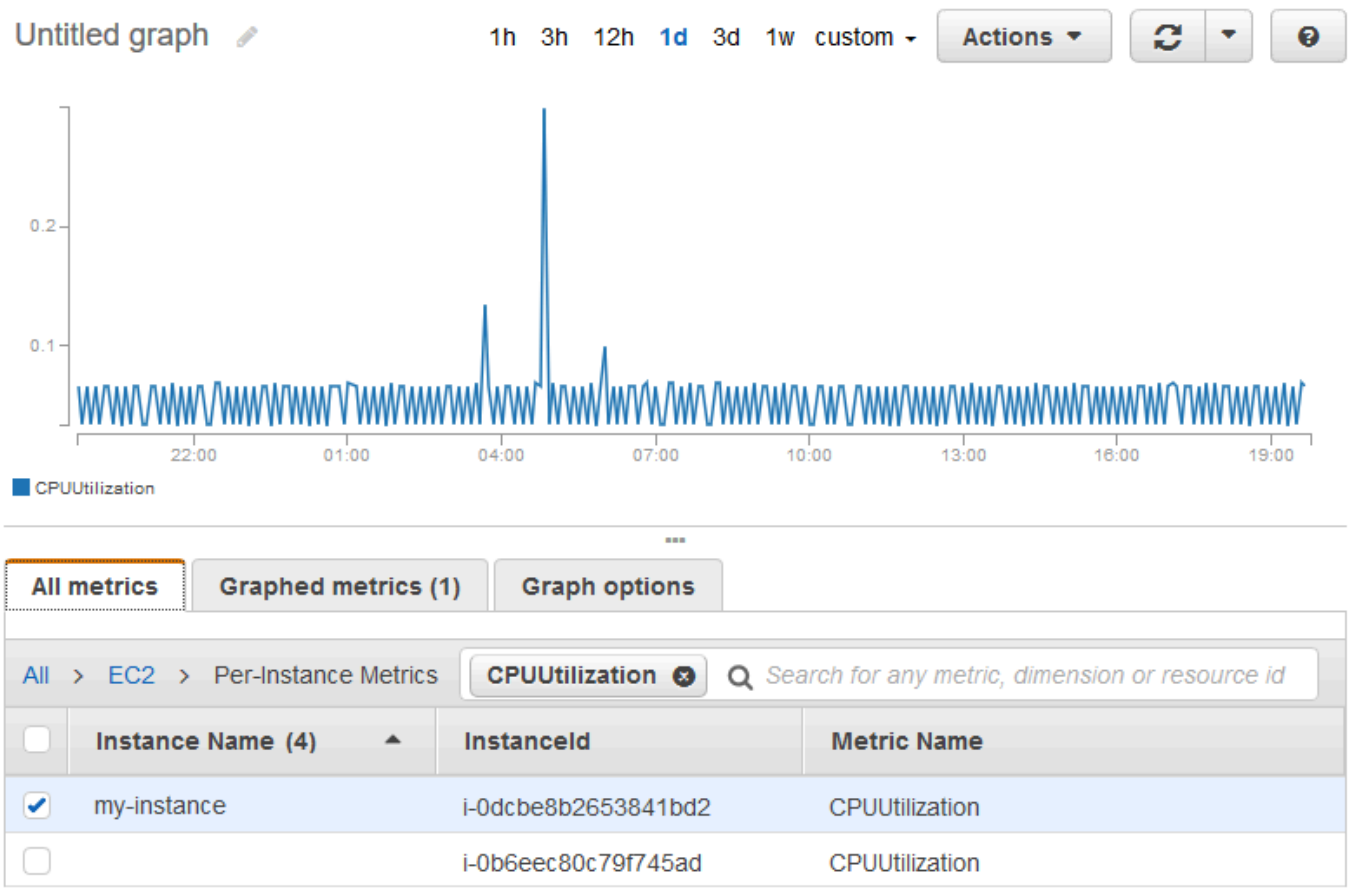
2. 탐색 창에서 지표를 선택합니다.
3. EC2 지표 네임스페이스를 선택합니다.



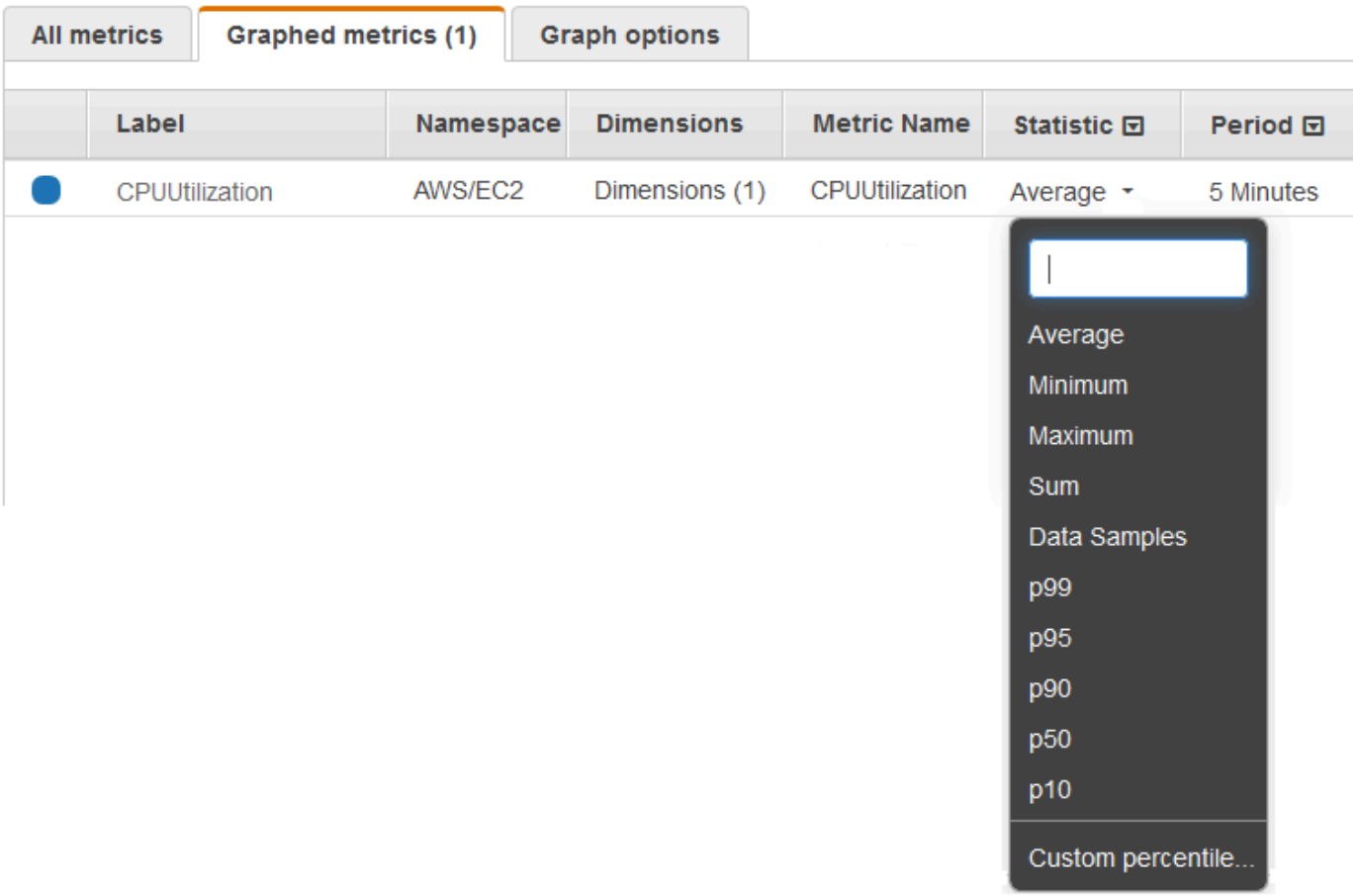
4. 인스턴스별 지표 측정기준을 선택합니다.



5. 검색 필드에 **`CPUUtilization`**을 입력하고 Enter를 누릅니다. 특정 인스턴스의 행을 선택합니다. 그러면 해당 인스턴스의 `CPUUtilization` 지표 그래프가 표시됩니다. 그래프 이름을 변경하려면 연필 아이콘을 선택합니다. 시간 범위를 변경하려면 제공되는 값 중 하나를 선택하거나 맞춤을 선택합니다.



6. 통계를 변경하려면 그래프로 표시된 지표(Graphed metrics) 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 통계 또는 사전 정의된 백분위수 중 하나를 선택하거나 사용자 지정 백분위수(예: **p99.999**)를 지정합니다.



7. 기간을 변경하려면 그래프로 표시된 지표(Graphed metrics) 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 다른 값을 선택합니다.

AWS CLI를 사용하여 EC2 인스턴스별 CPU 사용률을 얻으려면

다음과 같이 get-metric-statistics 명령을 사용하여 지정한 인스턴스의 `CPUUtilization` 지표를 확인합니다.

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=InstanceId,Value=i-1234567890abcdef0 --statistics Maximum \
--start-time 2016-10-18T23:18:00 --end-time 2016-10-19T23:18:00 --period 360
```

24시간이라는 요청된 시간 간격에서 6분마다 통계 값이 반환됩니다. 각 값은 6분이라는 특정 기간 동안 지정된 인스턴스의 최대 CPU 사용률을 나타냅니다. 데이터 요소는 시간 순서대로 반환되지 않습니다. 다음은 예제 출력의 시작 부분을 보여줍니다(전체 출력에는 24시간 동안 6분마다 반환된 데이터 요소가 포함됨).

```
{
```

```
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:18:00Z",
            "Maximum": 0.33000000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-19T03:18:00Z",
            "Maximum": 99.670000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-19T07:18:00Z",
            "Maximum": 0.34000000000000002,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

여러 리소스에서 통계 집계

여러 리소스에서 AWS 리소스에 대한 지표를 집계할 수 있습니다. 지표는 리전 간에 완전히 별개이지만, 지표 수학을 사용하여 리전 전반에서 유사한 지표를 집계할 수 있습니다. 자세한 내용은 지표 수학 사용 단원을 참조하십시오.

예를 들어 세부 모니터링이 활성화된 EC2 인스턴스에 대한 통계를 집계할 수 있습니다. 기본 모니터링을 사용하는 인스턴스는 포함되지 않습니다. 따라서 1분 기간으로 데이터를 제공하는 세부 모니터링(추가 요금 부과)을 활성화해야 합니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에 대한 세부 모니터링 사용 또는 사용 중지 단원을 참조하세요.

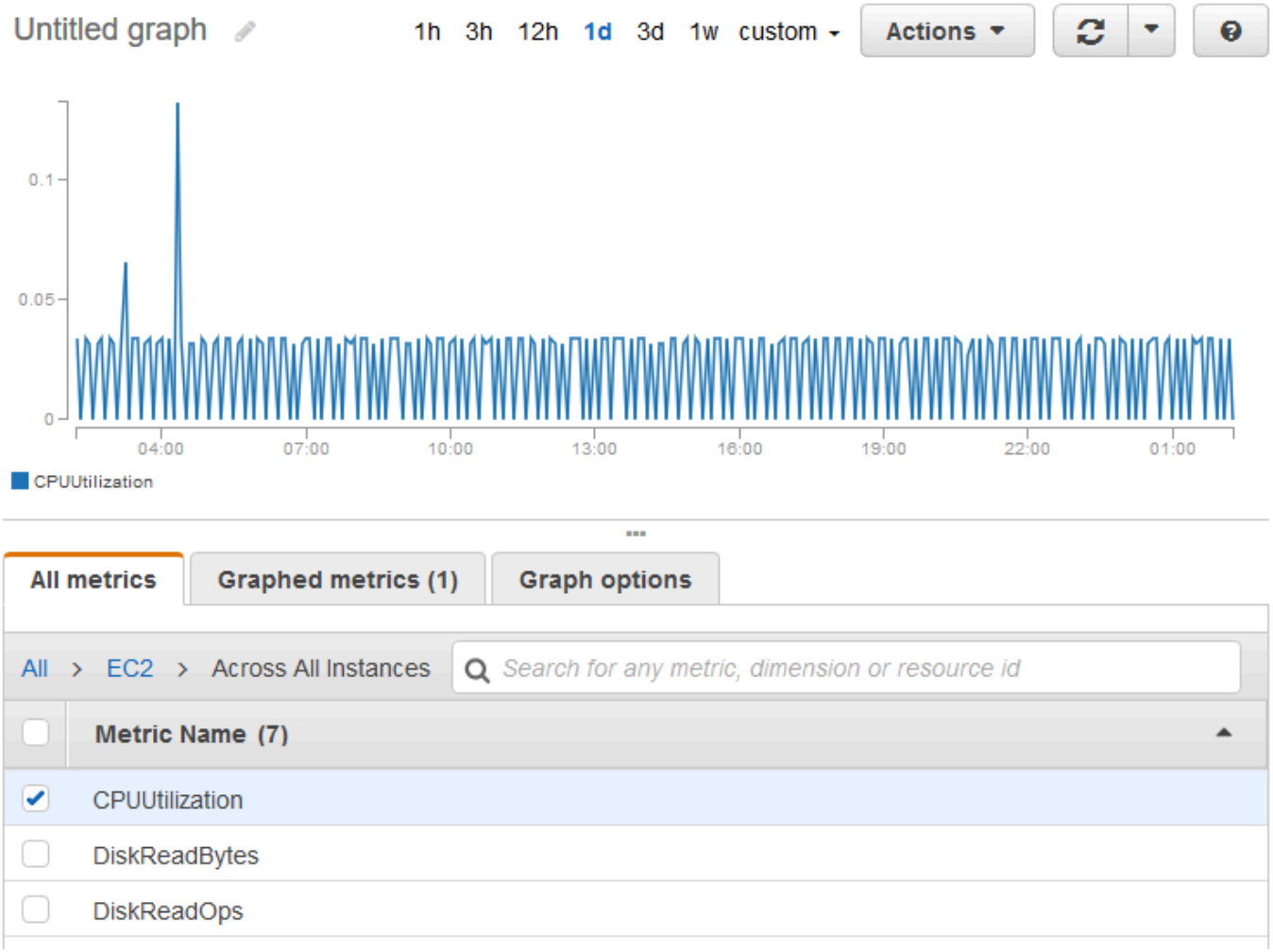
이 예제는 EC2 인스턴스의 평균 CPU 사용률을 확인하는 방법을 보여줍니다. 지정된 측정기준이 없으므로 CloudWatch에서는 AWS/EC2 네임스페이스의 모든 측정기준에 대한 통계를 반환합니다. 다른 지표에 대한 통계를 얻으려면 CloudWatch 지표를 게시하는 AWS 서비스 단원을 참조하세요.

> **Important**
>
> AWS 네임스페이스에서 모든 측정기준을 검색하는 이 기법은 CloudWatch에 게시하는 사용자 지정 네임스페이스에는 적용되지 않습니다. 사용자 지정 네임스페이스를 사용하는 경우 데이

터 요소가 포함된 통계를 검색하려면 특정 데이터 요소와 연결된 전체 측정기준 세트를 지정해야 합니다.

EC2 인스턴스에 대한 평균 CPU 사용률을 표시하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. EC2 네임스페이스를 선택한 후 전체 인스턴스를 선택합니다.
4. `CPUUtilization`을 포함하는 행을 선택합니다. 그러면 모든 EC2 인스턴스에 대한 지표 그래프가 표시됩니다. 그래프 이름을 변경하려면 연필 아이콘을 선택합니다. 시간 범위를 변경하려면 제공되는 값 중 하나를 선택하거나 맞춤을 선택합니다.



5. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 통계 또는 사전 정의된 백분위수 중 하나를 선택하거나 사용자 지정 백분위수(예: **`p95.45`**)를 지정합니다.

6. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 다른 값을 선택합니다.

AWS CLI를 사용하여 EC2 인스턴스 간 평균 CPU 사용률을 얻으려면

다음과 같이 get-metric-statistics 명령을 사용합니다.

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization
 --statistics "Average" "SampleCount" \
--start-time 2016-10-11T23:18:00 --end-time 2016-10-12T23:18:00 --period 3600
```

다음은 예 출력입니다.

```
{
    "Datapoints": [
        {
            "SampleCount": 238.0,
            "Timestamp": "2016-10-12T07:18:00Z",
            "Average": 0.038235294117647062,
            "Unit": "Percent"
        },
        {
            "SampleCount": 240.0,
            "Timestamp": "2016-10-12T09:18:00Z",
            "Average": 0.16670833333333332,
            "Unit": "Percent"
        },
        {
            "SampleCount": 238.0,
            "Timestamp": "2016-10-11T23:18:00Z",
            "Average": 0.041596638655462197,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

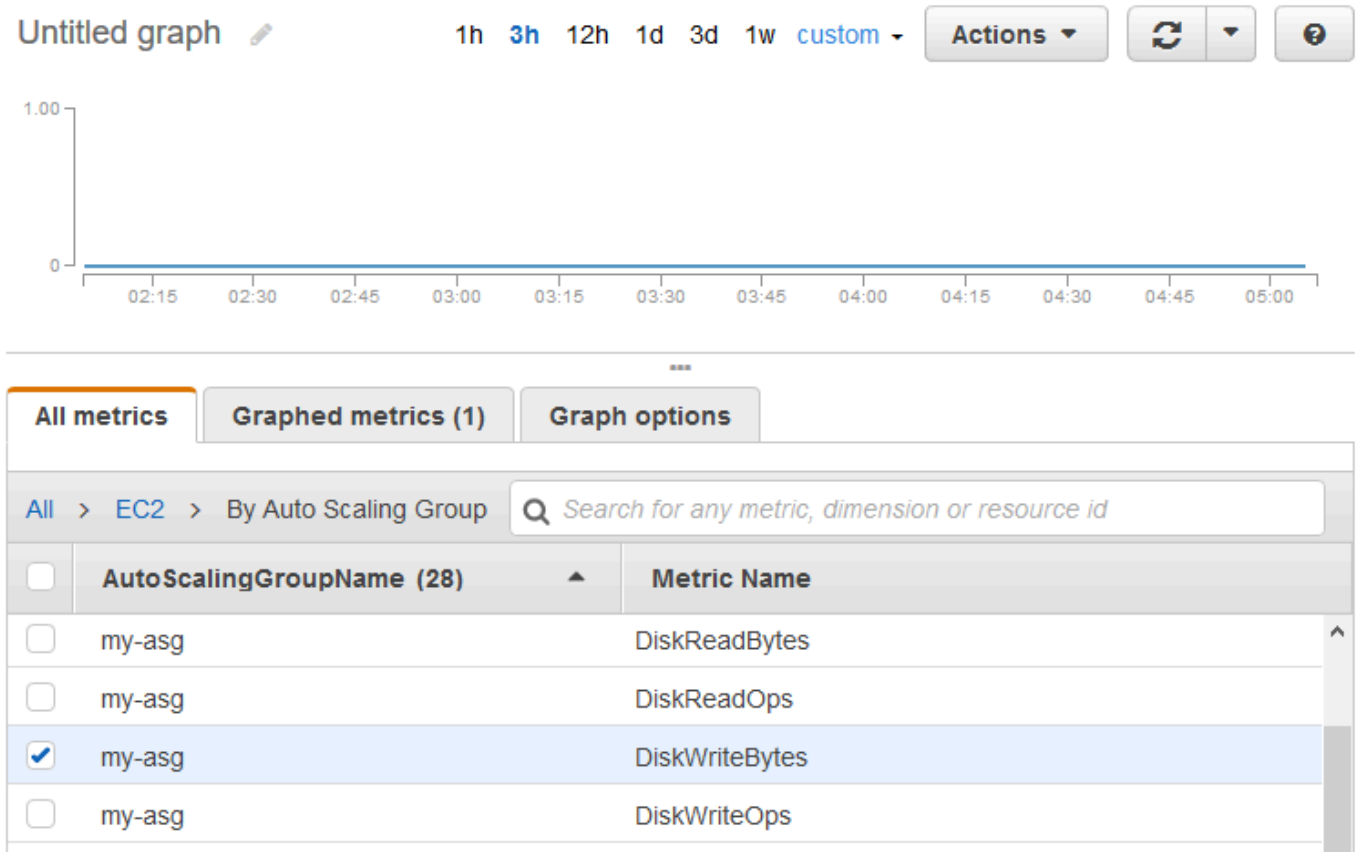
Auto Scaling 그룹별 통계 집계

EC2 인스턴스에 대한 통계를 하나의 Auto Scaling 그룹에 집계할 수 있습니다. 지표는 리전 간에 완전히 별개이지만, CloudWatch 지표 수학을 사용하여 여러 리전에서 지표를 집계하고 변환할 수 있습니다. 교차 계정 대시보드를 사용하여 다른 계정의 지표에 대한 지표 수학을 수행할 수도 있습니다.

이 예에서는 한 Auto Scaling 그룹의 디스크에 쓴 총 바이트 수를 가져오는 방법을 보여 줍니다. 총수는 지정된 Auto Scaling 그룹의 모든 EC2 인스턴스에서 24시간 간격으로 1분의 기간에 대해 계산됩니다.

콘솔을 사용하여 Auto Scaling 그룹의 인스턴스에 대한 DiskWriteBytes를 표시하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [지표(Metrics)]를 선택합니다.
3. EC2 네임스페이스를 선택한 후 Auto Scaling 그룹별을 선택합니다.
4. DiskWriteBytes 지표 및 특정 Auto Scaling 그룹에 대한 행을 선택합니다. 그러면 오토 스케일링 그룹의 인스턴스에 대한 지표 그래프가 표시됩니다. 그래프 이름을 변경하려면 연필 아이콘을 선택합니다. 시간 범위를 변경하려면 제공되는 값 중 하나를 선택하거나 맞춤을 선택합니다.



5. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 통계 또는 사전 정의된 백분위수 중 하나를 선택하거나 사용자 지정 백분위수(예: **p95.45**)를 지정합니다.
6. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 다른 값을 선택합니다.

AWS CLI를 사용하여 Auto Scaling 그룹의 인스턴스에 대한 DiskWriteBytes를 가져오려면

다음과 같이 get-metric-statistics 명령을 사용합니다.

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name DiskWriteBytes
--dimensions Name=AutoScalingGroupName,Value=my-asg --statistics "Sum" "SampleCount" \
--start-time 2016-10-16T23:18:00 --end-time 2016-10-18T23:18:00 --period 360
```

출력의 예제는 다음과 같습니다.

```
{
    "Datapoints": [
        {
            "SampleCount": 18.0,
            "Timestamp": "2016-10-19T21:36:00Z",
            "Sum": 0.0,
            "Unit": "Bytes"
        },
        {
            "SampleCount": 5.0,
            "Timestamp": "2016-10-19T21:42:00Z",
            "Sum": 0.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "DiskWriteBytes"
}
```

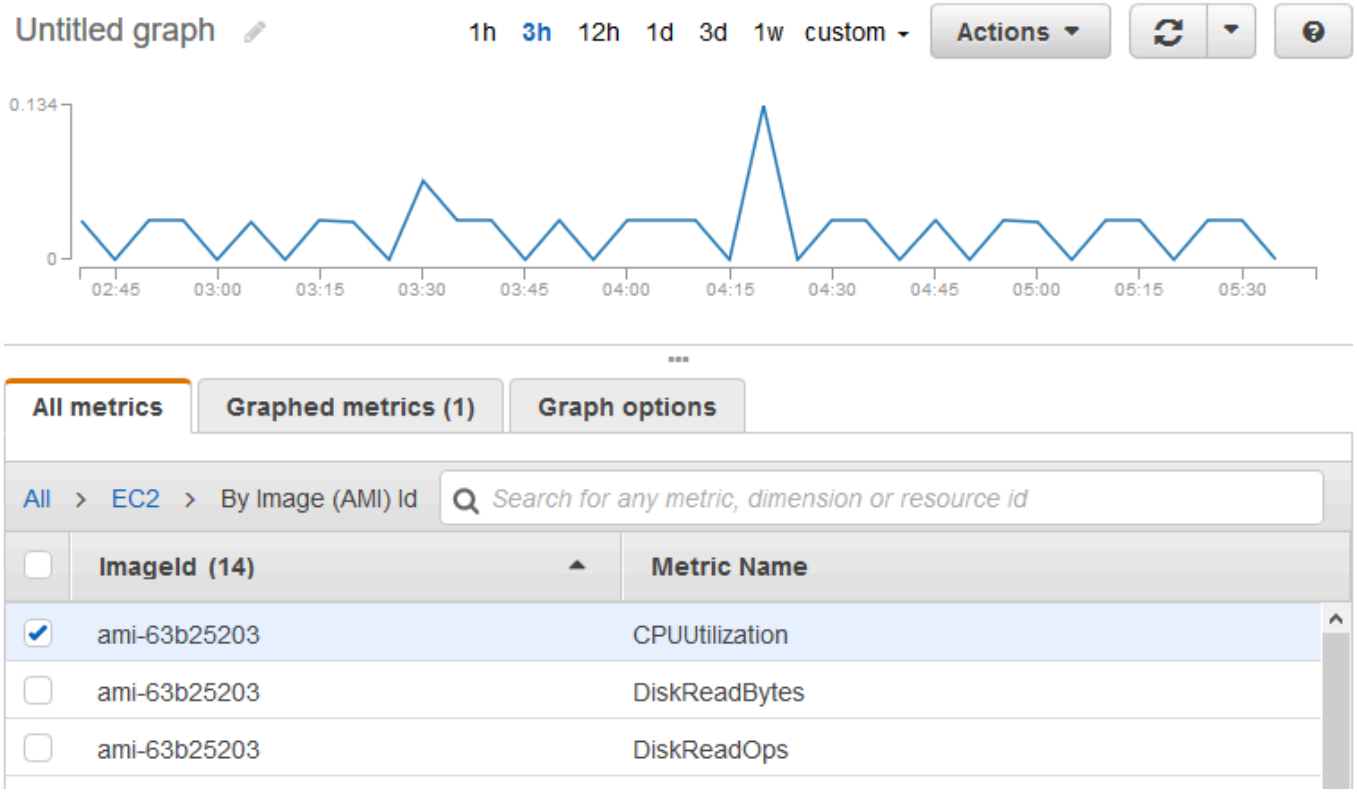
Amazon Machine Image(AMI)별 통계 집계

세부 모니터링이 활성화된 EC2 인스턴스에 대해 통계를 집계할 수 있습니다. 기본 모니터링을 사용하는 인스턴스는 포함되지 않습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에 대한 세부 모니터링 사용 또는 사용 중지 단원을 참조하세요.

이 예제는 지정된 AMI를 사용하는 모든 인스턴스의 평균 CPU 사용률을 확인하는 방법을 보여줍니다. 평균은 1일 기간의 60초 시간 간격에 대한 평균입니다.

콘솔을 사용하여 AMI의 평균 CPU 사용률을 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [지표(Metrics)]를 선택합니다.
3. EC2 네임스페이스를 선택한 후 이미지(AMI) ID별을 선택합니다.
4. `CPUUtilization` 지표 행과 특정 AMI를 선택합니다. 그러면 지정한 AMI의 지표 그래프가 표시됩니다. 그래프 이름을 변경하려면 연필 아이콘을 선택합니다. 시간 범위를 변경하려면 제공되는 값 중 하나를 선택하거나 맞춤을 선택합니다.



5. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 통계 또는 사전 정의된 백분위수 중 하나를 선택하거나 사용자 지정 백분위수(예: **`p95.45`**)를 지정합니다.
6. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 다른 값을 선택합니다.

AWS CLI를 사용하여 AMI당 평균 CPU 사용률을 얻으려면

다음과 같이 get-metric-statistics 명령을 사용합니다.

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=ImageId,Value=ami-3c47a355 --statistics Average \
--start-time 2016-10-10T00:00:00 --end-time 2016-10-11T00:00:00 --period 3600
```

이 작업은 1일 간격으로 1시간 값인 통계를 반환합니다. 각 값은 지정된 AMI를 실행 중인 EC2 인스턴스의 평균 CPU 사용률을 나타냅니다. 출력의 예제는 다음과 같습니다.

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-10T07:00:00Z",
            "Average": 0.041000000000000009,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-10T14:00:00Z",
            "Average": 0.079579831932773085,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-10T06:00:00Z",
            "Average": 0.036000000000000011,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

사용자 지정 지표 게시

AWS CLI 또는 API를 사용하여 CloudWatch에 자체 지표를 게시할 수 있습니다. AWS Management Console을 사용하여 게시된 지표의 통계 그래프를 볼 수 있습니다.

CloudWatch는 지표에 관한 데이터를 일련의 데이터 요소로 저장합니다. 각 데이터 요소에는 연결된 시간스탬프가 있습니다. 통계 세트라는 집계된 데이터 요소 세트를 게시할 수도 있습니다.

주제

- 고분해능 지표

- 측정기준 사용
- 단일 데이터 포인트 게시
- 통계 세트 게시
- 값 0 게시
- 지표 게시 중지

고분해능 지표

각 지표는 다음 중 하나입니다.

- 표준 분해능 - 1분 세분화 데이터
- 고분해능 - 1초 세분화 데이터

AWS 서비스에 의해 생성되는 지표는 기본적으로 표준 분해능입니다. 사용자 지정 지표를 게시할 때는 지표를 표준 분해능 또는 고분해능으로 정의할 수 있습니다. 고분해능 지표를 게시할 경우 CloudWatch는 이 지표를 1초 분해능으로 저장합니다. 그러면 사용자는 1초, 5초, 10초, 30초 또는 60초의 배수 기간으로 지표를 읽고 검색할 수 있습니다.

고분해능 지표는 애플리케이션의 단기(1분 미만) 활동을 보다 즉각적으로 관찰할 수 있게 해줍니다. 사용자 지정 지표에 대해 `PutMetricData`를 호출할 때마다 요금이 부과되며, 따라서 고분해능 지표에 대해 `PutMetricData`를 자주 호출할수록 요금이 증가할 수 있다는 점에 유의하세요. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

고분해능 지표에 대해 경보를 설정할 경우 고분해능 경보를 10초 또는 30초 기간으로 지정하거나 60초의 배수 기간으로 정기 경보를 설정할 수 있습니다. 10초 또는 30초 기간의 고분해능 경보는 요금이 더 비쌉니다.

측정기준 사용

사용자 지정 지표에서 `--dimensions` 파라미터는 공통입니다. 측정기준은 지표가 무엇이고 어떤 데이터가 저장되었는지 훨씬 명확하게 보여줍니다. 지표 하나에 측정기준을 최대 30개까지 가질 수 있으며, 각 측정기준은 이름-값 페어로 정의됩니다.

서로 다른 명령을 사용하면 측정기준을 지정하는 방법도 달라집니다. put-metric-data에서는 각 측정기준을 *`MyName`*=*`MyValue`*로 지정하고, get-metric-statistics 또는 put-metric-alarm에서는 `Name=`*`MyName`*, `Value=`*`MyValue`* 형식을 사용합니다. 예를 들어, 다음 명령은 이름이 `Buffers` 및 `InstanceId`인 두 측정기준으로 `InstanceType` 지표를 게시합니다.

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace MyNameSpace --unit
 Bytes --value 231434333 --dimensions InstanceId=1-23456789,InstanceType=m1.small
```

이 명령은 동일한 지표에 대한 통계를 검색합니다. 단일 측정기준의 Name 및 Value 부분은 쉼표로 구분하지만, 측정기준이 여러 개인 경우에는 하나의 측정기준과 그 다음 측정기준 사이에 공백을 둡니다.

```
aws cloudwatch get-metric-statistics --metric-name Buffers --namespace MyNameSpace --
dimensions Name=InstanceId,Value=1-23456789 Name=InstanceType,Value=m1.small --start-
time 2016-10-15T04:00:00Z --end-time 2016-10-19T07:00:00Z --statistics Average --period
 60
```

단일 지표에 여러 개의 측정기준이 포함된 경우에는 get-metric-statistics를 사용할 때 정의된 모든 측정기준에 대해 값을 지정해야 합니다. 예를 들어 Amazon S3 지표 BucketSizeBytes에는 BucketName 및 StorageType 측정기준이 포함되어 있으므로 get-metric-statistics를 사용하여 두 측정기준을 모두 지정해야 합니다.

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --start-time
 2017-01-23T14:23:00Z --end-time 2017-01-26T19:30:00Z --period 3600 --namespace
 AWS/S3 --statistics Maximum --dimensions Name=BucketName,Value=MyBucketName
 Name=StorageType,Value=StandardStorage --output table
```

지표에 정의되는 측정기준을 확인하려면 list-metrics 명령을 사용합니다.

단일 데이터 포인트 게시

신규 또는 기존 지표에서 단일 데이터 요소를 게시하려면 하나의 값과 타임스탬프와 함께 put-metric-data 명령을 사용하세요. 예를 들어, 다음 각 작업은 데이터 요소 하나를 게시합니다.

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 2 --timestamp 2016-10-20T12:00:00.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 4 --timestamp 2016-10-20T12:00:01.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 5 --timestamp 2016-10-20T12:00:02.000Z
```

새 지표 이름으로 이 명령을 호출하면 CloudWatch가 지표를 생성합니다. 그러지 않으면 CloudWatch는 지정된 기존의 지표에 데이터를 연결합니다.

> **Note**
>
> 지표를 작성할 때 get-metric-statistics 명령을 사용하여 새 지표에 대한 통계를 검색할 수 있기까지 최대 2분이 소요될 수 있습니다. 그러나 list-metrics 명령을 사용하여 검색된 지표 목록에 새 지표가 나타나려면 최대 15분이 걸릴 수 있습니다.

타임스탬프가 1000분의 1초만큼 세분화된 데이터 요소를 게시할 수 있지만, CloudWatch는 1초의 최소 단위로 데이터를 집계합니다. CloudWatch는 각 기간 동안 수신된 값의 평균(모든 항목의 합계를 항목 수로 나눈 값)은 물론 동일한 기간의 샘플 수, 최댓값, 최솟값을 기록합니다. 예를 들어 이전 예의 `PageViewCount` 지표에는 타임스탬프가 몇 초 간격인 데이터 요소 3개가 들어 있습니다. 기간을 1분으로 설정한 경우 이러한 데이터 요소 모두에 1분 기간 내의 타임스탬프가 있기 때문에 CloudWatch는 3개의 데이터 요소를 집계합니다.

get-metric-statistics 명령을 사용하여 게시된 데이터 요소에 따라 통계를 검색할 수 있습니다.

```
aws cloudwatch get-metric-statistics --namespace MyService --metric-name PageViewCount
 \
--statistics "Sum" "Maximum" "Minimum" "Average" "SampleCount" \
--start-time 2016-10-20T12:00:00.000Z --end-time 2016-10-20T12:05:00.000Z --period 60
```

출력의 예제는 다음과 같습니다.

```
{
    "Datapoints": [
        {
            "SampleCount": 3.0,
            "Timestamp": "2016-10-20T12:00:00Z",
            "Average": 3.6666666666666665,
            "Maximum": 5.0,
            "Minimum": 2.0,
            "Sum": 11.0,
            "Unit": "None"
        }
    ],
    "Label": "PageViewCount"
}
```

통계 세트 게시

CloudWatch에 게시하기 전에 데이터를 집계할 수 있습니다. 분당 여러 데이터 요소가 있는 경우 데이터를 집계하면 put-metric-data에 대한 호출 수가 최소화됩니다. 예를 들어, 서로 3초 내에 있는 데이터 요소 3개에 대해 put-metric-data를 여러 번 호출하는 대신 `--statistic-values` 파라미터를 사용하여 호출 한 번으로 게시한 통계 세트로 데이터를 집계할 수 있습니다.

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService
 --statistic-values Sum=11,Minimum=2,Maximum=5,SampleCount=3 --
timestamp 2016-10-14T12:00:00.000Z
```

CloudWatch가 백분위수를 계산하려면 원시 데이터 요소가 필요합니다. 대신 통계 세트를 사용해 데이터를 게시하면 아래 조건 중 하나를 충족하기 전까지 이 데이터에 대한 백분위수 통계를 검색할 수 없습니다.

- 통계 세트의 `SampleCount`는 1입니다.
- 통계 세트의 `Minimum` 및 `Maximum`이 동일합니다.

값 0 게시

데이터가 훨씬 산발적이고 연결된 데이터가 없는 기간이 있으면 해당 기간에 대해 0 값(0)을 게시하거나 값을 전혀 게시하지 않도록 선택할 수 있습니다. `PutMetricData`에 대한 정기적인 호출을 통해 애플리케이션의 상태를 모니터링하는 경우 값을 게시하지 않는 대신 0을 게시하려고 할 수 있습니다. 예를 들어 애플리케이션에서 지표를 게시하는 데 실패한 경우 5분마다 이를 알리도록 CloudWatch 경보를 설정할 수 있습니다. 애플리케이션에서 연결된 데이터가 없는 기간에 대해 0을 게시하도록 하려고 합니다.

또한 데이터 요소의 총 수를 추적하거나 최소값 또는 평균과 같은 통계에 값이 0인 데이터 요소를 포함하려는 경우 0을 게시할 수도 있습니다.

지표 게시 중지

CloudWatch에 사용자 지정 지표 게시를 중지하려면 PutMetricData 사용을 중지하도록 애플리케이션 또는 서비스의 코드를 변경합니다. CloudWatch는 애플리케이션에서 지표를 가져오지 않고 푸시된 내용만 수신하므로 지표 게시를 중지하려면 소스에서 지표를 중지해야 합니다.

Amazon CloudWatch 경보 사용

Amazon CloudWatch에서 지표 경보와 복합 경보를 생성할 수 있습니다.

- 지표 경보는 단일 CloudWatch 지표를 감시하거나 CloudWatch 지표를 기반으로 하는 수학 표현식의 결과를 감시합니다. 이러한 경보는 여러 기간에 대해 지정된 임곗값과 지표 또는 표현식의 값 비교하여 하나 이상의 작업을 수행합니다. 작업은 Amazon SNS 주제에 알림을 전송하거나, Amazon EC2 작업 또는 Amazon EC2 Auto Scaling 작업을 수행하거나, Systems Manager에서 OpsItem 또는 인시던트를 생성하는 것일 수 있습니다.
- 복합 경보에는 사용자가 생성한 다른 경보의 경보 상태를 고려하는 규칙 표현식이 포함됩니다. 복합 경보는 규칙의 모든 조건이 충족되는 경우에만 ALARM 상태로 전환됩니다. 복합 경보의 규칙 표현식에 지정된 경보에는 지표 경보 및 기타 복합 경보가 포함될 수 있습니다.

  복합 경보를 사용하면 경보 노이즈를 줄일 수 있습니다. 여러 지표 경보를 생성할 수 있으며, 복합 경보를 생성하고 복합 경보에 대해서만 경보를 설정할 수도 있습니다. 예를 들어 모든 기본 지표 경보가 ALARM 상태인 경우에만 복합 경보가 ALARM 상태로 전환되도록 할 수 있습니다.

  복합 경보는 경보 상태가 변경될 때 Amazon SNS 알림을 전송할 수 있고 경보가 ALARM 상태가 될 때 Systems Manager OpsItem 또는 인시던트를 생성할 수 있지만, EC2 작업 또는 Auto Scaling 작업을 수행할 수는 없습니다.

> **Note**
>
> AWS 계정에서 원하는 만큼 경보를 생성할 수 있습니다.

대시보드에 경보를 추가할 수 있으므로 여러 리전에 걸쳐 AWS 리소스 및 애플리케이션에 대한 경보를 모니터링하고 수신할 수 있습니다. 대시보드에 경보를 추가하면 경보가 `INSUFFICIENT_DATA` 상태인 경우 회색으로, `ALARM` 상태인 경우 빨간색으로 바뀝니다. 경보가 `OK` 상태인 경우 색상이 표시되지 않습니다.

CloudWatch 콘솔 탐색 창의 즐겨찾기 및 최근 항목(Favorites and recents) 옵션에서 최근에 방문한 경보를 즐겨찾기에 추가할 수도 있습니다. 즐겨찾기 및 최근 항목(Favorites and recents) 옵션에는 즐겨찾는 경보 및 최근에 방문한 경보에 대한 열이 있습니다.

경보는 경보 상태가 변경될 때만 작업을 호출합니다. 단, Auto Scaling 작업이 있는 경보는 예외입니다. Auto Scaling 작업의 경우 경보는 분당 한 번씩 계속해서 경보가 새 상태로 유지되는 작업을 호출합니다.

경보는 동일한 계정의 지표를 감시할 수 있습니다. CloudWatch 콘솔에서 교차 계정 기능을 사용 설정한 경우 다른 AWS 계정의 지표를 감시하는 경보를 생성할 수도 있습니다. 교차 계정 복합 경보 생성은 지원되지 않습니다. ANOMALY_DETECTION_BAND, INSIGHT_RULE 및 SERVICE_QUOTA 함수가 교차 계정 경보에 대해 지원되지 않는다는 점을 제외하고 수학 표현식을 사용하는 교차 계정 경보 생성이 지원됩니다.

> **Note**
>
> CloudWatch는 지정된 작업을 테스트하거나 검증하지 않으며 존재하지 않은 작업을 호출하려는 시도로 인해 발생하는 Amazon EC2 Auto Scaling 또는 Amazon SNS 오류를 감지하지도 않습니다. 경보 작업이 존재하는지 확인하십시오.

지표 경보 상태

지표 경보에는 다음과 같은 상태가 있을 수 있습니다.

- OK – 지표 또는 표현식이 정의된 임곗값 내에 있습니다.
- ALARM – 지표 또는 표현식이 정의된 임곗값을 벗어났습니다.
- INSUFFICIENT_DATA – 경보가 방금 시작되었거나 지표를 사용할 수 없거나 지표에서 경보 상태를 결정하는 데 사용할 수 있는 데이터가 충분하지 않습니다.

경보 평가

경보를 생성할 때 다음과 같은 세 가지 설정을 지정하여 CloudWatch가 경보 상태를 변경할 시기를 평가할 수 있도록 합니다.

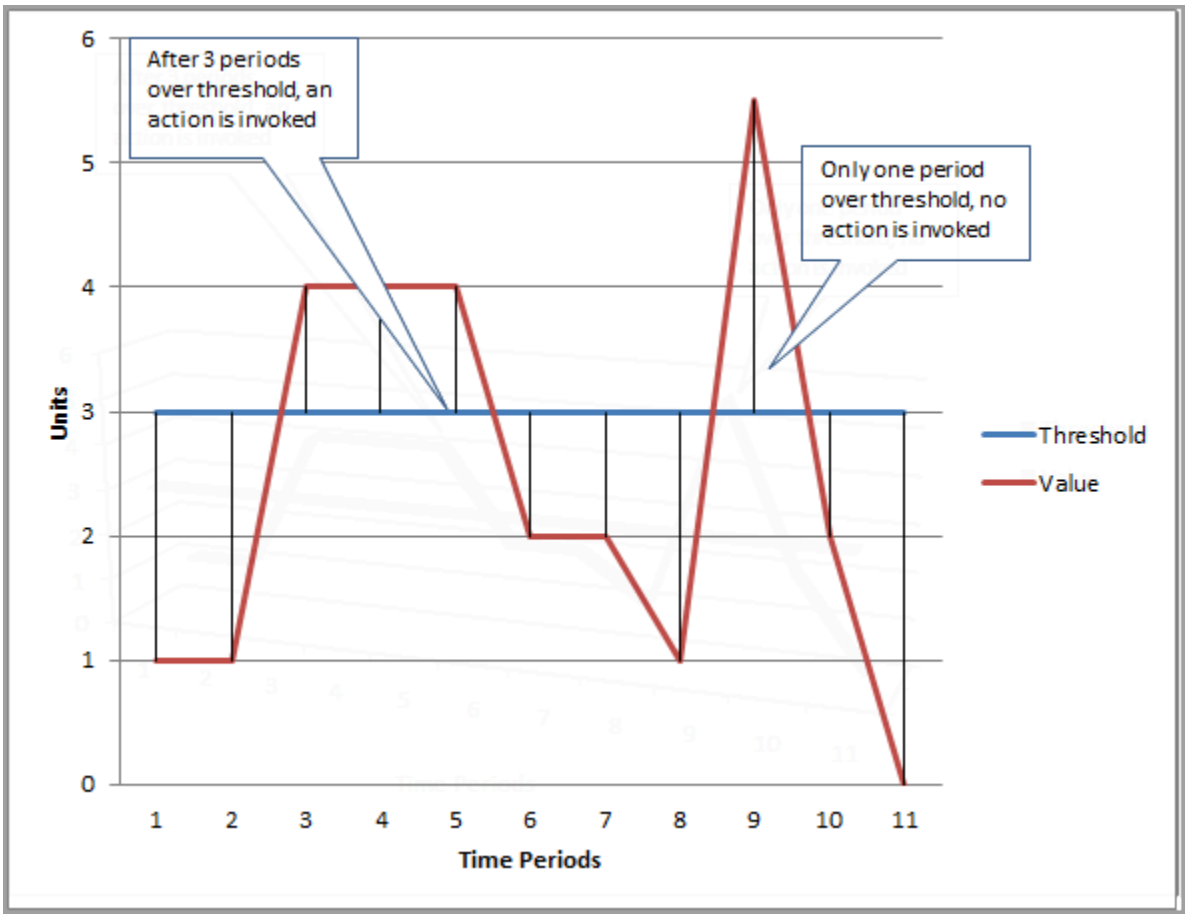
- 기간은 경보에 대해 개별 데이터 포인트를 생성하기 위해 지표 또는 표현식을 평가하는 기간입니다. 초로 표시됩니다.
- [평가 기간(Evaluation Periods)]은 경보 상태를 결정할 때 평가할 가장 최근의 기간 또는 데이터 요소의 수입니다.

- 경보에 대한 데이터 요소(Datapoints to Alarm)는 평가 기간 내에 경보가 ALARM 상태에 도달하게 만드는 위반 데이터 요소의 수입니다. 위반 데이터 포인트가 연속적일 필요는 없지만, 평가 기간(Evaluation Period)과 동일한 마지막 데이터 포인트 수 이내여야 합니다.

1분 이상인 기간의 경우 경보는 1분마다 평가되며 평가는 기간 및 평가 기간에 정의된 기간을 기준으로 합니다. 예를 들어 기간이 5분(300초)이고 평가 기간이 1인 경우 5분이 끝날 때 1분에서 5분까지의 데이터를 기반으로 경보가 평가됩니다. 그런 다음 6분이 끝나면 2분에서 6분까지의 데이터를 기반으로 경보를 평가합니다.

경보 기간이 10초 또는 30초인 경우 경보는 10초마다 평가됩니다.

다음 그림에서 지표 경보에 대한 경보 임곗값은 3개 단위로 설정됩니다. [평가 기간(Evaluation Period)]과 [경보에 대한 데이터 요소(Datapoints to Alarm)]가 둘 다 3입니다. 즉, 가장 최근의 연속된 세 기간에서 기존 데이터 요소가 모두 임곗값을 초과하면 경보가 ALARM 상태가 됩니다. 그림에서는 기간 3에서 6 사이에 이러한 일이 발생합니다. 기간 6에서는 값이 임곗값 아래로 떨어지므로 평가 대상 기간 중 하나가 위반 상태가 아닙니다. 따라서 경보 상태가 다시 OK로 변경됩니다. 9번째 기간에 다시 한 번 임곗값이 위반되지만, 오직 하나의 기간 동안에만 그렇습니다. 결과적으로 경보 상태는 OK로 남아 있습니다.



[평가 기간(Evaluation Period)]과 [경보에 대한 데이터 요소(Datapoints to Alarm)]를 다른 값으로 구성하는 경우 이는 'M out of N(N 중 M)' 경보를 설정한 것입니다. [경보에 대한 데이터 요소(Datapoints to Alarm)]가 ('M')이고 [평가 기간(Evaluation Periods)]은 ('N')입니다. 평가 간격은 평가 기간에 기간 길이를 곱한 값입니다. 예를 들어, 1분 기간으로 5개의 데이터 포인트 중 4개를 구성하는 경우 평가 간격은 5분입니다. 10분의 기간으로 3개의 데이터 포인트 중 3개를 구성하는 경우 평가 간격은 30분입니다.

> **Note**
>
> 경보를 생성한 직후 데이터 요소가 누락되고 경보를 생성하기 전에 지표가 CloudWatch에 보고된 경우 CloudWatch는 경보를 평가할 때 경보가 생성되기 전부터 가장 최근의 데이터 요소를 검색합니다.

경보 작업

경보 상태가 OK, ALARM, INSUFFICIENT_DATA 간에 변경될 때 경보가 수행하는 작업을 지정할 수 있습니다.

세 가지 상태 각각으로 전환하기 위한 대부분의 작업을 설정할 수 있습니다. Auto Scaling 작업을 제외한 모든 작업은 상태 전환 시에만 수행되며 상태가 몇 시간 또는 며칠 동안 지속되는 경우 다시 수행되지 않습니다. 임곗값이 위반되면 경보가 이메일을 보내고 위반 조건이 끝나면 또 다른 작업을 보내는 여러 작업이 허용된다는 사실을 활용할 수 있습니다. 이를 통해 규모 조정 또는 복구 작업이 예상한 시점에 트리거되고 원하는 대로 작동하는지 확인할 수 있습니다.

다음이 경보 작업으로 지원됩니다.

- Amazon Simple Notification Service 주제를 사용하여 한 명 이상의 구독자에게 알립니다. 구독자는 개인일 뿐만 아니라 애플리케이션일 수도 있습니다. Amazon SNS에 대한 자세한 내용은 Amazon SNS란 무엇인가요? 단원을 참조하세요.
- Lambda 함수를 간접적으로 호출합니다. 이는 경보 상태 변경에 대한 사용자 지정 작업을 자동화하는 가장 쉬운 방법입니다.
- EC2 지표를 기반으로 하는 경보는 EC2 인스턴스 중지, 종료, 재부팅 또는 복구와 같은 EC2 작업을 수행할 수도 있습니다. 자세한 내용은 EC2 인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성 단원을 참조하십시오.
- 경보는 오토 스케일링의 규모를 조정하는 작업을 수행할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling의 단계 및 단순 크기 조정 정책 단원을 참조하세요.
- 경보는 Systems Manager Ops Center에서 OpsItem을 생성하거나 AWS Systems Manager Incident Manager에서 인시던트를 생성할 수 있습니다. 이러한 작업은 경보가 ALARM 상태가 될 때만 수행됩

니다. 자세한 내용은 경보에서 OpsItem을 생성하도록 CloudWatch 구성 및 인시던트 생성 단원을 참조하세요.

Lambda 경보 작업

CloudWatch 경보는 주어진 상태 변경에 대해 Lambda 함수의 비동기 호출을 보장합니다. 단, 다음과 같은 경우는 예외입니다.

- 함수가 존재하지 않는 경우.
- CloudWatch가 Lambda 함수를 호출할 권한이 없는 경우.

CloudWatch가 Lambda 서비스에 연결할 수 없거나 다른 이유로 메시지가 거부된 경우, CloudWatch는 호출이 성공할 때까지 재시도합니다. Lambda는 메시지를 대기열에 추가하여 실행 재시도를 처리합니다. Lambda가 오류를 처리하는 방법에 대한 정보 등 이 실행 모델에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 비동기 호출을 참조하세요.

동일한 계정이나 다른 AWS 계정에서 Lambda 함수를 호출할 수 있습니다.

Lambda 함수를 경보 작업으로 간접적으로 호출하도록 경보를 지정하는 경우 함수 이름, 함수 별칭 또는 특정 버전의 함수를 지정하도록 선택할 수 있습니다.

Lambda 함수를 경보 작업으로 지정할 때 CloudWatch 서비스 보안 주체가 함수를 호출할 수 있도록 함수에 대한 리소스 정책을 생성해야 합니다.

이를 수행하는 한 가지 방법은 다음 예제와 같이 AWS CLI를 사용하는 것입니다.

```
aws lambda add-permission \
--function-name my-function-name \
--statement-id AlarmAction \
--action 'lambda:InvokeFunction' \
--principal lambda.alarms.cloudwatch.amazonaws.com \
--source-account 111122223333 \
--source-arn arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name
```

또는 다음 예제 중 하나와 비슷한 정책을 생성한 다음, 함수에 할당할 수 있습니다.

다음 예제에서는 경보가 있는 계정을 지정하여 해당 계정(111122223333)의 경보만 함수를 간접적으로 호출할 수 있도록 합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [{
        "Sid": "AlarmAction",
        "Effect": "Allow",
        "Principal": {
            "Service": "lambda.alarms.cloudwatch.amazonaws.com"
        },
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:us-east-1:444455556666:function:function-name",
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "111122223333"
            }
        }
    }]
}
```

다음 예제는 범위가 좁아 지정된 계정의 지정된 경보만 함수를 간접적으로 호출할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "AlarmAction",
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.alarms.cloudwatch.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:444455556666:function:function-name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "111122223333",
          "AWS:SourceArn": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
        }
      }
    }]
}
```

소스 계정을 지정하지 않는 정책은 혼동된 대리자 문제에 취약하므로 이러한 정책은 생성하지 않는 것이 좋습니다.

CloudWatch에서 Lambda로 전송된 이벤트 객체

Lambda 함수를 경보 작업으로 구성하면 CloudWatch는 함수를 간접적으로 호출할 때 Lambda 함수에 JSON 페이로드를 전송합니다. 이 JSON 페이로드는 함수의 이벤트 객체 역할을 합니다. 이 JSON 객체에서 데이터를 추출하여 함수에 사용할 수 있습니다. 다음은 지표 경보의 이벤트 객체에 대한 예제입니다.

```
{
  'source': 'aws.cloudwatch',
  'alarmArn': 'arn:aws:cloudwatch:us-east-1:444455556666:alarm:lambda-demo-metric-
alarm',
  'accountId': '444455556666',
  'time': '2023-08-04T12:36:15.490+0000',
  'region': 'us-east-1',
  'alarmData': {
    'alarmName': 'lambda-demo-metric-alarm',
    'state': {
      'value': 'ALARM',
      'reason': 'test',
      'timestamp': '2023-08-04T12:36:15.490+0000'
    },
    'previousState': {
      'value': 'INSUFFICIENT_DATA',
      'reason': 'Insufficient Data: 5 datapoints were unknown.',
      'reasonData':
 '{"version":"1.0","queryDate":"2023-08-04T12:31:29.591+0000","statistic":"Average","period":60
[],"threshold":5.0,"evaluatedDatapoints":[{"timestamp":"2023-08-04T12:30:00.000+0000"},
{"timestamp":"2023-08-04T12:29:00.000+0000"},
{"timestamp":"2023-08-04T12:28:00.000+0000"},
{"timestamp":"2023-08-04T12:27:00.000+0000"},
{"timestamp":"2023-08-04T12:26:00.000+0000"}]}',
      'timestamp': '2023-08-04T12:31:29.595+0000'
    },
    'configuration': {
      'description': 'Metric Alarm to test Lambda actions',
      'metrics': [
        {
          'id': '1234e046-06f0-a3da-9534-EXAMPLEe4c',
          'metricStat': {
            'metric': {
              'namespace': 'AWS/Logs',
              'name': 'CallCount',
              'dimensions': {
```

```
                'InstanceId': 'i-12345678'
              }
            },
            'period': 60,
            'stat': 'Average',
            'unit': 'Percent'
          },
          'returnData': True
        }
      ]
    }
  }
}
```

다음은 복합 경보의 이벤트 객체에 대한 예제입니다.

```
{
  'source': 'aws.cloudwatch',
  'alarmArn': 'arn:aws:cloudwatch:us-east-1:111122223333:alarm:SuppressionDemo.Main',
  'accountId': '111122223333',
  'time': '2023-08-04T12:56:46.138+0000',
  'region': 'us-east-1',
  'alarmData': {
    'alarmName': 'CompositeDemo.Main',
    'state': {
      'value': 'ALARM',
      'reason': 'arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild transitioned to ALARM at Friday 04
 August, 2023 12:54:46 UTC',
      'reasonData': '{"triggeringAlarms":[{"arn":"arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild","state":
{"value":"ALARM","timestamp":"2023-08-04T12:54:46.138+0000"}}]}',
      'timestamp': '2023-08-04T12:56:46.138+0000'
    },
    'previousState': {
      'value': 'ALARM',
      'reason': 'arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild transitioned to ALARM at Friday 04
 August, 2023 12:54:46 UTC',
      'reasonData': '{"triggeringAlarms":[{"arn":"arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild","state":
{"value":"ALARM","timestamp":"2023-08-04T12:54:46.138+0000"}}]}',
      'timestamp': '2023-08-04T12:54:46.138+0000',
```

```
      'actionsSuppressedBy': 'WaitPeriod',
      'actionsSuppressedReason': 'Actions suppressed by WaitPeriod'
    },
    'configuration': {
      'alarmRule': 'ALARM(CompositeDemo.FirstChild) OR
 ALARM(CompositeDemo.SecondChild)',
      'actionsSuppressor': 'CompositeDemo.ActionsSuppressor',
      'actionsSuppressorWaitPeriod': 120,
      'actionsSuppressorExtensionPeriod': 180
    }
  }
}
```

CloudWatch 경보가 누락 데이터를 처리하는 방법 구성

때로 지표에 대해 예상되는 데이터 요소의 일부가 CloudWatch에 보고되지 않는 경우도 있습니다. 연결이 끊기거나 서버가 정지할 때, 설계에 따라 지표 보고 데이터가 간헐적으로만 전송될 때 이런 일이 일어날 수 있습니다.

CloudWatch를 사용하면 경보를 평가할 때 누락된 데이터 요소를 처리하는 방법을 지정할 수 있습니다. 이렇게 하면 모니터링 중인 데이터 유형에 적합한 경우에만 ALARM 상태가 되도록 경보를 구성할 수 있습니다. 누락된 데이터에 문제가 없는 경우의 거짓 긍정을 피할 수 있습니다.

각 경보가 항상 세 가지 상태 중 하나인 것과 마찬가지로 CloudWatch에 보고된 각각의 특정 데이터 요소는 다음과 같은 세 범주 중 하나에 속합니다.

- 위반하지 않음(임곗값에서)
- 위반(임곗값 위반)
- 누락됨

각 경보에 대해 CloudWatch가 누락된 데이터 요소를 다음 중 하나로 처리하도록 지정할 수 있습니다.

- notBreaching – 누락 데이터 요소를 '양호'하고 임곗값 내에 있는 것으로 처리합니다.
- breaching - 누락 데이터 요소를 '불량'하고 임곗값을 위반한 것으로 처리합니다
- ignore - 현재 경보 상태를 유지합니다.
- missing - 경보 평가 범위의 모든 데이터 요소가 누락된 경우 경보를 INSUFFICIENT_DATA 상태로 전환합니다.

가장 좋은 선택은 지표 유형과 경보 용도에 따라 다릅니다. 예를 들어 데이터를 지속적으로 보고하는 지표를 사용하여 애플리케이션 롤백 경보를 생성하는 경우 누락된 데이터 포인트는 문제를 나타낼 수 있으므로 위반으로 처리하는 것이 좋습니다. 그러나 Amazon DynamoDB의 `ThrottledRequests`와 같이 오류가 발생할 때만 데이터 요소를 생성하는 지표의 경우 누락 데이터를 `notBreaching`으로 처리할 수 있습니다. 기본값은 `missing`입니다.

> **Important**
>
> Amazon EC2 지표에 구성된 경보는 누락된 지표 데이터 포인트가 있는 경우 일시적으로 INSUFICITENT_DATA 상태로 전환될 수 있습니다. 드물기는 하지만, Amazon EC2 인스턴스가 정상인 경우에도 지표 보고가 중단되면 이 문제가 발생할 수 있습니다. 중지, 종료, 재부팅 또는 복구 작업을 수행하도록 구성된 Amazon EC2 지표에 대한 경보의 경우 누락된 데이터를 `missing`으로 처리하고 경보 상태일 때만 해당 경보를 트리거하도록 경보를 구성하는 것이 좋습니다.

경보에 대한 최상의 옵션을 선택하면 불필요하고 오해의 소지가 있는 경보 조건 변경을 막을 수 있으며, 시스템 상태를 보다 정확하게 나타낼 수 있습니다.

> **Important**
>
> 누락된 데이터를 처리하는 방법에 대해 다른 옵션을 선택하더라도 `AWS/DynamoDB` 네임스페이스에서 메트릭을 평가하는 경보는 항상 누락된 데이터를 무시합니다. `AWS/DynamoDB` 메트릭에 누락된 데이터가 있는 경우 해당 지표를 평가하는 경보는 현재 상태를 유지합니다.

데이터가 누락되었을 때 경보 상태 평가 방법

경보가 상태 변경 여부를 평가할 때마다 CloudWatch는 [평가 기간(Evaluation Periods)]으로 지정된 수보다 더 높은 수의 데이터 요소를 검색하려고 합니다. 검색을 시도하는 데이터 포인트의 정확한 수는 경보 기간의 길이, 표준 해상도 또는 고해상도 지표에 토대를 두고 있는지 여부에 따라 달라집니다. 검색을 시도하는 데이터 포인트의 기간이 평가 범위입니다.

CloudWatch가 이러한 데이터 요소를 검색하면 다음과 같이 진행됩니다.

- 평가 범위 내에 누락된 데이터 요소가 없다면 CloudWatch는 가장 최근에 수집된 데이터 요소를 기반으로 경보를 평가합니다. 평가된 데이터 요소의 수는 경보의 [평가 기간(Evaluation Periods)]과 같습니다. 평가 범위보다 훨씬 이전의 추가 데이터 요소는 필요하지 않으며 무시됩니다.

- 평가 범위의 일부 데이터 요소가 누락되었지만 평가 범위에서 성공적으로 검색된 기존 데이터 요소의 총수가 경보의 [평가 기간(Evaluation Periods)] 이상인 경우 CloudWatch는 평가 범위보다 훨씬 이전의 필요한 추가 데이터 요소를 포함하여 성공적으로 검색된 가장 최근의 실제 데이터 요소를 기반으로 경보 상태를 평가합니다. 이 경우 누락 데이터 처리 방법에 대한 값이 필요 없으며, 이를 무시합니다.
- 평가 범위의 일부 데이터 요소가 누락되었으며 검색된 실제 데이터 요소의 수가 경보의 [평가 기간(Evaluation Periods)] 수보다 적은 경우 CloudWatch는 누락 데이터 처리 방법에 대해 지정된 결과로 누락 데이터 요소를 채운 다음, 경보를 평가합니다. 그러나 평가 범위의 실제 데이터 요소는 모두 평가에 포함됩니다. CloudWatch는 되도록 몇 번만 누락 데이터 요소를 사용합니다.

> **Note**
>
> 이 동작의 특정 사례에서는 CloudWatch 경보가 지표 흐름이 중지된 후 일정 기간 동안 마지막 데이터 요소 집합을 반복적으로 재평가할 수 있습니다. 이 재평가를 통해 지표 스트림 중지 직전에 상태가 변한 경우 경보가 상태를 변경하고 작업을 다시 실행할 수 있습니다. 이 동작을 완화하려면 더 짧은 기간을 사용하십시오.

다음은 경보 평가 동작에 대한 예를 설명한 테이블입니다. 첫 번째 표에서 [경보에 대한 데이터 요소(Datapoints to Alarm)]와 [평가 기간(Evaluation Periods)]은 둘 다 3입니다. CloudWatch는 가장 최근 3개의 데이터 요소 중 일부가 누락된 경우 경보를 평가할 때 가장 최근 데이터 요소 5개를 검색합니다. 5는 경보의 평가 범위입니다.

평가 범위가 5이므로 1열에는 가장 최근 데이터 요소 5개가 표시됩니다. 이러한 데이터 요소는 가장 최근 데이터 요소가 오른쪽에 표시됩니다. 0는 비위반 데이터 요소, X는 위반 데이터 요소, -는 누락 데이터 요소입니다.

2열은 필요한 데이터 요소 3개 중 누락된 개수를 표시합니다. 가장 최신 데이터 요소 5개가 평가되었더라도 경보 상태 평가를 위해 3개(Evaluation Periods(평가 기간)에 대한 설정)만 필요합니다. 2열의 데이터 요소 개수는 누락된 데이터 요소 처리 방법에 대한 설정을 사용하여 반드시 "채워야" 하는 데이터 요소의 수입니다.

3~6열의 열 머리글은 누락 데이터를 처리하는 방법으로 가능한 값입니다. 이러한 열의 행에는 누락 데이터를 처리할 수 있는 이러한 각 방법에 대해 설정된 경보 상태가 표시됩니다.

데이터 포인트	채워야 하는 데이터 요소 수	MISSING	IGNORE	위반	위반하지 않음
0 - X - X	0	OK	OK	OK	OK
- - - - 0	2	OK	OK	OK	OK
- - - - -	3	INSUFFICIENT_DATA	현재 상태 유지	ALARM	OK
0 X X - X	0	ALARM	ALARM	ALARM	ALARM
- - X - -	2	ALARM	현재 상태 유지	ALARM	OK

앞 테이블의 2행에서는 누락 데이터를 위반으로 처리하는 경우에도 경보 상태는 OK로 유지됩니다. 기존 데이터 포인트 중 하나가 위반 상태가 아니며, 위반으로 처리되는 2개의 누락 데이터 포인트와 함께 이를 평가하기 때문입니다. 다음번에 이 경보를 평가할 때도 데이터가 여전히 누락된 경우 해당 비위반 데이터 요소가 더 이상 평가 범위에 있지 않기 때문에 경보는 ALARM 상태가 됩니다.

가장 최근의 데이터 요소 5개가 모두 누락된 세 번째 행은 누락 데이터 처리 방법에 대한 다양한 설정이 경보 상태에 어떻게 영향을 주는지를 보여 줍니다. 누락된 데이터 요소를 위반으로 간주하는 경우 경보는 ALARM 상태가 되지만, 비위반으로 간주하는 경우 경보는 OK 상태가 됩니다. 누락된 데이터 요소를 무시하면 경보는 누락 데이터 요소 이전의 현재 상태를 유지합니다. 그리고 누락된 데이터 요소를 단지 누락으로 간주한다면 경보에 평가를 수행하기에 충분한 최근 실제 데이터가 없으며 경보는 INSUFFICIENT_DATA 상태가 됩니다.

네 번째 행에서는 가장 최근의 데이터 요소 세 개가 위반이며 경보의 [평가 기간(Evaluation Periods)]과 [경보에 대한 데이터 요소(Datapoints to Alarm)]가 둘 다 3으로 설정되어 있으므로 경보가 모든 경우에 ALARM 상태가 됩니다. 이 경우 누락된 데이터 요소는 무시되며 누락 데이터 평가 방법에 대한 설정은 필요하지 않습니다. 평가할 실제 데이터 요소가 3개 있기 때문입니다.

5행은 '조기 경보 상태'라고 하는 특별한 경우의 경보 평가를 나타냅니다. 자세한 내용은 경보 상태 조기 전환 방지 단원을 참조하세요.

다음 테이블의 경우 기간은 다시 5분이며, Datapoints to Alarm(경보에 대한 데이터 포인트)는 2, Evaluation Periods(평가 기간)는 3입니다. 'N 중 M' 경보는 '3 중 2'입니다.

평가 범위는 5입니다. 이것은 검색되는 최근 데이터 포인트의 최대 수이며 일부 데이터 포인트가 누락된 경우 사용할 수 있습니다.

데이터 포인트	누락 데이터 포인트 가운데 수(#)	누락	IGNORE	위반	위반하지 않음
0 - X - X	0	ALARM	ALARM	ALARM	ALARM
0 0 X 0 X	0	ALARM	ALARM	ALARM	ALARM
0 - X - -	1	OK	OK	ALARM	OK
- - - - 0	2	OK	OK	ALARM	OK
- - - - X	2	ALARM	현재 상태 유지	ALARM	OK

1행과 2행에서는 가장 최근의 데이터 요소 3개 중 2개가 위반이므로 경보는 항상 ALARM 상태가 됩니다. 2행에서는 가장 최근 데이터 요소 3개가 누락되지 않았기 때문에 평가 범위에서 가장 오래된 두 데이터 요소가 필요하지 않으므로 오래된 이 두 데이터 요소는 무시됩니다.

3행과 4행에서는 누락된 데이터를 위반으로 처리하는 경우에만 경보가 ALARM 상태가 됩니다(여기서는 가장 최근의 누락 데이터 요소 두 개가 모두 위반으로 처리됨). 4행에서 위반으로 처리되는 이 두 누락 데이터 요소는 ALARM 상태를 트리거하는 데 필요한 위반 데이터 요소 두 개를 제공합니다.

5행은 '조기 경보 상태'라고 하는 특별한 경우의 경보 평가를 나타냅니다. 자세한 내용은 다음 섹션을 참조하세요.

경보 상태 조기 전환 방지

CloudWatch 경보 평가에는 데이터가 간헐적일 때 조기에 경보가 ALARM 상태가 되는 거짓 경보를 방지하기 위한 로직이 포함됩니다. 이전 단원에 있는 표의 5행에 표시된 예가 이 로직을 보여 줍니다. 해당 행과 다음 예에서 [평가 기간(Evaluation Periods)]은 3이고 평가 범위는 데이터 요소 5개입니다. [경보에 대한 데이터 요소(Datapoints to Alarm)]는 3입니다. 단, [경보에 대한 데이터 요소(Datapoints to Alarm)]가 2인 'M out of N(N 중 M)' 예는 예외입니다.

경보의 가장 최근 데이터가 - - - - X이며 누락 데이터 요소 4개가 있고 가장 최근 데이터 요소로 위반 데이터 요소가 있다고 가정합니다. 다음 데이터 요소는 비위반일 수 있으므로 데이터가 - - -

- X 또는 - - - X -이고 [경보에 대한 데이터 요소(Datapoints to Alarm)]가 3일 경우 경보는 즉시 ALARM 상태가 되지 않습니다. 이렇게 하면 다음 데이터 요소가 비위반이고 데이터가 - - - X 0 또는 - - X - 0가 되도록 하는 경우 거짓 긍정이 방지됩니다.

그러나 마지막 몇 개의 데이터 요소가 - - X - -인 경우 누락된 데이터 요소를 누락으로 처리하더라도 경보는 ALARM 상태가 됩니다. 이는 [평가 기간(Evaluation Periods)] 동안 사용 가능한 가장 오래된 위반 데이터 요소 수가 최소한 [경보에 대한 데이터 요소(Datapoints to Alarm)]의 값만큼 오래되고 더 최근의 다른 모든 데이터 요소가 위반 또는 누락인 경우 경보가 항상 ALARM 상태가 되도록 설계되었기 때문입니다. 이 경우 사용 가능한 데이터 요소의 총수가 M([경보에 대한 데이터 요소(Datapoints to Alarm)])보다 적더라도 경보는 ALARM 상태가 됩니다.

이 경보 로직은 'M out of N(N 중 M)' 경보에도 적용됩니다. 평가 범위 동안 가장 오래된 위반 데이터 요소가 최소한 [경보에 대한 데이터 요소(Datapoints to Alarm)]의 값만큼 오래되고 더 최근의 모든 데이터 요소가 위반 또는 누락인 경우 경보는 M([경보에 대한 데이터 요소(Datapoints to Alarm)]) 값에 상관없이 ALARM 상태가 됩니다.

고분해능 경보

고분해능 지표에 대해 경보를 설정할 경우 고분해능 경보를 10초 또는 30초 기간으로 지정하거나 60초의 배수 기간으로 정기 경보를 설정할 수 있습니다. 고분해능 경보는 요금이 더 비쌉니다. 고분해능 지표에 대한 자세한 내용은 사용자 지정 지표 게시 단원을 참조하세요.

수학 표현식에 대한 경보

하나 이상의 CloudWatch 지표를 기반으로 하는 수학 표현식의 결과에 대한 경보를 설정할 수 있습니다. 경보에 사용되는 수학 표현식에는 지표를 10개까지 포함할 수 있습니다. 각 지표의 기간은 동일해야 합니다.

수학 표현식을 기반으로 하는 경보의 경우 누락된 데이터 포인트를 처리하는 방법을 CloudWatch에 지정할 수 있습니다. 이 경우 수학 표현식이 해당 데이터 포인트에 대한 값을 반환하지 않으면 데이터 포인트가 누락된 것으로 간주됩니다.

수학 표현식을 기반으로 하는 경보는 Amazon EC2 작업을 수행할 수 없습니다.

지표 수학 표현식 및 구문에 대한 자세한 내용은 지표 수학 사용 단원을 참조하세요.

백분위수 기반 CloudWatch 경보 및 데이터 샘플 부족

경보를 위한 통계로 백분위수를 설정하면 정확한 통계 평가를 위한 데이터가 충분하지 않을 때 어떻게 할 것인지 지정할 수 있습니다. 경보가 통계를 어떻게든 평가하도록 하고 가능하면 경보 상태를 변경하도록 선택할 수 있습니다. 또는 샘플 크기가 작을 때 경보가 지표를 무시하고 통계적으로 의미가 있을 정도로 충분한 데이터가 모일 때까지 기다렸다가 평가할 수 있습니다.

0.5(포함) ~ 1.00(제외) 범위 백분위수의 경우, 평가 기간 동안 10/(1-백분위수) 보다 적은 데이터 포인트가 있을 때 이 설정이 사용됩니다. 예를 들어 p99 백분위수에서 경보 샘플이 1,000개보다 적을 경우 이 설정이 사용됩니다. 0 ~ 0.5(제외) 범위 백분위수의 경우, 10/백분위수 보다 적은 백분위수가 있을 때 이 설정이 사용됩니다.

CloudWatch 경보의 공통 기능

다음 기능은 모든 CloudWatch 경보에 적용됩니다.

- 생성할 수 있는 경보 수에는 제한이 없습니다. 경보를 생성하거나 업데이트하려면 CloudWatch 콘솔, PutMetricAlarm API 작업 또는 AWS CLI의 put-metric-alarm 명령을 사용합니다.
- 경보 이름은 UTF-8 문자만 포함해야 하고 ASCII 제어 문자를 포함할 수 없습니다.
- CloudWatch 콘솔, DescribeAlarms API 작업 또는 AWS CLI의 describe-alarms 명령을 사용하여 현재 구성된 경보의 일부 또는 전체를 나열하고 특정 상태의 경보를 나열할 수 있습니다.
- DisableAlarmActions 및 EnableAlarmActions API 작업 또는 AWS CLI의 disable-alarm-actions 및 enable-alarm-actions 명령을 사용하여 경보를 사용 설정 및 사용 중지할 수 있습니다.
- SetAlarmState API 작업 또는 AWS CLI의 set-alarm-state 명령을 사용함으로써 경보를 임의의 상태로 설정하여 경보를 테스트할 수 있습니다. 이러한 일시적인 상태 변경은 다음 경보 비교 시까지만 지속됩니다.
- 사용자 지정 지표를 생성하기 전에 사용자 지정 지표에 대한 경보를 생성할 수 있습니다. 경보가 유효하려면 사용자 지정 지표에 대한 모든 측정기준을 비롯해 지표 네임스페이스 및 지표 이름을 경보 정의에 포함시켜야 합니다. 이렇게 하려면 PutMetricAlarm API 작업 또는 AWS CLI의 put-metric-alarm 명령을 사용하면 됩니다.
- CloudWatch 콘솔, DescribeAlarmHistory API 작업 또는 AWS CLI의 describe-alarm-history 명령을 사용하여 경보 기록을 볼 수 있습니다. CloudWatch는 30일 동안 경보 기록을 유지합니다. 각 상태 전환은 고유한 타임스탬프로 표시됩니다. 드문 경우지만 기록에 상태 변경에 대한 알림이 두 개 이상 있을 수 있습니다. 이 경우 타임스탬프를 사용하여 고유한 상태 변경을 확인할 수 있습니다.

- CloudWatch 콘솔 탐색 창의 즐겨찾기 및 최근 항목(Favorites and recents) 옵션에서 즐겨찾기에 추가하려는 경보 위에 마우스를 놓고 옆에 있는 별 기호를 선택하여 경보를 즐겨찾기에 추가할 수 있습니다.
- 경보에 대한 평가 기간의 수에 각 평가 기간의 길이를 곱한 값이 1일을 초과할 수 없습니다.

> **Note**
>
> 일부 AWS 리소스는 특정 조건에서 지표 데이터를 CloudWatch에 전송하지 않습니다. 예를 들어 Amazon EBS는 Amazon EC2 인스턴스에 연결되지 않은 사용 가능한 볼륨에 대한 지표 데이터를 전송하지 않을 수 있습니다. 해당 볼륨에 대해 모니터링할 지표 활동이 없기 때문입니다. 이러한 지표에 대한 경보 세트가 있으면 상태가 `INSUFFICIENT_DATA`로 변경됩니다. 이는 리소스가 비활성 상태임을 나타내지만 그렇다고 반드시 문제가 있음을 의미하지는 않습니다. 각 경보가 누락된 데이터를 처리하는 방법을 지정할 수 있습니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하십시오.

AWS 서비스에 대한 모범 사례 경보 권장 사항

CloudWatch는 경보 권장 사항을 기본으로 제공합니다. 이는 다른 AWS 서비스에서 게시한 지표에 대해 생성할 것을 권장하는 CloudWatch 경보입니다. 이러한 권장 사항은 모니터링의 모범 사례를 따르기 위해 경보를 설정해야 하는 지표를 식별하는 데 도움이 됩니다. 또한 권장 사항에서는 설정할 경보 임곗값을 제안합니다. 권장 사항을 따르면 AWS 인프라에 대한 중요한 모니터링을 놓치지 않을 수 있습니다.

경보 권장 사항을 찾으려면 CloudWatch 콘솔의 지표 섹션을 사용하여 경보 권장 사항 필터 토글을 선택합니다. 콘솔에서 권장 경보로 이동한 다음 권장 경보를 생성하면 CloudWatch에서 일부 경보 설정을 미리 채울 수 있습니다. 일부 권장 경보의 경우 경보 임곗값도 미리 입력됩니다. 콘솔을 사용하여 권장 경보에 대한 코드형 인프라 경보 정의를 다운로드한 다음 이 코드를 사용하여 AWS CloudFormation, AWS CLI 또는 Terraform에서 경보를 생성할 수도 있습니다.

권장되는 경보에서 권장 경보 목록을 볼 수도 있습니다.

생성한 경보에 대해서는 CloudWatch에서 생성한 다른 모든 경보와 동일한 요금이 부과됩니다. 권장 사항을 사용하면 추가 요금이 발생하지 않습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

권장 알람 찾고 생성하기

다음 단계에 따라 CloudWatch에서 경보 설정을 권장하는 지표를 찾고 선택적으로 경보 중 하나를 생성합니다. 첫 번째 절차에서는 권장 경보가 있는 지표를 찾는 방법과 이러한 경보 중 하나를 생성하는 방법을 설명합니다.

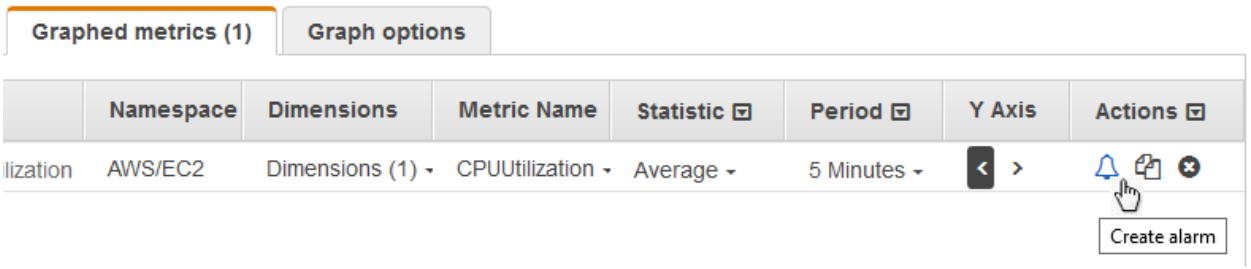
AWS/Lambda 또는 AWS/S3와 같은 AWS 네임스페이스의 모든 권장 경보에 대한 코드형 인프라 경보 정의를 대량으로 다운로드할 수도 있습니다. 관련 지침은 이 주제의 후반부에서 설명합니다.

권장 경보가 포함된 지표를 찾고 단일 권장 경보 생성하기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 모든 지표를 선택합니다.
3. 지표 표에서 경보 권장 사항을 선택합니다.

   지표 네임스페이스 목록은 경보 권장 사항이 있고 계정의 서비스가 게시하고 있는 지표만 포함하도록 필터링됩니다.
4. 서비스의 네임스페이스를 선택합니다.

   이 네임스페이스의 지표 목록은 경보 권장 사항이 있는 지표만 포함하도록 필터링됩니다.
5. 지표에 대한 경보 의도와 권장 임곗값을 보려면 세부 정보 보기를 선택합니다.
6. 지표 중 하나에 대한 경보를 생성하려면 다음 중 하나를 수행합니다.
   - 콘솔을 사용하여 경보를 생성하려면 다음을 수행합니다.
     a. 지표의 확인란을 선택하고 그래프로 표시된 지표 탭을 선택합니다.
     b. 경보 아이콘을 선택합니다.



     경보 권장 사항에 따라 지표 이름, 통계, 기간이 채워진 경보 생성 마법사가 나타납니다. 권장 사항에 특정 임곗값이 포함된 경우 해당 값도 미리 채워집니다.

     c. 다음을 선택합니다.

d. 알림에서 경보가 ALARM 상태, OK 상태 또는 INSUFFICIENT_DATA 상태로 전환될 때 알릴 SNS 주제를 선택합니다.

   경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

   경보에서 알림을 보내지 않게 하려면 제거를 선택합니다.

e. 경보가 Auto Scaling 또는 EC2 작업을 수행하도록 하려면 해당 버튼을 선택하고 경보 상태 및 수행할 작업을 선택합니다.

f. 마친 후에는 다음을 선택합니다.

g. 경보 이름 및 설명을 입력합니다. 이름은 ASCII 문자만 포함해야 합니다. 그리고 다음(Next)을 선택합니다.

h. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

- 코드형 인프라 경보 정의를 다운로드하여 AWS CloudFormation, AWS CLI 또는 Terraform에서 사용하려면 경보 코드 다운로드를 선택하고 원하는 형식을 선택합니다. 다운로드한 코드에는 지표 이름, 통계, 임곗값에 대한 권장 설정이 포함됩니다.

AWS 서비스의 모든 권장 경보에 대한 코드형 인프라 경보 정의 다운로드하기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 모든 지표를 선택합니다.
3. 지표 표에서 경보 권장 사항을 선택합니다.

   지표 네임스페이스 목록은 경보 권장 사항이 있고 계정의 서비스가 게시하고 있는 지표만 포함하도록 필터링됩니다.
4. 서비스의 네임스페이스를 선택합니다.

   이 네임스페이스의 지표 목록은 경보 권장 사항이 있는 지표만 포함하도록 필터링됩니다.
5. 다운로드 경보 코드는 이 네임스페이스의 지표에 대해 권장되는 경보 수를 표시합니다. 모든 권장 경보에 대한 코드형 인프라 경보 정의를 다운로드하려면 경보 코드 다운로드를 선택한 다음 원하는 코드 형식을 선택합니다.

권장되는 경보

다음 섹션에는 모범 사례 경보의 설정을 권장하는 지표가 나열되어 있습니다. 각 지표에 대해 측정 기준, 경보 의도, 권장 임곗값, 임곗값 근거, 기간 및 데이터 포인트 수도 표시됩니다.

일부 지표는 목록에 두 번 등장할 수 있습니다. 이는 해당 지표의 다른 측정 기준 조합에 대해 다른 경보가 권장되는 경우에 발생합니다.

경보를 보낼 데이터 포인트는 경보를 ALARM 상태로 보내기 위해 위반해야 하는 데이터 포인트의 수입니다. 평가 기간은 경보를 평가할 때 고려되는 기간의 수입니다. 이 숫자가 같으면 연속된 기간 수의 값이 임곗값을 초과한 경우에만 경보가 ALARM 상태로 전환됩니다. 경보를 보낼 데이터 포인트가 평가 기간보다 낮으면 "M out of N" 경보가 되며, 데이터 포인트로 구성된 평가 기간 내에 적어도 경보를 보낼 데이터 포인트의 데이터 포인트가 위반되면 경보가 ALARM 상태로 전환됩니다. 자세한 내용은 경보 평가 단원을 참조하십시오.

주제

- Amazon API Gateway
- Amazon EC2 Auto Scaling
- Amazon CloudFront
- Amazon Cognito
- Amazon DynamoDB
- Amazon EBS
- Amazon EC2
- Amazon ElastiCache
- Amazon EC2(AWS/ElasticGPUs)
- Amazon ECS
- Container Insights가 포함된 Amazon ECS
- Amazon EFS
- Container Insights가 포함된 Amazon EKS
- Amazon Kinesis Data Streams
- Lambda
- Lambda Insights
- Amazon VPC(AWS/NATGateway)
- AWS 프라이빗 링크(AWS/PrivateLinkEndpoints)

- AWS 프라이빗 링크(AWS/PrivateLinkServices)
- Amazon RDS
- Amazon Route 53 Public Data Plane
- Amazon S3
- S3ObjectLambda
- Amazon SNS
- Amazon SQS
- AWS VPN

Amazon API Gateway

4XXError

측정 기준: ApiName, 스테이지

경보 설명: 이 경보는 높은 비율의 클라이언트 측 오류를 감지합니다. 이는 권한 부여 또는 클라이언트 요청 매개변수에 문제가 있음을 의미할 수 있습니다. 리소스가 제거되었거나 클라이언트가 존재하지 않는 리소스를 요청하고 있음을 의미할 수도 있습니다. CloudWatch Logs를 활성화하고 4XX 오류의 원인이 될 수 있는 오류가 있는지 확인해 보세요. 또한 상세한 CloudWatch 지표를 활성화하여 리소스 및 방법별로 이 지표를 확인하고 오류의 원인을 좁히는 것도 고려해 보세요. 구성된 제한 한도를 초과하여 오류가 발생할 수도 있습니다. 응답 및 로그에서 예상치 못한 오류가 429개라는 높은 비율로 보고되는 경우 이 가이드에 따라 문제를 해결하세요.

인텐트: 이 경보는 API Gateway 요청에 대해 높은 비율의 클라이언트 측 오류를 탐지할 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 제안된 임곗값은 전체 요청의 5% 이상에서 4XX 오류가 발생하는 경우를 감지합니다. 하지만 요청의 트래픽과 허용 가능한 오류 발생률에 맞게 임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다. 자주 발생하는 4XX 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

5XXError

측정 기준: ApiName, 스테이지

경보 설명: 이 경보는 높은 비율의 서버 측 오류 감지에 도움이 됩니다. 이는 API 백엔드, 네트워크 또는 API 게이트웨이와 백엔드 API 간의 통합에 문제가 있음을 나타낼 수 있습니다. 이 설명서는 5xx 오류의 원인을 해결하는 데 도움이 될 수 있습니다.

인텐트: 이 경보는 API Gateway 요청에 대해 높은 비율의 서버 측 오류를 탐지할 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 제안된 임곗값은 전체 요청의 5% 이상에서 5XX 오류가 발생하는 경우를 감지합니다. 그러나 허용 가능한 오류 발생률뿐만 아니라 요청의 트래픽에 따라 임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대한 허용 가능한 오류 발생율을 확인한 다음 그에 따라 임곗값을 조정할 수도 있습니다. 자주 발생하는 5XX 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: GREATER_THAN_THRESHOLD

개수

측정 기준: ApiName, 스테이지

경보 설명: 이 경보는 REST API 단계에서 낮은 트래픽 볼륨을 감지하는 데 도움이 됩니다. 이는 잘못된 엔드포인트 사용과 같이 API를 호출하는 애플리케이션과 관련된 문제를 나타내는 지표일 수 있습니다. 또한 API의 구성 또는 권한 문제로 인해 클라이언트가 연결할 수 없게 되었음을 나타내는 지표일 수도 있습니다.

인텐트: 이 경보는 REST API 단계에서 예기치 않게 낮은 트래픽 볼륨을 감지할 수 있습니다. API가 정상 조건에서 예측 가능하고 일관된 수의 요청을 수신하는 경우 이 경보를 생성하는 것이 좋습니다. 자세한 CloudWatch 지표를 활성화하고 방법 및 리소스별 정상 트래픽 볼륨을 예측할 수 있는 경우, 각 리소스 및 방법에 대한 트래픽 볼륨 감소를 보다 세밀하게 모니터링할 수 있도록 대체 경보를 생성하는 것이 좋습니다. 이 경보는 일정하고 일관된 트래픽이 예상되지 않는 API에는 권장되지 않습니다.

통계: SampleCount

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 과거 데이터 분석을 기반으로 임곗값을 설정하여 API의 예상 기준 요청 수를 결정합니다. 임곗값을 매우 높게 설정하면 정상적이고 트래픽이 적을 것으로 예상되는 기간에 경보가 너무 민감해질 수 있습니다. 반대로 값을 매우 낮게 설정하면 경보가 이례적으로 작은 트래픽 볼륨 감소를 놓칠 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: LESS_THAN_THRESHOLD

개수

측정 기준: ApiName, 스테이지, 리소스, 방법

경보 설명: 이 경보는 해당 스테이지에서 REST API 리소스 및 방법에 대해 낮은 트래픽 볼륨을 감지하는 데 도움이 됩니다. 이는 잘못된 엔드포인트 사용과 같이 API를 호출하는 애플리케이션과 관련된 문제를 나타내는 지표일 수 있습니다. 또한 API의 구성 또는 권한 문제로 인해 클라이언트가 연결할 수 없게 되었음을 나타내는 지표일 수도 있습니다.

인텐트: 이 경보는 해당 단계의 REST API 리소스 및 방법에 대해 예기치 않게 낮은 트래픽 볼륨을 감지할 수 있습니다. API가 정상 조건에서 예측 가능하고 일관된 수의 요청을 수신하는 경우 이 경보를 생성하는 것이 좋습니다. 이 경보는 일정하고 일관된 트래픽이 예상되지 않는 API에는 권장되지 않습니다.

통계: SampleCount

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 과거 데이터 분석을 기반으로 임곗값을 설정하여 API의 예상 기준 요청 수를 결정합니다. 임곗값을 매우 높게 설정하면 정상적이고 트래픽이 적을 것으로 예상되는 기간에 경보가 너무 민감해질 수 있습니다. 반대로 값을 매우 낮게 설정하면 경보가 이례적으로 작은 트래픽 볼륨 감소를 놓칠 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: LESS_THAN_THRESHOLD

개수

측정 기준: ApiId, 스테이지

경보 설명: 이 경보는 HTTP API 단계에서 낮은 트래픽 볼륨을 감지하는 데 도움이 됩니다. 이는 잘못된 엔드포인트 사용과 같이 API를 호출하는 애플리케이션과 관련된 문제를 나타내는 지표일 수 있습니다. 또한 API의 구성 또는 권한 문제로 인해 클라이언트가 연결할 수 없게 되었음을 나타내는 지표일 수도 있습니다.

인텐트: 이 경보는 HTTP API 단계에서 예기치 않게 낮은 트래픽 볼륨을 감지할 수 있습니다. API가 정상 조건에서 예측 가능하고 일관된 수의 요청을 수신하는 경우 이 경보를 생성하는 것이 좋습니다. 자세한 CloudWatch 지표를 활성화하고 경로별 정상 트래픽 볼륨을 예측할 수 있는 경우, 각 경로에 대한 트래픽 볼륨 감소를 보다 세밀하게 모니터링할 수 있도록 대체 경보를 생성하는 것이 좋습니다. 이 경보는 일정하고 일관된 트래픽이 예상되지 않는 API에는 권장되지 않습니다.

통계: SampleCount

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 과거 데이터 분석을 기반으로 임곗값을 설정하여 API의 예상 기준 요청 수를 결정합니다. 임곗값을 매우 높게 설정하면 정상적이고 트래픽이 적을 것으로 예상되는 기간에 경보가 너무 민감해질 수 있습니다. 반대로 값을 매우 낮게 설정하면 경보가 이례적으로 작은 트래픽 볼륨 감소를 놓칠 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: LESS_THAN_THRESHOLD

개수

측정 기준: ApiId, 스테이지, 리소스, 방법

경보 설명: 이 경보는 해당 단계의 HTTP API에 대해 낮은 트래픽 볼륨을 감지하는 데 도움이 됩니다. 이는 잘못된 엔드포인트 사용과 같이 API를 호출하는 애플리케이션과 관련된 문제를 나타내는 지표일 수 있습니다. 또한 API의 구성 또는 권한 문제로 인해 클라이언트가 연결할 수 없게 되었음을 나타내는 지표일 수도 있습니다.

인텐트: 이 경보는 해당 스테이지에서 HTTP API 경로에 대해 예기치 않게 낮은 트래픽 볼륨을 감지할 수 있습니다. API가 정상 조건에서 예측 가능하고 일관된 수의 요청을 수신하는 경우 이 경보를 생성하는 것이 좋습니다. 이 경보는 일정하고 일관된 트래픽이 예상되지 않는 API에는 권장되지 않습니다.

통계: SampleCount

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 과거 데이터 분석을 기반으로 임곗값을 설정하여 API의 예상 기준 요청 수를 결정합니다. 임곗값을 매우 높게 설정하면 정상적이고 트래픽이 적을 것으로 예상되는 기간에 경보가 너무 민감해질 수 있습니다. 반대로 값을 매우 낮게 설정하면 경보가 이례적으로 작은 트래픽 볼륨 감소를 놓칠 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: LESS_THAN_THRESHOLD

IntegrationLatency

측정 기준: ApiId, 스테이지

경보 설명: 이 경보는 스테이지에서 API 요청에 대한 통합 지연 시간이 높은지 감지하는 데 도움이 됩니다. `IntegrationLatency` 지표 값을 백엔드의 해당 지연 시간 지표(예: Lambda 통합에 대한 `Duration` 지표)와 상호 연관시킬 수 있습니다. 이를 통해 성능 문제로 인해 API 백엔드가 클라이언트의 요청을 처리하는 데 더 오랜 시간이 걸리는지 또는 초기화 또는 콜드 시작으로 인해 다른 오버헤드가 발생하는지 확인할 수 있습니다. 또한 API에 CloudWatch Logs를 활성화하고 로그에서

높은 지연 시간 문제를 일으킬 수 있는 오류가 있는지 확인하는 것도 고려해 보세요. 또한 통합 지연 시간의 원인을 좁히는 데 도움이 되도록 상세한 CloudWatch 지표를 활성화하고 경로별로 이 지표를 확인하는 것도 고려해 보세요.

인텐트: 이 경보는 스테이지의 API Gateway 요청에서 통합 지연 시간이 긴 경우를 감지할 수 있습니다. WebSocket API에는 이 경보를 사용하는 것이 좋으며, HTTP API에는 지연 시간 지표에 대한 별도의 경보 권장 사항이 이미 있으므로 HTTP API의 경우 선택 사항으로 간주합니다. 자세한 CloudWatch 지표를 활성화하고 경로별로 통합 지연 시간 성능 요구 사항이 다른 경우, 각 경로의 통합 지연 시간을 보다 세밀하게 모니터링하려면 대체 경보를 생성하는 것이 좋습니다.

통계: p90

권장 임곗값: 2000.0

임곗값 정당화: 제안된 임곗값이 모든 API 워크로드에 적용되는 것은 아닙니다. 그러나 임곗값의 시작점으로 사용할 수 있습니다. 그런 다음 워크로드와 API의 허용 가능한 지연 시간, 성능, SLA 요구 사항에 따라 다른 임곗값을 선택할 수 있습니다. 일반적으로 API의 지연 시간이 길어도 괜찮다면 임곗값을 더 높게 설정하여 경보의 민감도를 낮추세요. 하지만 API가 거의 실시간에 가까운 응답을 제공할 것으로 예상되는 경우 임곗값을 더 낮게 설정하세요. 또한 과거 데이터를 분석하여 애플리케이션 워크로드의 예상 기준 지연 시간을 파악한 다음 그에 따라 임곗값을 적절히 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

IntegrationLatency

측정 기준: ApiId, 스테이지, 경로

경보 설명: 이 경보는 스테이지에서 경로에 대한 WebSocket API 요청의 통합 지연 시간이 높은지 감지하는 데 도움이 됩니다. `IntegrationLatency` 지표 값을 백엔드의 해당 지연 시간 지표(예: Lambda 통합에 대한 `Duration` 지표)와 상호 연관시킬 수 있습니다. 이를 통해 성능 문제로 인해 API 백엔드가 클라이언트의 요청을 처리하는 데 더 오랜 시간이 걸리는지 또는 초기화 또는 콜드 시작으로 인해 다른 오버헤드가 발생하는지 확인할 수 있습니다. 또한 API에 CloudWatch Logs를 활성화하고 로그에서 높은 지연 시간 문제를 일으킬 수 있는 오류가 있는지 확인하는 것도 고려해 보세요.

인텐트: 이 경보는 스테이지의 경로에 대한 API Gateway 요청에서 통합 지연 시간이 긴 경우를 감지할 수 있습니다.

통계: p90

권장 임곗값: 2000.0

임곗값 정당화: 제안된 임곗값이 모든 API 워크로드에 적용되는 것은 아닙니다. 그러나 임곗값의 시작점으로 사용할 수 있습니다. 그런 다음 워크로드와 API의 허용 가능한 지연 시간, 성능, SLA 요구 사항에 따라 다른 임곗값을 선택할 수 있습니다. 일반적으로 API의 지연 시간이 길어도 괜찮다면 임곗값을 더 높게 설정하여 경보의 민감도를 낮추세요. 하지만 API가 거의 실시간에 가까운 응답을 제공할 것으로 예상되는 경우 임곗값을 더 낮게 설정하세요. 또한 과거 데이터를 분석하여 애플리케이션 워크로드의 예상 기준 지연 시간을 파악한 다음 그에 따라 임곗값을 적절히 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latency

측정 기준: ApiName, 스테이지

경보 설명: 이 경보는 스테이지에서 높은 지연 시간을 감지합니다. `IntegrationLatency` 지표 값을 찾아 API 백엔드 지연 시간을 확인하세요. 두 지표가 대부분 일치하면 API 백엔드가 지연 시간 증가의 원인이므로 문제가 있는지 조사해야 합니다. CloudWatch Logs를 활성화하고 긴 지연 시간을 유발할 수 있는 오류가 있는지 확인하는 것도 고려해 보세요. 또한 상세한 CloudWatch 지표를 활성화하여 리소스 및 방법별로 이 지표를 확인하고 지연 시간의 원인을 좁히는 것도 고려해 보세요. 해당하는 경우 Lambda를 사용한 문제 해결 또는 엣지 최적화 API 엔드포인트 문제 해결 가이드를 참조하세요.

인텐트: 이 경보는 스테이지의 API Gateway 요청에서 지연 시간이 긴 경우를 감지할 수 있습니다. 자세한 CloudWatch 지표를 활성화하고 각 방법 및 리소스에 대한 지연 시간 성능 요구 사항이 다른 경우, 각 리소스 및 방법의 지연 시간을 보다 세밀하게 모니터링하려면 대체 경보를 생성하는 것이 좋습니다.

통계: p90

권장 임곗값: 2500.0

임곗값 정당화: 제안된 임곗값이 모든 API 워크로드에 적용되는 것은 아닙니다. 그러나 임곗값의 시작점으로 사용할 수 있습니다. 그런 다음 워크로드와 API의 허용 가능한 지연 시간, 성능, SLA 요구 사항에 따라 다른 임곗값을 선택할 수 있습니다. 일반적으로 API의 지연 시간이 길어도 괜찮다면 임곗값을 더 높게 설정하여 경보의 민감도를 낮추세요. 하지만 API가 거의 실시간에 가까운 응답을 제공할 것으로 예상되는 경우 임곗값을 더 낮게 설정하세요. 또한 과거 데이터를 분석하여 애플리케이션 워크로드의 예상 기준 지연 시간을 파악한 다음 그에 따라 임곗값을 적절히 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latency

측정 기준: ApiName, 스테이지, 리소스, 방법

경보 설명: 이 경보는 스테이지에서 리소스와 방법에 대해 높은 지연 시간을 감지합니다. `IntegrationLatency` 지표 값을 찾아 API 백엔드 지연 시간을 확인하세요. 두 지표가 대부분 일치하면 API 백엔드가 지연 시간 증가의 원인이므로 성능 문제가 있는지 조사해야 합니다. CloudWatch Logs를 활성화하고 긴 지연 시간을 유발할 수 있는 오류가 있는지 확인하는 것도 고려해 보세요. 해당하는 경우 Lambda를 사용한 문제 해결 또는 엣지 최적화 API 엔드포인트 문제 해결 가이드를 참조하세요.

인텐트: 이 경보는 스테이지의 API Gateway 요청에서 리소스와 방법에 대해 지연 시간이 긴 경우를 감지할 수 있습니다.

통계: p90

권장 임곗값: 2500.0

임곗값 정당화: 제안된 임곗값이 모든 API 워크로드에 적용되는 것은 아닙니다. 그러나 임곗값의 시작점으로 사용할 수 있습니다. 그런 다음 워크로드와 API의 허용 가능한 지연 시간, 성능, SLA 요구 사항에 따라 다른 임곗값을 선택할 수 있습니다. 일반적으로 API의 지연 시간이 길어도 괜찮다면 임곗값을 더 높게 설정하여 경보의 민감도를 낮추세요. 하지만 API가 거의 실시간에 가까운 응답을 제공할 것으로 예상되는 경우 임곗값을 더 낮게 설정하세요. 또한 과거 데이터를 분석하여 애플리케

이션 워크로드의 예상 기준 지연 시간을 파악한 다음 그에 따라 임곗값을 적절히 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latency

측정 기준: ApiId, 스테이지

경보 설명: 이 경보는 스테이지에서 높은 지연 시간을 감지합니다. `IntegrationLatency` 지표 값을 찾아 API 백엔드 지연 시간을 확인하세요. 두 지표가 대부분 일치하면 API 백엔드가 지연 시간 증가의 원인이므로 성능 문제가 있는지 조사해야 합니다. CloudWatch Logs를 활성화하고 긴 지연 시간을 유발할 수 있는 오류가 있는지 확인하는 것도 고려해 보세요. 또한 상세한 CloudWatch 지표를 활성화하여 경로별로 이 지표를 확인하고 지연 시간의 원인을 좁히는 것도 고려해 보세요. 해당하는 경우 Lambda 통합을 사용한 문제 해결 가이드도 참조할 수 있습니다.

인텐트: 이 경보는 스테이지의 API Gateway 요청에서 지연 시간이 긴 경우를 감지할 수 있습니다. 자세한 CloudWatch 지표를 활성화하고 경로별로 지연 시간 성능 요구 사항이 다른 경우, 각 경로의 지연 시간을 보다 세밀하게 모니터링하려면 대체 경보를 생성하는 것이 좋습니다.

통계: p90

권장 임곗값: 2500.0

임곗값 정당화: 제안된 임곗값이 모든 API 워크로드에 적용되는 것은 아닙니다. 하지만 이 값을 임곗값의 시작점으로 사용할 수 있습니다. 그런 다음 워크로드와 API의 허용 가능한 지연 시간, 성능, SLA 요구 사항에 따라 다른 임곗값을 선택할 수 있습니다. 일반적으로 API의 지연 시간이 길어도 괜찮다면 임곗값을 더 높게 설정하여 경보의 민감도를 낮출 수 있습니다. 하지만 API가 거의 실시간에 가까운 응답을 제공할 것으로 예상되는 경우 임곗값을 더 낮게 설정하세요. 또한 과거 데이터를 분석하여 애플리케이션 워크로드의 예상 기준 지연 시간을 파악한 다음 그에 따라 임곗값을 적절히 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latency

측정 기준: ApiId, 스테이지, 리소스, 방법

경보 설명: 이 경보는 스테이지에서 경로에 대해 높은 지연 시간을 감지합니다. `IntegrationLatency` 지표 값을 찾아 API 백엔드 지연 시간을 확인하세요. 두 지표가 대부분 일치하면 API 백엔드가 지연 시간 증가의 원인이므로 성능 문제가 있는지 조사해야 합니다. CloudWatch Logs를 활성화하고 긴 지연 시간을 유발할 수 있는 오류가 있는지 확인하는 것도 고려해 보세요. 해당하는 경우 Lambda 통합을 사용한 문제 해결 가이드도 참조할 수 있습니다.

인텐트: 이 경보는 스테이지의 경로에 대한 API Gateway 요청에서 지연 시간이 긴 경우를 감지할 수 있습니다.

통계: p90

권장 임곗값: 2500.0

임곗값 정당화: 제안된 임곗값이 모든 API 워크로드에 적용되는 것은 아닙니다. 하지만 이 값을 임곗값의 시작점으로 사용할 수 있습니다. 그런 다음 워크로드와 API의 허용 가능한 지연 시간, 성능, SLA 요구 사항에 따라 다른 임곗값을 선택할 수 있습니다. 일반적으로 API의 지연 시간이 길어도 괜찮다면 임곗값을 더 높게 설정하여 경보의 민감도를 낮추세요. 하지만 API가 거의 실시간에 가까운 응답을 제공할 것으로 예상되는 경우 임곗값을 더 낮게 설정하세요. 또한 과거 데이터를 분석하여 애플리케이션 워크로드의 예상 기준 지연 시간을 파악한 다음 그에 따라 임곗값을 적절히 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

4xx

측정 기준: ApiId, 스테이지

경보 설명: 이 경보는 높은 비율의 클라이언트 측 오류를 감지합니다. 이는 권한 부여 또는 클라이언트 요청 매개변수에 문제가 있음을 의미할 수 있습니다. 경로가 제거되었거나 클라이언트가 API에

없는 경로를 요청하고 있음을 의미할 수도 있습니다. CloudWatch Logs를 활성화하고 4xx 오류의 원인이 될 수 있는 오류가 있는지 확인해 보세요. 또한 상세한 CloudWatch 지표를 활성화하여 경로별로 이 지표를 확인하고 오류의 원인을 좁히는 것도 고려해 보세요. 구성된 제한 한도를 초과하여 오류가 발생할 수도 있습니다. 응답 및 로그에서 예상치 못한 오류가 429개라는 높은 비율로 보고되는 경우 이 가이드에 따라 문제를 해결하세요.

인텐트: 이 경보는 API Gateway 요청에 대해 높은 비율의 클라이언트 측 오류를 탐지할 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 제안된 임곗값은 전체 요청의 5% 이상에서 4xx 오류가 발생하는 경우를 감지합니다. 하지만 요청의 트래픽과 허용 가능한 오류 발생률에 맞게 임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다. 자주 발생하는 4xx 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

5xx

측정 기준: ApiId, 스테이지

경보 설명: 이 경보는 높은 비율의 서버 측 오류 감지에 도움이 됩니다. 이는 API 백엔드, 네트워크 또는 API 게이트웨이와 백엔드 API 간의 통합에 문제가 있음을 나타낼 수 있습니다. 이 설명서는 5xx 오류의 원인을 해결하는 데 도움이 될 수 있습니다.

인텐트: 이 경보는 API Gateway 요청에 대해 높은 비율의 서버 측 오류를 탐지할 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 제안된 임곗값은 전체 요청의 5% 이상에서 5xx 오류가 발생하는 경우를 감지합니다. 하지만 요청의 트래픽과 허용 가능한 오류 발생률에 맞게 임곗값을 조정할 수 있습니다. 또한 과

거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다. 자주 발생하는 5xx 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: GREATER_THAN_THRESHOLD

MessageCount

측정 기준: ApiId, 스테이지

경보 설명: 이 경보는 WebSocket API 단계에서 낮은 트래픽 볼륨을 감지하는 데 도움이 됩니다. 이는 클라이언트가 API를 호출할 때 잘못된 엔드포인트 사용 문제 또는 백엔드에서 클라이언트에 메시지를 보내는 데 문제가 있음을 나타낼 수 있습니다. 또한 API의 구성 또는 권한 문제로 인해 클라이언트가 연결할 수 없게 되었음을 나타내는 지표일 수도 있습니다.

인텐트: 이 경보는 WebSocket API 단계에서 예기치 않게 낮은 트래픽 볼륨을 감지할 수 있습니다. API가 정상 조건에서 예측 가능하고 일관된 수의 메시지를 수신 및 전송하는 경우 이 경보를 생성하는 것이 좋습니다. 자세한 CloudWatch 지표를 활성화하고 경로별 정상 트래픽 볼륨을 예측할 수 있는 경우, 각 경로에 대한 트래픽 볼륨 감소를 보다 세밀하게 모니터링할 수 있도록 대체 경보를 생성하는 것이 좋습니다. 일정하고 일관된 트래픽이 예상되지 않는 API에는 이 경보를 권장하지 않습니다.

통계: SampleCount

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 과거 데이터 분석을 기반으로 임곗값을 설정하여 API의 예상 기준 메시지 수를 결정합니다. 임곗값을 매우 높게 설정하면 정상적이고 트래픽이 적을 것으로 예상되는 기간에 경보가 너무 민감해질 수 있습니다. 반대로 값을 매우 낮게 설정하면 경보가 이례적으로 작은 트래픽 볼륨 감소를 놓칠 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: LESS_THAN_THRESHOLD

MessageCount

측정 기준: ApiId, 스테이지, 경로

경보 설명: 이 경보는 해당 스테이지에서 WebSocket API 경로에 대해 낮은 트래픽 볼륨을 감지하는 데 도움이 됩니다. 이는 클라이언트가 API를 호출할 때 잘못된 엔드포인트 사용 문제 또는 백엔드에서 클라이언트에 메시지를 보내는 데 문제가 있음을 나타낼 수 있습니다. 또한 API의 구성 또는 권한 문제로 인해 클라이언트가 연결할 수 없게 되었음을 나타내는 지표일 수도 있습니다.

인텐트: 이 경보는 해당 스테이지에서 WebSocket API 경로에 대해 예기치 않게 낮은 트래픽 볼륨을 감지할 수 있습니다. API가 정상 조건에서 예측 가능하고 일관된 수의 메시지를 수신 및 전송하는 경우 이 경보를 생성하는 것이 좋습니다. 일정하고 일관된 트래픽이 예상되지 않는 API에는 이 경보를 권장하지 않습니다.

통계: SampleCount

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 과거 데이터 분석을 기반으로 임곗값을 설정하여 API의 예상 기준 메시지 수를 결정합니다. 임곗값을 매우 높게 설정하면 정상적이고 트래픽이 적을 것으로 예상되는 기간에 경보가 너무 민감해질 수 있습니다. 반대로 값을 매우 낮게 설정하면 경보가 이례적으로 작은 트래픽 볼륨 감소를 놓칠 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: LESS_THAN_THRESHOLD

ClientError

측정 기준: ApiId, 스테이지

경보 설명: 이 경보는 높은 비율의 클라이언트 오류를 감지합니다. 이는 권한 부여 또는 메시지 매개변수에 문제가 있음을 의미할 수 있습니다. 경로가 제거되었거나 클라이언트가 API에 없는 경로를 요청하고 있음을 의미할 수도 있습니다. CloudWatch Logs를 활성화하고 4xx 오류의 원인이 될 수 있는 오류가 있는지 확인해 보세요. 또한 상세한 CloudWatch 지표를 활성화하여 경로별로 이 지표를 확인하고 오류의 원인을 좁히는 것도 고려해 보세요. 구성된 제한 한도를 초과하여 오류가 발생

할 수도 있습니다. 응답 및 로그에서 예상치 못한 오류가 429개라는 높은 비율로 보고되는 경우 이 가이드에 따라 문제를 해결하세요.

인텐트: 이 경보는 WebSocket API Gateway 메시지에 대해 높은 비율의 클라이언트 오류를 탐지할 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 제안된 임곗값은 전체 요청의 5% 이상에서 4xx 오류가 발생하는 경우를 감지합니다. 요청의 트래픽과 허용 가능한 오류 발생률에 맞게 임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다. 자주 발생하는 4xx 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

ExecutionError

측정 기준: ApiId, 스테이지

경보 설명: 이 경보는 높은 비율의 실행 오류 감지에 도움이 됩니다. 이는 통합으로 인한 5xx 오류, 권한 문제 또는 통합의 성공적인 호출을 방해하는 기타 요인(예: 통합이 제한 또는 삭제되는 경우)이 원인일 수 있습니다. API의 CloudWatch Logs를 활성화하고 로그에서 오류의 유형과 원인을 확인해 보세요. 또한 상세한 CloudWatch 지표를 활성화하여 경로별로 이 지표를 확인하고 오류의 원인을 좁히는 것도 고려해 보세요. 이 설명서는 모든 연결 오류의 원인을 해결하는 데 도움이 될 수 있습니다.

인텐트: 이 경보는 WebSocket API Gateway 메시지에 대해 높은 비율의 실행 오류를 탐지할 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 제안된 임곗값은 전체 요청의 5% 이상에서 실행 오류가 발생하는 경우를 감지합니다. 요청의 트래픽과 허용 가능한 오류 발생률에 맞게 임곗값을 조정할 수 있습니다. 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다. 자주 발생하는 실행 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: GREATER_THAN_THRESHOLD

Amazon EC2 Auto Scaling

GroupInServiceCapacity

측정 기준: AutoScalingGroupName

경보 설명: 이 경보는 그룹의 용량이 워크로드에 필요한 용량보다 낮을 경우 이를 감지하는 데 도움이 됩니다. 문제를 해결하려면 조정 활동에서 시작 실패가 있는지 확인하고 원하는 용량 구성이 올바른지 확인하세요.

인텐트: 이 경보는 Auto Scaling 그룹에서 시작 실패 또는 시작 일시 중단으로 인한 낮은 가용성을 감지할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 임곗값은 워크로드 실행에 필요한 최소 용량이어야 합니다. 대부분의 경우 이 값을 GroupDesiredCapacity 지표와 일치하도록 설정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: LESS_THAN_THRESHOLD

Amazon CloudFront

5xxErrorRate

측정 기준: DistributionId, Region=Global

경보 설명: 이 경보는 오리진 서버의 5xx 오류 응답 비율을 모니터링하여 CloudFront 서비스에 문제가 있는지 감지하는 데 도움이 됩니다. 서버 관련 문제를 이해하는 데 도움이 되는 오리진의 오류 응답 문제 해결을 참조하세요. 또한 추가 메트릭을 활성화하고 자세한 오류 메트릭을 확인하세요.

인텐트: 이 경보는 오리진 서버의 요청 처리 문제 또는 CloudFront와 오리진 서버 간의 통신 문제를 감지하는 데 사용됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 5xx 응답의 허용 오차에 따라 크게 달라집니다. 과거 데이터와 추세를 분석한 다음 그에 따라 임곗값을 설정할 수 있습니다. 5xx 오류는 일시적인 문제로 인해 발생할 수 있으므로 경보가 너무 민감하지 않도록 임곗값을 0보다 큰 값으로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

OriginLatency

측정 기준: DistributionId, Region=Global

경보 설명: 경보는 오리진 서버가 응답하는 데 시간이 너무 오래 걸리는지 모니터링하는 데 도움이 됩니다. 서버가 응답하는 데 시간이 너무 오래 걸리면 타임아웃이 발생할 수 있습니다. 지속적으로 높은 `OriginLatency` 값이 발생하는 경우 오리진 서버의 애플리케이션에서 지연된 응답을 찾아서 수정하기를 참조하세요.

인텐트: 이 경보는 오리진 서버가 응답하는 데 시간이 너무 오래 걸리는 문제를 감지하는 데 사용됩니다.

통계: p90

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 오리진 응답 제한 시간의 약 80% 값을 계산하고 그 결과를 임곗값으로 사용해야 합니다. 이 지표가 오리진 응답 제한 시간 값에 지속적으로 근접할 경우 504 오류가 발생할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

FunctionValidationErrors

측정 기준: DistributionId, FunctionName, Region=Global

경보 설명: 이 경보는 CloudFront Functions의 검증 오류를 모니터링하여 오류를 해결하기 위한 조치를 취할 수 있도록 도와줍니다. CloudWatch 함수 로그를 분석하고 함수 코드를 검토하여 문제의 근본 원인을 찾아 해결합니다. CloudFront Functions의 일반적인 구성 오류를 이해하려면 엣지 함수에 대한 제한을 참조하세요.

인텐트: 이 경보는 CloudFront Functions의 검증 오류를 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: 0보다 큰 값은 검증 오류를 나타냅니다. 검증 오류는 CloudFront Functions를 CloudFront로 넘겨줄 때 문제가 있다는 것을 의미하므로 임곗값을 0으로 설정하는 것이 좋습니다. 예를 들어 CloudFront는 요청을 처리하기 위해 HTTP 호스트 헤더가 필요합니다. 사용자가 CloudFront Functions 코드에서 호스트 헤더를 삭제하는 것을 막을 방법은 없습니다. 하지만 CloudFront가 응답을 받고 호스트 헤더가 누락된 경우 CloudFront에서 검증 오류가 발생합니다.

기간: 60

경보를 보낼 데이터 포인트: 2

평가 기간: 2

비교 연산자: GREATER_THAN_THRESHOLD

FunctionExecutionErrors

측정 기준: DistributionId, FunctionName, Region=Global

경보 설명: 이 경보는 CloudFront Functions의 실행 오류를 모니터링하여 오류를 해결하기 위한 조치를 취할 수 있도록 도와줍니다. CloudWatch 함수 로그를 분석하고 함수 코드를 검토하여 문제의 근본 원인을 찾아 해결합니다.

인텐트: 이 경보는 CloudFront Functions의 실행 오류를 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: 실행 오류는 런타임에 발생하는 코드에 문제가 있음을 나타내므로 임곗값을 0으로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

FunctionThrottles

측정 기준: DistributionId, FunctionName, Region=Global

경보 설명: 이 경보는 CloudFront Functions의 제한 여부를 모니터링하는 데 도움이 됩니다. 함수가 제한되면 실행 시간이 너무 오래 걸린다는 의미입니다. 함수 제한을 방지하려면 함수 코드를 최적화하는 것이 좋습니다.

인텐트: 이 경보는 문제에 대응하고 해결하여 원활한 고객 경험을 제공할 수 있도록 CloudFront Functions가 제한되는 시점을 감지합니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: 함수 제한 문제를 더 빨리 해결할 수 있도록 임곗값을 0으로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Amazon Cognito

SignUpThrottles

측정 기준: UserPool, UserPoolClient

경보 설명: 이 경보는 제한이 발생한 요청 수를 모니터링합니다. 사용자가 지속적으로 병목 현상을 겪는 경우 서비스 할당량 증가를 요청하여 한도를 늘려야 합니다. 할당량 증가를 요청하는 방법은 Amazon Cognito의 할당량 섹션을 참조하세요. 사전 조치를 취하려면 사용 할당량 추적을 고려해 보세요.

인텐트: 이 경보는 제한된 가입 요청의 발생을 모니터링하는 데 도움이 됩니다. 가입 경험의 저하를 완화하기 위한 조치를 언제 실행해야 하는지 알 수 있습니다. 지속적인 요청 제한은 사용자의 가입 경험에 부정적인 영향을 미칩니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 제대로 프로비저닝된 사용자 풀에서는 여러 데이터 포인트에 걸쳐 제한이 발생하지 않아야 합니다. 따라서 예상 워크로드의 일반적인 임곗값은 0이어야 합니다. 버스트가 빈번하고 불규칙한 워크로드의 경우 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한을 결정한 다음 그에 따라 임곗값을 조정할 수 있습니다. 제한이 발생한 요청은 다시 시도하여 애플리케이션에 미치는 영향을 최소화해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

SignInThrottles

측정 기준: UserPool, UserPoolClient

경보 설명: 이 경보는 제한이 발생한 사용자 인증 요청 수를 모니터링합니다. 사용자가 지속적으로 병목 현상을 겪는 경우 서비스 할당량 증가를 요청하여 한도를 늘려야 할 수 있습니다. 할당량 증가를 요청하는 방법은 Amazon Cognito의 할당량 섹션을 참조하세요. 사전 조치를 취하려면 사용 할당량 추적을 고려해 보세요.

인텐트: 이 경보는 제한된 가입 요청의 발생을 모니터링하는 데 도움이 됩니다. 가입 경험의 저하를 완화하기 위한 조치를 언제 실행해야 하는지 알 수 있습니다. 지속적인 요청 제한은 부정적인 사용자 인증 경험입니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 제대로 프로비저닝된 사용자 풀에서는 여러 데이터 포인트에 걸쳐 제한이 발생하지 않아야 합니다. 따라서 예상 워크로드의 일반적인 임곗값은 0이어야 합니다. 버스트가 빈번하고 불규칙한 워크로드의 경우 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한을 결정한 다음 그에 따라 임곗값을 조정할 수 있습니다. 제한이 발생한 요청은 다시 시도하여 애플리케이션에 미치는 영향을 최소화해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

TokenRefreshThrottles

측정 기준: UserPool, UserPoolClient

경보 설명: 요청의 트래픽에 적합하고 토큰 새로 고침 요청의 허용 가능한 제한과 일치하도록 임곗값을 설정할 수 있습니다. 제한은 너무 많은 요청으로부터 시스템을 보호하는 데 사용됩니다. 하지만 정상 트래픽에 대한 프로비저닝이 부족한지 모니터링하는 것도 중요합니다. 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한을 찾은 다음 경보 임곗값을 허용 가능한 제한 수준보다 높게 조정할 수 있습니다. 제한이 발생한 요청은 일시적이므로 애플리케이션/서비스에서 재시도해야 합니다. 따라서 임곗값이 매우 낮으면 경보가 민감해질 수 있습니다.

인텐트: 이 경보는 제한된 토큰 새로 고침 요청의 발생을 모니터링하는 데 도움이 됩니다. 이를 통해 잠재적 문제를 완화하고 원활한 사용자 경험과 인증 시스템의 상태 및 안정성을 보장하기 위한 조치를 언제 실행해야 하는지 알 수 있습니다. 지속적인 요청 제한은 부정적인 사용자 인증 경험입니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 요청의 트래픽과 토큰 새로 고침 요청에 대한 허용 가능한 제한에 적합하게 임곗값을 설정/조정할 수 있습니다. 제한은 너무 많은 요청으로부터 시스템을 보호하기 위한 것이지만, 정상 트래픽에 대한 프로비저닝이 부족한지 모니터링하고 이로 인해 제한이 발생하는지 확인하는 것이 중요합니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한을 확인하고, 임곗값을 일반적인 허용 제한 수준보다 높게 조정할 수 있습니다. 제한이 발생한 요청은 일시적이므로 애플리케이션/서비스에서 재시도해야 합니다. 따라서 임곗값이 매우 낮으면 경보가 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

FederationThrottles

측정 기준: UserPool, UserPoolClient, IdentityProvider

경보 설명: 이 경보는 제한이 발생한 ID 페더레이션 요청 수를 모니터링합니다. 제한이 지속적으로 발생하는 경우 서비스 할당량 증가를 요청하여 한도를 늘려야 할 수 있습니다. 할당량 증가를 요청하는 방법은 Amazon Cognito의 할당량 섹션을 참조하세요.

인텐트: 이 경보는 제한된 ID 페더레이션 요청의 발생을 모니터링하는 데 도움이 됩니다. 이를 통해 성능 병목 현상이나 잘못된 구성에 사전 대응하고 사용자에게 원활한 인증 경험을 제공할 수 있습니다. 지속적인 요청 제한은 부정적인 사용자 인증 경험입니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 요청의 트래픽에 적합하고 ID 페더레이션 요청의 허용 가능한 제한과 일치하도록 임곗값을 설정할 수 있습니다. 제한은 너무 많은 요청으로부터 시스템을 보호하는 데 사용됩니다.

하지만 정상 트래픽에 대한 프로비저닝이 부족한지 모니터링하는 것도 중요합니다. 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한을 찾은 다음 임곗값을 허용 가능한 제한 수준보다 높은 값으로 설정할 수 있습니다. 제한이 발생한 요청은 일시적이므로 애플리케이션/서비스에서 재시도해야 합니다. 따라서 임곗값이 매우 낮으면 경보가 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Amazon DynamoDB

AccountProvisionedReadCapacityUtilization

측정 기준: 없음

경보 설명: 이 경보는 계정의 읽기 용량이 프로비저닝된 한도에 도달하는지 여부를 감지합니다. 이 경우 읽기 용량 사용량에 대한 계정 할당량을 늘릴 수 있습니다. Service Quotas를 사용하여 읽기 용량 단위의 현재 할당량을 확인하고 증가를 요청할 수 있습니다.

인텐트: 이 경보는 계정의 읽기 용량 사용률이 프로비저닝된 읽기 용량 사용률에 근접하는지 여부를 감지할 수 있습니다. 사용률이 최대 한도에 도달하면 DynamoDB가 읽기 요청을 제한하기 시작합니다.

통계: Maximum

권장 임곗값: 80.0

임곗값 정당화: 임곗값을 80%로 설정하면 최대 용량에 도달하기 전에 계정 한도 상향 조정과 같은 조치를 실행하여 제한을 방지할 수 있습니다.

기간: 300

경보를 보낼 데이터 포인트: 2

평가 기간: 2

비교 연산자: GREATER_THAN_THRESHOLD

AccountProvisionedWriteCapacityUtilization

측정 기준: 없음

경보 설명: 이 경보는 계정의 쓰기 용량이 프로비저닝된 한도에 도달하는지 여부를 감지합니다. 이 경우 쓰기 용량 사용량에 대한 계정 할당량을 늘릴 수 있습니다. Service Quotas를 사용하여 쓰기 용량 단위의 현재 할당량을 확인하고 증가를 요청할 수 있습니다.

인텐트: 이 경보는 계정의 쓰기 용량 사용률이 프로비저닝된 쓰기 용량 사용률에 근접하는지 여부를 감지할 수 있습니다. 사용률이 최대 한도에 도달하면 DynamoDB가 쓰기 요청을 제한하기 시작합니다.

통계: Maximum

권장 임곗값: 80.0

임곗값 정당화: 임곗값을 80%로 설정하면 최대 용량에 도달하기 전에 계정 한도 상향 조정과 같은 조치를 실행하여 제한을 방지할 수 있습니다.

기간: 300

경보를 보낼 데이터 포인트: 2

평가 기간: 2

비교 연산자: GREATER_THAN_THRESHOLD

AgeOfOldestUnreplicatedRecord

측정 기준: TableName, DelegatedOperation

경보 설명: 이 경보는 Kinesis 데이터 스트림으로의 복제에서 지연을 감지합니다. 정상 작동 시 `AgeOfOldestUnreplicatedRecord`는 밀리초 단위여야 합니다. 이 수는 실패한 복제 시도가 고객이 제어하는 구성 선택으로 인해 발생한 경우 시도 횟수를 기준으로 증가합니다. 복제 시도 실패로 이어질 수 있는 고객이 제어하는 구성의 예로는 Kinesis 데이터 스트림 용량이 과소 프로비저닝되어 과도한 제한이 발생하는 경우 또는 Kinesis 데이터 스트림의 액세스 정책을 수동으로 업데이트하여 DynamoDB가 데이터 스트림에 데이터를 추가하지 못하는 경우가 있습니다. 이 지표를 가능한 한 낮게 유지하려면 적절한 Kinesis 데이터 스트림 용량을 프로비저닝하고 DynamoDB의 권한이 변경되지 않도록 해야 합니다.

인텐트: 이 경보는 실패한 복제 시도와 그에 따른 Kinesis 데이터 스트림으로의 복제 지연을 모니터링할 수 있습니다.

통계: Maximum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 밀리초 단위로 측정된 원하는 복제 지연에 따라 임곗값을 설정합니다. 이 값은 워크로드의 요구 사항 및 예상 성능에 따라 달라집니다.

기간: 300

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: GREATER_THAN_THRESHOLD

FailedToReplicateRecordCount

측정 기준: TableName, DelegatedOperation

경보 설명: 이 경보는 DynamoDB가 Kinesis 데이터 스트림으로 복제하지 못한 레코드 수를 감지합니다. 34KB보다 큰 특정 항목은 Kinesis Data Streams의 1MB 항목 크기 제한보다 큰 데이터 레코드를 변경하기 위해 크기가 확장될 수 있습니다. 이 크기 확장은 34KB보다 큰 이러한 항목에 많은 수의 부울 또는 빈 속성 값을 포함할 때 발생합니다. 부울 및 빈 속성 값은 DynamoDB에 1바이트로 저장되지만, Kinesis Data Streams 복제를 위한 표준 JSON을 사용하여 직렬화되면 최대 5바이트까지 확장합니다. DynamoDB는 이러한 변경 레코드를 Kinesis 데이터 스트림에 복제할 수 없습니다. DynamoDB는 이러한 변경 데이터 레코드를 건너뛰고 자동으로 후속 레코드의 복제를 계속합니다.

인텐트: 이 경보는 Kinesis 데이터 스트림의 항목 크기 제한 때문에 DynamoDB가 Kinesis 데이터 스트림에 복제하지 못한 레코드 수를 모니터링할 수 있습니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: DynamoDB가 복제에 실패한 모든 레코드를 감지하려면 임곗값을 0으로 설정합니다.

기간: 60

경보를 보낼 데이터 포인트: 1

평가 기간: 1

비교 연산자: GREATER_THAN_THRESHOLD

ReadThrottleEvents

측정 기준: TableName

경보 설명: 이 경보는 DynamoDB 테이블에 대해 제한이 발생하는 읽기 요청 수가 많은지 여부를 감지합니다. 문제를 해결하려면 Amazon DynamoDB의 제한 문제 해결을 참조하세요.

인텐트: 이 경보는 DynamoDB 테이블에 대한 읽기 요청의 지속적인 제한을 감지할 수 있습니다. 읽기 요청의 지속적인 제한은 워크로드 읽기 작업에 부정적인 영향을 미치고 시스템의 전반적인 효율성을 감소시킬 수 있습니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 허용 가능한 제한 수준을 고려하여 DynamoDB 테이블의 예상 읽기 트래픽에 따라 임곗값을 설정합니다. 프로비저닝이 부족하고 일관된 제한이 발생하지 않는지 모니터링하는 것이 중요합니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한 수준을 찾은 다음 임곗값을 일반적인 제한 수준보다 높게 조정할 수 있습니다. 제한이 발생한 요청은 일시적이므로 애플리케이션 또는 서비스에서 재시도해야 합니다. 따라서 임곗값이 매우 낮으면 경보가 너무 민감해져 원치 않는 상태 전환이 발생할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

ReadThrottleEvents

측정 항목: TableName, GlobalSecondaryIndexName

경보 설명: 이 경보는 DynamoDB 테이블의 글로벌 보조 인덱스에 대해 제한이 발생하는 읽기 요청 수가 많은지 여부를 감지합니다. 문제를 해결하려면 Amazon DynamoDB의 제한 문제 해결을 참조하세요.

인텐트: 이 경보는 DynamoDB 테이블의 글로벌 보조 인덱스에 대한 읽기 요청의 지속적인 제한을 감지할 수 있습니다. 읽기 요청의 지속적인 제한은 워크로드 읽기 작업에 부정적인 영향을 미치고 시스템의 전반적인 효율성을 감소시킬 수 있습니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 허용 가능한 제한 수준을 고려하여 DynamoDB 테이블의 예상 읽기 트래픽에 따라 임곗값을 설정합니다. 프로비저닝이 부족하고 일관된 제한이 발생하지 않는지 모니터링하는 것이 중요합니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한 수준을 찾은 다음 임곗값을 허용 가능한 일반적인 제한 수준보다 높게 조정할 수 있습니다. 제한이 발생한 요청은 일시적이므로 애플리케이션 또는 서비스에서 재시도해야 합니다. 따라서 임곗값이 매우 낮으면 경보가 너무 민감해져 원치 않는 상태 전환이 발생할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

ReplicationLatency

측정 기준: TableName, ReceivingRegion

경보 설명: 경보는 글로벌 테이블의 특정 리전에 있는 복제본이 소스 리전보다 뒤처지는지 감지합니다. AWS 리전의 성능이 저하되고 해당 리전에 복제 테이블이 있는 경우에 지연 시간이 증가할 수 있습니다. 이 경우 애플리케이션의 읽기 및 쓰기 작업을 다른 AWS 리전으로 일시적으로 리디렉션할 수 있습니다. 2017.11.29(레거시)의 글로벌 테이블을 사용하는 경우 각 복제 테이블의 WCU(쓰기 용량 단위)가 동일한지 확인해야 합니다. 용량 관리를 위한 모범 사례 및 요구 사항의 권장 사항을 따를 수도 있습니다.

인텐트: 경보는 특정 리전의 복제 테이블이 다른 리전의 변경 내용을 복제하는 데 뒤쳐지는지 여부를 감지할 수 있습니다. 이로 인해 복제본이 다른 복제본과 다를 수 있습니다. 각 AWS 리전의 복제 지연 시간을 알고 복제 지연 시간이 계속 증가할 경우 알림을 보내는 것이 좋습니다. 테이블 복제는 글로벌 테이블에만 적용됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 사용 사례에 따라 크게 달라집니다. 일반적으로 3분이 넘는 복제 지연 시간은 조사 원인입니다. 복제 지연의 심각도와 요구 사항을 검토하고 과거 추세를 분석한 다음 그에 따라 임곗값을 선택합니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

SuccessfulRequestLatency

측정 기준: TableName, Operation

경보 설명: 이 경보는 DynamoDB 테이블 작업(경보에서 `Operation`의 차원 값으로 표시됨)에서 높은 지연 시간을 감지합니다. Amazon DynamoDB의 지연 시간 문제를 해결하려면 문제 해결 문서를 참조하세요.

인텐트: 이 경보는 DynamoDB 테이블 작업의 높은 지연 시간을 감지할 수 있습니다. 작업 지연 시간이 길어지면 시스템의 전체 효율성에 부정적인 영향을 미칠 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: DynamoDB는 GetItem, PutItem 등과 같은 싱글톤 작업에 대해 평균적으로 10밀리초 미만의 지연 시간을 지원합니다. 하지만 워크로드와 관련된 작업 유형 및 테이블의 지연 시간에 대한 허용 가능한 허용 오차를 기반으로 임곗값을 설정할 수 있습니다. 이 지표의 과거 데이터를 분석하여 테이블 작업에 대한 일반적인 지연 시간을 찾은 다음 임곗값을 작업의 심각한 지연을 나타내는 숫자로 설정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_THRESHOLD

SystemErrors

측정 기준: TableName

경보 설명: 이 경보는 DynamoDB 테이블 요청에서 지속적으로 많은 시스템 오류를 감지합니다. 5xx 오류가 계속 발생하면 AWS서비스 상태 대시보드를 열어 서비스와 관련된 운영 문제를 확인하

세요. 이 경보를 사용하면 DynamoDB에서 장기간 내부 서비스 문제가 발생하는 경우 알림을 받을 수 있으며, 이를 통해 클라이언트 애플리케이션이 직면하고 있는 문제와의 상관 관계를 파악할 수 있습니다. 자세한 내용은 DynamoDB 오류 처리를 참조하세요.

인텐트: 이 경보는 DynamoDB 테이블 요청에 대한 지속적인 시스템 오류를 감지할 수 있습니다. 시스템 오류는 DynamoDB의 내부 서비스 오류를 나타내며 클라이언트가 겪고 있는 문제와의 상관 관계를 파악하는 데 도움이 됩니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 허용 가능한 시스템 오류 수준을 고려하여 예상 트래픽에 따라 임곗값을 설정합니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 수를 찾은 다음 그에 따라 임곗값을 조정할 수 있습니다. 시스템 오류는 일시적이므로 애플리케이션/서비스에서 재시도해야 합니다. 따라서 임곗값이 매우 낮으면 경보가 너무 민감해져 원치 않는 상태 전환이 발생할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

ThrottledPutRecordCount

측정 기준: TableName, DelegatedOperation

경보 설명: 이 경보는 변경 데이터 캡처를 Kinesis로 복제하는 동안 Kinesis 데이터 스트림에 의해 제한되는 레코드를 감지합니다. 이러한 제한은 Kinesis 데이터 스트림 용량이 충분하지 않기 때문에 발생합니다. 제한이 지나치게 많고 자주 발생하는 경우 관찰된 테이블의 쓰기 처리량에 비례하여 Kinesis 스트림 샤드 수를 늘려야 할 수 있습니다. Kinesis 데이터 스트림의 크기 결정에 대한 자세한 내용은 Kinesis 데이터 스트림의 초기 크기 결정을 참조하세요.

인텐트: 이 경보는 Kinesis 데이터 스트림 용량이 부족하여 Kinesis 데이터 스트림에 의해 제한된 레코드 수를 모니터링할 수 있습니다.

통계: Maximum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 예외적으로 사용량이 최고조에 달할 때 약간의 제한이 발생할 수 있지만, 복제 지연 시간이 길어지지 않도록 제한된 레코드 수를 가능한 한 낮게 유지해야 합니다(DynamoDB가 Kinesis 데이터 스트림으로 제한된 레코드의 전송을 다시 시도). 정기적으로 과도한 제한을 포착하는 데 도움이 될 수 있는 숫자로 임곗값을 설정합니다. 또한 이 지표의 과거 데이터를 분석하여 애플리케이션 워크로드에 적합한 제한 속도를 찾을 수 있습니다. 사용 사례에 따라 애플리케이션에서 허용할 수 있는 값으로 임곗값을 조정합니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_THRESHOLD

UserErrors

측정 기준: 없음

경보 설명: 이 경보는 DynamoDB 테이블 요청에서 지속적으로 많은 사용자 오류를 감지합니다. 문제가 지속되는 동안 클라이언트 애플리케이션 로그를 확인하여 요청이 잘못된 이유를 확인할 수 있습니다. HTTP 상태 코드 400을 확인하여 발생하는 오류 유형을 확인하고 그에 따라 조치를 취할 수 있습니다. 유효한 요청을 생성하려면 애플리케이션 로직을 수정해야 할 수 있습니다.

인텐트: 이 경보는 DynamoDB 테이블 요청에 대한 지속적인 사용자 오류를 감지할 수 있습니다. 요청된 작업에 대한 사용자 오류는 클라이언트가 잘못된 요청을 생성하고 실패했음을 의미합니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 클라이언트 측 오류를 감지하려면 임곗값을 0으로 설정합니다. 또는 매우 적은 수의 오류로 인해 경보가 트리거되지 않게 하려면 더 높은 값으로 설정할 수 있습니다. 요청 시 사용 사례와 트래픽에 따라 결정합니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_THRESHOLD

WriteThrottleEvents

측정 기준: TableName

경보 설명: 이 경보는 DynamoDB 테이블에 대해 제한이 발생하는 쓰기 요청 수가 많은지 여부를 감지합니다. 문제를 해결하려면 Amazon DynamoDB의 제한 문제 해결을 참조하세요.

인텐트: 이 경보는 DynamoDB 테이블에 대한 쓰기 요청의 지속적인 제한을 감지할 수 있습니다. 쓰기 요청의 지속적인 제한은 워크로드 쓰기 작업에 부정적인 영향을 미치고 시스템의 전반적인 효율성을 감소시킬 수 있습니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 허용 가능한 제한 수준을 고려하여 DynamoDB 테이블의 예상 쓰기 트래픽에 따라 임곗값을 설정합니다. 프로비저닝이 부족하고 일관된 제한이 발생하지 않는지 모니터링하는 것이 중요합니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한 수준을 찾은 다음 임곗값을 허용 가능한 일반적인 제한 수준보다 높게 조정할 수 있습니다. 제한이 발생한 요청은 일시적이므로 애플리케이션/서비스에서 재시도해야 합니다. 따라서 임곗값이 매우 낮으면 경보가 너무 민감해져 원치 않는 상태 전환이 발생할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

WriteThrottleEvents

측정 항목: TableName, GlobalSecondaryIndexName

경보 설명: 이 경보는 DynamoDB 테이블의 글로벌 보조 인덱스에 대해 제한이 발생하는 쓰기 요청 수가 많은지 여부를 감지합니다. 문제를 해결하려면 Amazon DynamoDB의 제한 문제 해결을 참조하세요.

인텐트: 이 경보는 DynamoDB 테이블의 글로벌 보조 인덱스에 대한 쓰기 요청의 지속적인 제한을 감지할 수 있습니다. 쓰기 요청의 지속적인 제한은 워크로드 쓰기 작업에 부정적인 영향을 미치고 시스템의 전반적인 효율성을 감소시킬 수 있습니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 허용 가능한 제한 수준을 고려하여 DynamoDB 테이블의 예상 쓰기 트래픽에 따라 임곗값을 설정합니다. 프로비저닝이 부족하고 일관된 제한이 발생하지 않는지 모니터링하는 것이 중요합니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 제한 수준을 찾은 다음 임곗값을 허용 가능한 일반적인 제한 수준보다 높게 조정할 수 있습니다. 제한이 발생한 요청은 일시적이므로 애플리케이션/서비스에서 재시도해야 합니다. 따라서 임곗값이 매우 낮으면 경보가 너무 민감해져 원치 않는 상태 전환이 발생할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Amazon EBS

VolumeStalledIOCheck

차원: VolumeId, InstanceId

경보 설명: 이 경보는 Amazon EBS 볼륨의 IO 성능을 모니터링하는 데 도움이 됩니다. 이 검사는 Amazon EBS 볼륨의 기반이 되는 스토리지 하위 시스템의 하드웨어 또는 소프트웨어 문제, Amazon EC2 인스턴스의 Amazon EBS 볼륨 연결 가능성에 영향을 미치는 물리적 호스트의 하드웨어 문제 등 Amazon EBS 인프라의 근본적인 문제를 탐지하고, 인스턴스와 Amazon EBS 볼륨 간의 연결 문제를 감지할 수 있습니다. 멈춘 IO 검사가 실패하면 AWS에서 문제를 해결할 때까지 기다리거나, 영향을 받는 볼륨을 교체하거나 볼륨이 연결된 인스턴스를 중지하고 다시 시작하는 등의 조치를 취할 수 있습니다. 대부분의 경우 이 지표가 실패하면 Amazon EBS는 몇 분 내에 볼륨을 자동으로 진단하고 복구합니다.

의도: 이 경보는 Amazon EBS 볼륨의 상태를 감지하여 해당 볼륨이 손상되어 I/O 작업을 완료할 수 없게 된 시기를 확인할 수 있습니다.

통계: Maximum

권장 임곗값: 1.0

임곗값 정당화: 상태 확인이 실패하면 이 지표의 값은 1입니다. 임곗값은 상태 확인이 실패할 때마다 경보가 ALARM 상태가 되도록 설정됩니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Amazon EC2

CPUUtilization

측정 기준: InstanceId

경보 설명: 이 경보는 EC2 인스턴스의 CPU 사용률을 모니터링하는 데 도움이 됩니다. 애플리케이션에 따라 지속적으로 높은 사용률 수준이 정상일 수 있습니다. 그러나 성능이 저하되고 애플리케이션이 디스크 I/O, 메모리 또는 네트워크 리소스의 제약을 받지 않는 경우 CPU 한도가 초과되면 리소스 병목 현상이나 애플리케이션 성능 문제가 발생할 수 있습니다. CPU 사용률이 높으면 CPU 사용량이 더 많은 인스턴스로 업그레이드해야 할 수도 있습니다. 세부 모니터링을 활성화한 경우 기간을 300초가 아닌 60초로 변경할 수 있습니다. 자세한 내용은 인스턴스에 대한 세부 모니터링 활성화 또는 비활성화를 참조하세요.

인텐트: 이 경보는 높은 CPU 사용률을 감지하는 데 사용됩니다.

통계: Average

권장 임곗값: 80.0

임곗값 정당화: 일반적으로 CPU 사용률 임곗값을 70~80%로 설정할 수 있습니다. 하지만 허용 가능한 성능 수준과 워크로드 특성에 따라 이 값을 조정할 수 있습니다. 일부 시스템의 경우 지속적으로 높은 CPU 사용률이 정상이고 문제로 표시되지 않을 수 있지만, 다른 시스템에서는 문제가 발생할 수 있습니다. 과거의 CPU 사용률 데이터를 분석하여 사용량을 식별하고 시스템에 적합한 CPU 사용률을 찾은 다음 그에 따라 임곗값을 설정합니다.

기간: 300

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: GREATER_THAN_THRESHOLD

StatusCheckFailed

측정 기준: InstanceId

경보 설명: 이 경보는 시스템 상태 확인 및 인스턴스 상태 확인 모두를 모니터링하는 데 도움이 됩니다. 둘 중 한나의 상태 확인이 실패하는 경우 이 경보는 ALARM 상태여야 합니다.

인텐트: 이 경보는 시스템 상태 확인 실패와 인스턴스 상태 확인 실패를 비롯한 인스턴스의 근본적인 문제를 감지하는 데 사용됩니다.

통계: Maximum

권장 임곗값: 1.0

임곗값 정당화: 상태 확인이 실패하면 이 지표의 값은 1입니다. 임곗값은 상태 확인이 실패할 때마다 경보가 ALARM 상태가 되도록 설정됩니다.

기간: 300

경보를 보낼 데이터 포인트: 2

평가 기간: 2

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

StatusCheckFailed_AttachedEBS

측정 기준: InstanceId

경보 설명: 이 경보는 인스턴스에 연결된 Amazon EBS 볼륨이 연결 가능하고 I/O 작업을 완료할 수 있는지 모니터링할 수 있습니다. 이 상태 확인은 다음과 같은 컴퓨팅 또는 Amazon EBS 인프라의 기본 문제를 감지합니다.

- Amazon EBS 볼륨의 기반이 되는 스토리지 하위 시스템의 하드웨어 또는 소프트웨어 문제
- Amazon EBS 볼륨의 연결성에 영향을 주는 물리적 호스트의 하드웨어 문제
- 인스턴스와 Amazon EBS 볼륨 간의 연결 문제

연결된 EBS 상태 확인에 실패하면 Amazon이 문제를 해결할 때까지 기다리거나 영향을 받는 볼륨을 교체 또는 인스턴스 중지 후 재시작 등의 조치를 취할 수 있습니다.

인텐트: 이 경보는 인스턴스에 연결된 연결할 수 없는 Amazon EBS 볼륨을 감지하는 데 사용됩니다. 이로 인해 I/O 작업이 실패할 수 있습니다.

통계: Maximum

권장 임곗값: 1.0

임곗값 정당화: 상태 확인이 실패하면 이 지표의 값은 1입니다. 임곗값은 상태 확인이 실패할 때마다 경보가 ALARM 상태가 되도록 설정됩니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Amazon ElastiCache

CPUUtilization

측정 기준: CacheClusterId, CacheNodeId

경보 설명: 이 경보는 데이터베이스 엔진 프로세스와 인스턴스에서 실행 중인 기타 프로세스를 포함하여 전체 ElastiCache 인스턴스의 CPU 사용률을 모니터링하는 데 도움이 됩니다. AWS Elasticache는 Memcached와 Redis의 두 가지 엔진 유형을 지원합니다. Memcached 노드에서 CPU 사용률이 높아지면 인스턴스 유형을 확장하거나 새 캐시 노드를 추가하는 것을 고려해야 합니다. Redis의 경우 주 워크로드가 읽기 요청인 경우 캐시 클러스터에 읽기 전용 복제본을 추가하는 것을 고려해야 합니다. 주 워크로드가 쓰기 요청인 경우 클러스터형 모드에서 실행하는 경우 샤드를 추가하여 더 많은 프라이머리 노드에 워크로드를 분산하고, Redis를 비클러스터형 모드에서 실행하는 경우 인스턴스 유형을 확장하는 것을 고려해야 합니다.

인텐트: 이 경보는 ElastiCache 호스트의 높은 CPU 사용률을 감지하는 데 사용됩니다. 엔진이 아닌 프로세스를 포함하여 전체 인스턴스의 CPU 사용량을 폭넓게 파악하면 도움이 됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 임곗값을 애플리케이션의 중요한 CPU 사용률 수준을 반영하는 백분율로 설정합니다. Memcached의 경우 엔진은 최대 num_threads 코어를 사용할 수 있습니다. Redis의 경우 엔진은 대부분 단일 스레드이지만 I/O 가속화를 위해 가능한 경우 추가 코어를 사용할 수 있습니다. 대부분의 경우 임곗값을 사용 가능한 CPU의 약 90%로 설정할 수 있습니다. Redis는 단일 스레드이기 때문에 실제 임곗값은 노드 총 용량의 일부로 계산해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

CurrConnections

측정 기준: CacheClusterId, CacheNodeId

경보 설명: 이 경보는 높은 연결 수를 감지하며 이는 과부하 또는 성능 문제를 나타낼 수 있습니다. `CurrConnections`가 계속 증가하면 사용 가능한 65,000개의 연결이 고갈될 수 있습니다. 이는 애플리케이션 측에서 연결이 잘못 종료되고 서버 측에서는 설정된 상태로 남아 있는 것일 수 있습니다. 연결 풀링 또는 유휴 연결 제한 시간을 사용하여 클러스터에 대한 연결 수를 제한하거나, Redis의 경우 클러스터에서 tcp-keepalive를 조정하여 잠재적 데드 피어를 탐지하고 종료하는 방안을 고려해 보세요.

인텐트: 경보를 통해 ElastiCache 클러스터의 성능 및 안정성에 영향을 미칠 수 있는 높은 연결 수를 식별할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 클러스터의 허용 가능한 연결 범위에 따라 크게 달라집니다. ElastiCache 클러스터의 용량 및 예상 워크로드를 검토하고, 정기적인 사용 중 과거 연결 수를 분석하여 기준을 설정한 다음 그에 따라 임곗값을 선택합니다. 각 노드는 최대 65,000개의 동시 연결을 지원합니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_THRESHOLD

DatabaseMemoryUsagePercentage

측정 기준: CacheClusterId

경보 설명: 이 경보는 클러스터의 메모리 사용률을 모니터링하는 데 도움이 됩니다. `DatabaseMemoryUsagePercentage`가 100%에 도달하면 Redis maxmemory 정책이 트리거되고 선택한 정책에 따라 제거가 발생할 수 있습니다. 캐시에 제거 정책과 일치하는 객체가 없는 경우 쓰기 작업이 실패합니다. 일부 워크로드는 제거를 예상하거나 제거에 의존하지만 그렇지 않은 경우 클러스터의 메모리 용량을 늘려야 합니다. 프라이머리 노드를 추가하여 클러스터를 확장하거나 더 큰 노드 유형을 사용하여 클러스터를 확장할 수 있습니다. 자세한 내용은 Redis 클러스터용 ElastiCache 확장을 참조하세요.

인텐트: 이 경보는 클러스터의 높은 메모리 사용률을 감지하여 클러스터에 쓸 때 오류를 방지할 수 있습니다. 애플리케이션에서 제거가 발생할 것으로 예상되지 않는 경우 클러스터를 확장해야 하는 시기를 알면 도움이 됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 애플리케이션의 메모리 요구 사항과 ElastiCache 클러스터의 메모리 용량에 따라 임곗값을 클러스터의 중요 메모리 사용량 수준을 반영하는 백분율로 설정해야 합니다. 과거 메모리 사용량 데이터를 허용 가능한 메모리 사용량 임곗값에 대한 참조로 사용할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

EngineCPUUtilization

측정 기준: CacheClusterId

경보 설명: 이 경보는 ElastiCache 인스턴스에서 Redis 엔진 스레드의 CPU 사용률을 모니터링하는 데 도움이 됩니다. 엔진의 CPU 사용량이 높은 일반적인 이유는 CPU 사용량이 높은 오래 실행되는 명령, 많은 요청 수, 짧은 시간 내에 새 클라이언트 연결 요청 증가, 캐시에 새 데이터를 저장할 메모리가 충분하지 않을 경우 강제 제거 등이 있습니다. 노드를 추가하거나 인스턴스 유형을 확장하여 Redis용 ElastiCache 클러스터를 확장하는 것을 고려해야 합니다.

인텐트: 이 경보는 Redis 엔지 스레드의 높은 CPU 사용률을 감지하는 데 사용됩니다. 데이터베이스 엔진 자체의 CPU 사용량을 모니터링하는 경우에 유용합니다.

통계: Average

권장 임곗값: 90.0

임곗값 정당화: 임곗값을 애플리케이션의 중요한 엔진 CPU 사용률 수준을 반영하는 백분율로 설정합니다. 애플리케이션과 예상 워크로드를 사용하여 클러스터를 벤치마킹하고 EngineCPUUtilization과 성능의 상관 관계를 참조한 다음 그에 따라 임곗값을 설정할 수 있습니다. 대부분의 경우 임곗값을 사용 가능한 CPU의 약 90%로 설정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

ReplicationLag

측정 기준: CacheClusterId

경보 설명: 이 경보는 ElastiCache 클러스터의 복제 상태를 모니터링하는 데 도움이 됩니다. 복제 지연이 길어지면 프라이머리 노드나 복제본이 복제 속도를 따라갈 수 없다는 뜻입니다. 쓰기 작업이 너무 많으면 프라이머리 노드를 추가하여 클러스터를 확장하거나 더 큰 노드 유형을 사용하여 확장하는 것을 고려해 보세요. 자세한 내용은 Redis 클러스터용 ElastiCache 확장을 참조하세요. 읽기 요청량이 너무 많아 읽기 전용 복제본에 과부하가 걸리면 읽기 전용 복제본을 추가하는 것이 좋습니다.

인텐트: 이 경보는 프라이머리 노드의 데이터 업데이트와 복제본 노드와의 동기화 사이에서 지연을 감지하는 데 사용됩니다. 읽기 전용 복제본 클러스터 노드의 데이터 일관성을 보장하는 데 도움이 됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 애플리케이션의 요구 사항과 복제 지연의 잠재적 영향에 따라 임곗값을 설정합니다. 허용 가능한 복제 지연에 대해서는 애플리케이션의 예상 쓰기 속도와 네트워크 상황을 고려해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

Amazon EC2(**AWS/ElasticGPUs**)

GPUConnectivityCheckFailed

측정 기준: InstanceId, EGPUId

경보 설명: 이 경보는 인스턴스와 Elastic Graphics 액셀러레이터 간의 연결 장애를 감지하는 데 도움이 됩니다. Elastic Graphics에서는 인스턴스 네트워크를 사용하여 OpenGL 명령을 원격으로 연결된 그래픽 카드에 전송합니다. 또한 Elastic Graphics 액셀러레이터를 사용하여 OpenGL 애플리케이션을 실행하는 데스크톱은 일반적으로 원격 액세스 기술을 사용하여 액세스됩니다. OpenGL 렌더링 관련된 성능 문제와 데스크톱 원격 액세스 기술 관련 성능 문제를 구별해야 합니다. 이 문제에 대한 자세한 내용은 애플리케이션 성능 문제 조사를 참조하세요.

인텐트: 이 경보는 인스턴스에서 Elastic Graphics 액셀러레이터로의 연결 문제를 감지하는 데 사용됩니다.

통계: Maximum

권장 임곗값: 0.0

임곗값 정당화: 임곗값 1은 연결에 실패했음을 나타냅니다.

기간: 300

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: GREATER_THAN_THRESHOLD

GPUHealthCheckFailed

측정 기준: InstanceId, EGPUId

경보 설명: 이 경보는 Elastic 그래픽 액셀러레이터의 상태가 언제 비정상인지 알 수 있도록 도와줍니다. 액셀러레이터가 정상이 아닌 경우 비정상 상태 문제 해결의 문제 해결 단계를 참조하세요.

인텐트: 이 경보는 Elastic Graphics 액셀러레이터가 정상이 아닌지 감지하는 데 사용됩니다.

통계: Maximum

권장 임곗값: 0.0

임곗값 정당화: 임곗값 1은 상태 확인이 실패했음을 나타냅니다.

기간: 300

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: GREATER_THAN_THRESHOLD

Amazon ECS

CPUReservation

측정 기준: ClusterName

경보 설명: 이 경보는 ECS 클러스터의 높은 CPU 예약을 감지하는 데 도움이 됩니다. CPU 예약이 높으면 클러스터에 등록된 해당 작업에 사용할 CPU가 부족하다는 의미일 수 있습니다. 문제를 해결하려면 용량을 추가하거나 클러스터를 확장하거나 Auto Scaling을 설정합니다.

인텐트: 경보는 클러스터의 작업에 예약된 총 CPU 단위 수가 클러스터에 등록된 총 CPU 단위에 도달하는지 여부를 감지하는 데 사용됩니다. 이를 통해 클러스터를 확장해야 하는 시기를 알 수 있습니다. 클러스터의 총 CPU 단위에 도달하면 작업에 사용할 CPU가 부족해질 수 있습니다. EC2 용량 공급자의 관리 규모 조정이 활성화되거나 Fargate를 용량 공급자에 연결한 경우에는 이 경보를 사용하지 않는 것이 좋습니다.

통계: Average

권장 임곗값: 90.0

임곗값 정당화: CPU 예약 임곗값을 90%로 설정합니다. 또는 클러스터 특성에 따라 더 낮은 값을 선택해도 됩니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

CPUUtilization

측정 기준: ClusterName, ServiceName

경보 설명: 이 경보는 ECS 서비스의 높은 CPU 사용률을 감지하는 데 도움이 됩니다. 진행 중인 ECS 배포가 없는 경우 CPU 사용률이 최대치를 초과하면 리소스 병목 현상이나 애플리케이션 성능 문제가 있다는 의미일 수 있습니다. 문제를 해결하려면 CPU 제한을 늘리면 됩니다.

인텐트: 이 경보는 ECS 서비스의 높은 CPU 사용률을 감지하는 데 사용됩니다. CPU 사용률이 지속적으로 높으면 리소스 병목 현상이나 애플리케이션 성능 문제가 있다는 의미일 수 있습니다.

통계: Average

권장 임곗값: 90.0

임곗값 정당화: CPU 사용률에 대한 서비스 지표가 사용률 100%를 초과할 수 있습니다. 그러나 다른 서비스에 영향을 미치지 않도록 높은 CPU 사용률 지표를 모니터링하는 것이 좋습니다. 임곗값을 약 90~95%로 설정합니다. 향후 다른 서비스에서 문제가 발생하지 않도록 실제 사용량을 반영하여 작업 정의를 업데이트하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

MemoryReservation

측정 기준: ClusterName

경보 설명: 이 경보는 ECS 클러스터의 높은 메모리 예약을 감지하는 데 도움이 됩니다. 메모리 예약이 많으면 클러스터의 리소스 병목 현상이 나타날 수 있습니다. 문제를 해결하려면 서비스 작업의 성능을 분석하여 작업의 메모리 사용률을 최적화할 수 있는지 확인하세요. 또한 메모리를 더 등록하거나 Auto Scaling을 설정할 수 있습니다.

인텐트: 경보는 클러스터의 작업에 예약된 총 메모리 단위가 클러스터에 등록된 총 메모리 단위에 도달하는지 여부를 감지하는 데 사용됩니다. 이를 통해 클러스터를 확장해야 하는 시기를 알 수 있습니다. 클러스터의 총 메모리 단위에 도달하면 클러스터가 새 작업을 시작하지 못할 수 있습니다. EC2 용량 공급자의 관리 규모 조정이 활성화되거나 Fargate를 용량 공급자에 연결한 경우에는 이 경보를 사용하지 않는 것이 좋습니다.

통계: Average

권장 임곗값: 90.0

임곗값 정당화: 메모리 예약 임곗값을 90%로 설정합니다. 클러스터 특성에 따라 이 값을 더 낮게 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

HTTPCode_Target_5XX_Count

측정 기준: ClusterName, ServiceName

경보 설명: 이 경보는 ECS 서비스에 대한 서버 측 오류 수가 많을 경우 이를 감지하는 데 도움이 됩니다. 이는 오류가 발생하여 서버가 요청을 처리할 수 없게 되었음을 의미할 수 있습니다. 문제를 해결하려면 애플리케이션 로그를 확인하세요.

인텐트: 이 경보는 ECS 서비스에 대한 많은 서버 측 오류 수를 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 평균 트래픽의 약 5%에 해당하는 값을 계산하고 이 값을 임곗값의 시작점으로 사용합니다. `RequestCount` 지표를 사용하여 평균 트래픽을 확인할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다. 자주 발생하는 5XX 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

TargetResponseTime

측정 기준: ClusterName, ServiceName

경보 설명: 이 경보는 ECS 서비스 요청에 대한 높은 목표 응답 시간을 감지하는 데 도움이 됩니다. 이는 문제가 발생하여 서비스에서 요청을 제 시간에 처리할 수 없게 되었음을 의미할 수 있습니다. 문제를 해결하려면 CPU 사용률 지표를 확인하고 서비스에 CPU가 부족한지 확인하거나 서비스에서 사용하는 다른 다운스트림 서비스의 CPU 사용률을 확인하세요.

인텐트: 이 경보는 ECS 서비스 요청에 대한 높은 목표 응답 시간을 감지하는 데 사용됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 사용 사례에 따라 크게 달라집니다. 서비스의 목표 응답 시간에 대한 중요도 및 요구 사항을 검토하고 이 지표의 과거 동작을 분석하여 적절한 임곗값 수준을 결정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Container Insights가 포함된 Amazon ECS

EphemeralStorageUtilized

측정 기준: ClusterName, ServiceName

경보 설명: 이 경보는 Fargate 클러스터에서 활용되는 높은 임시 스토리지를 탐지하는 데 도움이 됩니다. 임시 스토리지가 지속적으로 높은 경우 임시 스토리지 사용량을 확인하고 임시 스토리지를 늘릴 수 있습니다.

의도: 이 경보는 Fargate 클러스터의 높은 임시 스토리지 사용량을 탐지하는 데 사용됩니다. 임시 스토리지 사용률이 지속적으로 높으면 디스크가 가득 차서 컨테이너에 장애가 발생할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 근거: 임곗값을 임시 스토리지 크기의 약 90%로 설정합니다. Fargate 클러스터의 허용 가능한 임시 스토리지 사용률을 기준으로 이 값을 조정할 수 있습니다. 일부 시스템의 경우 사용률이 지속적으로 높은 임시 스토리지를 사용하는 것이 정상일 수 있지만 다른 시스템의 경우 컨테이너 장애로 이어질 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

RunningTaskCount

측정 기준: ClusterName, ServiceName

경보 설명: 이 경보는 ECS 서비스에서 실행 중인 작업 수가 적은 것을 탐지하는 데 도움이 됩니다. 실행 중인 작업 수가 너무 적으면 애플리케이션이 서비스 로드를 처리할 수 없어 성능 문제가 발생할 수 있습니다. 실행 중인 작업이 없는 경우 Amazon ECS 서비스를 사용할 수 없거나 배포 문제가 있을 수 있습니다.

의도: 이 경보는 실행 중인 작업 수가 너무 적은지 여부를 탐지하는 데 사용됩니다. 실행 중인 작업이 지속적으로 적다는 것은 ECS 서비스 배포 또는 성능 문제를 의미할 수 있습니다.

통계: Average

권장 임곗값: 0.0

임곗값 근거: ECS 서비스의 최소 실행 작업 수를 기준으로 임곗값을 조정할 수 있습니다. 실행 중인 작업 수가 0인 경우 Amazon ECS 서비스를 사용할 수 없습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: LESS_THAN_OR_EQUAL_TO_THRESHOLD

instance_filesystem_utilization

측정기준: InstanceId, ContainerInstanceId, ClusterName

경보 설명: 이 경보는 ECS 클러스터의 높은 파일 시스템 사용률을 탐지하는 데 도움이 됩니다. 파일 시스템 사용률이 지속적으로 높으면 디스크 사용량을 확인합니다.

의도: 이 경보는 Amazon ECS 클러스터의 높은 파일 시스템 사용률을 탐지하는 데 사용됩니다. 파일 시스템 사용률이 지속적으로 높으면 리소스 병목 현상이나 애플리케이션 성능 문제를 의미할 수 있으며, 이로 인해 새 작업이 실행되지 않을 수 있습니다.

통계: Average

권장 임곗값: 90.0

임곗값 근거: 파일 시스템 사용률의 임곗값을 약 90~95%로 설정할 수 있습니다. Amazon ECS 클러스터의 허용 가능한 파일 시스템 용량 수준을 기준으로 이 값을 조정할 수 있습니다. 일부 시스템의 경우 파일 시스템 사용률이 지속적으로 높으면 정상일 뿐 문제가 아닌 것일 수 있지만, 다른 시스템에서는 문제의 원인이 되어 성능 문제로 이어지고 새 작업을 실행하지 못하게 될 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Amazon EFS

PercentIOLimit

측정 기준: FileSystemId

경보 설명: 이 경보는 워크로드가 파일 시스템에서 사용할 수 있는 I/O 한도 내에서 유지되게 하는 데 도움이 됩니다. 지표가 지속적으로 I/O 한도에 도달하는 경우 애플리케이션을 최대 I/O 성능 모드로 사용하는 파일 시스템으로 이동하는 것을 고려해 보세요. 문제를 해결하려면 파일 시스템에 연결된 클라이언트와 파일 시스템을 제한하는 클라이언트의 애플리케이션을 확인합니다.

인텐트: 이 경보는 파일 시스템이 범용 성능 모드의 I/O 제한에 얼마나 가깝게 도달해있는지를 감지하는 데 사용됩니다. I/O 비율이 지속적으로 높으면 I/O 요청에 따라 파일 시스템을 충분히 확장할 수 없고, 파일 시스템을 사용하는 애플리케이션에서 파일 시스템이 리소스 병목 현상을 일으킬 수 있다는 의미일 수 있습니다.

통계: Average

권장 임곗값: 100.0

임곗값 정당화: 파일 시스템이 I/O 한도에 도달하면 읽기 및 쓰기 요청에 대한 응답이 느려질 수 있습니다. 따라서 파일 시스템을 사용하는 애플리케이션에 영향을 미치지 않도록 지표를 모니터링하는 것이 좋습니다. 임곗값은 약 100%로 설정할 수 있습니다. 하지만 파일 시스템 특성에 따라 이 값을 더 낮게 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

BurstCreditBalance

측정 기준: FileSystemId

경보 설명: 이 경보는 파일 시스템 사용에 적용할 수 있는 버스트 크레딧 밸런스가 있는지 확인하는 데 도움이 됩니다. 사용 가능한 버스트 크레딧이 없는 경우 낮은 처리량으로 인해 파일 시스템에 대한 애플리케이션 액세스가 제한됩니다. 지표가 일관되게 0으로 떨어지면 처리량 모드를 탄력적 또는 프로비저닝된 처리량 모드로 변경하는 것이 좋습니다.

인텐트: 이 경보는 파일 시스템의 낮은 버스트 크레딧 밸런스를 감지하는 데 사용됩니다. 지속적으로 낮은 버스트 크레딧 밸런스는 처리량 저하 및 I/O 지연 시간 증가를 나타내는 지표가 될 수 있습니다.

통계: Average

권장 임곗값: 0.0

임곗값 정당화: 파일 시스템에서 버스트 크레딧이 부족하고 기준 처리량이 더 낮은 경우에도 EFS는 모든 파일 시스템에 1MiBps의 측정된 처리량을 계속 지원합니다. 하지만 파일 시스템이 애플리케이션의 리소스 병목 현상으로 작용하지 않도록 지표의 버스트 크레딧 밸런스가 낮은지 모니터링하는 것이 좋습니다. 임곗값은 약 0바이트로 설정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: LESS_THAN_OR_EQUAL_TO_THRESHOLD

Container Insights가 포함된 Amazon EKS

node_cpu_utilization

측정 기준: ClusterName

경보 설명: 이 경보는 EKS 클러스터의 워커 노드에서 높은 CPU 사용률을 감지하는 데 도움이 됩니다. 사용률이 지속적으로 높으면 워커 노드를 CPU가 더 큰 인스턴스로 교체하거나 시스템을 수평적으로 확장해야 할 수 있습니다.

인텐트: 이 경보는 시스템 성능이 저하되지 않도록 EKS 클러스터에 있는 워커 노드의 CPU 사용률을 모니터링하는 데 도움이 됩니다.

통계: Maximum

권장 임곗값: 80.0

임곗값 정당화: 시스템에서 영향을 받기 시작하기 전에 문제를 디버깅할 충분한 시간을 확보할 수 있도록 임곗값을 80% 이하로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

node_filesystem_utilization

측정 기준: ClusterName

경보 설명: 이 경보는 EKS 클러스터의 워커 노드에서 높은 파일 시스템 사용률을 감지하는 데 도움이 됩니다. 사용률이 지속적으로 높으면 워커 노드를 업데이트하여 디스크 볼륨을 늘리거나 수평적으로 확장해야 할 수 있습니다.

경보 설명: 이 경보는 EKS 클러스터의 워커 노드에서 높은 파일 시스템 사용률을 모니터링하는 데 도움이 됩니다. 사용률이 100%에 도달하면 애플리케이션 장애, 디스크 I/O 병목 현상, 포드 제거 또는 노드가 완전히 응답하지 않게 될 수 있습니다.

통계: Maximum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 디스크 압력이 충분하면(즉 디스크가 꽉 차면) 노드가 비정상으로 표시되고 포드가 노드에서 제거됩니다. 디스크 압력이 큰 노드의 포드는 사용 가능한 파일 시스템이 kubelet에 설정된 제거 임곗값보다 낮을 경우에 제거됩니다. 클러스터에서 노드가 제거되기 전에 대응할 충분한 시간을 확보할 수 있도록 경보 임곗값을 설정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

node_memory_utilization

측정 기준: ClusterName

경보 설명: 이 경보는 EKS 클러스터의 워커 노드에서 높은 메모리 사용률을 감지하는 데 도움이 됩니다. 사용률이 지속적으로 높으면 포드 복제본 수를 조정하거나 애플리케이션을 최적화해야 할 수 있습니다.

인텐트: 이 경보는 시스템 성능이 저하되지 않도록 EKS 클러스터에 있는 워커 노드의 메모리 사용률을 모니터링하는 데 도움이 됩니다.

통계: Maximum

권장 임곗값: 80.0

임곗값 정당화: 시스템에서 영향을 받기 시작하기 전에 문제를 디버깅할 충분한 시간을 확보할 수 있도록 임곗값을 80% 이하로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

pod_cpu_utilization_over_pod_limit

측정 기준: ClusterName, Namespace, Service

경보 설명: 이 경보는 EKS 클러스터의 포드에서 높은 CPU 사용률을 감지하는 데 도움이 됩니다. 사용률이 지속적으로 높으면 영향을 받는 포드의 CPU 한도를 늘려야 할 수 있습니다.

인텐트: 이 경보는 EKS 클러스터의 Kubernetes 서비스에 속하는 포드의 CPU 사용률을 모니터링하는 데 도움이 되며 이를 통해 서비스의 포드가 예상보다 높은 CPU를 소비하고 있는지 빠르게 식별할 수 있습니다.

통계: Maximum

권장 임곗값: 80.0

임곗값 정당화: 시스템에서 영향을 받기 시작하기 전에 문제를 디버깅할 충분한 시간을 확보할 수 있도록 임곗값을 80% 이하로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

pod_memory_utilization_over_pod_limit

측정 기준: ClusterName, Namespace, Service

경보 설명: 이 경보는 EKS 클러스터의 포드에서 높은 메모리 사용률을 감지하는 데 도움이 됩니다. 사용률이 지속적으로 높으면 영향을 받는 포드의 메모리 한도를 늘려야 할 수 있습니다.

인텐트: 이 경보는 시스템 성능이 저하되지 않도록 EKS 클러스터에 있는 포드의 메모리 사용률을 모니터링하는 데 도움이 됩니다.

통계: Maximum

권장 임곗값: 80.0

임곗값 정당화: 시스템에서 영향을 받기 시작하기 전에 문제를 디버깅할 충분한 시간을 확보할 수 있도록 임곗값을 80% 이하로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Amazon Kinesis Data Streams

GetRecords.IteratorAgeMilliseconds

측정 기준: StreamName

경보 설명: 이 경보는 반복자 최대 수명이 너무 높은 경우 이를 감지할 수 있습니다. 실시간 데이터 처리 애플리케이션의 경우 지연 허용치에 따라 데이터 보존을 구성합니다. 이는 보통 몇 분 이내입니다. 기록 데이터를 처리하는 애플리케이션의 경우 이 지표를 사용하여 캐치업 속도를 모니터링합니다. 데이터 손실을 막는 빠른 해결책은 문제를 해결하는 동안 보존 기간을 늘리는 것입니다. 또한 소비자 애플리케이션에서 레코드를 처리하는 작업자 수를 늘릴 수 있습니다. 반복자 수명이 점점 증가하는 가장 일반적인 원인은 물리적 리소스가 부족하거나 레코드 처리 로직이 스트림 처리량의 증가에 따라 조정되지 않기 때문입니다. 자세한 내용은 링크를 참조하세요.

인텐트: 이 경보는 스트림의 데이터가 너무 오래 보존되거나 레코드 처리 속도가 너무 느려 만료되는 것을 감지하는 데 사용됩니다. 이를 통해 스트림 보존 기간의 100%에 도달한 후에도 데이터 손실을 방지할 수 있습니다.

통계: Maximum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 스트림 보존 기간 및 레코드의 처리 지연 허용 한도에 따라 크게 달라집니다. 요구 사항을 검토하고 과거 추세를 분석한 다음 임곗값을 심각한 처리 지연을 나타내는 밀리초로 설정합니다. 반복자 수명이 보존 기간(기본적으로 24시간, 최대 365일까지 구성 가능)의 50%를 경과하면 레코드 만료로 인해 데이터가 손실될 위험이 있습니다. 지표를 모니터링하여 샤드가 이 한도에 도달하지 않게 할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

GetRecords.Success

측정 기준: StreamName

경보 설명: 이 지표는 소비자가 스트림에서 데이터를 성공적으로 읽을 때마다 증가합니다. `GetRecords`는 예외가 발생해도 데이터를 반환하지 않습니다. 가장 일반적인 예외는

ProvisionedThroughputExceededException입니다. 스트림에 대한 요청 속도가 너무 높거나 주어진 시간 동안 사용 가능한 처리량이 이미 제공되었기 때문입니다. 요청의 횟수 또는 크기를 줄입니다. 자세한 내용은 Amazon Kinesis Data Streams 개발자 안내서의 스트림 제한 및 AWS의 오류 재시도 및 지수 백오프를 참조하세요.

인텐트: 이 경보는 소비자가 스트림에서 레코드를 검색하는 데 실패하는지 여부를 감지할 수 있습니다. 이 지표에 대해 경보를 설정하면 오류율 증가 또는 검색 성공 감소 등 데이터 소비 관련 문제를 사전에 감지할 수 있습니다. 이를 통해 적시에 조치를 취하여 잠재적 문제를 해결하고 원활한 데이터 처리 파이프라인을 유지할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 스트림에서 레코드 검색의 중요도에 따라 실패한 레코드에 대한 애플리케이션의 허용 범위를 기반으로 임곗값을 설정합니다. 임곗값은 성공한 작업에 해당하는 비율이어야 합니다. 과거 GetRecords 지표 데이터를 허용 가능한 실패율에 대한 참조로 사용할 수 있습니다. 또한 실패한 레코드는 다시 시도될 수 있으므로 임곗값을 설정할 때는 재시도를 고려해야 합니다. 이렇게 하면 일시적 스파이크로 인해 불필요한 경보가 트리거되는 것을 방지할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: LESS_THAN_THRESHOLD

PutRecord.Success

측정 기준: StreamName

경보 설명: 이 경보는 실패한 PutRecord 작업 수가 임곗값을 초과할 때 이를 감지합니다. 데이터 생산자 로그를 조사하여 실패의 근본 원인을 찾습니다. 가장 일반적인 이유는 ProvisionedThroughputExceededException 문제를 일으킨 샤드의 프로비저닝 처리량이 충분하지 않기 때문입니다. 스트림에 대한 요청 속도가 너무 높거나 샤드로 인제스트하려는 처리량이 너무 높아서 발생합니다. 요청의 횟수 또는 크기를 줄입니다. 자세한 내용은 스트림 제한 및 AWS의 오류 재시도 및 지수 백오프를 참조하세요.

인텐트: 이 경보는 스트림으로의 레코드 수집 실패 여부를 감지할 수 있습니다. 스트림에 데이터를 쓸 때 발생하는 문제를 식별하는 데 도움이 됩니다. 이 지표에 경보를 설정하면 제작자가 스트림에

데이터를 게시할 때 발생하는 문제(예: 오류율 증가 또는 레코드 게시 성공 감소)를 사전에 감지할 수 있습니다. 이를 통해 적시에 조치를 취하여 잠재적 문제를 해결하고 신뢰할 수 있는 데이터 수집 프로세스를 유지할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 서비스에서 데이터 수집 및 처리의 중요도에 따라 실패한 레코드에 대한 애플리케이션의 허용 범위를 기반으로 임곗값을 설정합니다. 임곗값은 성공한 작업에 해당하는 비율이어야 합니다. 과거 PutRecord 지표 데이터를 허용 가능한 실패율에 대한 참조로 사용할 수 있습니다. 또한 실패한 레코드는 다시 시도될 수 있으므로 임곗값을 설정할 때는 재시도를 고려해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: LESS_THAN_THRESHOLD

PutRecords.FailedRecords

측정 기준: StreamName

경보 설명: 이 경보는 실패한 `PutRecords` 수가 임곗값을 초과할 때 이를 감지합니다. Kinesis Data Streams는 각 `PutRecords` 요청의 모든 레코드를 처리하려고 시도하지만 단일 레코드 장애가 발생해도 후속 레코드 처리가 중단되지 않습니다. 이러한 실패의 주요 원인은 스트림 또는 개별 샤드의 처리량 초과입니다. 일반적인 원인은 트래픽 스파이크와 네트워크 지연 현상으로 인해 레코드가 스트림에 일정하지 않게 도착하는 것입니다. 따라서 성공적으로 처리되지 않은 레코드를 찾아 후속 호출에서 재시도해야 합니다. 자세한 내용은 PutRecords 사용 시 실패 처리를 참조하세요.

인텐트: 이 경보는 일괄 작업을 사용하여 스트림에 레코드를 넣을 때 일관된 오류를 감지할 수 있습니다. 이 지표에 경보를 설정하면 실패한 레코드의 증가를 사전에 감지하고 적시에 조치를 취하여 근본적인 문제를 해결하며 원활하고 안정적인 데이터 수집 프로세스를 보장할 수 있습니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 실패한 레코드에 대한 애플리케이션의 허용 범위를 반영하여 실패 레코드 수로 임곗값을 설정합니다. 과거 데이터를 허용 가능한 실패 값에 대한 참조로 사용할 수 있습니다. 또한 실

패한 레코드는 후속 PutRecords 호출에서 재시도될 수 있으므로 임곗값을 설정할 때 재시도를 고려해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

ReadProvisionedThroughputExceeded

측정 기준: StreamName

경보 설명: 경보는 읽기 처리량 용량 제한을 초래하는 레코드 수를 추적합니다. 지속적으로 제한이 발생하는 경우 스트림에 샤드를 추가하여 프로비저닝된 읽기 처리량을 늘리는 것을 고려해야 합니다. 스트림에서 실행 중인 소비자 애플리케이션이 두 개 이상이고 `GetRecords` 한도를 공유하는 경우 Enhanced Fan-Out을 통해 새 소비자 애플리케이션을 등록하는 것이 좋습니다. 샤드를 추가해도 제한 수가 줄어들지 않으면 특정 "핫" 샤드가 다른 샤드보다 더 많이 읽히고 있을 수 있습니다. 향상된 모니터링을 활성화하고 "핫" 샤드를 찾아 분할합니다.

인텐트: 이 경보는 소비자가 프로비저닝된 읽기 처리량(보유한 샤드 수에 따라 결정됨)을 초과할 경우 제한되는지 여부를 감지할 수 있습니다. 이 경우 스트림에서 읽을 수 없게 되며 스트림이 백업을 시작할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 일반적으로 제한이 발생한 요청은 재시도될 수 있으므로 임곗값을 0으로 설정하면 경보가 너무 민감해집니다. 하지만 지속적인 제한은 스트림의 읽기에 영향을 줄 수 있으므로 경보를 트리거해야 합니다. 애플리케이션 및 재시도 구성에 대해 제한이 발생한 요청에 따라 임곗값을 백분율로 설정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

SubscribeToShardEvent.MillisBehindLatest

측정 기준: StreamName, ConsumerName

경보 설명: 이 경보는 애플리케이션의 레코드 처리 지연이 임곗값을 초과할 때 이를 감지합니다. 다운스트림 애플리케이션에 대한 API 작업 실패와 같은 일시적인 문제로 인해 지표가 갑자기 증가할 수 있습니다. 이러한 문제가 지속적으로 발생하는지 조사해야 합니다. 일반적인 원인은 물리적 리소스가 충분하지 않거나 스트림 처리량 증가에 따라 확장되지 않는 레코드 처리 로직 때문에 소비자가 레코드를 충분히 빠르게 처리하지 못하는 것입니다. 중요 경로에서 호출을 차단하면 레코드 처리 속도가 느려지는 경우가 많습니다. 샤드 수를 늘려 병렬 처리를 늘릴 수 있습니다. 또한 수요가 최고조에 달할 때는 기본 프로세싱 노드에 충분한 물리적 리소스가 있는지 확인해야 합니다.

인텐트: 이 경보는 스트림의 샤드 이벤트 구독 지연을 감지할 수 있습니다. 이는 처리 지연을 나타내며 소비자 애플리케이션의 성능 또는 전체 스트림 상태와 관련된 잠재적 문제를 식별하는 데 도움이 될 수 있습니다. 처리 지연이 심각해지면 병목 현상이나 소비자 애플리케이션 비효율성을 조사하고 해결하여 실시간 데이터 처리를 보장하고 데이터 백로그를 최소화해야 합니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 애플리케이션이 허용할 수 있는 지연에 따라 크게 달라집니다. 애플리케이션의 요구 사항을 검토하고 과거 추세를 분석한 다음 그에 따라 임곗값을 선택합니다. SubscribeToShard 호출이 성공하면 소비자는 최대 5분 동안 지속적인 연결을 통해 SubscribeToShardEvent 이벤트를 수신하기 시작합니다. 이 시간이 지난 후 계속 레코드를 수신하려면 SubscribeToShard를 다시 호출하여 구독을 갱신해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

WriteProvisionedThroughputExceeded

측정 기준: StreamName

경보 설명: 이 경보는 쓰기 처리량 용량 제한을 초래하는 레코드 수가 임곗값에 도달했을 때 이를 감지합니다. 생산자가 프로비저닝된 쓰기 처리량(보유한 샤드 수에 따라 결정됨)을 초과하면 전송이 제한되고 레코드를 스트림에 넣을 수 없게 됩니다. 지속적인 제한 문제를 해결하려면 스트림에 샤

드를 추가하는 것을 고려해야 합니다. 이렇게 하면 프로비저닝된 쓰기 처리량이 증가하고 향후 제한이 방지됩니다. 레코드를 수집할 때 파티션 키 선택도 고려해야 합니다. 무작위 파티션 키는 가급적 스트림의 샤드 전체에 균등하게 레코드를 분산시키기 때문에 선호됩니다.

인텐트: 이 경보는 스트림이나 샤드의 제한으로 인해 생산자가 레코드 작성을 거부당하는지 여부를 감지할 수 있습니다. 스트림이 Provisioned 모드인 경우 이 경보를 설정하면 데이터 스트림이 한도에 도달했을 때 사전 조치를 취할 수 있으므로 프로비저닝된 용량을 최적화하거나 적절한 조정 조치를 취하여 데이터 손실을 방지하고 원활한 데이터 처리를 유지할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 일반적으로 제한이 발생한 요청은 재시도될 수 있으므로 임곗값을 0으로 설정하면 경보가 너무 민감해집니다. 하지만 지속적인 제한은 스트림 쓰기에 영향을 미칠 수 있으므로 이를 감지하도록 경보 임곗값을 설정해야 합니다. 애플리케이션 및 재시도 구성에 대해 제한이 발생한 요청에 따라 임곗값을 백분율로 설정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Lambda

ClaimedAccountConcurrency

측정 기준: 없음

경보 설명: 이 경보는 Lambda 함수의 동시성이 계정의 리전 수준 동시성 한도에 근접하는지 모니터링하는 데 도움이 됩니다. 동시 실행 한도에 도달하면 함수에 제한이 발생하기 시작합니다. 제한 현상을 방지하려면 다음 작업을 수행할 수 있습니다.

1. 이 리전의 동시 접속자 수 증가를 요청합니다.
2. 미사용 예약 동시성 또는 프로비저닝된 동시성을 식별하여 줄일 수 있습니다.
3. 함수의 성능 문제를 식별하고 처리 속도를 개선하여 처리량을 개선합니다.
4. 함수를 간접 호출할 때마다 더 많은 메시지가 처리되도록 함수의 배치 크기를 늘립니다.

인텐트: 이 경보는 Lambda 함수의 동시성이 계정의 리전 수준 동시성 할당량에 근접하는지 사전에 감지하여 조치를 취할 수 있도록 해줍니다. `ClaimedAccountConcurrency`가 계정의 리전 수준 동시성 할당량에 도달하면 함수가 제한됩니다. 예약 동시성(RC) 또는 프로비저닝된 동시성(PC)을 사용하는 경우 이 경보를 사용하면 `ConcurrentExecutions`보다 동시성 사용률을 더 잘 파악할 수 있습니다.

통계: Maximum

권장 임곗값: 상황에 따라 다름

임계값 정당화: 리전 내 계정에 설정된 동시 접속자 수 할당량의 약 90%의 값을 계산하고 그 결과를 임계값으로 사용해야 합니다. 기본적으로 리전 내 모든 함수에 걸쳐 사용자 계정의 동시성 할당량은 1,000입니다. 하지만 Service Quotas 대시보드에서 계정의 할당량을 확인해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_THRESHOLD

오류

측정 기준: FunctionName

경보 설명: 이 경보는 높은 오류 수를 감지합니다. 오류에는 코드에서 발생하는 예외와 Lambda 런타임에서 발생하는 예외가 포함됩니다. 함수와 관련된 로그를 확인하여 문제를 진단할 수 있습니다.

인텐트: 경보는 함수 호출 시 오류 수가 많은 경우를 감지하는 데 도움이 됩니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 임곗값을 0보다 큰 수로 설정합니다. 정확한 값은 애플리케이션의 오류 허용 오차에 따라 달라질 수 있습니다. 함수가 처리하는 호출의 중요도를 이해해야 합니다. 일부 애플리케이션의 경우 어떠한 오류도 허용되지 않을 수 있지만 다른 애플리케이션에서는 일정한 오차 범위를 허용할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: GREATER_THAN_THRESHOLD

제한

측정 기준: FunctionName

경보 설명: 이 경보는 제한이 발생한 간접 호출 요청 수가 많음을 감지합니다. 스케일 업에 사용할 수 있는 동시성이 없는 경우 제한이 발생합니다. 이 문제를 해결할 몇 가지 방법이 있습니다. 1) 이 리전의 AWS Support에 동시성 증가를 요청합니다. 2) 함수의 성능 문제를 식별하고 처리 속도를 개선하여 처리량을 개선합니다. 3) 함수를 간접 호출할 때마다 더 많은 메시지가 처리되도록 함수의 배치 크기를 늘립니다.

인텐트: 경보는 Lambda 함수에 대해 많은 수의 제한된 간접 호출 요청 수를 감지하는 데 도움이 됩니다. 제한으로 인해 요청이 계속 거부되는지, 지속적인 제한을 피하기 위해 Lambda 함수 성능을 개선하거나 동시성 용량을 늘려야 하는지를 파악해야 합니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 임곗값을 0보다 큰 수로 설정합니다. 정확한 임곗값은 애플리케이션의 허용 오차에 따라 달라질 수 있습니다. 함수의 사용 및 조정 요구 사항에 따라 임곗값을 설정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

지속 시간

측정 기준: FunctionName

경보 설명: 이 경보는 Lambda 함수가 이벤트를 처리하는 데 오랜 시간이 걸리는 것을 감지합니다. 함수 코드가 변경되어 함수를 실행하는 데 시간이 더 오래 걸리거나 함수의 종속성이 더 오래 걸리기 때문에 지속 시간이 길어질 수 있습니다.

인텐트: 이 경보는 Lambda 함수의 긴 실행 기간을 감지할 수 있습니다. 런타임 기간이 길면 함수를 간접 호출하는 데 시간이 더 오래 걸리고 Lambda가 더 많은 수의 이벤트를 처리하는 경우 간접 호

출의 동시 실행 용량에도 영향을 미칠 수 있습니다. Lambda 함수의 실행 시간이 계속 예상보다 길어지는지 파악해야 합니다.

통계: p90

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 기간 임곗값은 애플리케이션과 워크로드, 성능 요구 사항에 따라 달라집니다. 고성능 요구 사항의 경우 임곗값을 더 짧은 시간으로 설정하여 함수가 기대치를 충족하는지 확인합니다. 또한 기간 지표에 대한 과거 데이터를 분석하여 소요 시간이 함수의 예상 성능과 일치하는지 확인한 다음 임곗값을 과거 평균보다 긴 시간으로 설정할 수 있습니다. 임곗값을 구성된 함수의 제한 시간보다 낮게 설정해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

ConcurrentExecutions

측정 기준: FunctionName

경보 설명: 이 경보는 함수의 동시성이 계정의 리전 수준 동시성 한도에 근접하는지 모니터링하는 데 도움이 됩니다. 동시 실행 한도에 도달하면 함수에 제한이 발생하기 시작합니다. 제한 현상을 방지하려면 다음 작업을 수행할 수 있습니다.

1. 이 리전의 동시 접속자 수 증가를 요청합니다.
2. 함수의 성능 문제를 식별하고 처리 속도를 개선하여 처리량을 개선합니다.
3. 함수를 간접 호출할 때마다 더 많은 메시지가 처리되도록 함수의 배치 크기를 늘립니다.

예약된 동시성 및 프로비저닝된 동시성 사용률을 더 잘 파악하려면 대신 새 지표 `ClaimedAccountConcurrency`에 경보를 설정하세요.

인텐트: 이 경보는 함수의 동시성이 계정의 리전 수준 동시성 할당량에 근접하는지 사전에 감지하여 조치를 취할 수 있도록 해줍니다. 함수가 계정의 리전 수준 동시성 할당량에 도달하면 함수가 제한됩니다.

통계: Maximum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 임곗값을 해당 리전의 계정에 설정된 동시성 할당량의 약 90%로 설정합니다. 기본적으로 리전 내 모든 함수에 걸쳐 사용자 계정의 동시성 할당량은 1,000입니다. 하지만 AWS 지원팀에 문의하여 계정의 할당량을 늘릴 수 있으므로 계정 할당량을 확인합니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_THRESHOLD

Lambda Insights

Lambda Insights 지표에 대해 모범 사례 경보를 설정하는 것이 좋습니다.

memory_utilization

측정 기준: function_name

경보 설명: 이 경보는 Lambda 함수의 메모리 사용률이 구성된 한도에 근접하는지 감지하는 데 사용됩니다. 문제 해결을 위해 1) 코드 최적화를 시도합니다. 2) 메모리 요구량을 정확하게 예측하여 메모리 할당량을 적절하게 조정합니다. 이에 대한 내용은 Lambda Power Tuning을 참조하세요. 3) 연결 풀링을 사용합니다. RDS 데이터베이스의 연결 풀링에 대한 내용은 Lambda와 함께 Amazon RDS 프록시 사용을 참조하세요. 4) 호출 사이에 대량의 데이터를 메모리에 저장하지 않도록 함수를 설계하는 것도 고려해 볼 수 있습니다.

인텐트: 이 경보는 Lambda 함수의 메모리 사용률이 구성된 한도에 근접하는지 감지하는 데 사용됩니다.

통계: Average

입곗값 제안: 90.0

임곗값 정당화: 메모리 사용률이 할당된 메모리의 90%를 초과할 때 경보를 받으려면 임곗값을 90%로 설정합니다. 메모리 사용량에 대한 워크로드가 우려되는 경우 이 값을 더 낮게 조정할 수 있습니다. 또한 이 지표의 과거 데이터를 확인하고 그에 따라 임곗값을 설정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_THRESHOLD

Amazon VPC(**AWS/NATGateway**)

ErrorPortAllocation

측정 기준: NatGatewayId

경보 설명: 이 경보는 NAT 게이트웨이가 새 연결에 포트를 할당할 수 없는 경우를 감지하는 데 도움이 됩니다. 이 문제를 해결하려면 NAT 게이트웨이의 포트 할당 오류 해결을 참조하세요.

인텐트: 이 경보는 NAT 게이트웨이가 소스 포트를 할당할 수 없는지 여부를 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: ErrorPortAllocation의 값이 0보다 크면 NATGateway를 통해 인기 있는 단일 대상에 대한 동시 연결이 너무 많이 열려 있음을 의미합니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

PacketsDropCount

측정 기준: NatGatewayId

경보 설명: 이 경보는 NAT 게이트웨이가 패킷을 삭제한 시점을 감지하는 데 도움이 됩니다. 이는 NAT 게이트웨이 문제로 인해 발생할 수 있으므로 AWS 서비스 상태 대시보드에서 해당 리전의 AWS NAT 게이트웨이 상태를 확인합니다. 이를 통해 NAT 게이트웨이를 사용하는 트래픽과 관련된 네트워크 문제의 상관 관계를 파악할 수 있습니다.

인텐트: 이 경보는 NAT 게이트웨이가 패킷을 삭제하는지 여부를 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: NAT 게이트웨이 총 트래픽의 0.01%를 계산하고 이 결과를 임곗값으로 사용해야 합니다. NAT 게이트웨이 트래픽의 과거 데이터를 사용하여 임곗값을 결정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

AWS 프라이빗 링크(**AWS/PrivateLinkEndpoints**)

PacketsDropped

측정 기준: VPC Id, VPC 엔드포인트 Id, 엔드포인트 유형,서브넷 Id, 서비스 이름

경보 설명: 이 경보는 엔드포인트에서 삭제한 패킷 수를 모니터링하여 엔드포인트 또는 엔드포인트 서비스가 비정상인지 여부를 감지하는 데 도움이 됩니다. VPC 엔드포인트에 도착하는 크기가 8500바이트보다 큰 패킷은 삭제됩니다. 문제 해결은 인터페이스 VPC 엔드포인트와 엔드포인트 서비스 간의 연결 문제를 참조하세요.

인텐트: 이 경보는 엔드포인트 또는 엔드포인트 서비스가 비정상인지 여부를 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 사용 사례에 따라 임곗값을 설정합니다. 엔드포인트 또는 엔드포인트 서비스의 비정상 상태를 파악하려면 엄청난 데이터 손실이 발생하기 전에 문제를 해결할 수 있도록 임곗값을 낮게 설정해야 합니다. 과거 데이터를 사용하여 삭제된 패킷의 허용 범위를 파악하고 그에 따라 임곗값을 설정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

AWS 프라이빗 링크(**AWS/PrivateLinkServices**)

RstPacketsSent

측정 기준: 서비스 Id, 로드 밸런서 Arn, Az

경보 설명: 이 경보를 사용하면 엔드포인트로 전송되는 재설정 패킷 수를 기반으로 엔드포인트 서비스의 비정상 대상을 탐지할 수 있습니다. 서비스 소비자와의 연결 오류를 디버깅할 때 서비스가 RstPacketsSent 지표를 사용하여 연결을 재설정하는지 또는 네트워크 경로에서 다른 오류가 발생하는지 확인할 수 있습니다.

인텐트: 이 경보는 엔드포인트 서비스의 비정상 대상을 탐지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 임곗값은 사용 사례에 따라 다릅니다. 사용 사례에서 대상이 비정상인 것을 허용할 수 있는 경우 임곗값을 높게 설정할 수 있습니다. 사용 사례에서 대상이 비정상인 것을 허용할 수 없는 경우 임곗값을 매우 낮게 설정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Amazon RDS

CPUUtilization

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 지속적으로 높은 CPU 사용률을 모니터링하는 데 도움이 됩니다. CPU 사용률은 비유휴 시간을 측정합니다. 향상된 모니터링 또는 성능 개선 도우미를 사용하여 MariaDB, MySQL, Oracle 및 PostgreSQL에 대해 CPU 시간(`guest`, `irq`, `wait`, `nice` 등)을 가장 많이 소비하는 대기 시간을 검토하는 것이 좋습니다. 그런 다음, CPU를 가장 많이 소비하는 쿼리를 평가합니다. 워크로드를 조정할 수 없는 경우 더 큰 DB 인스턴스 클래스로 이동하는 것을 고려합니다.

의도: 이 경보는 매우 긴 응답 시간과 시간 초과를 방지하기 위해 지속적으로 높은 CPU 사용률을 탐지하는 데 사용됩니다. CPU 사용률의 마이크로 버스팅을 확인하려는 경우 경보 평가 시간을 더 짧게 설정할 수 있습니다.

통계: Average

권장 임곗값: 90.0

임곗값 근거: CPU 사용률이 무작위로 급증해도 데이터베이스 성능이 저하되지는 않지만 CPU가 계속 높게 유지되면 향후 데이터베이스 요청에 방해가 될 수 있습니다. 전체 데이터베이스 워크로드에 따라 RDS/Aurora 인스턴스의 CPU가 높으면 전체 성능이 저하될 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

DatabaseConnections

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 많은 수의 연결을 탐지합니다. 기존 연결을 검토하여 '비활동' 상태이거나 제대로 닫히지 않은 연결을 종료합니다. 연결 풀링을 사용하여 새 연결 수를 제한하는 것이 좋습니다. 또는 더 많은 메모리를 가진 클래스를 사용하도록 DB 인스턴스 크기를 늘려서 'max_connections'의 기본값을 높이거나, 워크로드를 지원할 수 있는 경우 RDS와 Aurora MySQL 및 PostgreSQL에서 현재 클래스의 'max_connections' 값을 늘립니다.

의도: 이 경보는 최대 DB 연결 수에 도달했을 때 연결이 거부되는 것을 방지하는 데 사용됩니다. DB 인스턴스 클래스를 자주 변경하는 경우 메모리와 기본 최대 연결 수가 변경되므로 이 경보는 사용하지 않는 것이 좋습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 근거: 허용되는 연결 수는 DB 인스턴스 클래스의 크기 및 프로세스/연결과 관련된 데이터베이스 엔진별 파라미터에 따라 달라집니다. 데이터베이스의 최대 연결 수의 90~95% 사이 값을 계산하고 해당 결과를 임곗값으로 사용해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

EBSByteBalance%

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 남아 있는 낮은 비율의 처리량 크레딧을 모니터링하는 데 도움이 됩니다. 문제 해결은 RDS의 대기 시간 문제를 참조하세요.

의도: 이 경보는 버스트 버킷에 남아 있는 낮은 비율의 처리량 크레딧을 탐지하는 데 사용됩니다. 바이트 밸런스 비율이 낮으면 처리량 병목 문제가 발생할 수 있습니다. Aurora PostgreSQL 인스턴스에는 이 경보를 사용하지 않는 것이 좋습니다.

통계: Average

권장 임곗값: 10.0

임곗값 근거: 처리량 크레딧 밸런스가 10% 미만이면 나쁨으로 간주되므로 임곗값을 적절히 설정해야 합니다. 애플리케이션이 워크로드에 대해 낮은 처리량을 허용할 수 있는 경우 더 낮은 임곗값을 설정할 수도 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: LESS_THAN_THRESHOLD

EBSIOBalance%

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 남아 있는 낮은 비율의 IOPS 크레딧을 모니터링하는 데 도움이 됩니다. 문제 해결은 RDS의 대기 시간 문제를 참조하세요.

의도: 이 경보는 버스트 버킷에 남아 있는 낮은 비율의 I/O 크레딧을 탐지하는 데 사용됩니다. IOPS 밸런스 비율이 낮으면 IOPS 병목 문제가 발생할 수 있습니다. Aurora 인스턴스에는 이 경보를 사용하지 않는 것이 좋습니다.

통계: Average

권장 임곗값: 10.0

임곗값 근거: IOPS 크레딧 밸런스가 10% 미만이면 나쁨으로 간주되므로 임곗값을 적절히 설정합니다. 애플리케이션이 워크로드에 대해 낮은 IOPS를 허용할 수 있는 경우 더 낮은 임곗값을 설정할 수도 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: LESS_THAN_THRESHOLD

FreeableMemory

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 데이터베이스 연결이 급증하거나 인스턴스가 높은 메모리 압박을 받고 있음을 의미할 수 있는 사용 가능한 메모리 부족을 모니터링하는 데 도움이 됩니다. `FreeableMemory` 외에도 `SwapUsage`에 대한 CloudWatch 지표를 모니터링하여 메모리 부족을 확인합니다. 인스턴스 메모리 소비가 자주 너무 높으면 워크로드를 확인하거나 인스턴스 클래스를 업그레이드해야 합니다. Aurora 리더 DB 인스턴스의 경우 클러스터에 리더 DB 인스턴스를 추가하는 것을 고려합니다. Aurora 문제 해결에 대한 자세한 내용은 여유 메모리 부족 문제를 참조하세요.

의도: 이 경보는 메모리 부족으로 인한 연결 거부를 방지하는 데 사용됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 근거: 워크로드 및 인스턴스 클래스에 따라 임곗값을 다르게 설정하는 것이 적절할 수 있습니다. 가용 메모리가 장기간 동안 전체 메모리의 25% 미만으로 떨어지지 않는 것이 좋습니다. Aurora의 경우 임곗값을 5%에 가깝게 설정할 수 있습니다. 지표가 0에 가까울수록 DB 인스턴스가 최대한 스케일 업되었음을 의미하기 때문입니다. 이 지표의 과거 동작을 분석하여 적절한 임곗값 수준을 결정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: LESS_THAN_THRESHOLD

FreeLocalStorage

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 사용 가능한 로컬 스토리지 부족을 모니터링하는 데 도움이 됩니다. Aurora PostgreSQL 호환 버전은 오류 로그와 임시 파일을 저장하는 데 로컬 스토리지를 사용합니다. Aurora MySQL은 로컬 스토리지를 사용하여 오류 로그, 일반 로그, 느린 쿼리 로그, 감사 로그 및 비 InnoDB 임시 테이블을 저장합니다. 이러한 로컬 스토리지 볼륨은 Amazon EBS Store에서 백업하며, 더 큰 D 인스턴스 클래스를 사용하여 확장할 수 있습니다. 문제 해결을 위해서는 Aurora PostgreSQL 호환과 MySQL 호환을 확인하세요.

의도: 이 경보는 Aurora Serverless v2 이상을 사용하지 않는 경우 Aurora DB 인스턴스가 로컬 스토리지 한도에 얼마나 근접했는지 탐지하는 데 사용됩니다. 임시 테이블 및 로그 파일과 같은 비영구 데이터를 로컬 스토리지에 저장하면 로컬 스토리지 용량이 초과될 수 있습니다. 이 경보를 통해 DB 인스턴스의 로컬 스토리지가 부족해질 때 발생하는 공간 부족 오류를 방지할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 근거: 볼륨 사용 속도와 추세를 기반으로 사용 가능한 스토리지 양의 약 10~20%를 계산한 다음, 해당 결과를 임곗값으로 사용하여 볼륨이 한도에 도달하기 전에 사전에 조치를 취해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: LESS_THAN_THRESHOLD

FreeStorageSpace

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 사용 가능한 스토리지 공간이 부족한지 감시합니다. 스토리지 용량 한도에 자주 도달하는 경우 데이터베이스 스토리지 스케일 업을 고려합니다. 애플리케이션의 예기치 않은 수요 증가에 대비해 약간의 버퍼를 포함합니다. 또는 RDS 스토리지 Auto Scaling을 활성화하는 것도 고려합니다. 또한 사용하지 않거나 오래된 데이터 및 로그를 삭제하여 더 많은 공간을 확보하는 것도 고려합니다. 자세한 내용은 RDS 스토리지 부족 문서 및 PostgreSQL 스토리지 문제 문서를 참조하세요.

의도: 이 경보는 스토리지 가득 참 문제를 방지하는 데 도움이 됩니다. 이렇게 하면 데이터베이스 인스턴스의 스토리지가 부족할 때 발생하는 가동 중지를 방지할 수 있습니다. 스토리지 Auto Scaling을 활성화했거나 데이터베이스 인스턴스의 스토리지 용량을 자주 변경하는 경우에는 이 경보를 사용하지 않는 것이 좋습니다.

통계: Minimum

권장 임곗값: 상황에 따라 다름

임곗값 근거: 임곗값은 현재 할당된 스토리지 공간에 따라 달라집니다. 일반적으로 할당된 스토리지 공간의 10% 값을 계산하고 해당 결과를 임곗값으로 사용해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: LESS_THAN_THRESHOLD

MaximumUsedTransactionID(MaximumUsedTransactionIDs)

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 PostgreSQL의 트랜잭션 ID 랩어라운드를 방지하는 데 도움이 됩니다. 이 블로그의 문제 해결 단계를 참조하여 문제를 조사하고 해결합니다. 또한 이 블로그를 참조하여 autovacuum 개념, 일반적인 문제 및 모범 사례에 대해 더 자세히 알아볼 수 있습니다.

의도: 이 경보는 PostgreSQL의 트랜잭션 ID 랩어라운드를 방지하는 데 사용됩니다.

통계: Average

권장 임곗값: 1.0E9

임곗값 근거: 이 임곗값을 10억으로 설정하면 문제를 조사할 시간을 확보할 수 있습니다. 기본 autovacuum_freeze_max_age 값은 2억입니다. 가장 오래된 트랜잭션의 기간이 10억이면 autovacuum은 이 임곗값을 2억 개의 트랜잭션 ID 목표 미만으로 유지하는 데 문제가 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 1

평가 기간: 1

비교 연산자: GREATER_THAN_THRESHOLD

ReadLatency

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 긴 읽기 지연 시간을 모니터링하는 데 도움이 됩니다. 스토리지 지연 시간이 길면 워크로드가 리소스 제한을 초과하기 때문입니다. 인스턴스 및 할당된 스토리지 구성을 기준으로 I/O 사용률을 검토할 수 있습니다. Amazon RDS 인스턴스의 IOPS 병목 현상으로 인해 발생하는 Amazon EBS 볼륨의 대기 시간 문제를 해결하려면 어떻게 해야 합니까?를 참조하세요. Aurora의 경우 I/O-Optimized 스토리지 구성이 있는 인스턴스 클래스로 전환할 수 있습니다. 지침은 Planning I/O in Aurora를 참조하세요.

의도: 이 경보는 긴 읽기 지연 시간을 탐지하는 데 사용됩니다. 데이터베이스 디스크는 일반적으로 읽기/쓰기 지연 시간이 짧지만 작업 지연 시간이 길어질 수 있는 문제가 있을 수 있습니다.

통계: p90

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 사용 사례에 따라 크게 달라집니다. 읽기 지연 시간이 20밀리초보다 길면 조사가 필요할 수 있습니다. 애플리케이션의 읽기 작업 지연 시간이 길어질 수 있는 경우 더 높은 임곗값을 설정할 수도 있습니다. 읽기 지연 시간의 중요도와 요구 사항을 검토하고 이 지표의 과거 동작을 분석하여 적절한 임곗값 수준을 결정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

ReplicaLag

측정기준: DBInstanceIdentifier

경보 설명: 이 알람은 복제본이 기본 인스턴스보다 뒤처진 시간(초)을 파악하는 데 도움이 됩니다. PostgreSQL 읽기 전용 복제본은 원본 데이터베이스 인스턴스에서 사용자 트랜잭션이 발생하지 않는 경우 최대 5분의 복제 지연을 보고합니다. ReplicaLag 지표가 0에 도달하면 복제본이 기본 DB 인스턴스를 따라잡은 것입니다. ReplicaLag 지표가 -1을 반환하는 경우 복제가 현재 활성 상태가 아닙니다. RDS PostgreSQL과 관련된 지침은 복제 모범 사례를 참조하고 `ReplicaLag` 및 관련 오류 문제 해결은 ReplicaLag 문제 해결을 참조하세요.

의도: 이 경보는 기본 인스턴스의 장애 시 발생할 수 있는 데이터 손실을 반영하는 복제 지연을 탐지할 수 있습니다. 복제본이 기본 인스턴스보다 너무 뒤쳐져 기본 인스턴스에 장애가 발생하면 복제본에서 기본 인스턴스에 있던 데이터가 누락됩니다.

통계: Maximum

권장 임곗값: 60.0

임곗값 근거: 일반적으로 허용되는 지연은 애플리케이션에 따라 다릅니다. 60초를 넘지 않는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: GREATER_THAN_THRESHOLD

WriteLatency

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 긴 쓰기 지연 시간을 모니터링하는 데 도움이 됩니다. 스토리지 지연 시간이 길면 워크로드가 리소스 제한을 초과하기 때문입니다. 인스턴스 및 할당된 스토리지 구성을 기준으로 I/O 사용률을 검토할 수 있습니다. Amazon RDS 인스턴스의 IOPS 병목 현상으로 인해 발생하는 Amazon EBS 볼륨의 대기 시간 문제를 해결하려면 어떻게 해야 합니까?를 참조하세요. Aurora의 경우 I/O-Optimized 스토리지 구성이 있는 인스턴스 클래스로 전환할 수 있습니다. 지침은 Planning I/O in Aurora를 참조하세요.

의도: 이 경보는 긴 쓰기 지연 시간을 탐지하는 데 사용됩니다. 데이터베이스 디스크는 일반적으로 읽기/쓰기 지연 시간이 짧지만 작업 지연 시간이 길어지는 문제가 발생할 수 있습니다. 이를 모니터링하면 디스크 지연 시간이 예상만큼 낮은지 확인할 수 있습니다.

통계: p90

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 사용 사례에 따라 크게 달라집니다. 쓰기 지연 시간이 20밀리초보다 길면 조사가 필요할 수 있습니다. 애플리케이션의 쓰기 작업 지연 시간이 길어질 수 있는 경우 더 높은 임곗값을 설정할 수도 있습니다. 쓰기 지연 시간의 중요도와 요구 사항을 검토하고 이 지표의 과거 동작을 분석하여 적절한 임곗값 수준을 결정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

DBLoad

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 높은 DB 로드를 모니터링하는 데 도움이 됩니다. 프로세스 수가 vCPUs 수를 초과하면 프로세스가 대기열에 추가되기 시작합니다. 대기열이 늘어나면 성능에 영향이 있습니다. DB 로드가 최대 vCPU를 초과하는 경우가 많고 기본 대기 상태가 CPU인 경우 CPU에 과부하가 걸립니다. 이 경우 성능 개선 도우미/향상된 모니터링에서 `CPUUtilization`, `DBLoadCPU` 및 대기 중인 작업을 모니터링할 수 있습니다. 연결 수를 인스턴스에 맞게 조절하거나, CPU 로드가 높은 SQL 쿼리를 미세 조정하거나, 인스턴스 클래스의 크기를 늘리는 것이 좋습니다. 대기 상태의 인스턴스가 높고 일관적이라는 것은 해결해야 할 병목 현상이나 리소스 경합 문제가 있을 수 있음을 나타냅니다.

의도: 이 경보는 높은 DB 로드를 탐지하는 데 사용됩니다. DB 로드가 높으면 DB 인스턴스에서 성능 문제가 발생할 수 있습니다. 이 경보는 서버리스 DB 인스턴스에는 해당되지 않습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 근거: 최대 vCPU 값은 DB 인스턴스의 vCPU(가상 CPU) 코어 수에 따라 결정됩니다. 최대 vCPU에 따라 다른 임곗값이 적절할 수 있습니다. 이상적으로 DB 로드가 vCPU 라인을 넘지 않아야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

AuroraVolumeBytesLeftTotal

측정기준: DBClusterIdentifier

경보 설명: 이 경보는 낮은 잔여 총 볼륨을 모니터링하는 데 도움이 됩니다. 남은 총 볼륨이 크기 제한에 도달하면 클러스터는 공간 부족 오류를 보고합니다. Aurora 스토리지는 클러스터 볼륨의 데이터에 따라 자동으로 규모가 조정되며 DB 엔진 버전에 따라 최대 128TiB 또는 64TiB까지 확장됩니다. 더 이상 필요하지 않은 테이블과 데이터베이스를 삭제하여 스토리지를 줄이는 것을 고려합니다. 자세한 내용은 스토리지 조정을 확인하세요.

의도: 이 경보는 Aurora 클러스터가 볼륨 크기 제한에 얼마나 근접했는지 탐지하는 데 사용됩니다. 이 경보는 클러스터의 공간이 부족할 때 발생하는 공간 부족 오류를 방지할 수 있습니다. 이 경보는 Aurora MySQL에만 권장됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 근거: 볼륨 사용량 증가 속도 및 추세를 기준으로 실제 크기 제한의 10~20%를 계산한 다음, 해당 결과를 임곗값으로 사용하여 볼륨이 제한에 도달하기 전에 사전 조치를 취해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: LESS_THAN_THRESHOLD

AuroraBinlogReplicaLag

측정기준: DBClusterIdentifier, 역할=WRITER

경보 설명: 이 경보는 Aurora writer 인스턴스 복제의 오류 상태를 모니터링하는 데 도움이 됩니다. 자세한 내용은 AWS 리전 간 Aurora MySQL DB 클러스터 복제를 참조하세요. 문제 해결은 Aurora MySQL 복제 문제를 참조하세요.

의도: 이 경보는 작성자 인스턴스가 오류 상태이고 소스를 복제할 수 없는지 여부를 탐지하는 데 사용됩니다. 이 경보는 Aurora MySQL에만 권장됩니다.

통계: Average

권장 임곗값: -1.0

임곗값 근거: 복제본이 오류 상태인 경우 Aurora MySQL에서 이 값을 게시하므로 임곗값으로 -1을 사용하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 2

평가 기간: 2

비교 연산자: LESS_THAN_OR_EQUAL_TO_THRESHOLD

BlockedTransactions

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 Aurora DB 인스턴스에서 차단된 높은 트랜잭션 수를 모니터링하는 데 도움이 됩니다. 차단된 트랜잭션은 롤백 또는 커밋으로 종료될 수 있습니다. 높은 동시성, 트랜잭션 유휴 또는 장기 실행 트랜잭션으로 인해 트랜잭션이 차단될 수 있습니다. 문제 해결은 Aurora MySQL 설명서를 참조하세요.

의도: 이 경보는 트랜잭션 롤백 및 성능 저하를 방지하기 위해 Aurora DB 인스턴스에서 많은 수의 차단된 트랜잭션을 탐지하는 데 사용됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 근거: `ActiveTransactions` 지표를 사용하여 인스턴스의 전체 트랜잭션 중 5%를 계산하고 해당 결과를 임곗값으로 사용해야 합니다. 또한 차단된 트랜잭션의 중요도와 요구 사항을 검토하고 이 지표의 과거 동작을 분석하여 적절한 임곗값 수준을 결정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

BufferCacheHitRatio

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 Aurora 클러스터의 지속적으로 낮은 캐시 적중률을 모니터링하는 데 도움이 됩니다. 적중률이 낮다면 이 DB 인스턴스에 대한 쿼리가 디스크로 자주 이동한다는 뜻입니다. 문제를 해결하려면 워크로드를 조사하여 어떤 쿼리가 이 동작을 일으키는지 확인하고 DB 인스턴스 RAM 권장 사항 문서를 참조하세요.

의도: 이 경보는 Aurora 인스턴스의 지속적인 성능 저하를 방지하기 위해 일관되게 낮은 캐시 적중률을 탐지하는 데 사용됩니다.

통계: Average

권장 임곗값: 80.0

임곗값 근거: 버퍼 캐시 적중률의 임곗값을 80%로 설정할 수 있습니다. 하지만 허용 가능한 성능 수준과 워크로드 특성에 따라 이 값을 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 10

평가 기간: 10

비교 연산자: LESS_THAN_THRESHOLD

EngineUptime

측정기준: DBClusterIdentifier, 역할=WRITER

경보 설명: 이 경보는 라이터 DB 인스턴스의 가동 중지 시간이 짧은지 모니터링하는 데 도움이 됩니다. 라이터 DB 인스턴스는 재부팅, 유지 관리, 업그레이드 또는 장애 조치로 인해 다운될 수 있습니다. 클러스터의 장애 조치로 인해 가동 시간이 0에 도달하고 클러스터에 하나 이상의 Aurora 복제본이 있으면 실패 이벤트 중에 Aurora 복제본이 기본 라이터 인스턴스로 승격됩니다. DB 클러스터의 가용성을 높이려면 두 개 이상의 서로 다른 가용 영역에 하나 이상의 Aurora 복제본을 생성하는 것을 고려합니다. 자세한 내용은 Aurora 가동 중지에 영향을 미치는 요인을 참조하세요.

의도: 이 경보는 Aurora 라이터 DB 인스턴스의 가동 중지 여부를 탐지하는 데 사용됩니다. 이를 통해 중단 또는 장애 조치로 인해 발생하는 라이터 인스턴스의 장기 실행 장애를 방지할 수 있습니다.

통계: Average

권장 임곗값: 0.0

임곗값 근거: 실패 이벤트로 인해 잠시 중단이 발생하며 그 동안 읽기 및 쓰기 작업은 예외와 함께 실패합니다. 하지만, 일반적인 서비스 복구 시간은 60초 미만이지만 대부분 30초 미만에 복원됩니다.

기간: 60

경보를 보낼 데이터 포인트: 2

평가 기간: 2

비교 연산자: LESS_THAN_OR_EQUAL_TO_THRESHOLD

RollbackSegmentHistoryListLength

측정기준: DBInstanceIdentifier

경보 설명: 이 경보는 Aurora 인스턴스의 일관되게 높은 롤백 세그먼트 기록 길이를 모니터링하는 데 도움이 됩니다. InnoDB 기록 목록 길이가 길면 오래된 행 버전, 쿼리 및 데이터베이스 종료가 많아 속도가 느려진 것입니다. 자세한 내용 및 문제 해결은 InnoDB 기록 목록 길이가 크게 늘어남을 참조하세요.

의도: 이 경보는 일관되게 긴 롤백 세그먼트 기록 길이를 탐지하는 데 사용됩니다. 이를 통해 Aurora 인스턴스에서 지속적인 성능 저하와 높은 CPU 사용률을 방지할 수 있습니다. 이 경보는 Aurora MySQL에만 권장됩니다.

통계: Average

권장 임곗값: 1000000.0

임곗값 근거: 이 임곗값을 100만으로 설정하면 문제를 조사할 시간을 확보할 수 있습니다. 하지만 허용 가능한 성능 수준과 워크로드 특성에 따라 이 값을 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

StorageNetworkThroughput

측정기준: DBClusterIdentifier, 역할=WRITER

경보 설명: 이 경보는 높은 스토리지 네트워크 처리량을 모니터링하는 데 도움이 됩니다. 스토리지 네트워크 처리량이 EC2 인스턴스의 총 네트워크 대역폭을 초과할 경우 읽기 및 쓰기 지연 시간이 길어져 성능이 저하될 수 있습니다. AWS 콘솔에서 EC2 인스턴스 유형을 확인할 수 있습니다. 문제 해결을 위해 쓰기/읽기 지연 시간에 대한 변경 사항을 확인하고 이 지표에서도 경보가 발생했는지 평가합니다. 경보가 발생한 경우 경보가 트리거된 시간 동안의 워크로드 패턴을 평가합니다. 이를 통해 워크로드를 최적화하여 총 네트워크 트래픽 양을 줄일 수 있는지 확인할 수 있습니다. 이것이 불가능한 경우 인스턴스 크기 조정을 고려해야 할 수도 있습니다.

의도: 이 경보는 높은 스토리지 네트워크 처리량을 탐지하는 데 사용됩니다. 높은 처리량을 탐지하면 네트워크 패킷 손실과 성능 저하를 방지할 수 있습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 근거: EC2 인스턴스 유형의 총 네트워크 대역폭의 약 80~90%를 계산한 다음, 해당 결과를 임곗값으로 사용하여 네트워크 패킷이 영향을 받기 전에 사전 조치를 취해야 합니다. 또한 스토리지 네트워크 처리량의 중요도와 요구 사항을 검토하고 이 지표의 과거 동작을 분석하여 적절한 임곗값 수준을 결정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Amazon Route 53 Public Data Plane

HealthCheckStatus

측정 기준: HealthCheckId

경보 설명: 이 경보는 상태 검사기에 따라 비정상 엔드포인트를 탐지하는 데 도움이 됩니다. 장애가 발생하여 비정상 상태가 된 이유를 이해하려면 Route 53 Health Check Console의 상태 검사기 탭을 사용하여 각 리전의 상태와 상태 확인의 마지막 실패를 확인합니다. 상태 탭에는 엔드포인트가 비정상으로 보고된 이유도 표시됩니다. 문제 해결 단계를 참조하세요.

인텐트: 이 경보는 Route53 상태 검사기를 사용하여 비정상 엔드포인트를 탐지합니다.

통계: Average

권장 임곗값: 1.0

임곗값 정당화: 엔드포인트가 정상이면 상태가 1로 보고됩니다. 1 미만이면 모두 비정상입니다.

기간: 60

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: LESS_THAN_THRESHOLD

Amazon S3

4xxErrors

측정 기준: BucketName, FilterId

경보 설명: 이 경보를 통해 클라이언트 요청에 대한 응답으로 생성된 4xx 오류 상태 코드의 총 개수를 보고할 수 있습니다. 예를 들어 403 오류 코드는 잘못된 IAM 정책을 나타내고, 404 오류 코드는 클라이언트 애플리케이션이 제대로 작동하지 않음을 나타낼 수 있습니다. S3 서버 액세스 로깅을 일시적으로 활성화하면 HTTP 상태 및 오류 코드 필드를 사용하여 문제의 원인을 정확히 찾아낼 수 있습니다. 오류 코드에 대한 자세한 내용은 오류 응답을 참조하세요.

인텐트: 이 경보는 일반적인 4xx 오류 발생률에 대한 기준을 만드는 데 사용되므로 설정 문제를 나타낼 수 있는 비정상이 있는지 확인할 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 권장 임곗값은 전체 요청의 5% 이상에서 4XX 오류가 발생하는지 감지하는 것입니다. 자주 발생하는 4XX 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다. 허용 가능한 수준의 4XX 오류를 고려하여 요청 로드에 맞게 임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

5xxErrors

측정 기준: BucketName, FilterId

경보 설명: 이 경보는 많은 수의 서버 측 오류 감지에 도움이 됩니다. 이러한 오류는 클라이언트가 요청을 했지만 서버가 완료하지 못했음을 나타냅니다. 이를 통해 애플리케이션이 S3로 인해 직면

한 문제의 상관 관계를 파악할 수 있습니다. 오류를 효율적으로 처리하거나 줄이는 데 도움이 되는 자세한 내용은 성능 설계 패턴 최적화를 참조하세요. 오류는 S3 문제로 인해 발생할 수도 있으므로 AWS 서비스 상태 대시보드에서 해당 리전의 Amazon S3 상태를 확인합니다.

인텐트: 이 경보는 5xx 오류로 인해 애플리케이션에 문제가 발생하는지 감지하는 데 도움이 될 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 전체 요청의 5% 이상에서 5XXError가 발생하는지 감지하도록 임곗값을 설정하는 것이 좋습니다. 하지만 요청의 트래픽과 허용 가능한 오류 발생률에 맞게 임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

OperationsFailedReplication

측정 기준: SourceBucket, DestinationBucket, RuleId

경보 설명: 이 경보는 복제 실패를 이해하는 데 도움이 됩니다. 이 지표는 S3 CRR 또는 S3 SRR을 사용하여 복제된 새 객체의 상태를 추적하고 S3 배치 복제를 사용하여 복제된 기존 객체도 추적합니다. 자세한 내용은 복제 문제 해결을 참조하세요.

인텐트: 이 경보는 실패한 복제 작업이 있는지 감지하는 데 사용됩니다.

통계: Maximum

권장 임곗값: 0.0

임곗값 정당화: 이 지표는 작업이 성공하면 값을 0으로 내보내고, 1분 동안 수행된 복제 작업이 없는 경우에는 아무 것도 출력하지 않습니다. 지표가 0보다 큰 값을 내보내는 경우 복제 작업은 실패입니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

S3ObjectLambda

4xxErrors

측정 기준: AccessPointName, DataSourceARN

경보 설명: 이 경보는 클라이언트 요청에 대한 응답으로 생성된 4xx 오류 상태 코드의 총 개수를 보고하는 데 도움이 됩니다. S3 서버 액세스 로깅을 일시적으로 활성화하면 HTTP 상태 및 오류 코드 필드를 사용하여 문제의 원인을 정확히 찾아낼 수 있습니다.

인텐트: 이 경보는 일반적인 4xx 오류 발생률에 대한 기준을 만드는 데 사용되므로 설정 문제를 나타낼 수 있는 비정상이 있는지 확인할 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 전체 요청의 5% 이상에서 4XXError가 발생하는지 감지하도록 임곗값을 설정하는 것이 좋습니다. 자주 발생하는 4XX 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다. 허용 가능한 수준의 4XX 오류를 고려하여 요청 로드에 맞게 임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

5xxErrors

측정 기준: AccessPointName, DataSourceARN

경보 설명: 이 경보는 많은 수의 서버 측 오류 감지에 도움이 됩니다. 이러한 오류는 클라이언트가 요청을 했지만 서버가 완료하지 못했음을 나타냅니다. 이러한 오류는 S3 문제로 인해 발생할 수도

있으므로 AWS 서비스 상태 대시보드에서 해당 리전의 Amazon S3 상태를 확인합니다. 이를 통해 애플리케이션이 S3로 인해 직면한 문제의 상관 관계를 파악할 수 있습니다. 이러한 오류를 효율적으로 처리하거나 줄이는 데 도움이 되는 자세한 내용은 성능 설계 패턴 최적화를 참조하세요.

인텐트: 이 경보는 5xx 오류로 인해 애플리케이션에 문제가 발생하는지 감지하는 데 도움이 될 수 있습니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 전체 요청의 5% 이상에서 5XX 오류가 발생하는지 감지하도록 임곗값을 설정하는 것이 좋습니다. 하지만 요청의 트래픽과 허용 가능한 오류 발생률에 맞게 임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

LambdaResponse4xx

측정 기준: AccessPointName, DataSourceARN

경보 설명: 이 경보는 S3 Object Lambda에 대한 호출에서 장애(500s)를 감지하고 진단하는 데 도움이 됩니다. 이러한 오류는 요청에 대한 응답을 담당하는 Lambda 함수의 오류 또는 잘못된 구성으로 인해 발생할 수 있습니다. 객체 Lambda 액세스 포인트와 관련된 Lambda 함수의 CloudWatch 로그 스트림을 조사하면 S3 객체 Lambda의 응답을 기반으로 문제의 원인을 정확히 찾아낼 수 있습니다.

인텐트: 이 경보는 WriteGetObjectResponse 호출에서 발생하는 4xx 클라이언트 오류를 탐지하는 데 사용됩니다.

통계: Average

권장 임곗값: 0.05

임곗값 정당화: 전체 요청의 5% 이상에서 4XXError가 발생하는지 감지하도록 임곗값을 설정하는 것이 좋습니다. 자주 발생하는 4XX 오류는 경보가 필요합니다. 하지만 임곗값을 너무 낮게 설정하면 경보가 너무 민감해질 수 있습니다. 허용 가능한 수준의 4XX 오류를 고려하여 요청 로드에 맞게

임곗값을 조정할 수 있습니다. 또한 과거 데이터를 분석하여 애플리케이션 워크로드에 대해 허용 가능한 오류 발생률을 확인한 다음 그에 따라 임곗값을 조정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_THRESHOLD

Amazon SNS

NumberOfMessagesPublished

측정 기준: TopicName

경보 설명: 이 경보는 게시된 SNS 메시지 수가 너무 적을 때 이를 감지할 수 있습니다. 문제 해결을 위해 게시자가 보내는 트래픽이 적은 이유를 확인합니다.

인텐트: 이 경보를 사용하면 알림 게시의 현저한 저하를 사전에 모니터링하고 감지할 수 있습니다. 이를 통해 애플리케이션 또는 비즈니스 프로세스의 잠재적 문제를 식별하여 적절한 조치를 취해 예상되는 알림 흐름을 유지할 수 있습니다. 시스템에서 처리하는 트래픽이 최소화될 것으로 예상되면 이 경보를 생성해야 합니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 게시되는 메시지 수는 애플리케이션에 게시될 것으로 예상되는 메시지 수와 일치해야 합니다. 또한 과거 데이터, 추세 및 트래픽을 분석하여 적절한 임곗값을 찾을 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: LESS_THAN_THRESHOLD

NumberOfNotificationsDelivered

측정 기준: TopicName

경보 설명: 이 경보는 전달된 SNS 메시지 수가 너무 적을 때 이를 감지할 수 있습니다. 이는 의도하지 않은 엔드포인트 구독 취소나 메시지 지연을 유발하는 SNS 이벤트 때문일 수 있습니다.

인텐트: 이 경보는 전송된 메시지 양의 감소를 감지하는 데 도움이 됩니다. 시스템에서 처리하는 트래픽이 최소화될 것으로 예상되면 이 경보를 생성해야 합니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 전달되는 메시지 수는 생성된 예상 메시지 수 및 소비자 수와 일치해야 합니다. 또한 과거 데이터, 추세 및 트래픽을 분석하여 적절한 임곗값을 찾을 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: LESS_THAN_THRESHOLD

NumberOfNotificationsFailed

측정 기준: TopicName

경보 설명: 이 경보는 실패한 SNS 메시지 수가 너무 많을 때 이를 감지할 수 있습니다. 실패한 알림 문제를 해결하려면 CloudWatch Logs에 로깅을 활성화합니다. 로그를 확인하면 실패한 구독자와 해당 구독자가 반환하는 상태 코드를 찾는 데 도움이 될 수 있습니다.

인텐트: 이 경보를 통해 알림 전달과 관련된 문제를 사전에 발견하고 적절한 조치를 취해 문제를 해결할 수 있습니다.

통계: Sum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 실패한 알림의 영향에 따라 크게 달라집니다. 최종 사용자에게 제공되는 SLA, 내결함성, 알림의 중요도를 검토하고 과거 데이터를 분석한 다음 그에 따라 임곗값을 선택합니다. SQS, Lambda 또는 Firehose 구독만 있는 주제의 경우 실패한 알림 수는 0이어야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

NumberOfNotificationsFilteredOut-InvalidAttributes

측정 기준: TopicName

경보 설명: 이 경보는 게시자 또는 구독자의 잠재적 문제를 모니터링하고 해결하는 데 도움이 됩니다. 게시자가 잘못된 속성을 가진 메시지를 게시하고 있는지 또는 구독자에게 부적절한 필터가 적용되었는지 확인합니다. 또한 CloudWatch Logs를 분석하여 문제의 근본 원인을 찾을 수 있습니다.

인텐트: 경보는 게시된 메시지가 유효하지 않은지 또는 구독자에게 부적절한 필터가 적용되었는지 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: 잘못된 속성은 게시자의 실수인 경우가 많습니다. 정상 시스템에서는 잘못된 속성이 예상되지 않으므로 임곗값을 0으로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

NumberOfNotificationsFilteredOut-InvalidMessageBody

측정 기준: TopicName

경보 설명: 이 경보는 게시자 또는 구독자의 잠재적 문제를 모니터링하고 해결하는 데 도움이 됩니다. 게시자가 메시지 본문이 잘못된 메시지를 게시하고 있는지 또는 구독자에게 부적절한 필터가 적용되었는지 확인합니다. 또한 CloudWatch Logs를 분석하여 문제의 근본 원인을 찾을 수 있습니다.

인텐트: 경보는 게시된 메시지가 유효하지 않은지 또는 구독자에게 부적절한 필터가 적용되었는지 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: 잘못된 메시지 본문은 게시자의 실수인 경우가 많습니다. 정상 시스템에서는 잘못된 메시지 본문이 예상되지 않으므로 임곗값을 0으로 설정하는 것이 좋습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

NumberOfNotificationsRedrivenToDlq

측정 기준: TopicName

경보 설명: 이 경보는 DLQ(Dead Letter Queue)로 이동되는 메시지 수를 모니터링하는 데 도움이 됩니다.

인텐트: 경보는 DLQ(Dead Letter Queue)로 이동된 메시지를 감지하는 데 사용됩니다. SNS가 SQS, Lambda 또는 Firehose와 결합된 경우 이 경보를 생성하는 것이 좋습니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: 구독자 유형과 상관없이 정상 시스템에서는 메시지를 DLQ(Dead Letter Queue)로 이동해서는 안 됩니다. 메시지가 대기열에 도착하면 알림을 받는 것이 좋습니다. 근본 원인을 식별하여 해결할 수 있고 DLQ(Dead Letter Queue)에 있는 메시지를 다시 입력하여 데이터 손실을 방지할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

NumberOfNotificationsFailedToRedriveToDlq

측정 기준: TopicName

경보 설명: 이 경보는 DLQ(Dead Letter Queue)로 이동할 수 없는 메시지를 모니터링하는 데 도움이 됩니다. DLQ(Dead Letter Queue)가 존재하고 올바르게 구성되어 있는지 확인합니다. 또한 SNS

에 DLQ(Dead Letter Queue) 액세스 권한이 있는지도 확인합니다. 자세한 내용은 DLQ(Dead Letter Queue) 설명서를 참조하세요.

인텐트: 경보는 DLQ(Dead Letter Queue)로 이동하지 못한 메시지를 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: 메시지를 DLQ(Dead Letter Queue)로 이동할 수 없는 경우는 거의 항상 실수입니다. 권장 임곗값은 0입니다. 즉, 대기열이 구성되면 처리에 실패한 모든 메시지가 DLQ(Dead Letter Queue)에 도달할 수 있어야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

SMSMonthToDateSpentUSD

측정 기준: TopicName

경보 설명: 이 경보는 SNS에서 메시지를 전송할 수 있는 계정 할당량이 충분한지 모니터링하는 데 도움이 됩니다. 할당량에 도달하면 SNS에서 SMS 메시지를 전송할 수 없습니다. 사용자의 월별 SMS 지출 할당량 설정 또는 AWS에서 지출 할당량 증가 요청에 대한 자세한 내용은 SMS 메시징 기본 설정을 참조하세요.

인텐트: 이 경보는 계정에 SMS 메시지가 성공적으로 전송되기에 충분한 할당량이 있는지 확인하는 데 사용됩니다.

통계: Maximum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 계정의 할당량(계정 지출 한도)에 따라 임곗값을 설정합니다. 할당량 증가를 요청할 시간을 확보할 수 있도록 할당량 한도에 도달했음을 충분히 일찍 알려주는 임곗값을 선택합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

SMSSuccessRate

측정 기준: TopicName

경보 설명: 이 경보는 SMS 메시지 전송 실패율을 모니터링하는 데 도움이 됩니다. Cloudwatch Logs를 설정하여 장애의 특성을 파악하고 이에 따라 조치를 취할 수 있습니다.

인텐트: 이 경보는 SMS 메시지 전송 실패를 탐지하는 데 사용됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: SMS 메시지 전송 실패에 대한 허용 한도에 맞춰 경보 임곗값을 설정합니다.

기간: 60

경보를 보낼 데이터 포인트: 5

평가 기간: 5

비교 연산자: GREATER_THAN_THRESHOLD

Amazon SQS

ApproximateAgeOfOldestMessage

측정 기준: QueueName

경보 설명: 이 경보는 대기열에 있는 가장 오래된 메시지의 수명을 감시합니다. 이 경보를 사용하여 소비자가 원하는 속도로 SQS 메시지를 처리하고 있는지 모니터링할 수 있습니다. 소비자 수 또는 소비자 처리량을 늘려 메시지 사용 기간을 줄이는 방안을 고려해 보세요. 이 지표를 `ApproximateNumberOfMessagesVisible`와 함께 사용하여 대기열 백로그의 크기 및 메시지 처리 속도를 결정할 수 있습니다. 메시지가 처리되기 전에 삭제되는 것을 방지하려면 잠재적 독약 메시지를 차단하도록 DLQ(Dead Letter Queue)를 구성하는 것이 좋습니다.

인텐트: 이 경보는 QueueName 대기열에 있는 가장 오래된 메시지의 보존 기간이 너무 긴지 여부를 감지하는 데 사용됩니다. 긴 기간은 메시지가 충분히 빨리 처리되지 않았거나 대기열에 남아 처리할 수 없는 독약 메시지가 있다는 의미일 수 있습니다.

통계: Maximum

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 예상 메시지 처리 시간에 따라 크게 달라집니다. 과거 데이터를 사용하여 평균 메시지 처리 시간을 계산한 다음, 임곗값을 대기열의 소비자가 예상한 최대 SQS 메시지 처리 시간보다 50% 더 높게 설정할 수 있습니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

ApproximateNumberOfMessagesNotVisible

측정 기준: QueueName

경보 설명: 이 경보는 QueueName와 관련하여 많은 수의 처리 중 메시지를 감지하는 데 도움이 됩니다. 문제 해결을 위해 메시지 백로그 감소를 확인하세요.

인텐트: 이 경보는 대기열에 있는 많은 수의 처리 중인 메시지를 탐지하는 데 사용됩니다. 소비자가 가시성 제한 시간 내에 메시지를 삭제하지 않으면 대기열이 폴링될 때 메시지가 대기열에 다시 나타납니다. FIFO 대기열의 경우 처리 중 메시지가 최대 20,000개까지 있을 수 있습니다. 이 할당량에 도달해도 SQS는 오류 메시지를 반환하지 않습니다. FIFO 대기열은 처음 2만 개의 메시지를 검토하여 사용 가능한 메시지 그룹을 결정합니다. 즉, 단일 메시지 그룹에 메시지 백로그가 있는 경우 백로그에서 메시지를 성공적으로 사용할 때까지 나중에 대기열로 전송된 다른 메시지 그룹의 메시지를 사용할 수 없습니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 이 경보의 권장 임곗값은 처리 중인 예상 메시지 수에 따라 크게 달라집니다. 과거 데이터를 사용하여 처리 중인 최대 예상 메시지 수를 계산하고 임곗값을 이 값의 50% 이상으로 설정할 수 있습니다. 대기열의 소비자가 대기열에서 메시지를 처리하지만 삭제하지는 않는 경우 이 수가 갑자기 증가합니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

ApproximateNumberOfMessagesVisible

측정 기준: QueueName

경보 설명: 이 경보는 메시지 대기열 백로그가 예상보다 커지는지 감시하며 소비자가 너무 느리거나 소비자가 충분하지 않음을 나타냅니다. 이 경보가 ALARM 상태가 되면 소비자 수를 늘리거나 소비자 속도를 높이는 것을 고려해 보세요.

인텐트: 이 경보는 활성 대기열의 메시지 수가 너무 많아 소비자가 메시지를 처리하는 속도가 느리거나 메시지를 처리할 소비자가 충분하지 않은지 여부를 감지하는 데 사용됩니다.

통계: Average

권장 임곗값: 상황에 따라 다름

임곗값 정당화: 표시되는 메시지 수가 예상보다 많으면 소비자가 메시지를 예상 속도로 처리하지 못하고 있음을 나타냅니다. 이 임곗값을 설정할 때 과거 데이터를 고려해야 합니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

NumberOfMessagesSent

측정 기준: QueueName

경보 설명: 이 경보는 QueueName와 관련하여 생산자로부터 전송되는 메시지가 없는지 감지하는 데 도움이 됩니다. 문제 해결을 위해 생산자가 메시지를 보내지 않는 이유를 확인합니다.

인텐트: 이 경보는 생산자가 메시지 전송을 중단하는 시점을 감지하는 데 사용됩니다.

통계: Sum

권장 임곗값: 0.0

임곗값 정당화: 전송된 메시지 수가 0인 경우 생산자는 메시지를 보내지 않습니다. 이 대기열의 TPS가 낮으면 그에 따라 EvaluationPeriods 수를 늘립니다.

기간: 60

경보를 보낼 데이터 포인트: 15

평가 기간: 15

비교 연산자: LESS_THAN_OR_EQUAL_TO_THRESHOLD

AWS VPN

TunnelState

측정 기준: VpnId

경보 설명: 이 경보는 하나 이상의 터널 상태가 DOWN인지 파악하는 데 도움이 됩니다. 문제 해결을 위해 VPN 터널 문제 해결을 참조하세요.

인텐트: 이 경보는 하나 이상의 터널이 이 VPN의 DOWN 상태에 있는지 감지하여 영향을 받는 VPN 문제를 해결하는 데 사용됩니다. 터널이 하나만 구성된 네트워크의 경우 이 경보는 항상 ALARM 상태에 있습니다.

통계: Minimum

권장 임곗값: 1.0

임곗값 정당화: 값이 1보다 작으면 하나 이상의 터널이 DOWN 상태임을 나타냅니다.

기간: 300

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: LESS_THAN_THRESHOLD

TunnelState

측정 기준: TunnelIpAddress

경보 설명: 이 경보는 터널의 상태가 DOWN인지 파악하는 데 도움이 됩니다. 문제 해결을 위해 VPN 터널 문제 해결을 참조하세요.

인텐트: 이 경보는 터널이 DOWN 상태에 있는지 감지하여 영향을 받는 VPN 문제를 해결하는 데 사용됩니다. 터널이 하나만 구성된 네트워크의 경우 이 경보는 항상 ALARM 상태에 있습니다.

통계: Minimum

권장 임곗값: 1.0

임곗값 정당화: 값이 1보다 작으면 터널이 DOWN 상태임을 나타냅니다.

기간: 300

경보를 보낼 데이터 포인트: 3

평가 기간: 3

비교 연산자: LESS_THAN_THRESHOLD

지표에 대한 경보

다음 섹션의 단계에서는 지표에서 CloudWatch 경보를 생성하는 방법에 대해 설명합니다.

정적 임곗값을 기반으로 CloudWatch 경보 생성

감시할 경보에 대한 CloudWatch 지표와 해당 지표에 대한 임곗값을 선택합니다. 지표가 지정된 수의 평가 기간에 대한 임곗값을 위반할 경우 경보가 ALARM 상태가 됩니다.

CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정에 경보를 생성하는 경우 이 모니터링 계정에 연결된 소스 계정의 지표를 관찰하도록 경보를 설정할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

단일 지표를 기반으로 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms) 모든 경보(All Alarms)를 선택합니다.
3. 경보 생성(Create alarm)을 선택하세요.
4. 지표 선택을 선택합니다.
5. 다음 중 하나를 수행하십시오.
   - 원하는 지표가 포함된 서비스 네임스페이스를 선택합니다. 옵션이 나타날 때 계속해서 옵션을 선택하면 선택 범위가 좁아집니다. 지표 목록이 표시되면 원하는 지표 옆에 있는 확인란을 선택합니다.
   - 검색 상자에 지표 이름, 계정 ID, 계정 레이블, 차원 또는 리소스 ID를 입력합니다. 그런 다음 결과 중 하나를 선택하고 지표 목록이 표시될 때까지 계속 진행합니다. 원하는 지표 옆의 확인란을 선택합니다.

6. 그래프로 표시된 지표 탭을 선택합니다.

   a. 통계에서 통계 또는 사전 정의된 백분위수 중 하나를 선택하거나, 사용자 지정 백분위수(예: **p95.45**)를 지정합니다.

   b. 기간에서 경보에 대한 평가 기간을 선택합니다. 경보를 평가할 때 각 기간이 하나의 데이터 포인트로 집계됩니다.

      또한 경보를 생성할 때 Y축 범례를 왼쪽 또는 오른쪽에 표시할지 여부를 선택할 수도 있습니다. 이러한 기본 설정은 경보를 생성하는 동안에만 사용됩니다.

   c. 지표 선택을 선택하세요.

      선택한 지표 및 통계에 대한 그래프와 기타 정보가 표시된 Specify metric and conditions(지표 및 조건 지정) 페이지가 나타납니다.

7. 조건에서 다음을 지정합니다.

   a. 지표가 **### ##** 항상에서 지표가 임곗값보다 크거나, 작거나, 같아야 하는지 여부를 지정합니다. than...에서 임곗값을 지정합니다.

   b. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

      N 중 M 경보를 생성하려면 두 번째 값에 지정한 값보다 낮은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가 단원을 참조하세요.

   c. 누락 데이터 처리에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하세요.

   d. 경보가 모니터링된 통계 값으로 백분위수를 사용하는 경우에는 샘플이 부족한 백분위수 상자가 표시됩니다. 샘플 비율이 낮은 사례를 평가 또는 무시할지 여부를 선택할 때 이 상자를 사용합니다. ignore (maintain alarm state)(무시(경보 상태 유지))를 선택하면 샘플 크기가 너무 작을 때 현재 경보 상태가 항상 유지됩니다. 자세한 내용은 백분위수 기반 CloudWatch 경보 및 데이터 샘플 부족 단원을 참조하세요.

8. 다음(Next)을 선택합니다.

9. 알림(Notification)에서 경보가 ALARM 상태, OK 상태 또는 INSUFFICIENT_DATA 상태일 때 알릴 SNS 주제를 선택합니다.

   경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

CloudWatch 크로스 계정 관측성에서 여러 AWS 계정으로 알림을 전송하도록 선택할 수 있습니다. 예를 들어, 모니터링 계정과 소스 계정 모두로 알림을 전송할 수 있습니다.

경보에서 알림을 보내지 않게 하려면 제거를 선택합니다.

10. 경보가 Auto Scaling, EC2, Lambda 또는 Systems Manager 작업을 수행하도록 하려면 해당 버튼을 선택하고 경보 상태와 수행할 작업을 선택합니다. 경보는 ALARM 상태가 될 때만 Systems Manager 작업을 수행할 수 있습니다. Systems Manager 작업에 대한 자세한 내용은 경보에서 OpsItem을 생성하도록 CloudWatch 구성 및 인시던트 생성을 참조하세요.

> **Note**
>
> SSM Incident Manager 작업을 수행하는 경보를 생성하려면 특정 권한이 있어야 합니다. 자세한 내용은 AWS Systems Manager Incident Manager의 자격 증명 기반 정책 예 단원을 참조하세요.

11. 마친 후에는 다음을 선택합니다.
12. 경보 이름 및 설명을 입력합니다. 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다. 그리고 다음(Next)을 선택합니다.
13. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

대시보드에 경보를 추가할 수도 있습니다. 자세한 내용은 CloudWatch 대시보드에서 경보 위젯 추가 또는 제거 단원을 참조하십시오.

지표 수학 표현식을 기반으로 CloudWatch 경보 생성

지표 수학 표현식을 기반으로 경보를 생성하려면 표현식에 사용할 CloudWatch 지표를 하나 이상 선택합니다. 그런 다음 표현식, 임곗값 및 평가 기간을 지정합니다.

SEARCH 표현식을 기반으로 경보를 생성할 수 없습니다. 검색 표현식은 여러 시계열을 반환하고 수학 표현식 기반 경보는 하나의 시계열만 관찰할 수 있기 때문입니다.

지표 수학 표현식을 기반으로 경보 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택한 다음 모든 경보(All alarms)를 선택합니다.

3. 경보 생성(Create alarm)을 선택하십시오.
4. 지표 선택(Select Metric)을 선택하고 다음 작업 중 하나를 수행합니다.
   - AWS 네임스페이스(namespaces) 드롭다운 또는 사용자 지정 네임스페이스(Custom namespaces) 드롭다운에서 네임스페이스를 선택합니다. 네임스페이스를 선택한 후에는 올바른 지표 옆의 확인란을 선택하는 지표 목록이 나타날 때까지 옵션을 계속 선택합니다.
   - 검색 상자를 사용하여 지표, 계정 ID, 차원 또는 리소스 ID를 찾습니다. 지표, 측정기준 또는 리소스 ID를 입력한 후 올바른 지표 옆의 확인란을 선택하는 지표 목록이 나타날 때까지 옵션을 계속 선택합니다.
5. (선택 사항) 지표 수학 표현식에 다른 지표를 추가하려면 검색 상자를 사용하여 특정 지표를 찾을 수 있습니다. 지표 수학 표현식에 지표를 10개까지 추가할 수 있습니다.
6. 그래프로 표시된 지표(Graphed metrics) 탭을 선택합니다. 전에 추가한 지표 각각에 대해 다음 작업을 수행합니다.

   a. 통계(Statistic) 열에서 드롭다운 메뉴를 선택합니다. 드롭다운 메뉴에서 미리 정의된 통계 또는 백분위수 중 하나를 선택합니다. 드롭다운 메뉴의 검색 상자를 사용하여 사용자 지정 백분위수를 지정합니다.

   b. 기간(Period) 열에서 드롭다운 메뉴를 선택합니다. 드롭다운 메뉴에서 미리 정의된 평가 기간 중 하나를 선택합니다.

      경보를 생성하는 동안 Y축 범례를 그래프의 왼쪽 또는 오른쪽에 표시할지 여부를 지정할 수도 있습니다.

   > **Note**
   >
   > CloudWatch가 경보를 평가하면 기간이 단일 데이터 포인트로 집계됩니다.

7. 수학 추가(Add math) 드롭다운을 선택한 다음 미리 정의된 지표 수학 표현식 목록에서 빈 표현식으로 시작(Start with an empty expression)을 선택합니다.

   빈 표현식으로 시작(Start with an empty expression)을 선택하면 수학 표현식을 적용하거나 편집할 수 있는 수학 표현식 상자가 나타납니다.
8. 수학 표현식 상자에 수학 표현식을 입력한 다음 적용(Apply)을 선택합니다.

   적용(Apply)을 선택하면 ID 열이 레이블(Label) 열 옆에 나타납니다.

   현재 수학 표현식 공식의 일부로 지표 또는 다른 지표 수학 표현식의 결과를 사용하려면 ID 열 아래 표시된 값을 사용합니다. ID의 값을 변경하려면 현재 값 옆에 있는 펜 앤 페이퍼 아이콘을 선택

합니다. 새 값은 소문자로 시작해야 하며 숫자, 문자, 밑줄 기호를 포함할 수 있습니다. ID의 값을 좀 더 의미 있는 이름으로 변경하면 경보 그래프를 이해하기가 더욱 쉬워집니다.

지표 수학에 사용할 수 있는 함수에 대한 자세한 내용은 지표 수학 구문 및 함수 섹션을 참조하세요.

9. (선택 사항) 새 수학 표현식의 수식에 다른 수학 표현식의 지표 및 결과를 사용해 수학 표현식을 추가합니다.
10. 경보에 사용할 표현식이 있는 경우 페이지에서 다른 모든 표현식 및 지표 왼쪽에 있는 확인란을 선택 취소합니다. 경보에 사용할 표현식 옆의 확인란만 선택해야 합니다. 경보에 사용하려고 선택한 표현식은 단일 시계열을 생성하고, 그래프에 선을 하나만 표시해야 합니다. 그런 다음 지표 선택을 선택합니다.

    선택한 수학 표현식에 대한 그래프와 기타 정보가 표시된 지표 및 조건 지정 페이지가 표시됩니다.
11. 표현식이 **### ##** 항상에서 표현식이 임곗값보다 크거나, 작거나, 같아야 하는지 여부를 지정합니다. than...에서 임곗값을 지정합니다.
12. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

    N 중 M 경보를 생성하려면 두 번째 값에 지정한 값보다 낮은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가 단원을 참조하세요.
13. 누락 데이터 처리에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하세요.
14. 다음(Next)을 선택합니다.
15. 알림(Notification)에서 경보가 ALARM 상태, OK 상태 또는 INSUFFICIENT_DATA 상태일 때 알릴 SNS 주제를 선택합니다.

    경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

    경보에서 알림을 보내지 않게 하려면 제거를 선택합니다.
16. 경보가 Auto Scaling, EC2, Lambda 또는 Systems Manager 작업을 수행하도록 하려면 해당 버튼을 선택하고 경보 상태와 수행할 작업을 선택합니다. Lambda 함수를 경보 작업으로 선택하는 경우 함수 이름 또는 ARN을 지정하고 필요에 따라 함수의 특정 버전을 선택할 수 있습니다.

경보는 ALARM 상태가 될 때만 Systems Manager 작업을 수행할 수 있습니다. Systems Manager 작업에 대한 자세한 내용은 경보에서 OpsItem을 생성하도록 CloudWatch 구성 및 인시던트 생성 단원을 참조하세요.

> **Note**
>
> SSM Incident Manager 작업을 수행하는 경보를 생성하려면 특정 권한이 있어야 합니다. 자세한 내용은 AWS Systems Manager Incident Manager의 자격 증명 기반 정책 예 단원을 참조하세요.

17. 마친 후에는 다음을 선택합니다.
18. 경보 이름 및 설명을 입력합니다. 다음을 선택합니다.

    이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.
19. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

대시보드에 경보를 추가할 수도 있습니다. 자세한 내용은 CloudWatch 대시보드에서 경보 위젯 추가 또는 제거 단원을 참조하십시오.

Metrics Insights 쿼리를 기반으로 CloudWatch 경보 생성

단일 시계열을 반환하는 Metrics Insights 쿼리에 대한 경보를 설정할 수 있습니다. 이는 인프라 또는 애플리케이션 플릿에서 집계된 지표를 관찰하는 동적 경보를 생성하는 데 특히 유용할 수 있습니다. 경보를 한 번 생성하면 플릿에서 리소스가 추가되거나 제거될 때 경보가 조정됩니다. 예를 들어 모든 인스턴스의 CPU 사용률을 관찰하는 경보를 생성할 수 있으며 인스턴스를 추가하거나 제거할 때 경보가 동적으로 조정됩니다.

전체 지침은 Metrics Insights 쿼리에 대한 경보 생성 섹션을 참조하세요.

연결된 데이터 소스를 기반으로 경보 생성

CloudWatch에 없는 데이터 소스의 지표를 감시하는 경보를 생성할 수 있습니다. 이러한 다른 데이터 소스에 대한 연결을 생성하는 방법에 대한 자세한 내용은 다른 데이터 소스의 쿼리 지표 섹션을 참조하세요.

연결한 데이터 소스의 지표에 대한 경보 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표, 모든 지표를 선택합니다.
3. 다중 소스 쿼리 탭을 선택합니다.
4. 데이터 소스에서 사용할 데이터 소스를 선택합니다.
5. 쿼리 작성기가 쿼리가 경보에 사용할 지표를 검색하는 데 필요한 정보를 입력하라는 메시지를 표시합니다. 워크플로는 각 데이터 소스마다 다르며 데이터 소스에 맞게 조정됩니다. 예를 들어, Amazon Managed Service for Prometheus 및 Prometheus 데이터 소스의 경우 쿼리 도우미가 포함된 PromQL 쿼리 편집기 상자가 나타납니다.
6. 쿼리 구성을 마치면 쿼리 그래프로 표시를 선택합니다.
7. 샘플 그래프가 예상과 같으면 경보 생성을 선택합니다.
8. 지표 및 조건 지정 페이지가 나타납니다. 사용 중인 쿼리가 둘 이상의 시계열을 생성하는 경우 페이지 상단에 경고 배너가 표시됩니다. 그럴 경우 집계 함수에서 시계열을 집계하는 데 사용할 함수를 선택합니다.
9. (선택 사항) 경보에 대한 레이블을 추가합니다.
10. ***your-metric-name***이 다음과 같은 경우에 항상...에서 보다 큼, 보다 크거나 같음, 보다 작거나 같음 또는 보다 작음을 선택합니다. :에 임곗값에 대한 숫자를 지정합니다.
11. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

    N개 중 M번째 경보를 생성하려면 두 번째 값의 숫자보다 작은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가 단원을 참조하십시오.
12. 누락 데이터 처리(Missing data treatment)에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하십시오.
13. Next(다음)를 선택합니다.
14. 알림에서 경보가 ALARM, OK 또는 INSUFFICIENT_DATA 상태로 전환될 때 알림을 보낼 Amazon SNS 주제를 지정합니다.

    a. (선택 사항) 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내려면 Add notification(알림 추가)을 선택합니다.

> **Note**
>
> 경보 상태가 되는 경우와 함께 데이터 부족 상태가 되는 경우에도 조치를 취하도록 경보를 설정하는 것이 좋습니다. 데이터 소스에 연결되는 Lambda 함수와 관련된 많은 문제로 인해 경보가 데이터 부족 상태로 전환될 수 있기 때문입니다.

b. (선택 사항) Amazon SNS 알림을 보내지 않으려면 제거를 선택합니다.

15. 경보가 Auto Scaling, EC2, Lambda 또는 Systems Manager 작업을 수행하도록 하려면 해당 버튼을 선택하고 경보 상태와 수행할 작업을 선택합니다. Lambda 함수를 경보 작업으로 선택하는 경우 함수 이름 또는 ARN을 지정하고 필요에 따라 함수의 특정 버전을 선택할 수 있습니다.

    경보는 ALARM 상태가 될 때만 Systems Manager 작업을 수행할 수 있습니다. Systems Manager 작업에 대한 자세한 내용은 경보에서 OpsItem을 생성하도록 CloudWatch 구성 및 인시던트 생성 단원을 참조하세요.

    > **Note**
    >
    > SSM Incident Manager 작업을 수행하는 경보를 생성하려면 특정 권한이 있어야 합니다. 자세한 내용은 AWS Systems Manager Incident Manager의 자격 증명 기반 정책 예 단원을 참조하세요.

16. 다음을 선택합니다.
17. 이름 및 설명에서 경보의 이름과 설명을 입력하고 다음을 선택합니다. 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.

    > **Tip**
    >
    > 경보 이름에는 UTF-8 문자만 포함되어야 합니다. ASCII 제어 문자를 포함할 수 없습니다.

18. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

연결된 데이터 소스의 경보에 대한 세부 정보

- CloudWatch는 경보를 평가할 때 경보 기간이 1분 이상이더라도 1분마다 평가를 수행합니다. 경보가 작동하려면 Lambda 함수가 기간 길이의 배수뿐만 아니라 임의의 분에 시작하는 타임스탬프 목록을 반환할 수 있어야 합니다. 이러한 타임스탬프는 한 기간 길이 간격으로 떨어져 있어야 합니다.

  따라서 Lambda에서 쿼리한 데이터 소스가 기간 길이의 배수인 타임스탬프만 반환할 수 있는 경우 함수는 `GetMetricData` 요청에서 예상하는 타임스탬프와 일치하도록 가져온 데이터를 '다시 샘플링'해야 합니다.

  예를 들어, 기간이 5분인 경보는 매번 1분씩 이동하는 5분 기간을 사용하여 1분마다 평가됩니다. 이 경우

  - 12:15:00의 경보 평가에 대해 CloudWatch는 타임스탬프가 `12:00:00`, `12:05:00` 및 `12:10:00`인 데이터 포인트를 예상합니다.
  - 그런 다음 12:16:00의 경보 평가에 대해 CloudWatch는 타임스탬프가 `12:01:00`, `12:06:00` 및 `12:11:00`인 데이터 포인트를 예상합니다.
- CloudWatch가 경보를 평가할 때 Lambda 함수에서 반환한 데이터 포인트 중 예상 타임스탬프와 일치하지 않는 모든 데이터 포인트는 삭제되고 나머지 예상 데이터 포인트를 사용하여 경보가 평가됩니다. 예를 들어, 경보가 `12:15:00`에 평가되면 CloudWatch는 타임스탬프가 `12:00:00`, `12:05:00` 및 `12:10:00`인 데이터를 예상합니다. 타임스탬프가 `12:00:00`, `12:05:00`, `12:06:00` 및 `12:10:00`인 데이터를 수신하면 `12:06:00`의 데이터가 삭제되고 CloudWatch는 다른 타임스탬프를 사용하여 경보를 평가합니다.

  그런 다음 `12:16:00`의 다음 평가에 대해 CloudWatch는 타임스탬프가 `12:01:00`, `12:06:00` 및 `12:11:00`인 데이터를 예상합니다. 타임스탬프가 `12:00:00`, `12:05:00` 및 `12:10:00`인 데이터만 있는 경우 12:16:00에 이러한 데이터 포인트가 모두 무시되고 누락된 데이터를 처리하기 위해 경보를 지정한 방법에 따라 알람이 해당 상태로 전환됩니다. 자세한 내용은 경보 평가 단원을 참조하십시오.
- `INSUFFICIENT_DATA` 상태로 전환될 때 조치를 취하도록 이러한 경보를 생성하는 것이 좋습니다. 여러 Lambda 함수 실패 사용 사례에서는 누락된 데이터를 처리하기 위해 경보를 설정한 방식에 관계없이 경보가 `INSUFFICIENT_DATA`로 전환되기 때문입니다.
- Lambda 함수가 오류를 반환하거나 부분 데이터를 반환하는 경우
  - Lambda 함수를 호출하는 데 권한 문제가 있는 경우, 생성 시 누락된 데이터를 처리하도록 경보를 지정한 방법에 따라 경보에서 누락된 데이터 전환이 발생하기 시작합니다.
  - Lambda 함수가 `'StatusCode' = 'PartialData'`를 반환하면 경보 평가가 실패하고 세 번의 시도 후에 경보가 `INSUFFICIENT_DATA`로 전환됩니다.

  - Lambda 함수에서 발생하는 다른 오류로 인해 경보가 INSUFFICIENT_DATA로 전환됩니다.
- Lambda 함수에서 요청한 지표에 약간의 지연이 발생하여 마지막 데이터 포인트가 항상 누락되는 경우 해결 방법을 사용해야 합니다. N개 중 M개의 경보를 생성하거나 경보 평가 기간을 늘릴 수 있습니다. M개 중 N개의 경보에 대한 자세한 내용은 경보 평가 섹션을 참조하세요.

이상 탐지를 기반으로 CloudWatch 경보 생성

과거 지표 데이터를 분석하고 예상 값의 모델을 생성하는 CloudWatch 이상 탐지를 기반으로 경보를 생성할 수 있습니다. 기댓값은 지표의 일반적인 시간별, 일별, 주별 패턴을 고려합니다.

이상 탐지 임곗값에 대한 값을 설정합니다. 그러면 CloudWatch는 모델과 함께 이 임곗값을 사용하여 지표 값의 '정상' 범위를 결정합니다. 임곗값에 대한 값이 클수록 '정상' 값의 밴드가 더 두꺼워집니다.

지표 값이 기댓값 밴드 이상이거나 이하일 때, 아니면 두 경우 모두 경보가 트리거되도록 설정할 수 있습니다.

단일 지표 및 지표 수학 표현식의 출력에 대한 이상 탐지 경보를 생성할 수도 있습니다. 이러한 표현식을 사용하여 이상 탐지 밴드를 시각화하는 그래프를 만들 수 있습니다.

CloudWatch 교차 계정 관찰성을 위해 모니터링 계정으로 설정된 계정에서 모니터링 계정의 지표뿐만 아니라 소스 계정의 지표에도 이상 탐지기를 만들 수 있습니다.

자세한 내용은 CloudWatch 이상 탐지 사용 단원을 참조하십시오.

> **Note**
>
> 시각화 목적으로 지표 콘솔의 지표에 이상 탐지를 이미 사용 중인데 해당 지표에 이상 탐지 경보를 생성한 경우, 경보에 설정한 임곗값은 시각화를 위해 이미 설정한 임곗값을 변경하지 않습니다. 자세한 내용은 그래프 생성 단원을 참조하십시오.

이상 탐지에 기반하여 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms) 모든 경보(All Alarms)를 선택합니다.
3. 경보 생성(Create alarm)을 선택하세요.
4. 지표 선택을 선택합니다.

5. 다음 중 하나를 수행하십시오.
   - 메트릭이 포함된 서비스 네임스페이스를 선택한 다음 계속하여 나타나는 옵션을 선택하여 옵션 범위를 좁힙니다. 지표 목록이 표시되면 원하는 지표 옆에 있는 확인란을 선택합니다.
   - 검색 상자에 지표 이름, 차원 또는 리소스 ID를 입력합니다. 그런 다음 결과 중 하나를 선택하고 지표 목록이 표시될 때까지 계속 진행합니다. 지표 옆의 확인란을 선택합니다.
6. 그래프로 표시된 지표를 선택합니다.

   a. (선택 사항) 통계에서 드롭다운을 선택한 후 미리 정의된 통계 또는 백분위수 중 하나를 선택합니다. 드롭다운 메뉴의 검색 상자를 사용하여 **p95.45**와/과 같은 사용자 지정 백분위수를 지정합니다.

   b. (선택 사항) 기간에서 드롭다운을 선택한 후 미리 정의된 평가 기간 중 하나를 선택합니다.

   > **Note**
   >
   > CloudWatch가 경보를 평가하면 기간이 단일 데이터 포인트로 조정됩니다. 이상 탐지 경보의 경우, 값은 1분 이상이어야 합니다.

7. 다음을 선택합니다.
8. 조건에서 다음을 지정합니다.

   a. Anomaly Detection(이상 탐지)를 선택합니다.

   이 지표 및 통계에 대한 모델이 이미 있는 경우 CloudWatch는 화면 상단의 그래프에 이상 탐지 밴드 미리보기를 표시합니다. 경보를 생성한 후 그래프에 실제 이상 탐지 밴드가 표시되는 데 최대 15분이 소요될 수 있습니다. 그 전에 표시되는 밴드는 이상 탐지 밴드의 근사치입니다.

   > **Tip**
   >
   > 화면 상단에 더 긴 기간의 그래프를 보려면 페이지 오른쪽 위에 있는 편집을 선택합니다.

   이 지표 및 통계에 대한 모델이 아직 없는 경우 경보 생성을 완료하면 CloudWatch에서는 이상 탐지 밴드를 생성합니다. 새 모델의 경우 그래프에 실제 이상 탐지 밴드가 표시되는 데 최대 3시간이 소요될 수 있습니다. 새 모델을 훈련하는 데 최대 2주가 소요될 수 있는데, 그래야 이상 감지 대역이 더 정확한 기대값을 표시할 수 있습니다.

b. **##**가 다음인 경우 항상(Whenever metric is)에서 경보를 트리거할 시기를 지정합니다. (예: 지표가 밴드보다 크거나, 작거나, 밴드 외부(어느 방향이든)일 때)

c. 이상 탐지 임곗값에서 이상 탐지 임곗값에 사용할 숫자를 선택합니다. 숫자가 클수록 지표의 변화에 더 잘 대응할 수 있는 "정상" 값의 두꺼운 밴드가 생성됩니다. 숫자가 작을수록 지표 편차가 더 작은 ALARM 상태로 변하는 얇은 밴드가 생성됩니다. 이 숫자는 정수일 필요는 없습니다.

d. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

   N개 중 M번째 경보를 생성하려면 두 번째 값의 숫자보다 작은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가 단원을 참조하십시오.

e. 누락 데이터 처리(Missing data treatment)에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하십시오.

f. 경보가 모니터링된 통계 값으로 백분위수를 사용하는 경우에는 샘플이 부족한 백분위수 상자가 표시됩니다. 샘플 비율이 낮은 사례를 평가 또는 무시할지 여부를 선택할 때 이 상자를 사용합니다. 무시(경보 상태 유지)(Ignore (maintain alarm state))를 선택하면 샘플 크기가 너무 작을 때 현재 경보 상태가 항상 유지됩니다. 자세한 내용은 백분위수 기반 CloudWatch 경보 및 데이터 샘플 부족 단원을 참조하세요.

9. 다음(Next)을 선택합니다.

10. 알림(Notification)에서 경보가 ALARM 상태, OK 상태 또는 INSUFFICIENT_DATA 상태일 때 알릴 SNS 주제를 선택합니다.

    동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내려면 알림 추가(Add notification)를 선택합니다.

    경보에 알림을 전송하지 않으려면 제거(Remove)를 선택합니다.

11. 상태가 변경될 때 EC2 작업을 수행하거나 Lambda 함수를 간접적으로 호출하도록 경보를 설정하거나 경보 상태가 될 때 Systems Manager OpsItem 또는 인시던트를 생성하도록 설정할 수 있습니다. 이렇게 하려면 해당 버튼을 선택한 다음 경보 상태 및 수행할 작업을 선택합니다.

    Lambda 함수를 경보 작업으로 선택하는 경우 함수 이름 또는 ARN을 지정하고 필요에 따라 함수의 특정 버전을 선택할 수 있습니다.

Systems Manager 작업에 대한 자세한 내용은 경보에서 OpsItem을 생성하도록 CloudWatch 구성 및 인시던트 생성을 참조하세요.

> **Note**
>
> AWS Systems Manager Incident Manager 작업을 수행하는 경보를 생성하려면 특정 권한이 있어야 합니다. 자세한 내용은 AWS Systems Manager Incident Manager의 자격 증명 기반 정책 예 단원을 참조하세요.

12. 다음을 선택합니다.
13. 이름 및 설명에서 경보의 이름과 설명을 입력하고 다음을 선택합니다. 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.

> **Tip**
>
> 경보 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다.

14. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

이상 탐지 모델 수정

경보를 생성한 후 이상 탐지 모델을 수정할 수 있습니다. 모델 생성에 특정 기간이 사용되지 않도록 제외할 수 있습니다. 교육 데이터에서 시스템 중단, 배포 및 휴일과 같은 비정상적인 이벤트를 제외하는 것이 중요합니다. 일광 절약 시간제 변경에 대해 모델을 조정할지 여부도 지정할 수 있습니다.

경보에 대한 이상 탐지 모델을 수정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms) 모든 경보(All Alarms)를 선택합니다.
3. 경보 이름을 선택합니다. 필요하면 검색 상자를 사용하여 경보를 찾습니다.
4. 분석, 지표에서를 선택합니다.
5. 세부 정보 열에서 ANOMALY_DETECTION_BAND, 이상 탐지 모델 편집을 선택합니다.
6. 모델을 생성하는 데 사용되는 기간을 제외하려면 종료 날짜 옆의 달력 아이콘을 선택합니다. 그런 다음, 교육에서 제외할 날짜와 시간을 선택하거나 입력하고 적용(Apply)을 선택합니다.

7. 지표가 일광절약시간제 변화에 민감한 경우, 지표 시간대(Metric timezone) 상자에서 적절한 시간대를 선택합니다.
8. 업데이트를 선택합니다.

이상 탐지 모델 삭제

경보에 대한 이상 탐지를 사용하면 계정에 요금이 발생합니다. 경보에 이상 탐지 모델이 더 이상 필요하지 않은 경우 경보를 먼저 삭제하고 모델을 두 번째로 삭제하는 것이 좋습니다. 이상 탐지 경보가 평가되면 누락된 이상 탐지기가 사용자를 대신하여 생성됩니다. 경보를 삭제하지 않고 모델을 삭제하면 알람이 자동으로 모델을 다시 생성합니다.

경보를 삭제하는 방법

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms), 모든 경보(All Alarms)를 선택합니다.
3. 경보 이름을 선택합니다.
4. 작업, 삭제를 선택합니다.
5. 확인 상자에서 삭제를 선택합니다.

경보에 사용된 이상 탐지 모델 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택한 다음 모든 지표를 선택합니다.
3. Browse(찾아보기)를 선택한 다음 이상 탐지 모델이 포함된 지표를 선택합니다. 검색 상자에서 지표를 검색하거나 옵션을 통해 선택하여 지표를 선택할 수 있습니다.
   - (선택 사항) 원래 인터페이스를 사용하는 경우 All metrics(모든 지표)를 선택한 다음 이상 감지 모델이 포함된 지표를 선택합니다. 검색 상자에서 지표를 검색하거나 옵션을 통해 선택하여 지표를 선택할 수 있습니다.
4. 그래프로 표시된 지표(Graphed metrics)를 선택합니다.
5. Graphed metrics(그래프로 표시된 지표) 탭에서 제거하려는 이상 탐지 모델의 이름을 선택하고 Delete anomaly detection model(이상 탐지 모델 삭제)을 선택합니다.
   - (선택 사항) 원래 인터페이스를 사용하는 경우 Edit model(모델 편집)을 선택합니다. 새 화면으로 이동합니다. 새 화면에서 Delete model(모델 삭제)을 선택한 다음 Delete(삭제)를 선택합니다.

로그에 대한 경보

다음 섹션의 단계에서는 로그에 대한 CloudWatch 경보를 생성하는 방법을 설명합니다.

로그 그룹-지표 필터를 기반으로 CloudWatch 경보 생성

이 섹션의 절차는 로그 그룹 지표 필터를 기반으로 경보를 생성하는 방법을 설명합니다. 지표 필터를 사용하면 데이터가 CloudWatch로 전송될 때 로그 데이터에서 용어와 패턴을 찾을 수 있습니다. 자세한 내용을 알아보려면 Amazon CloudWatch Logs 사용 설명서의 필터를 사용하여 로그 이벤트에서 지표 생성을 참조하세요. 로그 그룹 지표 필터를 기반으로 경보를 생성하려면 먼저 다음 작업을 완료해야 합니다.

- 로그 그룹 생성 자세한 내용을 알아보려면 Amazon CloudWatch Logs 사용 설명서의 로그 그룹 및 로그 스트림 작업을 참조하세요.
- 지표 필터를 생성합니다. 자세한 내용을 알아보려면 Amazon CloudWatch Logs 사용 설명서의 로그 그룹에 대한 지표 필터 생성을 참조하세요.

로그 그룹 지표 필터를 기반으로 경보 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 로그를 선택한 다음, 로그 그룹을 선택합니다.
3. 지표 필터가 포함된 로그 그룹을 선택합니다.
4. Metric filters(지표 필터)를 선택합니다.
5. 지표 필터 탭에서 경보의 기반으로 사용할 지표 필터의 확인란을 선택합니다.
6. 경보 생성(Create alarm)을 선택하십시오.
7. (선택 사항) Metric(지표)에서 Metric name(지표 이름), Statistic(통계) 및 Period(기간)를 편집합니다.
8. 조건에서 다음을 지정합니다.
    a. 임곗값 유형에서 정적 또는 이상 탐지를 선택합니다.
    b. Whenever ***your-metric-name*** is . . .(your-metric-name이 다음과 같은 경우에 항상…)에서 Greater(보다 큼), Greater/Equal(보다 크거나 같음), Lower/Equal(보다 작거나 같음) 또는 Lower(보다 작음)를 선택합니다.
    c. than . . .(:)에 임곗값에 대한 숫자를 지정합니다.
9. 추가 구성을 선택합니다.

a. Data points to alarm(경보를 보낼 데이터 포인트)에서 경보가 ALARM 상태로 전환되도록 트리거하는 데이터 포인트 수를 지정합니다. 일치하는 값을 지정하면 해당 기간이 연속적으로 위반되는 경우 경보가 ALARM 상태가 됩니다. N개 중 M번째 경보를 생성하려면 두 번째 값에 대해 지정한 숫자보다 작은 값을 첫 번째 값에 지정합니다. 자세한 내용은 Amazon CloudWatch 경보 사용을 참조하세요.

b. Missing data treatment(누락된 데이터 처리)에서 경보가 평가될 때 누락된 데이터를 처리하는 방법을 지정하는 옵션을 선택합니다.

10. 다음을 선택합니다.

11. Notification(알림)에서 경보가 ALARM, OK 또는 INSUFFICIENT_DATA 상태일 때 알림을 보낼 Amazon SNS 주제를 지정합니다.

    a. (선택 사항) 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내려면 Add notification(알림 추가)을 선택합니다.

    b. (선택 사항) 알림을 보내지 않으려면 Remove(제거)를 선택합니다.

12. 경보가 Auto Scaling, EC2, Lambda 또는 Systems Manager 작업을 수행하도록 하려면 해당 버튼을 선택하고 경보 상태와 수행할 작업을 선택합니다. Lambda 함수를 경보 작업으로 선택하는 경우 함수 이름 또는 ARN을 지정하고 필요에 따라 함수의 특정 버전을 선택할 수 있습니다.

    경보는 ALARM 상태가 될 때만 Systems Manager 작업을 수행할 수 있습니다. Systems Manager 작업에 대한 자세한 내용은 경보에서 OpsItem을 생성하도록 CloudWatch 구성 및 인시던트 생성 단원을 참조하세요.

    > **Note**
    >
    > SSM Incident Manager 작업을 수행하는 경보를 생성하려면 특정 권한이 있어야 합니다. 자세한 내용은 AWS Systems Manager Incident Manager의 자격 증명 기반 정책 예 단원을 참조하세요.

13. 다음을 선택합니다.

14. 이름 및 설명에서, 경보에 대한 이름과 설명을 입력하세요. 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.

15. Preview and create(미리 보기 및 생성)에서 구성이 올바른지 확인한 다음 Create alarm(경보 생성)을 선택합니다.

경보 결합

CloudWatch를 사용하면 여러 경보를 하나의 복합 경보로 결합하여 전체 애플리케이션 또는 리소스 그룹에 대해 요약되고 집계된 상태 지표를 생성할 수 있습니다. 복합 경보는 다른 경보의 상태를 모니터링하여 상태를 확인하는 경보입니다. 사용자는 Boolean 논리를 사용하여 모니터링되는 경보의 상태를 결합하는 규칙을 정의합니다.

복합 경보를 사용하면 집계된 수준에서만 조치를 실행하므로 경보 노이즈를 줄일 수 있습니다. 예를 들어, 웹 서버와 관련된 경보가 트리거되는 경우 복합 경보를 생성하여 웹 서버 팀에 알림을 보낼 수 있습니다. 이러한 경보 중 하나라도 ALARM 상태로 전환되면 복합 경보는 스스로 ALARM 상태가 되어 팀에 알림을 보냅니다. 웹 서버와 관련된 다른 경보가 ALARM 상태로 전환되더라도 복합 경보가 이미 기존 상황에 대해 알렸기 때문에 팀에 새 알림이 과도하게 오지 않습니다.

또한 복합 경보를 사용하여 복잡한 경보 조건을 생성하고 여러 조건이 충족될 때만 조치를 취할 수 있습니다. 예를 들어, CPU 경보와 메모리 경보를 결합하여 CPU와 메모리 경보가 모두 트리거된 경우에만 팀에 알리는 복합 경보를 생성하 수 있습니다.

복합 경보 사용

복합 경보를 사용하는 경우 두 가지 옵션이 있습니다.

- 복합 경보 수준에서만 수행할 작업을 구성하고, 조치가 없는 기본 모니터링 경보를 만듭니다
- 복합 경보 수준에서 다양한 작업의 조합을 구성합니다. 예를 들어, 복합 경보 작업에서는 문제가 광범위하게 발생하는 경우 다른 팀을 참여시킬 수 있습니다.

복합 경보는 다음과 같은 작업만 수행할 수 있습니다.

- Amazon SNS 주제 알림
- Lambda 함수 간접 호출
- Systems Manager Ops Center에 OpsItem 생성
- Systems Manager Incident Manager에 인시던트 생성

> **Note**
>
> 복합 경보의 모든 기본 경보는 복합 경보와 동일한 계정 및 동일한 리전에 있어야 합니다. 그러나 CloudWatch 크로스 계정 관측성 모니터링 계정에서 복합 경보를 설정하면 기본 경보가 다른 소스 계정과 모니터링 계정 자체에서 지표를 관찰할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

단일 복합 경보로 100개의 기본 경보를 모니터링할 수 있고, 150개의 복합 경보로 단일 기본 경보를 모니터링할 수 있습니다.

규칙 표현식

모든 복합 경보에는 규칙 표현식이 포함됩니다. 규칙 표현식은 모니터링하고 상태를 확인할 다른 경보를 복합 경보에 알려줍니다. 규칙 표현식은 지표 경보 및 복합 경보를 참조할 수 있습니다. 규칙 표현식에서 경보를 참조할 경우, 다음 세 가지 상태 중 경보가 어떤 상태로 전환될지를 결정하는 함수를 경보에 지정합니다.

- 경보

  경보가 ALARM 상태인 경우 ALARM ("alarm-name or alarm-ARN")이 TRUE입니다.
- 정상

  경보가 OK 상태인 경우 OK ("alarm-name or alarm-ARN")가 TRUE입니다.
- INSUFFICIENT_DATA

  명명된 경보가 INSUFFICIENT_DATA 상태인 경우 INSUFFICIENT_DATA ("alarm-name or alarm-ARN")가 TRUE입니다.

> **Note**
>
> TRUE는 항상 TRUE로 평가되고 FALSE는 항상 FALSE로 평가됩니다.

예제 표현식

요청 파라미터 `AlarmRule`은 논리 연산자 `AND`, `OR` 및 `NOT`을 지원하므로 여러 함수를 단일 표현식으로 결합할 수 있습니다. 다음 예제 표현식은 복합 경보의 기본 경보를 구성하는 방법을 보여줍니다.

- `ALARM(CPUUtilizationTooHigh) AND ALARM(DiskReadOpsTooHigh)`

  이 표현식은 `CPUUtilizationTooHigh`와 `DiskReadOpsTooHigh`가 `ALARM` 상태인 경우에만 복합 경보를 `ALARM` 상태로 전환하도록 지정합니다.
- `ALARM(CPUUtilizationTooHigh) AND NOT ALARM(DeploymentInProgress)`

이 표현식은 CPUUtilizationTooHigh가 ALARM 상태이고 DeploymentInProgress가 ALARM 상태인 경우에만 복합 경보를 ALARM 상태로 전환하도록 지정합니다. 배포 기간 동안 경보 노이즈를 줄이는 복합 경보의 예입니다.

- `(ALARM(CPUUtilizationTooHigh) OR ALARM(DiskReadOpsTooHigh)) AND OK(NetworkOutTooHigh)`

  이 표현식은 (ALARM(CPUUtilizationTooHigh) 또는 (DiskReadOpsTooHigh)가 ALARM 상태이고 (NetworkOutTooHigh)가 OK상태인 경우에만 복합 경보를 ALARM 상태로 전환하도록 지정합니다. 네트워크 문제가 발생한 동안 기본 경보 중 하나가 ALARM 상태가 아닐 때 알림을 보내지 않음으로써 경보 노이즈를 줄이는 복합 경보의 예입니다.

주제

복합 경보 생성

이 섹션의 단계에서는 CloudWatch 콘솔을 사용하여 복합 경보를 생성하는 방법을 설명합니다. API 또는 AWS CLI를 사용하여 복합 경보를 생성할 수도 있습니다. 자세한 내용은 PutCompositeAlarm 또는 put-composite-alarm을 참조하세요.

복합 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택한 다음 모든 경보(All alarms)를 선택합니다.
3. 경보 목록에서, 규칙 표현식에서 참조하려는 각 기존 경보 옆에 있는 확인란을 선택한 다음 Create composite alarm(복합 경보 생성)을 선택합니다.
4. Specify composite alarm conditions(복합 경보 조건 지정)에서 새 복합 경보에 대한 규칙 표현식을 지정합니다.

> **Note**
>
> 경보 목록에서 선택한 경보가 Conditions(조건) 상자에 자동으로 나열됩니다. 기본적으로, 각 경보에 대해 ALARM 함수가 지정되어 있으며, 각 경보는 논리 연산자 OR로 결합됩니다.

다음 하위 단계를 사용하여 규칙 표현식을 수정할 수 있습니다.

a. 각 경보의 필수 상태를 ALARM에서 OK 또는 INSUFFICENT_DATA로 변경할 수 있습니다.

b. 규칙 표현식의 논리 연산자를 OR에서 AND 또는 NOT으로 변경할 수 있으며, 괄호를 추가하여 함수를 그룹화할 수 있습니다.

c. 규칙 표현식에 다른 경보를 포함하거나 규칙 표현식에서 경보를 삭제할 수 있습니다.

예: 조건이 있는 규칙 표현식

```
(ALARM("CPUUtilizationTooHigh") OR
ALARM("DiskReadOpsTooHigh")) AND
OK("NetworkOutTooHigh")
```

이 예제 규칙 표현식에서는 ALARM("CPUUtilizationTooHigh" 또는 "DiskReadOpsTooHigh")이 ALARM 상태이고 동시에 OK("NetworkOutTooHigh")가 OK 상태일 때 복합 경보가 ALARM 상태로 전환됩니다.

5. 마친 후에는 다음을 선택합니다.
6. Configure actions(작업 구성)에서 다음을 선택할 수 있습니다.

Notification(알림)에서

- Select an exisiting SNS topic(기존 SNS 주제 선택), Create a new SNS topic(새 SNS 주제 생성) 또는 Use a topic ARN(주제 ARN 사용)을 선택하여 알림을 수신할 SNS 주제를 정의합니다.
- Add notification(알림 추가)을 선택하여 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보냅니다.
- Remove(제거)를 선택하여 경보가 알림을 보내거나 작업을 수행하지 않게 합니다.

(선택 사항) 상태가 변경될 때 경보가 Lambda 함수를 호출하도록 하려면 Lambda 작업 추가를 선택합니다. 그런 다음, 함수 이름 또는 ARN을 지정하고 필요에 따라 함수의 특정 버전을 선택합니다.

Systems Manager 작업(Systems Manager action)에서

- Add Systems Manager action(Systems Manager 작업 추가)을 선택하여 경보가 ALARM 상태로 전환될 경우 SSM 작업을 수행할 수 있게 합니다.

Systems Manager 작업에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 경보에서 OpsItem을 생성하도록 CloudWatch 구성과 Incident Manager 사용 설명서의 인시던트 생성을 참조하세요. SSM Incident Manager 작업을 수행하는 경보를 생성하려면 올바른 권한이 있어야 합니다. 자세한 내용은 Incident Manager 사용 설명서에서 AWS Systems Manager Incident Manager의 자격 증명 기반 정책 예를 참조하세요.

7. 마친 후에는 다음을 선택합니다.
8. Add name and description(이름 및 설명 추가)에 새 복합 경보의 경보 이름과 선택 사항인 설명을 입력합니다. 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.
9. 마친 후에는 다음을 선택합니다.
10. Preview and create(미리 보기 및 생성)에서 정보를 확인한 다음 Create composite alarm(복합 경보 생성)을 선택합니다.

> **Note**
>
> 하나의 복합 경보와 다른 복합 경보가 서로 종속되는 복합 경보의 주기를 생성할 수 있습니다. 이 시나리오에서는 복합 경보가 더 이상 평가되지 않으며, 서로 종속되어 있으므로 복합 경보를 삭제할 수 없습니다. 복합 경보 간의 종속 주기를 없애는 가장 쉬운 방법은 복합 경보 중 하나에서 `AlarmRule` 함수를 `False`로 변경하는 것입니다.

복합 경보 동작 억제

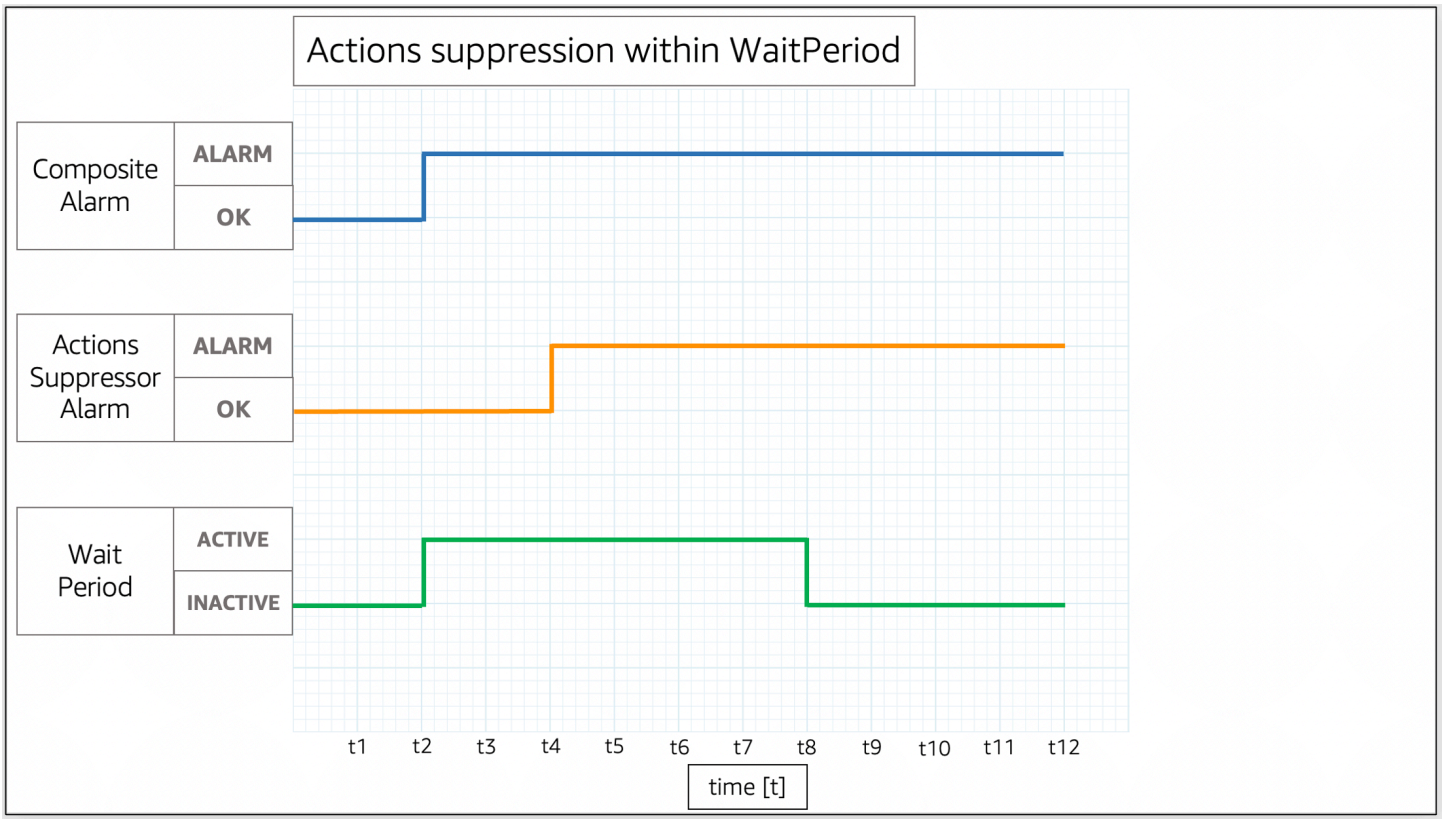
복합 경보를 사용하면 여러 경보의 상태를 종합적으로 볼 수 있으며, 경보가 트리거될 것으로 예상되는 일반적인 상황이 있습니다. 예를 들어, 애플리케이션의 유지 관리 기간 중이거나 진행 중인 사고를 조사하는 경우입니다. 이러한 상황에서는 복합 경보의 동작을 억제하여 원치 않는 알림이나 새로운 인시던트 티켓이 생성되는 것을 방지할 수 있습니다

복합 경보 작업 억제 기능을 사용하면 경보를 억제 경보로 정의할 수 있습니다. 억제 경보는 복합 경보가 작업을 수행하지 않도록 합니다. 예를 들어 지원 리소스의 상태를 나타내는 억제 경보를 지정할 수 있습니다. 지원 리소스가 다운된 경우 억제 경보는 복합 경보가 알림을 보내지 못하게 합니다. 복합 경보 작업 억제 기능은 경보 노이즈를 줄이는 데 도움이 되므로, 경보를 관리하는 데 허비하는 시간을 줄이고 운영에 더 많은 시간을 할애할 수 있습니다.

억제 경보는 복합 경보를 구성할 때 지정합니다. 모든 경보는 억제 경보로 작동할 수 있습니다. 억제 경보의 상태가 OK에서 ALARM으로 변경되면 복합 경보가 더 이상 작업을 수행하지 않습니다. 억제 경보의 상태가 ALARM에서 OK로 변경되면 복합 경보가 작업을 재개합니다.

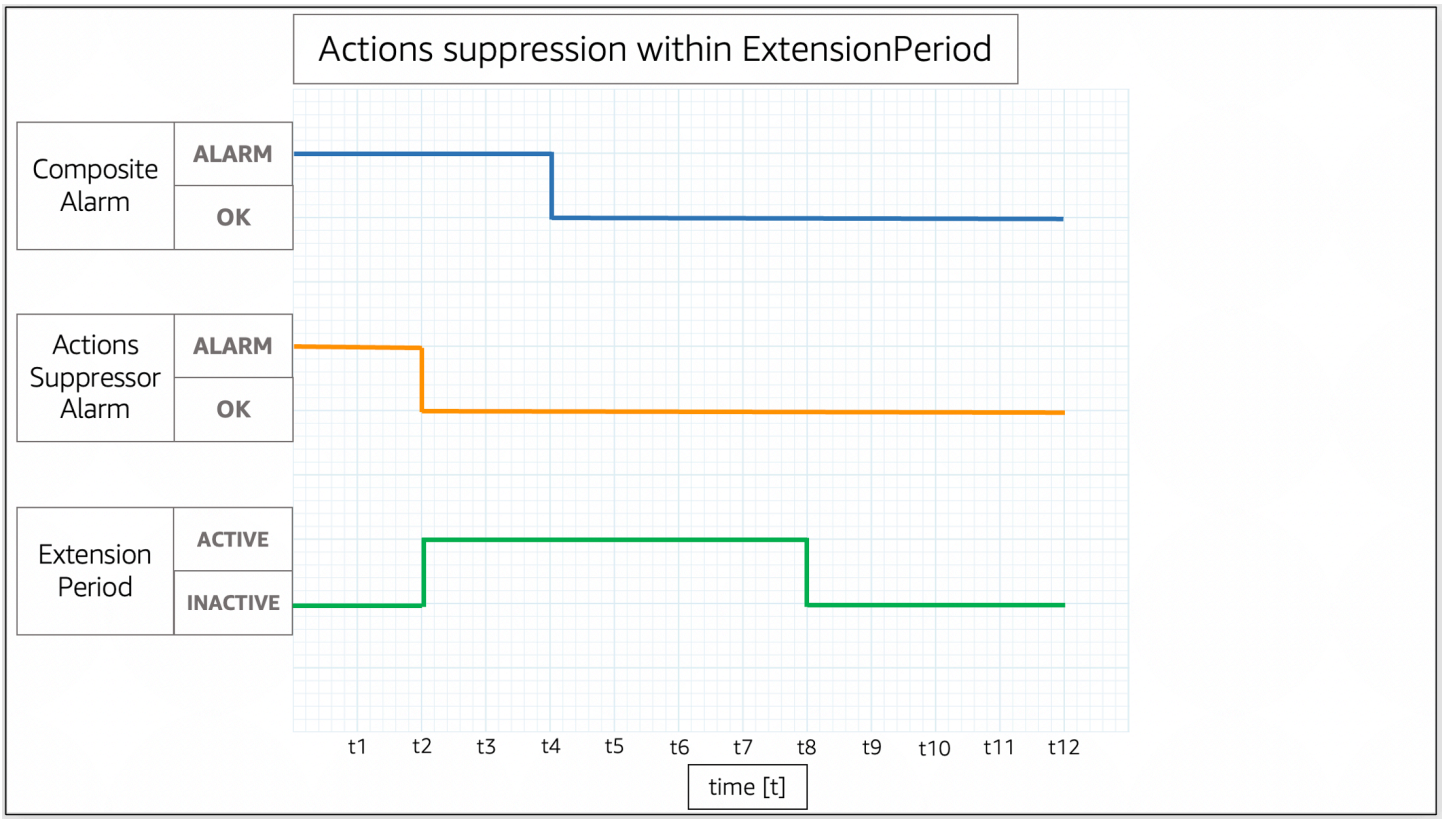
`WaitPeriod` 및 `ExtensionPeriod`

억제 경보를 지정할 때 `WaitPeriod` 및 `ExtensionPeriod` 파라미터를 설정합니다. 이들 파라미터는 억제 경보의 상태가 바뀌는 동안 복합 경보가 예기치 않게 작업을 수행하는 것을 방지합니다. `WaitPeriod`를 사용하여 억제 경보가 OK 상태에서 ALARM 상태로 변경될 때 발생할 수 있는 지연을 상쇄합니다. 예를 들어 억제 경보가 60초 이내에 OK 상태에서 ALARM 상태로 변경되는 경우 `WaitPeriod`를 60초로 설정합니다.



이 이미지에서 복합 경보는 t2에 OK 상태에서 ALARM 상태로 변경됩니다. `WaitPeriod`가 t2에 시작되어 t8에 끝납니다. 이렇게 하면 t8에 `WaitPeriod`가 만료되어 복합 경보의 작업을 억제하기 전까지, 억제 경보가 t4에 상태를 OK에서 ALARM으로 변경할 시간을 확보할 수 있습니다.

`ExtensionPeriod`를 사용하여, 억제 경보가 OK 상태로 변경된 후 복합 경보가 OK 상태로 변경될 때 발생할 수 있는 지연을 상쇄합니다. 예를 들어 억제 경보가 OK 상태로 변경되고 나서 60초 이내에 복합 경보가 OK 상태로 변경되는 경우 `ExtensionPeriod`를 60초 설정합니다.



이 이미지에서 억제 경보는 t2에 ALARM 상태에서 OK 상태로 변경됩니다. ExtensionPeriod가 t2에 시작되어 t8에 끝납니다. 이렇게 하면 t8에 ExtensionPeriod가 만료되기 전에 복합 경보가 ALARM 상태에서 OK 상태로 변경할 시간을 확보할 수 있습니다.

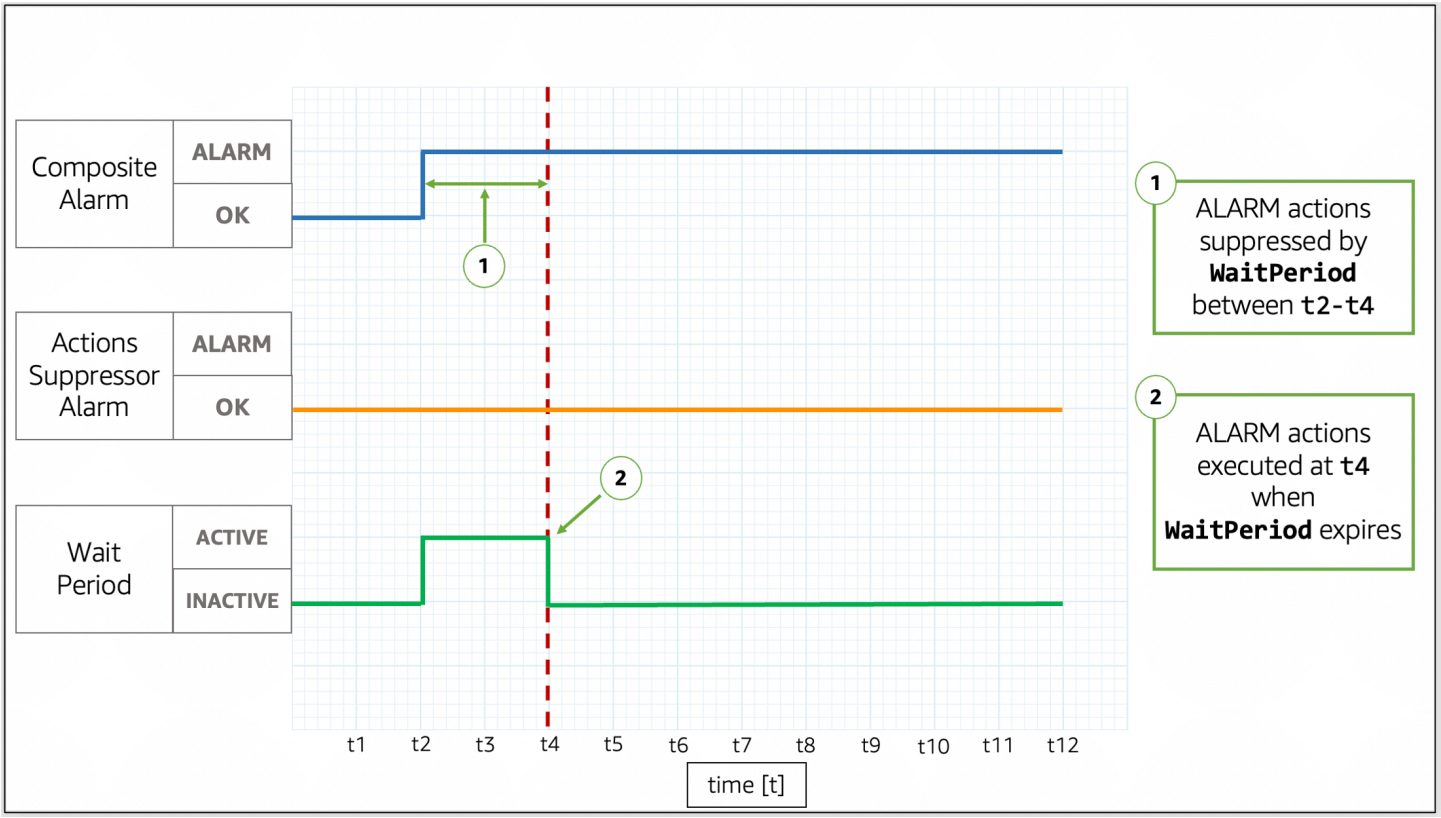
WaitPeriod와 ExtensionPeriod가 활성화되면 복합 경보가 작업을 수행하지 않습니다. ExtensionPeriod와 WaitPeriod가 비활성화되면 복합 경보가 현재 상태에 따라 작업을 수행합니다. CloudWatch는 1분마다 지표 경보를 평가하므로 각 파라미터의 값을 60초로 설정하는 것이 좋습니다. 파라미터는 초 단위의 원하는 정수로 설정할 수 있습니다.

다음 예에서는WaitPeriod과 ExtensionPeriod를 사용하여 복합 경보가 예기치 않게 작업을 수행하지 않도록 방지하는 방법을 자세히 설명합니다.

> **Note**
>
> 다음 예에서 WaitPeriod는 두 시간 단위로 구성되고 ExtensionPeriod는 세 시간 단위로 구성됩니다.

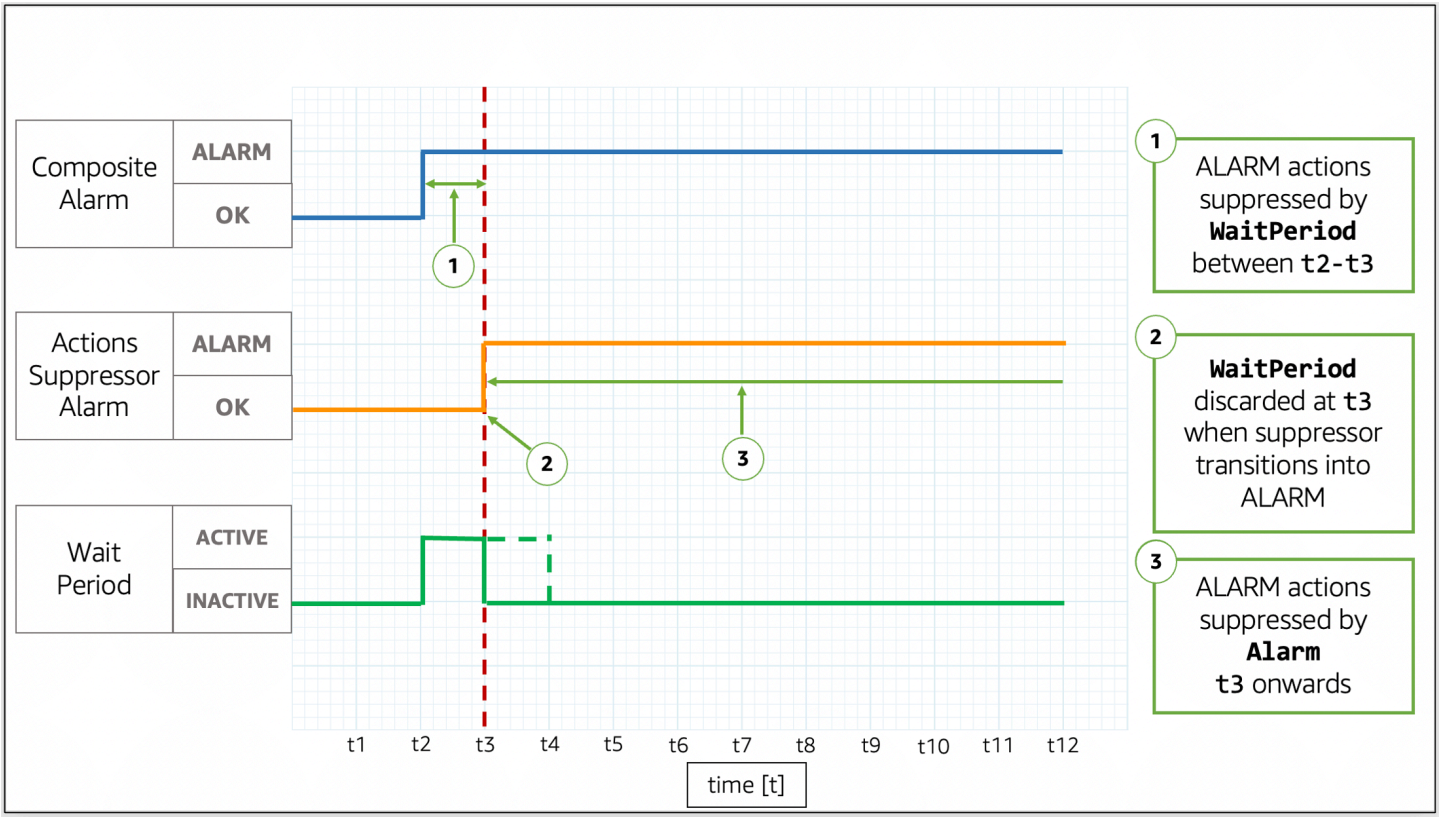
예제

예 1: **`WaitPeriod`** 후에 작업이 억제되지 않음



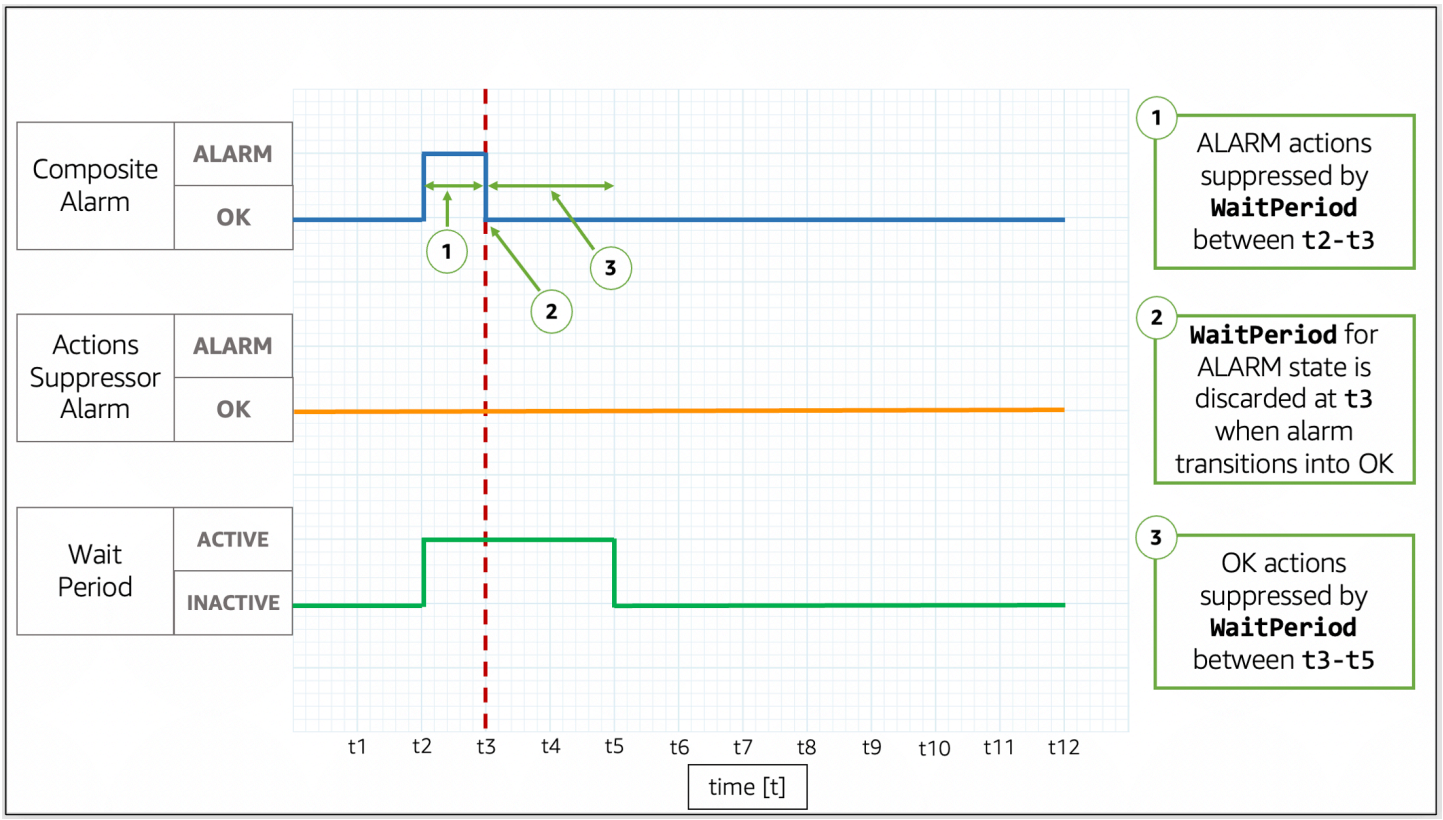
이 이미지에서 복합 경보는 t2에 OK 상태에서 ALARM 상태로 변경됩니다. `WaitPeriod`가 t2에 시작되어 t4에 끝나므로 복합 경보가 작업을 수행하지 못하게 할 수 있습니다. 억제 경보가 아직 OK 상태이므로, t4에 `WaitPeriod`가 만료되고 나면 복합 경보가 작업을 수행합니다.

예 2: **`WaitPeriod`**가 만료되기 전에 경보에 의해 동작이 억제됨



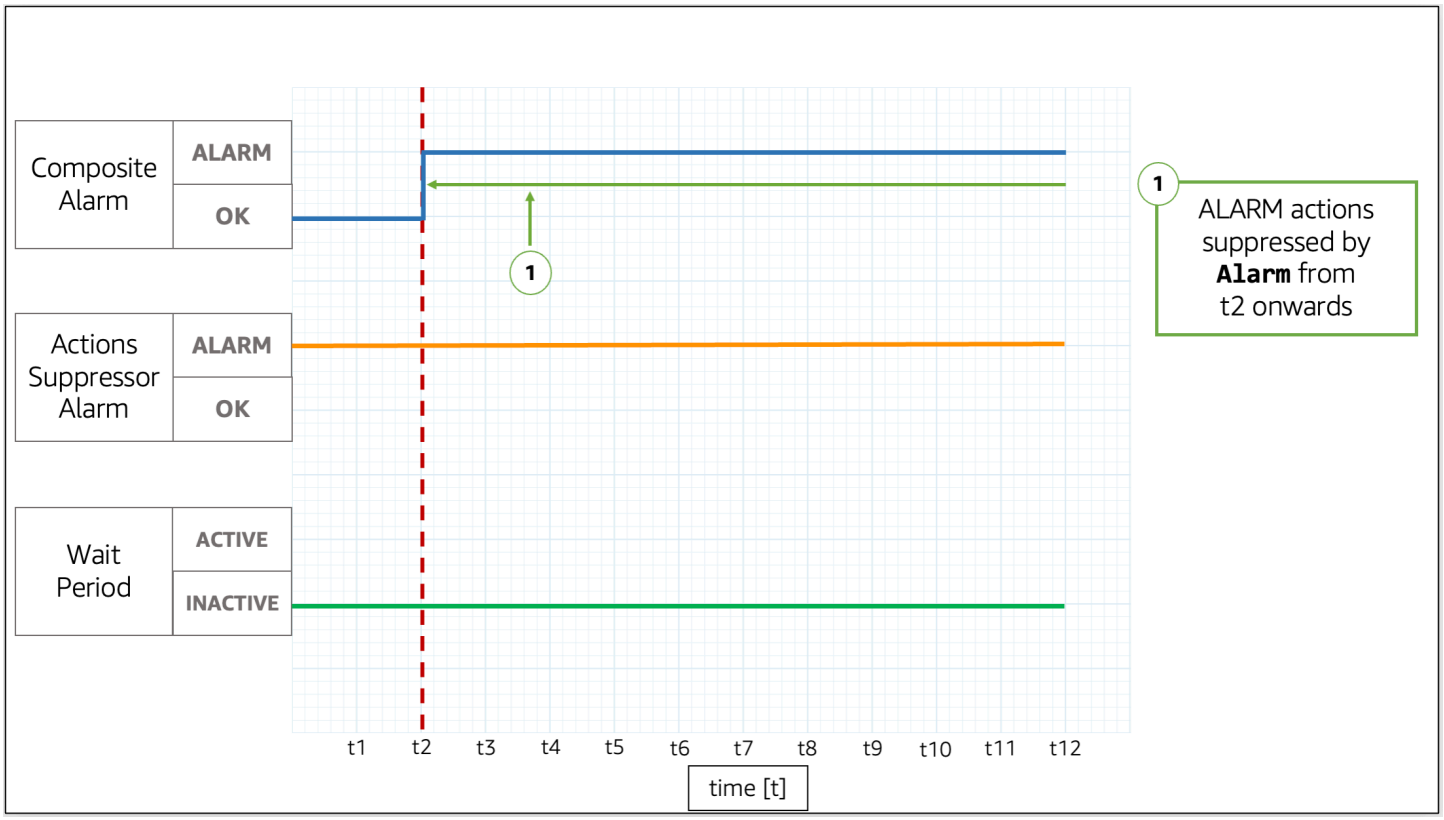
이 이미지에서 복합 경보는 t2에 OK 상태에서 ALARM 상태로 변경됩니다. WaitPeriod가 t2에 시작되어 t4에 끝납니다. 따라서 억제 경보가 t3에 OK 상태에서 ALARM 상태로 변경할 시간을 확보할 수 있습니다. t3에 억제 경보의 상태가 OK에서 ALARM으로 변경되므로, t2에 시작되는 WaitPeriod가 폐기되고 억제 경보가 이제 복합 경보가 작업을 수행하지 못하게 합니다.

예 3: **WaitPeriod**에 의해 작업이 억제될 때의 상태 전환



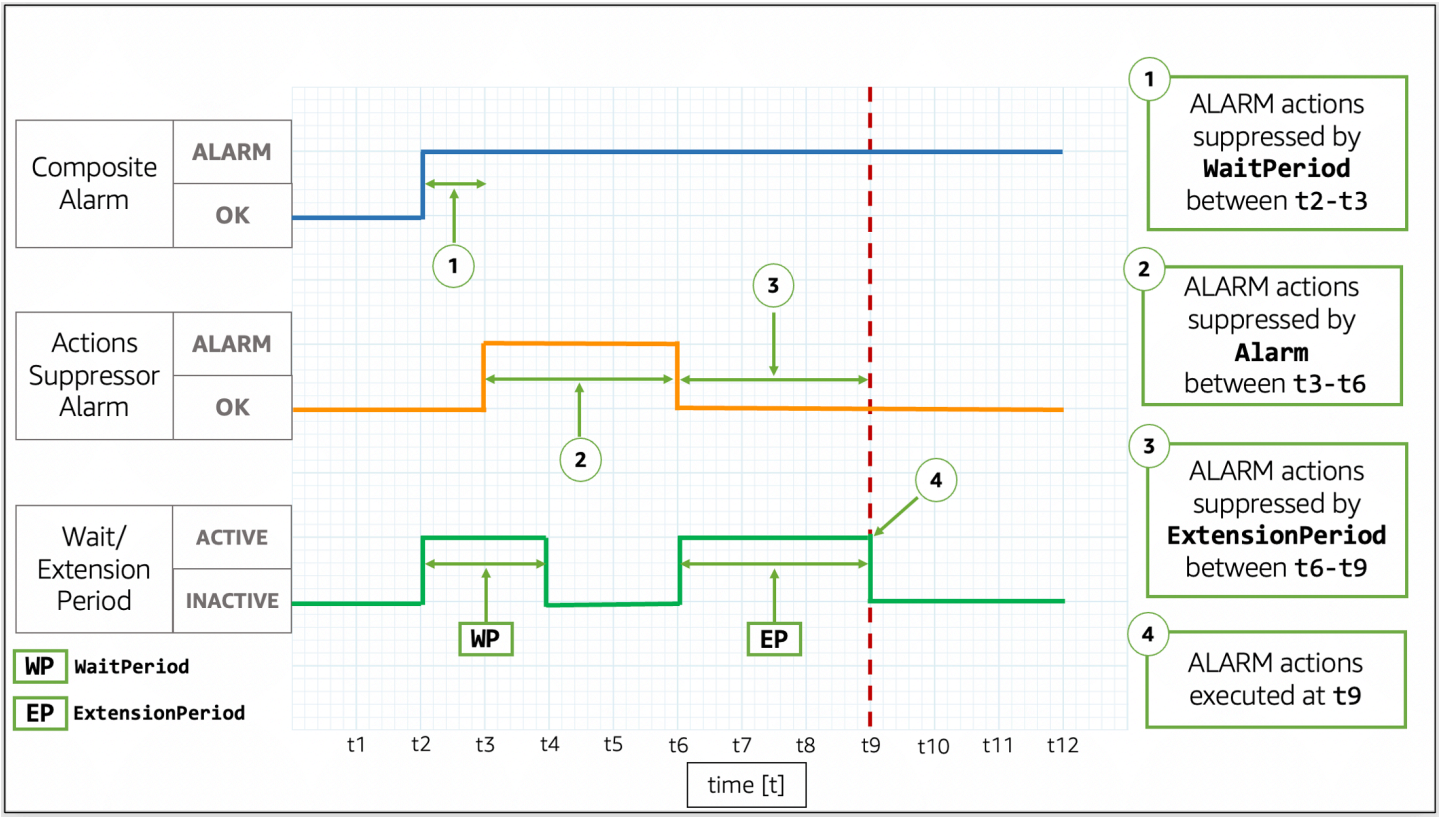
이 이미지에서 복합 경보는 t2에 OK 상태에서 ALARM 상태로 변경됩니다. WaitPeriod가 t2에 시작되어 t4에 끝납니다. 따라서 억제 경보가 상태를 변경할 시간을 확보할 수 있습니다. 복합 경보가 t3에 OK 상태로 다시 변경되므로, t2에 시작된 WaitPeriod가 폐기됩니다. 새 WaitPeriod는 t3에 시작되어 t5에 끝납니다. 새 WaitPeriod가 t5에 만료되면 복합 경보가 작업을 수행합니다.

예 4: 경보에 의해 작업이 억제될 때의 상태 전환



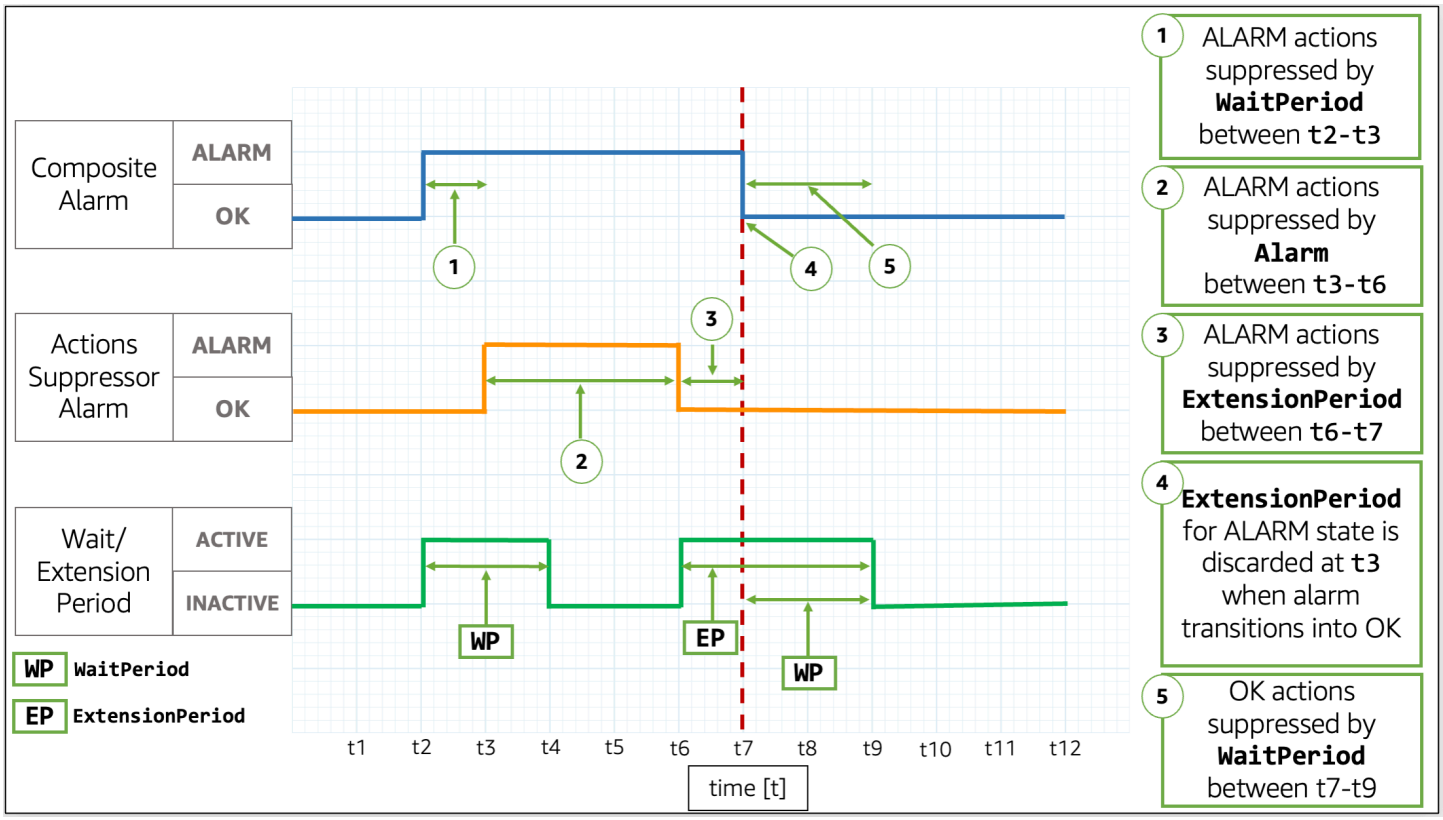
이 이미지에서 복합 경보는 t2에 OK 상태에서 ALARM 상태로 변경됩니다. 억제 경보가 이미 ALARM 상태입니다. 억제 경보는 복합 경보가 작업을 수행하는 것을 방지합니다.

예 5: **ExtensionPeriod** 후에 작업이 억제되지 않음



이 이미지에서 복합 경보는 t2에 OK 상태에서 ALARM 상태로 변경됩니다. WaitPeriod가 t2에 시작되어 t4에 끝납니다. 따라서 t6에 복합 경보의 작업을 억제하기 전까지, 억제 경보가 t3에 상태를 OK에서 ALARM으로 변경할 시간을 확보할 수 있습니다. t3에 억제 경보의 상태가 OK에서 ALARM으로 변경되므로, t2에 시작된 WaitPeriod가 폐기됩니다. t6에 억제 경보가 OK 상태로 변경됩니다. ExtensionPeriod가 t6에 시작되어 t9에 끝납니다. ExtensionPeriod이(가) 만료되면 복합 경보가 작업을 수행합니다.

예 6: **ExtensionPeriod**에 의해 작업이 억제될 때의 상태 전환



이 이미지에서 복합 경보는 t2에 OK 상태에서 ALARM 상태로 변경됩니다. WaitPeriod가 t2에 시작되어 t4에 끝납니다. 따라서 t6에 복합 경보의 작업을 억제하기 전까지, 억제 경보가 t3에 상태를 OK에서 ALARM으로 변경할 시간을 확보할 수 있습니다. t3에 억제 경보의 상태가 OK에서 ALARM으로 변경되므로, t2에 시작된 WaitPeriod가 폐기됩니다. t6에 억제 경보가 다시 OK 상태로 변경됩니다. ExtensionPeriod가 t6에 시작되어 t9에 끝납니다. t7에 복합 경보가 다시 OK 상태로 변경되면, ExtensionPeriod가 폐기되고 새 WaitPeriod가 t7에 시작되어 t9에 끝납니다.

> **Tip**
>
> 억제 경보를 바꾸면 모든 활성 WaitPeriod 또는 ExtensionPeriod가 폐기됩니다.

경보 변경에 따른 조치

CloudWatch는 경보의 상태가 변경될 때와 경보의 구성이 업데이트될 때 두 가지 유형의 경보 변경 사항을 사용자에게 알릴 수 있습니다.

경보가 평가할 때 ALARM, OK 또는 INSUFFICIENT_DATA와 같은 상태로 변경될 수 있습니다. 이러한 경보 상태 변경은 인시던트 발생 가능성, 정상 복귀 또는 지표를 사용할 수 없음을 알릴 수 있습니다. 이러한 경우 다음 옵션 중 하나를 사용하여 사용자의 참여를 유도하거나 경보를 보낼 수 있습니다.

- 경보 조치의 일환으로 SNS 주제에 알림을 보내도록 경보를 구성할 수 있습니다. 그런 다음 이메일 알림 및 SMS와 같은 채널을 포함하여 A2A(Application-to-Application) 메시징은 물론 A2P(Application-to-Person) 알림에 대해 SNS 주제를 구성할 수 있습니다. SNS 주제에 대해 정의한 모든 대상이 경보 알림을 수신합니다. 자세한 내용은 Amazon SNS 이벤트 대상을 참조하세요.
- 경보 상태 변경 이벤트에 대한 알림을 구성할 수 있습니다. AWS 사용자 알림은 이러한 알림을 구성하는 기본 방법을 제공하며 권장되는 접근 방식입니다.

또한 CloudWatch는 경보 상태가 변경될 때마다 그리고 경보가 생성, 삭제 또는 업데이트될 때마다 Amazon EventBridge로 이벤트를 전송합니다. EventBridge가 이러한 이벤트를 수신할 때 조치를 취하거나 알림을 받도록 EventBridge 규칙을 작성할 수 있습니다.

주제

경보 변경 시 사용자에게 알림

이 섹션에서는 AWS 사용자 알림 또는 Amazon Simple Notification Service를 사용하여 사용자에게 경보 변경 사항을 알리는 방법을 설명합니다.

AWS 사용자 알림 설정

AWS 사용자 알림을 사용하여 CloudWatch 경보 상태 변경 및 구성 변경 이벤트에 대한 알림을 수신할 전송 채널을 설정할 수 있습니다. 이벤트가 지정한 규칙과 일치하면 알림을 받습니다. 이메일, AWS Chatbot 채팅 알림 또는 AWS 콘솔 모바일 애플리케이션 푸시 알림 등 여러 채널을 통해 이벤트에 대한 알림을 받을 수 있습니다. 콘솔 알림 센터에서도 알림을 볼 수 있습니다. 사용자 알림은 집계를 지원하므로 특정 이벤트 중에 받는 알림 수를 줄일 수 있습니다.

AWS 사용자 알림을 사용하여 생성하는 알림 구성은 대상 경보 상태별로 구성할 수 있는 작업 수 제한에 포함되지 않습니다. AWS 사용자 알림은 특정 경보나 패턴을 허용 목록에 추가하거나 거부하는 고급 필터를 지정하지 않는 한 Amazon EventBridge로 전송되는 이벤트와 일치하므로 계정 및 선택한 리전의 모든 경보에 대한 알림을 전송합니다.

다음 고급 필터의 예에서는 `ServerCpuTooHigh`라는 경보의 경보 상태가 OK에서 ALARM으로 변경된 경우와 일치합니다.

```
{
"detail": {
    "alarmName": ["ServerCpuTooHigh"],
    "previousState": { "value": ["OK"] },
    "state": { "value": ["ALARM"] }
  }
}
```

EventBridge 이벤트에서 경보가 게시한 모든 속성을 사용하여 필터를 생성할 수 있습니다. 자세한 내용은 경보 이벤트 및 EventBridge 단원을 참조하십시오.

Amazon SNS 알림 설정

Amazon Simple Notification Service를 사용하여 모바일 문자 메시지(SMS) 및 이메일 메시지를 포함하여 애플리케이션 간(A2A) 메시지와 애플리케이션 대 사람(A2P) 메시지를 모두 보낼 수 있습니다. 자세한 내용은 Amazon SNS 이벤트 대상을 참조하세요.

경보가 취할 수 있는 모든 상태에 대해 SNS 주제에 메시지를 보내도록 경보를 구성할 수 있습니다. 특정 경보의 상태와 관련해 구성하는 모든 Amazon SNS 주제는 해당 경보 및 상태에 구성할 수 있는 작업 수 제한에 포함됩니다. 계정의 모든 경보에서 동일한 Amazon SNS 주제로 메시지를 보낼 수 있으며 애플리케이션(A2A) 및 사람(A2P) 소비자 모두에게 동일한 Amazon SNS 주제를 사용할 수 있습니다. 이 구성은 경보 수준에서 이루어지므로 구성한 경보만 선택한 Amazon SNS 항목으로 메시지를 전송합니다.

먼저, 주제를 생성한 다음 구독합니다. 선택적으로 테스트 메시지를 주제에 게시할 수 있습니다. 예시는 AWS Management Console을 사용하여 Amazon SNS 주제 설정 섹션을 참조하세요. 또는 자세한 내용은 Amazon SNS 시작하기를 참조하세요.

또는 AWS Management Console을 사용하여 CloudWatch 경보를 만들려는 경우 경보를 만들 때 주제를 만들 수 있으므로 이 절차를 건너뛰어도 됩니다.

CloudWatch 경보를 생성할 때 경보가 입력하는 모든 대상 상태에 대한 작업을 추가할 수 있습니다. 알림을 받으려는 상태에 대한 Amazon SNS 알림을 추가하고 이전 단계에서 만든 Amazon SNS 주제를 선택하여 경보가 선택한 상태에 진입할 때 이메일 알림을 보냅니다.

> **Note**
>
> Amazon SNS 주제를 생성할 때 해당 주제를 표준 주제 또는 FIFO 주제로 선택합니다. CloudWatch는 두 가지 유형의 주제에 대한 모든 경보 알림을 게시하도록 보장합니다. 그러나 FIFO 주제를 사용하더라도 CloudWatch가 순서에 맞지 않게 알림을 주제로 보내는 경우가 드물게 있습니다. FIFO 주제를 사용하는 경우 경보는 경보 알림의 메시지 그룹 ID를 경보의 ARN 해시로 설정합니다.

혼동된 대리자 문제 방지

교차 서비스 혼동된 대리인 보안 문제를 방지하려면 Amazon SNS 리소스에 액세스할 수 있는 권한을 CloudWatch에 부여하는 Amazon SNS 리소스 정책에 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 키를 사용하는 것이 좋습니다.

다음 예제 리소스 정책은 `aws:SourceArn` 조건 키를 사용하여 지정된 계정의 CloudWatch 경보에서만 사용하도록 `SNS:Publish` 권한을 좁힙니다.

```
{
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudwatch.amazonaws.com"
        },
        "Action": "SNS:Publish",
        "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cloudwatch:us-east-2:111122223333:alarm:*"
            },
            "StringEquals": {
                "aws:SourceAccount": "111122223333"
            }
        }
    }]
}
```

경보 ARN에 ASCII가 아닌 문자가 포함된 경우 `aws:SourceAccount` 전역 조건 키만 사용하여 권한을 제한합니다.

AWS Management Console을 사용하여 Amazon SNS 주제 설정

먼저, 주제를 생성한 다음 구독합니다. 선택적으로 테스트 메시지를 주제에 게시할 수 있습니다.

SNS 주제를 생성하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. Amazon SNS 대시보드의 [일반 작업(Common actions)]에서 [주제 생성(Create Topic)]을 선택합니다.
3. 새로운 주제 생성 대화 상자의 주제 이름에 주제 이름(예: **`my-topic`**)을 입력합니다.
4. 주제 생성을 선택합니다.
5. 다음 태스크에 대한 [주제 ARN(Topic ARN)]을 복사합니다(예를 들어 arn:aws:sns:us-east-1:111122223333:my-topic).

SNS 주제를 구독하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 구독과 구독 생성을 선택합니다.
3. 구독 생성 대화 상자의 주제 ARN에서 이전 작업에서 생성한 주제 ARN을 붙여 넣습니다.
4. 프로토콜에서 이메일을 선택합니다.
5. 엔드포인트에 알림을 받는 데 사용할 수 있는 이메일 주소를 입력한 다음 구독 생성을 선택합니다.
6. 이메일 애플리케이션에서 AWS 알림에서 보낸 메시지를 연 다음, 구독을 확인합니다.

   웹 브라우저에 Amazon SNS의 확인 응답이 표시됩니다.

SNS 주제에 테스트 메시지를 게시하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 주제를 선택합니다.
3. 주제 페이지에서 주제를 선택하고 주제 게시를 선택합니다.
4. 메시지 게시 페이지의 제목에 메시지에 대한 제목 줄을 입력하고 메시지에 간단한 메시지를 입력합니다.
5. 메시지 게시를 선택합니다.
6. 해당 메시지를 받았는지 이메일을 확인합니다.

AWS CLI를 사용하여 SNS 주제 설정

먼저 SNS 주제를 생성한 다음, 해당 주제에 직접 메시지를 게시해서 제대로 구성이 되었는지 테스트합니다.

SNS 주제를 설정하려면

1. 아래와 같이 create-topic 명령을 사용하여 주제를 생성합니다.

   ```
   aws sns create-topic --name my-topic
   ```

   Amazon SNS는 다음 형식의 주제 ARN을 반환합니다.

   ```
   {
       "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic"
   }
   ```

2. subscribe 명령을 사용하여 구독 이메일 주소를 주제에 연결합니다. 구독 요청이 성공하면 구독 확인 이메일 메시지를 받게 됩니다.

   ```
   aws sns subscribe --topic-arn arn:aws:sns:us-east-1:111122223333:my-topic --
   protocol email --notification-endpoint my-email-address
   ```

   Amazon SNS는 다음을 반환합니다.

   ```
   {
       "SubscriptionArn": "pending confirmation"
   }
   ```

3. 이메일 애플리케이션에서 AWS 알림에서 보낸 메시지를 연 다음, 구독을 확인합니다.

   웹 브라우저에 Amazon Simple Notification Service의 확인 응답이 표시됩니다.

4. list-subscriptions-by-topic 명령을 사용하여 구독을 확인합니다.

   ```
   aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-
   east-1:111122223333:my-topic
   ```

   Amazon SNS는 다음을 반환합니다.

   ```
   {
   ```

```
  "Subscriptions": [
    {
        "Owner": "111122223333",
        "Endpoint": "me@mycompany.com",
        "Protocol": "email",
        "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic",
        "SubscriptionArn": "arn:aws:sns:us-east-1:111122223333:my-topic:64886986-
bf10-48fb-a2f1-dab033aa67a3"
    }
  ]
}
```

5. (선택 사항) publish 명령을 사용하여 해당 주제로 테스트 메시지를 게시합니다.

```
aws sns publish --message "Verification" --topic arn:aws:sns:us-
east-1:111122223333:my-topic
```

Amazon SNS는 다음을 반환합니다.

```
{
    "MessageId": "42f189a0-3094-5cf6-8fd7-c2dde61a4d7d"
}
```

6. 해당 메시지를 받았는지 이메일을 확인합니다.

경보 이벤트 및 EventBridge

CloudWatch는 CloudWatch 경보가 생성, 업데이트, 삭제 또는 변경될 때마다 Amazon EventBridge에 이벤트를 전송합니다. EventBridge 및 이러한 이벤트를 사용하여 경보 상태가 변경될 때 알림과 같은 작업을 수행하는 규칙을 작성할 수 있습니다. 자세한 내용은 Amazon EventBridge란 무엇인가요? 단원을 참조하세요.

CloudWatch는 EventBridge로의 경보 상태 변경 이벤트 전달을 보장합니다.

CloudWatch의 이벤트 샘플

이 단원에는 CloudWatch의 이벤트 예가 나와 있습니다.

단일 지표 상태 변경 경보

```
{
    "version": "0",
```

```
    "id": "c4c1c1c9-6542-e61b-6ef0-8c4d36933a92",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-02T17:04:40Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
    ],
    "detail": {
        "alarmName": "ServerCpuTooHigh",
        "configuration": {
            "description": "Goes into alarm when server CPU utilization is too high!",
            "metrics": [
                {
                    "id": "30b6c6b2-a864-43a2-4877-c09a1afc3b87",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            },
                            "name": "CPUUtilization",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Average"
                    },
                    "returnData": true
                }
            ]
        },
        "previousState": {
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints
 [0.0666851903306472 (01/10/19 13:46:00)] was not greater than the threshold (50.0)
 (minimum 1 datapoint for ALARM -> OK transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-01T13:56:40.985+0000\",\"startDate\":\"2019-10-01T13:46:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[0.0666851903306472],
\"threshold\":50.0}",
            "timestamp": "2019-10-01T13:56:40.987+0000",
            "value": "OK"
        },
        "state": {
```

```
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints
 [99.50160229693434 (02/10/19 16:59:00)] was greater than the threshold (50.0) (minimum
 1 datapoint for OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-02T17:04:40.985+0000\",\"startDate\":\"2019-10-02T16:59:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[99.50160229693434],
\"threshold\":50.0}",
            "timestamp": "2019-10-02T17:04:40.989+0000",
            "value": "ALARM"
        }
    }
}
```

지표 수식 상태 변경 경보

```
{
    "version": "0",
    "id": "2dde0eb1-528b-d2d5-9ca6-6d590caf2329",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-02T17:20:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh"
    ],
    "detail": {
        "alarmName": "TotalNetworkTrafficTooHigh",
        "configuration": {
            "description": "Goes into alarm if total network traffic exceeds 10Kb",
            "metrics": [
                {
                    "expression": "SUM(METRICS())",
                    "id": "e1",
                    "label": "Total Network Traffic",
                    "returnData": true
                },
                {
                    "id": "m1",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
```

```
                            },
                            "name": "NetworkIn",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                },
                {
                    "id": "m2",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            },
                            "name": "NetworkOut",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                }
            ]
        },
        "previousState": {
            "reason": "Unchecked: Initial alarm creation",
            "timestamp": "2019-10-02T17:20:03.642+0000",
            "value": "INSUFFICIENT_DATA"
        },
        "state": {
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints [45628.0
 (02/10/19 17:10:00)] was greater than the threshold (10000.0) (minimum 1 datapoint for
 OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-02T17:20:48.551+0000\",\"startDate\":\"2019-10-02T17:10:00.000+0000\",
\"period\":300,\"recentDatapoints\":[45628.0],\"threshold\":10000.0}",
            "timestamp": "2019-10-02T17:20:48.554+0000",
            "value": "ALARM"
        }
    }
}
```

이상 탐지 상태 변경 경보


```
{
    "version": "0",
    "id": "daafc9f1-bddd-c6c9-83af-74971fcfc4ef",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-03T16:00:04Z",
    "region": "us-east-1",
    "resources": ["arn:aws:cloudwatch:us-east-1:123456789012:alarm:EC2 CPU Utilization
 Anomaly"],
    "detail": {
        "alarmName": "EC2 CPU Utilization Anomaly",
        "state": {
            "value": "ALARM",
            "reason": "Thresholds Crossed: 1 out of the last 1 datapoints [0.0
 (03/10/19 15:58:00)] was less than the lower thresholds [0.020599444741798756] or
 greater than the upper thresholds [0.3006915352732461] (minimum 1 datapoint for OK ->
 ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-03T16:00:04.650+0000\",\"startDate\":\"2019-10-03T15:58:00.000+0000\",
\"period\":60,\"recentDatapoints\":[0.0],\"recentLowerThresholds\":
[0.020599444741798756],\"recentUpperThresholds\":[0.3006915352732461]}",
            "timestamp": "2019-10-03T16:00:04.653+0000"
        },
        "previousState": {
            "value": "OK",
            "reason": "Thresholds Crossed: 1 out of the last 1 datapoints
 [0.166666666664241 (03/10/19 15:57:00)] was not less than the lower thresholds
 [0.0206719426210418] or not greater than the upper thresholds [0.30076870222143803]
 (minimum 1 datapoint for ALARM -> OK transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-03T15:59:04.670+0000\",\"startDate\":\"2019-10-03T15:57:00.000+0000\",
\"period\":60,\"recentDatapoints\":[0.166666666664241],\"recentLowerThresholds\":
[0.0206719426210418],\"recentUpperThresholds\":[0.30076870222143803]}",
            "timestamp": "2019-10-03T15:59:04.672+0000"
        },
        "configuration": {
            "description": "Goes into alarm if CPU Utilization is out of band",
            "metrics": [{
                "id": "m1",
                "metricStat": {
                    "metric": {
```

```
                        "namespace": "AWS/EC2",
                        "name": "CPUUtilization",
                        "dimensions": {
                            "InstanceId": "i-12345678901234567"
                        }
                    },
                    "period": 60,
                    "stat": "Average"
                },
                "returnData": true
            }, {
                "id": "ad1",
                "expression": "ANOMALY_DETECTION_BAND(m1, 0.8)",
                "label": "CPUUtilization (expected)",
                "returnData": true
            }]
        }
    }
}
```

억제 경보가 있는 복합 경보의 상태 변경

```
{
    "version": "0",
    "id": "d3dfc86d-384d-24c8-0345-9f7986db0b80",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-22T15:57:45Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "actionsSuppressedReason": "Actions suppressed by WaitPeriod",
            "value": "ALARM",
            "reason": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:SuppressionDemo.EventBridge.FirstChild transitioned to ALARM
 at Friday 22 July, 2022 15:57:45 UTC",
```

```
            "reasonData": "{\"triggeringAlarms\":[{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:ServerCpuTooHigh\",\"state\":{\"value\":\"ALARM\",\"timestamp
\":\"2022-07-22T15:57:45.394+0000\"}}]}",
            "timestamp": "2022-07-22T15:57:45.394+0000"
        },
        "previousState": {
            "value": "OK",
            "reason": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:SuppressionDemo.EventBridge.Main was created and its alarm
 rule evaluates to OK",
            "reasonData": "{\"triggeringAlarms\":[{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:TotalNetworkTrafficTooHigh\",\"state\":{\"value\":\"OK\",
\"timestamp\":\"2022-07-14T16:28:57.770+0000\"}},{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:ServerCpuTooHigh\",\"state\":{\"value\":\"OK\",\"timestamp\":
\"2022-07-14T16:28:54.191+0000\"}}]}",
            "timestamp": "2022-07-22T15:56:14.552+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180
        }
    }
}
```

복합 경보 생성

```
{

    "version": "0",
    "id": "91535fdd-1e9c-849d-624b-9a9f2b1d09d0",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-03-03T17:06:22Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
```

```
        "operation": "create",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-03-03T17:06:22.289+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "alarmName": "ServiceAggregatedAlarm",
            "description": "Aggregated monitor for instance",
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:22.289+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

억제 경보가 있는 복합 경보 생성

```
{
    "version": "0",
    "id": "454773e1-09f7-945b-aa2c-590af1c3f8e0",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T13:59:46Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "create",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
```

```
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:46.425+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

지표 경보 업데이트

```
{

    "version": "0",
    "id": "bc7d3391-47f8-ae47-f457-1b4d06118d50",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-03-03T17:06:34Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
    ],
    "detail": {
        "alarmName": "ServerCpuTooHigh",
        "operation": "update",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-03-03T17:06:13.757+0000"
        },
        "configuration": {
            "evaluationPeriods": 1,
            "threshold": 80,
            "comparisonOperator": "GreaterThanThreshold",
            "treatMissingData": "ignore",
            "metrics": [
                {
                    "id": "86bfa85f-b14c-ebf7-8916-7da014ce23c0",
                    "metricStat": {
                        "metric": {
```

```
                        "namespace": "AWS/EC2",
                        "name": "CPUUtilization",
                        "dimensions": {
                            "InstanceId": "i-12345678901234567"
                        }
                    },
                    "period": 300,
                    "stat": "Average"
                },
                "returnData": true
            }
        ],
        "alarmName": "ServerCpuTooHigh",
        "description": "Goes into alarm when server CPU utilization is too high!",
        "actionsEnabled": true,
        "timestamp": "2022-03-03T17:06:34.267+0000",
        "okActions": [],
        "alarmActions": [],
        "insufficientDataActions": []
    },
    "previousConfiguration": {
        "evaluationPeriods": 1,
        "threshold": 70,
        "comparisonOperator": "GreaterThanThreshold",
        "treatMissingData": "ignore",
        "metrics": [
            {
                "id": "d6bfa85f-893e-b052-a58b-4f9295c9111a",
                "metricStat": {
                    "metric": {
                        "namespace": "AWS/EC2",
                        "name": "CPUUtilization",
                        "dimensions": {
                            "InstanceId": "i-12345678901234567"
                        }
                    },
                    "period": 300,
                    "stat": "Average"
                },
                "returnData": true
            }
        ],
        "alarmName": "ServerCpuTooHigh",
        "description": "Goes into alarm when server CPU utilization is too high!",
```

```
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:13.757+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

억제 경보가 있는 복합 경보 업데이트

```
{
    "version": "0",
    "id": "4c6f4177-6bd5-c0ca-9f05-b4151c54568b",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T13:59:56Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "update",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "value": "ALARM",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 360,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:56.290+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        },
```

```
        "previousConfiguration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:46.425+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

지표 수학 경보 삭제

```
{

    "version": "0",
    "id": "f171d220-9e1c-c252-5042-2677347a83ed",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-03-03T17:07:13Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh"
    ],
    "detail": {
        "alarmName": "TotalNetworkTrafficTooHigh",
        "operation": "delete",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-03-03T17:06:17.672+0000"
        },
        "configuration": {
            "evaluationPeriods": 1,
            "threshold": 10000,
            "comparisonOperator": "GreaterThanThreshold",
            "treatMissingData": "ignore",
            "metrics": [{
```

```
                "id": "m1",
                "metricStat": {
                    "metric": {
                        "namespace": "AWS/EC2",
                        "name": "NetworkIn",
                        "dimensions": {
                            "InstanceId": "i-12345678901234567"
                        }
                    },
                    "period": 300,
                    "stat": "Maximum"
                },
                "returnData": false
            },
            {
                "id": "m2",
                "metricStat": {
                    "metric": {
                        "namespace": "AWS/EC2",
                        "name": "NetworkOut",
                        "dimensions": {
                            "InstanceId": "i-12345678901234567"
                        }
                    },
                    "period": 300,
                    "stat": "Maximum"
                },
                "returnData": false
            },
            {
                "id": "e1",
                "expression": "SUM(METRICS())",
                "label": "Total Network Traffic",
                "returnData": true
            }
        ],
        "alarmName": "TotalNetworkTrafficTooHigh",
        "description": "Goes into alarm if total network traffic exceeds 10Kb",
        "actionsEnabled": true,
        "timestamp": "2022-03-03T17:06:17.672+0000",
        "okActions": [],
        "alarmActions": [],
        "insufficientDataActions": []
```

```
        }
    }
}
```

억제 경보가 있는 복합 경보 삭제

```
{
    "version": "0",
    "id": "e34592a1-46c0-b316-f614-1b17a87be9dc",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T14:00:01Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "delete",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "value": "ALARM",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 360,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:56.290+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

경보 관리

CloudWatch 경보 편집 또는 삭제

기존 경보를 편집하거나 삭제할 수 있습니다.

기존 경보의 이름을 변경할 수 없습니다. 경보를 복사하고 새 경보에 다른 이름을 지정할 수 있습니다. 경보를 복사하려면 경보 목록에서 경보 이름 옆에 있는 확인란을 선택하고 작업, 복사를 선택합니다.

경보를 편집하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms), 모든 경보(All Alarms)를 선택합니다.
3. 경보 이름을 선택합니다.
4. 태그를 추가하거나 제거하려면 태그 탭을 선택한 다음 태그 관리를 선택합니다.
5. 경보의 다른 부분을 편집하려면 작업, 편집을 선택합니다.

   선택한 지표 및 통계에 대한 그래프와 기타 정보가 표시된 Specify metric and conditions(지표 및 조건 지정) 페이지가 나타납니다.
6. 지표를 변경하려면 편집을 선택하고 모든 지표 탭을 선택한 후 다음 중 하나를 수행합니다.
   - 원하는 지표가 포함된 서비스 네임스페이스를 선택합니다. 옵션이 나타날 때 계속해서 옵션을 선택하면 선택 범위가 좁아집니다. 지표 목록이 표시되면 원하는 지표 옆에 있는 확인란을 선택합니다.
   - 검색 상자에 지표 이름, 측정기준 또는 리소스 ID를 입력하고 Enter 키를 누릅니다. 그런 다음 결과 중 하나를 선택하고 지표 목록이 표시될 때까지 계속 진행합니다. 원하는 지표 옆의 확인란을 선택합니다.

   지표 선택을 선택하세요.
7. 경보의 다른 측면을 변경하려면 적절한 옵션을 선택합니다. 경보가 ALARM 상태가 되거나 누락된 데이터가 처리되는 방법을 변경하기 위해 위반해야 하는 데이터 포인트 수를 변경하려면 추가 구성을 선택합니다.
8. 다음을 선택합니다.
9. 알림, Auto Scaling action(Auto Scaling 작업) 및 EC2 작업에서 경보가 트리거될 때 수행한 동작을 선택적으로 편집합니다. 그리고 다음(Next)을 선택합니다.
10. 선택적으로 경보 설명을 변경합니다.

기존 경보의 이름을 변경할 수 없습니다. 경보를 복사하고 새 경보에 다른 이름을 지정할 수 있습니다. 경보를 복사하려면 경보 목록에서 경보 이름 옆에 있는 확인란을 선택하고 작업, 복사를 선택합니다.

11. 다음을 선택합니다.
12. Preview and create(미리 보기 및 생성)에서 정보 및 조건이 원하는 내용인지 확인한 다음 Update alarm(경보 업데이트)을 선택합니다.

Amazon SNS 콘솔을 사용하여 생성된 이메일 알림 목록을 업데이트하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 주제를 선택한 다음, 알림 목록(주제)에 대한 ARN을 선택합니다.
3. 다음 중 하나를 수행하십시오.
   - 이메일 주소를 추가하려면 구독 생성을 선택합니다. 프로토콜에서 이메일을 선택합니다. 엔드포인트에 새 수신자의 이메일 주소를 입력합니다. 구독 생성을 선택합니다.
   - 이메일 주소를 제거하려면 구독 ID를 선택합니다. 기타 구독 작업과 구독 삭제를 선택합니다.
4. [Publish to topic]을 선택합니다.

경보를 삭제하는 방법

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택합니다.
3. 경보 이름 왼쪽에 있는 확인란을 선택하고 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

Auto Scaling 경보 숨김

AWS Management Console에서 경보를 확인할 때 Amazon EC2 Auto Scaling과 Application Auto Scaling 모두에 대한 경보를 숨길 수 있습니다. 이 기능은 AWS Management Console에서만 사용할 수 있습니다.

일시적으로 Auto Scaling 경보 숨기기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 탐색 창에서 경보(Alarms), 모든 경보(All alarms) 및 자동 크기 조정 경보 숨기기(Hide Auto Scaling alarms)를 차례로 선택합니다.

경보 사용 사례 및 예

다음 섹션에서는 일반적인 경보 사용 사례에 대한 예 및 자습서를 제공합니다.

청구 경보를 생성하여 예상 AWS 요금을 모니터링하세요.

Amazon CloudWatch를 사용하여 예상 AWS 요금을 모니터링할 수 있습니다. AWS 계정에 대한 예상 요금 모니터링을 사용 설정하면 예상 요금이 계산되어 지표 데이터로서 매일 여러 번 CloudWatch에 전송됩니다.

결제 지표 데이터는 미국 동부(버지니아 북부) 리전에 저장되며 전 세계 요금을 나타냅니다. 이 데이터에는 사용한 AWS의 모든 서비스에 대한 예상 요금과 전반적인 총 AWS 예상 요금이 들어 있습니다.

계정 결제가 지정한 임곗값을 초과하면 경보가 작동합니다. 현재 결제가 임곗값을 초과하는 경우에만 경보가 작동합니다. 해당 시점까지의 월 사용량을 기준으로 추정하지 않습니다.

요금이 임곗값을 초과한 시점에 결제 경보를 생성할 경우 경보가 즉시 `ALARM` 상태가 됩니다.

> **Note**
>
> 이미 청구된 CloudWatch 요금을 분석하는 방법은 CloudWatch 결제 및 비용 섹션을 참조하세요.

Tasks

- 결제 알림 사용 설정
- 결제 경보 생성
- 결제 경보 삭제

결제 알림 사용 설정

예상 요금에 대한 경보를 생성할 수 있으려면 먼저 결제 알림을 활성화해야 합니다. 그래야만 예상되는 AWS 요금을 모니터링하고 결제 지표 데이터를 사용하여 경보를 생성할 수 있습니다. 결제 알림을 활

성화하고 나면 데이터 수집을 비활성화할 수 없습니다. 그러나 생성된 모든 결제 경보를 삭제할 수는 있습니다.

결제 알림을 처음 활성화하고 나서 결제 데이터를 확인하고 결제 경보를 설정할 수 있기까지 약 15분 정도의 시간이 걸립니다.

요구 사항

- 계정 루트 사용자 자격 증명을 사용하여 로그인하거나 결제 정보를 볼 수 있는 권한을 부여받은 IAM 사용자로 로그인해야 합니다.
- 통합 결제 계정의 경우 결제 계정으로 로그인하면 연결된 각 계정에 대한 결제 데이터를 찾을 수 있습니다. 통합 계정에 대해서뿐만 아니라 연결된 각 계정에 대한 서비스별 총 예상 요금 및 예상 요금에 대한 결제 데이터를 볼 수 있습니다.
- 통합 결제 계정에서 멤버에 연결된 계정 지표는 지급인 계정이 결제 알림 받기 기본 설정을 사용하도록 설정한 경우에만 캡처됩니다. 관리/지급인 계정인 계정을 변경하는 경우 새 관리/지급인 계정에서 결제 알림을 사용해야 합니다.
- APN 계정에 대한 결제 지표는 CloudWatch에 게시되지 않으므로 계정이 Amazon 파트너 네트워크(APN)에 속하지 않아야 합니다. 자세한 내용은 AWS 파트너 네트워크를 참조하세요.

예상 요금 모니터링을 활성화하려면

1. AWS Billing 결제 콘솔을 https://console.aws.amazon.com/billing/에서 엽니다.
2. 탐색 창에서 결제 기본 설정(Billing preferences)을 선택합니다.
3. 알림 환경 설정에서 편집을 선택합니다.
4. CloudWatch 결제 알림 수신을 선택합니다.
5. 기본 설정 저장을 선택합니다.

결제 경보 생성

> **Important**
>
> 결제 경보를 생성하기 전에 Region(리전)을 US East (N. Virginia)(미국 동부(버지니아 북부))로 설정해야 합니다. 결제 지표 데이터는 이 리전에 저장되어 전 세계 요금을 나타냅니다. 또한 계정 또는 관리/지급인 계정(통합 결제를 사용하는 경우)에 대한 결제 알림을 활성화해야 합니다. 자세한 내용은 결제 알림 사용 설정 단원을 참조하십시오.

이 절차에서는 AWS에 대한 예상 요금이 정의된 임곗값을 초과할 때 알림을 보내는 경보를 생성합니다.

CloudWatch 콘솔을 사용하여 결제 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택한 다음 모든 경보(All alarms)를 선택합니다.
3. 경보 생성을 선택하세요.
4. 지표 선택을 선택하세요. Browse(찾아보기)에서 Billing(결제)을 선택한 다음 Total Estimated Charge(예상 요금 합계)를 선택합니다.

   > **Note**
   >
   > 결제/예상 요금 합계 지표가 표시되지 않으면 결제 알림을 활성화하고 리전을 미국 동부(버지니아 북부)로 변경합니다. 자세한 내용은 결제 알림 사용 설정 단원을 참조하십시오.

5. EstimatedCharges 지표 상자를 선택한 다음 Select metric(지표 선택)을 선택합니다.
6. 통계에서 최대를 선택합니다.
7. Period(기간)에서 6 hours(6시간)를 선택합니다.
8. 임곗값 유형에서 정적을 선택합니다.
9. Whenever EstimatedCharges is . . .(EstimatedCharges가 다음인 경우 항상…)에서 Greater(보다 큼)를 선택합니다.
10. . . . 보다에 대해 경보를 트리거하려는 값을 정의합니다. 예를 들어 USD **200**달러입니다.

    EstimatedCharges 지표 값은 미국 달러(USD)로만 표시되며, 통화 변환은 Amazon Services LLC에서 제공합니다. 자세한 내용은 AWS Billing란 무엇인가요?를 참조하세요.

    > **Note**
    >
    > 임곗값을 정의하면 미리 보기 그래프에 이번 달의 예상 요금이 표시됩니다.

11. 추가 구성을 선택하고 다음을 수행합니다.

    - Datapoints to alarm(경보를 보낼 데이터 포인트)에서 1 out of 1(1/1)을 지정합니다.
    - Missing data treatment(누락된 데이터 처리)에서 Treat missing data as missing(누락된 데이터를 누락으로 처리)을 선택합니다.

12. 다음을 선택합니다.

13. 알림에서 경보 내가 선택되어 있는지 확인하세요. 그런 다음 경보가 ALARM 상태일 때 알림을 받을 Amazon SNS 주제를 지정합니다. Amazon SNS 주제에 이메일 주소를 포함하면 청구 금액이 지정한 임곗값을 초과할 때 이메일을 받을 수 있습니다.

    기존 Amazon SNS 주제를 선택하거나, 새 Amazon SNS 주제를 생성하거나, 주제 ARN을 사용하여 다른 계정에 알릴 수 있습니다. 경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 전송하도록 하려면 Add notification(알림 추가)을 선택합니다.
14. 다음을 선택합니다.
15. Name and description(이름 및 설명)에 경보 이름을 입력합니다. 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다.

    - (선택 사항) 경보에 대한 설명을 입력합니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.
16. 다음을 선택합니다.
17. Preview and create(미리 보기 및 생성)에서 구성이 올바른지 확인한 다음 Create alarm(경보 생성)을 선택합니다.

결제 경보 삭제

결제 경보가 더 이상 필요하지 않다면 삭제할 수 있습니다.

결제 경보를 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 필요한 경우 리전을 미국 동부(버지니아 북부)로 변경합니다. 결제 지표 데이터는 이 리전에 저장되며 전 세계 요금을 반영합니다.
3. 탐색 창에서 경보(Alarms) 모든 경보(All Alarms)를 선택합니다.
4. 경보 옆의 확인란을 선택하고 작업, 삭제를 차례로 선택합니다.
5. 확인 메시지가 나타나면 예, 삭제합니다(Yes, Delete)를 선택합니다.

CPU 사용률 경보 생성

경보 상태가 OK에서 ALARM으로 변경될 때 Amazon SNS를 사용하여 알림을 보내는 CloudWatch 경보를 생성할 수 있습니다.

EC2 인스턴스의 평균 CPU 사용률이 지정된 기간 동안 연속해서 지정된 임곗값을 초과하면 경보 상태가 ALARM으로 바뀝니다.

AWS Management Console을 사용하여 CPU 사용량 경보 설정

다음 단계에 따라 AWS Management Console을 사용해 CPU 사용량 경보를 생성합니다.

CPU 사용량을 기반으로 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms), 모든 경보(All Alarms)를 선택합니다.
3. 경보 생성(Create alarm)을 선택하세요.
4. 지표 선택을 선택하세요.
5. 모든 지표 탭에서 EC2 지표를 선택합니다.
6. 지표 범주(예: 인스턴스별 지표)를 선택합니다.
7. 원하는 인스턴스가 InstanceId 열에 나열되고 CPUUtilization이 지표 이름(Metric Name) 열에 있는 행을 찾습니다. 이 행 옆의 확인란을 선택하고 지표 선택을 선택합니다.
8. 지표 및 조건 지정 아래에 있는 통계에서 평균을 선택하거나, 사전 정의된 백분위수 중 하나를 선택하거나, 사용자 지정 백분위수(예: **p95.45**)를 지정합니다.
9. 기간(예: **5 minutes**)을 선택합니다.
10. 조건에서 다음을 지정합니다.

    a. 임곗값 유형에서 정적을 선택합니다.

    b. CPUUtilization이(가) 다음과 같은 경우에 항상에서 보다 큼을 지정합니다. than...에서 CPU 사용률이 이 비율을 초과할 경우 경보를 ALARM 상태로 전환할 임곗값을 지정합니다. 예: 70.

    c. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

    N 중 M 경보를 생성하려면 두 번째 값에 지정한 값보다 낮은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가 단원을 참조하세요.

    d. 누락 데이터 처리에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하세요.

    e. 경보가 모니터링된 통계 값으로 백분위수를 사용하는 경우에는 샘플이 부족한 백분위수 상자가 표시됩니다. 샘플 비율이 낮은 사례를 평가 또는 무시할지 여부를 선택할 때 이 상자를 사

용합니다. ignore (maintain alarm state)(무시(경보 상태 유지))를 선택하면 샘플 크기가 너무 작을 때 현재 경보 상태가 항상 유지됩니다. 자세한 내용은 백분위수 기반 CloudWatch 경보 및 데이터 샘플 부족 단원을 참조하세요.

11. 다음을 선택합니다.
12. 알림에서 경보 상태를 선택하고 경보가 ALARM 상태일 때 알릴 SNS 주제를 선택합니다.

    경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

    경보에서 알림을 보내지 않게 하려면 제거를 선택합니다.
13. 마친 후에는 다음을 선택합니다.
14. 경보 이름 및 설명을 입력합니다. 다음을 선택합니다.

    이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.
15. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

AWS CLI를 사용하여 CPU 사용량 경보 설정

다음 단계에 따라 AWS CLI을 사용해 CPU 사용량 경보를 생성합니다.

CPU 사용량을 기반으로 경보를 생성하려면

1. SNS 주제를 설정합니다. 자세한 내용은 Amazon SNS 알림 설정 단원을 참조하세요.
2. 아래와 같이 put-metric-alarm 명령을 사용하여 경보를 생성합니다.

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70%" --metric-name CPUUtilization --namespace AWS/EC2 --statistic
 Average --period 300 --threshold 70 --comparison-operator GreaterThanThreshold --
dimensions  Name=InstanceId,Value=i-12345678 --evaluation-periods 2 --alarm-actions
 arn:aws:sns:us-east-1:111122223333:my-topic --unit Percent
```

3. set-alarm-state 명령으로 경보 상태를 강제로 변경하여 경보를 테스트합니다.

    a. 경보 상태를 INSUFFICIENT_DATA에서 OK로 변경합니다.

```
aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason
 "initializing" --state-value OK
```

b. 경보 상태를 OK에서 ALARM로 변경합니다.

```
aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason
 "initializing" --state-value ALARM
```

c. 경보에 관한 알림을 받았는지 확인합니다.

이메일을 전송하는 로드 밸런서 지연 경보 생성

Amazon SNS 알림을 설정하고 Classic Load Balancer에 대해 100ms를 초과하는 대기 시간을 모니터링하는 경보를 구성할 수 있습니다.

AWS Management Console을 사용하여 대기 시간 경보 설정

다음 단계에 따라 AWS Management Console을 사용해 로드 밸런서 지연 시간 경보를 생성합니다.

로드 밸런서 대기 시간 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms), 모든 경보(All Alarms)를 선택합니다.
3. Create alarm(경보 생성)을 선택하세요.
4. 범주별 CloudWatch 지표에서 ELB 지표 범주를 선택합니다.
5. Classic Load Balancer 및 [대기 시간(Latency)] 지표가 있는 행을 선택합니다.
6. 통계의 경우 평균을 선택하거나, 사전 정의된 백분위수 중 하나를 선택하거나, 사용자 지정 백분위수(예: **p95.45**)를 지정합니다.
7. 기간의 경우 1분을 선택합니다.
8. 다음을 선택합니다.
9. 경보 임곗값에 경보의 고유한 이름(예: **myHighCpuAlarm**)과 경보에 대한 설명(예: **Alarm when Latency exceeds 100s**)을 입력합니다. 경보 이름은 UTF-8 문자만 포함해야 하고 ASCII 제어 문자를 포함할 수 없습니다.

   이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.

10. 다음 경우 항상의 조건에서 >를 선택하고 **0.1**을 입력합니다. 기간에 **3**을 입력합니다.
11. 누락 데이터 포인트가 경보 상태 변경을 트리거하지 않도록 추가 설정의 누락 데이터 처리에서 무시(경보 상태 유지)를 선택합니다.

    경보가 적정 수의 데이터 샘플이 있는 상황만 평가하도록 샘플이 부족한 백분위수에서 ignore (maintain alarm state)(무시(경보 상태 유지))를 선택합니다.
12. 작업의 이 경보가 발생할 경우 항상에서 상태가 ALARM입니다를 선택합니다. 알림 보내기에서 기존 SNS 주제를 선택하거나 새로 만듭니다.

    SNS 주제를 생성하려면 새 목록을 선택합니다. [알림 보내기(Send notification to)]에 SNS 주제 이름(예: **myHighCpuAlarm**)을 입력하고, [이메일 목록(Email list)]에 경보가 ALARM 상태로 변경될 때 알림을 받을 이메일 주소 목록을 쉼표로 구분하여 입력합니다. 각 이메일 주소로 주제 구독 확인 이메일이 전송됩니다. 알림을 받으려면 먼저 구독을 확정해야 합니다.
13. 경보 생성을 선택합니다.

AWS CLI를 사용하여 대기 시간 경보 설정

다음 단계에 따라 AWS CLI을 사용해 로드 밸런서 지연 시간 경보를 생성합니다.

로드 밸런서 대기 시간 경보를 생성하려면

1. SNS 주제를 설정합니다. 자세한 내용은 Amazon SNS 알림 설정 단원을 참조하세요.
2. 아래와 같이 put-metric-alarm 명령을 사용하여 경보를 생성합니다.

```
aws cloudwatch put-metric-alarm --alarm-name lb-mon --alarm-description "Alarm
 when Latency exceeds 100s" --metric-name Latency --namespace AWS/ELB --statistic
 Average --period 60 --threshold 100 --comparison-operator GreaterThanThreshold --
dimensions Name=LoadBalancerName,Value=my-server --evaluation-periods 3 --alarm-
actions arn:aws:sns:us-east-1:111122223333:my-topic --unit Seconds
```

3. set-alarm-state 명령으로 경보 상태를 강제로 변경하여 경보를 테스트합니다.

   a. 경보 상태를 INSUFFICIENT_DATA에서 OK로 변경합니다.

   ```
   aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason
    "initializing" --state-value OK
   ```

   b. 경보 상태를 OK에서 ALARM로 변경합니다.

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason
 "initializing" --state-value ALARM
```

c. 경보에 대한 이메일 알림을 받았음을 확인합니다.

이메일을 전송하는 스토리지 처리량 경보 생성

SNS 알림을 설정하고 Amazon EBS가 100MB의 처리량을 초과할 때 트리거되는 경보를 구성할 수 있습니다.

AWS Management Console을 사용하여 스토리지 처리량 경보 설정

다음 단계에 따라 AWS Management Console을 사용하여 Amazon EBS 처리량을 기반으로 경보를 생성합니다.

스토리지 처리량 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms), 모든 경보(All Alarms)를 선택합니다.
3. Create alarm(경보 생성)을 선택하세요.
4. EBS 지표에서 지표 범주를 선택합니다.
5. 볼륨과 VolumeWriteBytes 지표가 있는 행을 선택합니다.
6. 통계의 경우 평균(Average)을 선택합니다. 기간의 경우 5분을 선택합니다. 다음을 선택합니다.
7. 경보 임곗값에 경보의 고유한 이름(예: **myHighWriteAlarm**)과 경보에 대한 설명(예: **VolumeWriteBytes exceeds 100,000 KiB/s**)을 입력합니다. 이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.
8. 다음 경우 항상의 조건에서 **>**를 선택하고 **100000**을 입력합니다. 기간에 연속 기간으로 **15**를 입력합니다.

   경보 미리 보기 아래에 임곗값이 그래픽으로 표시됩니다.
9. 누락 데이터 포인트가 경보 상태 변경을 트리거하지 않도록 추가 설정의 누락 데이터 처리에서 무시(경보 상태 유지)를 선택합니다.
10. 작업의 이 경보가 발생할 경우 항상에서 상태가 ALARM입니다를 선택합니다. 알림 보내기에서 기존 SNS 주제를 선택하거나 새로 만듭니다.

SNS 주제를 생성하려면 새 목록을 선택합니다. [알림 보내기(Send notification to)]에 SNS 주제 이름(예: **myHighCpuAlarm**)을 입력하고, [이메일 목록(Email list)]에 경보가 ALARM 상태로 변경될 때 알림을 받을 이메일 주소 목록을 쉼표로 구분하여 입력합니다. 각 이메일 주소로 주제 구독 확인 이메일이 전송됩니다. 수신자가 구독을 확인해야만 이 이메일 주소로 알림이 전송될 수 있습니다.

11. 경보 생성을 선택합니다.

AWS CLI를 사용하여 스토리지 처리량 경보 설정

다음 단계에 따라 AWS CLI를 사용하여 Amazon EBS 처리량을 기반으로 경보를 생성합니다.

스토리지 처리량 경보를 생성하려면

1. SNS 주제를 생성합니다. 자세한 내용은 Amazon SNS 알림 설정 단원을 참조하세요.
2. 경보를 만듭니다.

```
aws cloudwatch put-metric-alarm --alarm-name ebs-mon --alarm-description "Alarm
 when EBS volume exceeds 100MB throughput" --metric-name VolumeReadBytes --
namespace AWS/EBS --statistic Average --period 300 --threshold 100000000 --
comparison-operator GreaterThanThreshold --dimensions Name=VolumeId,Value=my-
volume-id --evaluation-periods 3 --alarm-actions arn:aws:sns:us-
east-1:111122223333:my-alarm-topic --insufficient-data-actions arn:aws:sns:us-
east-1:111122223333:my-insufficient-data-topic
```

3. set-alarm-state 명령으로 경보 상태를 강제로 변경하여 경보를 테스트합니다.

   a. 경보 상태를 INSUFFICIENT_DATA에서 OK로 변경합니다.

   ```
   aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
    "initializing" --state-value OK
   ```

   b. 경보 상태를 OK에서 ALARM로 변경합니다.

   ```
   aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
    "initializing" --state-value ALARM
   ```

   c. 경보 상태를 ALARM에서 INSUFFICIENT_DATA로 변경합니다.

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
 "initializing" --state-value INSUFFICIENT_DATA
```

d. 경보에 대한 이메일 알림을 받았음을 확인합니다.

AWS 데이터베이스의 성능 개선 도우미 카운터 지표에 대한 경보 생성

CloudWatch에는 Amazon 관계형 데이터베이스 서비스 및 Amazon DocumentDB(MongoDB 호환)의 성능 개선 도우미 카운터 지표를 CloudWatch로 가져오는 데 사용할 수 있는 DB_PERF_INSIGHTS 지표 수학 함수가 포함되어 있습니다. 또한 DB_PERF_INSIGHTS는 1분 미만의 간격으로 DBLoad 지표를 가져옵니다. 이러한 지표에 대해 CloudWatch 경보를 설정할 수 있습니다.

Amazon RDS 성능 개선 도우미에 대한 자세한 내용은 성능 개선 도우미를 통한 Amazon RDS 모니터링을 참조하세요.

Amazon DocumentDB 성능 개선 도우미에 대한 자세한 내용은 Monitoring with Performance Insights를 참조하세요.

DB_PERF_INSIGHTS 함수를 기반으로 하는 경보에는 이상 탐지가 지원되지 않습니다.

> **Note**
>
> DB_PERF_INSIGHTS가 검색하는 분 단위 이하의 고분해능 지표는 DBLoad 지표 또는 운영 체제 지표(고분해능에서 Enhanced Monitoring을 사용하도록 설정한 경우)에만 적용됩니다. Amazon RDS 확장 모니터링에 대한 자세한 내용은 Enhanced Monitoring을 사용하여 OS 지표 모니터링을 참조하세요.
>
> DB_PERF_INSIGHTS 함수를 사용하여 고분해능 경보를 생성할 수 있습니다. 고분해능 경보의 최대 평가 범위는 3시간입니다. CloudWatch 콘솔을 사용하여 DB_PERF_INSIGHTS 함수로 검색된 지표를 원하는 시간 범위에 대해 그래프로 표시할 수 있습니다.

성능 개선 도우미 지표를 기반으로 경보를 만들려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택한 다음 모든 경보(All alarms)를 선택합니다.
3. 경보 생성을 선택하세요.
4. 지표 선택을 선택합니다.

5. 수학 추가 드롭다운을 선택한 다음 목록에서 데이터베이스 성능 지표, DB_PERF_INSIGHTS를 선택합니다.

   DB_PERF_INSIGHTS를 선택하면 수학 표현식을 적용하거나 편집할 수 있는 수학 표현식 상자가 나타납니다.

6. 수학 표현식 상자에 DB_PERF_INSIGHTS 수학 표현식을 입력한 다음 적용을 선택합니다.

   예제: **`DB_PERF_INSIGHTS('RDS', 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', 'os.cpuUtilization.user.avg')`**

   > **Important**
   >
   > DB_PERF_INSIGHTS 수학 표현식을 사용할 때는 데이터베이스의 고유 데이터베이스 리소스 ID를 지정해야 합니다. 이는 데이터베이스 식별자와는 다릅니다. Amazon RDS 콘솔에서 데이터베이스 리소스 ID를 찾으려면 DB 인스턴스를 선택하여 세부 정보를 확인합니다. 그런 다음 구성 탭을 선택합니다. 그러면 리소스 ID가 구성 섹션에 표시됩니다.

   지표 연산에 사용할 수 있는 DB_PERF_INSIGHTS 함수 및 기타 함수에 대한 자세한 내용은 지표 수학 구문 및 함수 섹션을 참조하세요.

7. 지표 선택을 선택하세요.

   선택한 수학 표현식에 대한 그래프와 기타 정보가 표시된 지표 및 조건 지정 페이지가 표시됩니다.

8. 표현식이 ***### ##*** 항상에서 표현식이 임곗값보다 크거나, 작거나, 같아야 하는지 여부를 지정합니다. than...에서 임곗값을 지정합니다.

9. 추가 구성을 선택합니다. 경보에 대한 데이터 포인트에서 경보를 트리거하기 위해 평가 기간(데이터 포인트)이 ALARM 상태로 유지해야 하는 기간을 지정합니다. 두 값이 일치하는 경우 다수의 연속 기간이 위반되면 ALARM 상태가 되는 경보가 생성됩니다.

   N 중 M 경보를 생성하려면 두 번째 값에 지정한 값보다 낮은 값을 첫 번째 값에 지정합니다. 자세한 내용은 경보 평가 단원을 참조하세요.

10. 누락 데이터 처리에서 일부 데이터 포인트가 누락된 경우 경보가 어떻게 동작할지 선택합니다. 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성 단원을 참조하세요.

11. 다음(Next)을 선택합니다.

12. 알림(Notification)에서 경보가 ALARM 상태, OK 상태 또는 INSUFFICIENT_DATA 상태일 때 알릴 SNS 주제를 선택합니다.

경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

경보에서 알림을 보내지 않게 하려면 제거를 선택합니다.

13. 경보가 Auto Scaling, EC2, Lambda 또는 Systems Manager 작업을 수행하도록 하려면 해당 버튼을 선택하고 경보 상태와 수행할 작업을 선택합니다. Lambda 함수를 경보 작업으로 선택하는 경우 함수 이름 또는 ARN을 지정하고 필요에 따라 함수의 특정 버전을 선택할 수 있습니다.

    경보는 ALARM 상태가 될 때만 Systems Manager 작업을 수행할 수 있습니다. Systems Manager 작업에 대한 자세한 내용은 경보에서 OpsItem을 생성하도록 CloudWatch 구성 및 인시던트 생성 단원을 참조하세요.

    > **Note**
    >
    > SSM Incident Manager 작업을 수행하는 경보를 생성하려면 특정 권한이 있어야 합니다. 자세한 내용은 AWS Systems Manager Incident Manager의 자격 증명 기반 정책 예 단원을 참조하세요.

14. 마친 후에는 다음을 선택합니다.
15. 경보 이름 및 설명을 입력합니다. 다음을 선택합니다.

    이름에는 UTF-8 문자만 포함해야 하며 ASCII 제어 문자는 포함할 수 없습니다. 설명에 마크다운 서식을 포함할 수 있으며, 이는 CloudWatch 콘솔에서 경보 세부 정보 탭에만 표시됩니다. 마크다운은 런북이나 기타 내부 리소스에 대한 링크를 추가하는 데 유용할 수 있습니다.

16. 미리 보기 및 생성에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성을 선택합니다.

EC2 인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성

Amazon CloudWatch 경보 작업을 사용하면 EC2 인스턴스를 자동으로 중지, 종료, 재부팅 또는 복구하는 경보를 생성할 수 있습니다. 인스턴스를 더 이상 실행할 필요가 없을 때 중지 또는 종료 작업을 사용하여 비용을 절약할 수 있습니다. 재부팅 및 복구 작업을 사용하면 시스템 장애가 발생할 경우 인스턴스를 자동으로 재부팅하거나 새로운 하드웨어로 인스턴스를 복구할 수 있습니다.

인스턴스를 자동으로 중지하거나 종료해야 하는 경우는 매우 다양합니다. 예를 들어 일정 기간 동안 실행한 다음 작업을 완료하는 일괄 급여 처리 작업 또는 과학적 컴퓨팅 작업 전용 인스턴스가 있을 수 있습니다. 이러한 인스턴스를 유휴 상태로 유지하여 비용이 발생하도록 하는 대신 중지하거나 종료하면 비용을 절감할 수 있습니다. 경보 작업 중지와 종료 간의 주요 차이는 나중에 다시 실행해야 하는 경우

중지된 인스턴스는 쉽게 다시 시작할 수 있다는 점입니다. 또한 동일한 인스턴스 ID 및 루트 볼륨을 유지할 수 있습니다. 그러나 종료된 인스턴스를 다시 시작할 수는 없습니다. 대신, 새 인스턴스를 시작해야 합니다.

Amazon CloudWatch에서 제공하는 기본 및 세부 모니터링 지표(AWS/EC2 네임스페이스)를 비롯한 Amazon EC2 인스턴스별 지표 및 InstanceId 값이 실행 중인 유효한 Amazon EC2 인스턴스를 참조하는 동안에 "InstanceId=" 차원을 포함하는 사용자 지정 지표에 대해 설정된 경보에 중지, 종료 또는 재부팅 작업을 추가할 수 있습니다. `StatusCheckFailed_Instance`를 제외하고 인스턴스당 Amazon EC2 지표에 설정된 경보에 복구 작업을 추가할 수도 있습니다.

인스턴스를 재부팅, 중지 또는 종료할 수 있는 CloudWatch 경보 작업을 설정하려면 서비스 연결 IAM 역할인 AWSServiceRoleForCloudWatchEvents를 사용해야 합니다. AWSServiceRoleForCloudWatchEvents IAM 역할을 사용하면 AWS가 사용자를 대신하여 경보 작업을 수행할 수 있습니다.

CloudWatch Events에 대한 서비스 연결 역할을 생성하려면 다음 명령을 사용합니다.

```
aws iam create-service-linked-role --aws-service-name events.amazonaws.com
```

콘솔 지원

CloudWatch 콘솔 또는 Amazon EC2 콘솔을 사용하여 경보를 생성할 수 있습니다. 이 설명서의 절차에서는 CloudWatch 콘솔을 사용합니다. Amazon EC2 콘솔을 사용하는 절차는 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성 단원을 참조하세요.

권한

AWS Identity and Access Management(IAM) 계정을 사용하여 EC2 작업 또는 Systems Manager OpsItem 작업을 수행하는 경보를 생성하거나 수정하는 경우 `iam:CreateServiceLinkedRole` 권한이 있어야 합니다.

내용

- Amazon CloudWatch 경보에 중지 작업 추가하기
- Amazon CloudWatch 경보에 종료 작업 추가하기
- Amazon CloudWatch 경보에 재부팅 작업 추가하기
- Amazon CloudWatch 경보에 복구 작업 추가하기

- 트리거한 경보 및 작업 기록 보기

Amazon CloudWatch 경보에 중지 작업 추가하기

특정 임계값에 도달한 경우 Amazon EC2 인스턴스를 중지하는 경보를 만들 수 있습니다. 예를 들어 개발 또는 테스트 인스턴스를 실행한 후 종료하는 것을 잊을 수 있습니다. 24시간 동안 평균 CPU 사용률이 10% 아래로 떨어지는 경우 즉, 유휴 상태로 더 이상 사용되지 않는 경우 트리거되는 경보를 만들 수 있습니다. 필요에 맞춰 임계값 및 기간을 조정할 수 있습니다. 또한 경보가 트리거되면 이메일을 받을 수 있도록 SNS 알림을 추가할 수 있습니다.

Amazon Elastic Block Store 볼륨을 루트 디바이스로 사용하는 Amazon EC2 인스턴스는 중지하거나 종료할 수 있지만, 인스턴스 스토어를 루트 디바이스로 사용하는 인스턴스는 종료만 할 수 있습니다.

Amazon CloudWatch 콘솔을 사용하여 유휴 인스턴스를 중지하는 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms) 모든 경보(All Alarms)를 선택합니다.
3. 경보 생성(Create alarm)을 선택하세요.
4. 지표 선택(Select Metric)을 선택합니다.
5. AWS 네임스페이스(namespaces)에서 EC2를 선택합니다.
6. 다음을 따릅니다.
    a. 인스턴스별 지표(Per-Instance Metrics)를 선택합니다.
    b. 해당 인스턴스와 CPUUtilization 지표가 있는 행에서 확인란을 선택합니다.
    c. 그래프로 표시된 지표 탭을 선택합니다.
    d. 통계의 경우 평균(Average)을 선택합니다.
    e. 기간(예: **1 Hour**)을 선택합니다.
    f. 지표 선택을 선택하세요.
7. 경보 정의(Define Alarm) 단계에서 다음을 수행합니다.
    a. 조건(Conditions)에서 정적(Static)을 선택합니다.
    b. CPUUtilization이 있는 경우 항상(Whenever CPUUtilization is)에서 낮음(Lower)을 선택합니다.
    c. 보다(than)에 **10**을 입력합니다.
    d. 다음(Next)을 선택합니다.

e. 알림(Notification)의 알림 보내기(Send notification to)에서 기존 SNS 주제를 선택하거나 새로 만듭니다.

   SNS 주제를 생성하려면 새 목록(New list)을 선택합니다. 알림 보내기(Send notification to)에 SNS 주제의 이름을 입력합니다(예: Stop_EC2_Instance). 이메일 목록(Email list)에서 경보가 ALARM 상태로 변경될 때 알림을 받을 이메일 주소 목록을 쉼표로 구분하여 입력합니다. 각 이메일 주소로 주제 구독 확인 이메일이 전송됩니다. 수신자가 구독을 확인해야만 이 이메일 주소로 알림이 전송될 수 있습니다.

f. EC2 작업 추가(Add EC2 Action)를 선택합니다.

g. 경보 상태 트리거(Alarm state trigger)에서 경보(In Alarm)를 선택합니다. 다음 작업 수행(Take the following action)에서 이 인스턴스 중지(Stop this instance)를 선택합니다.

h. 다음을 선택합니다.

i. 경보 이름 및 설명을 입력합니다. 이름은 ASCII 문자만 포함해야 합니다. 그리고 다음(Next)을 선택합니다.

j. 미리 보기 및 생성(Preview and create)에서 정보 및 조건이 원하는 내용인지 확인한 다음 경보 생성(Create alarm)을 선택합니다.

Amazon CloudWatch 경보에 종료 작업 추가하기

인스턴스에 대해 종료 보호가 비활성화되어 있는 경우에 한해서 특정 임계값에 도달한 경우 EC2 인스턴스를 자동으로 종료하는 경보를 만들 수 있습니다. 예를 들어 인스턴스의 작업 완료 후 해당 인스턴스가 다시 필요 없는 경우 인스턴스를 종료하려고 할 수 있습니다. 나중에 인스턴스를 사용하려는 경우에는 종료하지 말고 중지해야 합니다. 인스턴스에 대한 종료 보호 사용 및 사용 중지에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에 대한 종료 보호 사용 설정 단원을 참조하세요.

Amazon CloudWatch 콘솔을 사용하여 유휴 인스턴스를 종료하는 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms), 경보 생성(Create Alarm)을 선택합니다.
3. 지표 선택(Select Metric) 단계에서 다음을 수행합니다.

   a. EC2 지표(EC2 Metrics)에서 인스턴스별 지표(Per-Instance Metrics)를 선택합니다.

   b. 해당 인스턴스와 CPUUtilization 지표가 있는 행을 선택합니다.

   c. 통계의 경우 평균(Average)을 선택합니다.

   d. 기간(예: **1 Hour**)을 선택합니다.

e. 다음을 선택합니다.

4. 경보 정의(Define Alarm) 단계에서 다음을 수행합니다.

   a. 경보 임계값(Alarm Threshold)에 경보의 고유 이름(예: Terminate EC2 instance)과 경보에 대한 설명(예: CPU 유휴 시간이 너무 길어서 EC2 인스턴스 종료)을 입력합니다. 경보 이름은 ASCII 문자만 포함해야 합니다.

   b. Whenever의 is에서 <를 선택하고 **10**을 입력합니다. 기간(for)에 연속 기간으로 **24**를 입력합니다.

      경보 미리 보기(Alarm Preview) 아래에 임계값이 그래픽으로 표시됩니다.

   c. 알림(Notification)의 알림 보내기(Send notification to)에서 기존 SNS 주제를 선택하거나 새로 만듭니다.

      SNS 주제를 생성하려면 새 목록(New list)을 선택합니다. 알림 보내기(Send notification to)에 SNS 주제의 이름을 입력합니다(예: Terminate_EC2_Instance). 이메일 목록(Email list)에서 경보가 ALARM 상태로 변경될 때 알림을 받을 이메일 주소 목록을 쉼표로 구분하여 입력합니다. 각 이메일 주소로 주제 구독 확인 이메일이 전송됩니다. 수신자가 구독을 확인해야만 이 이메일 주소로 알림이 전송될 수 있습니다.

   d. EC2 작업(EC2 Action)을 선택합니다.

   e. 이 경보가 발생할 경우 항상(Whenever this alarm)에 상태가 ALARM입니다(State is ALARM)를 선택합니다. 이 작업을 수행(Take this action)에서 인스턴스 종료(Terminate this instance)를 선택합니다.

   f. 경보 생성을 선택합니다.

Amazon CloudWatch 경보에 재부팅 작업 추가하기

Amazon EC2 인스턴스를 모니터링하고 인스턴스를 자동으로 재부팅하는 Amazon CloudWatch 경보를 만들 수 있습니다. 재부팅 경보 작업은 인스턴스 상태 확인 오류(복구 경보 작업은 시스템 상태 확인 오류에 적합)에 권장됩니다. 인스턴스 재부팅은 운영 체제 재부팅과 같습니다. 대부분의 경우 인스턴스를 재부팅하는 데는 몇 분 밖에 걸리지 않습니다. 인스턴스를 재부팅하는 경우 동일한 물리적 호스트에 남아 있으므로 퍼블릭 DNS 이름, 프라이빗 IP 주소 및 인스턴스 스토어 볼륨의 모든 데이터가 유지됩니다.

인스턴스를 재부팅해도 인스턴스를 중지했다가 다시 시작할 때와는 달리 새 인스턴스 청구 시간이 시작되지 않습니다. 인스턴스 재부팅에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 재부팅 단원을 참조하세요.

> **Important**
>
> 재부팅과 복원 작업 간에 경합 상태가 발생하지 않도록 하려면 재부팅 경보와 복원 경보에 동일한 평가 기간을 설정하지 마십시오. 재부팅 경보를 각각 1분의 평가 기간 3회로 설정하는 것이 좋습니다.

Amazon CloudWatch 콘솔을 사용하여 인스턴스를 재부팅하는 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms), 경보 생성(Create Alarm)을 선택합니다.
3. 지표 선택(Select Metric) 단계에서 다음을 수행합니다.
   a. EC2 지표(EC2 Metrics)에서 인스턴스별 지표(Per-Instance Metrics)를 선택합니다.
   b. 해당 인스턴스와 StatusCheckFailed_Instance 지표가 있는 행을 선택합니다.
   c. 통계의 경우 최소(Minimum)를 선택합니다.
   d. 기간(예: **1 Minute**)을 선택합니다.
   e. 다음을 선택합니다.
4. 경보 정의(Define Alarm) 단계에서 다음을 수행합니다.
   a. 경보 임계값(Alarm Threshold)에 경보의 고유 이름(예: Reboot EC2 instance)과 경보에 대한 설명(예: CPU 유휴 시간이 너무 길어서 EC2 인스턴스 재부팅)을 입력합니다. 경보 이름은 ASCII 문자만 포함해야 합니다.
   b. Whenever의 is에서 >를 선택하고 **0**을 입력합니다. 기간(for)에 연속 기간으로 **3**를 입력합니다.

      경보 미리 보기(Alarm Preview) 아래에 임계값이 그래픽으로 표시됩니다.
   c. 알림(Notification)의 알림 보내기(Send notification to)에서 기존 SNS 주제를 선택하거나 새로 만듭니다.

      SNS 주제를 생성하려면 새 목록(New list)을 선택합니다. 알림 보내기(Send notification to)에 SNS 주제의 이름을 입력합니다(예: Reboot_EC2_Instance). 이메일 목록(Email list)에서 경보가 ALARM 상태로 변경될 때 알림을 받을 이메일 주소 목록을 쉼표로 구분하여 입력합니다. 각 이메일 주소로 주제 구독 확인 이메일이 전송됩니다. 수신자가 구독을 확인해야만 이 이메일 주소로 알림이 전송될 수 있습니다.
   d. EC2 작업(EC2 Action)을 선택합니다.

e. 이 경보가 발생할 경우 항상(Whenever this alarm)에 상태가 ALARM입니다(State is ALARM)를 선택합니다. 이 작업을 수행(Take this action)에서 인스턴스 재부팅(Reboot this instance)을 선택합니다.

f. 경보 생성을 선택합니다.

Amazon CloudWatch 경보에 복구 작업 추가하기

Amazon EC2 인스턴스를 모니터링하고 기본 하드웨어 장애나 복구에 AWS 개입이 필요한 문제로 인해 인스턴스가 손상된 경우 인스턴스를 자동으로 복구하는 Amazon CloudWatch 경보를 생성할 수 있습니다. 종료한 인스턴스는 복구할 수 없습니다. 복구된 인스턴스는 인스턴스 ID, 프라이빗 IP 주소, 탄력적 IP 주소 및 모든 인스턴스 메타데이터를 포함하여 원본 인스턴스와 동일합니다.

`StatusCheckFailed_System` 경보가 트리거되고 복구 작업이 시작되면 경보를 생성하고 복구 작업을 연결했을 때 선택한 Amazon SNS 주제로 알림을 받게 됩니다. 인스턴스 복구 중에 인스턴스를 재부팅할 때 인스턴스가 마이그레이션되고 모든 인 메모리 데이터가 손실됩니다. 프로세스가 완료되면 해당 경보를 위해 구성해 둔 SNS 주제로 정보가 게시됩니다. 이 SNS 주제에 가입되어 있는 사람은 누구나 복구 시도 상태와 세부 지침이 포함된 이메일 알림을 받게 됩니다. 복구된 인스턴스에서 인스턴스를 재부팅하라는 메시지가 나타납니다.

복구 작업은 `StatusCheckFailed_Instance`가 아닌 `StatusCheckFailed_System`을 통해서만 사용할 수 있습니다.

시스템 상태 확인이 실패하게 되는 문제의 예를 들면 다음과 같습니다.

- 네트워크 연결 끊김
- 시스템 전원 중단
- 물리적 호스트의 소프트웨어 문제
- 네트워크 연결성에 영향을 주는 물리적 호스트의 하드웨어 문제

복구 작업은 일부 인스턴스 유형에서만 지원됩니다. 지원되는 인스턴스 유형과 기타 요구 사항에 대한 자세한 내용은 인스턴스 복구와 요구 사항을 참조하세요.

> **Important**
>
> 재부팅과 복원 작업 간에 경합 상태가 발생하지 않도록 하려면 재부팅 경보와 복원 경보에 동일한 평가 기간을 설정하지 마십시오. 복원 경보는 각각 1분의 평가 기간 2회로 설정하고 재부팅 경보는 각각 1분의 평가 기간 3회로 설정하는 것이 좋습니다.

Amazon CloudWatch 콘솔을 사용하여 인스턴스를 복구하는 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms), 경보 생성(Create Alarm)을 선택합니다.
3. 지표 선택(Select Metric) 단계에서 다음을 수행합니다.
   a. EC2 지표(EC2 Metrics)에서 인스턴스별 지표(Per-Instance Metrics)를 선택합니다.
   b. 해당 인스턴스와 StatusCheckFailed_System 지표가 있는 행을 선택합니다.
   c. 통계의 경우 최소(Minimum)를 선택합니다.
   d. 기간(예: **1 Minute**)을 선택합니다.

      > **Important**
      > 재부팅과 복원 작업 간에 경합 상태가 발생하지 않도록 하려면 재부팅 경보와 복원 경보에 동일한 평가 기간을 설정하지 마십시오. 복구 경보는 각각 1분의 평가 기간 2회로 설정하는 것이 좋습니다.

   e. 다음을 선택합니다.
4. 경보 정의(Define Alarm) 단계에서 다음을 수행합니다.
   a. 경보 임계값(Alarm Threshold)에 경보의 고유 이름(예: Recover EC2 instance)과 경보에 대한 설명(예: 상태 확인 실패 시 EC2 인스턴스 복구)을 입력합니다. 경보 이름은 ASCII 문자만 포함해야 합니다.
   b. Whenever의 is에서 **>**를 선택하고 **0**을 입력합니다. 기간(for)에 연속 기간으로 **2**를 입력합니다.
   c. 알림(Notification)의 알림 보내기(Send notification to)에서 기존 SNS 주제를 선택하거나 새로 만듭니다.

      SNS 주제를 생성하려면 새 목록(New list)을 선택합니다. 알림 보내기(Send notification to)에 SNS 주제의 이름을 입력합니다(예: Recover_EC2_Instance). 이메일 목록(Email list)에서 경보가 ALARM 상태로 변경될 때 알림을 받을 이메일 주소 목록을 쉼표로 구분하여 입력합니다. 각 이메일 주소로 주제 구독 확인 이메일이 전송됩니다. 수신자가 구독을 확인해야만 이 이메일 주소로 알림이 전송될 수 있습니다.
   d. EC2 작업(EC2 Action)을 선택합니다.

e. 이 경보가 발생할 경우 항상(Whenever this alarm)에 상태가 ALARM입니다(State is ALARM)를 선택합니다. 이 작업을 수행(Take this action)에서 인스턴스 복구(Recover this instance)를 선택합니다.

f. 경보 생성을 선택합니다.

트리거한 경보 및 작업 기록 보기

Amazon CloudWatch 콘솔을 사용하여 경보 및 작업 기록을 볼 수 있습니다. Amazon CloudWatch는 지난 30일 동안의 경보 및 작업 기록을 보관합니다.

트리거된 경보 및 작업 기록을 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택한 후 특정 경보를 선택합니다.
3. 가장 최근의 상태 변화와 함께 시간 및 지표 값을 보려면 세부 정보(Details)를 선택합니다.
4. 최신 기록 항목을 보려면 내역(History)을 선택합니다.

경보 및 태그 지정

태그는 리소스를 정리하고 분류하는 데 도움이 될 수 있는 키-값 쌍입니다. 특정 태그 값이 있는 리소스만 액세스하거나 변경할 수 있는 권한을 사용자에게 부여하여 사용자 권한 범위를 지정할 수도 있습니다. 리소스 태그 지정에 대한 추가적인 일반 정보는 AWS 리소스 태그 지정을 참조하세요.

다음 목록은 CloudWatch 경보에서 태그 지정이 작동하는 방식에 관한 일부 정보를 자세히 설명합니다.

- CloudWatch 리소스에 대한 태그를 설정하거나 업데이트하려면 `cloudwatch:TagResource` 권한이 있는 계정에 로그인해야 합니다. 예를 들어 경보를 생성하고 여기에 태그를 설정하려면 the `cloudwatch:PutMetricAlarm` 권한 외에 `cloudwatch:TagResource` 권한이 있어야 합니다. 조직 내에서 CloudWatch 리소스 생성 또는 업데이트 작업을 해야 하는 모든 사용자에게 `cloudwatch:TagResource` 권한이 있는지 확인하는 것이 좋습니다.
- 태그는 태그 기반 권한 부여 제어에 사용할 수 있습니다. 예를 들어, 태그를 기반으로 CloudWatch 호출을 특정 리소스로 제한하는 조건을 IAM 사용자 또는 역할 권한에 포함할 수 있습니다. 하지만 다음과 같은 점을 주의해야 합니다.
  - `aws:`로 시작하는 이름을 가진 태그는 태그 기반 권한 부여 제어에 사용할 수 없습니다.
  - 복합 경보는 태그 기반 권한 부여를 지원하지 않습니다.

Application Signals

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

CloudWatch Application Signals를 사용하면 AWS에서 애플리케이션을 자동으로 계측할 수 있으므로 현재 애플리케이션 상태를 모니터링하고 비즈니스 목표에 따라 장기적인 애플리케이션 성능을 추적할 수 있습니다. Application Signals는 애플리케이션, 서비스 및 종속성에 대한 통합 애플리케이션 중심 보기를 제공하고 애플리케이션 상태를 모니터링하고 분류하는 데 도움이 됩니다.

- Application Signals를 활성화하여 애플리케이션에서 지표와 트레이스를 자동으로 수집하고 호출 볼륨, 가용성, 지연 시간, 장애 및 오류와 같은 주요 지표를 표시합니다. 사용자 지정 코드를 작성하거나 대시보드를 만들지 않고도 현재 운영 상태를 빠르게 확인하고 분류하고 애플리케이션이 장기 성능 목표를 달성하고 있는지 여부를 확인합니다.
- Application Signals를 사용하여 서비스 수준 목표(SLO)를 생성하고 모니터링합니다. Application Signals가 수집하는 새로운 표준 애플리케이션 지표를 포함하여 CloudWatch 지표와 관련된 SLO 상태를 쉽게 생성하고 추적합니다. 서비스 목록 및 토폴로지 맵 내에서 애플리케이션 서비스의 서비스 수준 지표(SLI) 상태를 확인하고 추적할 수 있습니다. 경보를 생성하여 SLO를 추적하고 Application Signals가 수집하는 새로운 표준 애플리케이션 지표를 추적합니다.
- Application Signals가 자동으로 검색하는 애플리케이션 토폴로지 맵을 봅니다. 이를 통해 애플리케이션, 종속성 및 연결성을 시각적으로 확인할 수 있습니다.
- Application Signals는 CloudWatch RUM, CloudWatch Synthetics canary, AWS Service Catalog AppRegistry 및 Amazon EC2 Auto Scaling 등과 함께 작동하여 대시보드 및 맵 내에 클라이언트 페이지, Synthetics canary 및 애플리케이션 이름을 표시합니다.

일일 애플리케이션 모니터링에 Application Signals 사용

CloudWatch 콘솔 내에서 Application Signals를 일일 애플리케이션 모니터링의 일부로 사용합니다.

1. 서비스에 대한 서비스 수준 목표(SLO)를 생성한 경우 서비스 수준 목표(SLO) 페이지부터 시작합니다. 이를 통해 가장 중요한 서비스와 작업의 상태를 즉시 확인할 수 있습니다. SLO의 서비스 또는 작업 이름을 선택하여 서비스 세부 정보 페이지를 열고 문제를 해결하는 동안 자세한 서비스 정보를 확인합니다.

2. 서비스 페이지를 열어 모든 서비스의 요약을 확인하고 장애 발생률이 가장 높거나 지연 시간이 가장 긴 서비스를 빠르게 확인합니다. SLO를 생성한 경우 서비스 테이블을 보고 비정상 서비스 수준 지표(SLI)가 있는 서비스를 확인합니다. 특정 서비스가 비정상 상태인 경우 서비스를 선택하여 서비스 세부 정보 페이지를 열고 서비스 작업, 종속성, Synthetics canary 및 클라이언트 요청을 확인합니다. 그래프에서 지점을 선택하면 상관관계가 있는 트레이스를 확인할 수 있으므로 문제를 해결하고 운영 문제의 근본 원인을 식별할 수 있습니다.
3. 새 서비스가 배포되었거나 종속성이 변경된 경우 서비스 맵을 열어 애플리케이션 토폴로지를 검사합니다. 클라이언트, Synthetics canary, 서비스 및 종속성 간의 관계를 보여주는 애플리케이션 맵을 봅니다. SLI 상태를 빠르게 확인하고, 호출 볼륨, 장애 발생률, 지연 시간과 같은 주요 지표를 확인하고, 서비스 세부 정보 페이지에서 자세한 정보를 자세히 확인합니다.

Application Signals를 사용하면 요금이 부과됩니다. CloudWatch 요금에 대한 내용은 Amazon CloudWatch 요금을 참조하세요.

> **Note**
>
> CloudWatch Synthetics, CloudWatch RUM 또는 CloudWatch Evidently를 사용하기 위해 Application Signals를 활성화할 필요는 없습니다. 그러나 Synthetics와 CloudWatch RUM은 Application Signals와 함께 작동하여 이러한 기능을 함께 사용할 경우 이점을 제공합니다.

지원되는 언어 및 아키텍처

현재 Application Signals는 Java 및 Python 애플리케이션을 지원합니다.

Application Signals는 Amazon EKS, Amazon ECS 및 Amazon EC2에서 지원 및 테스트됩니다. Amazon EKS 클러스터에서는 서비스 및 클러스터의 이름을 자동으로 검색합니다. 다른 아키텍처에서는 Application Signals에 대해 해당 서비스를 활성화할 때 서비스 및 환경의 이름을 제공해야 합니다.

Amazon EC2에서 Application Signals를 활성화하는 지침은 CloudWatch 에이전트와 AWS Distro for OpenTelemetry를 지원하는 모든 아키텍처에서 유효해야 합니다. 그러나 Amazon ECS 및 Amazon EC2 이외의 아키텍처에서는 이 지침을 테스트하지 않았습니다.

지원되는 리전

이번 평가판 릴리스의 경우 Application Signals는 다음 리전에서 지원됩니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)

- 미국 서부(오레곤)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 유럽(아일랜드)

평가판 SDK

SDK의 평가판 버전을 다운로드할 수 있습니다.

> **Warning**
>
> API 작업 및 파라미터는 Application Signals가 정식 출시되기 전에 변경될 수 있습니다. 이러한 변경 사항은 주요 변경 사항일 수 있습니다. SDK의 평가판 버전을 프로덕션 용도로 사용하지 마세요.

평가판 SDK를 설치하려면 먼저 최신 버전의 AWS CLI 버전 2를 설치하거나 업데이트해야 합니다. 자세한 내용은 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.

그리고 다음 명령을 사용하여 Amazon S3 버킷에서 SDK zip 파일을 다운로드한 다음, 콘텐츠를 추출합니다. 각 SDK zip 파일에는 SDK 지침과 API 설명서가 포함되어 있습니다.

> **Note**
>
> SDK는 여러 프로그래밍 언어로 제공되므로 Application Signals API를 이러한 프로그래밍 언어에 사용할 수 있습니다. 하지만 애플리케이션을 자동 계측하여 Application Signals로 데이터를 전송하는 기능은 Java 및 Python 애플리케이션에서만 지원됩니다.

- Java V2 SDK: `aws s3 cp s3://application-signals-preview-sdk/awsJavaSdkV2.zip ./`
- JavaScript V3 SDK: `aws s3 cp s3://application-signals-preview-sdk/jsSdkV3.zip ./`
- JavaScript V2 SDK: `aws s3 cp s3://application-signals-preview-sdk/jsSdkV2.zip ./`
- Python SDK: `aws s3 cp s3://application-signals-preview-sdk/pythonSdk.zip ./`

- Kotlin SDK: `aws s3 cp s3://application-signals-preview-sdk/kotlin.zip ./`
- Android SDK: `aws s3 cp s3://application-signals-preview-sdk/andriod.zip ./`
- C++ SDK: `aws s3 cp s3://application-signals-preview-sdk/awsCppSdk.zip ./`
- PHP SDK: `aws s3 cp s3://application-signals-preview-sdk/awsSdkPhp.zip ./`
- Ruby SDK: `aws s3 cp s3://application-signals-preview-sdk/awsSdkRuby.zip ./`
- Go V2 SDK: `aws s3 cp s3://application-signals-preview-sdk/awsSdkGoV2.zip ./`
- Go V1 SDK: `aws s3 cp s3://application-signals-preview-sdk/go.zip ./`
- iOS SDK: `aws s3 cp s3://application-signals-preview-sdk/iOS.zip ./`

주제

Application Signals에 필요한 권한

⚠ Application Signals는 Amazon CloudWatch의 평가판 릴리스에 있으며 변경될 수 있습니다.

이 섹션에서는 Application Signals를 활성화, 관리 및 운영하는 데 필요한 권한을 설명합니다.

Application Signals를 활성화하고 관리할 수 있는 권한

Application Signals를 관리하려면 다음 권한으로 로그인해야 합니다.

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

```
"Statement": [
   {
       "Sid": "CloudWatchApplicationSignalsFullAccessPermissions",
       "Effect": "Allow",
       "Action": "application-signals:*",
       "Resource": "*"
   },
   {
       "Sid": "CloudWatchApplicationSignalsAlarmsPermissions",
       "Effect": "Allow",
       "Action": [
           "cloudwatch:DescribeAlarms"
       ],
       "Resource": "*"
   },
   {
       "Sid": "CloudWatchApplicationSignalsMetricsPermissions",
       "Effect": "Allow",
       "Action": [
           "cloudwatch:GetMetricData",
           "cloudwatch:ListMetrics"
       ],
       "Resource": "*"
   },
   {
       "Sid": "CloudWatchApplicationSignalsLogGroupPermissions",
       "Effect": "Allow",
       "Action": [
           "logs:StartQuery",
           "logs:DescribeMetricFilters"
       ],
       "Resource": "arn:aws:logs:*:*:log-group:/aws/application-signals/data:*"
   },
   {
       "Sid": "CloudWatchApplicationSignalsLogsPermissions",
       "Effect": "Allow",
       "Action": [
           "logs:GetQueryResults",
           "logs:StopQuery"
       ],
       "Resource": "*"
   },
   {
       "Sid": "CloudWatchApplicationSignalsSyntheticsPermissions",
```

```
        "Effect": "Allow",
        "Action": [
            "synthetics:DescribeCanaries",
            "synthetics:DescribeCanariesLastRun",
            "synthetics:GetCanaryRuns"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsRumPermissions",
        "Effect": "Allow",
        "Action": [
            "rum:BatchCreateRumMetricDefinitions",
            "rum:BatchDeleteRumMetricDefinitions",
            "rum:BatchGetRumMetricDefinitions",
            "rum:GetAppMonitor",
            "rum:GetAppMonitorData",
            "rum:ListAppMonitors",
            "rum:PutRumMetricsDestination",
            "rum:UpdateRumMetricDefinition"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsXrayPermissions",
        "Effect": "Allow",
        "Action": [
            "xray:GetTraceSummaries"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsPutMetricAlarmPermissions",
        "Effect": "Allow",
        "Action": "cloudwatch:PutMetricAlarm",
        "Resource": [
            "arn:aws:cloudwatch:*:*:alarm:SLO-AttainmentGoalAlarm-*",
            "arn:aws:cloudwatch:*:*:alarm:SLO-WarningAlarm-*",
            "arn:aws:cloudwatch:*:*:alarm:SLI-HealthAlarm-*"
        ]
    },
    {
        "Sid": "CloudWatchApplicationSignalsCreateServiceLinkedRolePermissions",
        "Effect": "Allow",
```

```
        "Action": "iam:CreateServiceLinkedRole",
        "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals",
        "Condition": {
            "StringLike": {
                "iam:AWSServiceName": "application-signals.cloudwatch.amazonaws.com"
            }
        }
    },
    {
        "Sid": "CloudWatchApplicationSignalsGetRolePermissions",
        "Effect": "Allow",
        "Action": "iam:GetRole",
        "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals"
    },
    {
        "Sid": "CloudWatchApplicationSignalsSnsWritePermissions",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:Subscribe"
        ],
        "Resource": "arn:aws:sns:*:*:cloudwatch-application-signals-*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsSnsReadPermissions",
        "Effect": "Allow",
        "Action": "sns:ListTopics",
        "Resource": "*"
    }
 ]
}
```

Amazon EC2에서 Application Signals, Kubernetes 또는 사용자 지정 아키텍처를 활성화하려면 사용자 지정 설정으로 다른 플랫폼에서 Application Signals활성화를 참조하세요. Amazon CloudWatch Observability EKS 추가 기능을 사용하여 Amazon EKS에서 Application Signals를 활성화하고 관리하려면 다음 권한이 필요합니다.

> **Important**
>
> 이러한 권한에는 Resource "*"가 있는 iam:PassRole과 Resource "*"가 있는 eks:CreateAddon이 포함됩니다. 이들은 강력한 권한이므로 주의해서 부여해야 합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
"Sid": "CloudWatchApplicationSignalsEksAddonManagementPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:AccessKubernetesApi",
                "eks:CreateAddon",
                "eks:DescribeAddon",
                "eks:DescribeAddonConfiguration",
                "eks:DescribeAddonVersions",
                "eks:DescribeCluster",
                "eks:DescribeUpdate",
                "eks:ListAddons",
                "eks:ListClusters",
                "eks:ListUpdates",
                "iam:ListRoles",
                "iam:PassRole"
            ],
            "Resource": "*"
        },
        {
"Sid":
 "CloudWatchApplicationSignalsEksCloudWatchObservabilityAddonManagementPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:DeleteAddon",
                "eks:UpdateAddon"
            ],
            "Resource": "arn:aws:eks:*:*:addon/*/amazon-cloudwatch-observability/*"
        }
    ]
}
```

Application Signals 대시보드에는 SLO가 연결된 AWS Service Catalog AppRegistry 애플리케이션이 표시됩니다. SLO 페이지에서 이러한 애플리케이션을 보려면 다음 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsTaggingReadPermissions",
            "Effect": "Allow",
            "Action": "tag:GetResources",
            "Resource": "*"
        }
    ]
}
```

Application Signals 운영

Application Signals를 사용하여 서비스와 SLO를 모니터링하는 서비스 운영자는 다음 읽기 전용 권한을 가진 계정에 로그인해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsReadOnlyAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "application-signals:BatchGet*",
                "application-signals:Get*",
                "application-signals:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsGetRolePermissions",
            "Effect": "Allow",
            "Action": "iam:GetRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals"
        },
        {
        "Sid": "CloudWatchApplicationSignalsLogGroupPermissions",
```

```
        "Effect": "Allow",
        "Action": [
            "logs:StartQuery",
            "logs:DescribeMetricFilters"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/application-signals/data:*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsLogsPermissions",
        "Effect": "Allow",
        "Action": [
            "logs:GetQueryResults",
            "logs:StopQuery"
        ],
        "Resource": "*"
    },
        {
            "Sid": "CloudWatchApplicationSignalsAlarmsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsMetricsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsSyntheticsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "synthetics:DescribeCanaries",
                "synthetics:DescribeCanariesLastRun",
                "synthetics:GetCanaryRuns"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "CloudWatchApplicationSignalsRumReadPermissions",
            "Effect": "Allow",
            "Action": [
                "rum:BatchGetRumMetricDefinitions",
                "rum:GetAppMonitor",
                "rum:GetAppMonitorData",
                "rum:ListAppMonitors"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsXrayReadPermissions",
            "Effect": "Allow",
            "Action": [
                "xray:GetTraceSummaries"
            ],
            "Resource": "*"
        }
    ]
}
```

Application Signals 대시보드에서 SLO와 연결된 AWS Service Catalog AppRegistry 애플리케이션을 확인하려면 다음 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsTaggingReadPermissions",
            "Effect": "Allow",
            "Action": "tag:GetResources",
            "Resource": "*"
        }
    ]
}
```

Amazon CloudWatch Observability EKS 추가 기능을 사용하여 Amazon EKS에서 Application Signals를 활성화했는지 확인하려면 다음 권한이 필요합니다.

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

```
        {
            "Sid": "CloudWatchApplicationSignalsEksReadPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:ListAddons",
                "eks:ListClusters"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsEksDescribeAddonReadPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:DescribeAddon"
            ],
            "Resource": "arn:aws:eks:*:*:addon/*/amazon-cloudwatch-observability/*"
        }
    ]
}
```

Application Signals 활성화

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

이 섹션의 항목에서는 환경에서 CloudWatch Application Signals를 활성화하는 방법을 설명합니다. Application Signals는 콘솔을 사용한 설정 워크플로를 통해 Amazon EKS 클러스터에서 지원됩니다. Amazon EC2를 비롯한 다른 플랫폼에서도 사용자 지정 설치 프로세스를 통해 지원됩니다.

주제

Application Signals 지원 시스템

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

Application Signals는 Amazon EKS, Amazon ECS 및 Amazon EC2에서 지원 및 테스트됩니다. Amazon EC2에서 Application Signals를 활성화하는 지침은 CloudWatch 에이전트와 AWS Distro for OpenTelemetry를 지원하는 모든 플랫폼에서 유효해야 하지만, 다른 플랫폼에서는 이 지침이 테스트되지 않았습니다.

Java 호환성

Application Signals는 Java 애플리케이션을 지원하며 AWS Distro for OpenTelemetry와 동일한 Java 라이브러리 및 프레임워크를 지원합니다. 자세한 내용은 지원되는 라이브러리, 프레임워크, 애플리케이션 서버 및 JVM을 참조하세요.

JVM 버전 8, 11 및 17이 지원됩니다.

Python 호환성

Application Signals는 AWS Distro for OpenTelemetry와 동일한 라이브러리 및 프레임워크를 지원합니다. 자세한 내용은 opentelemetry-python-contrib에서 지원되는 패키지를 참조하세요.

Python 버전 3.8 이상이 지원됩니다.

Python 애플리케이션에서 Application Signals를 활성화하기 전에 다음 사항에 유의하세요.

- 일부 컨테이너화된 애플리케이션에서는 PYTHONPATH 환경 변수가 누락되면 애플리케이션이 시작되지 않을 수 있습니다. 이 문제를 해결하려면 PYTHONPATH 환경 변수를 애플리케이션의 작업 디렉터리 위치로 설정해야 합니다. 이는 OpenTelemetry 자동 계측과 관련하여 알려진 문제 때문입니다. 이 문제에 대한 자세한 내용은 Python autoinstrumentation setting of PYTHONPATH is not compliant를 참조하세요.
- Django 애플리케이션의 경우 추가 필수 구성이 있으며, OpenTelemetry Python 설명서에서 설명합니다.
  - `--noreload` 플래그를 사용하여 자동 재로드를 방지합니다.
  - `DJANGO_SETTINGS_MODULE` 환경 변수를 Django 애플리케이션 `settings.py` 파일의 위치로 설정합니다. 이렇게 하면 OpenTelemetry가 올바르게 액세스하여 Django 설정에 통합할 수 있습니다.

OpenTelemetry 호환성 고려 사항

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

CloudWatch Application Signals로 애플리케이션을 온보딩하려면 먼저 애플리케이션에서 기존 애플리케이션 성능 모니터링 솔루션을 완전히 제거하는 것이 좋습니다. 여기에는 모든 계측 코드 및 구성 제거가 포함됩니다.

Application Signals는 OpenTelemetry 계측을 사용하지만 기존 OpenTelemetry 계측 또는 구성과의 호환성이 보장되지는 않습니다. 최상의 시나리오에서는 사용자 지정 지표와 같은 일부 OpenTelemetry 기능을 유지할 수 있습니다. 하지만 세부 정보는 다음 섹션을 참조하세요.

이미 OpenTelemetry를 사용하고 있는 경우 고려할 사항

애플리케이션에서 이미 OpenTelemetry를 사용하고 있는 경우 이 섹션의 나머지 부분에는 Application Signals와의 호환성을 확보하기 위한 중요한 정보가 포함되어 있습니다.

- Application Signals에 대해 애플리케이션을 활성화하기 전에 애플리케이션에서 OpenTelemetry 기반의 다른 자동 계측 에이전트 삽입을 제거해야 합니다. 이렇게 하면 구성 충돌을 방지하는 데 도움이 됩니다. 호환되는 OpenTelemetry API를 Application Signals와 함께 사용하여 수동 계측을 계속 사용할 수 있습니다.
- 수동 계측을 사용하여 애플리케이션에서 사용자 지정 범위 또는 지표를 생성하는 경우 계측의 복잡도에 따라 Application Signals를 활성화하면 데이터 생성이 중단되거나 기타 원치 않는 동작이 발생할 수 있습니다. OpenTelemetry에서 사용 가능한 일부 구성(이 섹션 뒷부분의 표에 언급된 구성 제외)을 사용하여 기존 지표 또는 범위의 원하는 동작을 유지할 수 있습니다. 이러한 구성에 대한 자세한 내용은 OpenTelemetry 설명서의 SDK Configuration을 참조하세요.

  예를 들어, `OTEL_EXPORTER_OTLP_METRICS_ENDPOINT` 구성 및 자체 관리형 OpenTelemetry Collector 인스턴스를 사용하면 사용자 지정 지표를 원하는 대상으로 계속 전송할 수 있습니다.
- 일부 환경 변수 또는 시스템 속성은 Application Signals와 함께 사용해서는 안 되지만, 표의 지침을 따르는 한 다른 환경 변수 또는 시스템 속성은 사용할 수 있습니다. 자세한 내용은 다음 표를 참조하세요.

환경 변수	Application Signals 사용 시 권장 사항
일반 환경 변수	
OTEL_SDK_DISABLED	true로 설정하면 안 됩니다.
OTEL_TRACES_EXPORTER	otlp로 설정해야 합니다.
OTEL_EXPORTER_OTLP_ENDPOINT	사용하면 안 됩니다.
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	사용하면 안 됩니다.
OTEL_ATTRIBUTE_COUNT_LIMIT	설정된 경우 CloudWatch Application Signals에 의해 추가된 범위 속성을 약 10개 더 포함할 수 있을 만큼 높게 설정해야 합니다.
OTEL_PROPAGATORS	설정된 경우 종료 추적에 대한 xray를 포함해야 합니다.
OTEL_TRACES_SAMPLER	설정된 경우 X-Ray 중앙 집중식 샘플링을 사용하도록 xray여야 합니다. 로컬 샘플링을 사용하려면 이를 parentbased_traceidratio 로 설정하고 OTEL_TRACES_SAMPLER_ARG 에 샘플링 속도를 지정합니다.
OTEL_TRACES_SAMPLER_ARG	기본값인 X-Ray 중앙 집중식 트레이스 샘플을 사용하는 경우 이 변수를 사용해서는 안 됩니다. 로컬 샘플링을 대신 사용하는 경우 이 변수에 샘플링 속도를 설정합니다. 예를 들어 샘플링 속도가 5%인 경우 0.05입니다.
Java 환경 변수	
OTEL_JAVA_ENABLED_RESOURCE_PROVIDERS	설정된 경우 AWS 리소스 탐지기를 포함해야 합니다.

환경 변수	Application Signals 사용 시 권장 사항
Python 환경 변수	
`OTEL_PYTHON_CONFIGURATOR`	사용되는 경우 `aws_configurator` 로 설정해야 함
`OTEL_PYTHON_DISTRO`	사용되는 경우 `aws_distro` 로 설정해야 함

Amazon EKS 클러스터에서 Application Signals 활성화

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

CloudWatch Application Signals는 Amazon EKS 클러스터에서 실행되는 Java 및 Python 애플리케이션에 대해 지원됩니다. Amazon EKS 클러스터의 애플리케이션에 대해 Application Signals를 활성화하려는 경우 다음 두 가지 옵션이 있습니다.

- 기존 Amazon EKS 클러스터의 애플리케이션에 대해 Application Signals를 활성화하려면 서비스를 사용하여 Amazon EKS 클러스터에서 Application Signals 활성화의 단계를 사용합니다.
- 샘플 애플리케이션을 사용하여 비프로덕션 환경에서 Application Signals를 사용해 보려면 샘플 앱을 사용하여 새 Amazon EKS 클러스터에서 Application Signals 활성화의 지침을 따릅니다. 이 워크플로는 AWS에서 제공하는 스크립트를 사용하여 새로운 Amazon EKS 클러스터를 생성하고 Application Signals에 대해 활성화된 샘플 애플리케이션을 설치합니다. 이를 통해 Application Signals의 엔드포인트 간 기능을 확인하고 테스트할 수 있습니다.

주제

서비스를 사용하여 Amazon EKS 클러스터에서 Application Signals 활성화

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

기존 Amazon EKS 클러스터의 애플리케이션에 대해 CloudWatch Application Signals를 활성화하려면 이 섹션의 지침을 따릅니다.

⚠ **Important**

Application Signals에 대해 활성화하려는 애플리케이션과 함께 이미 OpenTelemetry를 사용하고 있다면 Application Signals를 활성화하기 전에 OpenTelemetry 호환성 고려 사항 섹션을 참조하세요.

기존 Amazon EKS 클러스터의 애플리케이션에 Application Signals 활성화

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 서비스를 선택합니다.
3. 이 계정에서 아직 Application Signals를 활성화하지 않은 경우 서비스를 검색하는 데 필요한 권한을 Application Signals에 부여해야 합니다. 그렇게 하려면 다음을 수행합니다. 계정에 대해 한 번만 수행하면 됩니다.

   a. 서비스 검색 시작을 선택합니다.

   b. 확인란을 선택하고 서비스 검색 시작을 선택합니다.

      계정에서 처음으로 이 단계를 완료하면 AWSServiceRoleForCloudWatchApplicationSignals 서비스 역할이 생성됩니다. 이 역할은 Application Signals에 다음 권한을 부여합니다.

      - `xray:GetServiceGraph`
      - `logs:StartQuery`
      - `logs:GetQueryResults`
      - `cloudwatch:GetMetricData`
      - `cloudwatch:ListMetrics`
      - `tag:GetResources`

이에 대한 자세한 내용은 CloudWatch Application Signals에 대한 서비스 연결 역할 권한 섹션을 참조하세요.

4. Application Signals 활성화를 선택합니다.
5. 플랫폼 지정에서 EKS를 선택합니다.
6. EKS 클러스터 선택에서 Application Signals를 활성화하려는 클러스터를 선택합니다.
7. 이 클러스터에서 Amazon CloudWatch Observability EKS 추가 기능을 아직 활성화하지 않은 경우 활성화하라는 메시지가 나타납니다. 이런 경우 다음을 수행합니다.

   a. CloudWatch Observability EKS 추가 기능 추가를 선택합니다. Amazon EKS 콘솔이 표시됩니다.

   b. Amazon CloudWatch Observability 확인란을 선택하고 다음을 선택합니다.

      CloudWatch Observability EKS 추가 기능은 Amazon EKS에 대한 향상된 관찰성을 통해 Application Signals와 CloudWatch Container Insights를 모두 활성화합니다. Container Insights에 대한 자세한 정보는 Container Insights 섹션을 참조하세요.

   c. 설치할 최신 버전의 추가 기능을 선택합니다.

   d. 추가 기능에 사용할 IAM 역할을 선택합니다. 노드에서 상속을 선택하는 경우 워커 노드에서 사용하는 IAM 역할에 올바른 권한을 추가합니다. *my-worker-node-role*을 Kubernetes 워커 노드에서 사용하는 IAM 역할로 대체합니다.

      ```
      aws iam attach-role-policy \
      --role-name my-worker-node-role \
      --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
      --policy-arn arn:aws:iam::aws:policy/AWSXRayWriteOnlyAccess
      ```

   e. 추가 기능을 사용할 서비스 역할을 생성하려면 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 CloudWatch 에이전트 설치 섹션을 참조하세요.

   f. 다음을 선택하고 화면의 정보를 확인한 다음, 생성을 선택합니다.

   g. 다음 화면에서 CloudWatch Application Signals 활성화를 선택하여 CloudWatch 콘솔로 돌아가 프로세스를 완료합니다.

8. Application Signals에서 애플리케이션을 활성화하는 두 가지 옵션이 있습니다. 일관성을 위해 클러스터당 하나의 옵션을 선택하는 것이 좋습니다.

   - 콘솔 옵션이 더 간단합니다. 이 방법을 사용하면 포드가 즉시 재시작됩니다.

- 매니페스트 파일에 주석 추가 방법을 사용하면 포드를 언제 재시작할지 더 효과적으로 제어할 수 있고, 모니터링을 중앙에 집중시키고 싶지 않은 경우 보다 분산된 방식으로 모니터링을 관리할 수 있습니다.

Console

콘솔 옵션은 Amazon CloudWatch Observability EKS 추가 기능의 고급 구성을 사용하여 서비스에 대해 Application Signals를 설정합니다. 추가 기능에 대한 자세한 내용은 (선택 사항) 추가 구성 섹션을 참조하세요.

워크로드 및 네임스페이스 목록이 표시되지 않는 경우 이 클러스터에서 이를 볼 수 있는 적절한 권한이 있는지 확인하세요. 자세한 내용은 필수 권한 섹션을 참조하세요.

단일 워크로드 또는 전체 네임스페이스를 모니터링할 수 있습니다.

단일 워크로드 모니터링:

1. 모니터링할 워크로드 옆의 확인란을 선택합니다.
2. 워크로드의 언어를 선택합니다. Python 애플리케이션의 경우 계속하기 전에 애플리케이션이 필수 사전 조건을 준수하는지 확인하세요. 자세한 내용은 Application Signals가 활성화된 후 Python 애플리케이션이 시작되지 않음 단원을 참조하십시오.
3. 완료를 선택합니다. Amazon CloudWatch Observability EKS 추가 기능은 ADOT(AWS Distro for OpenTelemetry) 자동 계측 SDK를 포드에 즉시 삽입하고 포드 재시작을 트리거하여 애플리케이션 지표 및 추적 수집을 활성화합니다.

전체 네임스페이스 모니터링:

1. 모니터링할 네임스페이스 옆의 확인란을 선택합니다.
2. 워크로드의 언어를 선택합니다. 이는 현재 배포되어 있거나 향후 배포될 예정인지 여부와 무관하게 이 네임스페이스의 모든 워크로드에 적용됩니다. Python 애플리케이션의 경우 계속하기 전에 애플리케이션이 필수 사전 조건을 준수하는지 확인하세요. 자세한 내용은 Application Signals가 활성화된 후 Python 애플리케이션이 시작되지 않음 단원을 참조하십시오.
3. 완료를 선택합니다. Amazon CloudWatch Observability EKS 추가 기능은 ADOT(AWS Distro for OpenTelemetry) 자동 계측 SDK를 포드에 즉시 삽입하고 포드 재시작을 트리거하여 애플리케이션 지표 및 추적 수집을 활성화합니다.

다른 Amazon EKS 클러스터에서 Application Signals를 활성화하려면 서비스 화면에서 Application Signals 활성화를 선택합니다.

Annotate manifest file

CloudWatch 콘솔에서 서비스 모니터링 섹션은 클러스터의 매니페스트 YAML에 주석을 추가해야 한다고 설명합니다. 이 주석을 추가하면 Application Signals에 지표, 트레이스 및 로그를 전송하도록 애플리케이션이 자동 계측됩니다.

주석에는 다음 2가지 옵션이 있습니다.

- 워크로드에 주석 추가는 클러스터의 단일 워크로드를 자동으로 계측합니다.
- 네임스페이스에 주석 추가는 선택한 네임스페이스에 배포된 모든 워크로드를 자동으로 계측합니다.

이러한 옵션 중 하나를 선택하고 적절한 단계를 따릅니다.

- 단일 워크로드에 주석 추가:
  1. 워크로드에 주석 추가를 선택합니다.
  2. 워크로드 매니페스트 파일의 PodTemplate 섹션에 다음 줄 중 하나를 붙여넣습니다.
     - Java 워크로드: `annotations: instrumentation.opentelemetry.io/inject-java: "true"`
     - Python 워크로드: `annotations: instrumentation.opentelemetry.io/inject-python: "true"`

       Python 애플리케이션의 경우 추가 필수 구성이 있습니다. 자세한 내용은 Application Signals가 활성화된 후 Python 애플리케이션이 시작되지 않음 단원을 참조하십시오.
  3. 터미널에서 `kubectl apply -f` *`your_deployment_yaml`*을 입력하여 변경 사항을 적용합니다.
- 네임스페이스의 모든 워크로드에 주석 추가:
  1. 네임스페이스에 주석 추가를 선택합니다.
  2. 네임스페이스 매니페스트 파일의 메타데이터 섹션에 다음 줄 중 하나를 붙여넣습니다. 네임스페이스에 Java 및 Python 워크로드가 모두 포함된 경우 이 두 줄을 모두 네임스페이스 매니페스트 파일에 붙여넣습니다.

- 네임스페이스에 Java 워크로드가 있는 경우: `annotations: instrumentation.opentelemetry.io/inject-java: "true"`
- 네임스페이스에 Python 워크로드가 있는 경우: `annotations: instrumentation.opentelemetry.io/inject-python: "true"`

  Python 애플리케이션의 경우 추가 필수 구성이 있습니다. 자세한 내용은 Application Signals가 활성화된 후 Python 애플리케이션이 시작되지 않음 단원을 참조하십시오.

3. 터미널에서 `kubectl apply -f your_namespace_yaml`을 입력하여 변경 사항을 적용합니다.
4. 터미널에서 네임스페이스의 모든 포드를 다시 시작하는 명령을 입력합니다. 배포 워크로드를 다시 시작하는 명령의 예는 `kubectl rollout restart deployment -n namespace_name`입니다.

9. 완료 시 서비스 보기를 선택합니다. 그러면 Application Signals가 수집하는 데이터를 확인할 수 있는 Application Signals 서비스 뷰로 이동합니다. 데이터가 표시되는 데 몇 분 정도 걸릴 수 있습니다.

   다른 Amazon EKS 클러스터에서 Application Signals를 활성화하려면 서비스 화면에서 Application Signals 활성화를 선택합니다.

   서비스 뷰에 대한 자세한 내용은 Application Signals를 사용하여 애플리케이션의 운영 상태 모니터링 섹션을 참조하세요.

> **Note**
>
> Application Signals에서 Python 애플리케이션을 활성화할 때 염두에 두어야 할 몇 가지 고려 사항을 확인했습니다. 자세한 내용은 Application Signals가 활성화된 후 Python 애플리케이션이 시작되지 않음 단원을 참조하십시오.

샘플 앱을 사용하여 새 Amazon EKS 클러스터에서 Application Signals 활성화

> **Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.**

자체 애플리케이션을 계측하기 전에 샘플 앱에서 CloudWatch Application Signals를 사용해 보려면 이 섹션의 지침을 따르세요. 이 지침은 스크립트를 사용하여 Amazon EKS 클러스터를 생성하고, 샘플 애플리케이션을 설치하고, Application Signals와 함께 작동하도록 샘플 애플리케이션을 계측하는 데 도움이 됩니다.

샘플 애플리케이션은 4개의 마이크로서비스로 구성된 Spring 'Pet Clinic' 애플리케이션입니다. 이러한 서비스는 Amazon EC2의 Amazon EKS에서 실행되며 Application Signals 활성화 스크립트를 활용하여 Java 또는 Python 자동 계측 에이전트로 클러스터를 활성화합니다.

요구 사항

- 현재 Application Signals는 Java 및 Python 애플리케이션만 모니터링합니다.
- 인스턴스에 AWS CLI가 설치되어 있어야 합니다. AWS CLI 버전 2를 권장하지만 버전 1도 사용할 수 있습니다. AWS CLI 설치에 대한 자세한 내용은 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.
- 이 섹션의 스크립트는 Linux 및 macOS 환경에서 실행되도록 설계되었습니다. Windows 인스턴스의 경우 AWS Cloud9 환경을 사용하여 이러한 스크립트를 실행하는 것이 좋습니다. AWS Cloud9에 대한 자세한 내용은 AWS Cloud9란 무엇인가요?를 참조하세요.
- 지원되는 버전의 `kubectl`을 설치합니다. Amazon EKS 클러스터 컨트롤 플레인과 마이너 버전 차이가 1 이내인 `kubectl` 버전을 사용해야 합니다. 예를 들어, 1.26 `kubectl` 클라이언트는 Kubernetes 1.25, 1.26 및 1.27 클러스터로 작업해야 합니다. Amazon EKS 클러스터가 이미 있는 경우 `kubectl`에 대한 AWS 보안 인증을 구성해야 할 수 있습니다. 자세한 내용은 Amazon EKS 클러스터용 `kubeconfig` 파일 생성 또는 업데이트를 참조하세요.
- `eksctl`을 설치합니다. AWS CLI은 `eksctl`를 사용하여 AWS와 상호 작용합니다. 즉, AWS CLI와 동일한 AWS 보안 인증을 사용합니다. 자세한 내용은 `eksctl` 설치 또는 업데이트를 참조하세요.
- `jq`를 설치합니다. Application Signals 활성화 스크립트를 실행하려면 `jq`가 필요합니다. 자세한 내용은 다운로드 jq를 참조하세요.

1단계: 스크립트 다운로드

샘플 앱으로 CloudWatch Application Signals를 설정하는 스크립트를 다운로드하려면 압축된 GitHub 프로젝트 파일을 로컬 드라이브에 다운로드하고 압축을 풀거나 GitHub 프로젝트를 복제할 수 있습니다.

프로젝트를 복제하려면 터미널 창을 열고 지정된 작업 디렉터리에 다음 Git 명령을 입력합니다.

```
git clone https://github.com/aws-observability/application-signals-demo.git
```

2단계: 샘플 애플리케이션 구축 및 배포

샘플 애플리케이션 이미지를 구축하고 푸시하려면 다음 지침을 따릅니다.

3단계: Application Signals와 샘플 애플리케이션 배포 및 활성화

다음 단계를 완료하기 전에 샘플 앱을 사용하여 새 Amazon EKS 클러스터에서 Application Signals 활성화에 나열된 요구 사항을 완료했는지 확인합니다.

Application Signals와 샘플 애플리케이션 배포 및 활성화

1. 온보딩 스크립트의 압축을 푼 로컬 터미널에서 다음 명령을 입력합니다. *new-cluster-name*을 새 클러스터에 사용할 이름으로 바꿉니다. *region-name*을 us-west-1과 같은 AWS 리전 이름으로 바꿉니다.

   이 명령은 Application Signals가 활성화된 새 Amazon EKS 클러스터에서 실행되는 샘플 앱을 설정합니다.

   ```
   # assuming the current working directory is 'onboarding'
   # this script sets up a new cluster, enables Application Signals, and deploys the
   # sample application
   cd application-signals-demo/scripts/eks/appsignals/one-step && ./setup.sh new-
   cluster-name region-name
   ```

   설치 스크립트는 실행하는 데 약 30분이 걸리며 다음과 같은 작업을 수행합니다.

   - 지정된 리전에 새 Amazon EKS 클러스터를 생성합니다.
   - Application Signals(arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess 및 arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy)에 필요한 IAM 권한을 생성합니다.
   - CloudWatch 에이전트를 설치하고 CloudWatch 지표 및 X-Ray 트레이스용 샘플 애플리케이션을 자동으로 계측하여 Application Signals를 활성화합니다.
   - PetClinic Spring 샘플 애플리케이션을 동일한 Amazon EKS 클러스터에 배포합니다.
   - pc-add-vist, pc-create-owners, pc-visit-pet, pc-visit-vet, pc-clinic-traffic이라는 5개의 CloudWatch Synthetics canary를 생성합니다. 이러한 canary는 1분 간격으로 실행되어 샘플 앱에 대한 합성 트래픽을 생성하고 Application Signals에 Synthetics canary가 어떻게 나타나는지 보여줍니다.
   - PetClinic 애플리케이션을 위한 네 가지 서비스 수준 목표(SLO)를 다음과 같은 이름으로 생성합니다.

    - 소유자 검색 가능 여부
    - 소유자 검색 지연 시간
    - 소유자 등록 가능 여부
    - 소유자 등록 지연 시간
  - Application Signals에 다음 권한을 부여하는 사용자 지정 신뢰 정책을 사용하여 필요한 IAM 역할을 생성합니다.
    - `cloudwatch:PutMetricData`
    - `cloudwatch:GetMetricData`
    - `xray:GetServiceGraph`
    - `logs:StartQuery`
    - `logs:GetQueryResults`
2. (선택 사항) PetClinic 샘플 애플리케이션의 소스 코드를 검토하려는 경우 루트 폴더에서 해당 코드를 찾을 수 있습니다.

```
- application-signals-demo
  - spring-petclinic-admin-server
  - spring-petclinic-api-gateway
  - spring-petclinic-config-server
  - spring-petclinic-customers-service
  - spring-petclinic-discovery-server
  - spring-petclinic-vets-service
  - spring-petclinic-visits-service
```

3. 배포된 PetClinic 샘플 애플리케이션을 보려면 다음 명령을 실행하여 URL을 찾습니다.

```
kubectl get ingress
```

4단계: 샘플 애플리케이션 모니터링

이전 섹션의 단계를 완료하여 Amazon EKS 클러스터를 생성하고 샘플 애플리케이션을 배포한 후 Application Signals를 사용하여 애플리케이션을 모니터링할 수 있습니다.

> **Note**
>
> Application Signals 콘솔에서 채우기를 시작하려면 일부 트래픽이 샘플 애플리케이션에 도달해야 합니다. 이전 단계에서 샘플 애플리케이션으로 트래픽을 생성하는 CloudWatch Synthetics canary가 생성되었습니다.

서비스 상태 모니터링

활성화되면 CloudWatch Application Signals는 추가 설정 없이 서비스 목록을 자동으로 검색하여 채웁니다.

검색된 서비스 목록 보기 및 해당 서비스의 상태 모니터링

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, 서비스를 선택합니다.
3. 서비스, 해당 작업 및 종속성을 보려면 목록에서 서비스 중 하나의 이름을 선택합니다.

   이 통합 애플리케이션 중심 보기를 통해 사용자가 서비스와 상호 작용하는 방식을 전체적으로 파악할 수 있습니다. 이를 통해 성능 이상이 발생할 경우 문제를 분류할 수 있습니다. 서비스 뷰에 대한 자세한 내용은 Application Signals를 사용하여 애플리케이션의 운영 상태 모니터링 섹션을 참조하세요.
4. 서비스 작업 탭을 선택하여 해당 서비스 작업에 대한 표준 애플리케이션 지표를 봅니다. 작업은 예를 들어, 서비스가 호출하는 API 작업입니다.

   그런 다음, 해당 서비스의 단일 작업에 대한 그래프를 보려면 해당 작업 이름을 선택합니다.
5. 종속성 탭을 선택하여 각 종속성에 대한 중요한 애플리케이션 지표와 함께 애플리케이션의 종속성을 봅니다. 종속성에는 애플리케이션이 직접적으로 호출하는 AWS 서비스 및 타사 서비스가 포함됩니다.
6. 서비스 세부 정보 페이지에서 상관관계가 있는 트레이스를 보려면 테이블 위의 세 그래프 중 하나에서 데이터 포인트를 선택합니다. 그러면 새 창에 해당 기간의 필터링된 트레이스가 채워집니다. 이러한 트레이스는 선택한 그래프를 기준으로 정렬 및 필터링됩니다. 예를 들어, 지연 시간 그래프를 선택한 경우 트레이스는 서비스 응답 시간을 기준으로 정렬됩니다.
7. CloudWatch 콘솔 탐색 창에서 SLO를 선택합니다. 스크립트가 샘플 애플리케이션에 대해 생성한 SLO를 볼 수 있습니다. SLO에 대한 자세한 내용은 서비스 수준 목표(SLO) 섹션을 참조하세요.

(선택 사항) 5단계: 정리

애플리케이션 신호 테스트를 마치면 Amazon에서 제공하는 스크립트를 사용하여 샘플 애플리케이션을 위해 계정에 생성된 아티팩트를 정리하고 삭제할 수 있습니다. 정리를 수행하려면 다음 명령을 입력합니다. *new-cluster-name*을 샘플 앱용으로 만든 클러스터의 이름으로 바꾸고, *region*-name을 us-west-1 등의 AWS 리전 이름으로 바꿉니다.

```
cd application-signals-demo/scripts/eks/appsignals/one-step && ./cleanup.sh new-
cluster-name region-name
```

사용자 지정 설정으로 다른 플랫폼에서 Application Signals 활성화

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

이 섹션의 사용자 지정 설정 단계를 사용하여 Amazon EKS 이외의 플랫폼에서 CloudWatch Application Signals를 활성화합니다. 이러한 아키텍처에서는 CloudWatch 에이전트와 AWS Distro for OpenTelemetry를 직접 설치하고 구성합니다.

이러한 아키텍처에서 Application Signals는 서비스 또는 해당 클러스터 또는 호스트의 이름을 자동으로 검색하지 않습니다. 사용자 지정 설정 중에 이러한 이름을 지정해야 하며, 지정하는 이름은 Application Signals 대시보드에 표시됩니다.

주제

- 사용자 지정 설정을 사용하여 Amazon ECS에서 Application Signals를 활성화합니다.
- 사용자 지정 설정을 사용하여 Amazon EC2와 기타 플랫폼에서 Application Signals를 활성화합니다.

사용자 지정 설정을 사용하여 Amazon ECS에서 Application Signals를 활성화합니다.

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

이 사용자 지정 설정 지침을 사용하여 Amazon ECS의 애플리케이션을 CloudWatch Application Signals에 온보딩합니다. CloudWatch 에이전트와 AWS Distro for OpenTelemetry를 직접 설치하고 구성합니다.

Amazon ECS 클러스터에서 Application Signals는 서비스 또는 서비스가 실행되는 클러스터의 이름을 자동으로 검색하지 않습니다. 사용자 지정 설정 중에 이러한 이름을 지정해야 하며, 지정하는 이름은 Application Signals 대시보드에 표시됩니다.

> **Important**
>
> `awsvpc` 네트워크 모드만 지원됩니다.

1단계: 계정에서 Application Signals 활성화

이 계정에서 아직 Application Signals를 활성화하지 않은 경우 서비스를 검색하는 데 필요한 권한을 Application Signals에 부여해야 합니다. 그렇게 하려면 다음을 수행합니다. 계정에 대해 한 번만 수행하면 됩니다.

애플리케이션에 대해 Application Signals 활성화

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 서비스를 선택합니다.
3. 서비스 검색 시작을 선택합니다.
4. 확인란을 선택하고 서비스 검색 시작을 선택합니다.

   계정에서 처음으로 이 단계를 완료하면 AWSServiceRoleForCloudWatchApplicationSignals 서비스 역할이 생성됩니다. 이 역할은 Application Signals에 다음 권한을 부여합니다.

   - `xray:GetServiceGraph`
   - `logs:StartQuery`
   - `logs:GetQueryResults`
   - `cloudwatch:GetMetricData`
   - `cloudwatch:ListMetrics`
   - `tag:GetResources`

이에 대한 자세한 내용은 CloudWatch Application Signals에 대한 서비스 연결 역할 권한 섹션을 참조하세요.

2단계 - IAM 역할 생성

두 IAM 역할을 생성해야 합니다. 이러한 역할을 이미 생성한 경우 권한을 추가해야 할 수 있습니다.

- ECS 태스크 역할 - 컨테이너는 이 역할을 사용하여 실행됩니다. CloudWatchAgentServerPolicy 및 AWSXRayWriteOnlyAccess 외에도 권한은 애플리케이션에 필요한 것이어야 합니다.
- ECS 태스크 실행 역할 - Amazon ECS는 이 역할을 사용하여 컨테이너를 시작하고 실행합니다. 이 역할을 이미 생성한 경우 AmazonSSMReadOnlyAccess, AmazonECSTaskExecutionRolePolicy 및 CloudWatchAgentServerPolicy 정책을 해당 역할에 연결합니다.

  Amazon ECS에서 사용할 더 민감한 데이터를 저장해야 하는 경우 자세한 내용은 민감한 데이터 지정 단원을 참조하세요.

IAM 역할 생성에 대한 자세한 내용은 IAM 역할 생성 단원을 참조하세요.

3단계: CloudWatch 에이전트 구성 준비

먼저 Application Signals가 활성화된 에이전트 구성을 준비합니다. 이렇게 하려면 `/tmp/ecs-cwagent.json`이라는 로컬 파일을 생성합니다.

```
{
  "traces": {
    "traces_collected": {
      "app_signals": {}
    }
  },
  "logs": {
    "metrics_collected": {
      "app_signals": {}
    }
  }
}
```

그런 다음, 이 구성을 SSM Parameter Store에 업로드합니다. 이를 위해 다음 명령을 입력합니다. 파일에서 *$REGION*을 실제 리전 이름으로 바꿉니다.

```
aws ssm put-parameter \
--name "ecs-cwagent" \
--type "String" \
--value "`cat /tmp/ecs-cwagent.json`" \
--region "$REGION"
```

4단계: CloudWatch 에이전트를 사용하여 애플리케이션 계측

다음 단계는 CloudWatch Application Signals에 대해 애플리케이션을 계측하는 것입니다.

Java

CloudWatch 에이전트를 사용하여 Amazon ECS에서 애플리케이션 계측

1. 먼저 바인드 탑재를 지정합니다. 다음 단계에서 이 볼륨은 컨테이너 간에 파일을 공유하는 데 사용됩니다. 이 바인드 탑재는 이 절차의 뒷부분에서 사용합니다.

   ```
   "volumes": [
     {
       "name": "opentelemetry-auto-instrumentation"
     }
   ]
   ```

2. CloudWatch 에이전트 사이드카 정의를 추가합니다. 이렇게 하려면 애플리케이션의 작업 정의에 `ecs-cwagent`라는 새 컨테이너를 추가합니다. *$REGION*을 실제 리전 이름으로 바꿉니다. Amazon Elastic Container Registry의 최신 CloudWatch 컨테이너 이미지 경로로 바꿉니다. 자세한 내용은 Amazon ECR의 cloudwatch-agent를 참조하세요.

   ```
   {
     "name": "ecs-cwagent",
     "image": "$IMAGE",
     "essential": true,
     "secrets": [
       {
         "name": "CW_CONFIG_CONTENT",
         "valueFrom": "ecs-cwagent"
       }
     ],
     "logConfiguration": {
       "logDriver": "awslogs",
       "options": {
         "awslogs-create-group": "true",
   ```

```
      "awslogs-group": "/ecs/ecs-cwagent",
      "awslogs-region": "$REGION",
      "awslogs-stream-prefix": "ecs"
    }
  }
}
```

3. 애플리케이션의 작업 정의에 새 컨테이너 `init`를 추가합니다. *$IMAGE*를 AWS Distro for OpenTelemetry Amazon ECR 이미지 리포지토리의 최신 이미지로 바꿉니다.

```
{
  "name": "init",
  "image": "$IMAGE",
  "essential": false,
  "command": [
    "cp",
    "/javaagent.jar",
    "/otel-auto-instrumentation/javaagent.jar"
  ],
  "mountPoints": [
    {
      "sourceVolume": "opentelemetry-auto-instrumentation",
      "containerPath": "/otel-auto-instrumentation",
      "readOnly": false
    }
  ]
}
```

4. 애플리케이션 컨테이너에 다음 환경 변수를 추가합니다. 자세한 내용을 알아보려면 다음 섹션을 참조하세요.

환경 변수	Application Signals를 활성화하도록 설정
`OTEL_RESOURCE_ATTRIBUTES`	`$SVC_NAME` 을 애플리케이션 이름으로 바꿉니다. Application Signals 대시보드에 애플리케이션 이름으로 표시됩니다. `$HOST_ENV` 를 애플리케이션이 실행되는 호스트 환경으로 바꿉니다. 이는 Application Signals 대시보드에서 애플리케이션의 호스팅 위치 환경으로 표시됩니다.

환경 변수	Application Signals를 활성화하도록 설정
OTEL_AWS_APP_SIGNALS_ENABLED	Application Signals SpanMetrics Processor를 활성화하려면 true로 설정합니다.
OTEL_METRICS_EXPORTER	다른 지표 내보내기를 비활성화하려면 none으로 설정합니다.
OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT	CloudWatch 사이드카로 지표를 전송하려면 http://127.0.0.1:4315 로 설정합니다.
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	CloudWatch 사이드카로 트레이스를 전송하려면 http://127.0.0.1:4315 로 설정합니다.
OTEL_TRACES_SAMPLER	X-Ray를 트레이스 샘플러로 정의합니다.
OTEL_PROPAGATORS	전파자 중 하나로 X-Ray를 추가합니다.
JAVA_TOOL_OPTIONS	AWS Distro for OpenTelemetry Java 에이전트를 삽입합니다.

5. 이 절차의 1단계에서 정의한 볼륨을 opentelemetry-auto-instrumentation을 마운트합니다.

   Java 애플리케이션의 경우 다음을 사용합니다.

```
{
  "name": "app",
   ...
  "environment": [
    {
      "name": "OTEL_RESOURCE_ATTRIBUTES",
      "value": "aws.hostedin.environment=$HOST_ENV,service.name=$SVC_NAME"
    },
    {
      "name": "OTEL_AWS_APP_SIGNALS_ENABLED",
      "value": "true"
    },
    {
```

```
      "name": "OTEL_METRICS_EXPORTER",
      "value": "none"
    },
    {
      "name": "JAVA_TOOL_OPTIONS",
      "value": " -javaagent:/otel-auto-instrumentation/javaagent.jar"
    },
    {
      "name": "OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT",
      "value": "http://127.0.0.1:4315"
    },
    {
      "name": "OTEL_TRACES_SAMPLER",
      "value": "xray"
    },
    {
      "name": "OTEL_EXPORTER_OTLP_TRACES_ENDPOINT",
      "value": "http://127.0.0.1:4315"
    },
    {
      "name": "OTEL_PROPAGATORS",
      "value": "tracecontext,baggage,b3,xray"
    }
  ],
  "mountPoints": [
    {
      "sourceVolume": "opentelemetry-auto-instrumentation",
      "containerPath": "/otel-auto-instrumentation",
      "readOnly": false
    }
  ]
}
```

Python

Python 애플리케이션에서 Application Signals를 활성화하기 전에 다음 사항에 유의하세요.

- 일부 컨테이너화된 애플리케이션에서는 PYTHONPATH 환경 변수가 누락되면 애플리케이션이 시작되지 않을 수 있습니다. 이 문제를 해결하려면 PYTHONPATH 환경 변수를 애플리케이션의 작업 디렉터리 위치로 설정해야 합니다. 이는 OpenTelemetry 자동 계측과 관련하여 알려진 문제 때문입니다. 이 문제에 대한 자세한 내용은 Python autoinstrumentation setting of PYTHONPATH is not compliant를 참조하세요.

- Django 애플리케이션의 경우 추가 필수 구성이 있으며, OpenTelemetry Python 설명서에서 설명합니다.
  - `--noreload` 플래그를 사용하여 자동 재로드를 방지합니다.
  - `DJANGO_SETTINGS_MODULE` 환경 변수를 Django 애플리케이션 `settings.py` 파일의 위치로 설정합니다. 이렇게 하면 OpenTelemetry가 올바르게 액세스하여 Django 설정에 통합할 수 있습니다.

CloudWatch 에이전트를 사용하여 Amazon ECS에서 Python 애플리케이션 계측

1. 먼저 바인드 탑재를 지정합니다. 다음 단계에서 이 볼륨은 컨테이너 간에 파일을 공유하는 데 사용됩니다. 이 바인드 탑재는 이 절차의 뒷부분에서 사용합니다.

```
"volumes": [
  {
    "name": "opentelemetry-auto-instrumentation-python"
  }
]
```

2. CloudWatch 에이전트 사이드카 정의를 추가합니다. 이렇게 하려면 애플리케이션의 작업 정의에 `ecs-cwagent`라는 새 컨테이너를 추가합니다. *`$REGION`*을 실제 리전 이름으로 바꿉니다. Amazon Elastic Container Registry의 최신 CloudWatch 컨테이너 이미지 경로로 바꿉니다. 자세한 내용은 Amazon ECR의 cloudwatch-agent를 참조하세요.

```
{
  "name": "ecs-cwagent",
  "image": "$IMAGE",
  "essential": true,
  "secrets": [
    {
      "name": "CW_CONFIG_CONTENT",
      "valueFrom": "ecs-cwagent"
    }
  ],
  "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
      "awslogs-create-group": "true",
      "awslogs-group": "/ecs/ecs-cwagent",
      "awslogs-region": "$REGION",
      "awslogs-stream-prefix": "ecs"
```

```
    }
  }
}
```

3. 애플리케이션의 작업 정의에 새 컨테이너 `init`를 추가합니다. *`$IMAGE`*를 AWS Distro for OpenTelemetry Amazon ECR 이미지 리포지토리의 최신 이미지로 바꿉니다.

```
{
    "name": "init",
    "image": "$IMAGE",
    "essential": false,
    "command": [
        "cp",
        "-a",
        "/autoinstrumentation/.",
        "/otel-auto-instrumentation-python"
    ],
    "mountPoints": [
        {
            "sourceVolume": "opentelemetry-auto-instrumentation-python",
            "containerPath": "/otel-auto-instrumentation-python",
            "readOnly": false
        }
    ]
}
```

4. 애플리케이션 컨테이너에 다음 환경 변수를 추가합니다. 자세한 내용을 알아보려면 다음 섹션을 참조하세요.

환경 변수	Application Signals를 활성화하도록 설정
`OTEL_RESOURCE_ATTRIBUTES`	`$SVC_NAME` 을 애플리케이션 이름으로 바꿉니다. Application Signals 대시보드에 애플리케이션 이름으로 표시됩니다. `$HOST_ENV` 를 애플리케이션이 실행되는 호스트 환경으로 바꿉니다. 이는 Application Signals 대시보드에서 애플리케이션의 호스팅 위치 환경으로 표시됩니다.

환경 변수	Application Signals를 활성화하도록 설정
OTEL_AWS_APP_SIGNALS_ENABLED	Application Signals SpanMetrics Processor를 활성화하려면 true로 설정합니다.
OTEL_METRICS_EXPORTER	다른 지표 내보내기를 비활성화하려면 none으로 설정합니다.
OTEL_EXPORTER_OTLP_PROTOCOL	http/protobuf 로 설정하여 HTTP를 사용하여 지표 및 추적을 CloudWatch에 전송합니다.
OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT	CloudWatch 사이드카로 지표를 전송하려면 http://127.0.0.1:4316/v1/metrics 로 설정합니다.
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	CloudWatch 사이드카로 트레이스를 전송하려면 http://127.0.0.1:4316/v1/traces 로 설정합니다.
OTEL_TRACES_SAMPLER	X-Ray를 트레이스 샘플러로 정의합니다.
OTEL_PROPAGATORS	전파자 중 하나로 X-Ray를 추가합니다.
OTEL_PYTHON_DISTRO	aws_distro 로 설정하여 ADOT Python 계측을 사용합니다.
OTEL_PYTHON_CONFIGURATOR	aws_configuration 으로 설정하여 ADOT Python 구성을 사용합니다.
PYTHONPATH	컨테이너 내 애플리케이션 작업 디렉터리의 위치로 $APP_PATH 를 바꿉니다. 이는 Python 인터프리터가 애플리케이션 모듈을 찾는 데 필요합니다.
DJANGO_SETTINGS_MODULE	Django 애플리케이션에만 필요합니다. Django 애플리케이션 settings.py 파일의 위치로 설정합니다. $PATH_TO_SETTINGS 를 바꿉니다.

5. 이 절차의 1단계에서 정의한 볼륨을 opentelemetry-auto-instrumentation-python을 마운트합니다.

   Python 애플리케이션의 경우 다음을 사용합니다.

```
{
    "name": "app",
    ...
    "environment": [
        {
            "name": "PYTHONPATH",
            "value": "/otel-auto-instrumentation-python/opentelemetry/
instrumentation/auto_instrumentation:$APP_PATH:/otel-auto-instrumentation-
python"
        },
        {
            "name": "OTEL_EXPORTER_OTLP_PROTOCOL",
            "value": "http/protobuf"
        },
        {
            "name": "OTEL_TRACES_SAMPLER",
            "value": "xray"
        },
        {
            "name": "OTEL_TRACES_SAMPLER_ARG",
            "value": "endpoint=http://localhost:2000"
        },
        {
            "name": "OTEL_LOGS_EXPORTER",
            "value": "none"
        },
        {
            "name": "OTEL_PYTHON_DISTRO",
            "value": "aws_distro"
        },
        {
            "name": "OTEL_PYTHON_CONFIGURATOR",
            "value": "aws_configurator"
        },
        {
            "name": "OTEL_EXPORTER_OTLP_TRACES_ENDPOINT",
            "value": "http://localhost:4316/v1/traces"
        },
```

```
        {
            "name": "OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT",
            "value": "http://localhost:4316/v1/metrics"
        },
        {
            "name": "OTEL_METRICS_EXPORTER",
            "value": "none"
        },
        {
            "name": "OTEL_AWS_APP_SIGNALS_ENABLED",
            "value": "true"
        },
        {
            "name": "OTEL_RESOURCE_ATTRIBUTES",
            "value": "aws.hostedin.environment=$HOST_ENV,service.name=$SVC_NAM"
        },
        {
            "name": "DJANGO_SETTINGS_MODULE",
            "value": "$PATH_TO_SETTINGS.settings"
        }
    ],
    "mountPoints": [
        {
            "sourceVolume": "opentelemetry-auto-instrumentation-python",
            "containerPath": "/otel-auto-instrumentation-python",
            "readOnly": false
        }
    ]
}
```

5단계: 애플리케이션 배포

작업 정의의 새 버전을 생성하여 애플리케이션 클러스터에 배포합니다. 새로 생성된 작업에는 세 개의 컨테이너가 표시되어야 합니다.

- `init`
- `ecs-cwagent`
- `app`

사용자 지정 설정을 사용하여 Amazon EC2와 기타 플랫폼에서 Application Signals를 활성화합니다.

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

Amazon EC2 및 Amazon EKS가 아닌 기타 아키텍처에서 실행되는 애플리케이션의 경우, CloudWatch 에이전트와 AWS Distro for OpenTelemetry를 직접 설치하고 구성합니다. 사용자 지정 Application Signals 설정으로 활성화된 이러한 아키텍처에서는 Application Signals가 서비스 이름이나 서비스가 실행되는 호스트 또는 클러스터를 자동으로 검색하지 않습니다. 사용자 지정 설정 중에 이러한 이름을 지정해야 하며, 지정하는 이름은 Application Signals 대시보드에 표시됩니다.

다음 단계는 Amazon EC2 인스턴스에서 테스트되었지만 AWS Distro for OpenTelemetry를 지원하는 다른 아키텍처에서도 작동할 것으로 예상됩니다.

요구 사항

- Application Signals에 대한 지원을 받으려면 CloudWatch 에이전트와 AWS Distro for OpenTelemetry 에이전트 모두 최신 버전을 사용해야 합니다.
- 인스턴스에 AWS CLI가 설치되어 있어야 합니다. AWS CLI 버전 2를 권장하지만 버전 1도 사용할 수 있습니다. AWS CLI 설치에 대한 자세한 내용은 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.

⚠ Important

Application Signals에 대해 활성화하려는 애플리케이션과 함께 이미 OpenTelemetry를 사용하고 있다면 Application Signals를 활성화하기 전에 OpenTelemetry 호환성 고려 사항 섹션을 참조하세요.

1단계: 계정에서 Application Signals 활성화

이 계정에서 아직 Application Signals를 활성화하지 않은 경우 서비스를 검색하는 데 필요한 권한을 Application Signals에 부여해야 합니다. 그렇게 하려면 다음을 수행합니다. 계정에 대해 한 번만 수행하면 됩니다.

애플리케이션에 대해 Application Signals 활성화

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 서비스를 선택합니다.
3. 서비스 검색 시작을 선택합니다.
4. 확인란을 선택하고 서비스 검색 시작을 선택합니다.

   계정에서 처음으로 이 단계를 완료하면 AWSServiceRoleForCloudWatchApplicationSignals 서비스 역할이 생성됩니다. 이 역할은 Application Signals에 다음 권한을 부여합니다.

   - `xray:GetServiceGraph`
   - `logs:StartQuery`
   - `logs:GetQueryResults`
   - `cloudwatch:GetMetricData`
   - `cloudwatch:ListMetrics`
   - `tag:GetResources`

   이에 대한 자세한 내용은 CloudWatch Application Signals에 대한 서비스 연결 역할 권한 섹션을 참조하세요.

2단계: CloudWatch 에이전트 다운로드 및 시작

Amazon EC2 인스턴스에서 Application Signals 활성화의 일환으로 CloudWatch 에이전트 설치

1. 최신 버전의 CloudWatch 에이전트를 인스턴스에 다운로드합니다. 인스턴스에 CloudWatch 에이전트가 이미 설치되어 있는 경우 업데이트해야 할 수 있습니다. 2023년 11월 30일 이후에 릴리스된 에이전트 버전만 CloudWatch Application Signals를 지원합니다.

   CloudWatch 에이전트 다운로드에 대한 자세한 내용은 CloudWatch 에이전트 패키지 다운로드 섹션을 참조하세요.
2. CloudWatch 에이전트를 시작하기 전에 Application Signals를 활성화하도록 구성합니다. 다음 예제는 EC2 호스트의 지표와 트레이스 모두에 대해 Application Signals를 활성화하는 CloudWatch 에이전트 구성입니다.

   다음 명령을 입력하여 이 파일을 생성할 수 있습니다.

```
vim amazon-cloudwatch-agent.json
```

이 파일의 내용으로 다음을 추가합니다.

```
{
  "traces": {
    "traces_collected": {
      "app_signals": {}
    }
  },
  "logs": {
    "metrics_collected": {
      "app_signals": {}
    }
  }
}
```

3. Amazon EC2 인스턴스의 IAM 역할에 CloudWatchAgentServerPolicy 및 AWSXrayWriteOnlyAccess IAM 정책을 연결합니다.

   a. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.

   b. 역할을 선택하고 Amazon EC2 인스턴스에서 사용하는 역할을 찾습니다. 그런 다음, 해당 역할의 이름을 선택합니다.

   c. 권한 탭에서 권한 추가, 정책 연결을 선택합니다.

   d. CloudWatchAgentServerPolicy를 찾습니다. 필요한 경우 검색 상자를 사용합니다. 그런 다음 해당 정책의 확인란을 선택하고 권한 추가를 선택합니다.

   e. AWSXrayWriteOnlyAccess를 찾습니다. 필요한 경우 검색 상자를 사용합니다. 그런 다음 해당 정책의 확인란을 선택하고 권한 추가를 선택합니다.

4. 다음 명령을 입력하여 CloudWatch 에이전트를 시작합니다. *agent-config-file-path*를 `./amazon-cloudwatch-agent.json` 등의 CloudWatch 에이전트 구성 파일의 경로로 바꿉니다. 다음과 같이 `file:` 접두사를 포함해야 합니다.

```
export CONFIG_FILE_PATH=./amazon-cloudwatch-agent.json
```

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
-a fetch-config \
```

```
-m ec2 -s -c file:$CONFIG_FILE_PATH
```

3단계: 애플리케이션 계측 후 시작

다음 단계는 CloudWatch Application Signals에 대해 애플리케이션을 계측하는 것입니다.

Java

Amazon EC2 인스턴스에서 Application Signals 활성화의 일환으로 Java 애플리케이션 계측

1. 최신 버전의 AWS Distro for OpenTelemetry Java 자동 계측 에이전트를 다운로드합니다. 이 링크를 사용하여 최신 버전을 다운로드할 수 있습니다. aws-otel-java-instrumentation 릴리스에서 모든 릴리스 버전에 대한 정보를 볼 수 있습니다.
2. Application Signals의 이점을 최적화하려면 애플리케이션을 시작하기 전에 환경 변수를 사용하여 추가 정보를 제공합니다. 이 정보는 Application Signals 대시보드에 표시됩니다.

   a. `OTEL_RESOURCE_ATTRIBUTES` 변수에 대해 다음 정보를 키-값 페어로 지정합니다.

      - `aws.hostedin.environment`는 애플리케이션이 실행되는 환경을 설정합니다. 이는 Application Signals 대시보드에서 애플리케이션의 호스팅 위치 환경으로 표시됩니다. 이 속성 키는 Application Signals에서만 사용되며 X-Ray 트레이스 주석 및 CloudWatch 지표 측정기준으로 변환됩니다. 이 키에 대한 값을 제공하지 않으면 기본값인 `Generic`가 사용됩니다.
      - `service.name`은 서비스 이름을 설정합니다. Application Signals 대시보드에 애플리케이션의 서비스 이름으로 표시됩니다. 이 키에 대한 값을 제공하지 않으면 기본값인 `unknown_service`가 사용됩니다.

   b. `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` 변수에 대해 트레이스를 내보낼 기본 엔드포인트 URL을 지정합니다. CloudWatch 에이전트는 4315를 OLTP 포트로 노출합니다. Amazon EC2에서는 애플리케이션이 로컬 CloudWatch 에이전트와 통신하기 때문에 이 값을 `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4315`로 설정해야 합니다.

   c. `OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT` 변수에 대해 지표를 내보낼 기본 엔드포인트 URL을 지정합니다. CloudWatch 에이전트는 4315를 OLTP 포트로 노출합니다. Amazon EC2에서는 애플리케이션이 로컬 CloudWatch 에이전트와 통신하기 때문에 이 값을 `OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT=http://localhost:4315`로 설정해야 합니다.

d. JAVA_TOOL_OPTIONS 변수에 대해 AWS Distro for OpenTelemetry Java 자동 계측 에이전트가 저장되는 경로를 지정합니다.

```
export JAVA_TOOL_OPTIONS=' -javaagent:$ADOT_AGENT_PATH'
```

예:

```
export ADOT_AGENT_PATH=./aws-opentelemetry-agent.jar
```

e. OTEL_METRICS_EXPORTER 변수에 대해 값을 none으로 설정하는 것이 좋습니다. 이렇게 하면 다른 지표 내보내기가 비활성화되어 Application Signals 내보내기만 사용됩니다.

f. OTEL_AWS_APP_SIGNALS_ENABLED 변수의 경우 OTEL_AWS_APP_SIGNALS_ENABLED를 true로 설정하여 SpanMetricProcessor(SMP)를 활성화합니다. 이렇게 하면 트레이스에서 Application Signals 지표가 생성됩니다.

3. 이전 단계에서 설명한 환경 변수를 사용하여 애플리케이션을 시작합니다. 다음은 시작 스크립트의 예입니다.

```
JAVA_TOOL_OPTIONS=' -javaagent:$ADOT_AGENT_PATH' \
OTEL_METRICS_EXPORTER=none \
OTEL_AWS_APP_SIGNALS_ENABLED=true \
OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT=http://localhost:4315 \
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4315 \
OTEL_RESOURCE_ATTRIBUTES=aws.hostedin.environment=$YOUR_HOST_ENV,service.name=
$YOUR_SVC_NAME \
java -jar $MY_JAVA_APP.jar
```

Python

Amazon EC2 인스턴스에서 Application Signals 활성화의 일환으로 Python 애플리케이션 계측

1. 최신 버전의 AWS Distro for OpenTelemetry Python 자동 계측 에이전트를 다운로드합니다. 다음 명령을 실행하여 인증서를 설치합니다.

```
pip install aws-opentelemetry-distro
```

AWS Distro for OpenTelemetry Python 계측에서 모든 릴리스된 버전의 정보를 볼 수 있습니다.

2. Application Signals의 이점을 최적화하려면 애플리케이션을 시작하기 전에 환경 변수를 사용하여 추가 정보를 제공합니다. 이 정보는 Application Signals 대시보드에 표시됩니다.

   a. OTEL_RESOURCE_ATTRIBUTES 변수에 대해 다음 정보를 키-값 페어로 지정합니다.

      - aws.hostedin.environment는 애플리케이션이 실행되는 환경을 설정합니다. 이는 Application Signals 대시보드에서 애플리케이션의 호스팅 위치 환경으로 표시됩니다. 이 속성 키는 Application Signals에서만 사용되며 X-Ray 트레이스 주석 및 CloudWatch 지표 측정기준으로 변환됩니다. 이 키에 대한 값을 제공하지 않으면 기본값인 Generic가 사용됩니다.
      - service.name은 서비스 이름을 설정합니다. Application Signals 대시보드에 애플리케이션의 서비스 이름으로 표시됩니다. 이 키에 대한 값을 제공하지 않으면 기본값인 unknown_service가 사용됩니다.

   b. OTEL_EXPORTER_OTLP_PROTOCOL 변수의 경우 http/protobuf를 지정하여 HTTP를 통해 다음 단계에 나열된 CloudWatch 에이전트 엔드포인트로 원격 분석 데이터를 내보냅니다.

   c. OTEL_EXPORTER_OTLP_TRACES_ENDPOINT 변수에 대해 트레이스를 내보낼 기본 엔드포인트 URL을 지정합니다. CloudWatch 에이전트는 4316을 HTTP를 통한 OLTP 포트로 노출합니다. Amazon EC2에서는 애플리케이션이 로컬 CloudWatch 에이전트와 통신하기 때문에 이 값을 OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4316/v1/traces로 설정해야 합니다.

   d. OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT 변수에 대해 지표를 내보낼 기본 엔드포인트 URL을 지정합니다. CloudWatch 에이전트는 4316을 HTTP를 통한 OLTP 포트로 노출합니다. Amazon EC2에서는 애플리케이션이 로컬 CloudWatch 에이전트와 통신하기 때문에 이 값을 OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT=http://localhost:4316/v1/metrics로 설정해야 합니다.

   e. OTEL_METRICS_EXPORTER 변수에 대해 값을 none으로 설정하는 것이 좋습니다. 이렇게 하면 다른 지표 내보내기가 비활성화되어 Application Signals 내보내기만 사용됩니다.

   f. OTEL_AWS_APP_SIGNALS_ENABLED 변수의 경우 OTEL_AWS_APP_SIGNALS_ENABLED를 true로 설정하여 SpanMetricProcessor를 활성화합니다. 이렇게 하면 트레이스에서 Application Signals 지표가 생성됩니다.

3. 이전 단계에서 설명한 환경 변수를 사용하여 애플리케이션을 시작합니다. 다음은 시작 스크립트의 예입니다.

   - $HOST_ENV를 애플리케이션이 실행되는 호스트 환경으로 바꿉니다. 이는 Application Signals 대시보드에서 애플리케이션의 호스팅 위치 환경으로 표시됩니다.

- $SVC_NAME을 애플리케이션 이름으로 바꿉니다. Application Signals 대시보드에 애플리케이션 이름으로 표시됩니다.
- $PYTHON_APP을 애플리케이션 위치와 이름으로 바꿉니다.

```
OTEL_METRICS_EXPORTER=none \
OTEL_LOGS_EXPORTER=none \
OTEL_AWS_APP_SIGNALS_ENABLED=true \
OTEL_PYTHON_DISTRO=aws_distro \
OTEL_PYTHON_CONFIGURATOR=aws_configurator \
OTEL_EXPORTER_OTLP_PROTOCOL=http/protobuf \
OTEL_TRACES_SAMPLER=xray \
OTEL_TRACES_SAMPLER_ARG="endpoint=http://localhost:2000" \
OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT=http://localhost:4316/v1/metrics \
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4316/v1/traces \
OTEL_RESOURCE_ATTRIBUTES=aws.hostedin.environment=$HOST_ENV,service.name=
$SVC_NAME \
opentelemetry-instrument python $PYTHON_APP.py
```

Python 애플리케이션에서 Application Signals를 활성화하기 전에 다음 사항에 유의하세요.

- 일부 컨테이너화된 애플리케이션에서는 PYTHONPATH 환경 변수가 누락되면 애플리케이션이 시작되지 않을 수 있습니다. 이 문제를 해결하려면 PYTHONPATH 환경 변수를 애플리케이션의 작업 디렉터리 위치로 설정해야 합니다. 이는 OpenTelemetry 자동 계측과 관련하여 알려진 문제 때문입니다. 이 문제에 대한 자세한 내용은 Python autoinstrumentation setting of PYTHONPATH is not compliant를 참조하세요.
- Django 애플리케이션의 경우 추가 필수 구성이 있으며, OpenTelemetry Python 설명서에서 설명합니다.
  - `--noreload` 플래그를 사용하여 자동 재로드를 방지합니다.
  - `DJANGO_SETTINGS_MODULE` 환경 변수를 Django 애플리케이션 `settings.py` 파일의 위치로 설정합니다. 이렇게 하면 OpenTelemetry가 올바르게 액세스하여 Django 설정에 통합할 수 있습니다.

Application Signals 설치 문제 해결

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

이 섹션에서는 CloudWatch Application Signals 관련 문제 해결 팁을 제공합니다.

주제

- Application Signals가 활성화된 후 애플리케이션이 시작되지 않음
- Application Signals가 활성화된 후 Python 애플리케이션이 시작되지 않음
- CloudWatch 및 X-Ray에서 텔레메트리 데이터 누락
- 종속성 지표에 알 수 없음 값 있음
- Amazon CloudWatch Observability EKS 추가 기능 관리 시 ConfigurationConflict 처리

Application Signals가 활성화된 후 애플리케이션이 시작되지 않음

클러스터에서 Application Signals를 활성화한 후 Amazon EKS 클러스터의 애플리케이션이 시작되지 않는 경우 다음을 확인합니다.

- 애플리케이션이 다른 모니터링 솔루션에서 계측되었는지 확인합니다. Application Signals는 다른 계측 솔루션과의 공존을 지원하지 않습니다.
- 애플리케이션이 Application Signals 사용을 위한 호환성 요구 사항을 충족하는지 확인합니다. 자세한 내용은 Application Signals 지원 시스템 섹션을 참조하세요.
- 애플리케이션이 AWS Distro for OpenTelemetery Java 또는 Python 에이전트 및 CloudWatch 에이전트 이미지와 같은 Application Signals 아티팩트를 가져오지 못한 경우 네트워크 문제일 수 있습니다.

문제를 완화하려면 애플리케이션 배포 매니페스트에서 주석 `instrumentation.opentelemetry.io/inject-java: "true"` 또는 `instrumentation.opentelemetry.io/inject-python: "true"`를 제거하고 애플리케이션을 다시 배포합니다. 그런 다음, 애플리케이션이 작동하는지 확인합니다.

Application Signals가 활성화된 후 Python 애플리케이션이 시작되지 않음

PYTHONPATH 환경 변수 누락으로 인해 애플리케이션이 시작되지 않는 경우가 있다는 것은 OpenTelemetry 자동 계측과 관련하여 알려진 문제입니다. 이 문제를 해결하려면 PYTHONPATH 환경 변수를 애플리케이션의 작업 디렉터리 위치로 설정해야 합니다. 이 문제에 대한 자세한 내용은 Python autoinstrumentation setting of PYTHONPATH is not compliant with Python's module resolution behavior, breaking Django applications를 참조하세요.

Django 애플리케이션의 경우 추가 필수 구성이 있으며, OpenTelemetry Python 설명서에서 설명합니다.

- `--noreload` 플래그를 사용하여 자동 재로드를 방지합니다.
- `DJANGO_SETTINGS_MODULE` 환경 변수를 Django 애플리케이션 `settings.py` 파일의 위치로 설정합니다. 이렇게 하면 OpenTelemetry가 올바르게 액세스하여 Django 설정에 통합할 수 있습니다.

CloudWatch 및 X-Ray에서 텔레메트리 데이터 누락

Application Signals 대시보드에서 지표나 트레이스가 누락된 경우 다음과 같은 원인이 있을 수 있습니다. 마지막 업데이트 이후 Application Signals가 데이터를 수집하고 표시할 때까지 15분을 기다린 경우에만 이러한 원인을 조사합니다.

- 사용 중인 라이브러리와 프레임워크가 ADOT Java 에이전트에서 지원되는지 확인합니다. 자세한 내용은 라이브러리/프레임워크를 참조하세요.
- CloudWatch 에이전트가 실행 중인지 확인합니다. 먼저 CloudWatch 에이전트 포드의 상태를 확인하고 모든 포드가 `Running` 상태에 있는지 확인합니다.

  ```
  kubectl -n amazon-cloudwatch get pods.
  ```

  다음을 CloudWatch 에이전트 구성 파일에 추가하고 에이전트를 재시작합니다.

  ```
  "agent": {
  >>>>>>> streams
    "region": "${REGION}",
    "debug": true
  },
  ```

  그런 다음, CloudWatch 에이전트 포드에 오류가 있는지 확인합니다.

- CloudWatch 에이전트의 구성 문제를 확인합니다. 다음 내용이 여전히 CloudWatch 에이전트 구성 파일에 있고 에이전트가 추가된 후 에이전트가 다시 시작되었는지 확인합니다.

  ```
  "agent": {
    "region": "${REGION}",
    "debug": true
  },
  ```

  그런 다음, OpenTelemetry 디버깅 로그에서 `ERROR io.opentelemetry.exporter.internal.grpc.OkHttpGrpcExporter - Failed to export ...`와 같은 오류 메시지가 있는지 확인합니다. 이러한 메시지는 문제를 나타낼 수 있습니다.

  그래도 문제가 해결되지 않으면 `kubectl describe pod` 명령으로 포드를 설명하여 `OTEL_`로 시작하는 이름으로 환경 변수를 덤프하고 확인합니다.
- OpenTelemetry Python 디버그 로깅을 활성화하려면 환경 변수 `OTEL_PYTHON_LOG_LEVEL`을 `debug`로 설정하고 애플리케이션을 재배포합니다.
- CloudWatch 에이전트에서 데이터를 내보낼 수 있는 권한이 잘못되었거나 충분하지 않은지 확인합니다. CloudWatch 에이전트 로그에 `Access Denied` 메시지가 표시되는 경우 이것이 문제일 수 있습니다. CloudWatch 에이전트를 설치할 때 적용한 권한이 나중에 변경되거나 취소되었을 수 있습니다.
- 텔레메트리 데이터를 생성할 때 AWS Distro for OpenTelemetry(ADOT) 문제가 있는지 확인합니다.

  계측 주석 `instrumentation.opentelemetry.io/inject-java` 및 `sidecar.opentelemetry.io/inject-java`가 애플리케이션 배포에 적용되고 값이 `true`인지 확인합니다. 이 옵션이 없으면 ADOT 추가 기능이 올바르게 설치되더라도 애플리케이션 포드가 계측되지 않습니다.

  다음으로, `Init` 컨테이너가 애플리케이션에 적용되었고 `Ready` 상태가 `True`인지 확인합니다. `init` 컨테이너가 준비되지 않은 경우 상태를 참조하여 이유를 확인합니다.

  문제가 지속되면 다음을 수행하여 OpenTelemetry Java SDK에서 디버그 로깅을 활성화합니다. 그런 다음, `ERROR io.telemetry`로 시작하는 메시지를 찾습니다.

  디버그 로깅을 활성화하려면 환경 변수를 `OTEL_JAVAAGENT_DEBUG` true로 설정하고 애플리케이션을 재배포합니다.
- 지표/범위 내보내기가 데이터를 삭제하고 있을 수 있습니다. 애플리케이션 로그에서 `Failed to export...`를 포함하는 메시지가 있는지 확인합니다.

- CloudWatch 에이전트가 지표 또는 범위를 Application Signals로 전송할 때 제한을 받을 수 있습니다. CloudWatch 에이전트 로그에서 제한을 나타내는 메시지가 있는지 확인합니다.

종속성 지표에 알 수 없음 값 있음

Application Signals 대시보드에서 종속성 이름 또는 작업에 대해 UnknownOperation, UnknownRemoteService 또는 UnknownRemoteOperation이 표시되는 경우 알 수 없는 원격 서비스 및 알 수 없는 원격 작업에 대한 데이터 포인트의 발생이 해당 배포와 일치하는지 확인합니다. 이는 Application Signals의 알려진 문제이며 향후 릴리스에서 수정될 예정입니다.

Amazon CloudWatch Observability EKS 추가 기능 관리 시 ConfigurationConflict 처리

Amazon CloudWatch Observability EKS 추가 기능을 설치하거나 업데이트할 때 `Conflicts found when trying to apply. Will not continue due to resolve conflicts mode`로 시작하는 설명과 함께 `ConfigurationConflict` 유형의 `Health Issue`로 인해 발생한 오류가 발견된 경우, CloudWatch 에이전트와 ServiceAccount, ClusterRole, ClusterRoleBinding 등의 연결된 구성 요소가 클러스터에 이미 설치되어 있기 때문일 수 있습니다. 추가 기능이 CloudWatch 에이전트 및 연결된 구성 요소를 설치하려고 할 때 콘텐츠의 변경 사항이 탐지되면 기본적으로 클러스터의 리소스 상태를 덮어쓰지 않도록 설치 또는 업데이트가 실패합니다.

Amazon CloudWatch Observability EKS 추가 기능에 온보딩하려고 하는데 이 오류가 표시되는 경우 이전에 클러스터에 설치한 기존 CloudWatch 에이전트 설정을 삭제한 다음, EKS 추가 기능을 설치하는 것이 좋습니다. 사용자 지정 에이전트 구성과 같이 원래 CloudWatch 에이전트 설정에 대한 모든 사용자 지정을 백업하고 다음에 설치하거나 업데이트할 때 Amazon CloudWatch Observability EKS 추가 기능에 제공합니다. 이전에 Container Insights 온보딩을 위해 CloudWatch 에이전트를 설치한 경우 자세한 내용은 Container Insights의 CloudWatch 에이전트 및 Fluent Bit 삭제 섹션을 참조하세요.

또는 추가 기능은 `OVERWRITE`를 지정하는 기능이 있는 충돌 해결 구성 옵션을 지원합니다. 이 옵션을 사용하면 클러스터의 충돌을 덮어써서 추가 기능 설치 또는 업데이트를 진행할 수 있습니다. Amazon EKS 콘솔을 사용하는 경우 추가 기능을 생성하거나 업데이트할 때 선택적 구성 설정을 선택하면 충돌 해결 방법을 찾을 수 있습니다. AWS CLI를 사용하는 경우 명령에 `--resolve-conflicts OVERWRITE`를 제공하여 추가 기능을 생성하거나 업데이트할 수 있습니다.

Application Signals 구성

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

이 섹션에는 CloudWatch Application Signals 구성에 대한 정보가 들어 있습니다.

트레이스 샘플링 속도

기본적으로 Application Signals를 활성화하면 `reservoir=1/s` 및 `fixed_rate=5%`의 기본 샘플링 속도 설정을 사용하여 X-Ray 중앙 집중식 샘플링이 활성화됩니다. AWS Distro for OpenTelemetry (ADOT) SDK 에이전트의 환경 변수는 다음과 같이 설정됩니다.

환경 변수	값	참고
`OTEL_TRACES_SAMPLER`	`xray`	
`OTEL_TRACES_SAMPLER_ARG`	`endpoint=http://cloudwatch-agent.amazon-cloudwatch:2000`	CloudWatch 에이전트의 엔드포인트

샘플링 구성 변경에 대한 자세한 내용은 다음을 참조하세요.

- X-Ray 샘플링을 변경하려면 샘플링 규칙 사용자 정의를 참조하세요.
- ADOT 샘플링을 변경하려면 X-Ray 원격 샘플링을 위한 OpenTelemetry 컬렉터 구성을 참조하세요.

X-Ray 중앙 집중식 샘플링을 비활성화하고 로컬 샘플링을 대신 사용하려면 ADOT SDK Java 에이전트에 대해 다음과 같이 값을 설정합니다. 다음 예제에서는 샘플링 속도를 5%로 설정합니다.

환경 변수	값
`OTEL_TRACES_SAMPLER`	`parentbased_traceidratio`
`OTEL_TRACES_SAMPLER_ARG`	`0.05`

고급 샘플링 설정에 대한 자세한 내용은 OTEL_TRACES_SAMPLER를 참조하세요.

카디널리티가 높은 작업 관리

Application Signals에는 작업의 카디널리티를 관리하고 지표 내보내기를 관리하여 비용을 최적화하는 데 사용할 수 있는 CloudWatch 에이전트의 설정이 포함되어 있습니다. 기본적으로 지표 제한 기능은

시간 경과에 따른 서비스의 개별 작업 수가 기본 임계값인 500을 초과할 때 활성 상태가 됩니다. 구성 설정을 조정하여 동작을 미세 조정할 수 있습니다.

지표 제한이 활성화되었는지 확인

다음 방법을 사용하여 기본 지표 제한이 발생하는지 확인할 수 있습니다. 제한이 있는 경우 다음 섹션의 단계에 따라 카디널리티 제어 최적화를 고려해야 합니다.

- CloudWatch 콘솔에서 Application Signals, Services를 선택합니다. 이름이 AllOtherOperations인 Operation 또는 AllOtherRemoteOperations인 RemoteOperation이 표시되는 경우 지표 제한이 발생하고 있습니다.
- Application Signals에서 수집한 지표의 `Operation` 차원에 `AllOtherOperations` 값이 있는 경우 지표 제한이 발생하는 것입니다.
- Application Signals에서 수집한 지표의 `RemoteOperation` 차원에 `AllOtherRemoteOperations` 값이 있는 경우 지표 제한이 발생하는 것입니다.

카디널리티 제어 최적화

카디널리티 제어를 최적화하기 위해 다음을 수행할 수 있습니다.

- 사용자 지정 규칙을 만들어 작업을 집계합니다.
- 지표 제한 정책을 구성합니다.

사용자 지정 규칙을 만들어 작업을 집계합니다.

카디널리티가 높은 작업은 컨텍스트에서 추출된 부적절한 고유 값으로 인해 발생할 수 있습니다. 예를 들어 경로에 사용자 ID 또는 세션 ID가 포함된 HTTP/S 요청을 보내면 수백 개의 서로 다른 작업이 발생할 수 있습니다. 이러한 문제를 해결하려면 이러한 작업을 재작성하는 사용자 지정 규칙을 사용하여 CloudWatch 에이전트를 구성하는 것이 좋습니다.

`PUT /api/customer/owners/123`, `PUT /api/customer/owners/456` 등과 같은 개별 `RemoteOperation` 호출과 유사한 요청을 통한 다양한 지표 생성이 급증하는 경우 이러한 작업을 단일 `RemoteOperation`로 통합하는 것이 좋습니다. 한 가지 접근 방식은 특히 `PUT /api/customer/owners/{ownerId}`와 같이 `PUT /api/customer/owners/`로 시작하는 모든 `RemoteOperation` 호출을 통일된 형식으로 표준화하는 것입니다. 다음 예는 이를 보여 줍니다. 기타 사용자 지정 규칙에 대한 자세한 내용은 CloudWatch Application Signals 활성화 섹션을 참조하세요.

```
{
```

```
   "logs":{
      "metrics_collected":{
         "app_signals":{
            "rules":[
               {
                  "selectors":[
                     {
                        "dimension":"RemoteOperation",
                        "match":"PUT /api/customer/owners/*"
                     }
                  ],
                  "replacements":[
                     {
                        "target_dimension":"RemoteOperation",
                        "value":"PUT /api/customer/owners/{ownerId}"
                     }
                  ],
                  "action":"replace"
               }
            ]
         }
      }
   }
}
```

경우에 따라 카디널리티가 높은 지표가 `AllOtherRemoteOperations`에 집계되고, 어떤 특정 지표가 포함되는지 명확하지 않을 수 있습니다. CloudWatch 에이전트는 중단된 작업을 로깅할 수 있습니다. 중단된 작업을 식별하려면 다음 예시의 구성을 사용하여 문제가 다시 나타날 때까지 로깅을 활성화합니다. 그런 다음 CloudWatch 에이전트 로그(컨테이너 `stdout` 또는 EC2 로그 파일로 액세스 가능)를 검사하고 `drop metric data` 키워드를 검색합니다.

```
{
  "agent": {
    "config": {
      "agent": {
        "debug": true
      },
      "traces": {
        "traces_collected": {
          "app_signals": {
          }
        }
      },
```

```
      "logs": {
        "metrics_collected": {
          "app_signals": {
            "limiter": {
              "log_dropped_metrics": true
            }
          }
        }
      }
    }
  }
```

지표 제한 정책 생성

기본 지표 제한 구성으로 서비스의 카디널리티를 해결할 수 없는 경우 지표 제한기 구성을 사용자 지정할 수 있습니다. 이를 위해 CloudWatch 에이전트 구성 파일의 `logs/metrics_collected/app_signals` 섹션에 `limiter` 섹션을 추가합니다.

다음 예시에서는 지표 제한 임계값을 500개의 개별 지표에서 100개로 줄입니다.

```
{
  "logs": {
    "metrics_collected": {
      "app_signals": {
        "limiter": {
          "drop_threshold": 100
        }
      }
    }
  }
}
```

서비스 수준 목표(SLO)

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

Application Signals를 사용하여 중요한 비즈니스 운영을 위한 서비스에 대한 서비스 수준 목표를 생성할 수 있습니다. 이러한 서비스에 SLO를 생성하면 SLO 대시보드에서 SLO를 추적하여 가장 중요한 운영을 한눈에 파악할 수 있습니다.

운영자가 중요한 운영의 현재 상태를 확인하는 데 사용할 수 있는 빠른 뷰를 생성하는 것 외에도 SLO를 사용하여 서비스의 장기 성능을 추적하여 기대와 부합하는지 확인할 수 있습니다. 고객과 서비스 수준 계약을 체결한 경우 SLO는 이러한 조건을 충족할 수 있는 훌륭한 도구입니다.

SLO를 통해 서비스 상태를 평가하는 작업은 핵심 성능 지표인 서비스 수준 지표(SLI)를 기반으로 명확하고 측정 가능한 목표를 설정하는 것부터 시작됩니다. SLO는 설정된 임곗값 및 목표를 기준으로 SLI 성능을 추적하고 애플리케이션 성능이 임곗값에 얼마나 도달했는지 또는 얼마나 가까운지 보고합니다.

Application Signals는 주요 성능 지표에서 SLO를 설정하는 데 도움이 됩니다. Application Signals는 검색한 모든 서비스와 작업에 대한 `Latency` 및 `Availability` 지표를 자동으로 수집합니다. 이들 지표는 SLI로 사용하기에 이상적인 경우가 많습니다. SLO 생성 마법사를 사용하면 이러한 지표를 SLO에 사용할 수 있습니다. 그런 다음, Application Signals 대시보드를 사용하여 모든 SLO의 상태를 추적할 수 있습니다.

서비스가 호출하거나 사용하는 특정 운영에 대해 SLO를 설정할 수 있습니다. `Latency` 및 `Availability` 지표를 사용하는 것 외에도 CloudWatch 지표 또는 지표 표현식을 SLI로 사용할 수 있습니다.

SLO를 생성하는 것은 CloudWatch Application Signals를 최대한 활용하는 데 매우 중요합니다. SLO를 생성한 후에는 Application Signals 콘솔에서 SLO의 상태를 확인하여 이러한 중요한 서비스와 작업 중 어떤 것이 잘 수행되고 있고 어떤 것이 비정상인지 빠르게 확인할 수 있습니다. 추적할 SLO가 있으면 다음과 같은 주요 이점이 있습니다.

- 서비스 운영자는 SLI를 기준으로 측정된 중요 서비스의 현재 운영 상태를 더 쉽게 확인할 수 있습니다. 그러면 비정상 서비스와 작업을 신속하게 분류하고 식별할 수 있습니다.
- 측정 가능한 비즈니스 목표를 기준으로 서비스 성과를 장기간 추적할 수 있습니다.

SLO를 설정할 대상을 선택하면 중요한 것에 우선순위를 둘 수 있습니다. Application Signals 대시보드에는 우선순위를 지정한 항목에 대한 정보가 자동으로 표시됩니다.

SLO를 생성할 때 CloudWatch 경보를 동시에 생성하여 SLO를 모니터링하도록 선택할 수도 있습니다. 임곗값 위반과 경고 수준을 모니터링하는 경보를 설정할 수 있습니다. 이러한 경보는 SLO 지표가 설정한 임곗값을 위반하거나 경고 임곗값에 가까워지면 자동으로 알려줄 수 있습니다. 예를 들어, SLO가

경고 임곗값에 가까워지면 팀이 장기 성능 목표를 달성하기 위해 애플리케이션 변동을 늦춰야 할 수도 있음을 알 수 있습니다.

주제

SLO 개념

SLO에는 다음 구성 요소가 포함됩니다.

- 사용자가 지정하는 주요 성능 지표인 서비스 수준 지표(SLI). 애플리케이션에 대해 원하는 성능 수준을 나타냅니다. Application Signals는 검색한 모든 서비스와 작업에 대한 주요 지표인 `Latency`와 `Availability`를 자동으로 수집합니다. 이들은 SLO를 설정하는 데 이상적인 지표인 경우가 많습니다.

  SLI에 사용할 임곗값을 선택합니다. 예를 들어, 지연 시간은 200ms입니다.
- 각 시간 간격 동안 SLI가 임곗값을 충족할 것으로 예상되는 시간의 백분율인 목표 또는 달성 목표. 시간 간격은 짧게는 몇 시간 또는 길게는 1년일 수 있습니다.

  간격은 달력 간격 또는 연속 간격일 수 있습니다.
  - 달력 간격은 달력에 맞춰 조정됩니다(예: 매월 추적되는 SLO). CloudWatch는 한 달의 일수를 기준으로 상태, 예산 및 달성 수치를 자동으로 조정합니다. 달력 간격은 달력에 맞춰 측정되는 비즈니스 목표에 더 적합합니다.
  - 연속 간격은 순서대로 계산됩니다. 연속 간격은 애플리케이션의 최근 사용자 경험을 추적하는 데 더 적합합니다.
- 기간은 더 짧으며 여러 기간이 간격을 구성합니다. 애플리케이션의 성능은 간격 내의 각 기간 동안 SLI와 비교됩니다. 각 기간에 대해 애플리케이션이 필요한 성능을 달성했는지 여부가 결정됩니다.

예를 들어, 달력 간격이 1일이고 기간이 1분인 상태에서 목표를 99%로 설정하면 애플리케이션이 하루 중 1분 기 중 99% 동안 성공 임곗값을 충족하거나 달성해야 합니다. 충족하거나 달성하면 해당 날짜의

SLO가 충족된 것입니다. 다음 날은 새로운 평가 간격이며, 애플리케이션은 둘째 날의 SLO를 충족하려면 둘째 날의 1분 기간 중 99% 동안 성공 임곗값을 충족하거나 달성해야 합니다.

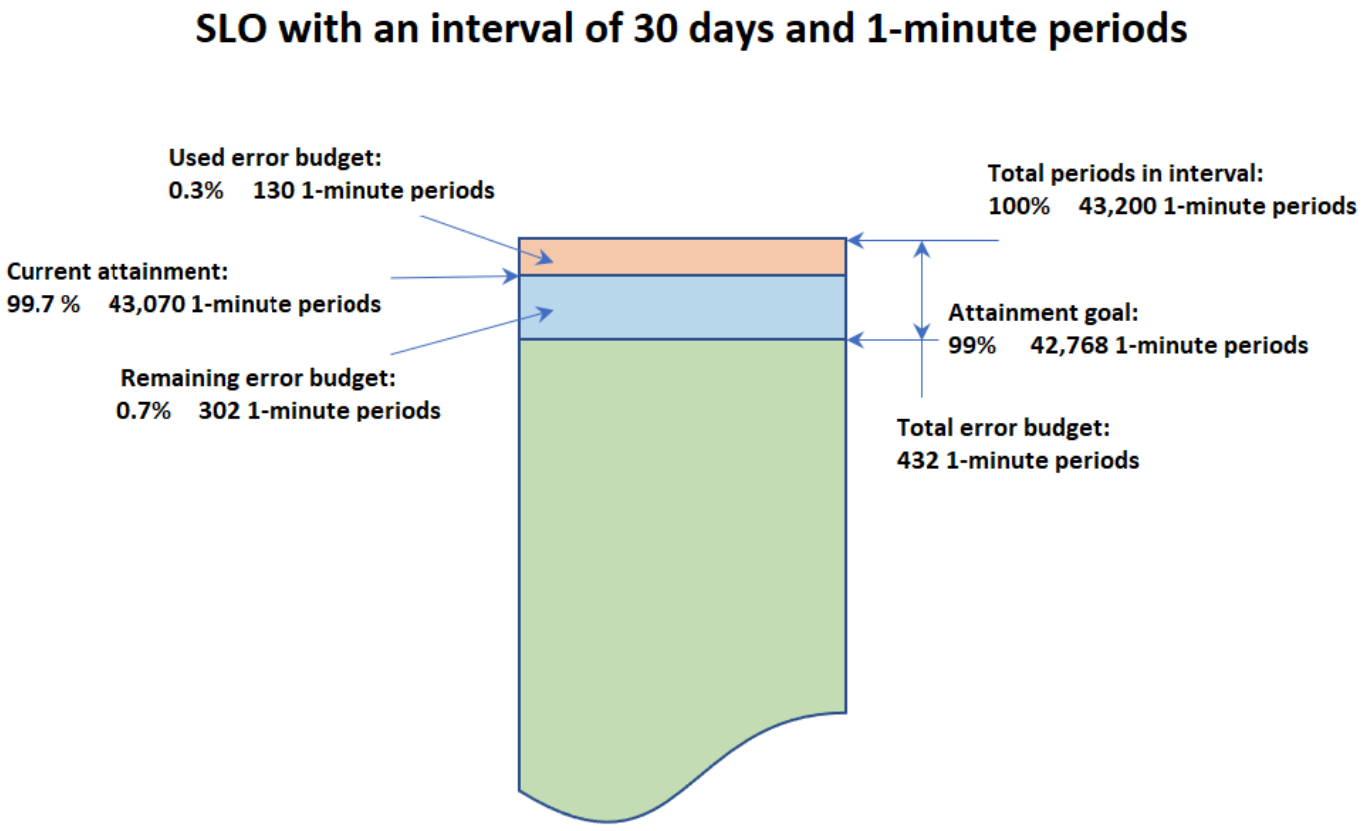
SLI는 Application Signals에서 수집한 새로운 표준 애플리케이션 지표 중 하나를 기준으로 할 수 있습니다. 또는 임의의 CloudWatch 지표 또는 지표 표현식일 수 있습니다. SLI에 사용할 수 있는 표준 애플리케이션 지표는 `Latency`와 `Availability`입니다. `Availability`는 성공적인 응답을 총 요청으로 나눈 값을 나타냅니다. (1 - 장애 발생률)*100으로 계산되며, 여기서 장애 응답은 `5xx` 오류입니다. 성공 응답은 `5XX` 오류가 없는 응답입니다. `4XX` 응답은 성공으로 처리됩니다.

> **Note**
>
> 현재는 기간 기준 계산만 지원됩니다. 향후 릴리스에서는 볼륨 또는 요청 기준 계산에 대한 지원이 계획되어 있습니다.

오류 할당 및 달성 계산

SLO에 대한 정보를 보면 SLO의 현재 상태와 오류 할당을 확인할 수 있습니다. 오류 할당은 임곗값을 위반할 수 있지만 여전히 SLO를 충족할 수 있는 간격 내의 시간입니다. 총 오류 할당은 전체 간격 동안 허용될 수 있는 총 위반 시간입니다. 남은 오류 할당은 현재 간격 동안 허용할 수 있는 남은 위반 시간입니다. 이는 전체 오류 할당에서 이미 발생한 위반 시간을 뺀 값입니다.

다음 그림은 30일 간격, 1분 기간 및 99% 달성 목표를 가진 목표에 대한 달성 및 오류 할당 개념을 보여줍니다. 30일에는 43,200번의 1분 기간이 포함됩니다. 43,200분의 99%는 42,768분이므로 SLO를 충족하려면 해당 월의 42,768분이 정상이어야 합니다. 현재 간격에서 지금까지 1분 기간 중 130번이 비정상이었습니다.



각 기간 내 성공 여부 판단

각 기간 내에서 SLI 데이터는 SLI에 사용된 통계를 기준으로 단일 데이터 포인트로 집계됩니다. 이 데이터 포인트는 기간의 전체 길이를 나타냅니다. 이 단일 데이터 포인트를 SLI 임곗값과 비교하여 기간이 정상인지 확인합니다. 대시보드에서 현재 시간 범위 중 비정상 기간을 확인하면 서비스 운영자에게 서비스를 분류해야 함을 알릴 수 있습니다.

기간이 비정상으로 확인되면 해당 기간 전체가 오류 할당에 대해 실패로 계산됩니다. 오류 할당을 추적하면 서비스가 장기간에 걸쳐 원하는 성능을 달성하고 있는지 알 수 있습니다.

SLO 생성

중요한 애플리케이션의 지연 시간 및 가용성 SLO를 모두 설정하는 것이 좋습니다. Application Signals에서 수집한 이러한 지표는 일반적인 비즈니스 목표에 부합합니다.

CloudWatch 지표 또는 단일 시계열을 생성하는 지표 수학 표현식에 SLO를 설정할 수도 있습니다.

계정에서 처음으로 SLO를 생성할 때 CloudWatch는 (아직 없는 경우) 계정에 AWSServiceRoleForCloudWatchApplicationSignals 서비스 연결 역할을 자동으로 생성합니다. 이 서비

스 연결 역할을 사용하여 CloudWatch Logs 데이터, X-Ray 추적 데이터, CloudWatch 지표 데이터, 태깅 데이터를 수집할 수 있습니다. CloudWatch 서비스 연결 역할에 대한 자세한 내용은 CloudWatch에 서비스 연결 역할 사용 섹션을 참조하세요.

SLO 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 서비스 수준 목표(SLO)를 선택합니다.
3. SLO 생성을 선택합니다.
4. SLO 이름을 입력합니다. 지연 시간 또는 가용성과 같은 적절한 키워드와 함께 서비스 또는 운영 이름을 포함하면 분류 중 SLO 상태가 무엇을 나타내는지 빠르게 식별할 수 있습니다.
5. 서비스 수준 지표(SLI) 설정에서 다음 중 하나를 수행합니다.
   - 표준 애플리케이션 지표인 `Latency` 또는 `Availability` 중 하나에 SLO 설정
     a. 서비스 작업을 선택합니다.
     b. 이 SLO에서 모니터링할 서비스를 선택합니다.
     c. 이 SLO에서 모니터링할 작업을 선택합니다.

        서비스 선택 및 작업 선택 드롭다운은 지난 24시간 동안 활성화된 서비스와 작업으로 채워집니다.
     d. 가용성 또는 지연 시간을 선택한 다음, 임곗값을 설정합니다.
   - 모든 CloudWatch 지표 또는 CloudWatch 지표 수학 표현식에 SLO 설정
     a. CloudWatch 지표를 선택합니다.
     b. CloudWatch 지표 선택을 선택합니다.

        지표 선택 화면이 나타납니다. 찾아보기 또는 쿼리 탭을 사용하여 원하는 지표를 찾거나 지표 수학 표현식을 생성합니다.

        원하는 지표를 선택한 후 그래프로 표시된 지표 탭을 선택하고 SLO에 사용할 통계 및 기간을 선택합니다. 그런 다음 지표 선택을 선택합니다.

        이러한 화면에 대한 자세한 내용은 지표 그래프 작성 및 CloudWatch 그래프에 수학 표현식 추가 섹션을 참조하세요.
     c. 조건 설정에서 성공 지표로 사용할 SLO의 비교 연산자와 임곗값을 선택합니다.
6. 5단계에서 서비스 작업을 선택한 경우 필요에 따라 추가 설정을 선택한 다음, 이 SLO의 기간 길이를 조정할 수 있습니다.

7. SLO의 간격과 달성 목표를 설정합니다. 간격 및 달성 목표와 이 둘의 상호 작용 방식에 대한 자세한 내용은 SLO 개념 섹션을 참조하세요.
8. (선택 사항) SLO에 대해 하나 이상의 CloudWatch 경보 또는 경고 임곗값을 설정합니다.

   a. CloudWatch 경보는 Amazon SNS를 사용하여 SLI 성능을 기준으로 애플리케이션이 비정상일 경우 사전에 알릴 수 있습니다.

      경보를 생성하려면 경보 확인란 중 하나를 선택하고 경보가 ALARM 상태가 될 때 알림에 사용할 Amazon SNS 주제를 입력하거나 생성합니다. CloudWatch 경보에 대한 자세한 내용은 Amazon CloudWatch 경보 사용 섹션을 참조하세요. 경보를 생성하면 요금이 부과됩니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

   b. 경고 임곗값을 설정하면 Application Signals 화면에 표시되어 현재 정상이더라도 충족되지 않을 위험이 있는 SLO를 식별하는 데 도움이 됩니다.

      경고 임곗값을 설정하려면 경고 임곗값에 임곗값을 입력합니다. SLO의 오류 할당이 경고 임곗값보다 낮으면 여러 Application Signals 화면에 SLO가 경고로 표시됩니다. 경고 임곗값은 오류 할당 그래프에도 표시됩니다. 경고 임곗값을 기준으로 하는 SLO 경고 경보를 생성할 수도 있습니다.

9. 이 SLO에 태그를 추가하려면 태그 탭을 선택한 다음, 새 태그 추가를 선택합니다. 태그를 사용하면 리소스를 손쉽게 관리, 식별, 정리, 검색 및 필터링할 수 있습니다. 태그 지정에 대한 자세한 내용은 AWS 리소스 태그 지정을 참조하세요.

   > **Note**
   >
   > 이 SLO와 관련된 애플리케이션이 AWS Service Catalog AppRegistry에 등록된 경우 `awsApplication` 태그를 사용하여 이 SLO를 AppRegistry의 해당 애플리케이션과 연결할 수 있습니다. 자세한 내용은 What is AppRegistry?를 참조하세요.

10. SLO 생성을 선택합니다. 경보를 하나 이상 생성하도록 선택한 경우 이를 반영하기 위해 버튼 이름이 변경됩니다.

SLO 상태 보기 및 분류

CloudWatch 콘솔에서 서비스 수준 목표 또는 서비스 옵션을 사용하여 SLO의 상태를 빠르게 확인할 수 있습니다. 서비스 뷰에서는 설정한 SLO를 기준으로 계산된 비정상 서비스 비율을 한눈에 볼 수 있습니다. 서비스 옵션 사용에 대한 자세한 내용은 Application Signals를 사용하여 애플리케이션의 운영 상태 모니터링 섹션을 참조하세요.

서비스 수준 목표 뷰는 조직의 거시적 뷰를 제공합니다. 충족된 SLO와 충족되지 않은 SLO를 전체적으로 볼 수 있습니다. 이를 통해 선택한 SLI에 따라 장기간에 걸쳐 얼마나 많은 서비스와 작업이 기대와 부합하는지 확인할 수 있습니다.

서비스 수준 목표 뷰를 사용하여 모든 SLO 보기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 서비스 수준 목표(SLO)를 선택합니다.

   서비스 수준 목표(SLO) 목록이 나타납니다.

   SLI 상태 열에서 SLO의 현재 상태를 빠르게 확인할 수 있습니다. 모든 비정상 SLO가 목록 맨 위에 오도록 SLO를 정렬하려면 비정상 SLO가 모두 맨 위에 올 때까지 SLI 상태 열을 선택합니다.

   SLO 테이블에는 다음과 같은 기본 열이 있습니다. 목록 위의 기어 아이콘을 선택하여 표시되는 열을 조정할 수 있습니다. 목표, SLI, 달성 및 간격에 대한 자세한 내용은 SLO 개념 섹션을 참조하세요.

   - SLO의 이름
   - 목표 열에는 SLO 목표를 달성하기 위해 SLI 임곗값을 성공적으로 충족해야 하는 각 간격 기간의 비율이 표시됩니다. SLO의 간격 길이도 표시됩니다.
   - SLI 상태에는 애플리케이션의 현재 작동 상태가 정상인지 여부가 표시됩니다. 현재 선택한 시간 범위 중 SLO에 대해 비정상인 기간이 있으면 SLI 상태에 비정상이 표시됩니다.
   - 종료 달성은 선택한 시간 범위가 종료되는 시점의 달성 기준입니다. 이 열을 기준으로 정렬하면 충족되지 않을 위험이 가장 높은 SLO를 확인할 수 있습니다.
   - 달성 델타는 선택한 시간 범위의 시작과 끝 사이의 달성 수준 차이입니다. 델타가 음수이면 지표가 하향 추세를 보이고 있는 것입니다. 이 열을 기준으로 정렬하면 SLO의 최신 추세를 확인할 수 있습니다.
   - 종료 오류 할당(%)은 해당 기간의 총 시간 중 비정상 기간이 있더라도 SLO를 성공적으로 달성할 수 있는 비율입니다. 이 값을 5%로 설정하고 해당 간격의 남은 기간 중 5% 이하에서 SLI가 비정상 상태인 경우에도 SLO는 여전히 성공적으로 달성됩니다.
   - 오류 할당 델타는 선택한 시간 범위의 시작과 끝 사이의 오류 할당 차이입니다. 델타가 음수이면 지표가 실패 추세를 보이고 있는 것입니다.
   - 종료 오류 할당(시간)은 해당 간격에서 비정상이면서도 SLO를 성공적으로 달성할 수 있는 실제 시간입니다. 예를 들어, 이 시간이 14분이면 남은 간격 중 14분 미만 동안 SLI가 비정상 상태인 경우에도 SLO를 성공적으로 달성할 수 있습니다.
   - 서비스, 작업 및 유형 열에는 이 SLO가 설정된 서비스와 작업에 대한 정보가 표시됩니다.

3. SLO의 달성 및 오류 할당 그래프를 보려면 SLO 이름 옆에 있는 라디오 버튼을 선택합니다.

   페이지 상단의 그래프에는 SLO 달성 및 오류 할당 상태가 표시됩니다. 이 SLO와 연결된 SLI 지표에 대한 그래프도 표시됩니다.

4. 목표에 미치지 못하는 SLO를 추가로 분류하려면 해당 SLO와 관련된 서비스 이름 또는 작업 이름을 선택합니다. 세부 정보 페이지로 이동하여 추가로 분류할 수 있습니다. 자세한 내용은 서비스 세부 정보 페이지에서 자세한 서비스 활동 및 운영 상태 확인 단원을 참조하십시오.

5. 페이지에 있는 차트와 테이블의 시간 범위를 변경하려면 화면 상단에서 새 시간 범위를 선택합니다.

기존 SLO 편집

기존 SLO를 편집하려면 다음 단계를 따르세요. SLO를 편집할 때 임곗값, 간격, 달성 목표 및 태그만 변경할 수 있습니다. 서비스, 운영 또는 지표와 같은 다른 측면을 변경하려면 기존 SLO를 편집하는 대신 새 SLO를 생성합니다.

기간 또는 임곗값과 같은 SLO 핵심 구성의 일부를 변경하면 달성 및 상태에 대한 이전의 모든 데이터 포인트 및 평가가 무효화됩니다. SLO가 효과적으로 삭제되고 다시 생성됩니다.

> **Note**
>
> SLO를 편집하는 경우 해당 SLO와 관련된 경보는 자동으로 업데이트되지 않습니다. 경보를 SLO와 동기화된 상태로 유지하려면 경보를 업데이트해야 할 수 있습니다.

기존 SLO 편집

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 서비스 수준 목표(SLO)를 선택합니다.
3. 편집하려는 SLO 옆의 라디오 단추를 선택하고 작업, SLO 편집을 선택합니다.
4. 변경한 후 변경 사항 저장을 선택합니다.

SLO 삭제

기존 SLO를 삭제하려면 다음 단계를 따르세요.

> **Note**
>
> SLO를 삭제하는 경우 해당 SLO와 관련된 경보는 자동으로 삭제되지 않습니다. 직접 삭제해야 합니다. 자세한 내용은 경보 관리 단원을 참조하십시오.

SLO 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 서비스 수준 목표(SLO)를 선택합니다.
3. 편집하려는 SLO 옆의 라디오 단추를 선택하고 작업, SLO 삭제를 선택합니다.
4. 확인(Confirm)을 선택합니다.

Application Signals를 사용하여 애플리케이션의 운영 상태 모니터링

> **Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.**

CloudWatch 콘솔 내에서 Application Signals를 사용하여 애플리케이션의 운영 상태를 모니터링하고 문제를 해결합니다.

- 애플리케이션 서비스 모니터링 - 일일 운영 모니터링의 일환으로 서비스 페이지를 사용하여 모든 서비스의 요약을 확인합니다. 장애 발생률 또는 지연 시간이 가장 긴 서비스를 확인하고 비정상 서비스 수준 지표(SLI)가 있는 서비스를 확인합니다. 서비스를 선택하여 서비스 세부 정보 페이지를 열고 세부 지표, 서비스 작업, Synthetics canary 및 클라이언트 요청을 확인합니다. 이는 운영 문제의 근본 원인을 해결하고 식별하는 데 도움이 될 수 있습니다.
- 애플리케이션 토폴로지 검사 - 서비스 맵을 사용하여 클라이언트, Synthetics canary, 서비스 및 종속성 간의 관계를 포함하여 시간 경과에 따른 애플리케이션 토폴로지를 이해하고 모니터링할 수 있습니다. 즉시 서비스 수준 지표(SLI) 상태를 확인하고 호출 볼륨, 장애 발생률, 지연 시간과 같은 주요 지표를 볼 수 있습니다. 드릴다운하여 서비스 세부 정보 페이지에서 더 자세한 정보를 확인합니다.

이러한 페이지를 사용하여 초기 탐지에서 근본 원인 식별에 이르기까지 운영 서비스 상태 문제를 신속하게 해결하는 방법을 보여주는 예제 시나리오를 살펴봅니다.

Application Signals가 운영 상태 모니터링을 활성화하는 방법

Application Signals에 대해 애플리케이션을 활성화하면 애플리케이션 서비스, API 및 해당 종속성이 자동으로 검색되어 서비스, 서비스 세부 정보 및 서비스 맵 페이지에 표시됩니다. Application Signals는 여러 소스에서 정보를 수집하여 서비스 검색 및 운영 상태 모니터링을 활성화합니다.

- AWS Distro for OpenTelemetry (ADOT) - Application Signals 활성화의 일환으로 OpenTelemetry Java 자동 계측 라이브러리는 CloudWatch 에이전트에서 수집한 지표와 트레이스를 내보내도록 구성됩니다. 지표와 트레이스는 서비스, 운영, 종속성 및 기타 서비스 정보를 검색하는 데 사용됩니다.
- 서비스 수준 목표(SLO) - 서비스에 대한 서비스 수준 목표를 생성하면 서비스, 서비스 세부 정보 및 서비스 맵 페이지에 서비스 수준 지표(SLI) 상태가 표시됩니다. SLI는 지연 시간, 가용성 및 기타 운영 지표를 모니터링할 수 있습니다.
- CloudWatch Synthetics canary - canary에서 X-Ray 추적을 구성하면 canary 스크립트에서 서비스에 대한 호출이 서비스와 연결되고 서비스 세부 정보 페이지 내에 표시됩니다.
- CloudWatch 실제 사용자 모니터링(RUM) - CloudWatch RUM 웹 클라이언트에서 X-Ray 추적을 활성화하면 서비스에 대한 요청이 자동으로 연결되어 서비스 세부 정보 페이지 내에 표시됩니다.
- AWS Service Catalog AppRegistry - Application Signals는 계정 내에서 AWS 리소스를 자동으로 검색하고 AppRegistry에서 생성된 논리적 애플리케이션으로 그룹화할 수 있도록 합니다. 서비스 페이지에 표시되는 애플리케이션 이름은 서비스가 실행되는 기본 컴퓨팅 리소스를 기반으로 합니다.

> **Note**
>
> Application Signals는 선택한 현재 시간 필터 내에서 내보낸 지표와 트레이스를 기반으로 서비스와 작업을 표시합니다. 기본적으로 지난 3시간입니다. 서비스, 작업, 종속성, Synthetics canary 또는 클라이언트 페이지에 대한 현재 시간 필터 내에 활동이 없는 경우 해당 활동은 표시되지 않습니다.
> 현재 최대 1,000개의 서비스를 표시할 수 있습니다. 서비스 및 서비스 토폴로지 검색은 최대 10분까지 지연될 수 있습니다. 서비스 수준 지표(SLI) 상태 평가가 최대 15분까지 지연될 수 있습니다.

서비스 페이지로 전체 서비스 활동 및 운영 상태 보기

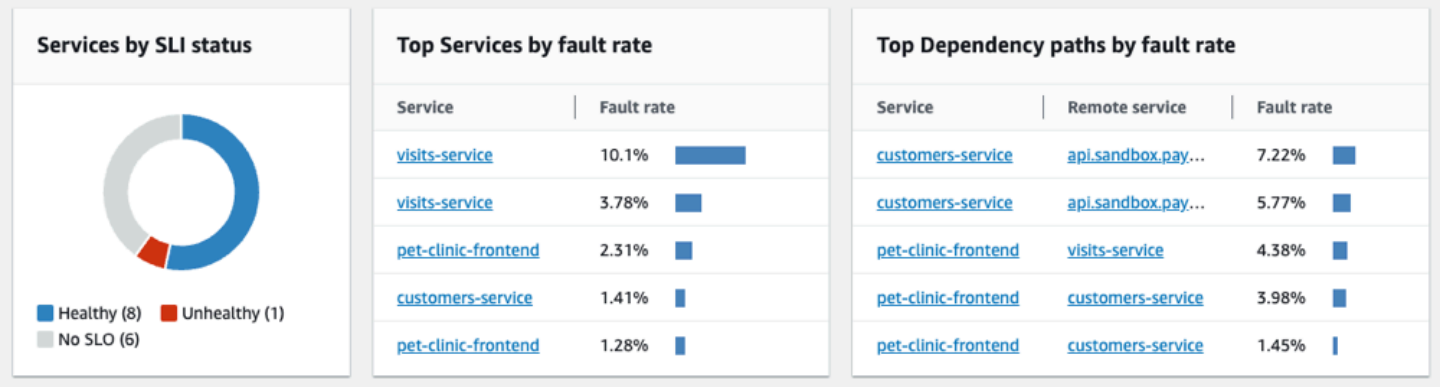
> **Application Signals는 Amazon CloudWatch의 평가판 릴리스에 있으며 변경될 수 있습니다.**

서비스 페이지를 사용하여 Application Signals에 대해 활성화된 서비스 목록을 확인합니다. 또한 운영 지표를 보고 어떤 서비스에 비정상 서비스 수준 지표(SLI)가 있는지 빠르게 확인할 수 있습니다. 운영 문제의 근본 원인을 파악하면서 드릴다운하여 성능 이상을 찾습니다. 이 페이지를 보려면 CloudWatch 콘솔을 열고 왼쪽 탐색 창의 Application Signals 섹션에서 서비스를 선택합니다.

서비스의 운영 상태 지표 탐색

서비스 페이지 상단에는 전체 서비스 작업 상태 그래프와 장애 발생률별 상위 서비스 및 서비스 종속성을 보여주는 여러 테이블이 있습니다. 왼쪽의 서비스 그래프에는 현재 페이지 수준 시간 필터 동안 정상 또는 비정상 서비스 수준 지표(SLI)가 있는 서비스 수의 분석 결과가 표시됩니다. SLI는 지연 시간, 가용성 및 기타 운영 지표를 모니터링할 수 있습니다.

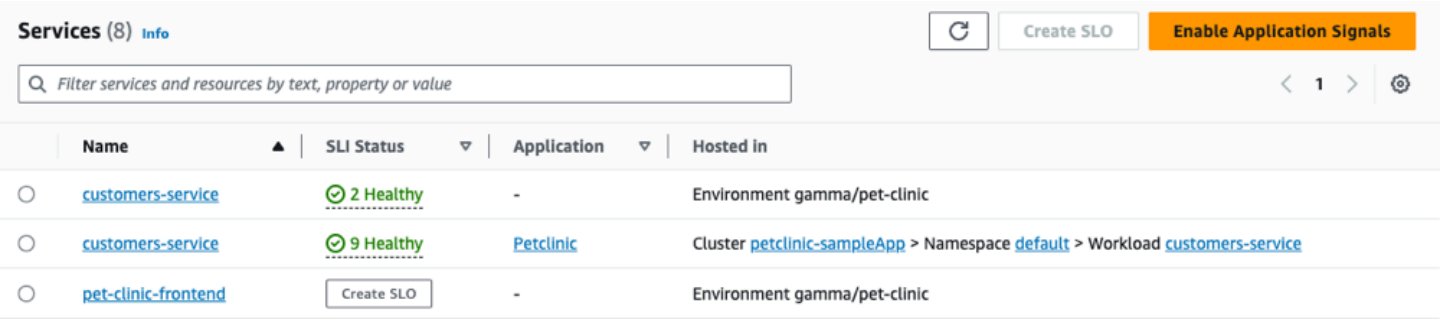
그래프 옆의 두 테이블에는 장애 발생률별 상위 서비스 목록이 표시됩니다. 두 테이블 중 하나에서 서비스 이름을 선택하면 서비스 세부 정보 페이지가 열리고 자세한 서비스 작업 세부 정보를 볼 수 있습니다. 종속성 경로를 선택하여 세부 정보 페이지를 열고 서비스 종속성 세부 정보를 확인할 수 있습니다. 페이지 오른쪽 상단에서 더 긴 기간 필터를 선택해도 두 테이블 모두 최대 최근 3시간 동안의 정보를 표시합니다.



서비스 테이블로 운영 상태 모니터링

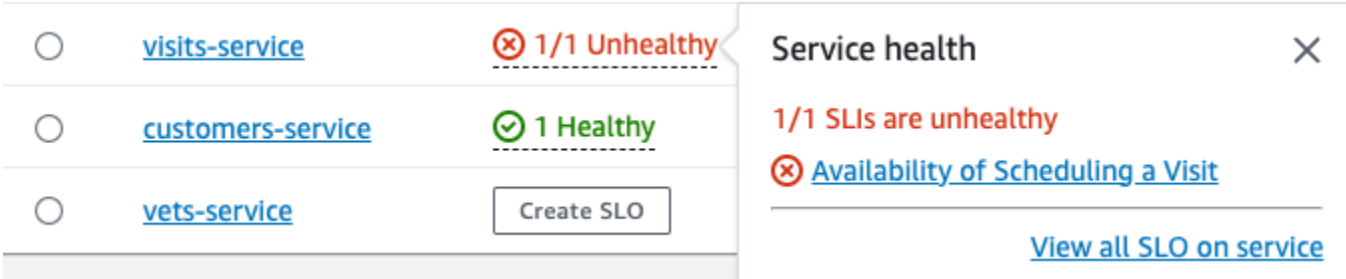
서비스 테이블에는 Application Signals에 대해 활성화된 서비스 목록이 표시됩니다. Application Signals 활성화를 선택하여 설정 페이지를 열고 서비스 구성을 시작합니다. 자세한 내용은 Application Signals 활성화를 참조하세요.

필터 텍스트 상자에서 하나 이상의 속성을 선택하여 원하는 항목을 더 쉽게 찾을 수 있도록 서비스 테이블을 필터링합니다. 각 속성을 선택하면 필터 기준이 안내됩니다. 필터 텍스트 상자 아래에 전체 필터가 표시됩니다. 언제든지 필터 지우기를 선택하여 테이블 필터를 제거할 수 있습니다.



테이블에서 서비스 이름을 선택하면 서비스 수준 지표, 운영 및 추가 세부 정보가 포함된 서비스 세부 정보 페이지를 볼 수 있습니다. 서비스의 기본 컴퓨팅 리소스를 AppRegistry의 애플리케이션 또는 AWS Management Console 홈페이지의 애플리케이션 카드와 연결한 경우 애플리케이션 이름을 선택하여 myApplications 콘솔 페이지에 애플리케이션 세부 정보를 표시합니다. Amazon EKS에서 호스팅되는 서비스의 경우, 호스팅 위치 열 내의 링크를 선택하면 CloudWatch Container Insights 내에서 클러스터, 네임스페이스 또는 워크로드를 볼 수 있습니다. Amazon ECS 또는 Amazon EC2에서 실행되는 서비스의 경우 환경 값이 표시됩니다.

테이블의 각 서비스에 대한 서비스 수준 지표(SLI) 상태가 표시됩니다. 서비스의 SLI 상태를 선택하면 비정상 SLI에 대한 링크와 해당 서비스에 대한 모든 SLO를 볼 수 있는 링크가 포함된 팝업이 표시됩니다.

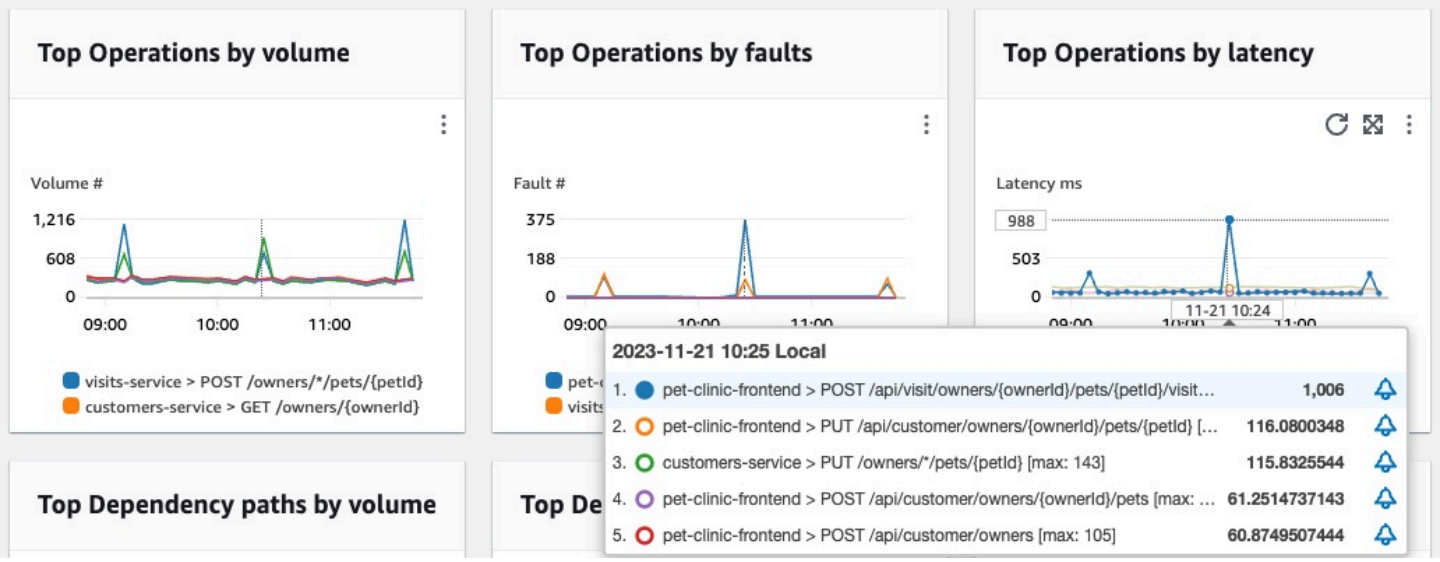


서비스에 대한 SLO가 생성되지 않은 경우 SLI 상태 열에서 SLO 생성 버튼을 선택합니다. 서비스에 대해 추가 SLO를 생성하려면 서비스 이름 옆에 있는 옵션 버튼을 선택한 다음, 테이블 오른쪽 상단에서 SLO 생성을 선택합니다. SLO를 생성하면 어떤 서비스와 작업이 잘 수행되고 있고, 어떤 것이 비정상인지 한눈에 파악할 수 있습니다. 자세한 내용은 서비스 수준 목표(SLO)를 참조하세요.

주요 작업 및 종속성 지표 보기

서비스 테이블 아래에서는 호출 볼륨, 장애, 지연 시간별로 모든 서비스의 상위 운영 및 종속성을 확인할 수 있습니다. 이 그래프 세트는 모든 서비스에서 비정상적일 수 있는 운영 또는 종속성에 대한 중요한 정보를 제공합니다. 그래프에서 아무 지점이나 선택하면 자세한 시리즈 정보가 포함된 팝업이 표시됩니다. 그래프 하단의 시리즈 설명을 가리키면 특정 작업 또는 종속성 경로에 대한 세부 지표가 포함

된 팝업이 표시됩니다. 그래프 오른쪽 상단의 컨텍스트 메뉴 버튼을 선택하면 CloudWatch 지표 또는 로그 페이지 보기를 비롯한 추가 옵션이 표시됩니다.



서비스 세부 정보 페이지에서 자세한 서비스 활동 및 운영 상태 확인

⚠ Application Signals는 Amazon CloudWatch의 평가판 릴리스에 있으며 변경될 수 있습니다.

애플리케이션을 계측할 때 Amazon CloudWatch Application Signals는 애플리케이션이 검색하는 모든 서비스를 매핑합니다. 서비스 세부 정보 페이지를 사용하여 단일 서비스에 대한 서비스 개요, 작업, 종속성, canary 및 클라이언트 요청을 확인합니다. 이러한 서비스 세부 정보 페이지를 보려면 다음을 수행합니다.

- CloudWatch 콘솔을 엽니다.
- 왼쪽 탐색 창의 Application Signals 섹션에서 서비스를 선택합니다.
- 서비스, 상위 서비스 또는 종속성 테이블에서 서비스 이름을 선택합니다.

서비스 세부 정보 페이지는 다음 탭으로 구성되어 있습니다.

- 개요 - 이 탭을 사용하여 작업, 종속성, Synthetics 및 클라이언트 페이지 수를 비롯한 단일 서비스 개요를 확인합니다. 탭에는 전체 서비스, 상위 작업 및 종속성에 대한 주요 지표가 표시됩니다. 이러한 지표로, 해당 서비스의 모든 서비스 작업 전반에 걸친 지연 시간, 장애 및 오류에 대한 시계열 데이터가 포함됩니다.

- 서비스 작업 - 이 탭을 사용하여 서비스에서 직접 호출하는 작업 목록과 각 작업 상태를 측정하는 주요 지표가 포함된 대화형 그래프를 확인합니다. 그래프에서 데이터 포인트를 선택하여 해당 데이터 포인트와 연결된 트레이스, 로그 또는 지표에 대한 정보를 얻을 수 있습니다.
- 종속성 - 이 탭을 사용하여 서비스에서 직접 호출하는 종속성 목록과 종속성 지표 목록을 확인합니다.
- Synthetics Canary - 이 탭을 사용하여 서비스에 대한 사용자 직접 호출을 시뮬레이션하는 Synthetics canary 목록과 해당 canary의 작동 방식에 대한 주요 성능 지표를 확인합니다.
- 클라이언트 페이지 - 이 탭을 사용하여 서비스를 직접 호출하는 클라이언트 페이지 목록 및 애플리케이션과 클라이언트의 상호 작용 품질을 측정하는 지표를 확인합니다.

서비스 개요 보기

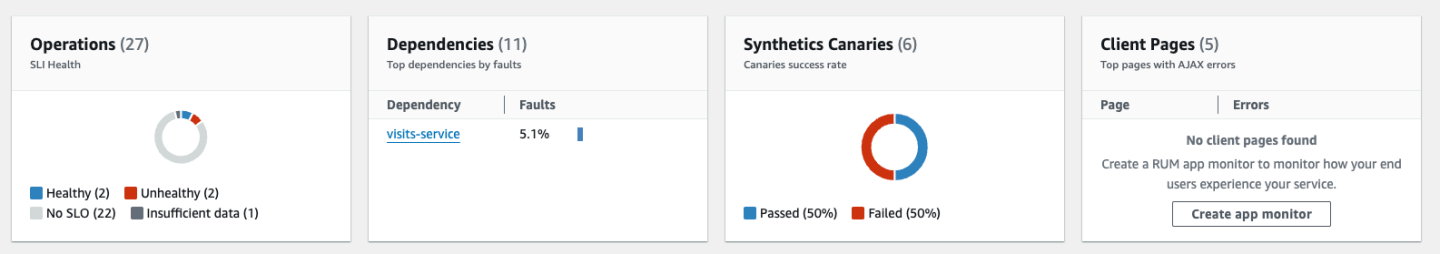
서비스 개요 페이지를 사용하여 단일 위치에서 모든 서비스 작업에 대한 개요 수준의 지표 요약을 확인합니다. 애플리케이션과 상호 작용하는 모든 작업, 종속성, 클라이언트 페이지 및 Synthetics canary의 성능을 확인합니다. 이 정보를 사용하면 문제를 식별하고, 오류를 해결하며, 최적화 기회를 찾기 위해 집중할 부분을 결정하는 데 도움이 됩니다.

서비스 세부 정보의 링크를 선택하면 특정 서비스와 관련된 정보를 확인합니다. 예를 들어 Amazon EKS에 호스팅되는 서비스의 경우 서비스 세부 정보 페이지에 클러스터, 네임스페이스 및 워크로드 정보가 표시됩니다. Amazon ECS 또는 Amazon EC2에 호스팅되는 서비스의 경우 서비스 세부 정보 페이지에 환경 값이 표시됩니다.

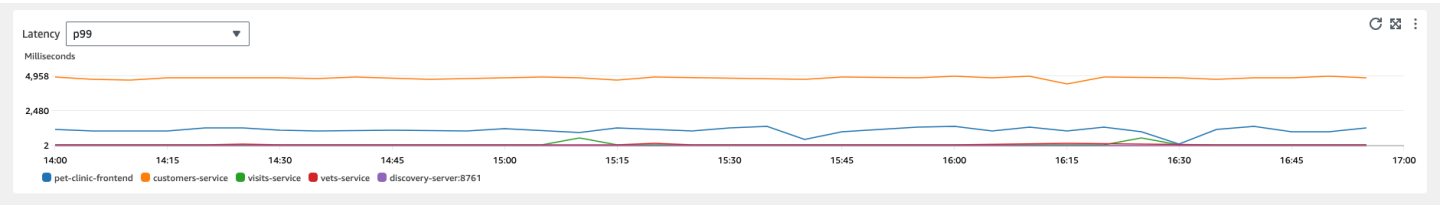
서비스에서 개요 탭에는 다음이 요약되어 표시됩니다.

- 작업 - 이 탭을 사용하여 서비스 작업 상태를 확인합니다. 서비스 수준 목표(SLO)의 일부로 정의된 서비스 수준 지표(SLI)로 상태가 결정됩니다.
- 종속성 - 이 테이블을 사용하여 애플리케이션에서 직접 호출하는 서비스의 상위 종속성(장애 발생률로 나열됨)을 확인합니다.
- Synthetics Canary - 이 탭을 사용하여 서비스와 연결된 엔드포인트 또는 API에 대한 시뮬레이션 직접 호출 결과와 실패한 canary 수를 확인합니다.
- 클라이언트 페이지 - 이 탭을 사용하여 비동기 JavaScript 및 XML(AJAX) 오류가 있는 클라이언트가 직접 호출한 상위 페이지를 확인합니다.

다음 그림은 서비스의 개요를 보여줍니다.

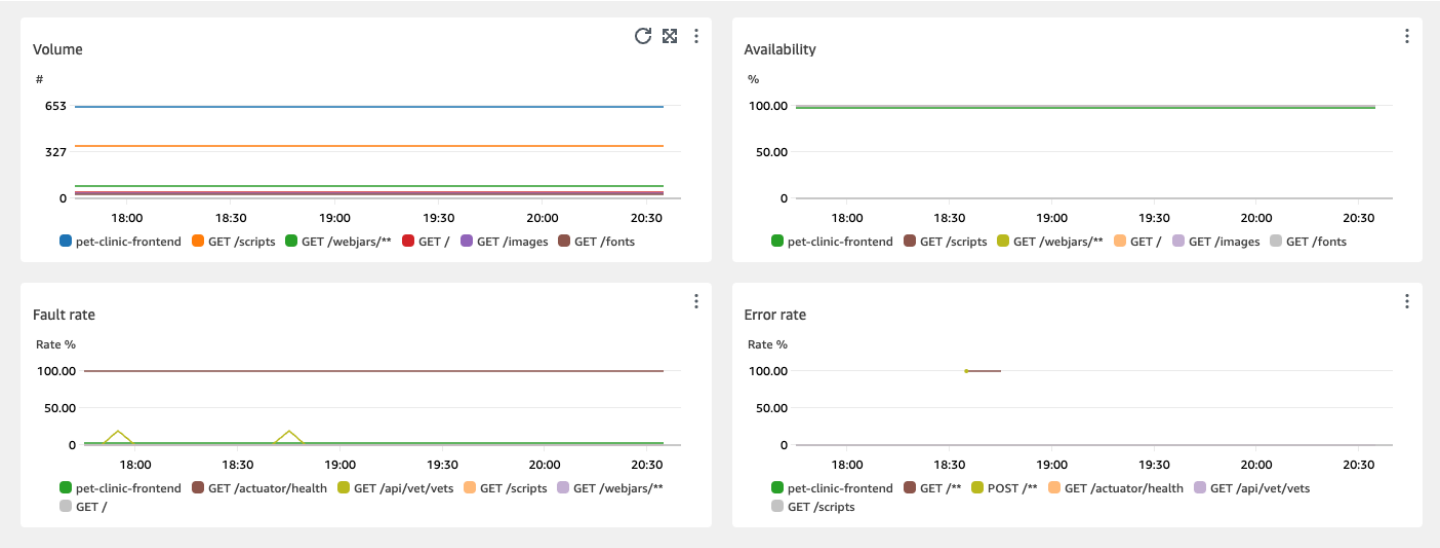


또한 개요 탭에는 모든 서비스에서 지연 시간이 가장 긴 종속성 그래프도 표시됩니다. 다음과 같이 p99, p90, p50 지연 시간 지표를 사용하여 총 서비스 지연 시간에 영향을 미치는 종속성을 신속하게 평가할 수 있습니다.



예를 들어, 이전 그래프는 고객-서비스 종속성에 대한 요청의 99%가 약 4,950밀리초 만에 완료되었음을 보여줍니다. 다른 종속성은 그보다 시간이 더 적게 걸렸습니다.

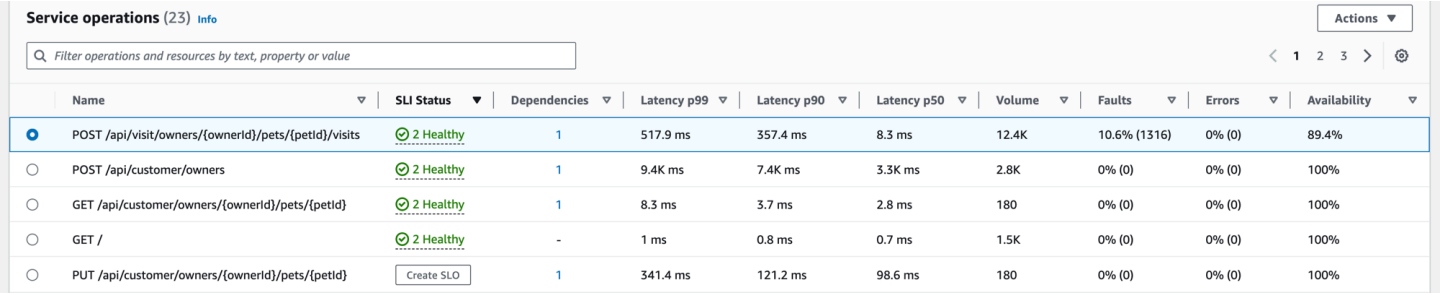
지연 시간별로 상위 4개 서비스 작업을 표시하는 그래프는 다음 이미지와 같이 해당 서비스에 대한 요청량, 가용성, 결함율 및 오류율을 보여줍니다.



서비스 작업 보기

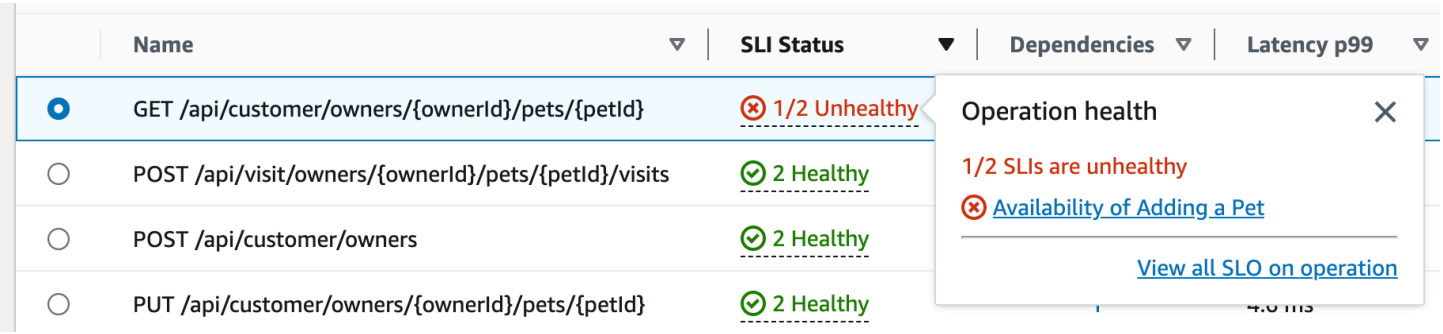
애플리케이션을 계측할 때 Application Signals는 애플리케이션이 직접 호출하는 모든 서비스 작업을 검색합니다. 서비스 작업 탭을 사용하여 서비스 작업과 선택한 작업의 성능을 측정하는 지표 집합이 포

함된 테이블을 확인합니다. 이러한 지표로는 다음 이미지와 같이 SLI 상태, 종속성 수, 지연 시간, 볼륨, 장애, 오류 및 가용성이 포함됩니다.



테이블을 필터링하여 필터 텍스트 상자에서 하나 이상의 속성을 선택해 서비스 작업을 더 쉽게 찾을 수 있습니다. 각 속성을 선택하면 필터 기준이 안내되며 필터 텍스트 상자 아래에 전체 필터가 표시됩니다. 언제든지 필터 지우기를 선택하여 테이블 필터를 제거할 수 있습니다.

다음 테이블과 같이 작업의 SLI 상태를 선택하여 비정상 SLI에 대한 링크와 해당 작업에 대한 모든 SLO를 볼 수 있는 링크가 포함된 팝업을 표시합니다.



서비스 작업 테이블에는 SLI 상태, 정상 또는 비정상 SLI 수, 각 작업에 대한 총 SLO 수가 나열됩니다.

SLI를 사용하여 지연 시간, 가용성 및 서비스의 작업 상태를 측정하는 기타 작업 지표를 모니터링합니다. SLO를 사용하여 서비스 및 작업의 성능과 상태를 확인합니다.

SLO를 생성하려면 다음을 수행합니다.

- 작업에 SLO가 없는 경우 SLI 상태 열에서 SLO 생성 버튼을 선택합니다.
- 작업에 이미 SLO가 있는 경우 다음을 수행합니다.
  - 작업 이름 옆의 라디오 버튼을 선택합니다.
  - 테이블 오른쪽 상단의 작업 아래쪽 화살표에서 SLO 생성을 선택합니다.

자세한 내용은 서비스 수준 목표(SLO)를 참조하세요.

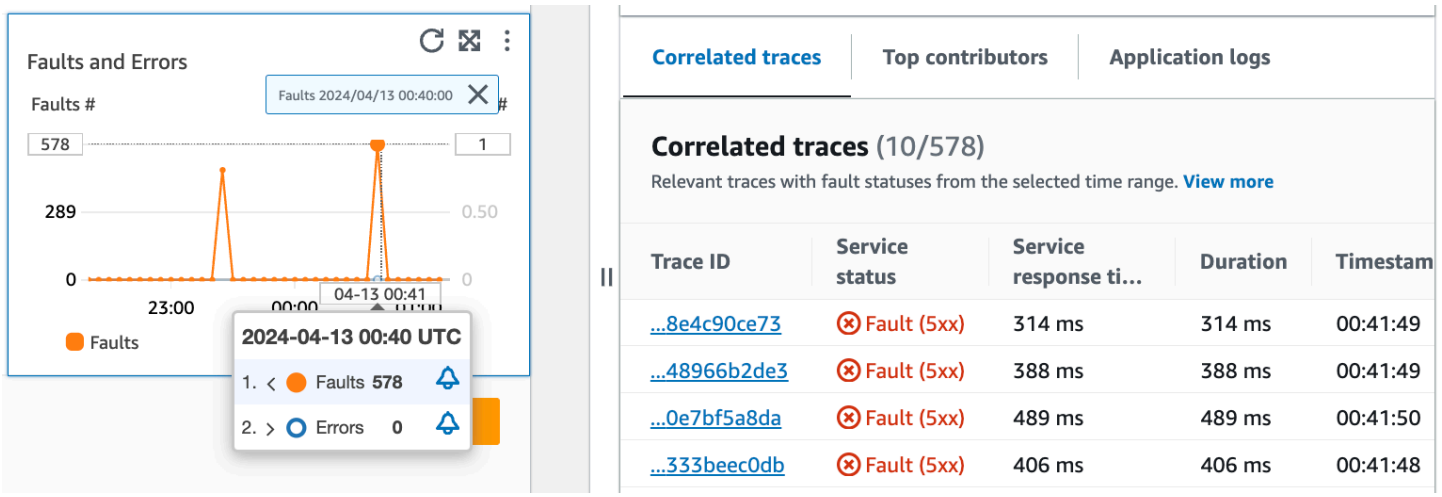
종속성 열에는 이 작업에서 직접적으로 호출하는 종속성 수가 표시됩니다. 이 숫자를 선택하여 선택한 작업으로 필터링된 종속성 탭을 엽니다.

서비스 작업 지표, 상관관계가 있는 트레이스 및 애플리케이션 로그 보기

Application Signals는 서비스 작업 지표를 AWS X-Ray 트레이스, CloudWatch Container Insights 및 애플리케이션 로그와 상관시킵니다. 이러한 지표를 사용하여 작업 상태 문제를 해결합니다. 지표를 그래픽 정보로 보려면 다음을 수행합니다.

1. 서비스 작업 테이블에서 서비스 작업을 선택하여 테이블 위에서 볼륨 및 가용성, 지연 시간 및 장애 및 오류에 대한 지표가 포함된 선택한 작업 그래프 세트를 확인합니다.
2. 그래프의 한 지점을 가리키면 추가 정보가 표시됩니다.
3. 점을 선택하여 그래프에서 선택한 지점의 상관관계가 있는 트레이스, 지표 및 애플리케이션 로그를 보여주는 진단 창이 열립니다.

다음 이미지는 그래프에서 특정 지점을 가리키면 표시되는 도구 설명과 지점을 클릭하면 표시되는 진단 창을 보여줍니다. 도구 설명에는 결함 및 오류 그래프의 관련 데이터 요소에 관한 정보가 포함되어 있습니다. 창에는 선택한 지점과 관련된 상관관계가 있는 트레이스, 상위 기여자, 애플리케이션 로그가 포함되어 있습니다.



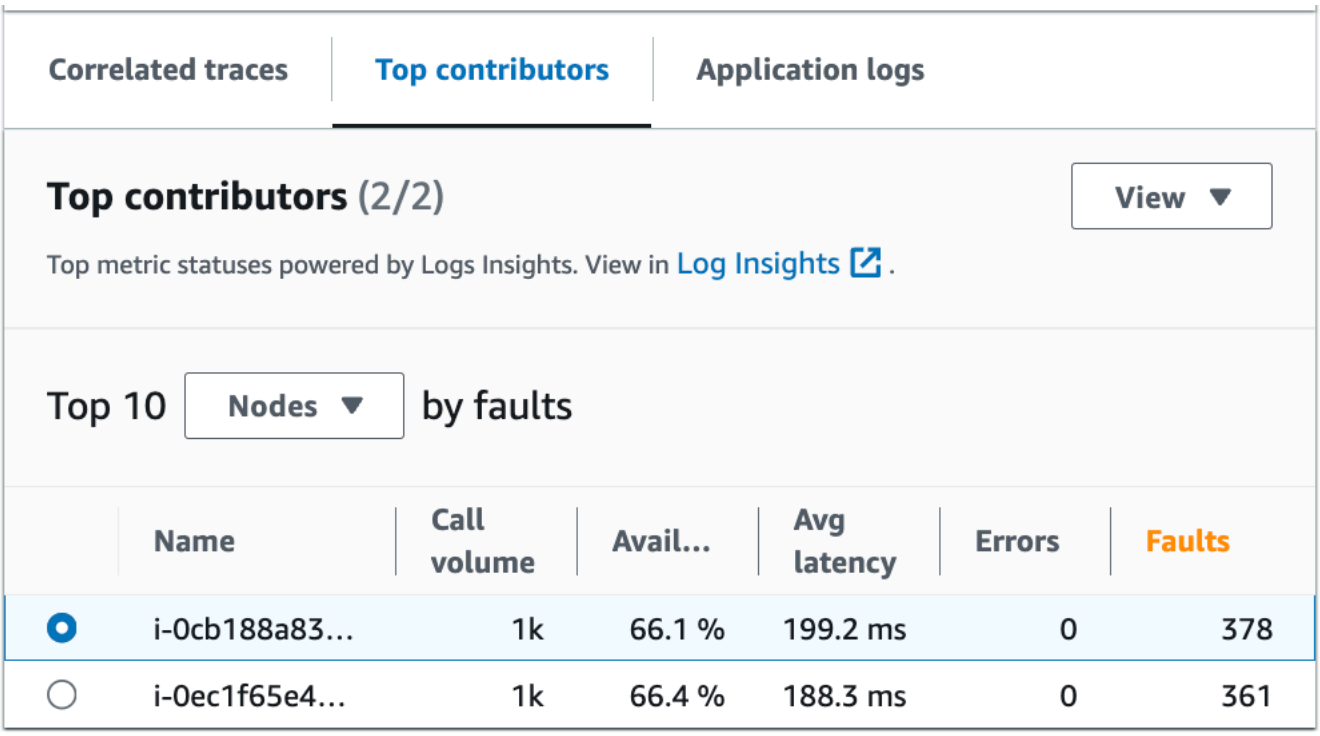
상관관계가 있는 트레이스

관련 트레이스를 살펴보고 트레이스와 관련된 근본적인 문제를 이해합니다. 상관관계가 있는 트레이스 또는 해당 트레이스와 연결된 모든 서비스 노드가 비슷하게 동작하는지 확인할 수 있습니다. 상관관계가 있는 트레이스를 검사하려면 상관관계가 있는 트레이스 테이블에서 트레이스 ID를 선택하고 선택한 트레이스의 X-Ray 트레이스 세부 정보 페이지를 엽니다. 트레이스 세부 정보 페이지에는 선택한 트레이스와 관련된 서비스 노드 맵과 트레이스 세그먼트의 타임라인이 포함되어 있습니다.

상위 기여자

지표에 대한 주요 입력 소스를 찾으려면 상위 기여자를 확인합니다. 기여자를 여러 구성 요소별로 그룹화하여 그룹 내 유사점을 찾고 트레이스 동작이 해당 그룹 사이에서 어떻게 다른지 파악합니다.

상위 기여자 탭에는 각 그룹의 직접 호출 볼륨, 가용성, 평균 지연 시간, 오류, 장애 지표가 제공됩니다. 다음 이미지 예제에서는 Amazon EKS 플랫폼에 배포된 애플리케이션의 지표 모음에 기여한 상위 기여자를 보여줍니다.



상위 기여자에는 다음 지표가 포함됩니다.

- 직접 호출 볼륨 - 직접 호출 볼륨을 사용하여 일정 기간 그룹에 대한 요청 수를 이해합니다.
- 가용성 - 가용성을 사용하여 그룹에서 장애가 감지되지 않은 시간의 비율을 확인합니다.
- 평균 지연 시간 - 지연 시간을 사용하여 조사 중인 요청이 이루어진 후 경과된 시간에 따라 일정 기간 그룹에 대해 요청이 실행된 평균 시간을 확인합니다. 15일 이내 이전에 이루어진 요청은 1분 간격으로 평가됩니다. 15~30일 이전에 이루어진 요청은 5분 간격으로 평가됩니다. 예를 들어 15일 전에 결함을 일으킨 요청을 조사하는 경우 호출량 지표는 5분 간격당 요청 수와 동일합니다.
- 오류 - 일정 기간 측정된 그룹당 오류 수.
- 장애 - 일정 기간 그룹당 결함 수.

Amazon EKS 또는 Kubernetes를 사용하는 상위 기여자

Amazon EKS 또는 Kubernetes에 배포된 애플리케이션의 경우 상위 기여자에 대한 정보를 사용하여 Node, Pod 및 PodTemplateHash로 그룹화된 작업 상태 지표를 확인합니다. 다음 정의가 적용됩니다.

- 포드는 스토리지와 리소스를 공유하는 하나 이상의 Docker 컨테이너 그룹입니다. 포드는 Kubernetes 플랫폼에 배포할 수 있는 가장 작은 단위입니다. 포드별로 그룹화하여 오류가 포드별 제한과 관련이 있는지 확인하세요.
- 노드는 포드를 실행하는 서버입니다. 노드별로 그룹화하여 오류가 노드별 제한과 관련이 있는지 확인하세요.
- 포드 템플릿 해시는 배포의 특정 버전을 찾는 데 사용됩니다. 포드 템플릿 해시별로 그룹화하여 오류가 특정 배포와 관련이 있는지 확인하세요.

Amazon EC2를 사용하는 상위 기여자

Amazon EKS에 배포된 애플리케이션의 경우 상위 기여자에 대한 정보를 사용하여 인스턴스 ID 및 Auto Scaling 그룹으로 그룹화된 작업 상태 지표를 확인합니다. 다음 정의가 적용됩니다.

- 인스턴스 ID는 서비스가 실행하는 Amazon EC2 인스턴스의 고유 식별자입니다. 인스턴스 ID별로 그룹화하여 오류가 특정 Amazon EC2 인스턴스와 관련이 있는지 확인합니다.
- Auto Scaling 그룹은 애플리케이션 요청을 처리하는 데 필요한 리소스를 확장 또는 축소할 수 있는 Amazon EC2 인스턴스의 컬렉션입니다. 오류가 그룹 내 인스턴스로 범위가 제한되어 있는지 확인하려면 Auto Scaling 그룹으로 그룹화합니다.

사용자 지정 플랫폼을 사용하는 상위 기여자

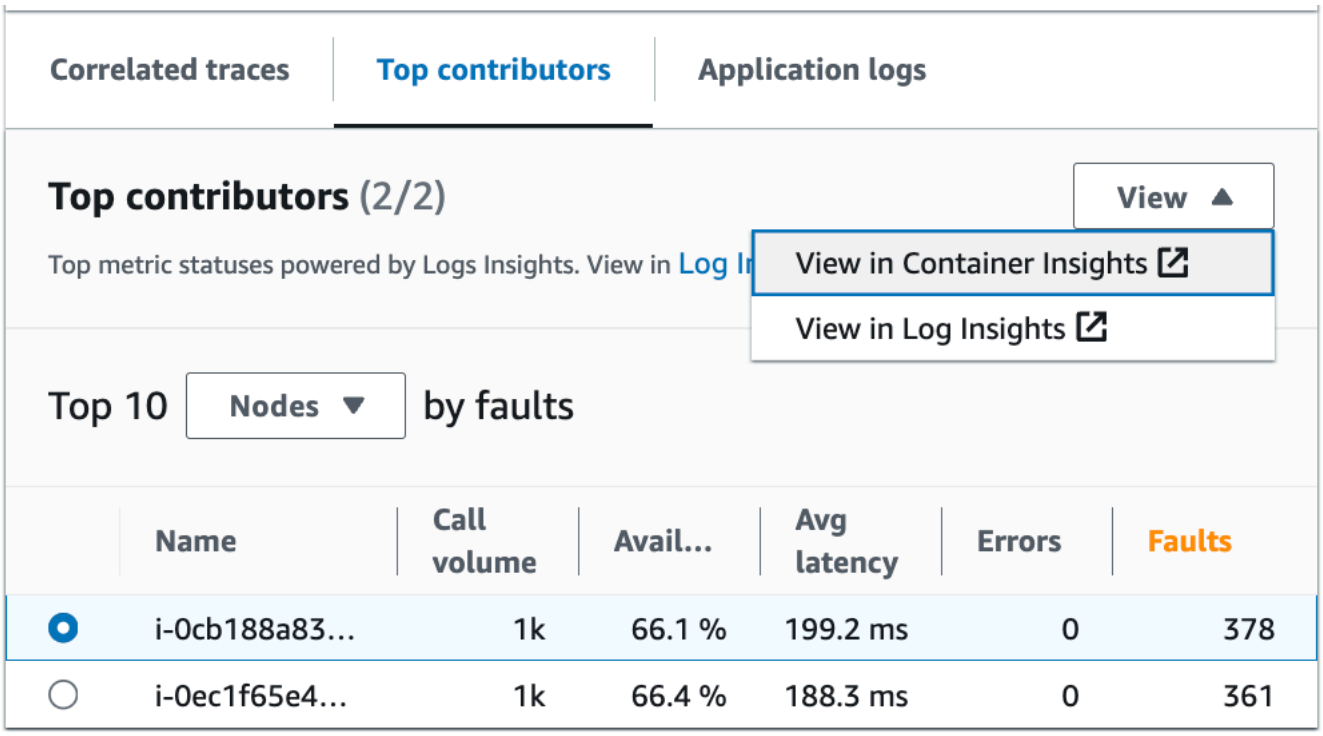
사용자 지정 계측을 사용하여 배포된 애플리케이션의 경우 상위 기여자에 대한 정보를 사용하여 호스트 이름으로 그룹화된 작업 상태 지표를 확인합니다. 다음 정의가 적용됩니다.

- 호스트 이름은 네트워크에 연결된 엔드포인트 또는 Amazon EC2 인스턴스와 같은 디바이스를 식별합니다. 호스트 이름으로 그룹화하여 오류가 특정 물리적 또는 가상 디바이스와 관련이 있는지 확인합니다.

Log Insights 및 Container Insights에서 상위 기여자 보기

Log Insights에서 상위 기여자에 대한 지표를 생성하는 자동 쿼리를 보고 수정합니다. Container Insights에서 포드 또는 노드와 같은 특정 그룹별로 인프라 성능 지표를 봅니다. 리소스 사용량을 기준으로 클러스터, 노드 또는 워크로드를 정렬하고 최종 사용자 환경이 영향을 받기 전에 이상 현상을 신

속하게 식별하거나 위험을 사전에 완화할 수 있습니다. 다음 이미지는 이러한 옵션을 선택하는 방법을 보여줍니다.



Container Insights에서는 Amazon EKS 또는 Amazon ECS 컨테이너에 대한 상위 기여자 그룹별 지표를 볼 수 있습니다. 예를 들어, 상위 기여자를 생성하기 위해 EKS 컨테이너를 포드별로 그룹화한 경우 Container Insights는 해당 포드에 대해 필터링된 지표와 통계를 보여줍니다.

Log Insights에서 다음 단계를 사용하여 상위 기여자 아래에서 지표를 생성하는 쿼리를 수정할 수 있습니다.

1. Log Insights에서 보기를 선택합니다. 열리는 Logs Insights 페이지에는 자동으로 생성되는 쿼리가 포함되어 있으며 다음 정보가 포함되어 있습니다.

   - 로그 클러스터 그룹 이름.
   - CloudWatch를 사용하여 조사하고 있던 작업.
   - 그래프에서 상호 작용한 작업 상태 지표의 집계.

   로그 결과는 서비스 그래프에서 데이터 포인트를 선택하기 전 마지막 5분 동안의 데이터를 표시하도록 자동 필터링됩니다.
2. 쿼리를 편집하려면 생성된 텍스트를 변경하는 내용으로 바꿉니다. 쿼리 생성기를 사용하여 새 쿼리를 생성하거나 기존 쿼리를 업데이트할 수도 있습니다.

애플리케이션 로그

애플리케이션 로그 탭의 쿼리를 사용하여 현재 로그 그룹 및 서비스에 대해 기록된 정보를 생성하고 타임스탬프를 삽입합니다. 로그 그룹은 애플리케이션을 구성할 때 정의할 수 있는 로그 스트림의 그룹입니다.

로그 그룹을 사용하여 다음을 비롯한 유사한 특성의 로그를 구성합니다.

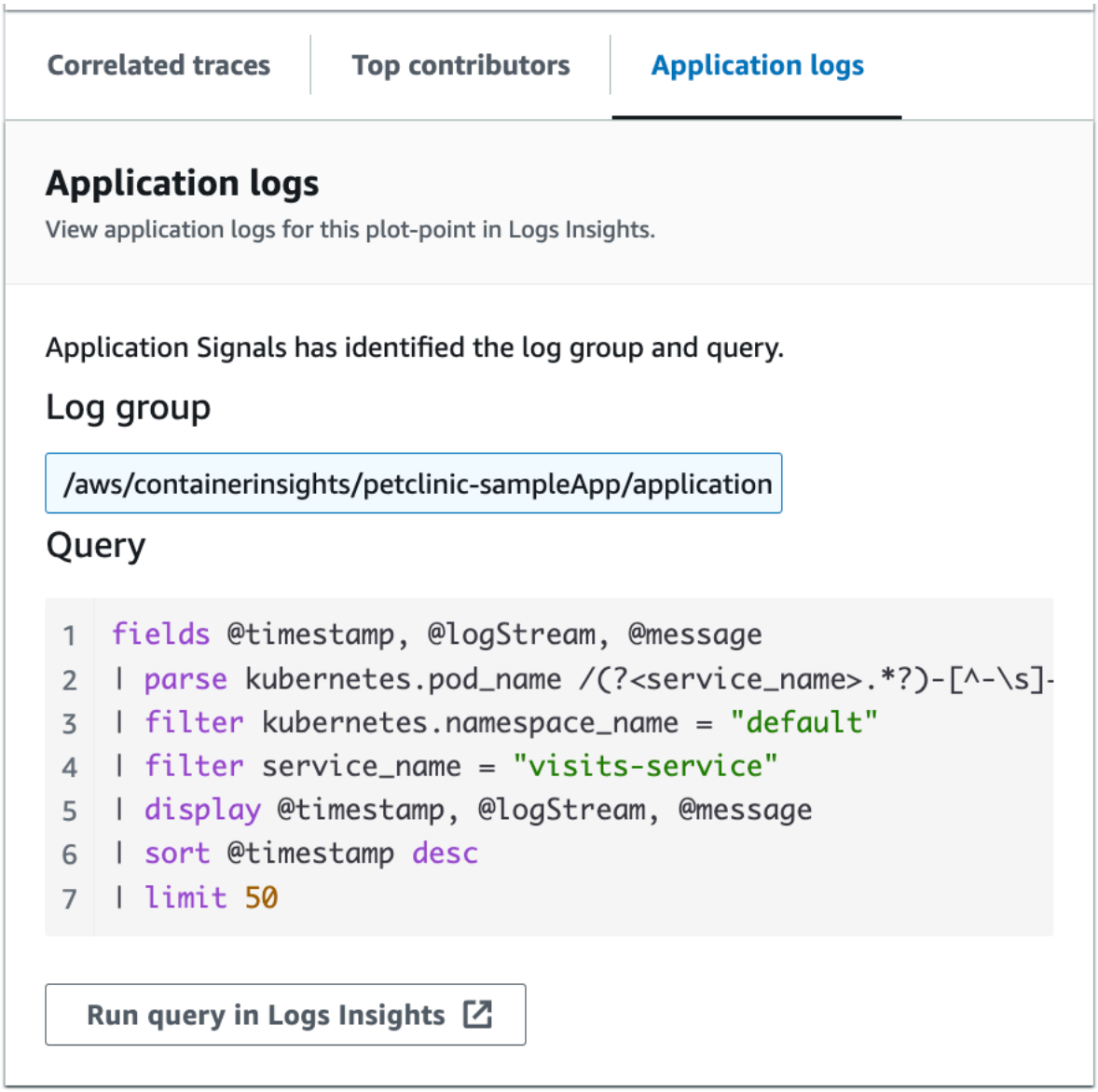
- 특정 조직, 소스 또는 기능에서 로그를 캡처합니다.
- 특정 사용자가 액세스하는 로그를 캡처합니다.
- 특정 시간 동안 로그를 캡처합니다.

이러한 로그 스트림을 사용하여 특정 그룹 또는 기간을 추적합니다. 또한 이러한 로그 그룹에 대한 모니터링 규칙, 경보 및 알림을 설정할 수 있습니다. 로그 그룹에 대한 자세한 내용은 로그 그룹 및 로그 스트림 작업을 참조하세요.

애플리케이션 로그 쿼리는 로그, 반복 텍스트 패턴 및 로그 그룹에 대한 그래픽 시각화를 반환합니다.

쿼리를 실행하려면 Logs Insights에서 쿼리 실행을 선택하여 자동 생성된 쿼리를 실행하거나 쿼리를 수정합니다. 쿼리를 편집하려면 자동 생성된 텍스트를 변경 내용으로 바꿉니다. 쿼리 생성기를 사용하여 새 쿼리를 생성하거나 기존 쿼리를 업데이트할 수도 있습니다.

다음 이미지는 서비스 작업 그래프에서 선택한 지점을 기반으로 자동 생성되는 샘플 쿼리를 보여줍니다.



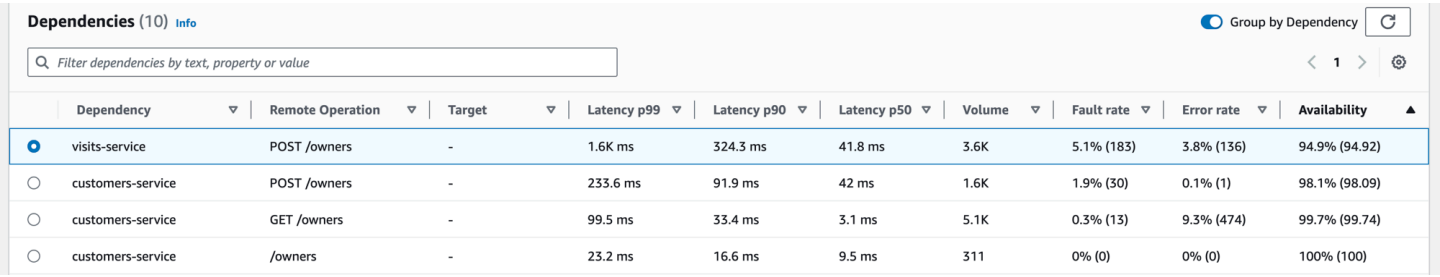
위 이미지에서 CloudWatch는 선택한 지점과 관련된 로그 그룹을 자동으로 감지하고, 이를 생성된 쿼리에 포함시켰습니다.

서비스 종속성 보기

종속성 테이블과 모든 서비스 작업 또는 단일 작업의 종속성에 대한 지표 세트를 표시하려면 종속성 탭을 선택합니다. 이 테이블에는 지연 시간, 호출 볼륨, 장애 발생률, 오류율 및 가용성에 대한 지표를 포함하여 Application Signals에서 검색한 종속성 목록이 포함되어 있습니다.

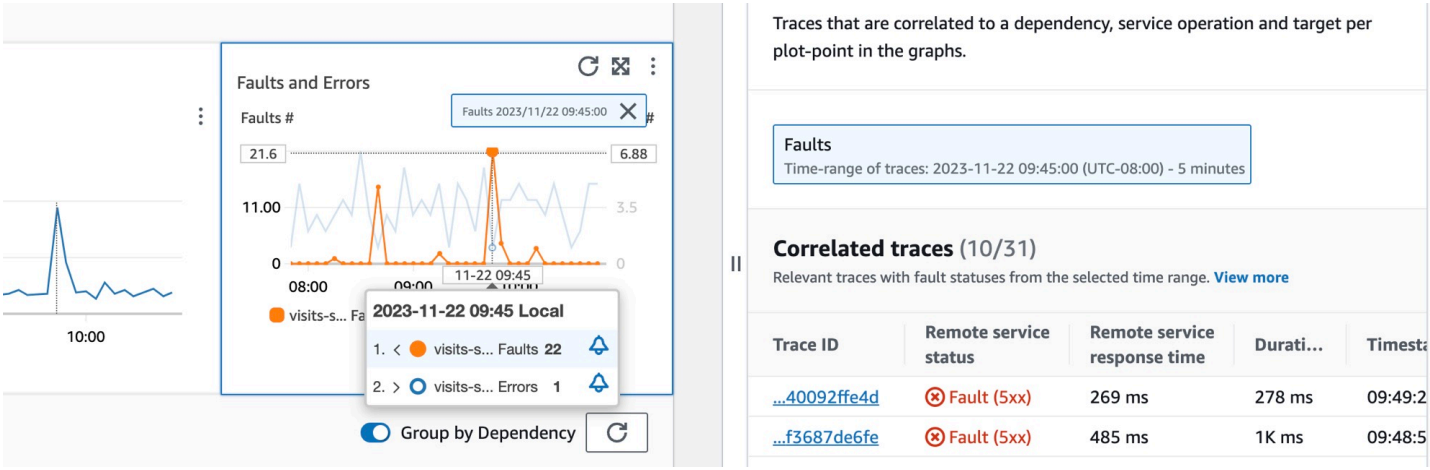
페이지 상단의 아래쪽 화살표에서 작업을 선택하여 해당 종속성을 보거나 모두를 선택하여 모든 작업에 대한 종속성을 확인합니다.

필터 텍스트 상자에서 하나 이상의 속성을 선택하여 원하는 항목을 더 쉽게 찾을 수 있도록 테이블을 필터링합니다. 각 속성을 선택하면 필터 기준이 안내되며 필터 텍스트 상자 아래에 전체 필터가 표시됩니다. 언제든지 필터 지우기를 선택하여 테이블 필터를 제거할 수 있습니다. 테이블의 오른쪽 상단에서 종속성별 그룹화를 선택하여 서비스 및 작업 이름별로 종속성을 그룹화합니다. 그룹화가 켜져 있는 경우 종속성 이름 옆에 있는 + 아이콘을 사용하여 종속성 그룹을 확장하거나 축소합니다.



종속성 열에는 종속성 서비스 이름이 표시되고, 원격 작업 열에는 서비스 작업 이름이 표시됩니다. AWS 서비스를 호출할 때 대상 열에는 DynamoDB 테이블 또는 Amazon SNS 대기열과 같은 AWS 리소스가 표시됩니다.

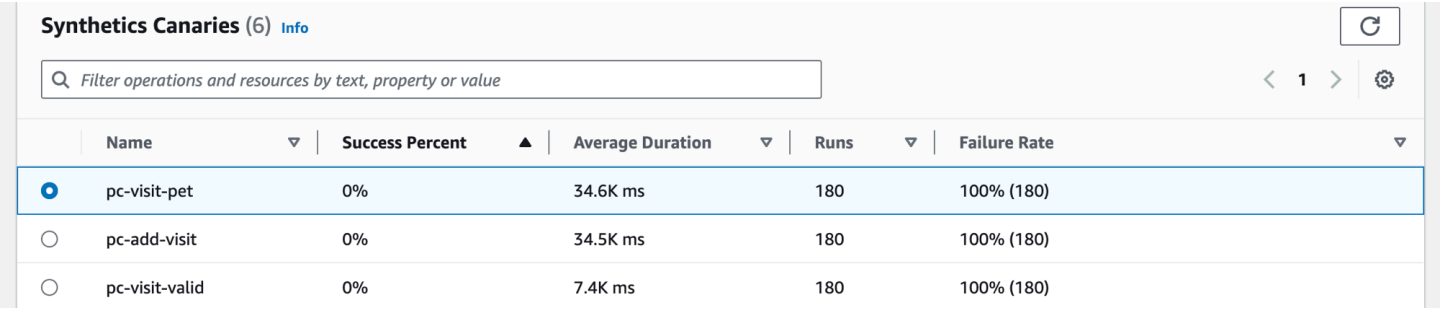
종속성을 선택하려면 종속성 테이블에서 종속성 옆에 있는 옵션을 선택합니다. 호출 볼륨, 가용성, 결함 및 오류에 대한 세부 지표를 표시하는 그래프 세트가 표시됩니다. 그래프의 한 지점을 가리키면 자세한 정보가 포함된 팝업이 표시됩니다. 그래프에서 한 지점을 선택하면 그래프에서 선택한 지점에 대한 상관관계가 있는 트레이스를 보여주는 진단 창이 열립니다. 상관관계가 있는 트레이스 테이블에서 트레이스 ID를 선택하면 선택한 트레이스의 X-Ray 트레이스 세부 정보 페이지가 열립니다.



Synthetics canary 보기

Synthetics Canary 탭을 선택하여 Synthetics Canary 테이블과 테이블의 각 canary에 대한 지표 세트를 표시합니다. 테이블에는 성공률, 평균 기간, 실행 횟수 및 실패율에 대한 지표가 포함되어 있습니다. AWS X-Ray 추적이 활성화된 canary만 표시됩니다.

Synthetics canary 테이블의 필터 텍스트 상자를 사용하여 관심 있는 canary를 찾습니다. 생성한 각 필터는 필터 텍스트 상자 아래에 표시됩니다. 언제든지 필터 지우기를 선택하여 테이블 필터를 제거할 수 있습니다.



canary 이름 옆에 있는 라디오 버튼을 선택하여 성공률, 오류, 기간 등의 세부 지표가 그래프로 표시된 탭 세트를 확인합니다. 그래프의 한 지점을 가리키면 자세한 정보가 포함된 팝업이 표시됩니다. 그래프에서 하나의 포인트를 선택하여 선택한 포인트와 상관된 canary 실행을 보여주는 진단 창을 엽니다. canary 실행을 선택하고 실행 런타임을 선택하여 로그, HTTP 아카이브(HAR) 파일, 스크린샷, 문제 해결에 도움이 되는 제안 단계 등 선택한 canary 실행에 대한 아티팩트를 확인합니다. 자세히 알아보기를 선택하여 Canary 실행 옆에 있는 CloudWatch Synthetics Canaries 페이지를 엽니다.

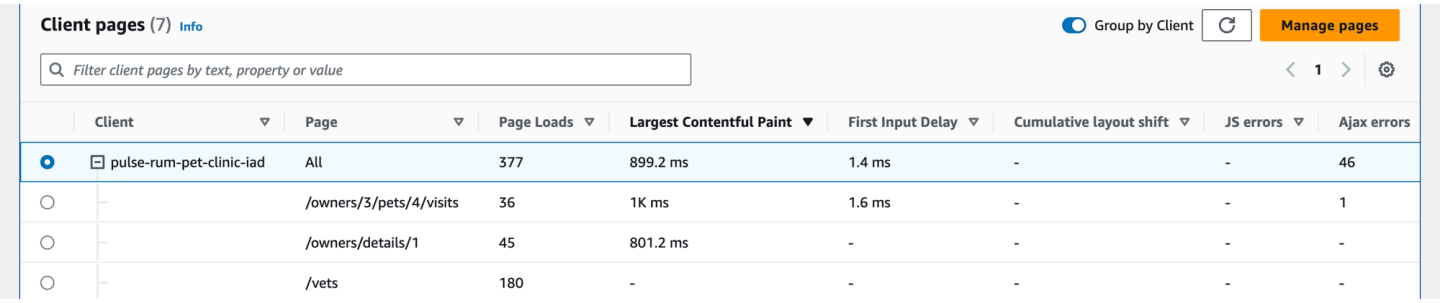
클라이언트 페이지 보기

클라이언트 페이지 탭을 선택하여 서비스를 직접 호출하는 클라이언트 웹 페이지 목록을 표시합니다. 선택한 클라이언트 페이지의 지표 세트를 사용하여 서비스 또는 애플리케이션과 상호 작용할 때 클라이언트의 경험 품질을 측정합니다. 이러한 지표로는 페이지 로드, 웹 바이탈, 오류가 포함됩니다.

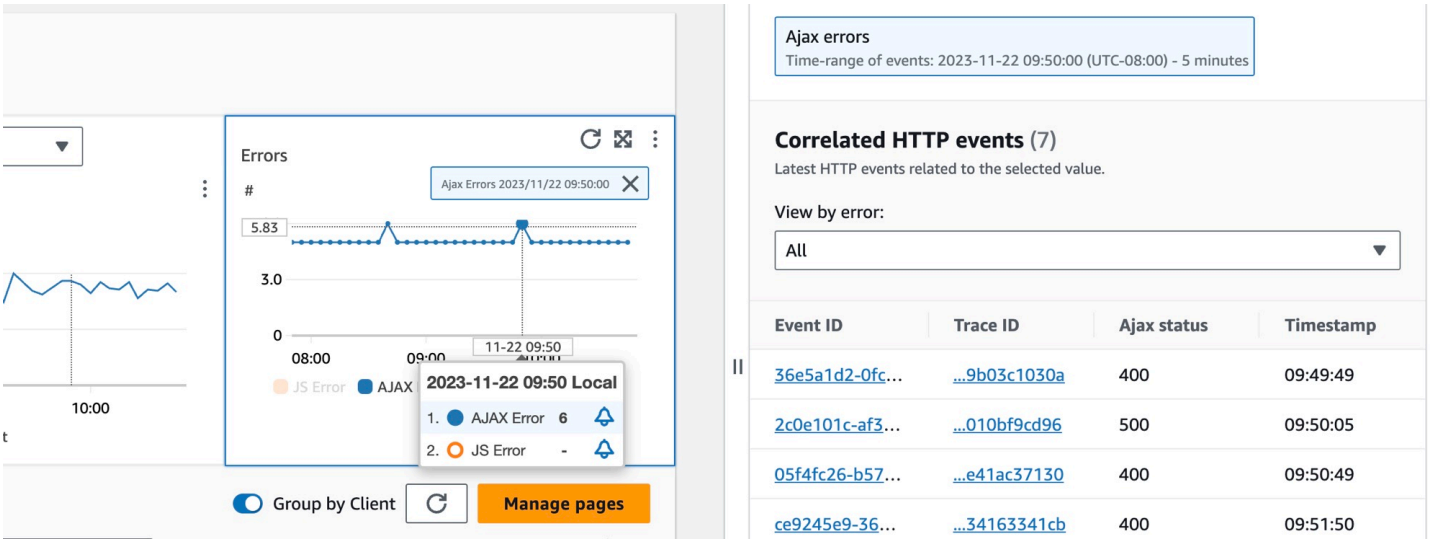
테이블에 클라이언트 페이지를 표시하려면 X-Ray 추적을 위해 CloudWatch RUM 웹 클라이언트를 구성하고 클라이언트 페이지에 대한 Application Signals 지표를 켜야 합니다. 페이지 관리를 선택하여 Application Signals 지표에 대해 활성화된 페이지를 선택합니다.

필터 텍스트 상자를 사용하여 필터 텍스트 상자 아래에서 관심 있는 클라이언트 페이지 또는 애플리케이션 모니터를 찾습니다. 필터 지우기를 선택하여 테이블 필터를 제거합니다. 클라이언트별로 클라이

언트 페이지를 그룹화하려면 클라이언트별로 그룹화를 선택합니다. 그룹화된 경우 클라이언트 이름 옆에 있는 + 아이콘을 선택하여 행을 확장하면 해당 클라이언트의 모든 페이지가 표시됩니다.



클라이언트 페이지를 선택하려면 클라이언트 페이지 테이블에서 클라이언트 페이지 옆에 있는 옵션을 선택합니다. 자세한 지표를 표시하는 그래프 세트가 표시됩니다. 그래프의 한 지점을 가리키면 자세한 정보가 포함된 팝업이 표시됩니다. 그래프에서 하나의 포인트를 선택하여 그래프에서 선택한 포인트와 상관된 성능 탐색 이벤트를 보여주는 진단 창을 엽니다. 선택한 이벤트에 대한 CloudWatch RUM 페이지 뷰를 열려면 탐색 이벤트 목록에서 이벤트 ID를 선택합니다.



> **Note**
>
> 클라이언트 페이지 내에서 AJAX 오류를 보려면 CloudWatch RUM 웹 클라이언트 버전 1.15 이상을 사용합니다.
>
> 현재 서비스당 최대 100개의 작업, canary 및 클라이언트 페이지와 최대 250개의 종속성을 표시할 수 있습니다.

CloudWatch 서비스 맵을 사용하여 애플리케이션 토폴로지 보기 및 작업 상태 모니터링

⚠ Application Signals는 Amazon CloudWatch의 평가판 릴리스에 있으며 변경될 수 있습니다.

Note

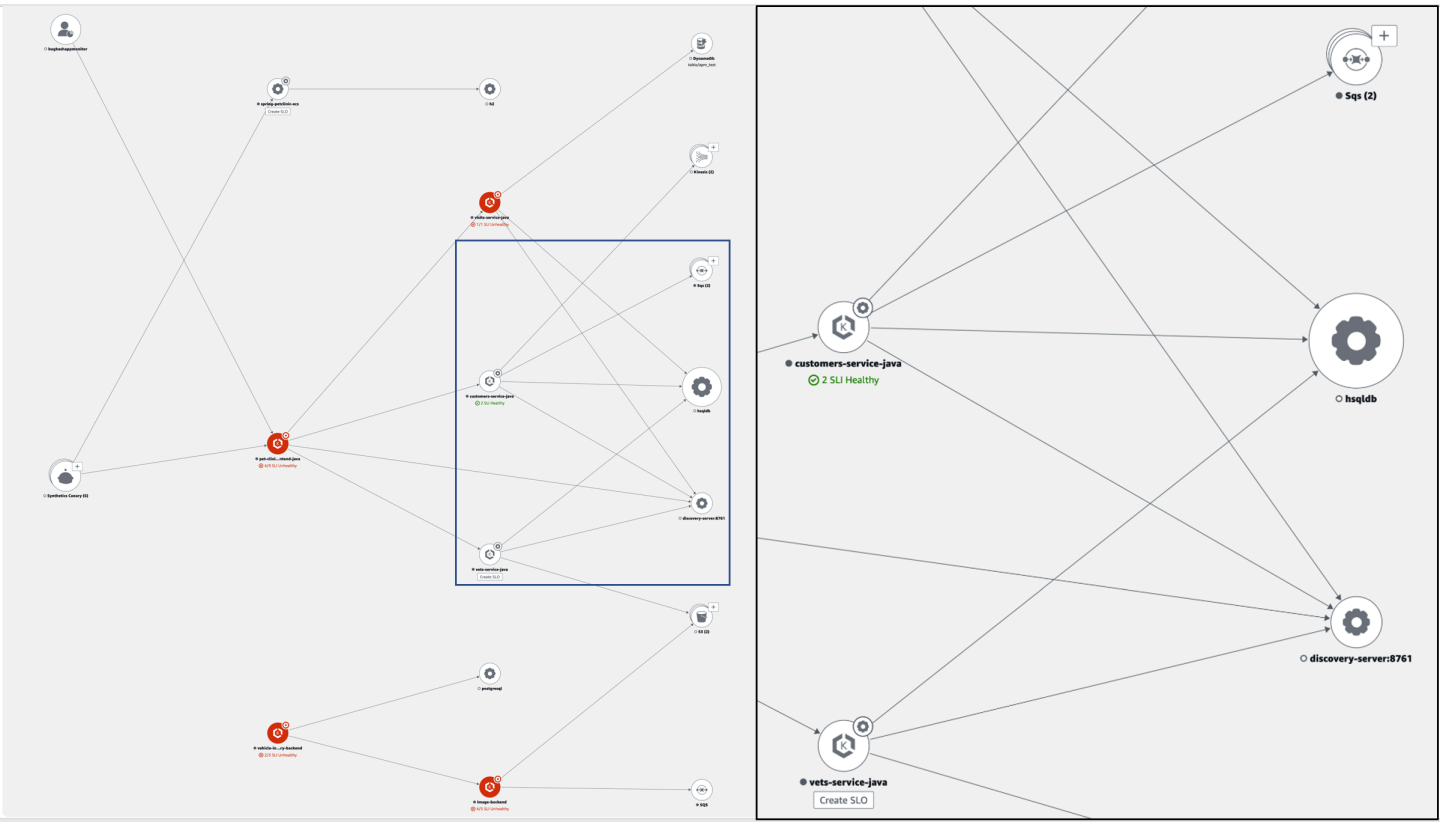
CloudWatch 서비스 맵은 ServiceLens 맵을 대체합니다. AWS X-Ray 트레이스를 기반으로 애플리케이션의 맵을 보려면 X-Ray 트레이스 맵을 엽니다. CloudWatch 콘솔의 왼쪽 탐색 창의 X-Ray 섹션에서 기록 맵을 선택합니다.

서비스 맵을 사용하여 애플리케이션 클라이언트, Synthetics canary, 서비스 및 종속성의 토폴로지를 보고 작업 상태를 모니터링합니다. 서비스 맵을 보려면 CloudWatch 콘솔을 열고 왼쪽 탐색 창의 Application Signals 섹션에서 서비스 맵을 선택합니다.

Application Signals에 대해 애플리케이션을 활성화한 후 서비스 맵을 사용하면 애플리케이션의 작업 상태를 더 쉽게 모니터링할 수 있습니다.

- 클라이언트, canary, 서비스 및 종속성 노드 간의 연결을 보면 애플리케이션 토폴로지와 실행 흐름을 이해하는 데 도움이 됩니다. 이는 서비스 운영자가 개발 팀이 아닌 경우 특히 유용합니다.
- 서비스 수준 목표(SLO)를 충족하거나 충족하지 못하는 서비스를 확인하세요. 서비스가 SLO를 충족하지 못하는 경우 다운스트림 서비스 또는 종속성이 문제의 원인인지 또는 여러 업스트림 서비스에 영향을 미치는지 빠르게 식별할 수 있습니다.
- 개별 클라이언트, Synthetics canary, 서비스 또는 종속성 노드를 선택하여 관련 지표를 확인합니다. 서비스 세부 정보 페이지에서는 작업, 종속성, Synthetics canary 및 클라이언트 페이지에 대한 자세한 정보를 표시합니다.
- 서비스 맵을 필터링하고 확대 또는 축소하여 더 쉽게 애플리케이션 토폴로지의 일부에 초점을 맞추거나 전체 맵을 볼 수 있습니다. 필터 텍스트 상자에서 속성을 하나 이상 선택하여 필터를 생성합니다. 각 속성을 선택하면 필터 기준이 안내됩니다. 필터 텍스트 상자 아래에 전체 필터가 표시됩니다. 언제든지 필터 지우기를 선택하여 필터를 제거할 수 있습니다.

다음 서비스 맵 예제에서는 상호 작용하는 구성 요소에 연결하는 엣지가 있는 서비스를 보여줍니다. SLO가 정의된 경우 서비스 맵에서는 상태도 표시합니다.

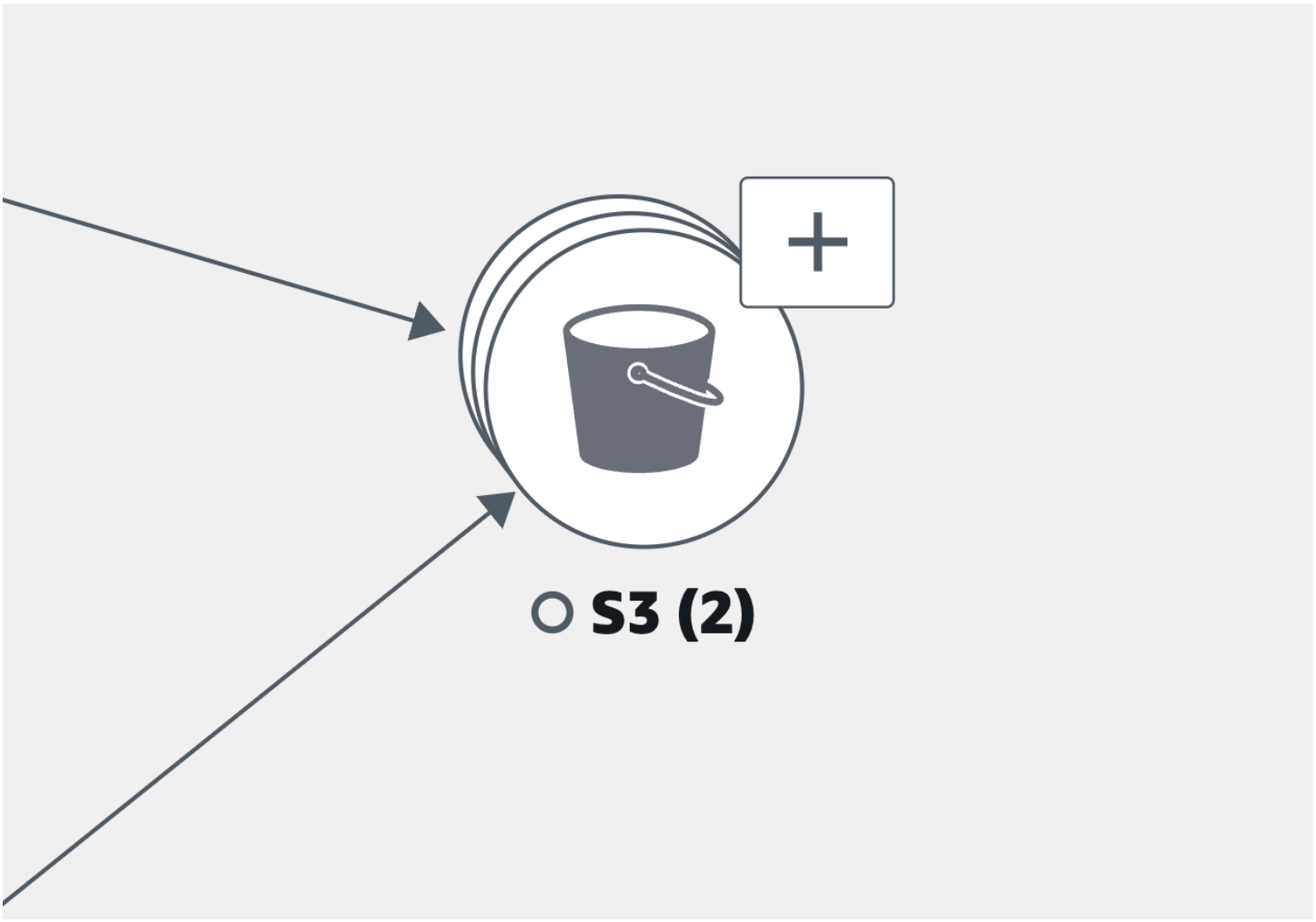


서비스 맵 탐색

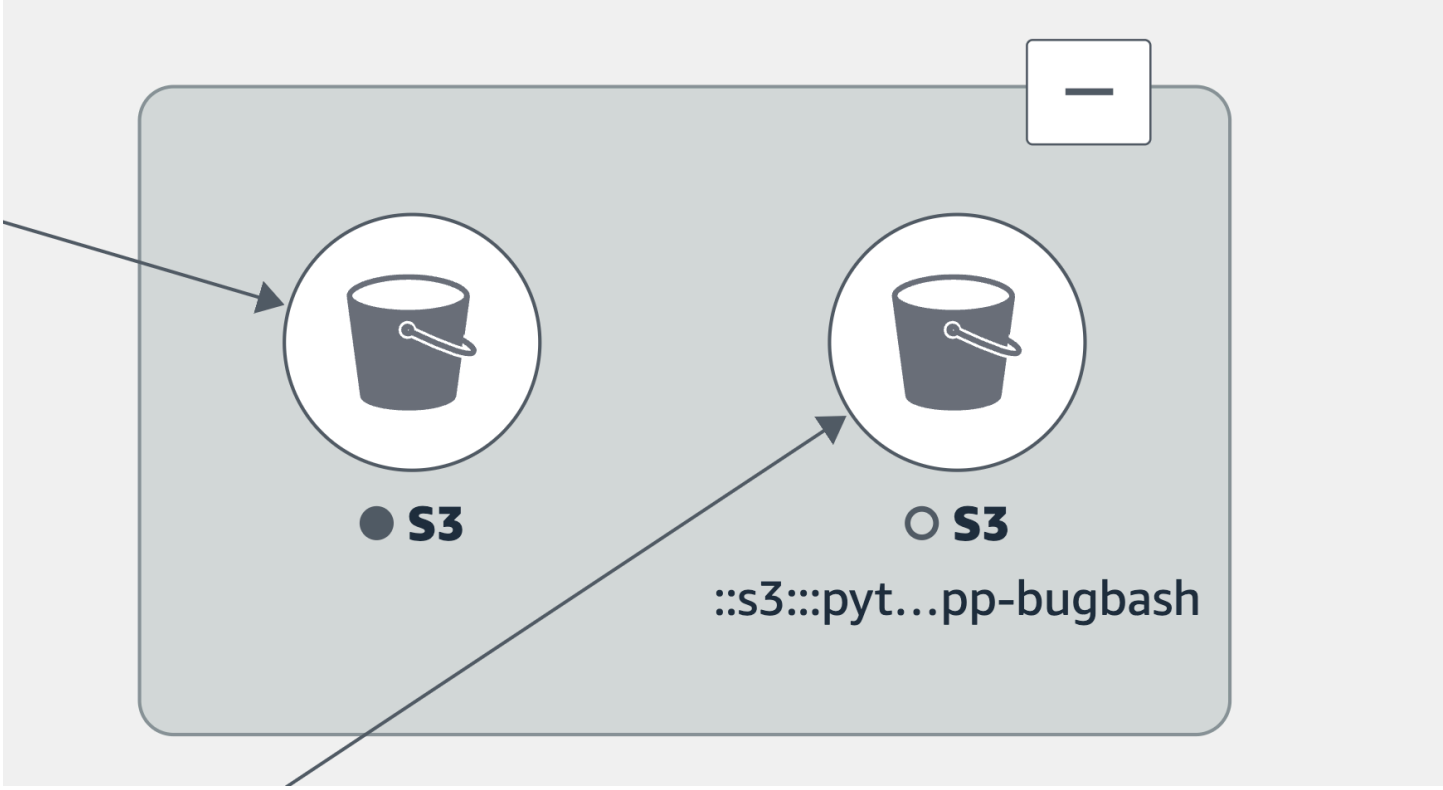
Application Signals에 대한 애플리케이션을 활성화하면 서비스 맵에서 서비스 및 종속성을 나타내는 노드를 표시합니다.

CloudWatch RUM 클라이언트 및 Synthetics canary에 대한 활성 추적 기능을 켜고 맵에서 클라이언트 및 canary 노드를 확인합니다.

기본적으로 같은 종류의 canary, RUM 클라이언트 및 AWS 서비스 종속성은 서비스 맵에서 확장 가능한 단일 아이콘으로 그룹화됩니다. AWS 외부의 서비스 종속성은 기본적으로 함께 그룹화되지 않습니다. 예를 들어 다음 이미지에서 모든 Amazon S3 버킷은 확장 가능한 하나의 아이콘으로 그룹화됩니다.



이전 이미지에서 Amazon S3 그룹화와 원본 서비스 사이의 레이블에는 종속성 아이콘 아래 괄호 안에 그룹에 대한 엣지 수가 표시됩니다. 다음 이미지와 같이 (+) 아이콘을 선택하여 그룹을 확장하고 개별 요소를 확인합니다.

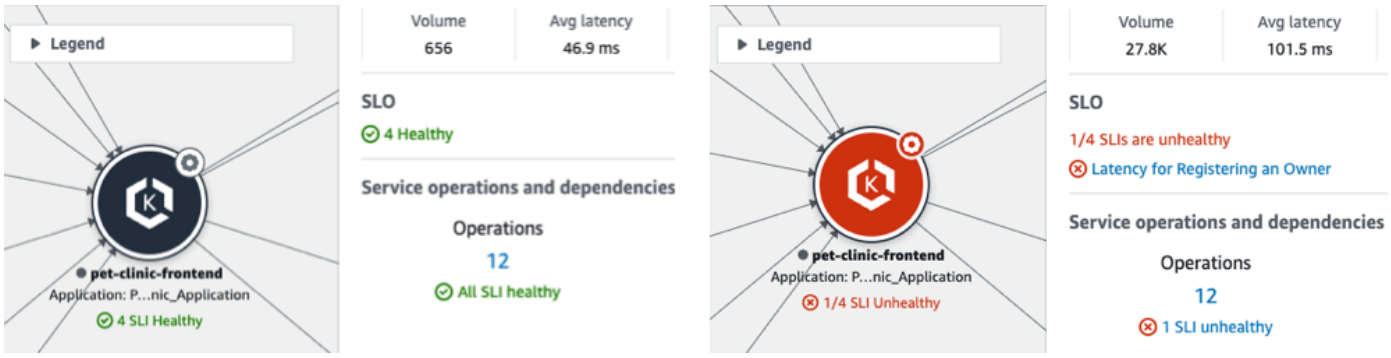


각 노드 종류와 노드 간 엣지(연결)을 탐색하는 방법에 대한 자세한 내용을 보려면 탭을 선택합니다.

View your application services

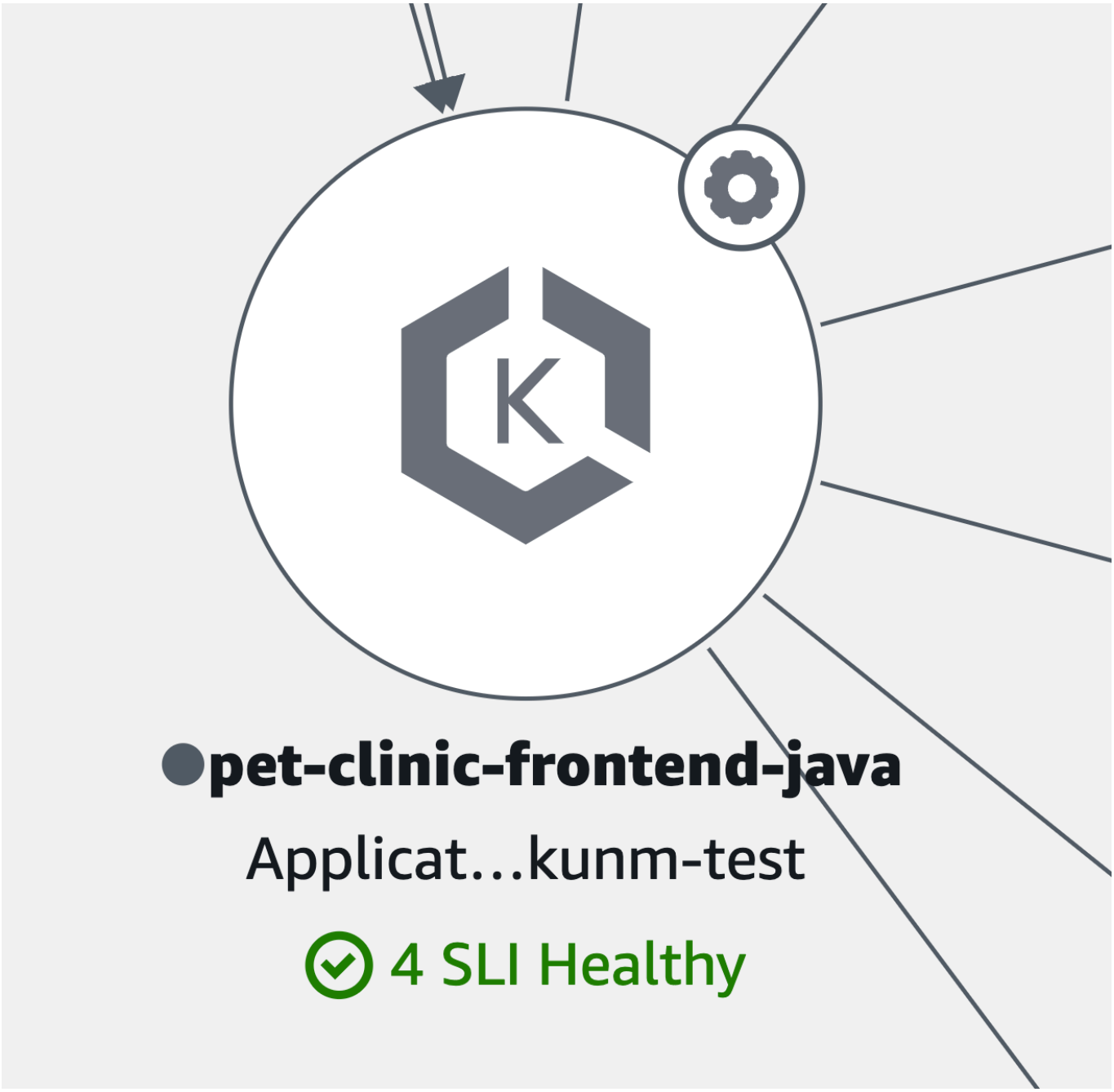
서비스 맵에서 애플리케이션 서비스와 해당 SLO 및 서비스 수준 지표(SLI)의 상태를 볼 수 있습니다. 서비스에 대한 SLO를 생성하지 않은 경우 서비스 노드 아래 SLO 생성 버튼을 선택합니다.

서비스 맵에 모든 서비스가 표시됩니다. 또한 다음 이미지와 같이 서비스를 사용하는 고객 및 canary와 서비스에서 호출하는 종속성도 보여줍니다.



다음 아이콘은 서비스 맵에 있는 애플리케이션 서비스 예를 나타냅니다.

- Amazon Elastic Kubernetes Service:



- Kubernetes 컨테이너:

- Amazon Elastic Compute Cloud(Amazon EC2):

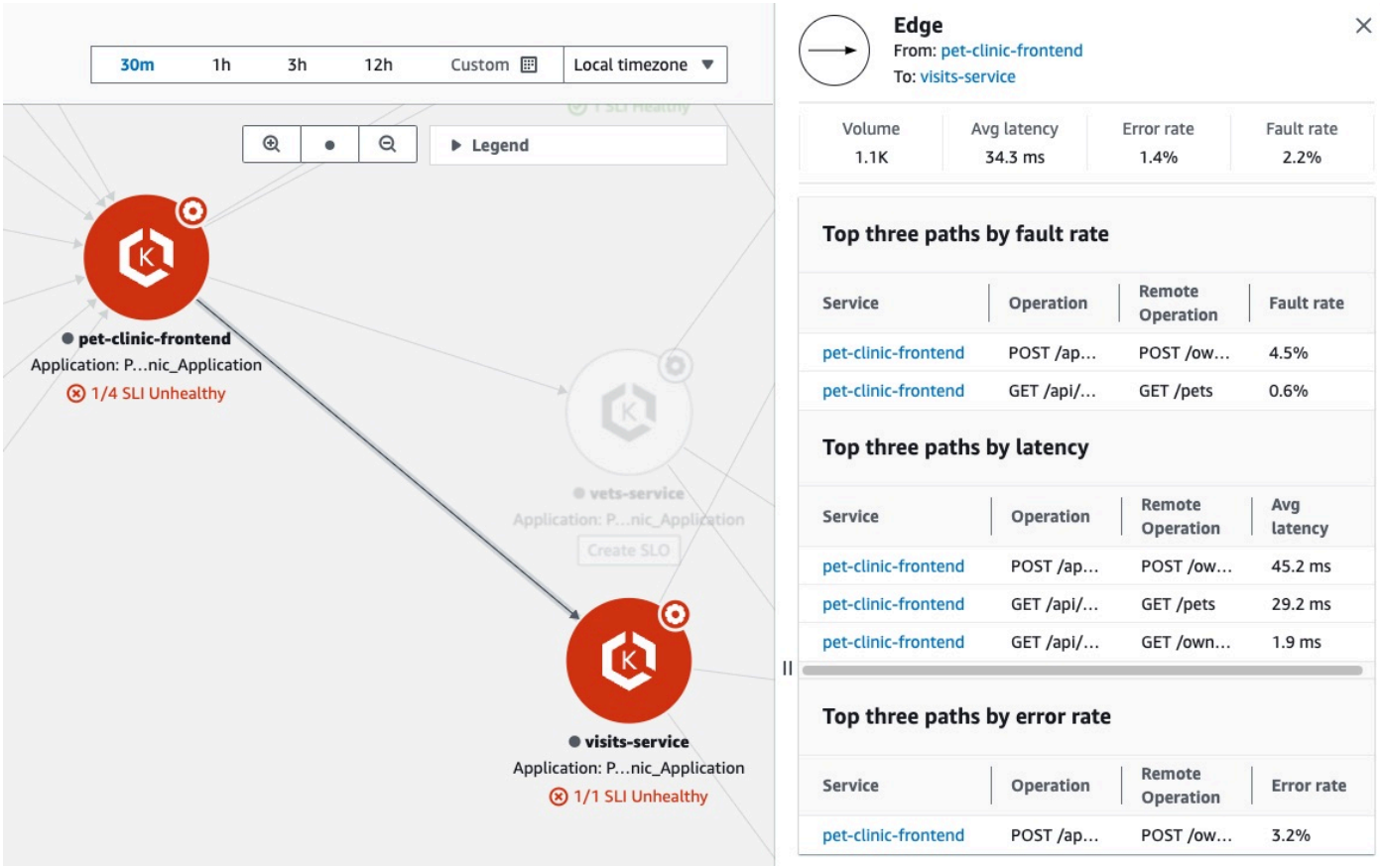
- 이전에 나열되지 않은 기타 애플리케이션 서비스 유형:

서비스 노드를 선택하면 다음과 같은 자세한 서비스 정보가 표시된 창이 열립니다.

- 호출 볼륨, 지연 시간, 오류 및 장애 발생률에 대한 지표
- 상태가 `healthy` 또는 `unhealthy`인 SLI 및 SLO의 수.
- SLO에 대한 자세한 내용을 확인할 수 있는 옵션.
- 서비스 작업, 종속성, Synthetics canary 및 클라이언트 페이지 수.
- 각 번호를 선택하여 해당 번호의 서비스 세부 정보 페이지를 여는 옵션.
- 애플리케이션 이름(AWS Management Console 홈페이지의 AppRegistry 또는 Applications 카드를 사용하여 기본 컴퓨팅 리소스를 애플리케이션과 연결한 경우).
  - MyApplications 콘솔 페이지에 애플리케이션 세부 정보를 표시하려면 애플리케이션 이름을 선택합니다.

- Amazon EKS에 호스팅되는 서비스의 경우 `Cluster`, `Namespace`, `Workload` 또는 Amazon ECS나 Amazon EC2에 호스팅되는 서비스의 경우 `Environment`. Amazon EKS 호스팅 서비스의 경우 링크를 선택하여 CloudWatch Container Insights를 열 수 있습니다.

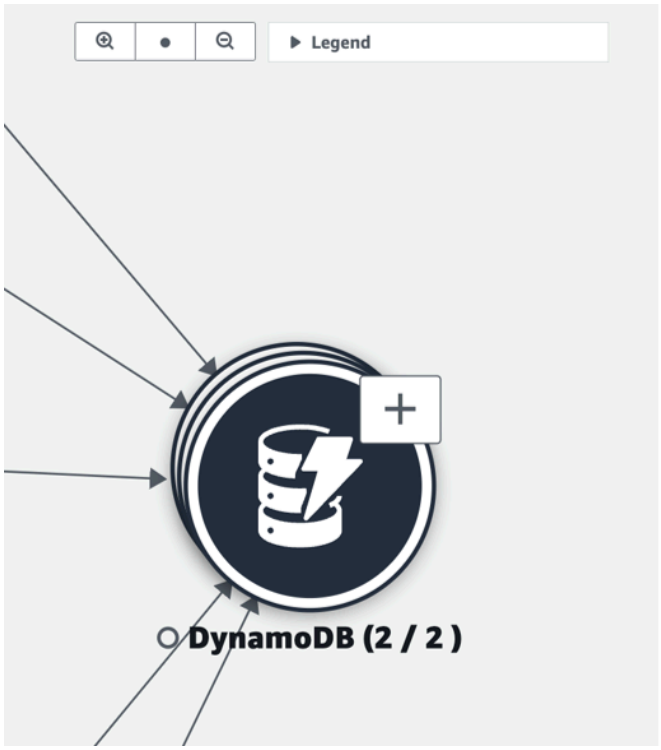
서비스 노드와 다운스트림 서비스 또는 종속성 노드 간의 엣지나 연결을 선택합니다. 그러면 이미지 예제와 같이 장애 발생률, 지연 시간 및 오류율을 기준으로 상위 경로를 포함하는 창이 열립니다. 창에서 있는 아무 링크나 선택하여 서비스 세부 정보 페이지를 열고 선택한 서비스 또는 종속성에 대한 세부 정보를 확인합니다.



View dependencies

애플리케이션 종속성은 직접 호출하는 서비스에 연결된 서비스 맵에 표시됩니다.

종속성 노드를 선택하면 장애 발생률, 지연 시간 및 오류율을 기준으로 상위 경로가 포함된 창을 엽니다. 아래 이미지 예제와 같이 서비스 또는 대상 링크를 선택하여 서비스 세부 정보 페이지를 열고 선택한 서비스 또는 종속성 대상에 대한 세부 정보를 확인합니다.



Volume	Avg latency	Error rate	Fault rate
-	-	-	-

Top three paths by fault rate

Service	Remote operation	Fault rate
No paths with faults		

Top three paths by latency

Service	Remote operation	Avg latency
billing-service-ec2-python	PutItem	282.8 ms
billing-service-python	PutItem	75.6 ms
visits-service-java	PutItem	64.9 ms

Top three paths by error rate

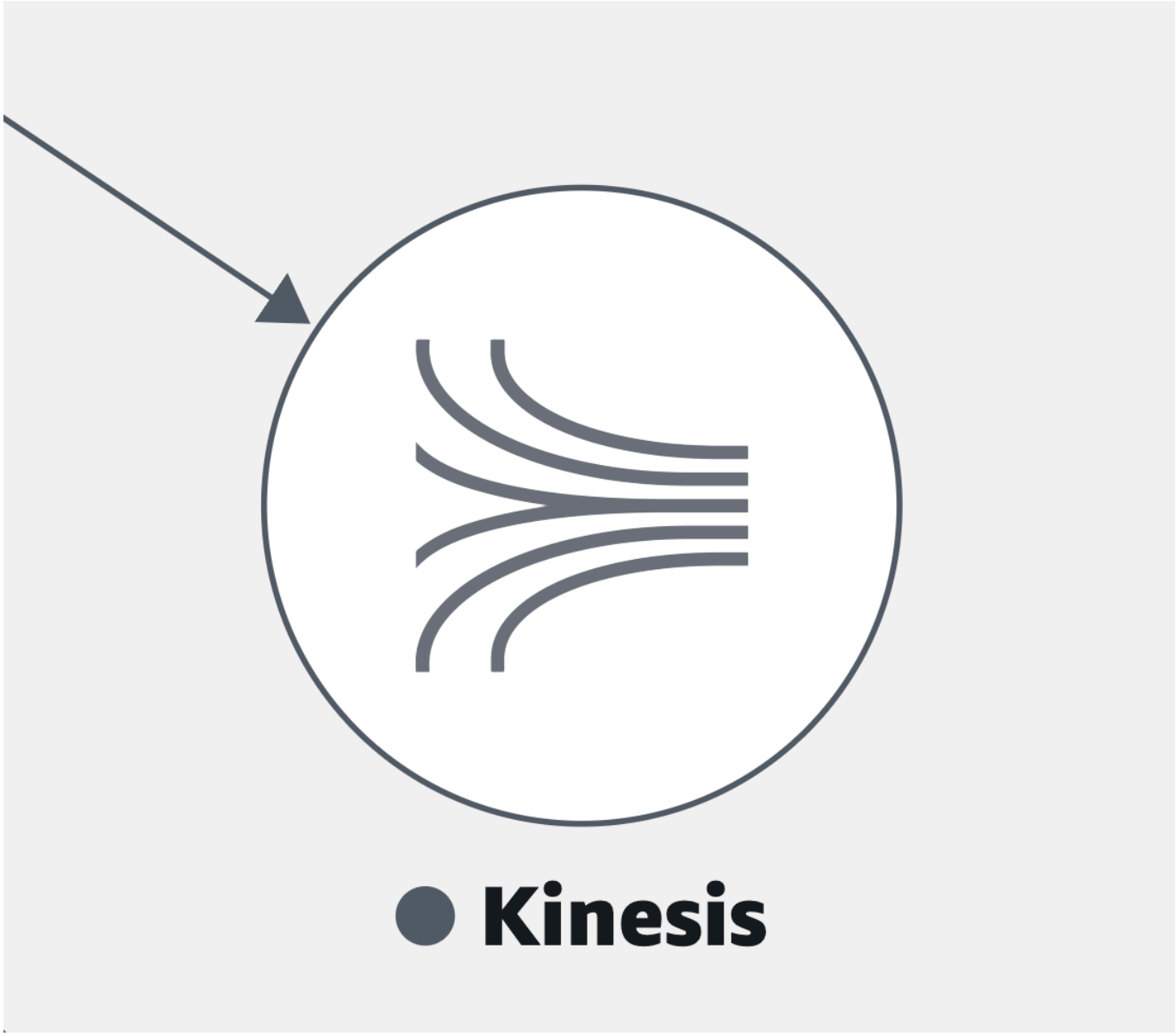
Service	Remote operation	Error rate
visits-service-java	PutItem	9.6%

서비스 종속성은 기본적으로 확장 가능한 단일 아이콘으로 그룹화됩니다. 이전 이미지와 같이 (+) 아이콘을 선택하여 그룹을 확장하고 개별 요소를 확인합니다.

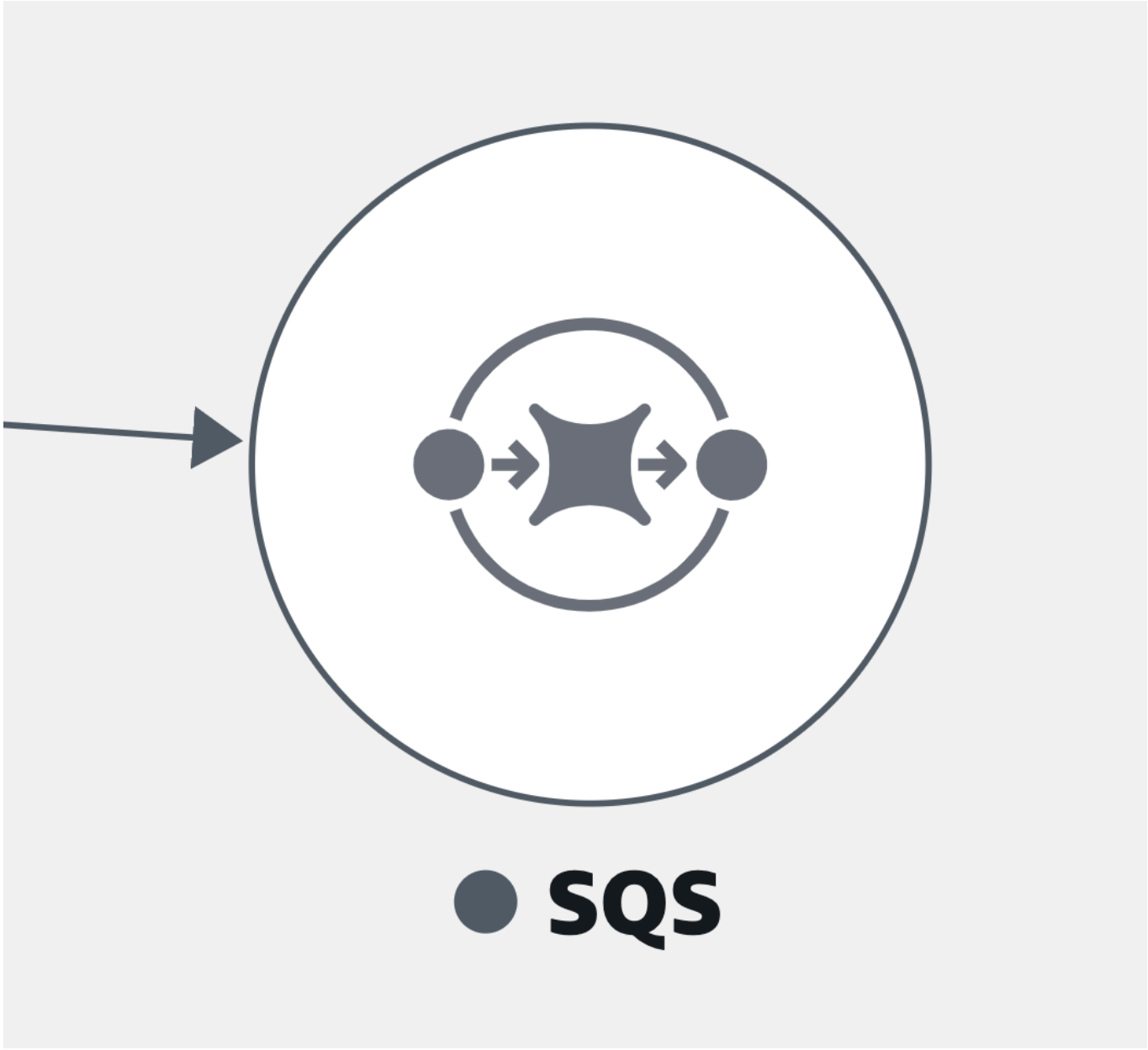
다음 아이콘은 서비스 맵에 있는 종속성 노드 예를 나타냅니다.

- Amazon S3 버킷:

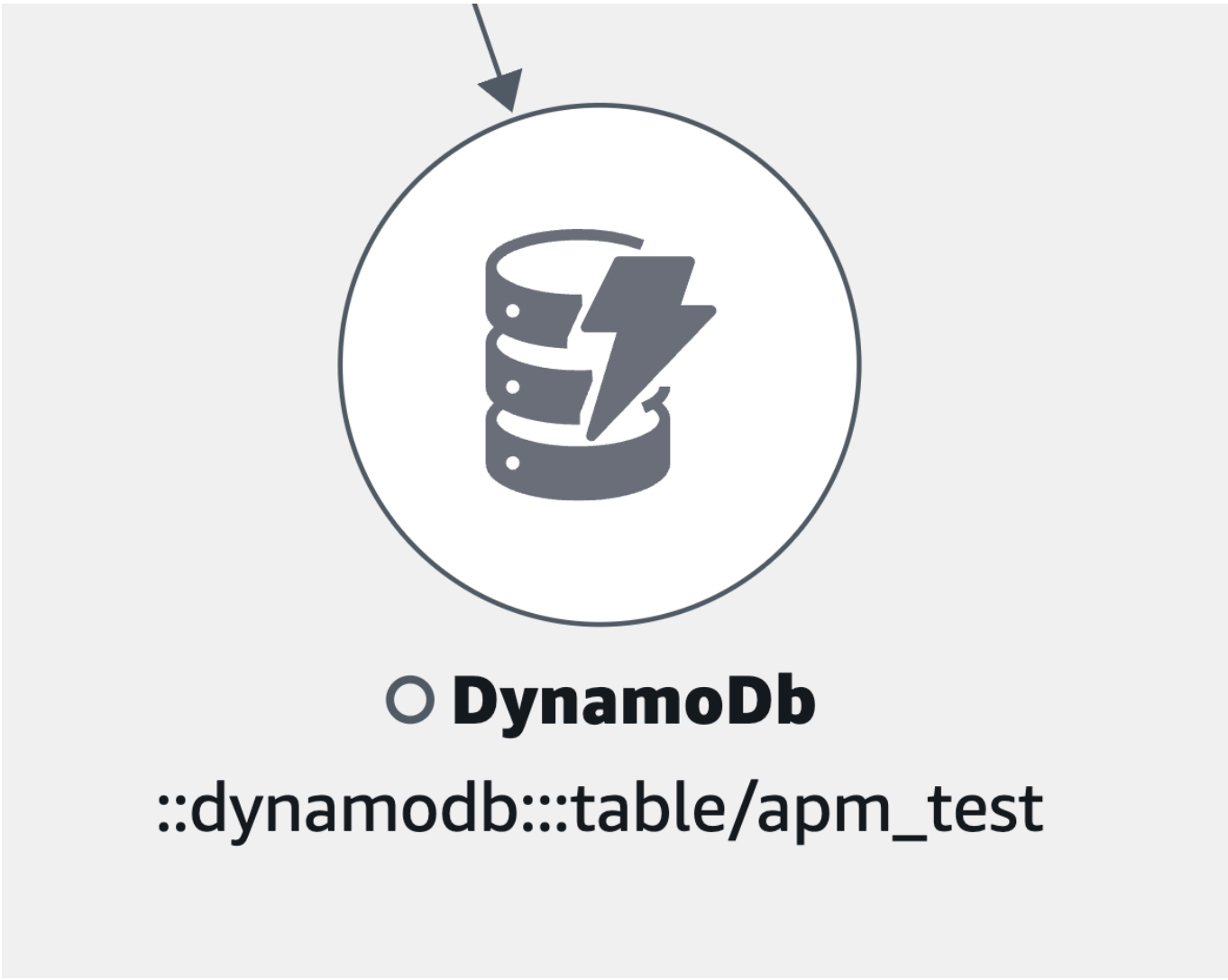
- Amazon Kinesis 스트림:



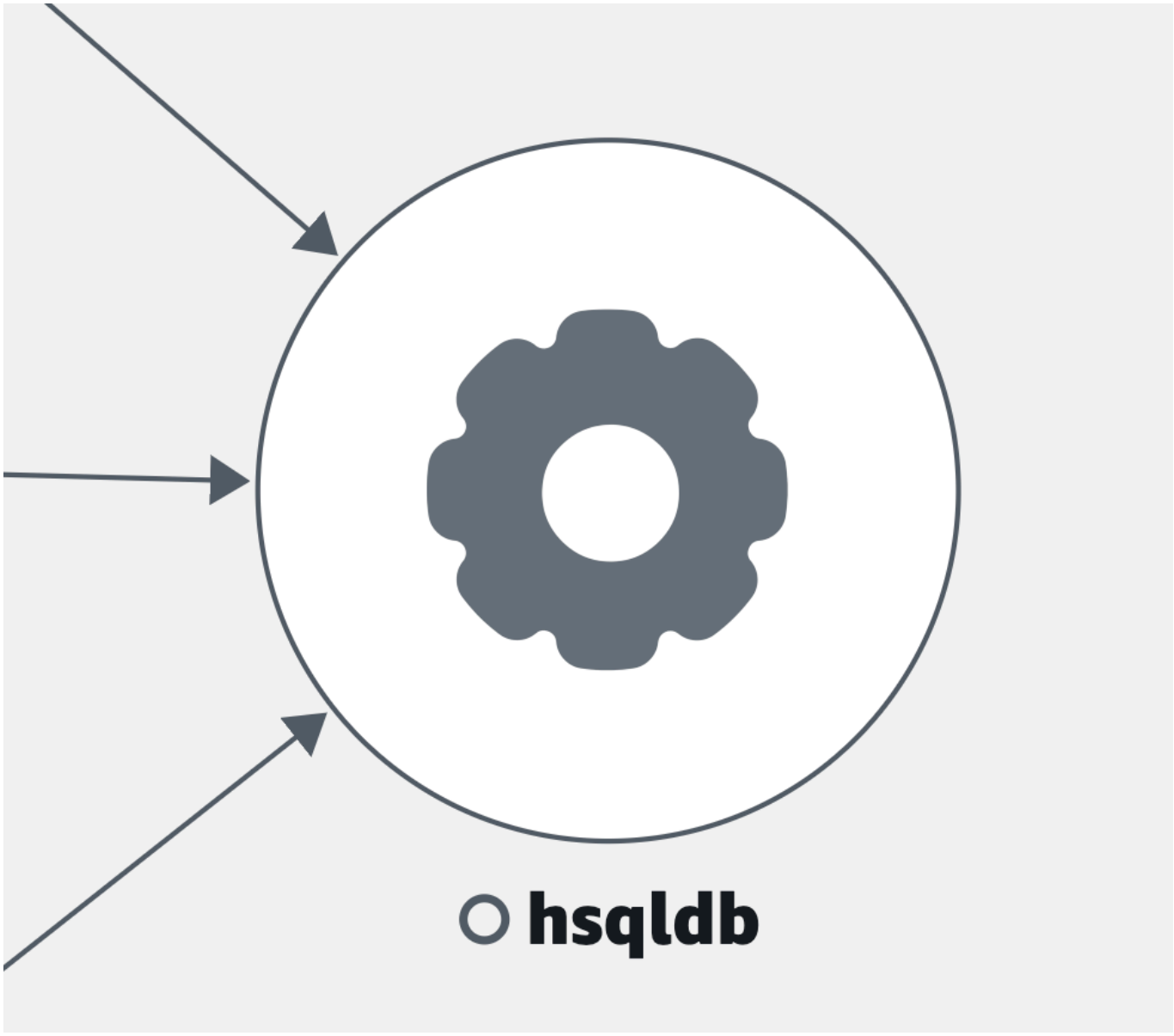
- Amazon Simple Queue Service(Amazon SQS):



- Amazon DynamoDB 테이블:



- 이전에 나열되지 않은 기타 종속성 유형:



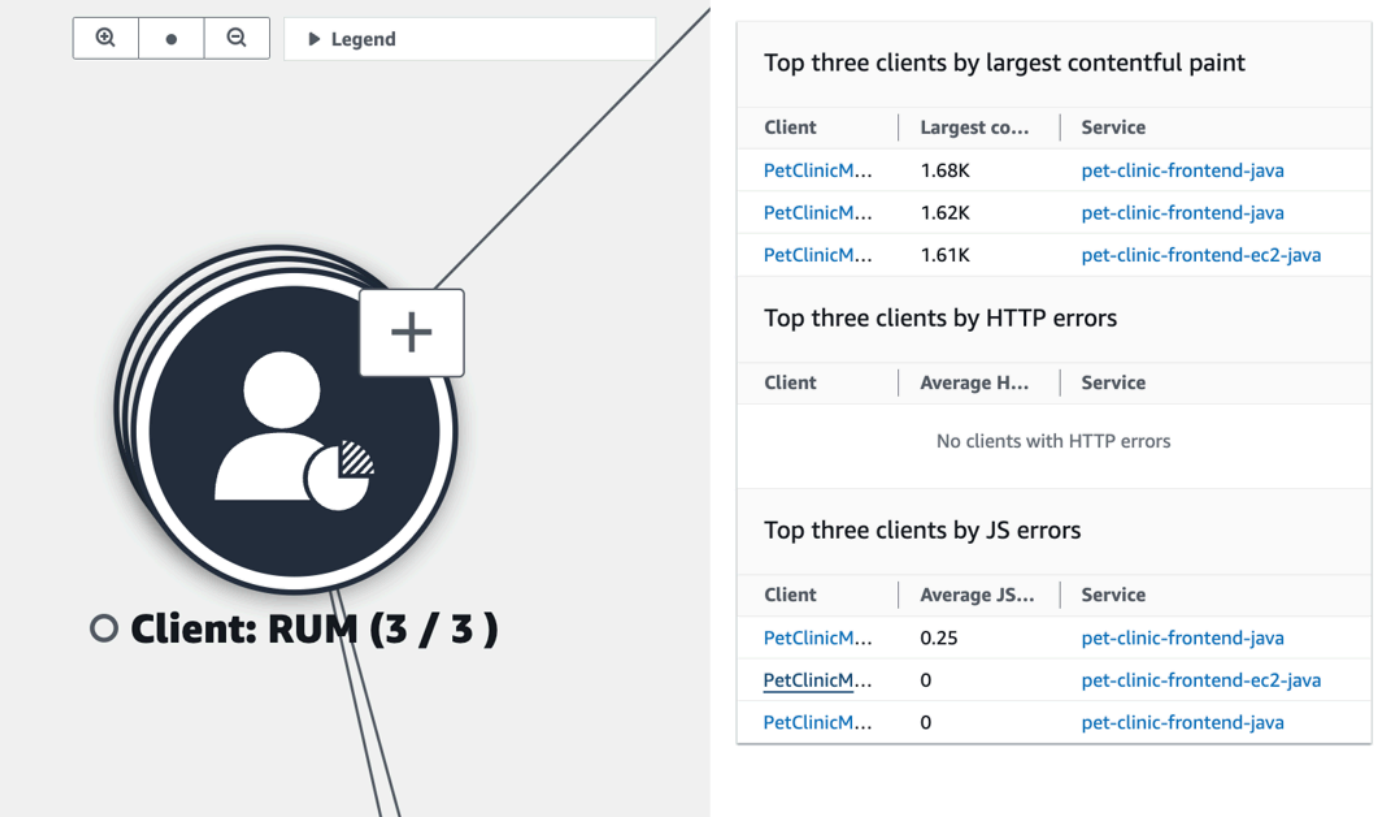
View clients

CloudWatch RUM 웹 클라이언트에 대한 X-Ray 추적 기능을 켠 후 해당 클라이언트가 직접 호출하는 서비스에 연결된 서비스 맵에 표시됩니다.

클라이언트 노드를 선택하여 자세한 클라이언트 정보를 표시하는 창을 엽니다.

- 페이지 로드, 평균 로드 시간, 오류, 평균 웹 바이탈에 대한 지표
- 오류 분류를 보여주는 그래프
- CloudWatch RUM에 클라이언트 세부 정보를 표시하는 링크

RUM 클라이언트는 기본적으로 확장 가능한 단일 아이콘으로 그룹화됩니다. 다음 이미지와 같이 (+) 아이콘을 선택하여 그룹을 확장하고 개별 요소를 확인합니다.



다음 아이콘은 서비스 맵에 있는 RUM 클라이언트 예를 나타냅니다.

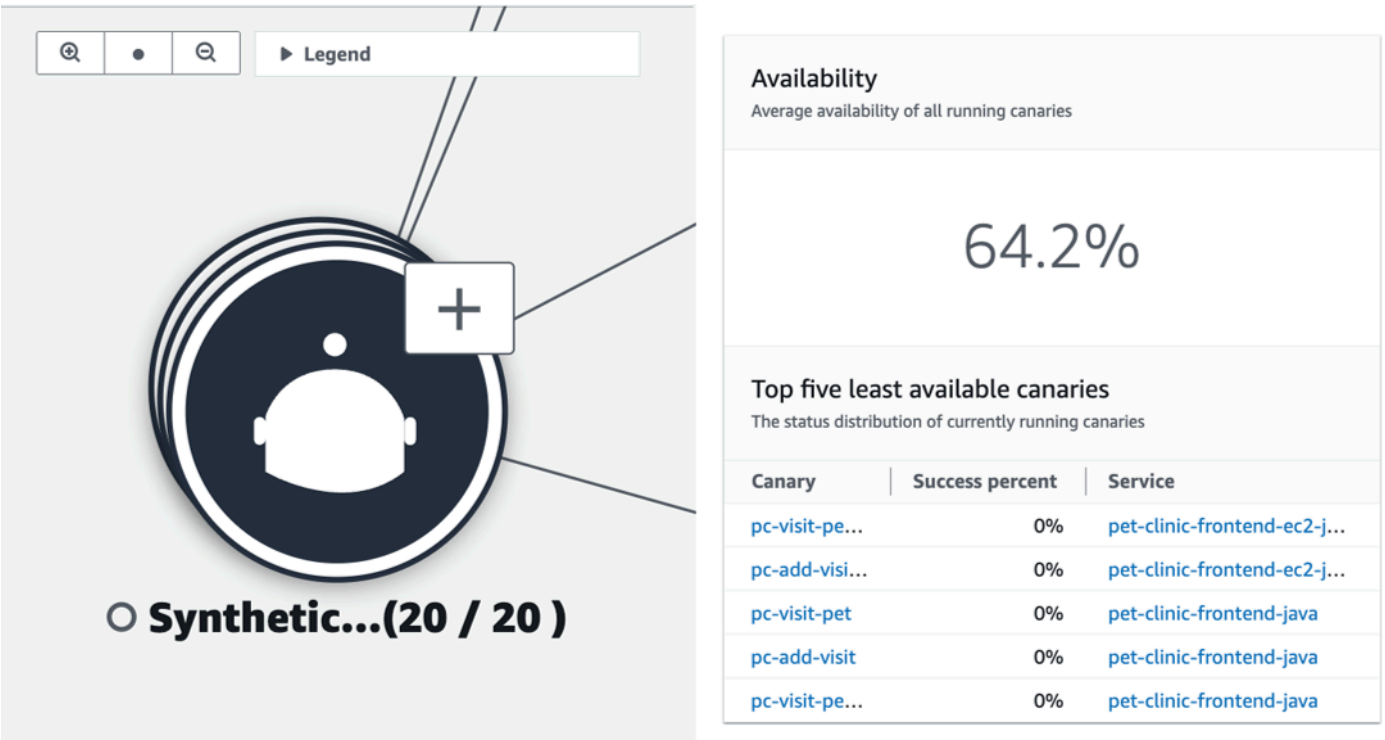
- RUM 클라이언트 -

> **Note**
>
> 클라이언트 페이지 내에서 AJAX 오류를 보려면 CloudWatch RUM 웹 클라이언트 버전 1.15 이상을 사용합니다.

View synthetics canaries

다음 이미지 예제와 같이 CloudWatch Synthetics canary에 대한 AWS X-Ray 추적 기능을 켠 후 직접 호출하는 서비스에 연결된 서비스 맵에 표시됩니다.

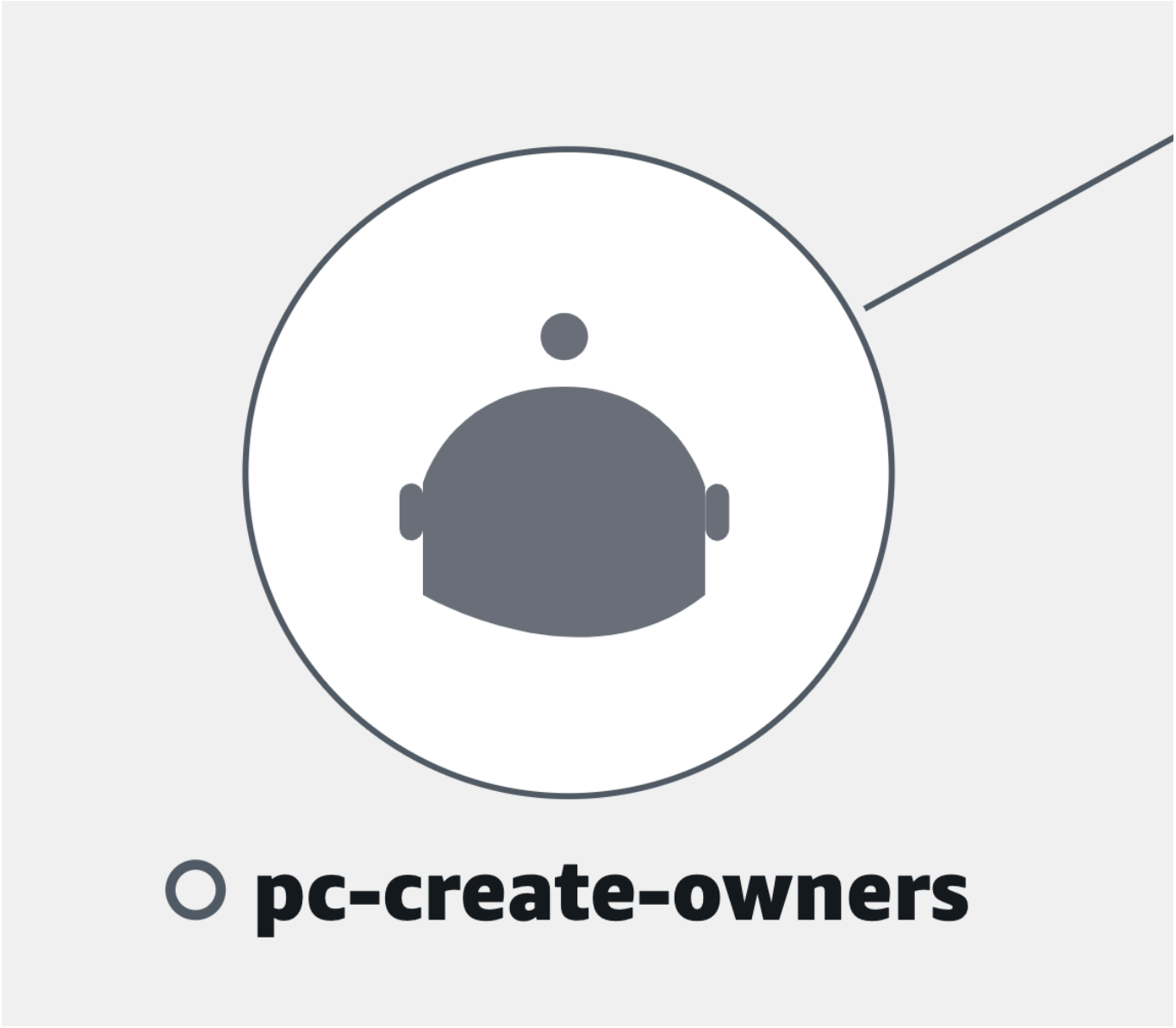
다음 이미지와 같이 canary 노드를 선택하여 자세한 canary 정보를 표시하는 창을 엽니다.

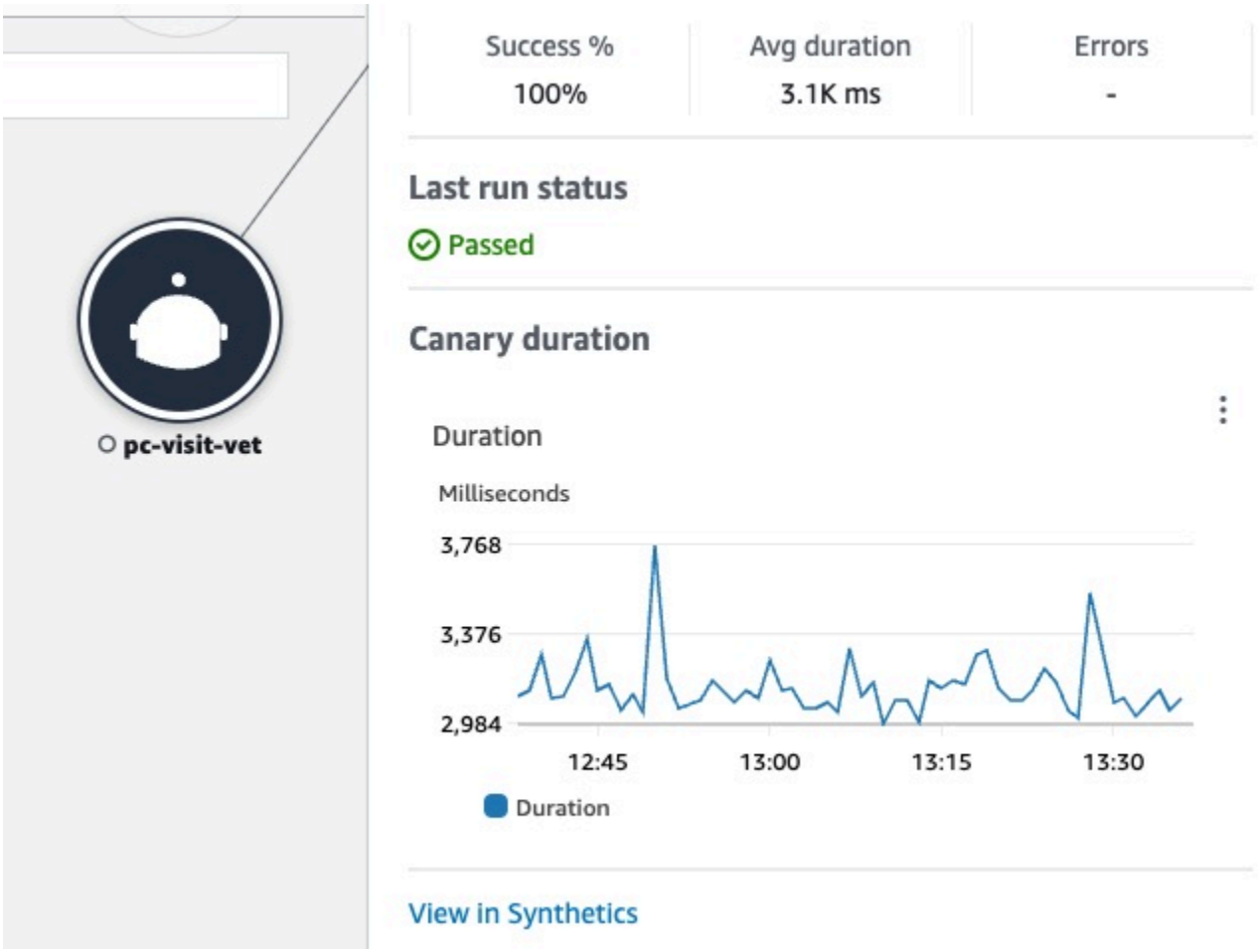


canary는 기본적으로 확장 가능한 단일 아이콘으로 그룹화됩니다. 이전 이미지와 같이 (+) 아이콘을 선택하여 그룹을 확장하고 개별 요소를 확인합니다.

다음 아이콘은 서비스 맵에 있는 클라이언트 예를 나타냅니다.

- Synthetics canary -


pc-create-owners



canary 노드 창에서 다음을 확인할 수 있습니다.

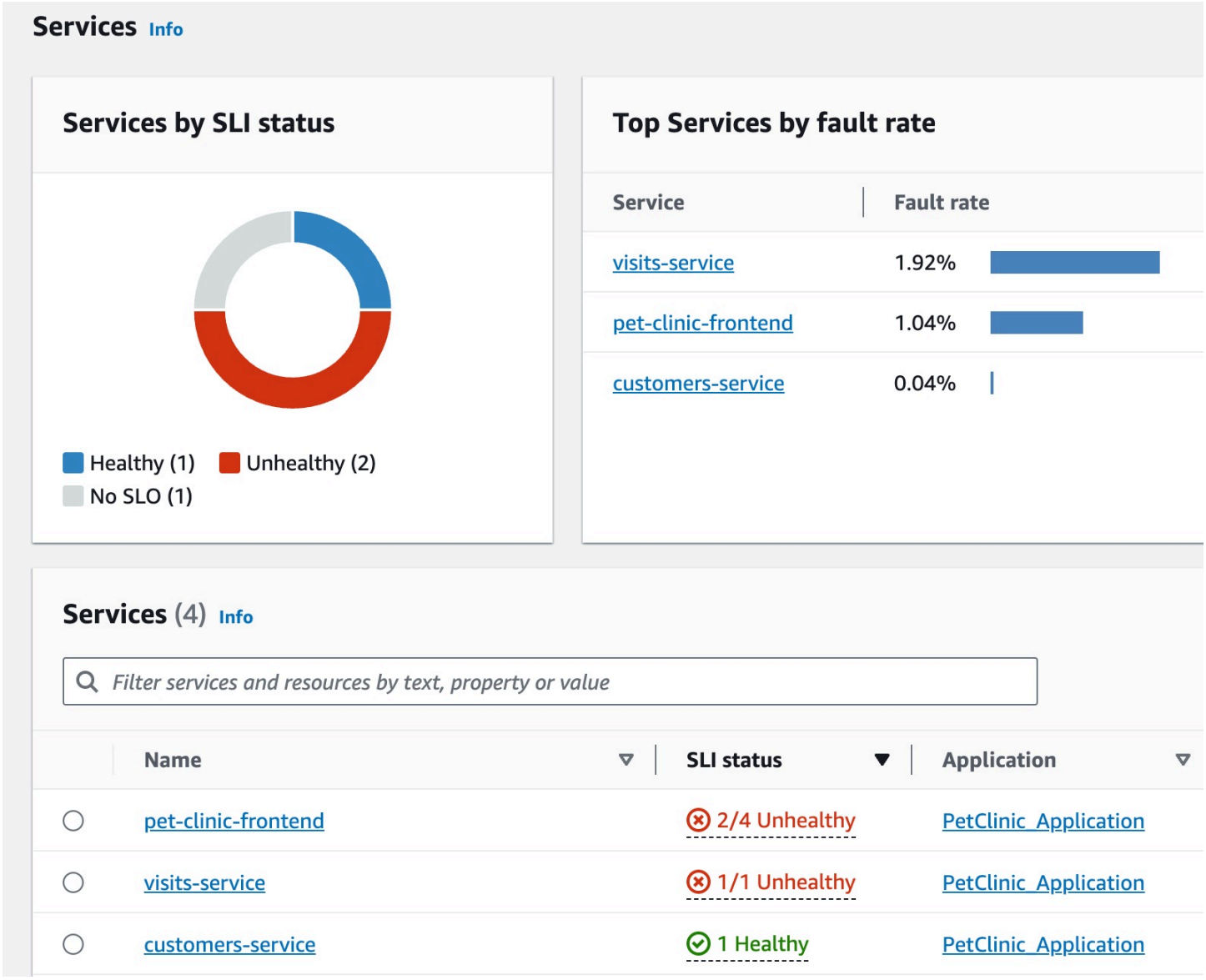
- 성공률, 평균 지속 시간, 오류에 대한 지표
- 마지막 canary 실행 상태
- canary 실행 지속 시간을 보여주는 그래프. 그래프 시리즈를 마우스로 가리키면 자세한 정보가 포함된 팝업이 표시됩니다.
- CloudWatch Synthetics에서 canary 세부 정보를 표시하는 링크.

예제: Application Signals를 사용하여 운영 상태 문제 해결

⚠ Application Signals는 Amazon CloudWatch의 평가판 릴리스에 있으며 변경될 수 있습니다.

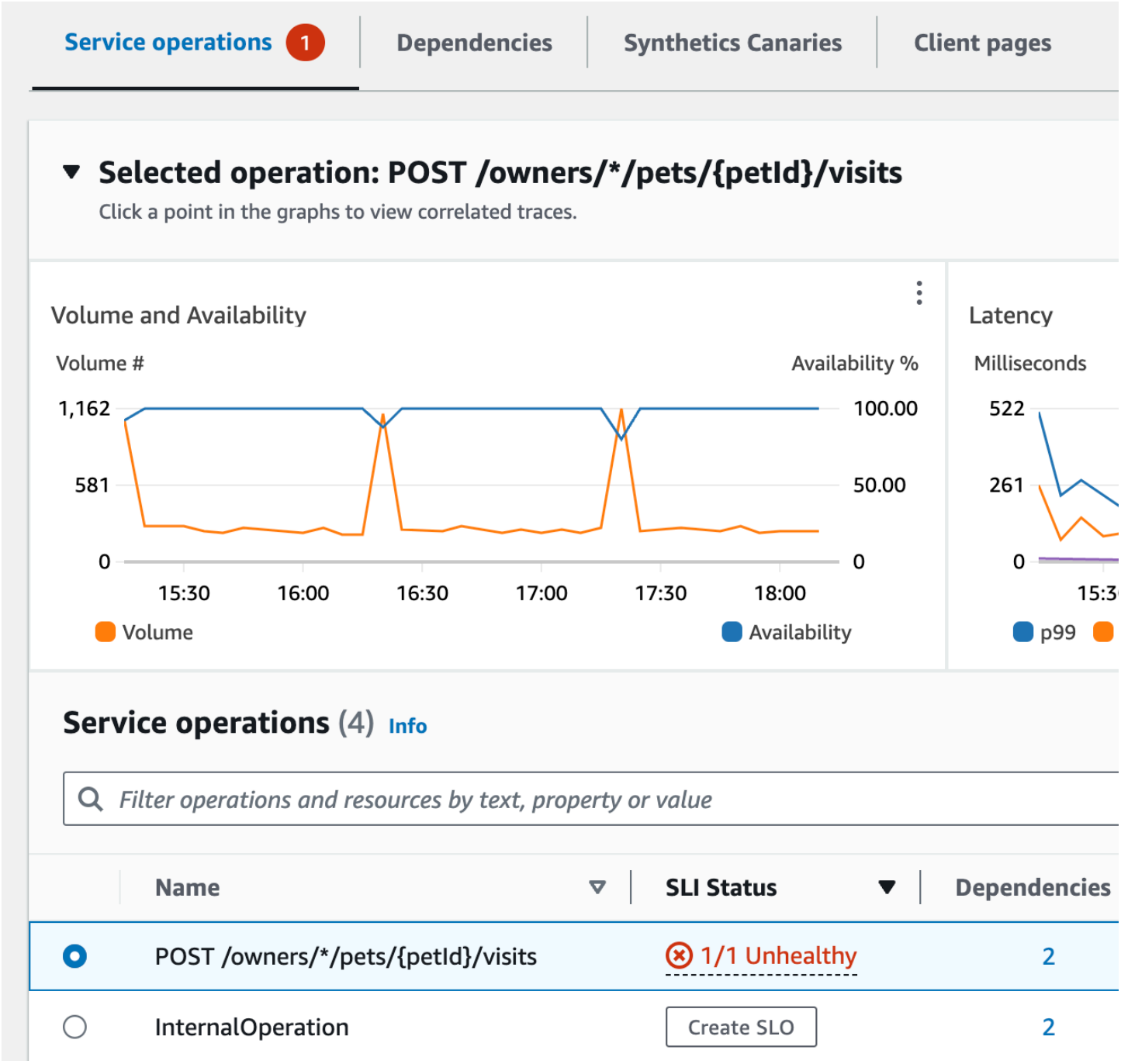
다음 시나리오는 Application Signals를 사용하여 서비스를 모니터링하고 서비스 품질 문제를 식별하는 방법에 대한 예제를 제공합니다. 드릴다운하여 잠재적 근본 원인을 파악하고 문제 해결을 위한 조치를 취합니다. 이 예제는 DynamoDB와 같은 AWS 서비스를 직접적으로 호출하는 여러 마이크로서비스로 구성된 반려동물 클리닉 애플리케이션에 초점을 맞추고 있습니다.

Jane은 반려동물 클리닉 애플리케이션의 운영 상태를 감독하는 DevOps 팀의 일원입니다. Jane의 팀은 애플리케이션의 가용성과 응답성을 보장하기 위해 노력하고 있습니다. 이들은 서비스 수준 목표(SLO)를 기준으로 이러한 비즈니스 약속과 비교하여 애플리케이션 성능을 측정합니다. Jane은 여러 비정상 서비스 수준 지표(SLI)에 대한 알림을 받습니다. CloudWatch 콘솔을 열고 서비스 페이지로 이동하면 비정상 상태인 여러 서비스가 표시됩니다.

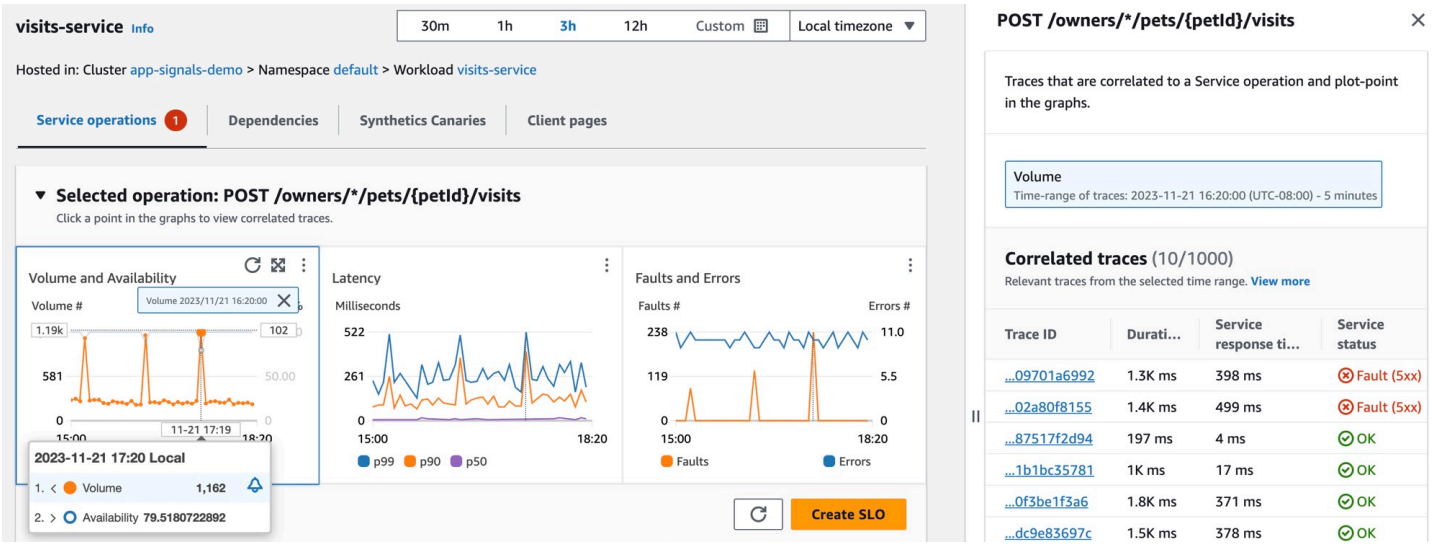


페이지 상단에서 Jane은 `visits-service`가 장애 발생률이 가장 높음을 확인합니다. 그래프에서 링크를 선택하면 해당 서비스의 서비스 세부 정보 페이지가 열립니다. 그녀는 서비스 작업 테이블에 비정

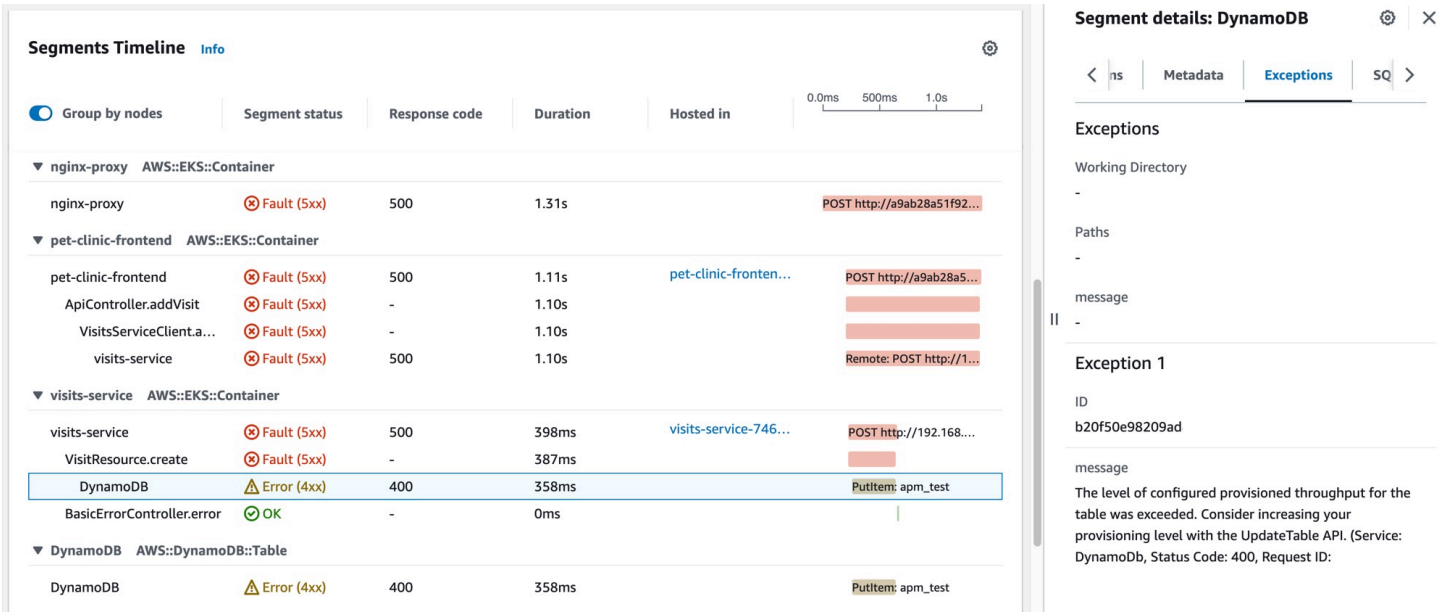
상 작업이 있음을 확인합니다. 이 작업을 선택하면 볼륨 및 가용성 그래프에서 주기적으로 호출 볼륨이 급증하는 것을 확인할 수 있는데, 이는 가용성 저하와 상관관계가 있는 것으로 보입니다.



Jane은 서비스 가용성 저하를 자세히 살펴보기 위해 그래프에서 가용성 데이터 포인트 중 하나를 선택합니다. 선택한 데이터 포인트와 상관관계가 있는 X-Ray 트레이스가 표시된 드로어가 열립니다. 그녀는 장애를 포함하는 트레이스가 여러 개 있음을 확인합니다.



Jane은 장애 상태의 상관관계가 있는 트레이스 중 하나를 선택합니다. 그러면 선택한 트레이스에 대한 X-Ray 트레이스 세부 정보 페이지가 열립니다. Jane은 세그먼트 타임라인 섹션으로 스크롤하여 DynamoDB 테이블을 직접적으로 호출하면 오류가 반환되는 것을 확인할 때까지 호출 경로를 따라갑니다. DynamoDB 세그먼트를 선택하고 오른쪽 드로어의 예외 탭으로 이동합니다.



Jane은 DynamoDB 리소스가 잘못 구성되어 고객 요청이 급증하는 동안 오류가 발생하는 것을 확인했습니다. DynamoDB 테이블의 프로비저닝된 처리량 수준이 주기적으로 초과되어 서비스 가용성 문제와 비정상 SLI가 발생합니다. 이 정보를 바탕으로 그녀의 팀은 더 높은 수준의 프로비저닝 처리량을 구성하고 애플리케이션의 고가용성을 보장할 수 있습니다.

수집된 표준 애플리케이션 지표

⚠ Application Signals는 평가판 릴리스입니다. 이 기능에 대한 피드백이 있는 경우 app-signals-feedback@amazon.com으로 문의해 주세요.

Application Signals는 검색된 서비스에서 표준 애플리케이션 지표를 수집합니다. 이러한 지표는 서비스 성능의 가장 중요한 측면인 지연 시간, 장애 및 오류와 관련이 있습니다. 이를 통해 문제를 식별하고, 성능 추세를 모니터링하고, 리소스를 최적화하여 전반적인 사용자 경험을 개선할 수 있습니다.

다음 표에는 CloudWatch에서 수집할 수 있는 지표가 나열되어 있습니다. 이러한 지표는 `AppSignals` 네임스페이스의 CloudWatch로 전송됩니다.

지표	설명
`Latency`	요청 후 데이터 전송이 시작되기까지의 지연 시간입니다. 단위: 밀리초
`Faults`	HTTP 5XX 서버 측 장애와 OpenTelemetry 스팬 상태 오류 수입니다. 단위: 없음
`Errors`	HTTP 4XX 클라이언트측 오류 수입니다. 이는 서비스 문제로 인한 것이 아닌 요청 오류로 간주됩니다. 따라서 Application Signals 대시보드에 표시되는 `Availability` 지표는 이러한 오류를 서비스 장애로 간주하지 않습니다. 단위: 없음

Application Signals 대시보드에 표시되는 `Availability` 지표는 (1 - **`Faults`**/Total)*100으로 계산됩니다. 성공적인 응답은 5XX 오류가 없는 모든 응답입니다. Application Signals가 `Availability`를 계산할 때 4XX 응답은 성공으로 처리됩니다.

수집된 측정기준 및 측정기준 조합

각 표준 애플리케이션 지표에 대해 다음과 같은 측정기준이 정의됩니다. 측정기준에 대한 자세한 내용은 차원 섹션을 참조하세요.

서비스 지표와 종속성 지표에 대해 서로 다른 측정기준이 수집됩니다. Application Signals에서 검색한 서비스 내에서 마이크로서비스 A가 마이크로서비스 B를 직접적으로 호출하면 마이크로서비스 B가 요청을 처리합니다. 이 경우 마이크로서비스 A는 종속성 지표를 내보내고, 마이크로서비스 B는 서비스 지표를 내보냅니다. 클라이언트가 마이크로서비스 A를 직접적으로 호출하면 마이크로서비스 A가 요청을 처리하고 서비스 지표를 내보냅니다.

서비스 지표의 측정기준

서비스 지표에 대해 다음 측정기준이 수집됩니다.

측정기준	설명
`Service`	서비스의 이름입니다.
`Operation`	API 작업 또는 기타 활동의 이름입니다.
`HostedIn.EKS.Cluster`	서비스가 실행되는 Amazon EKS 클러스터의 이름입니다. 이 측정기준은 서비스가 Amazon EKS에서 실행되는 경우에만 수집됩니다.
`HostedIn.K8s.Namespace`	서비스가 실행되는 Kubernetes 네임스페이스의 이름입니다. 이 측정기준은 서비스가 Amazon EKS에서 실행되는 경우에만 수집됩니다.
`HostedIn.Environment`	서비스가 실행되는 환경의 사용자 정의 이름입니다. 이 측정기준은 서비스가 Amazon EKS가 아닌 환경에서 실행되는 경우에만 수집됩니다.

CloudWatch 콘솔에서 이러한 지표를 볼 때 다음과 같은 측정기준 조합으로 보도록 선택할 수 있습니다.

- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`

Amazon EKS가 아닌 플랫폼의 경우 다음과 같은 측정기준 조합으로 서비스 지표를 볼 수도 있습니다.

- `Service`, `Operation`, `HostedIn.Environment`
- `Service`, `HostedIn.Environment`

종속성 지표에 대한 측정기준

종속성 지표에 대해 다음 측정기준이 수집됩니다.

측정기준	설명
`Service`	서비스의 이름입니다.
`Operation`	API 작업 또는 기타 활동의 이름입니다.
`RemoteService`	호출 중인 원격 서비스의 이름입니다.
`RemoteOperation`	호출 중인 API 작업의 이름입니다.
`HostedIn.EKS.Cluster`	서비스가 실행되는 Amazon EKS 클러스터의 이름입니다. 이 측정기준은 서비스가 Amazon EKS에서 실행되는 경우에만 수집됩니다.
`HostedIn.K8s.Namespace`	서비스가 실행되는 Kubernetes 네임스페이스의 이름입니다. 이 측정기준은 서비스가 Amazon EKS에서 실행되는 경우에만 수집됩니다.
`K8s.RemoteNamespace`	종속성 서비스가 실행되는 Kubernetes 네임스페이스의 이름입니다. 이 측정기준은 서비스가 Amazon EKS에서 실행되는 경우에만 수집됩니다.
`RemoteTarget`	원격 호출에 의해 간접적으로 호출된 리소스의 이름입니다. 원격 호출이 특정 리소스에 대한 것이 아닌 경우 이 측정기준은 가치가 없습니다. 이 측정기준은 서비스가 Amazon EKS에서 실행되는 경우에만 수집됩니다.
`HostedIn.Environment`	서비스가 실행되는 환경의 사용자 정의 이름입니다.

측정기준	설명
	이 측정기준은 서비스가 Amazon EKS가 아닌 환경에서 실행되는 경우에만 수집됩니다.

CloudWatch 콘솔에서 이러한 지표를 볼 때 다음과 같은 측정기준 조합으로 보도록 선택할 수 있습니다.

모든 플랫폼에서 실행

- `RemoteService`

Amazon EKS 클러스터에서 실행

- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `K8s.RemoteNamespace`, `RemoteTarget`
- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `K8s.RemoteNamespace`
- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `RemoteTarget`
- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `RemoteService`, `RemoteOperation`,
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `K8s.RemoteNamespace`
- `Service`, `HostedIn.EKS.Cluster`, `RemoteService`, `K8s.RemoteNamespace`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `K8s.RemoteNamespace`, `RemoteTarget`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `K8s.RemoteNamespace`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `RemoteTarget`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`

Amazon EKS 클러스터 이외의 플랫폼에서 실행

- `Service`, `Operation`, `HostedIn.Environment`
- `Service`, `HostedIn.Environment`
- `Service`, `Operation`, `HostedIn.Environment`, `RemoteService`, `RemoteOperation`, `RemoteTarget`
- `Service`, `Operation`, `HostedIn.Environment`, `RemoteService`, `RemoteOperation`,
- `Service`, `HostedIn.Environment`, `RemoteService`
- `Service`, `HostedIn.Environment`, `RemoteService`, `RemoteOperation`, `RemoteTarget`
- `Service`, `HostedIn.Environment`, `RemoteService`, `RemoteOperation`,

Synthetics 모니터링 사용

Amazon CloudWatch Synthetics를 사용하여 일정에 따라 실행되는 구성 가능한 스크립트인 canary를 생성하여 엔드포인트 및 API를 모니터링할 수 있습니다. canary는 동일한 경로를 따라 고객과 동일한 작업을 수행하므로 애플리케이션에 고객 트래픽이 없는 경우에도 고객 경험을 지속적으로 확인할 수 있습니다. Canary를 사용하면 고객보다 먼저 문제를 발견할 수 있습니다.

canary는 Node.js 또는 Python으로 작성된 스크립트로, Node.js 또는 Python을 프레임워크로 사용하는 Lambda 함수를 계정에 생성합니다. canary는 HTTP 프로토콜과 HTTPS 프로토콜 모두에서 작동합니다. canary는 CloudWatch Synthetics 라이브러리가 포함된 Lambda 계층을 사용합니다. 라이브러리에는 NodeJS canary를 위한 NodeJS 버전의 CloudWatch Synthetics와 Python canary를 위한 Python 버전의 CloudWatch Synthetics가 포함되어 있습니다. 계층은 CloudWatch Synthetics 서비스 계정에 속합니다. 라이브러리는 절대 고객 정보를 전송하거나 저장하지 않습니다. 모든 고객 데이터는 고객 계정에만 저장됩니다.

canary는 Puppeteer 또는 Selenium Webdriver를 통해 헤드리스 Google Chrome 브라우저에 대한 프로그래밍 방식 액세스를 제공합니다. Puppeteer에 대한 자세한 내용은 Puppeteer를 참조하세요. Selenium에 대한 자세한 내용은 www.selenium.dev/를 참조하세요.

canary는 엔드포인트의 가용성과 지연 시간을 확인하고 로드 시간 데이터 및 UI 스크린샷을 저장할 수 있습니다. REST API, URL 및 웹사이트 콘텐츠를 모니터링하고 피싱, 코드 주입 및 교차 사이트 스크립팅으로 인한 무단 변경 사항을 검사할 수 있습니다.

CloudWatch Synthetics는 애플리케이션 서비스, 클라이언트, Synthetics canary 및 서비스 종속성을 검색하고 모니터링할 수 있는 Application Signals와 통합됩니다. Application Signals를 사용하여 서비스의 목록 또는 시각적 맵을 확인하고, 서비스 수준 목표(SLO)를 기준으로 상태 지표를 확인하고, 더 자세한 문제 해결을 위해 상관관계가 있는 X-Ray 트레이스를 드릴다운할 수 있습니다. Application

Signals에서 canary를 보려면 X-Ray 활성 추적을 켭니다. canary는 서비스에 연결된 서비스 맵과 직접적으로 호출하는 서비스의 서비스 세부 정보 페이지에 표시됩니다.

카나리(Canary) 동영상 데모는 다음을 참조하세요.

- Amazon CloudWatch Synthetics 소개
- Amazon CloudWatch Synthetics 데모
- Amazon CloudWatch Synthetics를 사용하여 canary 생성
- Amazon CloudWatch Synthetics를 이용한 모니터링

canary를 한 번 실행하거나 정기적으로 실행할 수 있습니다. canary는 분당 1회꼴로 자주 실행할 수 있습니다. cron 및 rate 표현식을 모두 사용하여 canary를 예약할 수 있습니다.

canary를 생성하고 실행하기 전에 고려해야 할 보안 문제에 대한 자세한 내용은 Synthetics canary에 대한 보안 고려 사항 섹션을 참조하세요.

기본적으로 canary는 `CloudWatchSynthetics` 네임스페이스에 여러 CloudWatch 지표를 생성합니다. 이러한 지표에는 `CanaryName`이 측정기준으로 포함되어 있습니다. 또한 함수 라이브러리의 `executeStep()` 또는 `executeHttpStep()` 함수를 사용하는 canary에는 `StepName`이 측정기준으로 포함되어 있습니다. canary 함수 라이브러리에 대한 자세한 내용은 canary 스크립트에 사용할 수 있는 라이브러리 함수 섹션을 참조하세요.

CloudWatch Synthetics는 AWS X-Ray와 함께 CloudWatch를 사용하여 서비스에 대한 전체적인 뷰를 제공함으로써 성능 병목 현상을 더욱 효율적으로 파악하고 영향을 받는 사용자를 식별하도록 지원하는 X-Ray 트레이스 맵과 효과적으로 통합됩니다. CloudWatch Synthetics를 사용하여 생성한 canary는 트레이스 맵에 나타납니다. 자세한 내용은 X-Ray 트레이스 맵을 참조하세요.

CloudWatch Synthetics는 현재 모든 상용 AWS 리전과 GovCloud 리전에서 사용할 수 있습니다.

> **Note**
>
> 아시아 태평양(오사카)에서는 AWS PrivateLink가 지원되지 않습니다. 아시아태평양(자카르타)에서는 AWS PrivateLink 및 X-Ray가 지원되지 않습니다.

주제

- CloudWatch canary에 필요한 역할 및 권한

- canary 생성
- 그룹
- 로컬로 canary 테스트
- 실패한 canary 문제 해결
- canary 스크립트 샘플 코드
- canary 및 X-Ray 추적
- VPC에서 canary 실행
- canary 아티팩트 암호화
- canary 통계 및 세부 정보 보기
- canary가 게시한 CloudWatch 지표
- canary 편집 또는 삭제
- 여러 canary에 대한 런타임 시작, 중지, 삭제 또는 업데이트
- Amazon EventBridge를 사용하여 canary 이벤트 모니터링

CloudWatch canary에 필요한 역할 및 권한

canary를 생성 및 관리하는 사용자는 물론 canary 자체에 대해 특정 권한이 있어야 합니다.

CloudWatch canary를 관리하는 사용자에게 필요한 역할 및 권한

canary 세부 정보 및 canary 실행 결과를 보려면 `CloudWatchSyntheticsFullAccess` 또는 `CloudWatchSyntheticsReadOnlyAccess` 정책이 연결된 사용자로 로그인해야 합니다. 콘솔에서 모든 Synthetics 데이터를 읽으려면 `AmazonS3ReadOnlyAccess` 및 `CloudWatchReadOnlyAccess` 정책도 필요합니다. canary가 사용하는 소스 코드를 보려면 `AWSLambda_ReadOnlyAccess` 정책도 필요합니다.

canary를 생성하려면 `CloudWatchSyntheticsFullAccess` 정책 또는 유사한 권한 집합이 있는 사용자로 로그인해야 합니다. canary에 대한 IAM 역할을 생성하려면 다음 인라인 정책 문도 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
```

```
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
                "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
            ]
        }
    ]
}
```

> **Important**
>
> 사용자에게 `iam:CreateRole`, `iam:CreatePolicy` 및 `iam:AttachRolePolicy` 권한을 부여하면 해당 사용자에게 AWS 계정에 대한 전체 관리 액세스 권한이 부여됩니다. 예를 들어 이러한 권한을 가진 사용자는 모든 리소스에 대한 전체 권한을 가진 정책을 생성하고 해당 정책을 모든 역할에 연결할 수 있습니다. 이러한 권한을 부여한 사람에게 매우 주의해야 합니다.

정책 연결 및 사용자에게 권한 부여에 대한 자세한 내용은 IAM 사용자의 권한 변경 및 사용자 또는 역할의 인라인 정책을 포함하려면을 참조하세요.

canary에 필요한 역할 및 권한

각 canary는 특정 권한이 연결된 IAM 역할과 연결되어야 합니다. CloudWatch 콘솔을 사용하여 canary를 생성하는 경우 CloudWatch Synthetics에서 canary에 대한 IAM 역할을 생성하도록 선택할 수 있습니다. 이렇게 하면 역할에 필요한 권한이 부여됩니다.

IAM 역할을 직접 생성하거나 AWS CLI 또는 API를 사용하여 canary 생성 시 사용할 수 있는 IAM 역할을 생성하는 경우, 이 섹션에 나열된 권한이 해당 역할에 포함되어야 합니다.

canary에 대한 모든 IAM 역할에는 다음 신뢰 정책 문이 포함되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
```

또한 canary의 IAM 역할에는 다음 명령문 중 하나가 필요합니다.

AWS KMS를 사용하지 않거나 Amazon VPC 액세스가 필요하지 않은 기본 canary

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/s3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/s3/bucket/that/contains/canary/results"

            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
               "StringEquals": {
                "cloudwatch:namespace": "CloudWatchSynthetics"
               }
            }
        }
    ]
}
```

AWS KMS를 사용하여 canary 아티팩트를 암호화하지만 Amazon VPC 액세스가 필요하지 않은 canary

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource":
 "arn:aws:kms:KMS_key_region_name:KMS_key_account_id:key/KMS_key_id",
```

```
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "s3.region_name_of_the_canary_results_S3_bucket.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

AWS KMS를 사용하지 않지만 Amazon VPC 액세스는 필요한 canary

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
```

```
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": [
                "*"
            ]
       }
    ]
}
```

AWS KMS를 사용하여 canary 아티팩트를 암호화하고 Amazon VPC 액세스도 필요한 canary

VPC 사용을 시작하기 위해 비 VPC canary를 업데이트하는 경우, 다음 정책에 나열된 네트워크 인터페이스 권한을 포함하도록 canary의 역할을 업데이트해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": [
                "*"
            ]
       },
       {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource":
 "arn:aws:kms:KMS_key_region_name:KMS_key_account_id:key/KMS_key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "s3.region_name_of_the_canary_results_S3_bucket.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

CloudWatch Synthetics에 대한 AWS 관리형 정책

사용자, 그룹 또는 역할에 권한을 추가할 때 정책을 직접 작성하는 것보다 AWS관리형 정책을 사용하는 것이 더욱 편리합니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하려면 시간과 전문 지식이 필요합니다. 빨리 시작하려면 AWS관리형 정책을 사용할 수 있습니다. 이러한 정책은 일반적인 사용 사례에 적용되며 AWS계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 AWS 관리형 정책을 참조하세요.

AWS 서비스 유지 관리 및 AWS관리형 정책 업데이트입니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스는 때때로 AWS 관리형 정책의 권한을 변경합니다. 이 타입의 업데이트는 정책이 연결된 모든 보안 인증(사용자, 그룹 및 역할)에 적용됩니다.

AWS 관리형 정책에 대한 CloudWatch Synthetics 업데이트

이 서비스가 해당 변경 사항을 추적하기 시작한 이후부터 CloudWatch Synthetics의 AWS 관리형 정책 업데이트에 대한 세부 정보를 살펴봅니다. 이 페이지의 변경 사항에 관한 자동 알림을 받으려면 CloudWatch 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
CloudWatchSyntheticsFullAccess에서 중복 작업 제거	CloudWatch Synthetics는 CloudWatchSyntheticsFullAccess 정책에서 `s3:PutBucketEncryption` 및 `lambda:GetLayerVersionByArn` 작업을 제거했습니다. 이러한 작업은 정책의 다른 권한과 중복되기 때문입니다. 제거된 작업은 어떠한 권한도 제공하지 않았으며, 정책에서 부여한 권한에 대한 순 변경 사항은 없습니다.	2021년 3월 12일
CloudWatch Synthetics에서 변경 사항 추적 시작	CloudWatch Synthetics가 AWS 관리형 정책의 변경 사항을 추적하기 시작했습니다.	2021년 3월 10일

CloudWatchSyntheticsFullAccess

다음은 CloudWatchSyntheticsFullAccess 정책의 내용입니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "synthetics:*"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket",
            "s3:PutEncryptionConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::cw-syn-results-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:ListRoles",
            "s3:ListAllMyBuckets",
            "s3:GetBucketLocation",
            "xray:GetTraceSummaries",
            "xray:BatchGetTraces",
            "apigateway:GET"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:ListBucket"
         ],
         "Resource":"arn:aws:s3:::cw-syn-*"
      },
      {
         "Effect":"Allow",
```

```
         "Action":[
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::aws-synthetics-library-*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":[
            "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
         ],
         "Condition":{
            "StringEquals":{
               "iam:PassedToService":[
                  "lambda.amazonaws.com",
                  "synthetics.amazonaws.com"
               ]
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:GetRole"
         ],
         "Resource":[
            "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "cloudwatch:GetMetricData",
            "cloudwatch:GetMetricStatistics"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "cloudwatch:PutMetricAlarm",
            "cloudwatch:DeleteAlarms"
```

```
            ],
            "Resource":[
               "arn:aws:cloudwatch:*:*:alarm:Synthetics-*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "cloudwatch:DescribeAlarms"
            ],
            "Resource":[
               "arn:aws:cloudwatch:*:*:alarm:*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "lambda:CreateFunction",
               "lambda:AddPermission",
               "lambda:PublishVersion",
               "lambda:UpdateFunctionConfiguration",
               "lambda:GetFunctionConfiguration"
            ],
            "Resource":[
               "arn:aws:lambda:*:*:function:cwsyn-*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "lambda:GetLayerVersion",
               "lambda:PublishLayerVersion"
            ],
            "Resource":[
               "arn:aws:lambda:*:*:layer:cwsyn-*",
               "arn:aws:lambda:*:*:layer:Synthetics:*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "ec2:DescribeVpcs",
               "ec2:DescribeSubnets",
               "ec2:DescribeSecurityGroups"
```

```
         ],
         "Resource":[
            "*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "sns:ListTopics"
         ],
         "Resource":[
            "*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "sns:CreateTopic",
            "sns:Subscribe",
            "sns:ListSubscriptionsByTopic"
         ],
         "Resource":[
            "arn:*:sns:*:*:Synthetics-*"
         ]
      }
   ]
}
```

CloudWatchSyntheticsReadOnlyAccess

다음은 `CloudWatchSyntheticsReadOnlyAccess` 정책의 내용입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:Describe*",
                "synthetics:Get*",
                "synthetics:List*"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

사용자의 특정 canary 보기 제한하기

지정하는 canary에 대한 정보만 사용자가 볼 수 있도록 canary에 대한 정보를 보는 사용자의 기능을 제한할 수 있습니다. 이렇게 하려면 다음과 유사한 `Condition` 설명이 포함된 IAM 정책을 사용하고 해당 정책을 사용자 또는 IAM 역할에 연결합니다.

다음 예시에서는 `name-of-allowed-canary-1` 및 `name-of-allowed-canary-2`에 대한 정보 보기로만 사용자를 제한합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "synthetics:DescribeCanaries",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "synthetics:Names": [
                        "name-of-allowed-canary-1",
                        "name-of-allowed-canary-2"
                    ]
                }
            }
        }
    ]
}
```

CloudWatch Synthetics에서는 `synthetics:Names` 배열에서 최대 5개까지 항목 나열이 지원됩니다.

다음 예시와 같이 * 기호를 허용되는 canary 이름에서 와일드카드로 사용하는 정책을 생성할 수도 있습니다.

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

```
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "synthetics:DescribeCanaries",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "synthetics:Names": [
                        "my-team-canary-*"
                    ]
                }
            }
        }
    ]
}
```

첨부된 이러한 정책 중 하나로 로그인한 사용자는 canary 정보 보기에 CloudWatch 콘솔을 사용할 수 없습니다. 해당 사용자는 정책에 따라 권한이 부여된 canary에 대한 canary 정보만 DescribeCanaries API 또는 describe-canaries AWS CLI 명령을 사용하여 볼 수 있습니다.

canary 생성

> **Important**
>
> 소유권 또는 권한이 있는 엔드포인트 및 API만 모니터링하려면 Synthetics canary를 사용해야 합니다. canary 빈도 설정에 따라 이러한 엔드포인트에서 트래픽이 증가할 수 있습니다.

CloudWatch 콘솔을 사용하여 canary를 생성할 때 CloudWatch에서 제공하는 블루프린트를 사용하여 canary를 생성하거나 자체 스크립트를 작성할 수 있습니다. 자세한 내용은 canary 블루프린트 사용 단원을 참조하세요.

canary에 자체 스크립트를 사용하는 경우 AWS CloudFormation을 사용하여 canary를 생성할 수도 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::Synthetics::Canary 단원을 참조하세요.

자체 스크립트를 작성하는 경우 CloudWatch Synthetics가 라이브러리에 기본 제공하는 여러 함수를 사용할 수 있습니다. 자세한 내용은 Synthetics 런타임 버전 단원을 참조하십시오.

> **Note**
>
> canary를 생성할 때 생성되는 계층 중 하나는 앞에 `Synthetics`가 추가된 Synthetics 계층입니다. 이 계층은 Synthetics 서비스 계정이 소유하며 런타임 코드를 포함합니다.

canary를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Synthetics canary를 선택합니다.
3. Create Canary(canary 생성)를 선택합니다.
4. 다음 중 하나를 선택합니다.
   - 블루프린트 스크립트를 기반으로 canary를 생성하려면 블루프린트 사용을 선택한 다음 생성할 canary 유형을 선택합니다. 각 유형의 블루프린트 작업에 대한 자세한 내용은 canary 블루프린트 사용 단원을 참조하세요.
   - 사용자 고유의 Node.js 스크립트를 업로드하여 사용자 지정 canary를 생성하려면 Upload a script(스크립트 업로드)를 선택합니다.

     그런 다음 스크립트를 스크립트 영역으로 드래그하거나 파일 찾아보기를 선택하여 파일 시스템의 스크립트로 이동할 수 있습니다.
   - S3 버킷에서 스크립트를 가져오려면 [S3에서 가져오기(Import from S3)]를 선택합니다. 그런 다음 소스 위치에 canary에 대한 전체 경로를 입력하거나 S3 찾아보기를 선택합니다.

     사용하는 S3 버킷에 대한 `s3:GetObject` 및 `s3:GetObjectVersion` 권한이 있어야 합니다. 버킷은 canary를 생성할 리전과 동일한 AWS 리전에 있어야 합니다.
5. 이름에 canary의 이름을 입력합니다. 이 이름은 여러 페이지에서 사용되므로 다른 canary와 구별되는 설명이 포함된 이름을 지정하는 것이 좋습니다.
6. 애플리케이션 또는 엔드포인트 URL(Application or endpoint URL)에서 canary로 테스트할 URL을 입력합니다. 이 URL에는 프로토콜(예: https://)이 포함되어야 합니다.

   canary가 VPC에 있는 엔드포인트를 테스트하도록 하려면 이 절차의 뒷부분에서 VPC에 대한 정보도 입력해야 합니다.
7. canary에 대해 사용자 고유의 스크립트를 사용하는 경우 Lambda 핸들러에 canary를 시작할 진입점을 입력합니다. `syn-nodejs-puppeteer-3.4` 또는 `syn-python-selenium-1.1` 이전 버전의 런타임을 사용하는 경우 입력하는 문자열은 `.handler`로 끝나야 합니다. `syn-nodejs-`

`puppeteer-3.4` 또는 `syn-python-selenium-1.1` 이상의 런타임을 사용하는 경우 이 제한이 적용되지 않습니다.

8. 스크립트에서 환경 변수를 사용할 경우 [환경 변수(Environment variables)]를 선택한 다음, 스크립트에 정의된 각 환경 변수에 대해 값을 지정합니다. 자세한 내용은 환경 변수 단원을 참조하세요.
9. [일정(Schedule)]에서 이 canary를 한 번만 실행할지, rate 표현식을 사용하여 지속적으로 실행할지 또는 cron 표현식을 사용하여 일정을 지정할지 선택합니다.

   - CloudWatch 콘솔을 사용하여 지속적으로 실행되는 canary를 생성할 경우 1분에 한 번에서 1시간에 한 번 사이의 실행 비율을 선택할 수 있습니다.

   - canary 일정 지정을 위한 cron 표현식 작성에 대한 자세한 내용은 cron을 사용하여 canary 실행 예약 단원을 참조하세요.
10. (선택 사항) canary에 대한 시간 초과 값을 설정하려면 Additional configuration(추가 구성)을 선택한 다음 시간 초과 값을 지정합니다. Lambda 콜드 스타트와 canary 계측 부팅에 걸리는 시간을 15초 이상 허용합니다.
11. 데이터 보존(Data retention)에서 실패한 canary 실행과 성공한 canary 실행에 대한 정보를 보존할 기간을 지정합니다. 범위는 1~455일입니다.

    이 설정은 CloudWatch Synthetics가 콘솔에 저장하고 표시하는 데이터에만 영향을 줍니다. Amazon S3 버킷에 저장된 데이터나 canary에서 게시하는 로그 또는 지표에는 영향을 주지 않습니다.
12. 데이터 스토리지(Data Storage)에서 canary 테스트 실행의 데이터를 저장하는 데 사용할 S3 버킷을 선택합니다. 버킷 이름은 마침표(.)를 포함할 수 없습니다. 이 값을 비워 두면 기본 S3 버킷이 사용 또는 생성됩니다.

    `syn-nodejs-puppeteer-3.0` 이상 런타임을 사용하는 경우 텍스트 상자에 버킷의 URL을 입력할 때 현재 리전 또는 다른 리전의 버킷을 지정할 수 있습니다. 이전 런타임 버전을 사용하는 경우 버킷이 현재 리전에 있어야 합니다.
13. (선택 사항) 기본적으로 canary는 Amazon S3에 아티팩트를 저장하고, 해당 아티팩트는 AWS 관리형 AWS KMS 키를 사용하여 저장 시 암호화됩니다. 데이터 스토리지(Data Storage) 섹션의 추가 구성(Additional configuration)을 선택하여 다른 암호화 옵션을 사용할 수 있습니다. 그런 다음, 암호화에 사용할 키 유형을 선택할 수 있습니다. 자세한 내용은 canary 아티팩트 암호화 단원을 참조하십시오.
14. [액세스 권한(Access permissions)]에서 canary를 실행할 IAM 역할을 생성할지 아니면 기존 역할을 사용할지 여부를 선택합니다.

CloudWatch Synthetics에서 역할을 생성하도록 하면 필요한 모든 권한이 자동으로 포함됩니다. 역할을 직접 생성하는 경우 필요한 권한에 대한 내용은 canary에 필요한 역할 및 권한 섹션을 참조하세요.

canary를 생성할 때 CloudWatch 콘솔을 사용하여 canary에 대한 역할을 생성하는 경우 다른 canary에 대한 역할을 재사용할 수 없습니다. 이러한 역할은 하나의 canary에만 적용되기 때문입니다. 여러 canary에 사용할 수 있는 역할을 수동으로 생성한 경우에만 기존 역할을 사용할 수 있습니다.

기존 역할을 사용하려면 해당 역할을 Synthetics 및 Lambda에 전달할 `iam:PassRole` 권한이 있어야 합니다. 또한 `iam:GetRole` 권한도 있어야 합니다.

15. (선택 사항) [경보(Alarms)]에서 이 canary에 대해 기본 CloudWatch 경보를 생성할지 여부를 선택합니다. 경보를 생성하기로 선택한 경우 경보는 `Synthetics-Alarm-`*`canaryName`*`-`*`index`* 와 같은 이름 규칙으로 생성됩니다.

    `index`는 이 canary에 대해 생성된 각기 다른 경보를 나타내는 숫자입니다. 예를 들면 첫 번째 경보의 인덱스는 1이고 두 번째 경보의 인덱스는 2입니다.

16. (선택 사항) 이 canary가 VPC에 있는 엔드포인트를 테스트하도록 하려면 VPC 설정을 선택하고 다음을 수행합니다.

    a. 엔드포인트를 호스팅하는 VPC를 선택합니다.

    b. VPC에서 하나 이상의 서브넷을 선택합니다. 실행 중에 Lambda 인스턴스에 IP 주소를 할당할 수 없는 경우 퍼블릭 서브넷에서 Lambda 인스턴스를 실행하도록 구성할 수 없으므로 프라이빗 서브넷을 선택해야 합니다. 자세한 내용은 VPC의 리소스에 액세스하도록 Lambda 함수 구성 단원을 참조하세요.

    c. VPC에서 보안 그룹을 하나 이상 선택합니다.

    엔드포인트가 VPC에 있는 경우 CloudWatch 및 Amazon S3에 정보를 보내도록 canary를 사용 설정해야 합니다. 자세한 내용은 VPC에서 canary 실행 단원을 참조하세요.

17. (선택 사항) 태그에서 이 canary에 대한 태그로 하나 이상의 키-값 페어를 추가합니다. 태그를 사용하면 AWS 리소스를 식별 및 구성하고 AWS 비용을 추적할 수 있습니다. 자세한 내용은 Amazon CloudWatch 리소스 태그 지정 단원을 참조하세요.

18. (선택 사항) [활성 추적(Active tracing)]에서 이 canary에 대해 활성 X-Ray 추적을 사용할지 여부를 선택합니다. 이 옵션은 canary가 런타임 버전 syn-nodejs-2.0 이상을 사용하는 경우에만 사용할 수 있습니다. 자세한 내용은 canary 및 X-Ray 추적 단원을 참조하세요.

canary에 대해 생성되는 리소스

canary를 생성할 때 다음 리소스가 생성됩니다.

- 이름이 `CloudWatchSyntheticsRole-`*`canary-name`*`-`*`uuid`*인 IAM 역할(CloudWatch 콘솔을 사용하여 canary를 생성하고 canary에 대해 새 역할을 생성하도록 지정하는 경우)
- 이름이 `CloudWatchSyntheticsPolicy-`*`canary-name`*`-`*`uuid`*인 IAM 정책
- 이름이 `cw-syn-results-`*`accountID`*`-`*`region`*인 S3 버킷
- 이름이 `Synthetics-Alarm-`*`MyCanaryName`*인 경보(canary에 대해 경보를 생성하려는 경우)
- Lambda 함수 및 계층(블루프린트를 사용하여 canary를 생성하는 경우) 이러한 리소스에는 접두사 `cwsyn-`*`MyCanaryName`*이 있습니다.
- 이름이 `/aws/lambda/cwsyn-`*`MyCanaryName`*`-`*`randomId`*인 CloudWatch Logs 로그 그룹

canary 블루프린트 사용

이 섹션에서는 각 canary 블루프린트 및 각 블루프린트가 가장 적합한 작업에 대해 자세히 설명합니다. 다음과 같은 canary 유형에 대해 블루프린트가 제공됩니다.

- 하트비트 모니터
- API canary
- 잘못된 링크 검사기
- 시각적 모니터링
- canary 레코더
- GUI 워크플로우

블루프린트를 사용하여 canary를 생성할 때 CloudWatch 콘솔에서 필드를 작성하면 페이지의 [스크립트 편집기(Script editor)] 영역에 생성 중인 canary가 Node.js 스크립트로 표시됩니다. 이 영역에서 canary를 편집하여 추가로 사용자 지정할 수도 있습니다.

하트비트 모니터링

하트비트 스크립트는 지정된 URL을 로드하고 페이지의 스크린샷과 HTTP 아카이브 파일(HAR 파일)을 저장합니다. 또한 액세스한 URL의 로그도 저장합니다.

HAR 파일을 사용하여 웹페이지에 대한 자세한 성능 데이터를 볼 수 있습니다. 웹 요청 목록을 분석하고 항목에 대한 로드 시간과 같은 성능 문제를 파악할 수 있습니다.

canary에서 `syn-nodejs-puppeteer-3.1` 이상의 런타임 버전을 사용하는 경우 하트비트 모니터링 블루프린트를 사용하여 여러 URL을 모니터링하고 canary 실행 보고서의 단계 요약에서 각 URL의 상태, 지속 시간, 관련 스크린샷, 실패 원인을 확인할 수 있습니다.

API canary

API canary는 REST API의 기본 읽기 및 쓰기 기능을 테스트할 수 있습니다. REST는 representational state transfer의 약어로 개발자가 API를 생성할 때 따르는 일련의 규칙입니다. 이러한 규칙 중 하나에는 특정 URL에 대한 링크가 데이터 조각을 반환해야 한다고 명시되어 있습니다.

canary는 어느 API와도 함께 작동하며 모든 유형의 기능을 테스트할 수 있습니다. 각 canary는 여러 API를 호출할 수 있습니다.

런타임 버전 `syn-nodejs-2.2` 이상을 사용하는 canary에서 API canary 블루프린트는 API를 HTTP 단계로 모니터링하는 다단계 canary를 지원합니다. 단일 canary에서 여러 API를 테스트할 수 있습니다. 각 단계는 서로 다른 URL에 액세스하고 서로 다른 헤더를 사용하며 헤더 및 응답 본문을 캡처할지 여부에 대해 서로 다른 규칙을 사용할 수 있는 별도의 요청입니다. 헤더 및 응답 본문을 캡처하지 않음으로써 민감한 데이터가 기록되지 않도록 방지할 수 있습니다.

API canary의 각 요청은 다음 정보로 구성됩니다.

- 엔드포인트: 사용자가 요청하는 URL입니다.
- 메서드: 서버로 전송되는 요청의 유형입니다. REST API는 GET(읽기), POST(쓰기), PUT(업데이트), PATCH(업데이트) 및 DELETE(삭제) 작업을 지원합니다.
- 헤더: 클라이언트와 서버 모두에 정보를 제공합니다. 이는 인증 및 본문 내용에 대한 정보를 제공하는 데 사용됩니다. 유효한 헤더 목록은 HTTP 헤더를 참조하세요.
- 데이터(또는 본문): 서버에 전송할 정보가 포함되어 있습니다. 이는 POST, PUT, PATCH 또는 DELETE 요청에만 사용됩니다.

API canary 블루프린트는 GET 및 POST 메서드를 지원합니다. 이 블루프린트를 사용할 때는 헤더를 지정해야 합니다. 예를 들어 **`Authorization`**을 키로 지정하고 필요한 인증 데이터를 해당 키의 값으로 지정할 수 있습니다.

POST 요청을 테스트하는 경우 데이터 필드에 게시할 콘텐츠도 지정합니다.

API Gateway와의 통합

API 블루프린트는 Amazon API Gateway와 통합됩니다. 이를 통해 API Gateway API를 선택하고 canary와 동일한 AWS 계정 및 리전에서 스테이징하거나 교차 계정 및 교차 리전 API 모니터링을 위해

API Gateway에서 Swagger 템플릿을 업로드할 수 있습니다. 그런 다음, Scratch에서 나머지 세부 정보를 입력하는 대신 콘솔에서 선택하여 canary를 생성할 수 있습니다. API Gateway에 대한 자세한 내용은 Amazon API Gateway란? 단원을 참조하세요.

프라이빗 API 사용

Amazon API Gateway에서 프라이빗 API를 사용하는 canary를 생성할 수 있습니다. 자세한 내용은 Amazon API Gateway에서 프라이빗 API 생성을 참조하세요.

잘못된 링크 검사기

잘못된 링크 검사기는 `document.getElementsByTagName('a')`을 사용하여 테스트 중인 URL 내부의 모든 링크를 수집합니다. 지정한 링크 수까지만 테스트하며, URL 자체가 첫 번째 링크로 계산됩니다. 예를 들어 5개의 링크가 포함된 페이지의 모든 링크를 검사하려면 canary가 6개의 링크를 따르도록 지정해야 합니다.

`syn-nodejs-2.0-beta` 런타임 이상을 사용하여 생성된 잘못된 링크 검사기 canary는 다음과 같은 추가 기능을 지원합니다.

- 확인된 링크, 상태 코드, 실패 원인(있는 경우), 소스 및 대상 페이지 스크린샷이 포함된 보고서를 제공합니다.
- canary 결과를 살펴볼 때 잘못된 링크만 표시하도록 필터링한 다음, 실패 원인을 기반으로 링크를 수정할 수 있습니다.
- 이 버전은 각 링크에 대해 주석이 달린 소스 페이지 스크린샷을 캡처하고 링크가 발견된 앵커를 강조 표시합니다. 숨겨진 구성 요소에는 주석이 달려 있지 않습니다.
- 소스 페이지와 대상 페이지 모두, 소스 페이지만 또는 대상 페이지만 스크린샷을 캡처하도록 이 버전을 구성할 수 있습니다.
- 이 버전은 첫 페이지에서 더 많은 링크를 스크레이프한 경우에도 첫 번째 잘못된 링크 이후에 canary 스크립트가 중지되는 이전 버전의 문제를 수정합니다.

`syn-1.0`을 사용하는 기존 canary를 업데이트하여 새 런타임을 사용하려는 경우 canary를 삭제하고 다시 생성해야 합니다. 기존 canary를 새 런타임으로 업데이트해도 이러한 기능을 사용할 수는 없습니다.

잘못된 링크 검사기 canary는 다음과 같은 유형의 링크 오류를 감지합니다.

- 404 페이지를 찾을 수 없음
- 잘못된 호스트 이름

- 잘못된 URL. 예를 들어 URL에 대괄호가 없거나 여분의 슬래시가 있거나 잘못된 프로토콜을 사용합니다.
- 잘못된 HTTP 응답 코드
- 호스트 서버가 콘텐츠가 없고 응답 코드가 없는 빈 응답을 반환합니다.
- HTTP 요청은 canary 실행 중에 지속적으로 시간 초과됩니다.
- 호스트가 잘못 구성되었거나 사용 중이므로 계속 연결이 끊어집니다.

시각적 모니터링 블루프린트

시각적 모니터링 블루프린트에는 canary 실행 중에 생성한 스크린샷과 기준 canary 실행 중에 생성한 스크린샷을 비교하는 코드가 포함되어 있습니다. 두 스크린샷 간의 불일치가 임계 백분율을 초과하면 canary가 실패합니다. 시각적 모니터링은 syn-puppeteer-node-3.2 이상을 실행하는 canary에서 지원됩니다. 현재 Python 및 Selenium을 실행하는 canary에서는 지원되지 않습니다.

시각적 모니터링 블루프린트에는 시각적 모니터링을 사용 설정하는 기본 블루프린트 canary 스크립트에 다음 코드 줄이 포함되어 있습니다.

```
syntheticsConfiguration.withVisualCompareWithBaseRun(true);
```

이 줄을 스크립트에 추가한 후 canary가 처음 성공적으로 실행되면 해당 실행 중에 생성한 스크린샷을 비교를 위한 기준으로 사용합니다. 성공한 첫 번째 canary 실행 후 CloudWatch 콘솔을 사용하여 다음 중 하나를 수행하도록 canary를 편집할 수 있습니다.

- canary의 다음 실행을 새 기준으로 설정합니다.
- 현재 기준 스크린샷에 경계선을 그려서 시각적 비교 중에 무시할 스크린샷 영역을 지정합니다.
- 스크린샷을 제거하여 시각적 모니터링에 사용되지 않도록 합니다.

CloudWatch 콘솔을 사용하여 canary를 편집하는 방법에 대한 자세한 내용은 canary 편집 또는 삭제 단원을 참조하세요.

또한 `nextrun` 또는 `lastrun` 파라미터를 사용하거나 UpdateCanary API에서 canary 실행 ID를 지정하여 기준으로 사용되는 canary 실행을 변경할 수도 있습니다.

시각적 모니터링 청사진을 사용할 때 스크린샷을 생성할 URL을 입력하고 차이 임곗값을 백분율로 지정합니다. 기준 실행 후 해당 임곗값보다 큰 시각적 차이를 감지하는 canary의 향후 실행은 canary 실패를 트리거합니다. 기준 실행 후 canary를 편집하여 시각적 모니터링 중에 무시하려는 경계선을 기준 스크린샷에 '그릴' 수도 있습니다.

시각적 모니터링 기능은 ImageMagick 오픈 소스 소프트웨어 도구 키트를 통해 제공됩니다. 자세한 내용은 ImageMagick을 참조하세요.

canary 레코더

canary 레코더 블루프린트를 통해 CloudWatch Synthetics Recorder를 사용하여 웹 사이트에서의 클릭 및 입력 작업을 기록하고 동일한 단계를 따르는 canary를 생성하는 데 사용할 수 있는 Node.js 스크립트를 자동으로 생성할 수 있습니다. CloudWatch Synthetics Recorder는 Amazon에서 제공하는 Google Chrome 확장 프로그램입니다.

크레딧: CloudWatch Synthetics Recorder는 헤드리스 레코더를 기반으로 합니다.

자세한 내용은 Google Chrome용 CloudWatch Synthetics Recorder 사용 단원을 참조하세요.

GUI 워크플로 빌더

GUI 워크플로 빌더 블루프린트는 웹 페이지에서 작업을 수행할 수 있는지 확인합니다. 예를 들어 웹 페이지에 로그인 양식이 있는 경우 canary는 사용자 및 암호 필드를 채우고 양식을 제출하여 웹 페이지가 올바르게 작동하는지 확인할 수 있습니다.

블루프린트를 사용하여 이 유형의 canary를 생성할 때 웹 페이지에서 canary가 수행할 작업을 지정합니다. 사용할 수 있는 작업은 다음과 같습니다.

- 클릭 - 지정된 요소를 선택하고, 요소를 클릭하거나 선택하는 사용자를 시뮬레이션합니다.

  Node.js 스크립트에서 요소를 지정하려면 `[id=]` 또는 `a[class=]`를 사용합니다.

  Python 스크립트에서 요소를 지정하려면 `xpath //*[@id=]` 또는 `//*[@class=]`를 사용합니다.
- 선택기 확인 - 지정된 요소가 웹 페이지에 있는지 확인합니다. 이 테스트는 이전 작업으로 인해 올바른 요소가 페이지를 채우는지 확인하는 데 유용합니다.

  Node.js 스크립트에서 확인할 요소를 지정하려면 `[id=]` 또는 `a[class=]`를 사용합니다.

  Python 스크립트에서 확인할 요소를 지정하려면 `xpath //*[@id=]` 또는 `//*[class=]`를 사용합니다.
- 텍스트 확인 - 지정된 문자열이 대상 요소 내에 포함되어 있는지 확인합니다. 이 테스트는 이전 작업으로 인해 올바른 텍스트가 표시되는지 확인하는 데 유용합니다.

  Node.js 스크립트에서 요소를 지정하려면 `div[@id=]//h1`과 같은 형식을 사용합니다. 이 작업은 Puppeteer의 `waitForXPath` 함수를 사용하기 때문입니다.

Python 스크립트에서 요소를 지정하려면 `//*[@id=]` 또는 //*[@class=] 같은 xpath 형식을 사용합니다. 이 작업은 Selenium의 `implicitly_wait` 함수를 사용하기 때문입니다.

- 텍스트 입력 - 대상 요소에 지정된 텍스트를 작성합니다.

  Node.js 스크립트에서 확인할 요소를 지정하려면 `[id=]` 또는 `a[class=]`를 사용합니다.

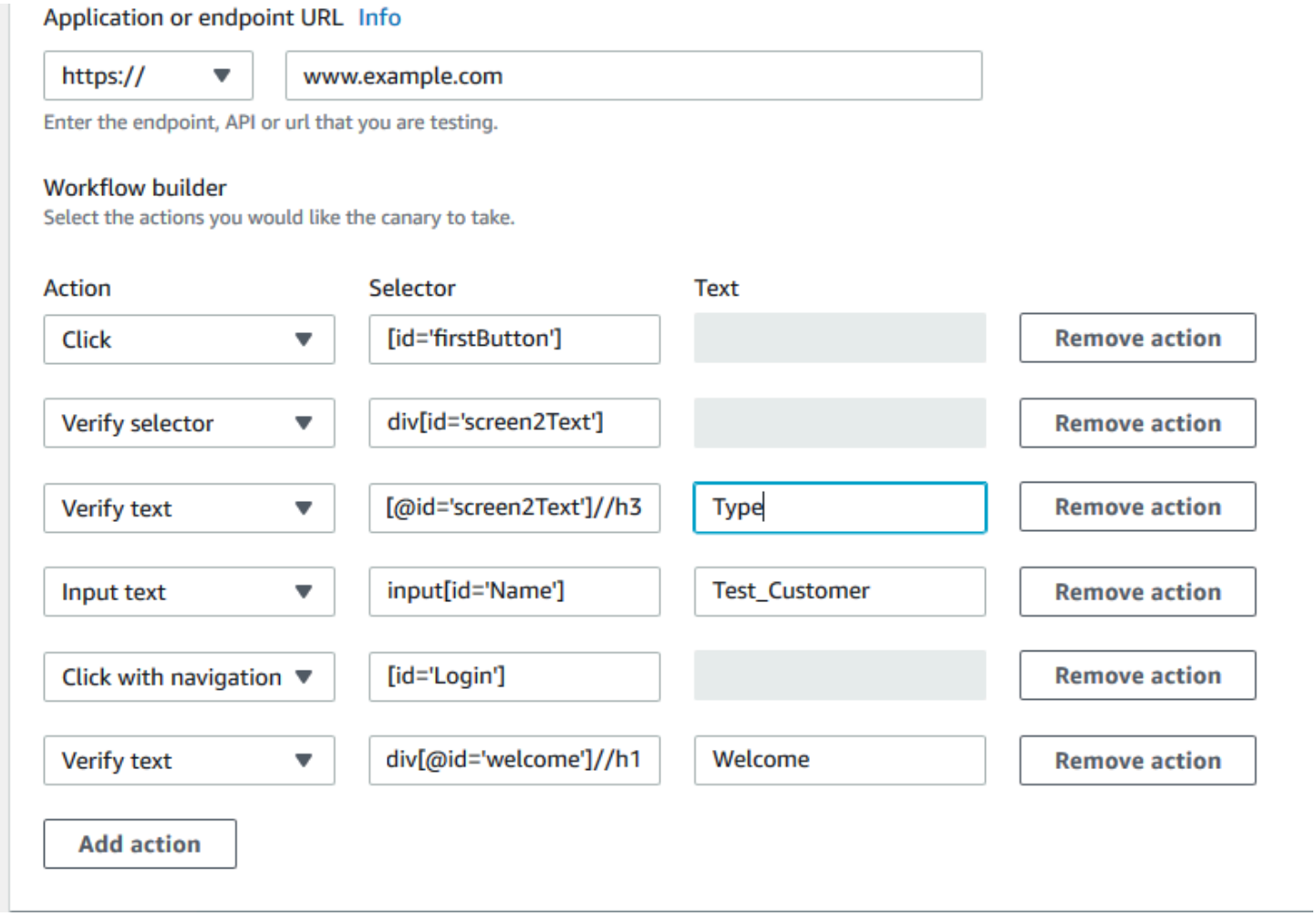
  Python 스크립트에서 확인할 요소를 지정하려면 `xpath //*[@id=]` 또는 `//*[@class=]`를 사용합니다.

- 탐색과 함께 클릭 - 지정된 요소를 선택한 후 전체 페이지가 로드될 때까지 기다립니다. 이는 페이지를 다시 로드해야 할 때 가장 유용합니다.

  Node.js 스크립트에서 요소를 지정하려면 `[id=]` 또는 `a[class=]`를 사용합니다.

  Python 스크립트에서 요소를 지정하려면 `xpath //*[@id=]` 또는 `//*[@class=]`를 사용합니다.

예를 들어 다음 블루프린트는 Node.js를 사용합니다. 지정된 URL의 firstButton을 클릭하고, 예상 텍스트가 있는 예상 선택기가 나타나는지 확인하고, 이름(Name) 필드에 `Test_Customer`라고 이름을 입력합니다. 로그인(Login) 버튼을 클릭한 후 다음 페이지에서 시작(Welcome) 텍스트가 표시되는지 확인하여 로그인이 성공했는지 확인합니다.



다음 런타임을 사용하는 GUI 워크플로 canary는 각 canary 실행에 대해 실행된 단계의 요약도 제공합니다. 각 단계와 관련된 스크린샷 및 오류 메시지를 사용하여 실패의 근본 원인을 찾을 수 있습니다.

- `syn-nodejs-2.0` 이상
- `syn-python-selenium-1.0` 이상

Google Chrome용 CloudWatch Synthetics Recorder 사용

Amazon은 canary를 더욱 쉽게 생성할 수 있도록 지원하기 위해 CloudWatch Synthetics Recorder를 제공합니다. 레코더는 Google Chrome 확장 프로그램입니다.

레코더는 웹 사이트에서의 클릭 및 입력 작업을 기록하고 동일한 단계를 따르는 canary를 생성하는 데 사용할 수 있는 Node.js 스크립트를 자동으로 생성합니다.

기록을 시작하면 CloudWatch Synthetics Recorder가 브라우저에서의 작업을 감지하고 이를 스크립트로 변환합니다. 필요에 따라 기록을 일시 중지하고 다시 시작할 수 있습니다. 기록을 중지하면 레

코더가 작업에 대해 Node.js 스크립트를 생성합니다. [클립보드에 복사(Copy to Clipboard)] 버튼을 사용하면 이 스크립트를 쉽게 복사할 수 있습니다. 그런 다음, 이 스크립트를 사용하여 CloudWatch Synthetics에서 canary를 생성할 수 있습니다.

크레딧: CloudWatch Synthetics Recorder는 헤드리스 레코더를 기반으로 합니다.

Google Chrome용 CloudWatch Synthetics Recorder 확장 프로그램 설치

CloudWatch Synthetics Recorder를 사용하려면 canary 생성을 시작하고 canary 레코더(Canary Recorder) 블루프린트를 선택하면 됩니다. 레코더를 아직 다운로드하지 않았을 때 이 작업을 수행하면 CloudWatch Synthetics 콘솔에서 레코더 다운로드 링크를 제공합니다.

또는 다음 단계에 따라 레코더를 직접 다운로드하여 설치할 수 있습니다.

CloudWatch Synthetics Recorder를 설치하려면

1. Google Chrome을 사용하여 다음 웹 사이트로 이동합니다. https://chrome.google.com/webstore/detail/cloudwatch-synthetics-rec/bhdnlmmgiplmbcdmkkdfplenecpegfno
2. [Chrome에 추가(Add to Chrome)]를 선택한 다음, [확장 프로그램 추가(Add extension)]를 선택합니다.

Google Chrome용 CloudWatch Synthetics Recorder 사용

CloudWatch Synthetics Recorder를 사용하여 canary를 생성하려면 CloudWatch 콘솔에서 canary 생성(Create canary)을 선택한 다음, 블루프린트 사용(Use a blueprint)에서 canary 레코더(Canary Recorder)를 선택하면 됩니다. 자세한 내용은 canary 생성 단원을 참조하세요.

또는 레코더를 사용하여 canary를 생성하는 데 단계를 즉시 사용하지 않고 기록만 할 수도 있습니다.

CloudWatch Synthetics Recorder를 사용하여 웹 사이트 작업 기록

1. 모니터링하려는 페이지로 이동합니다.
2. Chrome 확장 프로그램 아이콘을 선택한 다음, [CloudWatch Synthetics Recorder]를 선택합니다.
3. [기록 시작(Start Recording)]을 선택합니다.
4. 기록하려는 단계를 수행합니다. 기록을 일시 중지하려면 [일시 중지(Pause)]를 선택합니다.
5. 워크플로 기록을 마쳤으면 [기록 중지(Stop recording)]를 선택합니다.
6. [클립보드에 복사(Copy to clipboard)]를 선택하여 생성된 스크립트를 클립보드에 복사합니다. 또는 다시 시작하려면 [새 기록(New recording)]을 선택합니다.

7. 복사한 스크립트를 사용하여 canary를 생성하려면 복사한 스크립트를 레코더 블루프린트 인라인 편집기에 붙여넣거나 Amazon S3 버킷에 저장하고 거기에서 가져오면 됩니다.
8. canary를 즉시 생성하지 않을 경우 기록된 스크립트를 파일에 저장할 수 있습니다.

CloudWatch Synthetics Recorder의 알려진 제한 사항

Google Chrome용 CloudWatch Synthetics Recorder에는 현재 다음과 같은 제한 사항이 있습니다.

- ID가 없는 HTML 요소는 CSS 선택기를 사용합니다. 이러면 나중에 웹 페이지 구조가 변경되는 경우 canary가 손상될 수 있습니다. 향후 레코더 버전에서 이와 관련된 몇 가지 구성 옵션(예: data-id 사용)을 제공할 계획입니다.
- 레코더는 두 번 클릭 또는 복사/붙여넣기와 같은 작업을 지원하지 않으며 CMD+0과 같은 키 조합을 지원하지 않습니다.
- 페이지에 요소 또는 텍스트가 있는지 확인하려면 스크립트가 생성된 후 사용자가 어설션을 추가해야 합니다. 레코더는 요소 확인을 지원하지 않으며 해당 요소에 대한 어떠한 작업도 수행하지 않습니다. 이는 canary 워크플로 빌더의 '텍스트 확인' 또는 '요소 확인' 옵션과 유사합니다. 향후 레코더 버전에서 일부 어설션 지원을 추가할 계획입니다.
- 레코더는 기록이 시작된 탭의 모든 작업을 기록합니다. 팝업(예: 위치 추적 허용) 또는 팝업에서 다른 페이지로의 이동을 기록하지 않습니다.

Synthetics 런타임 버전

canary를 생성하거나 업데이트하는 경우 canary에 대한 Synthetics 런타임 버전을 선택합니다. Synthetics 런타임은 스크립트 핸들러를 호출하는 Synthetics 코드와 Lambda 번들 종속성 계층의 조합입니다.

CloudWatch Synthetics는 현재 스크립트 및 Puppeteer 프레임워크에 Node.js를 사용하는 런타임과 스크립팅에 Python을 사용하고 프레임워크에 Selenium Webdriver를 사용하는 런타임을 지원합니다.

최신 기능과 Synthetics 라이브러리 업데이트를 사용하도록 canary에 항상 최신 런타임 버전을 사용하는 것이 좋습니다.

canary를 생성할 때 생성되는 계층 중 하나는 앞에 `Synthetics`가 추가된 Synthetics 계층입니다. 이 계층은 Synthetics 서비스 계정이 소유하며 런타임 코드를 포함합니다.

> **Note**
>
> 새 버전의 Synthetics 런타임을 사용하도록 canary를 업그레이드할 때마다 canary에서 사용하는 모든 Synthetics 라이브러리 함수도 Synthetics 런타임이 지원하는 것과 동일한 버전의 NodeJS로 자동 업그레이드됩니다.

주제

CloudWatch Synthetics 런타임 지원 정책

Synthetics 런타임 버전에는 유지 관리 및 보안 업데이트가 적용됩니다. 런타임 버전의 구성 요소가 더 이상 지원되지 않는 경우 해당 Synthetics 런타임 버전은 더 이상 사용되지 않습니다.

사용 중단된 런타임 버전을 사용하여 canary를 생성할 수 없습니다. 사용 중단된 런타임을 사용하는 canary는 계속 실행됩니다. 이러한 canary를 중지, 시작 및 삭제할 수 있습니다. 지원되는 런타임 버전을 사용하도록 canary를 업데이트함으로써 사용 중지된 런타임 버전을 사용하는 기존 canary를 업데이트할 수 있습니다.

CloudWatch Synthetics는 향후 60일 이내에 사용 중지될 예정인 런타임을 사용하는 canary가 있는 경우 이를 이메일로 알려줍니다. 최신 릴리스에 포함된 새로운 기능, 보안, 성능 향상의 이점을 활용하려면 canary를 지원되는 런타임 버전으로 마이그레이션하는 것이 좋습니다.

canary를 새 런타임 버전으로 업데이트하려면 어떻게 해야 하나요?

CloudWatch 콘솔, AWS CloudFormation, AWS CLI 또는 AWS SDK를 사용하여 canary의 런타임 버전을 업데이트할 수 있습니다. CloudWatch 콘솔을 사용할 경우 한 번에 최대 5개의 canary를 업데이트할 수 있는데, canary 목록 페이지에서 canary를 선택한 다음, [작업(Actions)], [런타임 업데이트(Update Runtime)]를 선택하면 됩니다.

먼저, CloudWatch 콘솔을 사용하여 canary를 복제하고 해당 런타임 버전을 업데이트하여 업그레이드를 확인할 수 있습니다. 이렇게 하면 원래 canary의 복제인 또 다른 canary가 생성됩니다. 새 런타임 버전의 canary를 확인한 후에는 원래 canary의 런타임 버전을 업데이트하고 복제 canary를 삭제할 수 있습니다.

또한 업그레이드 스크립트를 사용하여 여러 canary를 업데이트할 수도 있습니다. 자세한 내용은 canary 런타임 업그레이드 스크립트 단원을 참조하세요.

canary를 업그레이드했는데 canary가 실패한 경우 실패한 canary 문제 해결 단원을 참조하세요.

런타임 사용 중단 날짜

런타임 버전	사용 중단 날짜
syn-nodejs-puppeteer-6.1	2024년 3월 8일
syn-nodejs-puppeteer-6.0	2024년 3월 8일
syn-nodejs-puppeteer-5.1	2024년 3월 8일
syn-nodejs-puppeteer-5.0	2024년 3월 8일
syn-nodejs-puppeteer-4.0	2024년 3월 8일
syn-nodejs-puppeteer-3.9	2024년 1월 8일
syn-nodejs-puppeteer-3.8	2024년 1월 8일
syn-python-selenium-2.0	2024년 3월 8일
syn-python-selenium-1.3	2024년 3월 8일
syn-python-selenium-1.2	2024년 3월 8일

런타임 버전	사용 중단 날짜
syn-python-selenium-1.1	2024년 3월 8일
syn-python-selenium-1.0	2024년 3월 8일
syn-nodejs-puppeteer-3.7	2024년 1월 8일
syn-nodejs-puppeteer-3.6	2024년 1월 8일
syn-nodejs-puppeteer-3.5	2024년 1월 8일
syn-nodejs-puppeteer-3.4	2022년 11월 13일
syn-nodejs-puppeteer-3.3	2022년 11월 13일
syn-nodejs-puppeteer-3.2	2022년 11월 13일
syn-nodejs-puppeteer-3.1	2022년 11월 13일
syn-nodejs-puppeteer-3.0	2022년 11월 13일
syn-nodejs-2.2	2021년 5월 28일
syn-nodejs-2.1	2021년 5월 28일
syn-nodejs-2.0	2021년 5월 28일
syn-nodejs-2.0-beta	2021년 2월 8일

런타임 버전	사용 중단 날짜
syn-1.0	2021년 5월 28일

canary 런타임 업그레이드 스크립트

canary 스크립트를 지원되는 런타임 버전으로 업그레이드하려면 다음 스크립트를 사용합니다.

```
const AWS = require('aws-sdk');

// You need to configure your AWS credentials and Region.
//   https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/setting-
credentials-node.html
//   https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/setting-
region.html

const synthetics = new AWS.Synthetics();

const DEFAULT_OPTIONS = {
  /**
   * The number of canaries to upgrade during a single run of this script.
   */
  count: 10,
  /**
   * No canaries are upgraded unless force is specified.
   */
  force: false
};

/**
 * The number of milliseconds to sleep between GetCanary calls when
 * verifying that an update succeeded.
 */
const SLEEP_TIME = 5000;

(async () => {
  try {
    const options = getOptions();

    const versions = await getRuntimeVersions();
    const canaries = await getAllCanaries();
    const upgrades = canaries
```

```
      .filter(canary => !versions.isLatestVersion(canary.RuntimeVersion))
      .map(canary => {
        return {
          Name: canary.Name,
          FromVersion: canary.RuntimeVersion,
          ToVersion: versions.getLatestVersion(canary.RuntimeVersion)
        };
      });

    if (options.force) {
      const promises = [];

      for (const upgrade of upgrades.slice(0, options.count)) {
        const promise = upgradeCanary(upgrade);
        promises.push(promise);
        // Sleep for 100 milliseconds to avoid throttling.
        await usleep(100);
      }

      const succeeded = [];
      const failed = [];
      for (let i = 0; i < upgrades.slice(0, options.count).length; i++) {
        const upgrade = upgrades[i];
        const promise = promises[i];
        try {
          await promise;
          console.log(`The update of ${upgrade.Name} succeeded.`);
          succeeded.push(upgrade.Name);
        } catch (e) {
          console.log(`The update of ${upgrade.Name} failed with error: ${e}`);
          failed.push({
            Name: upgrade.Name,
            Reason: e
          });
        }
      }

      if (succeeded.length) {
        console.group('The following canaries were upgraded successfully.');
        for (const name of succeeded) {
          console.log(name);
        }
        console.groupEnd()
      } else {
```

```
        console.log('No canaries were upgraded successfully.');
      }

      if (failed.length) {
        console.group('The following canaries were not upgraded successfully.');
        for (const failure of failed) {
          console.log('\x1b[31m', `${failure.Name}: ${failure.Reason}`, '\x1b[0m');
        }
        console.groupEnd();
      }
    } else {
      console.log('Run with --force [--count <count>] to perform the first <count>
 upgrades shown. The default value of <count> is 10.')
      console.table(upgrades);
    }
  } catch (e) {
    console.error(e);
  }
})();

function getOptions() {
  const force = getFlag('--force', DEFAULT_OPTIONS.force);
  const count = getOption('--count', DEFAULT_OPTIONS.count);
  return { force, count };

  function getFlag(key, defaultValue) {
    return process.argv.includes(key) || defaultValue;
  }
  function getOption(key, defaultValue) {
    const index = process.argv.indexOf(key);
    if (index < 0) {
      return defaultValue;
    }
    const value = process.argv[index + 1];
    if (typeof value === 'undefined' || value.startsWith('-')) {
      throw `The ${key} option requires a value.`;
    }
    return value;
  }
}

function getAllCanaries() {
  return new Promise((resolve, reject) => {
    const canaries = [];
```

```
    synthetics.describeCanaries().eachPage((err, data) => {
      if (err) {
        reject(err);
      } else {
        if (data === null) {
          resolve(canaries);
        } else {
          canaries.push(...data.Canaries);
        }
      }
    });
  });
}

function getRuntimeVersions() {
  return new Promise((resolve, reject) => {
    const jsVersions = [];
    const pythonVersions = [];
    synthetics.describeRuntimeVersions().eachPage((err, data) => {
      if (err) {
        reject(err);
      } else {
        if (data === null) {
          jsVersions.sort((a, b) => a.ReleaseDate - b.ReleaseDate);
          pythonVersions.sort((a, b) => a.ReleaseDate - b.ReleaseDate);
          resolve({
            isLatestVersion(version) {
              const latest = this.getLatestVersion(version);
              return latest === version;
            },
            getLatestVersion(version) {
              if (jsVersions.some(v => v.VersionName === version)) {
                return jsVersions[jsVersions.length - 1].VersionName;
              } else if (pythonVersions.some(v => v.VersionName === version)) {
                return pythonVersions[pythonVersions.length - 1].VersionName;
              } else {
                throw Error(`Unknown version ${version}`);
              }
            }
          });
        } else {
          for (const version of data.RuntimeVersions) {
            if (version.VersionName === 'syn-1.0') {
```

```
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-nodejs-2.')) {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-nodejs-puppeteer-')) {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-python-selenium-')) {
              pythonVersions.push(version);
            } else {
              throw Error(`Unknown version ${version.VersionName}`);
            }
          }
        }
      }
    });
  });
}

async function upgradeCanary(upgrade) {
  console.log(`Upgrading canary ${upgrade.Name} from ${upgrade.FromVersion} to
 ${upgrade.ToVersion}`);
  await synthetics.updateCanary({ Name: upgrade.Name, RuntimeVersion:
 upgrade.ToVersion }).promise();
  while (true) {
    await usleep(SLEEP_TIME);
    console.log(`Getting the state of canary ${upgrade.Name}`);
    const response = await synthetics.getCanary({ Name: upgrade.Name }).promise();
    const state = response.Canary.Status.State;
    console.log(`The state of canary ${upgrade.Name} is ${state}`);
    if (state === 'ERROR' || response.Canary.Status.StateReason) {
      throw response.Canary.Status.StateReason;
    }
    if (state !== 'UPDATING') {
      return;
    }
  }
}

function usleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}
```

Node.js 및 Puppeteer를 사용하는 런타임 버전

Node.js 및 Puppeteer의 첫 번째 런타임 버전 이름은 `syn-1.0`이었습니다. 이 후의 런타임 버전에는 `syn-`*`language`*`-`*`majorversion`*`.`*`minorversion`*이라는 명명 규칙이 있습니다. `syn-nodejs-puppeteer-3.0`부터 명명 규칙은 `syn-`*`language`*`-`*`framework`*`-`*`majorversion`*`.`*`minorversion`*입니다.

추가 `-beta` 접미사는 런타임 버전이 현재 베타 평가판 릴리스임을 보여 줍니다.

메이저 버전 번호가 동일한 런타임 버전은 이전 버전과 호환됩니다.

> **Important**
>
> 다음 CloudWatch Synthetics 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다.
>
> - `syn-nodejs-puppeteer-6.1`
> - `syn-nodejs-puppeteer-6.0`
> - `syn-nodejs-puppeteer-5.1`
> - `syn-nodejs-puppeteer-5.0`
> - `syn-nodejs-puppeteer-4.0`
>
> 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

> **Important**
>
> 중요: 포함된 AWS SDK for JavaScript v2 종속성은 향후 런타임 릴리스에서 AWS SDK for JavaScript v3을 사용하도록 제거되고 업데이트될 예정입니다. 이 경우 canary 코드 참조를 업데이트할 수 있습니다. 또는 소스 코드 zip 파일에 종속성으로 추가하여 포함된 AWS SDK for JavaScript v2 종속성을 계속 참조하고 사용할 수 있습니다.

모든 런타임 버전에 대한 참고 사항

`syn-nodejs-puppeteer-3.0` 런타임 버전을 사용할 경우 canary 스크립트가 Node.js 12.x와 호환되는지 확인하세요. 이전 버전의 `syn-nodejs` 런타임 버전을 사용하는 경우 스크립트가 Node.js 10.x와 호환되는지 확인하세요.

canary의 Lambda 코드는 최대 1GB의 메모리를 갖도록 구성됩니다. 구성된 시간 초과 값 후에 각 canary 실행 시간이 초과됩니다. canary에 대한 시간 제한 값이 지정되지 않은 경우 CloudWatch는 canary의 빈도에 따라 시간 제한 값을 선택합니다. 시간 초과 값을 구성하는 경우, Lambda 콜드 스타트와 canary 계측 부팅에 걸리는 시간을 15초 이상 허용하세요.

> **Note**
>
> 다음 CloudWatch Synthetics 런타임 버전은 2024년 1월 8일부터 사용 중지되었습니다. 이는 AWS Lambda에서 2023년 12월 4일부터 Lambda Node.js 14 런타임을 더 이상 사용하지 않기 때문입니다.
>
> - `syn-nodejs-puppeteer-3.9`
> - `syn-nodejs-puppeteer-3.8`
> - `syn-nodejs-puppeteer-3.7`
> - `syn-nodejs-puppeteer-3.6`
> - `syn-nodejs-puppeteer-3.5`
>
> 다음 CloudWatch Synthetics 런타임 버전은 2022년 11월 13일부터 더 이상 사용되지 않습니다. 이는 AWS Lambda에서 2022년 11월 14일부터 Lambda Node.js 12 런타임을 더 이상 사용하지 않기 때문입니다.
>
> - `syn-nodejs-puppeteer-3.4`
> - `syn-nodejs-puppeteer-3.3`
> - `syn-nodejs-puppeteer-3.2`
> - `syn-nodejs-puppeteer-3.1`
> - `syn-nodejs-puppeteer-3.0`
>
> 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

syn-nodejs-puppeteer-7.0

이 `syn-nodejs-puppeteer-7.0` 런타임은 Lambda 런타임 Node.js 18.x용 최신 런타임 버전입니다. Node.js 및 Puppeteer를 사용합니다.

주요 종속 항목:

- Lambda 런타임 Node.js 18.x
- Puppeteer-core 버전 21.9.0
- Chromium 버전 121.0.6167.139

코드 크기:

이 런타임에 패키징할 수 있는 코드 및 종속성 크기는 80MB입니다.

syn-nodejs-puppeteer-7.0의 새로운 기능:

- Puppeteer 및 Chromium의 번들링된 라이브러리의 버전 업데이트 - Puppeteer 및 Chromium 종속성이 새 버전으로 업데이트되었습니다.

> **Important**
>
> Puppeteer 19.7.0에서 Puppeteer 21.9.0으로 전환하면서 테스트 및 필터와 관련하여 주요 변경 사항이 도입되었습니다. 자세한 내용은 puppeteer: v20.0.0 및 puppeteer-core: v21.0.0의 주요 변경 사항 섹션을 참조하십시오.

AWS SDK v3으로의 업그레이드 권장

Lambda nodejs18.x 런타임은 AWS SDK v2를 지원하지 않습니다. AWS SDK v3로 마이그레이션하는 것이 강력하게 권장됩니다.

syn-nodejs-puppeteer-6.2

주요 종속 항목:

- Lambda 런타임 Node.js 18.x
- Puppeteer-core 버전 19.7.0
- Chromium 버전 111.0.5563.146

syn-nodejs-puppeteer-6.2의 새로운 기능:

- Chromium의 번들 라이브러리 업데이트 버전
- 임시 스토리지 모니터링 — 이 런타임은 고객 계정에 임시 스토리지 모니터링을 추가합니다.
- 버그 수정

syn-nodejs-puppeteer-5.2

이 `syn-nodejs-puppeteer-5.2` 런타임은 Lambda 런타임 Node.js 16.x용 최신 런타임 버전입니다. Node.js 및 Puppeteer를 사용합니다.

주요 종속 항목:

- Lambda 런타임 Node.js 16.x
- Puppeteer-core 버전 19.7.0
- Chromium 버전 111.0.5563.146

syn-nodejs-puppeteer-5.2의 새로운 기능:

- Chromium의 번들 라이브러리 업데이트 버전
- 버그 수정

syn-nodejs-puppeteer-6.1

> **Important**
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 18.x
- Puppeteer-core 버전 19.7.0
- Chromium 버전 111.0.5563.146

syn-nodejs-puppeteer-6.1의 새로운 기능:

- 안정성 개선 - 간헐적인 Puppeteer 실행 오류를 처리하기 위한 자동 재시도 로직이 추가되었습니다.
- 종속성 업그레이드 - 일부 타사 종속성 패키지에 대한 업그레이드입니다.
- Amazon S3 권한이 없는 canary - Amazon S3 권한이 없는 canary를 계속 실행할 수 있도록 버그가 수정되었습니다. Amazon S3 권한이 없는 이러한 canary는 Amazon S3에 스크린샷 또는 기타 아티

팩트를 업로드할 수 없습니다. canary 권한에 대한 자세한 내용은 canary에 필요한 역할 및 권한 섹션을 참조하세요.

> **Important**
>
> 중요: 포함된 AWS SDK for JavaScript v2 종속성은 향후 런타임 릴리스에서 AWS SDK for JavaScript v3을 사용하도록 제거되고 업데이트될 예정입니다. 이 경우 canary 코드 참조를 업데이트할 수 있습니다. 또는 소스 코드 zip 파일에 종속성으로 추가하여 포함된 AWS SDK for JavaScript v2 종속성을 계속 참조하고 사용할 수 있습니다.

syn-nodejs-puppeteer-6.0

> **Important**
>
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 18.x
- Puppeteer-core 버전 19.7.0
- Chromium 버전 111.0.5563.146

syn-nodejs-puppeteer-6.0의 새로운 기능:

- 종속성 업그레이드 - Node.js 종속성이 18.x로 업그레이드되었습니다.
- 인터셉트 모드 지원 - Puppeteer 협동 인터셉트 모드 지원이 Synthetics canary 런타임 라이브러리에 추가되었습니다.
- 추적 동작 변경 - 리소스 요청은 추적하지 않고 fetch 및 xhr 요청만 추적하도록 기본 추적 동작이 변경되었습니다. `traceResourceRequests` 옵션을 구성하여 리소스 요청 추적을 활성화할 수 있습니다.
- 기간 지표 개선 - 이제 canary가 아티팩트를 업로드하고, 스크린샷을 찍고, CloudWatch 지표를 생성하는 데 사용하는 작업 시간이 `Duration` 지표에서 제외됩니다. `Duration` 지표 값은 CloudWatch에 보고되며 Synthetics 콘솔에서도 확인할 수 있습니다.

- 버그 수정— canary 실행 중 Chromium이 충돌할 때 생성되는 코어 덤프를 정리합니다.

> **Important**
>
> 중요: 포함된 AWS SDK for JavaScript v2 종속성은 향후 런타임 릴리스에서 AWS SDK for JavaScript v3을 사용하도록 제거되고 업데이트될 예정입니다. 이 경우 canary 코드 참조를 업데이트할 수 있습니다. 또는 소스 코드 zip 파일에 종속성으로 추가하여 포함된 AWS SDK for JavaScript v2 종속성을 계속 참조하고 사용할 수 있습니다.

syn-nodejs-puppeteer-5.1

> **Important**
>
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 16.x
- Puppeteer-core 버전 19.7.0
- Chromium 버전 111.0.5563.146

syn-nodejs-puppeteer-5.1의 버그 수정:

- 버그 수정 - 이 런타임은 `syn-nodejs-puppeteer-5.0`에서 canary가 생성한 HAR 파일에 요청 헤더가 누락되는 버그를 수정합니다.

syn-nodejs-puppeteer-5.0

> **Important**
>
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 16.x
- Puppeteer-core 버전 19.7.0
- Chromium 버전 111.0.5563.146

syn-nodejs-puppeteer-5.0의 새로운 기능:

- 종속성 업그레이드 - Puppeteer-core 버전이 19.7.0으로 업데이트되었습니다. Chromium 버전이 111.0.5563.146으로 업그레이드되었습니다.

> **Important**
>
> 새 Puppeteer-core 버전은 이전 버전의 Puppeteer와 완전히 호환되지는 않습니다. 이 버전의 일부 변경 사항으로 인해 더 이상 사용되지 않는 Puppeteer 함수를 사용하는 기존 canary가 작동하지 않을 수 있습니다. 자세한 내용은 Puppeteer 변경 로그에서 Puppeteer-core 버전 19.7.0부터 6.0까지에 대한 변경 로그의 주요 변경 사항을 참조하세요.

syn-nodejs-puppeteer-4.0

> **Important**
>
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 16.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 92.0.4512

syn-nodejs-puppeteer-4.0의 새로운 기능:

- 종속성 업그레이드 - Node.js 종속성이 16.x로 업데이트되었습니다.

Node.js 및 Puppeteer에 대한 사용 중지된 런타임

Node.js 및 Puppeteer에 대한 다음 런타임은 더 이상 사용되지 않습니다.

syn-nodejs-puppeteer-3.9

> **Important**
> 이 런타임 버전은 2024년 1월 8일부터 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 14.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 92.0.4512

syn-nodejs-puppeteer-3.9의 새로운 기능:

- 종속성 업그레이드 - 일부 타사 종속성 패키지를 업그레이드합니다.

syn-nodejs-puppeteer-3.8

> **Important**
> 이 런타임 버전은 2024년 1월 8일부터 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 14.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 92.0.4512

syn-nodejs-puppeteer-3.8의 새로운 기능:

- 프로파일 클린업 - 이제 각 canary 실행 후 Chromium 프로파일이 클린업됩니다.

syn-nodejs-puppeteer-3.8의 버그 수정:

- 버그 수정 - 이전에는 스크린샷 없이 실행한 후 시각적 모니터링 canary가 제대로 작동하지 않는 경우가 있었습니다. 이 버그는 이제 수정되었습니다.

syn-nodejs-puppeteer-3.7

> **Important**
>
> 이 런타임 버전은 2024년 1월 8일부터 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 14.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 92.0.4512

syn-nodejs-puppeteer-3.7의 새로운 기능:

- 로깅 향상 - canary는 시간 초과 또는 충돌이 발생하더라도 Amazon S3에 로그를 업로드합니다.
- Lambda 계층 크기 감소 - canary에 사용되는 Lambda 계층의 크기가 34% 감소했습니다.

syn-nodejs-puppeteer-3.7의 버그 수정:

- 버그 수정 - 일본어, 중국어 간체 및 중국어 번체 글꼴이 제대로 렌더링됩니다.

syn-nodejs-puppeteer-3.6

> **Important**
>
> 이 런타임 버전은 2024년 1월 8일부터 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 14.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 92.0.4512

syn-nodejs-puppeteer-3.6의 새로운 기능:

- 보다 정확한 타임스탬프 - canary 실행의 시작 시간과 중지 시간이 이제 밀리초의 정밀도로 기록됩니다.

syn-nodejs-puppeteer-3.5

> **Important**
>
> 이 런타임 버전은 2024년 1월 8일부터 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 14.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 92.0.4512

syn-nodejs-puppeteer-3.5의 새로운 기능:

- 종속 항목 업데이트— 이 런타임의 유일한 새로운 기능은 업데이트된 종속 항목입니다.

syn-nodejs-puppeteer-3.4

> **Important**
>
> 이 런타임 버전은 2022년 11월 13일부터 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 12.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 88.0.4298.0

syn-nodejs-puppeteer-3.4의 새로운 기능:

- 사용자 지정 핸들러 함수 - 이제 canary 스크립트에 사용자 지정 핸들러 함수를 사용할 수 있습니다. 이전 런타임의 경우 `.handler`를 포함할 스크립트 엔트리 포인트가 필요합니다.

  canary 스크립트를 임의의 폴더에 넣고 폴더 이름을 핸들러의 일부로 전달할 수도 있습니다. 예를 들어, `MyFolder/MyScriptFile.functionname`을 진입점으로 사용할 수 있습니다.
- 확장된 HAR 파일 정보 - 이제 canary에서 생성한 HAR 파일에서 불량, 보류 중, 불완전한 요청을 볼 수 있습니다.

syn-nodejs-puppeteer-3.3

> **Important**
>
> 이 런타임 버전은 2022년 11월 13일에 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 12.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 88.0.4298.0

syn-nodejs-puppeteer-3.3의 새로운 기능:

- 아티팩트 암호화에 대한 추가 옵션: 이 런타임 이상을 사용하는 canary의 경우 canary가 Amazon S3에 저장하는 아티팩트를 암호화하는 AWS 관리형 키를 사용하는 대신 AWS KMS 고객 관리형 키 또는 Amazon S3 관리형 키 사용을 선택할 수 있습니다. 자세한 내용은 canary 아티팩트 암호화 단원을 참조하십시오.

syn-nodejs-puppeteer-3.2

> **Important**
>
> 이 런타임 버전은 2022년 11월 13일에 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 12.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 88.0.4298.0

syn-nodejs-puppeteer-3.2의 새로운 기능:

- 스크린샷을 사용한 시각적 모니터링 - 이 런타임 이상 버전을 사용하는 canary는 실행 중에 생성한 스크린샷을 동일한 스크린샷의 기준 버전과 비교할 수 있습니다. 스크린샷이 지정된 백분율 임곗값과 많이 다르면 canary가 실패합니다. 자세한 내용은 시각적 모니터링 또는 시각적 모니터링 블루프린트을 참조하세요.
- 민감한 데이터에 관한 새로운 기능 - 민감한 데이터가 canary 로그 및 보고서에 표시되지 않도록 방지할 수 있습니다. 자세한 내용은 SyntheticsLogHelper 클래스 단원을 참조하세요.
- 사용 중지된 함수 - `RequestResponseLogHelper` 클래스는 새로운 다른 구성 옵션을 위해 더 이상 사용되지 않습니다. 자세한 내용은 RequestResponseLogHelper 클래스 단원을 참조하세요.

syn-nodejs-puppeteer-3.1

> **Important**
>
> 이 런타임 버전은 2022년 11월 13일에 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 12.x
- Puppeteer-core 버전 5.5.0

- Chromium 버전 88.0.4298.0

syn-nodejs-puppeteer-3.1의 새로운 기능:

- CloudWatch 지표 구성 기능 - 이 런타임을 사용하면 필요하지 않은 지표를 사용 중지할 수 있습니다. 그렇지 않으면 canary는 각 canary 실행에 대한 다양한 CloudWatch 지표를 게시합니다.
- 스크린샷 연결 - 단계가 완료된 후 스크린샷을 canary 단계에 연결할 수 있습니다. 이렇게 하려면 [takeScreenshot] 메서드를 사용하여 스크린샷을 생성합니다. 이때 스크린샷을 연결하려는 단계의 이름을 사용합니다. 예를 들어 단계를 수행하고 대기 시간을 추가한 다음, 스크린샷을 생성할 수 있습니다.
- 하트비트 모니터 블루프린트가 여러 URL을 모니터링할 수 있음 -CloudWatch 콘솔에서 하트비트 모니터링 블루프린트를 사용하여 여러 URL을 모니터링하고 canary 실행 보고서의 단계 요약에서 각 URL의 상태, 지속 시간, 관련 스크린샷, 실패 원인을 확인할 수 있습니다.

syn-nodejs-puppeteer-3.0

> **Important**
>
> 이 런타임 버전은 2022년 11월 13일에 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 12.x
- Puppeteer-core 버전 5.5.0
- Chromium 버전 88.0.4298.0

syn-nodejs-puppeteer-3.0의 새로운 기능:

- 종속 항목 업그레이드 - 이 버전은 Puppeteer 버전 5.5.0, Node.js 12.x, Chromium 88.0.4298.0을 사용합니다.
- 교차 리전 버킷 액세스 - 이제 canary가 해당 로그 파일, 스크린샷, HAR 파일을 저장하는 버킷으로 다른 리전의 S3 버킷을 지정할 수 있습니다.
- 새 함수 사용 가능 - 이 버전에서는 canary 이름 및 Synthetics 런타임 버전을 검색하는 라이브러리 함수를 추가합니다.

자세한 내용은 Synthetics 클래스 단원을 참조하세요.

syn-nodejs-2.2

이 단원에는 `syn-nodejs-2.2` 런타임 버전에 관한 정보가 포함되어 있습니다.

> **Important**
>
> 이 런타임 버전은 2021년 5월 28일에 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 10.x
- Puppeteer-core 버전 3.3.0
- Chromium 버전 83.0.4103.0

syn-nodejs-2.2의 새로운 기능:

- HTTP 단계로 canary 모니터링 - 이제 단일 canary에서 여러 API를 테스트할 수 있습니다. 각 API는 별도의 HTTP 단계로 테스트되며, CloudWatch Synthetics는 단계 지표 및 CloudWatch Synthetics 단계 보고서를 사용하여 각 단계의 상태를 모니터링합니다. CloudWatch Synthetics는 각 HTTP 단계에 대해 `SuccessPercent` 및 `Duration` 지표를 생성합니다.

  이 기능은 executeHttpStep(stepName, requestOptions, callback, stepConfig) 함수에 의해 구현됩니다. 자세한 내용은 executeHttpStep(stepName, requestOptions, [callback], [stepConfig]) 단원을 참조하세요.

  API canary 블루프린트는 새로운 이 기능을 사용하도록 업데이트되었습니다.
- HTTP 요청 보고 - 이제 요청 또는 응답 헤더, 응답 본문, 상태 코드, 오류 및 성능 타이밍, TCP 연결 시간, TLS 핸드셰이크 시간, 첫 번째 바이트 시간, 콘텐츠 전송 시간과 같은 세부 정보를 캡처하는 자세한 HTTP 요청 보고서를 볼 수 있습니다. 내부적으로 HTTP/HTTPS 모듈을 사용하는 모든 HTTP 요청이 여기에 캡처됩니다. 헤더 및 응답 본문은 기본적으로 캡처되지 않지만 구성 옵션을 설정하여 사용 설정할 수 있습니다.

- 글로벌 및 단계 수준 구성 - canary의 모든 단계에 적용되는 글로벌 수준에서 CloudWatch Synthetics 구성을 설정할 수 있습니다. 또한 구성 키-값 페어를 전달하여 특정 옵션을 사용하거나 사용 중지함으로써 단계 수준에서 이러한 구성을 재정의할 수도 있습니다.

  자세한 내용은 SyntheticsConfiguration 클래스 단원을 참조하세요.
- 단계 실패 시 계속 구성 - 단계 실패 시 canary 실행을 계속하도록 선택할 수 있습니다. `executeHttpStep` 함수에서 이 옵션이 기본적으로 활성화되어 있습니다. 이 옵션은 글로벌 수준에서 한 번 설정하거나 단계마다 다르게 설정할 수 있습니다.

syn-nodejs-2.1

> **Important**
>
> 이 런타임 버전은 2021년 5월 28일에 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 10.x
- Puppeteer-core 버전 3.3.0
- Chromium 버전 83.0.4103.0

syn-nodejs-2.1의 새로운 기능:

- 구성 가능한 스크린샷 동작 - UI canary에 의한 스크린샷 캡처를 비활성화하는 기능을 제공합니다. 이전 버전의 런타임을 사용하는 canary에서 UI canary는 항상 각 단계 전후에 스크린샷을 캡처합니다. `syn-nodejs-2.1`에서는 이 옵션을 구성할 수 있습니다. 스크린샷을 비활성화하면 Amazon S3 스토리지 비용을 줄이고 HIPAA 규정을 준수하는 데 도움이 될 수 있습니다. 자세한 내용은 SyntheticsConfiguration 클래스 단원을 참조하세요.
- Google Chrome 시작 파라미터 사용자 지정 - 이제 canary가 Google Chrome 브라우저 창을 시작할 때 사용되는 인수를 구성할 수 있습니다. 자세한 내용은 launch(options) 단원을 참조하세요.

이전 버전의 canary 런타임과 비교했을 때 syn-nodejs-2.0 이상을 사용할 경우 canary 지속 시간이 약간 증가할 수 있습니다.

syn-nodejs-2.0

> **Important**
>
> 이 런타임 버전은 2021년 5월 28일에 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Lambda 런타임 Node.js 10.x
- Puppeteer-core 버전 3.3.0
- Chromium 버전 83.0.4103.0

syn-nodejs-2.0의 새로운 기능:

- 종속 항목 업그레이드 - 이 런타임 버전은 Puppeteer-core 버전 3.3.0 및 Chromium 버전 83.0.4103.0을 사용합니다.
- X-Ray 활성추적 지원 - canary에서 추적이 사용 설정된 경우 브라우저, AWS SDK, HTTP 또는 HTTPS 모듈을 사용하는 canary가 수행한 모든 호출에 대해 X-Ray 추적이 전송됩니다. 추적이 활성화된 canary는 X-Ray 트레이스 맵에 나타나며, 이는 추적이 활성화된 다른 서비스 또는 애플리케이션에 요청을 보내지 않는 경우에도 그렇습니다. 자세한 내용은 canary 및 X-Ray 추적 단원을 참조하십시오.
- Synthetics 보고 - CloudWatch Synthetics는 각 canary 실행에 대해 시작 시간, 종료 시간, 상태, 실패와 같은 데이터를 기록하는 `SyntheticsReport-PASSED.json` 또는 `SyntheticsReport-FAILED.json`이라는 보고서를 생성합니다. 또한 canary 스크립트의 각 단계에 대한 PASSED/FAILED 상태, 각 단계에 대해 캡처된 스크린샷 및 실패를 기록합니다.
- 잘못된 링크 검사기 보고서 - 이 런타임에 포함된 잘못된 링크 검사기의 새 버전은 확인된 링크, 상태 코드, 실패 원인(있는 경우), 소스 및 대상 페이지 스크린샷을 포함하는 보고서를 생성합니다.
- 새로운 CloudWatch 지표 - Synthetics는 `CloudWatchSynthetics` 네임스페이스에 `2xx`, `4xx`, `5xx` 및 `RequestFailed`라는 지표를 게시합니다. 이러한 지표는 canary 실행의 200s, 400s, 500s 및 요청 실패 수를 보여 줍니다. 이 런타임 버전에서는 이러한 지표가 UI canary에 대해서만 보고되며 API canary에 대해서는 보고되지 않습니다. 런타임 버전 `syn-nodejs-puppeteeer-2.2`부터는 API canary에 대해서도 보고됩니다.
- 정렬 가능한 HAR 파일 – 이제 상태 코드, 요청 크기, 지속 시간을 기준으로 HAR 파일을 정렬할 수 있습니다.

- 지표 타임스탬프 - 이제 CloudWatch 지표가 canary 실행 종료 시간 대신 Lambda 호출 시간을 기반으로 보고됩니다.

syn-nodejs-2.0의 버그 수정:

- canary 아티팩트 업로드 오류가 보고되지 않는 문제를 수정했습니다. 이러한 오류는 이제 실행 오류로 표시됩니다.
- 리디렉션된 요청(3xx)이 오류로 잘못 로그되는 문제를 수정했습니다.
- 스크린샷 번호가 0부터 시작되는 문제를 수정했습니다. 이제 스크린샷 번호가 1부터 시작됩니다.
- 중국어 및 일본어 글꼴의 스크린샷이 깨져 보이는 문제를 수정했습니다.

이전 버전의 canary 런타임과 비교했을 때 syn-nodejs-2.0 이상을 사용할 경우 canary 지속 시간이 약간 증가할 수 있습니다.

syn-nodejs-2.0-beta

> **Important**
>
> 이 런타임 버전은 2021년 2월 8일에 사용 중지되었습니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하세요.

주요 종속 항목:

- Lambda 런타임 Node.js 10.x
- Puppeteer-core 버전 3.3.0
- Chromium 버전 83.0.4103.0

syn-nodejs-2.0-beta의 새로운 기능:

- 종속 항목 업그레이드 - 이 런타임 버전은 Puppeteer-core 버전 3.3.0 및 Chromium 버전 83.0.4103.0을 사용합니다.
- Synthetics 보고 - CloudWatch Synthetics는 각 canary 실행에 대해 시작 시간, 종료 시간, 상태, 실패와 같은 데이터를 기록하는 `SyntheticsReport-PASSED.json` 또는 `SyntheticsReport-FAILED.json`이라는 보고서를 생성합니다. 또한 canary 스크립트의 각 단계에 대한 PASSED/FAILED 상태, 각 단계에 대해 캡처된 스크린샷 및 실패를 기록합니다.

- 잘못된 링크 검사기 보고서 - 이 런타임에 포함된 잘못된 링크 검사기의 새 버전은 확인된 링크, 상태 코드, 실패 원인(있는 경우), 소스 및 대상 페이지 스크린샷을 포함하는 보고서를 생성합니다.
- 새로운 CloudWatch 지표 - Synthetics는 `CloudWatchSynthetics` 네임스페이스에 `2xx`, `4xx`, `5xx` 및 `RequestFailed`라는 지표를 게시합니다. 이러한 지표는 canary 실행의 200s, 400s, 500s 및 요청 실패 수를 보여 줍니다. 이러한 지표가 UI canary에 대해서만 보고되며 API canary에 대해서는 보고되지 않습니다.
- 정렬 가능한 HAR 파일 – 이제 상태 코드, 요청 크기, 지속 시간을 기준으로 HAR 파일을 정렬할 수 있습니다.
- 지표 타임스탬프 - 이제 CloudWatch 지표가 canary 실행 종료 시간 대신 Lambda 호출 시간을 기반으로 보고됩니다.

syn-nodejs-2.0-beta의 버그 수정:

- canary 아티팩트 업로드 오류가 보고되지 않는 문제를 수정했습니다. 이러한 오류는 이제 실행 오류로 표시됩니다.
- 리디렉션된 요청(3xx)이 오류로 잘못 로그되는 문제를 수정했습니다.
- 스크린샷 번호가 0부터 시작되는 문제를 수정했습니다. 이제 스크린샷 번호가 1부터 시작됩니다.
- 중국어 및 일본어 글꼴의 스크린샷이 깨져 보이는 문제를 수정했습니다.

syn-1.0

> **Important**
>
> 이 런타임 버전은 2021년 5월 28일에 사용 중지될 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하세요.

첫 번째 Synthetics 런타임 버전은 `syn-1.0`입니다.

주요 종속 항목:

- Lambda 런타임 Node.js 10.x
- Puppeteer-core 버전 1.14.0
- Puppeteer-core 1.14.0과 일치하는 Chromium 버전

Python 및 Selenium Webdriver를 사용하는 런타임 버전

다음 단원에는 Python 및 Selenium Webdriver용 CloudWatch Synthetics 런타임 버전에 관한 정보가 포함되어 있습니다. Selenium은 오픈 소스 브라우저 자동화 도구입니다. Selenium에 대한 자세한 내용은 www.selenium.dev/를 참조하세요.

이러한 런타임 버전의 명명 규칙은 `syn-`*`language`*`-`*`framework`*`-`*`majorversion`*`.`*`minorversion`*입니다.

> **Important**
>
> 다음 CloudWatch Synthetics 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다.
>
> - `syn-python-selenium-2.0`
> - `syn-python-selenium-1.3`
> - `syn-python-selenium-1.2`
> - `syn-python-selenium-1.1`
> - `syn-python-selenium-1.0`
>
> 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

syn-python-selenium-3.0

버전 3.0은 Python과 Selenium에 대한 최신 CloudWatch Synthetics 런타임입니다.

주요 종속 항목:

- Python 3.8
- Selenium 4.15.1
- Chromium 버전 121.0.6167.139

syn-python-selenium-3.0의 새로운 기능:

- Chromium의 번들링된 라이브러리의 버전 업데이트 - Chromium 종속성이 새 버전으로 업데이트되었습니다.

syn-python-selenium-2.1

주요 종속 항목:

- Python 3.8
- Selenium 4.15.1
- Chromium 버전 111.0.5563.146

syn-python-selenium-2.1의 새로운 기능:

- Chromium의 번들링된 라이브러리의 버전 업데이트 - Chromium 및 Selenium 종속성이 새 버전으로 업데이트되었습니다.

syn-python-selenium-2.0

> **Important**
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Python 3.8
- Selenium 4.10.0
- Chromium 버전 111.0.5563.146

syn-python-selenium-2.0의 새로운 기능:

- 종속성 업데이트 - Chromium 및 Selenium 종속성이 새 버전으로 업데이트되었습니다.

syn-python-selenium-2.0의 버그 수정:

- 타임스탬프 추가 - canary 로그에 타임스탬프가 추가되었습니다.
- 세션 재사용 - 이제 canary가 이전 canary 실행의 세션을 재사용할 수 없도록 버그가 수정되었습니다.

syn-python-selenium-1.3

> **Important**
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Python 3.8
- Selenium 3.141.0
- Chromium 버전 92.0.4512.0

syn-python-selenium-1.3의 새로운 기능:

- 보다 정확한 타임스탬프 - canary 실행의 시작 시간과 중지 시간이 이제 밀리초의 정밀도로 기록됩니다.

syn-python-selenium-1.2

> **Important**
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Python 3.8
- Selenium 3.141.0
- Chromium 버전 92.0.4512.0

- 종속 항목 업데이트— 이 런타임의 유일한 새로운 기능은 업데이트된 종속 항목입니다.

syn-python-selenium-1.1

> **Important**
>
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Python 3.8
- Selenium 3.141.0
- Chromium 버전 83.0.4103.0

기능:

- 사용자 지정 핸들러 함수 - 이제 canary 스크립트에 사용자 지정 핸들러 함수를 사용할 수 있습니다. 이전 런타임의 경우 `.handler`를 포함할 스크립트 엔트리 포인트가 필요합니다.

  canary 스크립트를 임의의 폴더에 넣고 폴더 이름을 핸들러의 일부로 전달할 수도 있습니다. 예를 들어, `MyFolder/MyScriptFile.functionname`을 진입점으로 사용할 수 있습니다.
- 지표 및 단계 실패 구성을 추가하기 위한 구성 옵션 - 이러한 옵션은 Node.js canary의 런타임에서 이미 사용할 수 있었습니다. 자세한 내용은 SyntheticsConfiguration 클래스 단원을 참조하십시오.
- Chrome의 사용자 지정 인수 - 이제 시크릿 모드로 브라우저를 열거나 프록시 서버 구성을 전달할 수 있습니다. 자세한 내용은 Chrome() 단원을 참조하십시오.
- 교차 리전 아티팩트 버킷 - canary는 다른 리전의 Amazon S3 버킷에 아티팩트를 저장할 수 있습니다.
- **`index.py`** 문제 수정을 포함한 버그 수정 - 이전 런타임에서는 `index.py`로 이름이 지정된 canary 파일이 라이브러리 파일의 이름과 충돌하기 때문에 예외가 발생했습니다. 이제 이 문제가 해결됐습니다.

syn-python-selenium-1.0

> **Important**
>
> 이 런타임 버전은 2024년 3월 8일부터 더 이상 사용되지 않을 예정입니다. 자세한 내용은 CloudWatch Synthetics 런타임 지원 정책 단원을 참조하십시오.

주요 종속 항목:

- Python 3.8
- Selenium 3.141.0
- Chromium 버전 83.0.4103.0

기능:

- Selenium 지원 - Selenium 테스트 프레임워크를 사용하여 canary 스크립트를 작성할 수 있습니다. 다른 곳에서 CloudWatch Synthetics로 Selenium 스크립트를 가져올 수 있으며 최소한의 변경으로도 스크립트가 AWS 서비스에서 작동합니다.

canary 스크립트 작성

다음 섹션에서는 canary 스크립트를 작성하는 방법과 canary를 다른 AWS 서비스 그리고 외부 종속성 및 라이브러리와 통합하는 방법에 대해 설명합니다.

주제

- Node.js canary 스크립트 작성
- Python canary 스크립트 작성
- 기존 Selenium 스크립트를 변경하여 Synthetics canary로 사용
- 비표준 인증서를 인증하도록 기존 Puppeteer Synthetics 스크립트 변경

Node.js canary 스크립트 작성

주제

- Scratch에서 CloudWatch Synthetics canary 생성
- Node.js canary 파일 패키징

- 기존 Puppeteer 스크립트를 변경하여 Synthetics canary로 사용
- 환경 변수
- 다른 AWS 서비스와 canary 통합
- canary가 고정 IP 주소를 사용하도록 지정

Scratch에서 CloudWatch Synthetics canary 생성

다음은 최소 Synthetics canary 스크립트 예입니다. 이 스크립트는 성공적인 실행으로 전달되고 문자열을 반환합니다. 실패한 canary가 어떻게 보이는지 확인하려면 `let fail = false;`를 `let fail = true;`로 변경합니다.

canary 스크립트의 진입점 함수를 정의해야 합니다. 파일이 canary의 `ArtifactS3Location`으로 지정된 Amazon S3 위치에 어떻게 업로드되는지 보려면 /tmp 폴더 아래에 이러한 파일을 생성합니다. 스크립트가 실행되면 통과 또는 실패 상태 및 지속 시간 지표가 CloudWatch에 게시되고 /tmp 폴더의 파일이 S3에 업로드됩니다.

```
const basicCustomEntryPoint = async function () {

    // Insert your code here

    // Perform multi-step pass/fail check

    // Log decisions made and results to /tmp

    // Be sure to wait for all your code paths to complete
    // before returning control back to Synthetics.
    // In that way, your canary will not finish and report success
    // before your code has finished executing

    // Throw to fail, return to succeed
    let fail = false;
    if (fail) {
        throw "Failed basicCanary check.";
    }

    return "Successfully completed basicCanary checks.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
```

```
};
```

다음으로 Synthetics 로깅을 사용하도록 스크립트를 확장하고 AWS SDK를 사용하여 호출합니다. 데모를 위해 이 스크립트는 Amazon DynamoDB 클라이언트를 생성하고 DynamoDB listTables API를 호출합니다. 요청에 대한 응답을 로깅하고 요청이 성공했는지 여부에 따라 성공 또는 실패를 로깅합니다.

```
const log = require('SyntheticsLogger');
const AWS = require('aws-sdk');
// Require any dependencies that your script needs
// Bundle additional files and dependencies into a .zip file with folder structure
// nodejs/node_modules/additional files and folders

const basicCustomEntryPoint = async function () {

    log.info("Starting DynamoDB:listTables canary.");

    let dynamodb = new AWS.DynamoDB();
    var params = {};
    let request = await dynamodb.listTables(params);
    try {
        let response = await request.promise();
        log.info("listTables response: " + JSON.stringify(response));
    } catch (err) {
        log.error("listTables error: " + JSON.stringify(err), err.stack);
        throw err;
    }

    return "Successfully completed DynamoDB:listTables canary.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
};
```

Node.js canary 파일 패키징

Amazon S3 위치를 사용하여 canary 스크립트를 업로드하는 경우 zip 파일은 이 폴더 구조 `nodejs/node_modules/`*`myCanaryFilename.js file`* 아래에 스크립트를 포함해야 합니다.

둘 이상의 `.js` 파일이 있거나 스크립트가 달라지는 종속성이 있는 경우 모두 폴더 구조 `nodejs/node_modules/`*`myCanaryFilename.js file and other folders and files`*를 포함하는 단일 ZIP 파일로 번들할 수 있습니다. `syn-nodejs-puppeteer-3.4` 이상을 사용 중인 경우 canary

파일을 다른 폴더에 넣고 nodejs/node_modules/*myFolder*/*myCanaryFilename.js file and other folders and files*와 같은 폴더 구조를 만들 수 있습니다.

핸들러 이름

스크립트 진입점의 파일 이름과 일치하도록 canary의 스크립트 진입점(핸들러)을 myCanaryFilename.functionName으로 설정해야 합니다. syn-nodejs-puppeteer-3.4 이전 버전의 런타임을 사용하는 경우 functionName은 handler여야 합니다. syn-nodejs-puppeteer-3.4 이상을 사용 중인 경우 함수 이름을 핸들러로 선택할 수 있습니다. syn-nodejs-puppeteer-3.4 이상을 사용 중인 경우 canary를 별도의 폴더에 저장할 수도 있습니다(예: nodejs/node_modules/myFolder/my_canary_filename). 별도의 폴더에 저장하는 경우 스크립트 진입점에 해당 경로를 지정합니다(예: myFolder/my_canary_filename.functionName).

기존 Puppeteer 스크립트를 변경하여 Synthetics canary로 사용

이 섹션에서는 Puppeteer 스크립트를 가져와서 Synthetics canary 스크립트로 실행하도록 수정하는 방법에 대해 설명합니다. Puppeteer에 대한 자세한 내용은 Puppeteer API v1.14.0을 참조하세요.

다음 Puppeteer 스크립트 예로 시작하겠습니다.

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({path: 'example.png'});

  await browser.close();
})();
```

변환 단계는 다음과 같습니다.

- handler 함수를 생성하고 내보냅니다. 핸들러는 스크립트의 진입점 함수입니다. syn-nodejs-puppeteer-3.4 이전 버전의 런타임을 사용하는 경우 핸들러 함수의 이름을 handler로 지정해야 합니다. syn-nodejs-puppeteer-3.4 이상을 사용하는 경우 함수 이름은 맘대로 지정할 수 있지만 스크립트에서 사용되는 이름과 같아야 합니다. 또한 syn-nodejs-puppeteer-3.4 이상을 사용하는 경우 스크립트를 아무 폴더 아래에 저장하고 해당 폴더를 핸들러 이름의 일부로 지정할 수 있습니다.

  ```
  const basicPuppeteerExample = async function () {};
  ```

```
exports.handler = async () => {
    return await basicPuppeteerExample();
};
```

- Synthetics 종속성을 사용합니다.

```
var synthetics = require('Synthetics');
```

- Synthetics.getPage 함수를 사용하여 Puppeteer Page 객체를 가져옵니다.

```
const page = await synthetics.getPage();
```

Synthetics.getPage 함수에 의해 반환되는 페이지 객체에는 로깅을 위해 구성된 page.on `request`, `response` 및 `requestfailed` 이벤트가 있습니다. 또한 Synthetics에서는 페이지의 요청 및 응답에 대한 HAR 파일 생성을 설정하고 canary ARN을 페이지의 발신 요청의 사용자 에이전트 헤더에 추가합니다.

이제 스크립트를 Synthetics canary로 실행할 준비가 되었습니다. 다음은 업데이트된 스크립트입니다.

```
var synthetics = require('Synthetics');  // Synthetics dependency

const basicPuppeteerExample = async function () {
    const page = await synthetics.getPage(); // Get instrumented page from Synthetics
    await page.goto('https://example.com');
    await page.screenshot({path: '/tmp/example.png'}); // Write screenshot to /tmp
 folder
};

exports.handler = async () => {  // Exported handler function
    return await basicPuppeteerExample();
};
```

환경 변수

canary를 생성할 때 환경 변수를 사용할 수 있습니다. 환경 변수를 사용하면 단일 canary 스크립트를 작성한 다음, 다양한 값과 함께 해당 스크립트를 사용하여 유사한 태스크가 있는 여러 canary를 빠르게 생성할 수 있습니다.

예를 들어 조직에 소프트웨어 개발의 다양한 단계에 대한 엔드포인트(예: `prod`, `dev`, `pre-release`)가 있으며 이러한 엔드포인트 각각을 테스트하기 위해 canary를 생성해야 한다고 가정합니다. 소프트

웨어를 테스트하는 단일 canary 스크립트를 작성한 다음, 세 개의 canary 각각을 생성할 때 엔드포인트 환경 변수에 대해 다양한 값을 지정할 수 있습니다. 그런 다음, canary를 생성할 때 스크립트 및 환경 변수에 사용할 값을 지정합니다.

환경 변수의 이름에는 문자, 숫자, 밑줄 문자가 포함될 수 있습니다. 이름은 문자로 시작해야 하며 2자 이상이어야 합니다. 환경 변수의 총 크기는 4KB를 초과할 수 없습니다. Lambda 예약 환경 변수를 환경 변수의 이름으로 지정할 수 없습니다. 예약된 환경 변수에 대한 자세한 내용은 런타임 환경 변수 단원을 참조하세요.

> **Important**
>
> 환경 변수 키와 값은 암호화되지 않습니다. 환경 변수 키와 값에 민감한 정보를 저장하지 마세요.

다음 스크립트 예에서는 두 개의 환경 변수를 사용합니다. 이 스크립트는 웹 페이지를 사용할 수 있는지 여부를 확인하는 canary용입니다. 환경 변수를 사용하여 확인하는 URL과 사용하는 CloudWatch Synthetics 로그 수준을 모두 파라미터화합니다.

다음 함수는 `LogLevel`을 `LOG_LEVEL` 환경 변수의 값으로 설정합니다.

```
synthetics.setLogLevel(process.env.LOG_LEVEL);
```

다음 함수는 `URL`을 `URL` 환경 변수의 값으로 설정합니다.

```
const URL = process.env.URL;
```

다음 코드는 완전한 스크립트입니다. 이 스크립트를 사용하여 canary를 생성할 때 `LOG_LEVEL` 및 `URL` 환경 변수의 값을 지정합니다.

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const pageLoadEnvironmentVariable = async function () {

    // Setting the log level (0-3)
    synthetics.setLogLevel(process.env.LOG_LEVEL);
    // INSERT URL here
    const URL = process.env.URL;
```

```
    let page = await synthetics.getPage();
    //You can customize the wait condition here. For instance,
    //using 'networkidle2' may be less restrictive.
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});
    if (!response) {
        throw "Failed to load page!";
    }
    //Wait for page to render.
    //Increase or decrease wait time based on endpoint being monitored.
    await page.waitFor(15000);
    await synthetics.takeScreenshot('loaded', 'loaded');
    let pageTitle = await page.title();
    log.info('Page title: ' + pageTitle);
    log.debug('Environment variable:' + process.env.URL);

    //If the response status code is not a 2xx success code
    if (response.status() < 200 || response.status() > 299) {
        throw "Failed to load page!";
    }
};

exports.handler = async () => {
    return await pageLoadEnvironmentVariable();
};
```

스크립트에 환경 변수 전달

콘솔에서 canary를 생성할 때 스크립트에 환경 변수를 전달하려면 콘솔의 [환경 변수(Environment variables)] 섹션에서 환경 변수의 키 및 값을 지정합니다. 자세한 내용은 canary 생성 단원을 참조하세요.

API 또는 AWS CLI를 통해 환경 변수를 전달하려면 RunConfig 섹션에서 EnvironmentVariables 파라미터를 사용합니다. 다음은 Environment 및 Region 키가 있는 두 개의 환경 변수를 사용하는 canary를 생성하는 AWS CLI 명령의 예입니다.

```
aws synthetics create-canary --cli-input-json '{
   "Name":"nameofCanary",
   "ExecutionRoleArn":"roleArn",
   "ArtifactS3Location":"s3://cw-syn-results-123456789012-us-west-2",
   "Schedule":{
      "Expression":"rate(0 minute)",
```

```
      "DurationInSeconds":604800
   },
   "Code":{
      "S3Bucket": "canarycreation",
      "S3Key": "cwsyn-mycanaryheartbeat-12345678-d1bd-1234-
abcd-123456789012-12345678-6a1f-47c3-b291-123456789012.zip",
      "Handler":"pageLoadBlueprint.handler"
   },
   "RunConfig": {
      "TimeoutInSeconds":60,
      "EnvironmentVariables": {
         "Environment":"Production",
         "Region": "us-west-1"
      }
   },
   "SuccessRetentionPeriodInDays":13,
   "FailureRetentionPeriodInDays":13,
   "RuntimeVersion":"syn-nodejs-2.0"
}'
```

다른 AWS 서비스와 canary 통합

모든 canary는 AWS SDK 라이브러리를 사용할 수 있습니다. canary를 다른 AWS 서비스와 통합하기 위해 canary를 작성할 때 이 라이브러리를 사용할 수 있습니다.

이렇게 하려면 canary에 다음 코드를 추가해야 합니다. 다음 예에서는 AWS Secrets Manager가 canary와 통합되는 서비스로 사용됩니다.

- AWS SDK를 가져옵니다.

```
const AWS = require('aws-sdk');
```

- 통합하려는 AWS 서비스에 대한 클라이언트를 생성합니다.

```
const secretsManager = new AWS.SecretsManager();
```

- 클라이언트를 사용하여 해당 서비스에 대한 API 호출을 수행합니다.

```
var params = {
  SecretId: secretName
};
return await secretsManager.getSecretValue(params).promise();
```

다음 canary 스크립트 코드 조각은 Secrets Manager와의 통합 예를 자세히 보여 줍니다.

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const AWS = require('aws-sdk');
const secretsManager = new AWS.SecretsManager();

const getSecrets = async (secretName) => {
    var params = {
        SecretId: secretName
    };
    return await secretsManager.getSecretValue(params).promise();
}

const secretsExample = async function () {
    let URL = "<URL>";
    let page = await synthetics.getPage();

    log.info(`Navigating to URL: ${URL}`);
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});

    // Fetch secrets
    let secrets = await getSecrets("secretname")

    /**
    * Use secrets to login.
    *
    * Assuming secrets are stored in a JSON format like:
    * {
    *   "username": "<USERNAME>",
    *   "password": "<PASSWORD>"
    * }
    **/
    let secretsObj = JSON.parse(secrets.SecretString);
    await synthetics.executeStep('login', async function () {
        await page.type(">USERNAME-INPUT-SELECTOR<", secretsObj.username);
        await page.type(">PASSWORD-INPUT-SELECTOR<", secretsObj.password);

        await Promise.all([
          page.waitForNavigation({ timeout: 30000 }),
          await page.click(">SUBMIT-BUTTON-SELECTOR<")
        ]);
```

```
    });

    // Verify login was successful
    await synthetics.executeStep('verify', async function () {
        await page.waitForXPath(">SELECTOR<", { timeout: 30000 });
    });
};

exports.handler = async () => {
    return await secretsExample();
};
```

canary가 고정 IP 주소를 사용하도록 지정

canary가 고정 IP 주소를 사용하도록 canary를 설정할 수 있습니다.

canary가 고정 IP 주소를 사용하도록 지정하려면

1. 새 VPC를 생성합니다. 자세한 내용은 VPC에서 DNS 사용하기 단원을 참조하세요.
2. 새 인터넷 게이트웨이를 생성합니다. 자세한 내용은 VPC에 인터넷 게이트웨이 추가 단원을 참조하세요.
3. 새 VPC 내부에 퍼블릭 서브넷을 생성합니다.
4. VPC에 새 라우팅 테이블을 추가합니다.
5. `0.0.0.0/0`에서 인터넷 게이트웨이로 이동하는 경로를 새 라우팅 테이블에 추가합니다.
6. 새 라우팅 테이블을 퍼블릭 서브넷과 연결합니다.
7. 탄력적 IP 주소를 생성합니다. 자세한 내용은 탄력적 IP 주소 단원을 참조하세요.
8. 새 NAT 게이트웨이를 생성하여 퍼블릭 서브넷 및 탄력적 IP 주소에 할당합니다.
9. VPC 내부에 프라이빗 서브넷을 생성합니다.
10. `0.0.0.0/0`에서 NAT 게이트웨이로 이동하는 경로를 VPC 기본 라우팅 테이블에 추가합니다.
11. canary를 생성합니다.

Python canary 스크립트 작성

이 스크립트는 성공적인 실행으로 전달되고 문자열을 반환합니다. 실패한 canary가 어떻게 보이는지 확인하려면 fail = False를 fail = True로 변경합니다.

```
def basic_custom_script():
    # Insert your code here
```

```
    # Perform multi-step pass/fail check
    # Log decisions made and results to /tmp
    # Be sure to wait for all your code paths to complete
    # before returning control back to Synthetics.
    # In that way, your canary will not finish and report success
    # before your code has finished executing
    fail = False
    if fail:
        raise Exception("Failed basicCanary check.")
    return "Successfully completed basicCanary checks."
def handler(event, context):
    return basic_custom_script()
```

Python canary 파일 패키징

.py 파일이 두 개 이상 있거나 스크립트에 종속 항목이 있는 경우 이들을 모두 단일 ZIP 파일로 번들링할 수 있습니다. `syn-python-selenium-1.1` 런타임을 사용하는 경우 ZIP 파일은 `python` 폴더 내에 주요 canary .py 파일을 포함해야 합니다(예: `python/my_canary_filename.py`). `syn-python-selenium-1.1` 이상을 사용하는 경우 다른 폴더를 사용할 수 있습니다(예: `python/myFolder/my_canary_filename.py`).

이 ZIP 파일은 필요한 모든 폴더와 파일을 포함해야 하지만, `python` 폴더의 다른 파일을 포함하지 않아도 됩니다.

스크립트 진입점의 파일 이름 및 함수 이름과 일치하도록 canary의 스크립트 진입점을 `my_canary_filename.functionName`으로 설정해야 합니다. `syn-python-selenium-1.0` 런타임을 사용하는 경우 `functionName`은 `handler`여야 합니다. `syn-python-selenium-1.1` 이상을 사용 중인 경우 이 핸들러 이름 제한은 적용되지 않으며 canary를 별도의 폴더에 저장할 수도 있습니다(예: `python/myFolder/my_canary_filename.py`). 별도의 폴더에 저장하는 경우 스크립트 진입점에 해당 경로를 지정합니다(예: `myFolder/my_canary_filename.functionName`).

기존 Selenium 스크립트를 변경하여 Synthetics canary로 사용

canary로 사용할 기존의 Python 및 Selenium 스크립트를 빠르게 수정할 수 있습니다. Selenium에 대한 자세한 내용은 www.selenium.dev/를 참조하세요.

이 예에서는 다음 Selenium 스크립트로 시작합니다.

```
from selenium import webdriver

def basic_selenium_script():
    browser = webdriver.Chrome()
```

```
    browser.get('https://example.com')
    browser.save_screenshot('loaded.png')

basic_selenium_script()
```

변환 단계는 다음과 같습니다.

canary로 사용할 Selenium 스크립트를 변환하려면

1. 다음과 같이 aws_synthetics 모듈의 Selenium을 사용하도록 import 문을 변경합니다.

   ```
   from aws_synthetics.selenium import synthetics_webdriver as webdriver
   ```

   aws_synthetics의 Selenium 모듈은 canary가 지표 및 로그를 내보내고 HAR 파일을 생성하며 다른 CloudWatch Synthetics 기능과 함께 작동할 수 있도록 합니다.
2. 핸들러 함수를 생성하고 Selenium 메서드를 호출합니다. 핸들러는 스크립트의 진입점 함수입니다.

   syn-python-selenium-1.0을 사용하는 경우 핸들러 함수의 이름을 handler로 지정해야 합니다. syn-python-selenium-1.1 이상을 사용하는 경우 함수 이름은 맘대로 지정할 수 있지만 스크립트에서 사용되는 이름과 같아야 합니다. 또한 syn-python-selenium-1.1 이상을 사용하는 경우 스크립트를 아무 폴더 아래에 저장하고 해당 폴더를 핸들러 이름의 일부로 지정할 수 있습니다.

   ```
   def handler(event, context):
       basic_selenium_script()
   ```

이제 스크립트가 CloudWatch Synthetics canary로 업데이트되었습니다. 다음은 업데이트된 스크립트입니다.

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver

def basic_selenium_script():
    browser = webdriver.Chrome()
    browser.get('https://example.com')
    browser.save_screenshot('loaded.png')

def handler(event, context):
    basic_selenium_script()
```

비표준 인증서를 인증하도록 기존 Puppeteer Synthetics 스크립트 변경

Synthetics canary의 중요한 사용 사례 중 하나는 자체 엔드포인트를 모니터링하는 것입니다. 외부 트래픽을 수용할 준비가 되지 않은 엔드포인트를 모니터링하려는 경우 이러한 모니터링으로 인해 신뢰할 수 있는 타사 인증 기관에서 서명한 적절한 인증서가 없는 경우가 있을 수 있습니다.

이 시나리오에 대해 다음과 같이 두 가지 해결 방법이 있습니다.

- 클라이언트 인증서를 인증하려면 Amazon CloudWatch Synthetics를 사용하여 인증을 검증하는 방법-2부를 참조하십시오.
- 자체 서명된 인증서를 인증하려면 Amazon CloudWatch Synthetics에서 자체 서명된 인증서를 사용하여 인증을 검증하는 방법을 참조하십시오.

CloudWatch Synthetics canary를 사용할 때는 이 두 가지 옵션에만 국한되지 않습니다. canary 코드를 확장하여 이러한 특성을 확장하고 비즈니스 로직을 추가할 수 있습니다.

> **Note**
>
> Python 런타임에서 실행되는 Synthetics canary는 기본적으로 `--ignore-certificate-errors` 플래그가 활성화되어 있으므로 이러한 canary는 비표준 인증서 구성을 가진 사이트에 도달하는 데 문제가 없어야 합니다.

canary 스크립트에 사용할 수 있는 라이브러리 함수

CloudWatch Synthetics에는 canary로 사용할 Node.js 스크립트를 작성할 때 호출할 수 있는 여러 기본 제공 클래스 및 함수가 포함되어 있습니다.

일부 클래스 및 함수는 UI canary와 API canary 모두에 적용됩니다. 다른 함수는 UI canary에만 적용됩니다. UI canary는 `getPage()` 함수를 사용하고 웹 페이지 탐색 및 상호 작용을 위한 웹 드라이버로 Puppeteer를 사용하는 canary입니다.

> **Note**
>
> 새 버전의 Synthetics 런타임을 사용하도록 canary를 업그레이드할 때마다 canary에서 사용하는 모든 Synthetics 라이브러리 함수도 Synthetics 런타임이 지원하는 것과 동일한 버전의 NodeJS로 자동 업그레이드됩니다.

주제

- Node.js canary 스크립트에 사용할 수 있는 라이브러리 함수
- Selenium을 사용하는 Python canary 스크립트에 사용할 수 있는 라이브러리 함수

Node.js canary 스크립트에 사용할 수 있는 라이브러리 함수

이 단원에서는 Node.js canary 스크립트에 사용할 수 있는 라이브러리 함수를 설명합니다.

주제

- 모든 canary에 적용되는 Node.js 라이브러리 클래스 및 함수
- UI canary에만 적용되는 Node.js 라이브러리 클래스 및 함수
- API canary에만 적용되는 Node.js 라이브러리 클래스 및 함수

모든 canary에 적용되는 Node.js 라이브러리 클래스 및 함수

다음 Node.js용 CloudWatch Synthetics 라이브러리 함수는 모든 canary에 사용할 수 있습니다.

주제

- Synthetics 클래스
- SyntheticsConfiguration 클래스
- Synthetics Logger
- SyntheticsLogHelper 클래스

Synthetics 클래스

모든 canary에 사용할 수 있는 다음 함수는 Synthetics 클래스에 있습니다.

addExecutionError(errorMessage, ex);

`errorMessage`는 오류를 설명하며, `ex`는 발생한 예외입니다.

`addExecutionError`를 사용하여 canary에 대한 실행 오류를 설정할 수 있습니다. 이 함수는 스크립트 실행을 중단하지 않고 canary에 실패합니다. 또한 `successPercent` 지표에 영향을 주지 않습니다.

오류가 canary 스크립트의 성공 또는 실패를 나타내는 데 중요하지 않은 경우에만 오류를 실행 오류로 추적해야 합니다.

addExecutionError의 사용 예는 다음과 같습니다. 엔드포인트의 가용성을 모니터링하고 페이지가 로드된 후에 스크린샷을 생성합니다. 스크린샷 캡처 실패가 엔드포인트의 가용성을 결정하지 않기 때문에 스크린샷을 생성하는 동안 발생한 오류를 포착하여 실행 오류로 추가할 수 있습니다. 가용성 지표는 여전히 엔드포인트가 실행 중임을 나타내지만 canary 상태는 실패로 표시됩니다. 다음 샘플 코드 블록은 이러한 오류를 포착하여 실행 오류로 추가합니다.

```
try {
        await synthetics.takeScreenshot(stepName, "loaded");
    } catch(ex) {
        synthetics.addExecutionError('Unable to take screenshot ', ex);
    }
```

getCanaryName();

canary의 이름을 반환합니다.

getCanaryArn();

canary의 ARN을 반환합니다.

getCanaryUserAgentString();

canary의 사용자 지정 사용자 에이전트를 반환합니다.

getRuntimeVersion();

이 함수는 런타임 버전 syn-nodejs-puppeteer-3.0 이상에서 사용할 수 있습니다. 이 함수는 canary의 Synthetics 런타임 버전을 반환합니다. 예를 들어 반환 값이 syn-nodejs-puppeteer-3.0일 수 있습니다.

getLogLevel()

Synthetics 라이브러리에 대한 현재 로그 수준을 검색합니다. 가능한 값은 다음과 같습니다.

- 0 – 디버그
- 1 – 정보
- 2 – 경고
- 3 – 오류

예제

```
let logLevel = synthetics.getLogLevel();
```

setLogLevel()

Synthetics 라이브러리의 로그 수준을 설정합니다. 가능한 값은 다음과 같습니다.

- `0` – 디버그
- `1` – 정보
- `2` – 경고
- `3` – 오류

예제

```
synthetics.setLogLevel(0);
```

SyntheticsConfiguration 클래스

이 클래스는 `syn-nodejs-2.1` 런타임 버전 이상에서만 사용할 수 있습니다.

SyntheticsConfiguration 클래스를 사용하여 Synthetics 라이브러리 함수의 동작을 구성할 수 있습니다. 예를 들어 이 클래스를 사용하여 스크린샷을 캡처하지 않도록 `executeStep()` 함수를 구성할 수 있습니다.

canary의 모든 단계에 적용되는 글로벌 수준에서 CloudWatch Synthetics 구성을 설정할 수 있습니다. 또한 구성 키-값 페어를 전달하여 단계 수준에서 이러한 구성을 재정의할 수도 있습니다.

단계 수준에서 옵션을 전달할 수 있습니다. 예를 보려면 async executeStep(stepName, functionToExecute, [stepConfig]); 및 executeHttpStep(stepName, requestOptions, [callback], [stepConfig]) 단원을 참조하세요.

함수 정의:

setConfig(options)

*`options`*는 canary에 대해 구성 가능한 옵션의 집합인 객체입니다. 다음 단원에서는 *`options`*의 가능한 필드를 설명합니다.

모든 canary에 대한 setConfig(options)

syn-nodejs-puppeteer-3.2 이상을 사용하는 canary의 경우 setConfig의 (options)에는 다음 파라미터가 포함될 수 있습니다.

- includeRequestHeaders(boolean) - 보고서에 요청 헤더를 포함할지 여부입니다. 기본값은 false입니다.
- includeResponseHeaders(boolean) - 보고서에 응답 헤더를 포함할지 여부입니다. 기본값은 false입니다.
- restrictedHeaders(array) - 헤더가 포함된 경우 무시할 헤더 값 목록입니다. 이는 요청 헤더와 응답 헤더 모두에 적용됩니다. 예를 들어 includeRequestHeaders를 true로, restrictedHeaders를 ['Authorization']으로 전달하여 자격 증명을 숨길 수 있습니다.
- includeRequestBody(boolean) - 보고서에 요청 본문을 포함할지 여부입니다. 기본값은 false입니다.
- includeResponseBody(boolean) - 보고서에 응답 본문을 포함할지 여부입니다. 기본값은 false입니다.

CloudWatch 지표에 관한 setConfig(options)

syn-nodejs-puppeteer-3.1 이상을 사용하는 canary의 경우 setConfig의 (options)에는 canary에서 게시하는 지표를 결정하는 다음 부울 파라미터가 포함될 수 있습니다. 이러한 각 옵션의 기본값은 true입니다. aggregated로 시작하는 옵션은 CanaryName 측정기준 없이 지표를 내보낼지 여부를 결정합니다. 이러한 지표를 사용하여 모든 canary에 대한 집계 결과를 확인할 수 있습니다. 다른 옵션은 CanaryName 측정기준과 함께 지표를 내보낼지 여부를 결정합니다. 이러한 지표를 사용하여 각 개별 canary의 결과를 확인할 수 있습니다.

canary에서 내보내는 CloudWatch 지표 목록은 canary가 게시한 CloudWatch 지표 단원을 참조하세요.

- failedCanaryMetric(boolean) - 이 canary에 대한 Failed 지표를 (CanaryName 측정기준과 함께) 내보낼지 여부입니다. 기본값은 true입니다.
- failedRequestsMetric(boolean) - 이 canary에 대한 Failed requests 지표를 (CanaryName 측정기준과 함께) 내보낼지 여부입니다. 기본값은 true입니다.
- _2xxMetric(boolean) - 이 canary에 대한 2xx 지표를 (CanaryName 측정기준과 함께) 내보낼지 여부입니다. 기본값은 true입니다.
- _4xxMetric(boolean) - 이 canary에 대한 4xx 지표를 (CanaryName 측정기준과 함께) 내보낼지 여부입니다. 기본값은 true입니다.

- _5xxMetric(boolean) - 이 canary에 대한 5xx 지표를 (CanaryName 측정기준과 함께) 내보낼지 여부입니다. 기본값은 true입니다.
- stepDurationMetric(boolean) - 이 canary에 대한 Step duration 지표를 (CanaryName StepName 측정기준과 함께) 내보낼지 여부입니다. 기본값은 true입니다.
- stepSuccessMetric(boolean) - 이 canary에 대한 Step success 지표를 (CanaryName StepName 측정기준과 함께) 내보낼지 여부입니다. 기본값은 true입니다.
- aggregatedFailedCanaryMetric(boolean) - 이 canary에 대한 Failed 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.
- aggregatedFailedRequestsMetric(boolean) - 이 canary에 대한 Failed Requests 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.
- aggregated2xxMetric(boolean) - 이 canary에 대한 2xx 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.
- aggregated4xxMetric(boolean) - 이 canary에 대한 4xx 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.
- aggregated5xxMetric(boolean) - 이 canary에 대한 5xx 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.
- visualMonitoringSuccessPercentMetric(boolean) - 이 canary에 대한 visualMonitoringSuccessPercent 지표를 내보낼지 여부입니다. 기본값은 true입니다.
- visualMonitoringTotalComparisonsMetric(boolean) - 이 canary에 대한 visualMonitoringTotalComparisons 지표를 내보낼지 여부입니다. 기본값은 false입니다.
- stepsReport(boolean) - 단계 실행 요약을 보고할지 여부입니다. 기본값은 true입니다.
- includeUrlPassword(boolean) - URL에 표시되는 암호를 포함할지 여부입니다. 기본적으로 URL에 표시되는 암호를 로그 및 보고서에서 삭제하여 민감한 데이터가 노출되는 것을 방지합니다. 기본값은 false입니다.
- restrictedUrlParameters(array) - 수정할 URL 경로 또는 쿼리 파라미터의 목록입니다. 이는 로그, 보고서, 오류에 표시되는 URL에 적용됩니다. 파라미터는 대소문자를 구분하지 않습니다. 별표(*)를 값으로 전달하여 모든 URL 경로 및 쿼리 파라미터 값을 수정할 수 있습니다. 기본값은 빈 배열입니다.
- logRequest(boolean) - canary 로그에 모든 요청을 로그할지 여부입니다. UI canary의 경우에는 브라우저에서 보낸 각 요청을 로그합니다. 기본값은 true입니다.
- logResponse(boolean) - canary 로그에 모든 응답을 로그할지 여부입니다. UI canary의 경우에는 브라우저에서 수신한 모든 응답을 로그합니다. 기본값은 true입니다.

- `logRequestBody`(boolean) - canary 로그에 요청과 함께 요청 본문을 로그할지 여부입니다. 이 구성은 `logRequest`가 `true`인 경우에만 적용됩니다. 기본값은 `false`입니다.
- `logResponseBody`(boolean) - canary 로그에 응답과 함께 응답 본문을 로그할지 여부입니다. 이 구성은 `logResponse`가 `true`인 경우에만 적용됩니다. 기본값은 `false`입니다.
- `logRequestHeaders`(boolean) - canary 로그에 요청과 함께 요청 헤더를 로그할지 여부입니다. 이 구성은 `logRequest`가 `true`인 경우에만 적용됩니다. 기본값은 `false`입니다.

  `includeRequestHeaders`가 아티팩트의 헤더를 사용 설정한다는 점에 유의하세요.
- `logResponseHeaders`(boolean) - canary 로그에 응답과 함께 응답 헤더를 로그할지 여부입니다. 이 구성은 `logResponse`가 `true`인 경우에만 적용됩니다. 기본값은 `false`입니다.

  `includeResponseHeaders`가 아티팩트의 헤더를 사용 설정한다는 점에 유의하세요.

> **Note**
>
> 각 canary에 대한 `Duration` 및 `SuccessPercent` 지표를 항상(`CanaryName` 지표가 있거나 없거나 모두) 내보냅니다.

지표를 사용 또는 사용 중지하는 메서드

disableAggregatedRequestMetrics()

canary가 `CanaryName` 측정기준 없이 내보내지는 모든 요청 지표를 내보내지 못하도록 합니다.

disableRequestMetrics()

canary별 지표와 모든 canary에서 집계된 지표를 모두 포함하여 모든 요청 지표를 사용 중지합니다.

disableStepMetrics()

단계 성공 지표 및 단계 지속 시간 지표 모두를 포함하여 모든 단계 지표를 사용 중지합니다.

enableAggregatedRequestMetrics()

canary가 `CanaryName` 측정기준 없이 내보내지는 모든 요청 지표를 내보낼 수 있도록 합니다.

enableRequestMetrics()

canary별 지표와 모든 canary에서 집계된 지표를 모두 포함하여 모든 요청 지표를 사용하도록 설정합니다.

enableStepMetrics()

단계 성공 지표 및 단계 지속 시간 지표 모두를 포함하여 모든 단계 지표를 사용 설정합니다.

get2xxMetric()

canary가 `CanaryName` 측정기준과 함께 `2xx` 지표를 내보낼지 여부를 반환합니다.

get4xxMetric()

canary가 `CanaryName` 측정기준과 함께 `4xx` 지표를 내보낼지 여부를 반환합니다.

get5xxMetric()

canary가 `CanaryName` 측정기준과 함께 `5xx` 지표를 내보낼지 여부를 반환합니다.

getAggregated2xxMetric()

canary가 측정기준 없이 `2xx` 지표를 내보낼지 여부를 반환합니다.

getAggregated4xxMetric()

canary가 측정기준 없이 `4xx` 지표를 내보낼지 여부를 반환합니다.

getAggregatedFailedCanaryMetric()

canary가 측정기준 없이 `Failed` 지표를 내보낼지 여부를 반환합니다.

getAggregatedFailedRequestsMetric()

canary가 측정기준 없이 `Failed requests` 지표를 내보낼지 여부를 반환합니다.

getAggregated5xxMetric()

canary가 측정기준 없이 `5xx` 지표를 내보낼지 여부를 반환합니다.

getFailedCanaryMetric()

canary가 `CanaryName` 측정기준과 함께 `Failed` 지표를 내보낼지 여부를 반환합니다.

getFailedRequestsMetric()

canary가 `CanaryName` 측정기준과 함께 `Failed requests` 지표를 내보낼지 여부를 반환합니다.

getStepDurationMetric()

canary가 이 canary에 대한 `CanaryName` 측정기준과 함께 `Duration` 지표를 내보낼지 여부를 반환합니다.

getStepSuccessMetric()

canary가 이 canary에 대한 `CanaryName` 측정기준과 함께 `StepSuccess` 지표를 내보낼지 여부를 반환합니다.

with2xxMetric(_2xxMetric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `2xx` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with4xxMetric(_4xxMetric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `4xx` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with5xxMetric(_5xxMetric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `5xx` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withAggregated2xxMetric(aggregated2xxMetric)

이 canary에 대한 측정기준 없이 `2xx` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withAggregated4xxMetric(aggregated4xxMetric)

이 canary에 대한 측정기준 없이 `4xx` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withAggregated5xxMetric(aggregated5xxMetric)

이 canary에 대한 측정기준 없이 `5xx` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withAggregatedFailedCanaryMetric(aggregatedFailedCanaryMetric)

이 canary에 대한 측정기준 없이 `Failed` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withAggregatedFailedRequestsMetric(aggregatedFailedRequestsMetric)

이 canary에 대한 측정기준 없이 `Failed requests` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withFailedCanaryMetric(failedCanaryMetric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `Failed` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withFailedRequestsMetric(failedRequestsMetric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `Failed requests` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withStepDurationMetric(stepDurationMetric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `Duration` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withStepSuccessMetric(stepSuccessMetric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `StepSuccess` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

다른 기능을 사용 또는 사용 중지하는 메서드

withHarFile()

이 canary에 대한 HAR 파일을 생성할지 여부를 지정하는 부울 인수를 허용합니다.

withStepsReport()

이 canary에 대한 단계 실행 요약을 보고할지 여부를 지정하는 부울 인수를 허용합니다.

withIncludeUrlPassword()

URL에 표시되는 암호를 로그 및 보고서에 포함할지 여부를 지정하는 부울 인수를 허용합니다.

withRestrictedUrlParameters()

수정할 URL 경로 또는 쿼리 파라미터의 배열을 허용합니다. 이는 로그, 보고서, 오류에 표시되는 URL에 적용됩니다. 별표(*)를 값으로 전달하여 모든 URL 경로 및 쿼리 파라미터 값을 수정할 수 있습니다.

withLogRequest()

canary의 로그에 모든 요청을 로그할지 여부를 지정하는 부울 인수를 허용합니다.

withLogResponse()

canary의 로그에 모든 응답을 로그할지 여부를 지정하는 부울 인수를 허용합니다.

withLogRequestBody()

canary의 로그에 모든 요청 본문을 로그할지 여부를 지정하는 부울 인수를 허용합니다.

withLogResponseBody()

canary의 로그에 모든 응답 본문을 로그할지 여부를 지정하는 부울 인수를 허용합니다.

withLogRequestHeaders()

canary의 로그에 모든 요청 헤더를 로그할지 여부를 지정하는 부울 인수를 허용합니다.

withLogResponseHeaders()

canary의 로그에 모든 응답 헤더를 로그할지 여부를 지정하는 부울 인수를 허용합니다.

getHarFile()

canary가 HAR 파일을 생성할지 여부를 반환합니다.

getStepsReport()

canary가 단계 실행 요약을 보고할지 여부를 반환합니다.

getIncludeUrlPassword()

canary가 URL에 표시되는 암호를 로그 및 보고서에 포함할지 여부를 반환합니다.

getRestrictedUrlParameters()

canary가 URL 경로 또는 쿼리 파라미터를 수정할지 여부를 반환합니다.

getLogRequest()

canary가 canary의 로그에 모든 요청을 로그할지 여부를 반환합니다.

getLogResponse()

canary가 canary의 로그에 모든 응답을 로그할지 여부를 반환합니다.

getLogRequestBody()

canary가 canary의 로그에 모든 요청 본문을 로그할지 여부를 반환합니다.

getLogResponseBody()

canary가 canary의 로그에 모든 응답 본문을 로그할지 여부를 반환합니다.

getLogRequestHeaders()

canary가 canary의 로그에 모든 요청 헤더를 로그할지 여부를 반환합니다.

getLogResponseHeaders()

canary가 canary의 로그에 모든 응답 헤더를 로그할지 여부를 반환합니다.

모든 canary에 대한 함수

- `withIncludeRequestHeaders`(includeRequestHeaders)
- `withIncludeResponseHeaders`(includeResponseHeaders)
- `withRestrictedHeaders`(restrictedHeaders)
- `withIncludeRequestBody`(includeRequestBody)
- `withIncludeResponseBody`(includeResponseBody)
- `enableReportingOptions`() - 모든 보고 옵션 사용 설정-- includeRequestHeaders, includeResponseHeaders, includeRequestBody, includeResponseBody
- `disableReportingOptions`() - 모든 보고 옵션 사용 중지-- includeRequestHeaders, includeResponseHeaders, includeRequestBody, includeResponseBody

UI canary에 대한 setConfig(options)

UI canary의 경우 setConfig에는 다음 부울 파라미터가 포함될 수 있습니다.

- `continueOnStepFailure`(boolean) - 단계가 실패한 후 canary 스크립트를 계속 실행할지 여부입니다(단계 실패에 대해서는 executeStep 함수 참조). 단계가 실패한 경우 canary 실행은 여전히 실패로 표시됩니다. 기본값은 `false`입니다.
- `harFile`(boolean) - HAR 파일을 생성할지 여부입니다. 기본값은 `True`입니다.
- `screenshotOnStepStart`(boolean) - 단계를 시작하기 전에 스크린샷을 생성할지 여부입니다.
- `screenshotOnStepSuccess`(boolean) - 성공적인 단계를 완료한 후 스크린샷을 생성할지 여부입니다.
- `screenshotOnStepFailure`(boolean) - 단계가 실패한 후 스크린샷을 생성할지 여부입니다.

스크린샷을 사용 또는 사용 중지하는 메서드

disableStepScreenshots()

모든 스크린샷 옵션(screenshotOnStepStart, screenshotOnStepSuccess, screenshotOnStepFailure)을 사용 중지합니다.

enableStepScreenshots()

모든 스크린샷 옵션(screenshotOnStepStart, screenshotOnStepSuccess, screenshotOnStepFailure)을 사용하도록 설정합니다. 기본적으로 이러한 메서드는 모두 사용 설정되어 있습니다.

getScreenshotOnStepFailure()

단계가 실패한 후 canary가 스크린샷을 생성할지 여부를 반환합니다.

getScreenshotOnStepStart()

단계를 시작하기 전에 canary가 스크린샷을 생성할지 여부를 반환합니다.

getScreenshotOnStepSuccess()

단계를 성공적으로 완료한 후 canary가 스크린샷을 생성할지 여부를 반환합니다.

withScreenshotOnStepStart(screenshotOnStepStart)

단계를 시작하기 전에 스크린샷을 생성할지 여부를 나타내는 부울 인수를 허용합니다.

withScreenshotOnStepSuccess(screenshotOnStepSuccess)

단계를 성공적으로 완료한 후 스크린샷을 생성할지 여부를 나타내는 부울 인수를 허용합니다.

withScreenshotOnStepFailure(screenshotOnStepFailure)

단계가 실패한 후 스크린샷을 생성할지 여부를 나타내는 부울 인수를 허용합니다.

UI canary의 사용

먼저, Synthetics 종속 항목을 가져오고 구성을 가져옵니다.

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();
```

다음으로, 다음 옵션 중 하나를 사용하여 setConfig 메서드를 호출함으로써 각 옵션의 구성을 설정합니다.

```
// Set configuration values
    synConfig.setConfig({
        screenshotOnStepStart: true,
        screenshotOnStepSuccess: false,
        screenshotOnStepFailure: false
    });
```

Or

```
synConfig.withScreenshotOnStepStart(false).withScreenshotOnStepSuccess(true).withScreenshotOnSt
```

모든 스크린샷을 사용 중지하려면 이 예와 같이 disableStepScreenshots() 함수를 사용합니다.

```
synConfig.disableStepScreenshots();
```

코드의 어느 지점에서든 스크린샷을 사용 및 사용 중지할 수 있습니다. 예를 들어 한 단계에 대해서만 스크린샷을 사용 중지하려면 해당 단계를 실행하기 전에 스크린샷을 사용 중지한 다음, 이 단계가 끝나면 스크린샷을 사용하도록 설정합니다.

API canary에 대한 setConfig(options)

API canary의 경우 setConfig에는 다음 부울 파라미터가 포함될 수 있습니다.

- `continueOnHttpStepFailure`(boolean) - HTTP 단계가 실패한 후 canary 스크립트를 계속 실행할지 여부입니다(단계 실패에 대해서는 executeHttpStep 함수 참조). 단계가 실패한 경우 canary 실행은 여전히 실패로 표시됩니다. 기본값은 `true`입니다.

시각적 모니터링

시각적 모니터링에서는 canary 실행 중에 생성한 스크린샷과 기준 canary 실행 중에 생성한 스크린샷을 비교합니다. 두 스크린샷 간의 불일치가 임계 백분율을 초과하면 canary가 실패하며, canary 실행 보고서에서 색상으로 강조 표시된 차이가 있는 영역을 확인할 수 있습니다. 시각적 모니터링은 syn-puppeteer-node-3.2 이상을 실행하는 canary에서 지원됩니다. 현재 Python 및 Selenium을 실행하는 canary에서는 지원되지 않습니다.

시각적 모니터링을 사용하려면 canary 스크립트에 다음 코드 줄을 추가합니다. 자세한 내용은 SyntheticsConfiguration 클래스을(를) 참조하세요.

```
syntheticsConfiguration.withVisualCompareWithBaseRun(true);
```

이 줄을 스크립트에 추가한 후 canary가 처음 성공적으로 실행되면 해당 실행 중에 생성한 스크린샷을 비교를 위한 기준으로 사용합니다. 성공한 첫 번째 canary 실행 후 CloudWatch 콘솔을 사용하여 다음 중 하나를 수행하도록 canary를 편집할 수 있습니다.

- canary의 다음 실행을 새 기준으로 설정합니다.
- 현재 기준 스크린샷에 경계선을 그려서 시각적 비교 중에 무시할 스크린샷 영역을 지정합니다.
- 스크린샷을 제거하여 시각적 모니터링에 사용되지 않도록 합니다.

CloudWatch 콘솔을 사용하여 canary를 편집하는 방법에 대한 자세한 내용은 canary 편집 또는 삭제 단원을 참조하세요.

시각적 모니터링을 위한 기타 옵션

syntheticsConfiguration.withVisualVarianceThresholdPercentage(desiredPercentage)

시각적 비교에서 스크린샷 불일치에 대해 허용 가능한 백분율을 설정합니다.

syntheticsConfiguration.withVisualVarianceHighlightHexColor("#fafa00")

시각적 모니터링을 사용하는 canary 실행 보고서를 검토할 때 불일치 영역을 지정하는 강조 표시 색상을 설정합니다.

syntheticsConfiguration.withFailCanaryRunOnVisualVariance(failCanary)

임곗값보다 큰 시각적 차이가 있는 경우 canary 실패 여부를 설정합니다. 기본값은 canary 실패입니다.

Synthetics Logger

SyntheticsLogger는 동일한 로그 수준으로 콘솔과 로컬 로그 파일에 모두 로그를 기록합니다. 이 로그 파일은 로그 수준이 호출된 로그 함수의 원하는 로깅 수준 이하인 경우에만 두 위치에 기록됩니다.

로컬 로그 파일의 로깅 문 앞에는 호출된 함수의 로그 수준과 일치하도록 “DEBUG: “, “INFO: “등이 추가됩니다.

Synthetics canary 로깅과 동일한 로그 수준에서 Synthetics 라이브러리를 실행하려는 경우 SyntheticsLogger를 사용할 수 있습니다.

S3 결과 위치에 업로드되는 로그 파일을 생성할 때는 SyntheticsLogger를 사용할 필요가 없습니다. 대신 `/tmp` 폴더에 다른 로그 파일을 생성할 수 있습니다. `/tmp` 폴더 아래에 생성된 모든 파일은 S3의 결과 위치에 아티팩트로 업로드됩니다.

Synthetics Library Logger를 사용하려면 다음을 실행합니다.

```
const log = require('SyntheticsLogger');
```

유용한 함수 정의:

log.debug(***message***, ***ex***);

파라미터: *message*는 로깅할 메시지이며 *ex*는 기록할 예외입니다(있는 경우).

예제

```
log.debug("Starting step - login.");
```

log.error(***message***, ***ex***);

파라미터: *message*는 로깅할 메시지이며 *ex*는 기록할 예외입니다(있는 경우).

예제

```
try {
  await login();
catch (ex) {
```

```
  log.error("Error encountered in step - login.", ex);
}
```

log.info(***message***, ***ex***);

파라미터: *message*는 로깅할 메시지이며 *ex*는 기록할 예외입니다(있는 경우).

예제

```
log.info("Successfully completed step - login.");
```

log.log(***message***, ***ex***);

`log.info`에 대한 별칭입니다.

파라미터: *message*는 로깅할 메시지이며 *ex*는 기록할 예외입니다(있는 경우).

예제

```
 log.log("Successfully completed step - login.");
```

log.warn(***message***, ***ex***);

파라미터: *message*는 로깅할 메시지이며 *ex*는 기록할 예외입니다(있는 경우).

예제

```
log.warn("Exception encountered trying to publish CloudWatch Metric.", ex);
```

SyntheticsLogHelper 클래스

`SyntheticsLogHelper` 클래스는 런타임 `syn-nodejs-puppeteer-3.2` 이상 런타임에서 사용할 수 있습니다. 이 클래스는 CloudWatch Synthetics 라이브러리에 이미 초기화되어 있으며 Synthetics 구성으로 구성되어 있습니다. 스크립트에서 이 클래스를 종속 항목으로 추가할 수 있습니다. 이 클래스를 사용하면 URL, 헤더, 오류 메시지를 제거하여 민감한 정보를 수정할 수 있습니다.

> **Note**
>
> Synthetics는 로그하는 모든 URL과 오류 메시지를 Synthetics 구성 설정인 `restrictedUrlParameters`에 따라 로그, 보고서, HAR 파일, canary 실행 오류에 포함하기 전에 제거합니다. 스크립트에서 URL 또는 오류를 로그하는 경우에만 `getSanitizedUrl` 또

는 getSanitizedErrorMessage를 사용해야 합니다. Synthetics는 스크립트에서 발생시킨 canary 오류를 제외하고는 어떠한 canary 아티팩트도 저장하지 않습니다. canary 실행 아티팩트는 고객 계정에 저장됩니다. 자세한 내용은 Synthetics canary에 대한 보안 고려 사항 단원을 참조하세요.

getSanitizedUrl(url, stepConfig = null)

이 함수는 syn-nodejs-puppeteer-3.2 이상에서 사용할 수 있습니다. 이 함수는 구성에 따라 제거된 URL 문자열을 반환합니다. restrictedUrlParameters 속성을 설정하여 암호 및 access_token과 같은 민감한 URL 파라미터를 수정하도록 선택할 수 있습니다. 기본적으로 URL의 암호는 수정됩니다. 필요한 경우 includeUrlPassword를 true로 설정하여 URL 암호를 사용하도록 설정할 수 있습니다.

전달된 URL이 유효한 URL이 아닌 경우 이 함수는 오류를 발생시킵니다.

파라미터

- *url*은 문자열이며 제거할 URL입니다.
- *stepConfig*(선택 사항)는 이 함수의 글로벌 Synthetics 구성을 재정의합니다. stepConfig가 전달되지 않으면 글로벌 구성이 URL을 제거하는 데 사용됩니다.

예

이 예에서는 샘플 URL https://example.com/learn/home?access_token=12345&token_type=Bearer&expires_in=1200을 사용합니다. 이 예에서 access_token은 로그되어서는 안 되는 민감한 정보를 포함합니다. Synthetics 서비스는 어떠한 canary 실행 아티팩트도 저장하지 않습니다. 로그, 스크린샷, 보고서와 같은 아티팩트는 모두 고객 계정의 Amazon S3 버킷에 저장됩니다.

첫 번째 단계는 Synthetics 구성을 설정하는 것입니다.

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Import Synthetics logger for logging url
const log = require('SyntheticsLogger');

// Get Synthetics configuration
```

```
const synConfig = synthetics.getConfiguration();

// Set restricted parameters
synConfig.setConfig({
   restrictedUrlParameters: ['access_token'];
});
```

다음으로, URL을 제거하고 로그합니다.

```
// Import SyntheticsLogHelper dependency
const syntheticsLogHelper = require('SyntheticsLogHelper');

const sanitizedUrl = synthetics.getSanitizedUrl('https://example.com/learn/home?
access_token=12345&token_type=Bearer&expires_in=1200');
```

그러면 canary 로그에 다음이 로그됩니다.

```
My example url is: https://example.com/learn/home?
access_token=REDACTED&token_type=Bearer&expires_in=1200
```

다음 예와 같이 Synthetics 구성 옵션이 포함된 선택적 파라미터를 전달하여 URL에 대한 Synthetics 구성을 재정의할 수 있습니다.

```
const urlConfig = {
   restrictedUrlParameters = ['*']
};
const sanitizedUrl = synthetics.getSanitizedUrl('https://example.com/learn/home?
access_token=12345&token_type=Bearer&expires_in=1200', urlConfig);
logger.info('My example url is: ' + sanitizedUrl);
```

앞의 예는 모든 쿼리 파라미터를 수정하며 다음과 같이 로그됩니다.

```
My example url is: https://example.com/learn/home?
access_token=REDACTED&token_type=REDACTED&expires_in=REDACTED
```

getSanitizedErrorMessage

이 함수는 `syn-nodejs-puppeteer-3.2` 이상에서 사용할 수 있습니다. 이 함수는 Synthetics 구성에 따라 존재하는 URL을 제거하여 제거된 오류 문자열을 반환합니다. 선택적 `stepConfig` 파라미터를 전달하여 이 함수를 호출할 때 글로벌 Synthetics 구성을 재정의하도록 선택할 수 있습니다.

파라미터

- *error*는 제거할 오류이며, Error 객체 또는 문자열일 수 있습니다.
- *stepConfig*(선택 사항)는 이 함수의 글로벌 Synthetics 구성을 재정의합니다. stepConfig가 전달되지 않으면 글로벌 구성이 URL을 제거하는 데 사용됩니다.

예

이 예에서는 `Failed to load url: https://example.com/learn/home?access_token=12345&token_type=Bearer&expires_in=1200` 오류를 사용합니다.

첫 번째 단계는 Synthetics 구성을 설정하는 것입니다.

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Import Synthetics logger for logging url
const log = require('SyntheticsLogger');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();

// Set restricted parameters
synConfig.setConfig({
   restrictedUrlParameters: ['access_token'];
});
```

다음으로, 오류 메시지를 제거하고 로그합니다.

```
// Import SyntheticsLogHelper dependency
const syntheticsLogHelper = require('SyntheticsLogHelper');

try {
   // Your code which can throw an error containing url which your script logs
} catch (error) {
    const sanitizedErrorMessage = synthetics.getSanitizedErrorMessage(errorMessage);
    logger.info(sanitizedErrorMessage);
}
```

그러면 canary 로그에 다음이 로그됩니다.

```
Failed to load url: https://example.com/learn/home?
access_token=REDACTED&token_type=Bearer&expires_in=1200
```

getSanitizedHeaders(headers, stepConfig=null)

이 함수는 `syn-nodejs-puppeteer-3.2` 이상에서 사용할 수 있습니다. 이 함수는 `syntheticsConfiguration`의 `restrictedHeaders` 속성에 따라 제거된 헤더를 반환합니다. `restrictedHeaders` 속성에 지정된 헤더는 로그, HAR 파일, 보고서에서 수정됩니다.

파라미터

- *`headers`*는 제거할 헤더가 포함된 객체입니다.
- *`stepConfig`*(선택 사항)는 이 함수의 글로벌 Synthetics 구성을 재정의합니다. `stepConfig`가 전달되지 않으면 글로벌 구성이 헤더를 제거하는 데 사용됩니다.

UI canary에만 적용되는 Node.js 라이브러리 클래스 및 함수

다음 Node.js용 CloudWatch Synthetics 라이브러리 함수는 UI canary에만 사용할 수 있습니다.

주제

- Synthetics 클래스
- BrokenLinkCheckerReport 클래스
- SyntheticsLink 클래스

Synthetics 클래스

다음 함수는 Synthetics 클래스에 있습니다.

async addUserAgent(page, userAgentString);

이 함수는 지정된 페이지의 사용자 에이전트 헤더에 *`userAgentString`*을 추가합니다.

예제

```
await synthetics.addUserAgent(page, "MyApp-1.0");
```

페이지의 사용자 에이전트 헤더가 *`browsers-user-agent-header-value`*`MyApp-1.0`으로 설정됩니다.

async executeStep(stepName, functionToExecute, [stepConfig]);

제공된 단계를 실행하며 시작/성공/실패 로깅, 시작/성공/실패 스크린샷, 성공/실패 및 기간 지표로 래핑합니다.

> **Note**
>
> `syn-nodejs-2.1` 이상 런타임을 사용하는 경우 스크린샷을 생성할지 여부 및 시기를 구성할 수 있습니다. 자세한 내용은 SyntheticsConfiguration 클래스 섹션을 참조하세요.

`executeStep` 함수는 다음 작업도 수행합니다.

- 단계가 시작되었음을 기록합니다.
- 이름이 `<stepName>-starting`인 스크린샷을 캡처합니다.
- 타이머를 시작합니다.
- 제공된 함수를 실행합니다.
- 함수가 값을 정상적으로 반환하면 성공한 것으로 간주됩니다. 함수가 오류를 생성하면 실패한 것으로 간주됩니다.
- 타이머를 종료합니다.
- 단계 성공 또는 실패 여부를 기록합니다.
- 이름이 `<stepName>-succeeded` 또는 `<stepName>-failed`인 스크린샷을 캡처합니다.
- `stepName SuccessPercent` 지표(성공 시 100, 실패 시 0)를 내보냅니다.
- 단계 시작 및 종료 시간 기준의 값을 사용하여 `stepName Duration` 지표를 내보냅니다.
- 마지막으로, `functionToExecute`에서 반환된 값을 반환하고 `functionToExecute`에서 생성된 오류를 다시 생성합니다.

canary가 `syn-nodejs-2.0` 런타임 이상을 사용하는 경우 이 함수는 canary의 보고서에 단계 실행 요약도 추가합니다. 요약에는 시작 시간, 종료 시간, 상태(PASSED/FAILED), 실패 원인(실패한 경우), 각 단계를 실행하는 동안 캡처된 스크린샷과 같은 각 단계에 관한 세부 정보가 포함됩니다.

예제

```
await synthetics.executeStep('navigateToUrl', async function (timeoutInMillis = 30000)
 {
```

```
            await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:
 timeoutInMillis});});
```

응답:

functionToExecute에서 반환된 값을 반환합니다.

syn-nodejs-2.2로 업데이트

syn-nodejs-2.2부터 선택적으로 단계 구성을 전달하여 단계 수준에서 CloudWatch Synthetics 구성을 재정의할 수 있습니다. executeStep에 전달할 수 있는 옵션 목록은 SyntheticsConfiguration 클래스 단원을 참조하세요.

다음 예에서는 continueOnStepFailure에 대한 기본 false 구성을 true로 재정의하고 스크린샷을 생성할 시기를 지정합니다.

```
var stepConfig = {
    'continueOnStepFailure': true,
    'screenshotOnStepStart': false,
    'screenshotOnStepSuccess': true,
    'screenshotOnStepFailure': false
}

await executeStep('Navigate to amazon', async function (timeoutInMillis = 30000) {
      await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:
 timeoutInMillis});
 }, stepConfig);
```

getDefaultLaunchOptions();

getDefaultLaunchOptions() 함수는 CloudWatch Synthetics에서 사용하는 브라우저 시작 옵션을 반환합니다. 자세한 내용은 Launch options type 섹션을 참조하세요.

```
// This function returns default launch options used by Synthetics.
const defaultOptions = await synthetics.getDefaultLaunchOptions();
```

getPage();

현재 열린 페이지를 Puppeteer 객체로 반환합니다. 자세한 내용은 Puppeteer API v1.14.0을 참조하세요.

예제

```
let page = synthetics.getPage();
```

응답:

현재 브라우저 세션에서 현재 열려 있는 페이지(Puppeteer 객체)입니다.

getRequestResponseLogHelper();

> **Important**
>
> `syn-nodejs-puppeteer-3.2` 런타임 이상을 사용하는 canary에서는 `RequestResponseLogHelper` 클래스와 함께 이 함수가 더 이상 사용되지 않습니다. 이 함수를 사용하면 canary 로그에 경고가 표시됩니다. 이 함수는 향후 런타임 버전에서 제거됩니다. 이 함수를 사용할 경우 대신 RequestResponseLogHelper 클래스 클래스를 사용하세요.

이 함수는 요청 및 응답 로깅 플래그를 수정하기 위한 빌더 패턴으로 사용됩니다.

예제

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper().withLogRequestHeaders(fals
```

응답:

```
{RequestResponseLogHelper}
```

launch(options)

이 함수의 옵션은 `syn-nodejs-2.1` 런타임 버전 이상에서만 사용할 수 있습니다.

이 함수는 UI canary에만 사용됩니다. 이 함수는 기존 브라우저를 닫고 새 브라우저를 시작합니다.

> **Note**
>
> CloudWatch Synthetics는 스크립트 실행을 시작하기 전에 항상 브라우저를 시작합니다. 사용자 지정 옵션으로 새 브라우저를 시작하려는 경우 외에는 launch()를 호출할 필요가 없습니다.

(options)는 브라우저에 설정할 수 있는 구성 가능한 옵션 집합입니다. 자세한 내용은 Launch options type 섹션을 참조하세요.

옵션 없이 이 함수를 호출하면 Synthetics는 기본 인수인 `executablePath` 및 `defaultViewport`를 사용하여 브라우저를 시작합니다. CloudWatch Synthetics의 기본 뷰포트는 1920 x 1080입니다.

CloudWatch Synthetics에서 사용하는 시작 파라미터를 재정의할 수 있으며 브라우저를 시작할 때 추가 파라미터를 전달할 수 있습니다. 예를 들어 다음 코드 조각에서는 기본 인수와 기본 실행 경로를 사용하여 브라우저를 시작하지만 뷰포트는 800 x 600입니다.

```
await synthetics.launch({
        defaultViewport: {
            "deviceScaleFactor": 1,
            "width": 800,
            "height": 600
    }});
```

다음 샘플 코드에서는 CloudWatch Synthetics 시작 파라미터에 새로운 `ignoreHTTPSErrors` 파라미터를 추가합니다.

```
await synthetics.launch({
        ignoreHTTPSErrors: true
 });
```

다음과 같이 CloudWatch Synthetics 시작 파라미터의 인수에 `--disable-web-security` 플래그를 추가하여 웹 보안을 사용 중지할 수 있습니다.

```
// This function adds the --disable-web-security flag to the launch parameters
const defaultOptions = await synthetics.getDefaultLaunchOptions();
const launchArgs = [...defaultOptions.args, '--disable-web-security'];
await synthetics.launch({
     args: launchArgs
  });
```

RequestResponseLogHelper 클래스

> **Important**
>
> `syn-nodejs-puppeteer-3.2` 런타임 이상을 사용하는 canary에서는 이 클래스가 더 이상 사용되지 않습니다. 이 클래스를 사용하면 canary 로그에 경고가 표시됩니다. 이 함수는 향후

런타임 버전에서 제거됩니다. 이 함수를 사용할 경우 대신 RequestResponseLogHelper 클래스 클래스를 사용하세요.

요청 및 응답 페이로드의 세분화된 구성 및 문자열 표현 생성을 처리합니다.

```
class RequestResponseLogHelper {

    constructor () {
        this.request = {url: true, resourceType: false, method: false, headers: false,
 postData: false};
        this.response = {status: true, statusText: true, url: true, remoteAddress:
 false, headers: false};
    }

    withLogRequestUrl(logRequestUrl);

    withLogRequestResourceType(logRequestResourceType);

    withLogRequestMethod(logRequestMethod);

    withLogRequestHeaders(logRequestHeaders);

    withLogRequestPostData(logRequestPostData);


    withLogResponseStatus(logResponseStatus);

    withLogResponseStatusText(logResponseStatusText);

    withLogResponseUrl(logResponseUrl);

    withLogResponseRemoteAddress(logResponseRemoteAddress);

    withLogResponseHeaders(logResponseHeaders);
```

예제

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper()
.withLogRequestPostData(true)
.withLogRequestHeaders(true)
.withLogResponseHeaders(true));
```

응답:

```
{RequestResponseLogHelper}
```

setRequestResponseLogHelper();

> **Important**
>
> `syn-nodejs-puppeteer-3.2` 런타임 이상을 사용하는 canary에서는 `RequestResponseLogHelper` 클래스와 함께 이 함수가 더 이상 사용되지 않습니다. 이 함수를 사용하면 canary 로그에 경고가 표시됩니다. 이 함수는 향후 런타임 버전에서 제거됩니다. 이 함수를 사용할 경우 대신 RequestResponseLogHelper 클래스 클래스를 사용하세요.

이 함수는 요청 및 응답 로깅 플래그를 설정하기 위한 빌더 패턴으로 사용됩니다.

예제

```
synthetics.setRequestResponseLogHelper().withLogRequestHeaders(true).withLogResponseHeaders(tru
```

응답:

```
{RequestResponseLogHelper}
```

async takeScreenshot(name, suffix);

이름 및 접미사를 사용하여 현재 페이지의 스크린샷(.PNG)을 생성합니다(선택 사항).

예제

```
await synthetics.takeScreenshot("navigateToUrl", "loaded")
```

이 예에서는 `01-navigateToUrl-loaded.png`라는 스크린샷을 캡처하여 canary의 S3 버킷에 업로드합니다.

`stepName`을 첫 번째 파라미터로 전달하여 특정 canary 단계에 대한 스크린샷을 생성할 수 있습니다. 스크린샷은 보고서의 canary 단계에 연결되어 디버깅하는 동안 각 단계를 추적하는 데 도움이 됩니다.

CloudWatch Synthetics canary는 단계 시작 전(executeStep 함수)과 단계 완료 후에 자동으로 스크린샷을 생성합니다(스크린샷을 사용 중지하도록 canary를 구성하지 않은 경우). takeScreenshot 함수에서 단계 이름을 전달하면 더 많은 스크린샷을 생성할 수 있습니다.

다음 예에서는 stepName의 값으로 signupForm을 사용하여 스크린샷을 생성합니다. 스크린샷은 이름이 02-signupForm-address가 되며 canary 보고서의 signupForm이라는 단계에 연결됩니다.

```
await synthetics.takeScreenshot('signupForm', 'address')
```

BrokenLinkCheckerReport 클래스

이 클래스는 Synthetics 링크를 추가하는 메서드를 제공합니다. syn-nodejs-2.0-beta 런타임 버전 이상을 사용하는 canary에서만 지원됩니다.

BrokenLinkCheckerReport를 사용하려면 스크립트에 다음 줄을 포함합니다.

```
const BrokenLinkCheckerReport = require('BrokenLinkCheckerReport');

const brokenLinkCheckerReport = new BrokenLinkCheckerReport();
```

유용한 함수 정의:

addLink(*syntheticsLink*, isBroken)

*syntheticsLink*는 링크를 나타내는 SyntheticsLink 객체입니다. 이 함수는 상태 코드에 따라 링크를 추가합니다. 기본적으로 상태 코드를 사용할 수 없거나 상태 코드가 400 이상인 경우 이 함수는 링크가 잘못된 것으로 간주합니다. 선택적 파라미터 isBrokenLink를 true 또는 false 값과 함께 전달하여 이 기본 동작을 재정의할 수 있습니다.

이 함수에는 반환 값이 없습니다.

getLinks()

이 함수는 잘못된 링크 검사기 보고서에 포함된 SyntheticsLink 객체의 배열을 반환합니다.

getTotalBrokenLinks()

이 함수는 잘못된 링크의 총수를 나타내는 숫자를 반환합니다.

getTotalLinksChecked()

이 함수는 보고서에 포함된 링크의 총수를 나타내는 숫자를 반환합니다.

BrokenLinkCheckerReport 사용 방법

다음 canary 스크립트 코드 조각은 링크를 탐색하여 잘못된 링크 검사기 보고서에 추가하는 예를 보여 줍니다.

1. SyntheticsLink, BrokenLinkCheckerReport, Synthetics를 가져옵니다.

```
const BrokenLinkCheckerReport = require('BrokenLinkCheckerReport');
const SyntheticsLink = require('SyntheticsLink');

// Synthetics dependency
const synthetics = require('Synthetics');
```

2. 보고서에 대한 링크를 추가하려면 BrokenLinkCheckerReport의 인스턴스를 생성합니다.

```
let brokenLinkCheckerReport = new BrokenLinkCheckerReport();
```

3. URL을 탐색하여 잘못된 링크 검사기 보고서에 추가합니다.

```
let url = "https://amazon.com";

let syntheticsLink = new SyntheticsLink(url);

// Navigate to the url.
let page = await synthetics.getPage();

// Create a new instance of Synthetics Link
let link = new SyntheticsLink(url)

try {
    const response = await page.goto(url, {waitUntil: 'domcontentloaded', timeout:
 30000});
} catch (ex) {
    // Add failure reason if navigation fails.
    link.withFailureReason(ex);
}

if (response) {
    // Capture screenshot of destination page
    let screenshotResult = await synthetics.takeScreenshot('amazon-home', 'loaded');

    // Add screenshot result to synthetics link
    link.addScreenshotResult(screenshotResult);
```

```
    // Add status code and status description to the link
    link.withStatusCode(response.status()).withStatusText(response.statusText())
}

// Add link to broken link checker report.
brokenLinkCheckerReport.addLink(link);
```

4. 보고서를 Synthetics에 추가합니다. 그러면 각 canary 실행에 대해 `BrokenLinkCheckerReport.json`이라는 JSON 파일이 S3 버킷에 생성됩니다. 스크린샷, 로그, HAR 파일과 함께 각 canary 실행에 대한 링크 보고서를 콘솔에서 확인할 수 있습니다.

```
await synthetics.addReport(brokenLinkCheckerReport);
```

SyntheticsLink 클래스

이 클래스는 정보를 래핑하는 메서드를 제공합니다. `syn-nodejs-2.0-beta` 런타임 버전 이상을 사용하는 canary에서만 지원됩니다.

`SyntheticsLink`를 사용하려면 스크립트에 다음 줄을 포함합니다.

```
const SyntheticsLink = require('SyntheticsLink');

const syntheticsLink = new SyntheticsLink("https://www.amazon.com");
```

이 함수는 `syntheticsLink`*`Object`*를 반환합니다.

유용한 함수 정의:

withUrl(***`url`***)

*`url`*은 URL 문자열입니다. 이 함수는 `syntheticsLink`*`Object`*를 반환합니다.

withText(***`text`***)

*`text`*는 앵커 텍스트를 나타내는 문자열입니다. 이 함수는 `syntheticsLink`*`Object`*를 반환합니다. 이 함수는 링크에 해당하는 앵커 텍스트를 추가합니다.

withParentUrl(***`parentUrl`***)

*`parentUrl`*은 상위(소스 페이지) URL을 나타내는 문자열입니다. 이 함수는 `syntheticsLink`*`Object`*를 반환합니다.

withStatusCode(***statusCode***)

*statusCode*는 상태 코드를 나타내는 문자열입니다. 이 함수는 syntheticsLink*Object*를 반환합니다.

withFailureReason(***failureReason***)

*failureReason*은 실패 원인을 나타내는 문자열입니다. 이 함수는 syntheticsLink*Object*를 반환합니다.

addScreenshotResult(***screenshotResult***)

*screenshotResult*는 객체이며, Synthetics 함수 takeScreenshot에 의해 반환된 ScreenshotResult의 인스턴스입니다. 객체에는 다음이 포함됩니다.

- fileName - screenshotFileName을 나타내는 문자열
- pageUrl (선택 사항)
- error (선택 사항)

API canary에만 적용되는 Node.js 라이브러리 클래스 및 함수

다음 Node.js용 CloudWatch Synthetics 라이브러리 함수는 API canary에만 사용할 수 있습니다.

주제

- executeHttpStep(stepName, requestOptions, [callback], [stepConfig])

executeHttpStep(stepName, requestOptions, [callback], [stepConfig])

제공된 HTTP 요청을 단계로 실행하고 SuccessPercent(통과 또는 실패) 및 Duration 지표를 게시합니다.

executeHttpStep은 요청에 지정된 프로토콜에 따라 내부적으로 HTTP 또는 HTTPS 네이티브 함수를 사용합니다.

이 함수는 canary의 보고서에 단계 실행 요약도 추가합니다. 요약에는 다음과 같은 각 HTTP 요청에 관한 세부 정보가 포함됩니다.

- 시작 시간
- 종료 시간
- 상태(PASSED/FAILED)

- 실패 원인(실패한 경우)
- 요청 또는 응답 헤더, 본문, 상태 코드, 상태 메시지, 성능 타이밍과 같은 HTTP 호출 세부 정보

파라미터

stepName(***String***)

단계 이름을 지정합니다. 이 이름은 이 단계의 CloudWatch 지표 게시에도 사용됩니다.

requestOptions(***Object ## String***)

이 파라미터의 값은 URL, URL 문자열 또는 객체일 수 있습니다. 객체인 경우 HTTP 요청을 수행하기 위해 구성 가능한 옵션 집합이어야 합니다. Node.js 설명서에 있는 http.request(options[, callback])의 모든 옵션을 지원합니다.

이러한 Node.js 옵션 외에도 requestOptions는 추가 파라미터인 body를 지원합니다. body 파라미터를 사용하면 데이터를 요청 본문으로 전달할 수 있습니다.

callback(***response***)

(선택 사항) HTTP 응답을 사용하여 호출되는 사용자 함수입니다. 응답은 클래스: http.IncomingMessage 유형입니다.

stepConfig(***object***)

(선택 사항) 이 파라미터를 사용하여 이 단계의 다른 구성으로 글로벌 Synthetics 구성을 재정의할 수 있습니다.

executeHttpStep 사용 예

다음 일련의 예는 서로를 기반으로 하여 이 옵션의 다양한 사용을 보여 줍니다.

이 첫 번째 예에서는 요청 파라미터를 구성합니다. URL을 [requestOptions]로 전달할 수 있습니다.

```
let requestOptions = 'https://www.amazon.com';
```

또는 다음과 같은 옵션 집합을 전달할 수 있습니다.

```
let requestOptions = {
        'hostname': 'myproductsEndpoint.com',
        'method': 'GET',
        'path': '/test/product/validProductName',
```

```
        'port': 443,
        'protocol': 'https:'
    };
```

다음 예에서는 응답을 수락하는 콜백 함수를 생성합니다. 기본적으로 [콜백(callback)]을 지정하지 않으면 CloudWatch Synthetics는 상태가 200~299(포함)인지 확인합니다.

```
// Handle validation for positive scenario
    const callback = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 200 || res.statusCode > 299) {
                throw res.statusCode + ' ' + res.statusMessage;
            }

            let responseBody = '';
            res.on('data', (d) => {
                responseBody += d;
            });

            res.on('end', () => {
                // Add validation on 'responseBody' here if required. For ex, your
 status code is 200 but data might be empty
                resolve();
            });
        });
    };
```

다음 예에서는 글로벌 CloudWatch Synthetics 구성을 재정의하는 이 단계의 구성을 생성합니다. 이 예에서 단계 구성은 보고서의 요청 헤더, 응답 헤더, 요청 본문(게시물 데이터), 응답 본문을 허용하고 X-Amz-Security-Token' 및 'Authorization' 헤더 값을 제한합니다. 기본적으로 이러한 값은 보안상의 이유로 보고서에 포함되지 않습니다. 이러한 값을 포함하도록 선택하는 경우 데이터는 S3 버킷에만 저장됩니다.

```
// By default headers, post data, and response body are not included in the report for
 security reasons.
// Change the configuration at global level or add as step configuration for individual
 steps
let stepConfig = {
    includeRequestHeaders: true,
    includeResponseHeaders: true,
    restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted header
 values do not appear in report generated.
```

```
    includeRequestBody: true,
    includeResponseBody: true
};
```

이 마지막 예에서는 요청을 executeHttpStep에 전달하고 단계의 이름을 지정합니다.

```
await synthetics.executeHttpStep('Verify GET products API', requestOptions, callback,
 stepConfig);
```

이 일련의 예를 통해 CloudWatch Synthetics는 보고서의 각 단계에서 세부 정보를 추가하고 stepName을 사용하여 각 단계의 지표를 생성합니다.

`Verify GET products API` 단계의 `successPercent` 및 `duration` 지표를 확인할 수 있습니다. API 호출 단계의 지표를 모니터링하여 API 성능을 모니터링할 수 있습니다.

이러한 함수를 사용하는 완전한 스크립트 샘플은 다단계 API canary 단원을 참조하세요.

Selenium을 사용하는 Python canary 스크립트에 사용할 수 있는 라이브러리 함수

이 단원에서는 Python canary 스크립트에 사용할 수 있는 Selenium 라이브러리 함수를 설명합니다.

주제

- 모든 canary에 적용되는 Python 및 Selenium 라이브러리 클래스 및 함수
- UI canary에만 적용되는 Python 및 Selenium 라이브러리 클래스 및 함수

모든 canary에 적용되는 Python 및 Selenium 라이브러리 클래스 및 함수

다음 Python용 CloudWatch Synthetics Selenium 라이브러리 함수는 모든 canary에 사용할 수 있습니다.

주제

- SyntheticsConfiguration 클래스
- SyntheticsLogger 클래스

SyntheticsConfiguration 클래스

SyntheticsConfiguration 클래스를 사용하여 Synthetics 라이브러리 함수의 동작을 구성할 수 있습니다. 예를 들어 이 클래스를 사용하여 스크린샷을 캡처하지 않도록 `executeStep()` 함수를 구성할 수 있습니다.

글로벌 수준에서 CloudWatch Synthetics 구성을 설정할 수 있습니다.

함수 정의:

set_config(options)

```
from aws_synthetics.common import synthetics_configuration
```

*options*는 canary에 대해 구성 가능한 옵션의 집합인 객체입니다. 다음 단원에서는 *options*의 가능한 필드를 설명합니다.

- screenshot_on_step_start(boolean) - 단계를 시작하기 전에 스크린샷을 생성할지 여부입니다.
- screenshot_on_step_success(boolean) - 성공적인 단계를 완료한 후 스크린샷을 생성할지 여부입니다.
- screenshot_on_step_failure(boolean) - 단계가 실패한 후 스크린샷을 생성할지 여부입니다.

with_screenshot_on_step_start(screenshot_on_step_start)

단계를 시작하기 전에 스크린샷을 생성할지 여부를 나타내는 부울 인수를 허용합니다.

with_screenshot_on_step_success(screenshot_on_step_success)

단계를 성공적으로 완료한 후 스크린샷을 생성할지 여부를 나타내는 부울 인수를 허용합니다.

with_screenshot_on_step_failure(screenshot_on_step_failure)

단계가 실패한 후 스크린샷을 생성할지 여부를 나타내는 부울 인수를 허용합니다.

get_screenshot_on_step_start()

단계를 시작하기 전에 스크린샷을 생성할지 여부를 반환합니다.

get_screenshot_on_step_success()

단계를 성공적으로 완료한 후 스크린샷을 생성할지 여부를 반환합니다.

get_screenshot_on_step_failure()

단계가 실패한 후 스크린샷을 생성할지 여부를 반환합니다.

disable_step_screenshots()

모든 스크린샷 옵션(get_screenshot_on_step_start, get_screenshot_on_step_success, get_screenshot_on_step_failure)을 사용 중지합니다.

enable_step_screenshots()

모든 스크린샷 옵션(get_screenshot_on_step_start, get_screenshot_on_step_success, get_screenshot_on_step_failure)을 사용하도록 설정합니다. 기본적으로 이러한 메서드는 모두 사용 설정되어 있습니다.

CloudWatch 지표에 관한 setConfig(options)

`syn-python-selenium-1.1` 이상을 사용하는 canary의 경우 setConfig의 (options)에는 canary에서 게시하는 지표를 결정하는 다음 부울 파라미터가 포함될 수 있습니다. 이러한 각 옵션의 기본값은 `true`입니다. `aggregated`로 시작하는 옵션은 `CanaryName` 측정기준 없이 지표를 내보낼지 여부를 결정합니다. 이러한 지표를 사용하여 모든 canary에 대한 집계 결과를 확인할 수 있습니다. 다른 옵션은 `CanaryName` 측정기준과 함께 지표를 내보낼지 여부를 결정합니다. 이러한 지표를 사용하여 각 개별 canary의 결과를 확인할 수 있습니다.

canary에서 내보내는 CloudWatch 지표 목록은 canary가 게시한 CloudWatch 지표 단원을 참조하세요.

- `failed_canary_metric`(boolean) - 이 canary에 대한 `Failed` 지표를 (`CanaryName` 측정기준과 함께) 내보낼지 여부입니다. 기본값은 `true`입니다.
- `failed_requests_metric`(boolean) - 이 canary에 대한 `Failed requests` 지표를 (`CanaryName` 측정기준과 함께) 내보낼지 여부입니다. 기본값은 `true`입니다.
- `2xx_metric`(boolean) - 이 canary에 대한 `2xx` 지표를 (`CanaryName` 측정기준과 함께) 내보낼지 여부입니다. 기본값은 `true`입니다.
- `4xx_metric`(boolean) - 이 canary에 대한 `4xx` 지표를 (`CanaryName` 측정기준과 함께) 내보낼지 여부입니다. 기본값은 `true`입니다.
- `5xx_metric`(boolean) - 이 canary에 대한 `5xx` 지표를 (`CanaryName` 측정기준과 함께) 내보낼지 여부입니다. 기본값은 `true`입니다.
- `step_duration_metric`(boolean) - 이 canary에 대한 `Step duration` 지표를 (`CanaryName` `StepName` 측정기준과 함께) 내보낼지 여부입니다. 기본값은 `true`입니다.
- `step_success_metric`(boolean) - 이 canary에 대한 `Step success` 지표를 (`CanaryName` `StepName` 측정기준과 함께) 내보낼지 여부입니다. 기본값은 `true`입니다.
- `aggregated_failed_canary_metric`(boolean) - 이 canary에 대한 `Failed` 지표를 (`CanaryName` 측정기준 없이) 내보낼지 여부입니다. 기본값은 `true`입니다.

- aggregated_failed_requests_metric(boolean) - 이 canary에 대한 Failed Requests 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.
- aggregated_2xx_metric(boolean) - 이 canary에 대한 2xx 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.
- aggregated_4xx_metric(boolean) - 이 canary에 대한 4xx 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.
- aggregated_5xx_metric(boolean) - 이 canary에 대한 5xx 지표를 (CanaryName 측정기준 없이) 내보낼지 여부입니다. 기본값은 true입니다.

with_2xx_metric(2xx_metric)

이 canary에 대한 CanaryName 측정기준과 함께 2xx 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_4xx_metric(4xx_metric)

이 canary에 대한 CanaryName 측정기준과 함께 4xx 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_5xx_metric(5xx_metric)

이 canary에 대한 CanaryName 측정기준과 함께 5xx 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withAggregated2xxMetric(aggregated2xxMetric)

이 canary에 대한 측정기준 없이 2xx 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

withAggregated4xxMetric(aggregated4xxMetric)

이 canary에 대한 측정기준 없이 4xx 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_aggregated_5xx_metric(aggregated_5xx_metric)

이 canary에 대한 측정기준 없이 5xx 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_aggregated_failed_canary_metric(aggregated_failed_canary_metric)

이 canary에 대한 측정기준 없이 Failed 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_aggregated_failed_requests_metric(aggregated_failed_requests_metric)

이 canary에 대한 측정기준 없이 `Failed requests` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_failed_canary_metric(failed_canary_metric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `Failed` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_failed_requests_metric(failed_requests_metric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `Failed requests` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_step_duration_metric(step_duration_metric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `Duration` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

with_step_success_metric(step_success_metric)

이 canary에 대한 `CanaryName` 측정기준과 함께 `StepSuccess` 지표를 내보낼지 여부를 지정하는 부울 인수를 허용합니다.

지표를 사용 또는 사용 중지하는 메서드

disable_aggregated_request_metrics()

canary가 `CanaryName` 측정기준 없이 내보내지는 모든 요청 지표를 내보내지 못하도록 합니다.

disable_request_metrics()

canary별 지표와 모든 canary에서 집계된 지표를 모두 포함하여 모든 요청 지표를 사용 중지합니다.

disable_step_metrics()

단계 성공 지표 및 단계 지속 시간 지표 모두를 포함하여 모든 단계 지표를 사용 중지합니다.

enable_aggregated_request_metrics()

canary가 `CanaryName` 측정기준 없이 내보내지는 모든 요청 지표를 내보낼 수 있도록 합니다.

enable_request_metrics()

canary별 지표와 모든 canary에서 집계된 지표를 모두 포함하여 모든 요청 지표를 사용하도록 설정합니다.

enable_step_metrics()

단계 성공 지표 및 단계 지속 시간 지표 모두를 포함하여 모든 단계 지표를 사용 설정합니다.

UI canary의 사용

먼저, Synthetics 종속 항목을 가져오고 구성을 가져옵니다. 다음으로, 다음 옵션 중 하나를 사용하여 setConfig 메서드를 호출함으로써 각 옵션의 구성을 설정합니다.

```
from aws_synthetics.common import synthetics_configuration

synthetics_configuration.set_config(
     {
        "screenshot_on_step_start": False,
        "screenshot_on_step_success": False,
        "screenshot_on_step_failure": True
     }
)

or
```

Or

```
synthetics_configuration.with_screenshot_on_step_start(False).with_screenshot_on_step_success(F
```

모든 스크린샷을 사용 중지하려면 이 예와 같이 disableStepScreenshots() 함수를 사용합니다.

```
synthetics_configuration.disable_step_screenshots()
```

코드의 어느 지점에서든 스크린샷을 사용 및 사용 중지할 수 있습니다. 예를 들어 한 단계에 대해서만 스크린샷을 사용 중지하려면 해당 단계를 실행하기 전에 스크린샷을 사용 중지한 다음, 이 단계가 끝나면 스크린샷을 사용하도록 설정합니다.

UI canary에 대한 set_config(options)

`syn-python-selenium-1.1`을 시작으로 UI canary의 경우 `set_config`에 다음 부울 파라미터가 포함될 수 있습니다.

- `continue_on_step_failure`(boolean) - 단계가 실패한 후 canary 스크립트를 계속 실행할지 여부입니다(단계 실패에 대해서는 executeStep 함수 참조). 단계가 실패한 경우 canary 실행은 여전히 실패로 표시됩니다. 기본값은 `false`입니다.

SyntheticsLogger 클래스

synthetics_logger는 동일한 로그 수준에서 콘솔과 로컬 로그 파일 모두에 로그를 작성합니다. 이 로그 파일은 로그 수준이 호출된 로그 함수의 원하는 로깅 수준 이하인 경우에만 두 위치에 기록됩니다.

로컬 로그 파일의 로깅 문 앞에는 호출된 함수의 로그 수준과 일치하도록 “DEBUG: “, “INFO: “등이 추가됩니다.

Amazon S3 결과 위치에 업로드되는 로그 파일을 생성할 때는 synthetics_logger를 사용할 필요가 없습니다. 대신 /tmp 폴더에 다른 로그 파일을 생성할 수 있습니다. /tmp 폴더 아래에 생성된 파일은 S3 버킷의 결과 위치에 아티팩트로 업로드됩니다.

synthetics_logger를 사용하려면

```
from aws_synthetics.common import synthetics_logger
```

유용한 함수 정의:

로그 수준 가져오기:

```
log_level = synthetics_logger.get_level()
```

로그 수준 설정:

```
synthetics_logger.set_level()
```

지정된 수준의 메시지를 로그합니다. 수준은 다음 구문 예와 같이 DEBUG, INFO, WARN 또는 ERROR일 수 있습니다.

```
synthetics_logger.debug(message, *args, **kwargs)
```

```
synthetics_logger.info(message, *args, **kwargs)
```

```
synthetics_logger.log(message, *args, **kwargs)
```

```
synthetics_logger.warn(message, *args, **kwargs)
```

```
synthetics_logger.error(message, *args, **kwargs)
```

디버그 파라미터에 대한 자세한 내용은 표준 Python 설명서의 logging.debug를 참조하세요.

이러한 로깅 함수에서 message는 메시지 형식 문자열입니다. args는 문자열 형식 지정 연산자를 사용하여 msg에 병합되는 인수입니다.

kwargs에는 다음과 같은 세 가지 키워드 인수가 있습니다.

- exc_info - false로 평가되지 않은 경우 예외 정보를 로깅 메시지에 추가합니다.
- stack_info - 기본값은 false입니다. true인 경우 실제 로깅 호출을 포함하여 스택 정보를 로깅 메시지에 추가합니다.
- extra - 사용자 정의 속성으로 로깅 이벤트에 대해 생성된 LogRecord의 __dict__를 채우는 데 사용되는 사전을 전달하는 데 사용할 수 있는 세 번째 선택적 키워드 인수입니다.

예:

DEBUG 수준의 메시지 로그:

```
synthetics_logger.debug('Starting step - login.')
```

INFO 수준의 메시지 로그(logger.log는 logger.info의 동의어임):

```
synthetics_logger.info('Successfully completed step - login.')
```

또는

```
synthetics_logger.log('Successfully completed step - login.')
```

WARN 수준의 메시지 로그:

```
synthetics_logger.warn('Warning encountered trying to publish %s', 'CloudWatch Metric')
```

ERROR 수준의 메시지 로그:

```
synthetics_logger.error('Error encountered trying to publish %s', 'CloudWatch Metric')
```

예외 로그:

```
synthetics_logger.exception(message, *args, **kwargs)
```

ERROR 수준의 메시지를 로그합니다. 예외 정보가 로깅 메시지에 추가됩니다. 예외 핸들러에서만 이 함수를 호출해야 합니다.

예외 파라미터에 대한 자세한 내용은 표준 Python 설명서의 logging.exception을 참조하세요.

message는 메시지 형식 문자열입니다. args는 문자열 형식 지정 연산자를 사용하여 msg에 병합되는 인수입니다.

kwargs에는 다음과 같은 세 가지 키워드 인수가 있습니다.

- exc_info - false로 평가되지 않은 경우 예외 정보를 로깅 메시지에 추가합니다.
- stack_info - 기본값은 false입니다. true인 경우 실제 로깅 호출을 포함하여 스택 정보를 로깅 메시지에 추가합니다.
- extra - 사용자 정의 속성으로 로깅 이벤트에 대해 생성된 LogRecord의 __dict__를 채우는 데 사용되는 사전을 전달하는 데 사용할 수 있는 세 번째 선택적 키워드 인수입니다.

예제

```
synthetics_logger.exception('Error encountered trying to publish %s', 'CloudWatch
 Metric')
```

UI canary에만 적용되는 Python 및 Selenium 라이브러리 클래스 및 함수

다음 Python용 CloudWatch Synthetics Selenium 라이브러리 함수는 UI canary에만 사용할 수 있습니다.

주제

- SyntheticsBrowser 클래스
- SyntheticsWebDriver 클래스

SyntheticsBrowser 클래스

synthetics_webdriver.Chrome()을 호출하여 브라우저 인스턴스를 생성할 때 반환되는 브라우저 인스턴스는 SyntheticsBrowser 유형입니다. SyntheticsBrowser 클래스는 ChromeDriver를 제어하고 canary 스크립트가 브라우저를 구동할 수 있도록 하여 Selenium WebDriver가 Synthetics와 함께 작동할 수 있도록 합니다.

이 클래스는 표준 Selenium 메서드 외에 다음 메서드도 제공합니다.

set_viewport_size(width, height)

브라우저의 뷰포트를 설정합니다. 예제

```
browser.set_viewport_size(1920, 1080)
```

save_screenshot(filename, suffix)

스크린샷을 /tmp 디렉터리에 저장합니다. 스크린샷은 이 디렉터리에서 S3 버킷의 canary 아티팩트 폴더로 업로드됩니다.

filename은 스크린샷의 파일 이름이며, suffix는 스크린샷의 이름을 지정하는 데 사용할 선택적 문자열입니다.

예제

```
browser.save_screenshot('loaded.png', 'page1')
```

SyntheticsWebDriver 클래스

이 클래스를 사용하려면 스크립트에서 다음을 사용합니다.

```
from aws_synthetics.selenium import synthetics_webdriver
```

add_execution_error(errorMessage, ex);

errorMessage는 오류를 설명하며, ex는 발생한 예외입니다.

add_execution_error를 사용하여 canary에 대한 실행 오류를 설정할 수 있습니다. 이 함수는 스크립트 실행을 중단하지 않고 canary에 실패합니다. 또한 successPercent 지표에 영향을 주지 않습니다.

오류가 canary 스크립트의 성공 또는 실패를 나타내는 데 중요하지 않은 경우에만 오류를 실행 오류로 추적해야 합니다.

add_execution_error의 사용 예는 다음과 같습니다. 엔드포인트의 가용성을 모니터링하고 페이지가 로드된 후에 스크린샷을 생성합니다. 스크린샷 캡처 실패가 엔드포인트의 가용성을 결정하지 않기 때문에 스크린샷을 생성하는 동안 발생한 오류를 포착하여 실행 오류로 추가할 수 있습니다. 가용성 지

표는 여전히 엔드포인트가 실행 중임을 나타내지만 canary 상태는 실패로 표시됩니다. 다음 샘플 코드 블록은 이러한 오류를 포착하여 실행 오류로 추가합니다.

```
try:
    browser.save_screenshot("loaded.png")
except Exception as ex:
   self.add_execution_error("Unable to take screenshot", ex)
```

add_user_agent(user_agent_str)

브라우저의 사용자 에이전트 헤더에 `user_agent_str` 값을 추가합니다. 브라우저 인스턴스를 생성하기 전에 `user_agent_str`을 할당해야 합니다.

예제

```
synthetics_webdriver.add_user_agent('MyApp-1.0')
```

execute_step(step_name, function_to_execute)

하나의 함수를 처리합니다. 또한 다음을 수행합니다.

- 단계가 시작되었음을 기록합니다.
- 이름이 `<stepName>-starting`인 스크린샷을 캡처합니다.
- 타이머를 시작합니다.
- 제공된 함수를 실행합니다.
- 함수가 값을 정상적으로 반환하면 성공한 것으로 간주됩니다. 함수가 오류를 생성하면 실패한 것으로 간주됩니다.
- 타이머를 종료합니다.
- 단계 성공 또는 실패 여부를 기록합니다.
- 이름이 `<stepName>-succeeded` 또는 `<stepName>-failed`인 스크린샷을 캡처합니다.
- `stepName SuccessPercent` 지표(성공 시 100, 실패 시 0)를 내보냅니다.
- 단계 시작 및 종료 시간 기준의 값을 사용하여 `stepName Duration` 지표를 내보냅니다.
- 마지막으로, `functionToExecute`에서 반환된 값을 반환하고 `functionToExecute`에서 생성된 오류를 다시 생성합니다.

예제

```
from selenium.webdriver.common.by import By

def custom_actions():
        #verify contains
        browser.find_element(By.XPATH, "//*[@id=\"id_1\"][contains(text(),'login')]")
        #click a button
        browser.find_element(By.XPATH, '//*[@id="submit"]/a').click()

    await synthetics_webdriver.execute_step("verify_click", custom_actions)
```

Chrome()

Chromium 브라우저의 인스턴스를 시작하고 브라우저의 생성된 인스턴스를 반환합니다.

예제

```
browser = synthetics_webdriver.Chrome()
browser.get("https://example.com/)
```

시크릿 모드로 브라우저를 시작하려면 다음을 사용하세요.

```
add_argument('—-incognito')
```

프록시 설정을 추가하려면 다음을 사용하세요.

```
add_argument('--proxy-server=%s' % PROXY)
```

예제

```
from selenium.webdriver.chrome.options import Options
chrome_options = Options()
chrome_options.add_argument("—-incognito")
browser = syn_webdriver.Chrome(chrome_options=chrome_options)
```

cron을 사용하여 canary 실행 예약

cron 표현식을 사용하면 canary 일정을 계획할 때 유연하게 설정할 수 있습니다. cron 표현식에는 다음 표에 나열된 순서대로 5개 또는 6개의 필드가 있습니다. 필드는 공백으로 구분됩니다. 구문은 canary를 생성하는 데 CloudWatch 콘솔을 사용하는지, AWS CLI 또는 AWS SDK를 사용하는지에 따라 달라집

니다. 콘솔을 사용할 경우 처음 5개의 필드만 지정합니다. AWS CLI 또는 AWS SDK를 사용할 경우 6개 필드를 모두 지정하고 Year 필드에 *를 지정해야 합니다.

필드	허용된 값	허용되는 특수 문자
분	0~59	, - * /
시간	0~23	, - * /
날짜	1~31	, - * ? / L W
월	1-12 또는 JAN-DEC	, - * /
요일	1-7 또는 SUN-SAT	, - * ? L #
연도	*	

특수 문자

- ,(쉼표)를 사용하면 필드의 표현식에 여러 값을 포함할 수 있습니다. 예를 들어 Month 필드에서 JAN,FEB,MAR는 1월, 2월, 3월을 포함한다는 의미입니다.
- -(대시) 특수 문자로 범위를 지정할 수 있습니다. 예컨대, Day 필드에서 1-15는 지정된 달의 1일에서 15일까지 포함한다는 의미입니다.
- *(별표) 특수 문자로 필드에 모든 값을 포함할 수 있습니다. Hours 필드에서 *는 모든 시간을 포함한다는 의미입니다. 동일한 표현식의 Day-of-month 필드와 Day-of-week 필드 모두에 *를 사용할 수 없습니다. 필드 중 하나에 사용할 경우 다른 하나에는 반드시 ?를 사용해야 합니다.
- /(슬래시)로 증분을 지정할 수 있습니다. Minutes 필드에 1/10을 입력하면 지정한 시간의 1분부터 시작해서 매 10분 간격을 지정할 수 있습니다(예: 11분, 21분, 31분 등).
- ?(물음표)로 하나 또는 다른 하나를 지정할 수 있습니다. Day-of-month 필드에 7을 입력하면서 7일이 무슨 요일이든 상관없는 경우 Day-of-week 필드에 ?를 입력할 수 있습니다.
- '날짜' 또는 '요일' 필드에서 L 와일드카드로 해당 월 또는 주의 마지막 날을 지정할 수 있습니다.
- '날짜' 필드에서는 **W** 와일드카드로 어떤 한 평일을 지정할 수 있습니다. 예를 들어 '날짜' 필드에 **3W**를 입력하면 해당 월의 세 번째 평일에 가장 가까운 날을 지정할 수 있습니다.
- '요일' 필드의 # 와일드카드는 그 달에 속한 정해진 요일의 특정 인스턴스를 지정합니다. 예를 들어 3#2는 그 달의 두 번째 화요일입니다. 3은 각 주의 셋째 날이므로 화요일을 나타내며 2는 그 달에 속한 해당 유형의 두 번째 날을 나타냅니다.

제한 사항

- 같은 cron 표현식에서 '날짜' 및 '요일' 필드를 지정할 수 없습니다. 필드 중 하나에 값 또는 *(별표)를 지정하는 경우 다른 필드에는 ?(물음표)를 사용해야 합니다.
- 1분보다 빠른 속도로 이어지는 cron 표현식은 지원되지 않습니다.
- 실행되기 전에 1년 넘게 기다리도록 canary를 설정할 수 없으므로 `Year` 필드에 *만 지정할 수 있습니다.

예제

canary를 생성할 때 다음 샘플 cron 문자열을 참조할 수 있습니다. 다음 예는 AWS CLI 또는 AWS SDK를 사용하여 canary를 생성하거나 업데이트하기 위한 올바른 구문입니다. CloudWatch 콘솔을 사용하는 경우 각 예에서 마지막 *를 생략합니다.

표현식	의미
`0 10 * * ? *`	매일 오전 10시(UTC)에 실행
`15 12 * * ? *`	매일 오후 12시 15분(UTC)에 실행
`0 18 ? * MON-FRI *`	매주 월요일부터 금요일까지 오후 6시(UTC)에 실행
`0 8 1 * ? *`	매월 1일 오전 8시(UTC)에 실행
`0/10 * ? * MON-SAT *`	매주 월요일부터 토요일까지 10분마다 실행
`0/5 8-17 ? * MON-FRI *`	월요일부터 금요일까지 오전 8시부터 오후 5시 55분(UTC) 사이에 5분마다 실행

그룹

그룹을 만들어 교차 리전 canary를 포함하여 canary를 서로 연결할 수 있습니다. 그룹을 사용하면 canary를 관리하고 자동화할 수 있으며 그룹의 모든 canary에 대한 집계된 실행 결과 및 통계를 볼 수도 있습니다.

그룹은 전역 리소스입니다. 그룹을 만들면 그룹을 지원하는 모든 AWS 리전에 복제됩니다. 이러한 리전 중 하나의 canary를 그룹에 추가하고 해당 리전 중 하나에서 볼 수 있습니다. 그룹 ARN 형식은 생성된 리전 이름을 반영하지만 그룹의 사용 범위는 특정 리전으로만 제한되지 않습니다. 즉, 여러 리전의

canary를 동일한 그룹에 배치한 다음 해당 그룹을 사용하여 모든 canary를 단일 보기에서 보고 관리할 수 있습니다.

그룹은 기본적으로 비활성화된 지역을 제외한 모든 리전에서 지원됩니다. 리전에 대한 자세한 내용은 리전 활성화를 참조하세요.

각 그룹에는 최대 10개의 canary가 포함될 수 있습니다. 계정에는 최대 20개 그룹이 있을 수 있습니다. 한 canary는 최대 10개 그룹의 멤버일 수 있습니다.

그룹을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Synthetics canary를 선택합니다.
3. 그룹 생성을 선택합니다.
4. 그룹 이름에 그룹 이름을 입력합니다.
5. 이 그룹과 연결할 canary를 선택합니다. canary를 선택하려면 정확한 canary 이름에 전체 이름을 입력하고 검색을 선택합니다. canary 이름 옆의 확인란을 선택합니다. 다른 리전에 이름이 같은 canary가 여러 개 있는 경우 원하는 canary를 선택해야 합니다.

   이 단계를 반복하여 최대 10개의 canary를 그룹과 연결할 수 있습니다.
6. (선택 사항) 태그에서 이 그룹에 대한 태그로 하나 이상의 키-값 페어를 추가합니다. 태그를 사용하면 AWS 리소스를 식별 및 구성하고 AWS 비용을 추적할 수 있습니다. 자세한 내용은 Amazon CloudWatch 리소스 태그 지정 단원을 참조하십시오.
7. 그룹 생성을 선택합니다.

로컬로 canary 테스트

이 섹션에서는 Microsoft Visual Studio 코드 편집기 또는 JetBrains IDE 코드 편집기 내에서 직접 CloudWatch Synthetics canary를 수정, 테스트 및 디버깅하는 방법을 설명합니다. 로컬 디버깅 환경은 Serverless Application Model(SAM) 컨테이너를 통해 Lambda 함수를 시뮬레이션하여 Synthetics canary의 동작을 에뮬레이션합니다.

> **Note**
>
> 시각적 모니터링에 의존하는 canary를 로컬로 디버깅하는 작업은 비현실적입니다. 시각적 모니터링은 초기 실행 중에 기본 스크린샷을 캡처한 다음, 이러한 스크린샷을 후속 실행의 스크

린샷과 비교하는 방식에 의존합니다. 로컬 개발 환경에서는 실행이 저장되거나 추적되지 않으며, 각 반복은 종속되지 않은 독립형 실행입니다. canary 실행 기록이 없기 때문에 실제로 시각적 모니터링에 의존하는 canary를 디버깅할 수 없습니다.

사전 조건

1. HAR 파일 및 스크린샷과 같은 로컬 canary 테스트 실행에서 발생한 아티팩트를 저장하는 데 사용할 Amazon S3 버킷을 선택하거나 생성합니다. 이를 위해서는 IAM으로 프로비저닝해야 합니다. Amazon S3 버킷 설정을 건너뛰더라도 로컬에서 canary를 테스트할 수는 있지만, 누락된 버킷에 대한 오류 메시지가 표시되고 canary 아티팩트에 액세스할 수 없습니다.

   Amazon S3 버킷을 사용하는 경우 비용을 절감하려면 며칠 후에 객체를 삭제하도록 버킷 수명 주기를 설정하는 것이 좋습니다. 자세한 내용은 스토리지 수명 주기 관리를 참조하세요.
2. AWS 계정의 기본 AWS 프로필을 설정합니다. 자세한 내용은 Configuration and credential file settings를 참조하세요.
3. 디버그 환경의 기본 AWS 리전을 원하는 리전(예: `us-west-2`)으로 설정합니다.
4. AWS SAM CLI를 설치합니다. 자세한 내용은 AWS SAM CLI 설치 단원을 참조하세요.
5. Visual Studio Code Editor 또는 JetBrains IDE를 설치합니다. 자세한 내용은 Visual Studio Code 또는 JetBrains IDE 섹션을 참조하세요.
6. AWS SAM CLI를 사용하여 작업하려면 Docker를 설치합니다. docker 대몬을 시작해야 합니다. 자세한 내용은 Installing Docker to use with the AWS SAM CLI를 참조하세요.

   또는 Docker 런타임을 사용하는 경우 Rancher 등의 다른 컨테이너 관리 소프트웨어를 설치할 수 있습니다.
7. 원하는 편집기에 대한 AWS 툴킷 확장을 설치합니다. 자세한 내용은 Installing the AWS Toolkit for Visual Studio Code 또는 AWS Toolkit for JetBrains 설치를 참조하세요.

주제

- 테스트 및 디버깅 환경 설정
- Visual Studio Code IDE 사용
- JetBrains IDE 사용
- SAM CLI를 사용하여 로컬로 canary 실행
- 로컬 테스트 환경을 기존 canary 패키지에 통합

- CloudWatch Synthetics 런타임 변경
- 일반적인 오류

테스트 및 디버깅 환경 설정

먼저 다음 명령을 입력하여 AWS에서 제공하는 Github 리포지토리를 복제합니다. 리포지토리에는 Node.js canary 및 Python canary 모두에 대한 코드 샘플이 포함되어 있습니다.

```
git clone https://github.com/aws-samples/synthetics-canary-local-debugging-sample.git
```

그런 다음, canary 언어에 따라 다음 중 하나를 수행합니다.

Node.js canary의 경우

1. 다음 명령을 입력하여 Node.js canary 소스 디렉터리로 이동합니다.

   ```
   cd synthetics-canary-local-debugging-sample/nodejs-canary/src
   ```

2. 다음 명령을 입력하여 canary 종속성을 설치합니다.

   ```
   npm install
   ```

Python canary의 경우

1. 다음 명령을 입력하여 Python canary 소스 디렉터리로 이동합니다.

   ```
   cd synthetics-canary-local-debugging-sample/python-canary/src
   ```

2. 다음 명령을 입력하여 canary 종속성을 설치합니다.

   ```
   pip3 install -r requirements.txt -t .
   ```

Visual Studio Code IDE 사용

Visual Studio 시작 구성 파일은 `.vscode/launch.json`에 있습니다. 여기에는 Visual Studio 코드에서 템플릿 파일을 검색할 수 있는 구성이 포함되어 있습니다. canary를 간접 호출하는 데 필요한 파라미터와 함께 Lambda 페이로드를 정의합니다. 다음은 Node.js canary의 시작 구성입니다.

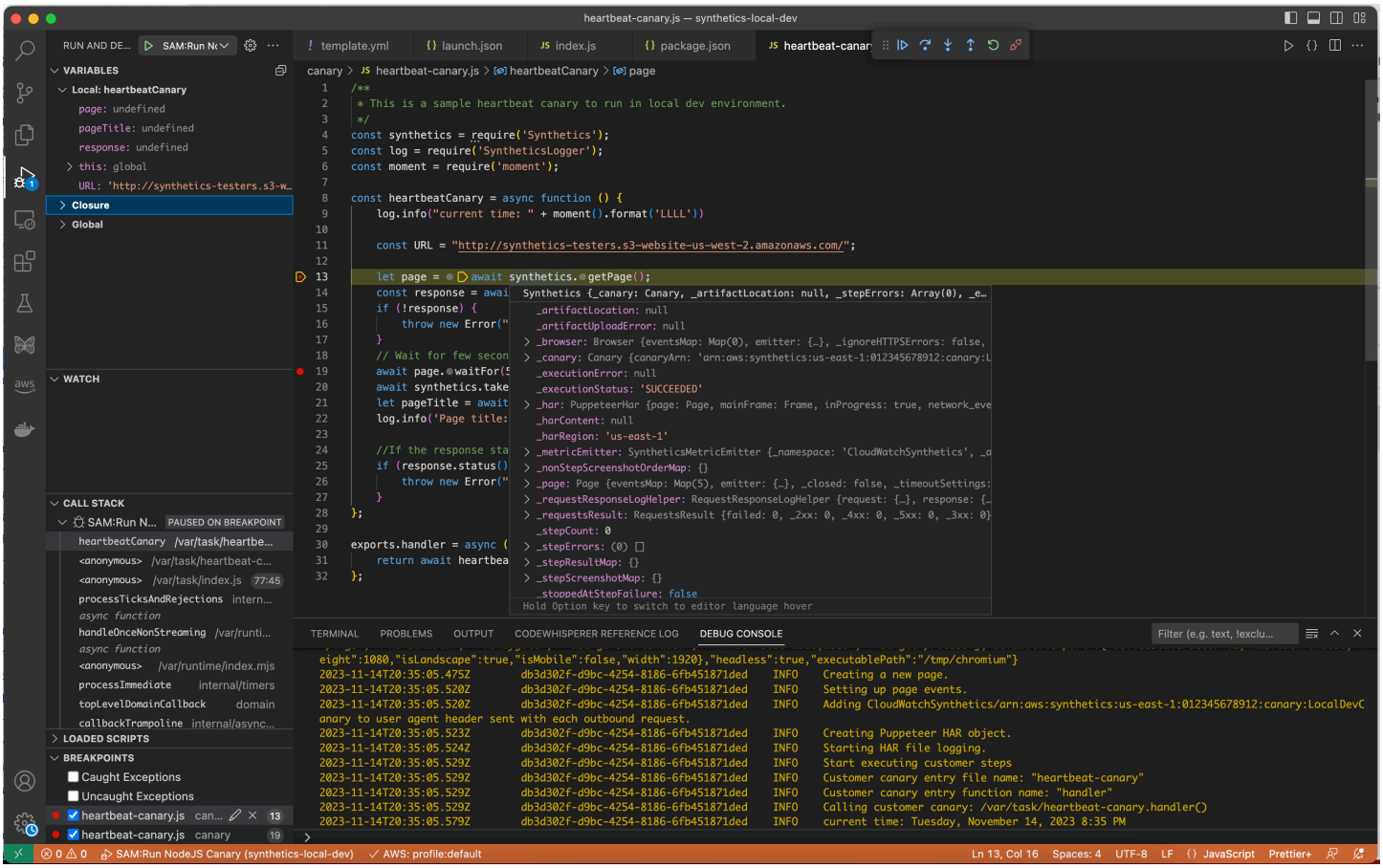
```
{
            ...
            ...
            "lambda": {
                "payload": {
                    "json": {
                        // Canary name. Provide any name you like.
                        "canaryName": "LocalSyntheticsCanary",
                        // Canary artifact location
                        "artifactS3Location": {
                            "s3Bucket": "cw-syn-results-123456789012-us-west-2",
                            "s3Key": "local-run-artifacts",
                        },
                        // Your canary handler name
                        "customerCanaryHandlerName": "heartbeat-canary.handler"
                    }
                },
                // Environment variables to pass to the canary code
                "environmentVariables": {}
            }
        }
    ]
}
```

선택적으로 페이로드 JSON에서 다음 필드를 제공할 수도 있습니다.

- `s3EncryptionMode` 유효한 값: `SSE_S3` | `SSE_KMS`
- `s3KmsKeyArn` 유효한 값: *KMS Key ARN*
- `activeTracing` 유효한 값: `true` | `false`
- `canaryRunId` 유효한 값: *UUID* 활성 추적이 활성화된 경우 이 파라미터가 필요합니다.

Visual Studio에서 canary를 디버깅하려면 canary 코드에 실행을 일시 중지하려는 중단점을 추가합니다. 중단점을 추가하려면 편집기 여백을 선택하고 편집기에서 실행 및 디버그 모드로 이동합니다. 재생 버튼을 클릭하여 canary를 실행합니다. canary가 실행되면 디버그 콘솔에 로그가 맞춤 조정되어 canary의 동작에 대한 실시간 인사이트를 제공합니다. 중단점을 추가하면 각 중단점에서 canary 실행이 일시 중지되므로 단계별로 코드를 실행하고 변수 값, 인스턴스 메서드, 객체 속성 및 함수 직접 호출 스택을 검사할 수 있습니다.

로컬로 canary를 실행하고 디버깅하는 데는 비용이 발생하지 않습니다. 단, Amazon S3 버킷에 저장된 아티팩트와 각 로컬 실행에서 생성되는 CloudWatch 지표는 예외입니다.



JetBrains IDE 사용

AWS Toolkit for JetBrains 확장을 설치한 후 Node.js canary를 디버깅하는 경우 Node.js 플러그인 및 JavaScript 디버거가 실행되도록 활성화되었는지 확인합니다. 방법은 다음과 같습니다.

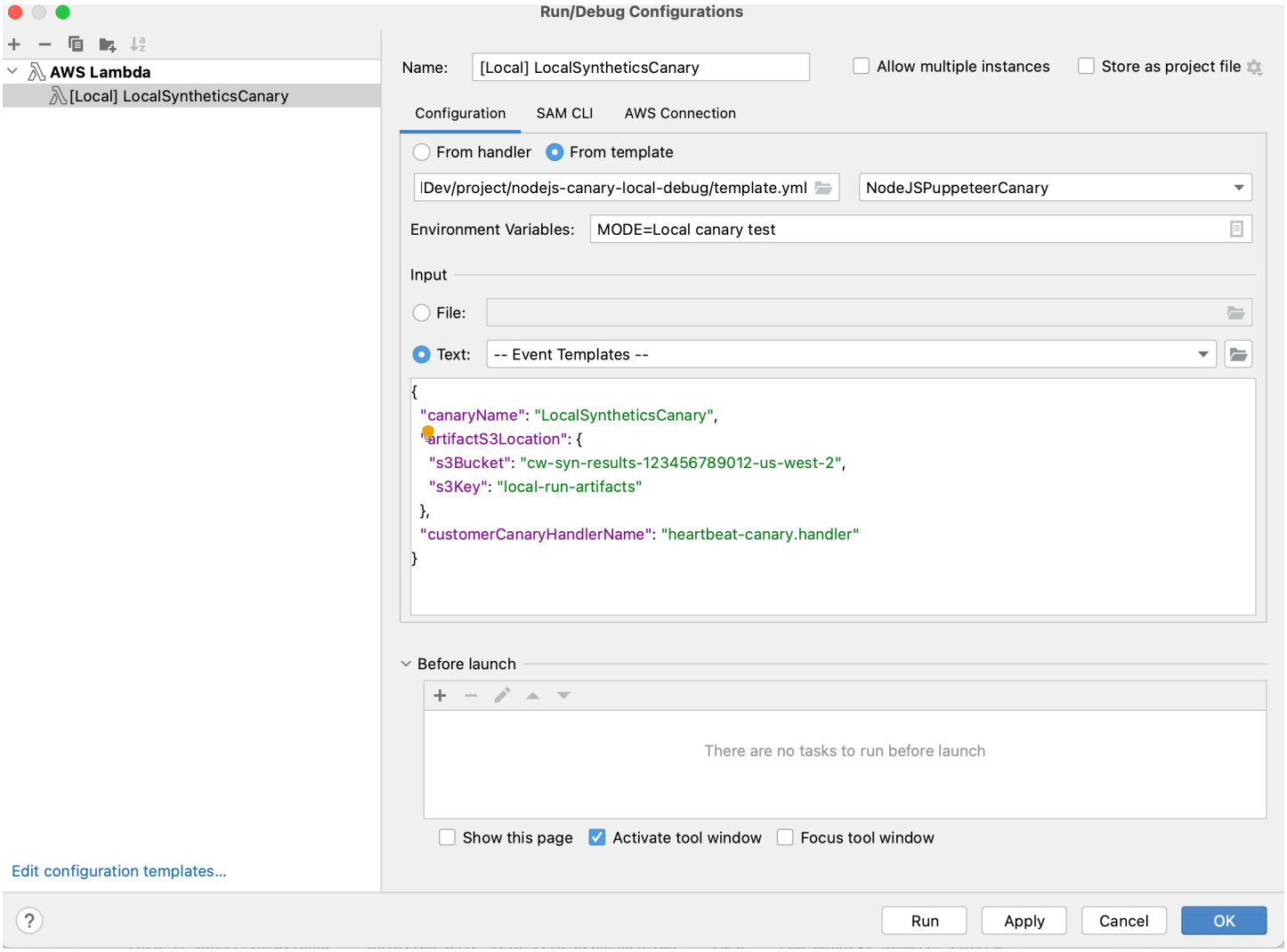
JetBrains IDE를 사용하여 canary 디버깅

1. JetBrains IDE의 왼쪽 탐색 창에서 Lambda를 선택하고 로컬 구성 템플릿을 선택합니다.
2. 실행 구성 이름(예: **`LocalSyntheticsCanary`**)을 입력합니다.
3. 템플릿에서를 선택하고 템플릿 필드에서 파일 브라우저를 선택한 다음, 프로젝트의 nodejs 디렉터리 또는 python 디렉터리에서 template.yml 파일을 선택합니다.
4. 입력 섹션에서 다음 화면과 같이 canary의 페이로드를 입력합니다.

```
{
 "canaryName": "LocalSyntheticsCanary",
 "artifactS3Location": {
     "s3Bucket": "cw-syn-results-123456789012-us-west-2",
     "s3Key": "local-run-artifacts"
```

```
  },
  "customerCanaryHandlerName": "heartbeat-canary.handler"
}
```

Visual Studio Code IDE 사용에 나열된 대로 페이로드 JSON에서 다른 환경 변수를 설정할 수도 있습니다.



SAM CLI를 사용하여 로컬로 canary 실행

Serverless Application Model(SAM) CLI를 사용하여 로컬로 canary를 실행하려면 다음 절차 중 하나를 사용합니다. `event.json`에서 `s3Bucket`에 대한 Amazon S3 버킷 이름을 지정해야 합니다.

SAM CLI를 사용하여 Node.js canary를 실행하는 방법

1. 다음 명령을 입력하여 소스 디렉터리로 이동합니다.

```
cd synthetics-canary-local-debugging-sample/nodejs-canary
```

2. 다음 명령을 입력합니다.

```
sam build
sam local invoke -e ../event.json
```

SAM CLI를 사용하여 Python canary를 실행하는 방법

1. 다음 명령을 입력하여 소스 디렉터리로 이동합니다.

```
cd synthetics-canary-local-debugging-sample/python-canary
```

2. 다음 명령을 입력합니다.

```
sam build
sam local invoke -e ../event.json
```

로컬 테스트 환경을 기존 canary 패키지에 통합

다음 세 개의 파일을 복사하여 로컬 canary 디버깅을 기존 canary 패키지에 통합할 수 있습니다.

- `template.yml` 파일을 canary 패키지 루트에 복사합니다. canary 코드가 있는 디렉터리를 가리키도록 `CodeUri`의 경로를 수정해야 합니다.
- Node.js canary에 대한 작업을 수행하는 경우 `cw-synthetics.js` 파일을 canary 소스 디렉터리에 복사합니다. Python canary에 대한 작업을 수행하는 경우 `cw-synthetics.py`를 canary 소스 디렉터리에 복사합니다.
- 시작 구성 파일(`.vscode/launch.json`)을 패키지 루트에 복사합니다. `.vscode` 디렉터리에 넣어야 합니다. 해당 위치가 아직 없으면 새로 생성합니다.

CloudWatch Synthetics 런타임 변경

디버깅의 일환으로 최신 런타임 대신, 다른 CloudWatch Synthetics 런타임을 사용하여 canary를 실행할 수도 있습니다. 이를 수행하려면 다음 테이블 중 하나에서 사용하려는 런타임을 찾습니다. 올바른 리전의 런타임을 선택해야 합니다. 그런 다음, 해당 런타임의 ARN을 `template.yml` 파일의 적절한 위치에 붙여넣고 canary를 실행합니다.

Node.js 런타임

syn-nodejs-puppeteer-7.0용 ARN

다음 테이블에는 해당 기능을 사용할 수 있는 각 AWS 리전에서 CloudWatch Synthetics 런타임의 `syn-nodejs-puppeteer-7.0` 버전에 대해 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	`arn:aws:lambda:us-east-1:378653112637:layer:Synthetics:44`
미국 동부(오하이오)	`arn:aws:lambda:us-east-2:772927465453:layer:Synthetics:46`
미국 서부(캘리포니아 북부)	`arn:aws:lambda:us-west-1:332033056316:layer:Synthetics:44`
미국 서부(오레곤)	`arn:aws:lambda:us-west-2:760325925879:layer:Synthetics:47`
아프리카(케이프타운)	`arn:aws:lambda:af-south-1:461844272066:layer:Synthetics:44`
아시아 태평양(홍콩)	`arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics:45`
아시아 태평양(하이데라바드)	`arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics:20`
아시아 태평양(자카르타)	`arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics:26`

지역	ARN
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics:18
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics:44
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics:30
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics:46
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics:49
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics:44
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics:44
캐나다(중부)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics:44
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics:76
중국(베이징)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics:45
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics:46
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics:44
유럽(아일랜드)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics:46

지역	ARN
유럽(런던)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics:44
유럽(밀라노)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics:45
유럽(파리)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics:44
유럽(스페인)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics:20
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics:44
유럽(취리히)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics:19
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics:17
중동(바레인)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics:44
중동(UAE)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics:19
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics:45
AWS GovCloud(미국 동부)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics:41
AWS GovCloud(미국 서부)	arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics:42

syn-nodejs-puppeteer-6.2용 ARN

다음 테이블에는 해당 기능을 사용할 수 있는 각 AWS 리전에서 CloudWatch Synthetics 런타임의 syn-nodejs-puppeteer-6.2 버전에 대해 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:378653112637:layer:Synthetics:41
미국 동부(오하이오)	arn:aws:lambda:us-east-2:772927465453:layer:Synthetics:43
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:332033056316:layer:Synthetics:41
미국 서부(오레곤)	arn:aws:lambda:us-west-2:760325925879:layer:Synthetics:44
아프리카(케이프타운)	arn:aws:lambda:af-south-1:461844272066:layer:Synthetics:41
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics:42
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics:17
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics:23
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics:15
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics:41
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics:27

지역	ARN
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics:42
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics:46
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics:41
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics:41
캐나다(중부)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics:41
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics:73
중국(베이징)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics:42
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics:43
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics:41
유럽(아일랜드)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics:43
유럽(런던)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics:41
유럽(밀라노)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics:42
유럽(파리)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics:41

지역	ARN
유럽(스페인)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics:17
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics:41
유럽(취리히)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics:16
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics:14
중동(바레인)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics:41
중동(UAE)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics:16
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics:42
AWS GovCloud(미국 동부)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics:39
AWS GovCloud(미국 서부)	arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics:39

syn-nodejs-puppeteer-5.2용 ARN

다음 테이블에는 해당 기능을 사용할 수 있는 각 AWS 리전에서 CloudWatch Synthetics 런타임의 `syn-nodejs-puppeteer-5.2` 버전에 대해 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:378653112637:layer:Synthetics:42

지역	ARN
미국 동부(오하이오)	arn:aws:lambda:us-east-2:772927465453:layer:Synthetics:44
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:332033056316:layer:Synthetics:42
미국 서부(오레곤)	arn:aws:lambda:us-west-2:760325925879:layer:Synthetics:45
아프리카(케이프타운)	arn:aws:lambda:af-south-1:461844272066:layer:Synthetics:42
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics:43
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics:18
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics:24
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics:16
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics:42
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics:28
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics:44
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics:47
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics:42

지역	ARN
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics:42
캐나다(중부)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics:42
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics:74
중국(베이징)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics:43
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics:44
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics:42
유럽(아일랜드)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics:44
유럽(런던)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics:42
유럽(밀라노)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics:43
유럽(파리)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics:42
유럽(스페인)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics:18
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics:42
유럽(취리히)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics:17

지역	ARN
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics:15
중동(바레인)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics:42
중동(UAE)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics:17
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics:43
AWS GovCloud(미국 동부)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics:40
AWS GovCloud(미국 서부)	arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics:40

Python 런타임

syn-python-selenium-3.0용 ARN

다음 테이블에는 해당 기능을 사용할 수 있는 각 AWS 리전에서 CloudWatch Synthetics 런타임의 `syn-python-selenium-3.0` 버전에 대해 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	aarn:aws:lambda:us-east-1:378653112637:layer:Synthetics_Selenium:32
미국 동부(오하이오)	arn:aws:lambda:us-east-2:772927465453:layer:Synthetics_Selenium:34
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:332033056316:layer:Synthetics_Selenium:32

지역	ARN
미국 서부(오레곤)	arn:aws:lambda:us-west-2:760325925879:layer:Synthetics_Selenium:34
아프리카(케이프타운)	arn:aws:lambda:af-south-1:461844272066:layer:Synthetics_Selenium:32
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics_Selenium:32
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics_Selenium:20
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics_Selenium:26
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics_Selenium:18
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics_Selenium:32
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics_Selenium:30
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics_Selenium:34
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics_Selenium:37
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics_Selenium:32
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics_Selenium:32
캐나다(중부)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics_Selenium:32

지역	ARN
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics_Selenium:76
중국(베이징)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics_Selenium:32
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics_Selenium:32
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics_Selenium:32
유럽(아일랜드)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics_Selenium:34
유럽(런던)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics_Selenium:32
유럽(밀라노)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics_Selenium:33
유럽(파리)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics_Selenium:32
유럽(스페인)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics_Selenium:20
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics_Selenium:32
유럽(취리히)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics_Selenium:19
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics_Selenium:17
중동(바레인)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics_Selenium:32

지역	ARN
중동(UAE)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics_Selenium:19
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics_Selenium:33
AWS GovCloud(미국 동부)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics_Selenium:30
AWS GovCloud(미국 서부)	arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics_Selenium:31

syn-python-selenium-2.1용 ARN

다음 테이블에는 해당 기능을 사용할 수 있는 각 AWS 리전에서 CloudWatch Synthetics 런타임의 `syn-python-selenium-2.1` 버전에 대해 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:378653112637:layer:Synthetics:29
미국 동부(오하이오)	arn:aws:lambda:us-east-2:772927465453:layer:Synthetics:31
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:332033056316:layer:Synthetics:29
미국 서부(오레곤)	arn:aws:lambda:us-west-2:760325925879:layer:Synthetics:31
아프리카(케이프타운)	arn:aws:lambda:af-south-1:461844272066:layer:Synthetics:29
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics:29

지역	ARN
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics:17
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics:23
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics:15
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics:29
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics:27
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics:30
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics:34
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics:29
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics:29
캐나다(중부)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics:29
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics:73
중국(베이징)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics:29
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics:29

지역	ARN
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics:29
유럽(아일랜드)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics:31
유럽(런던)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics:29
유럽(밀라노)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics:30
유럽(파리)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics:29
유럽(스페인)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics:17
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics:29
유럽(취리히)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics:16
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics:14
중동(바레인)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics:29
중동(UAE)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics:16
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics:30
AWS GovCloud(미국 동부)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics:29

지역	ARN
AWS GovCloud(미국 서부)	`arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics:29`

일반적인 오류

오류: AWS SAM 프로젝트를 로컬로 실행하려면 Docker가 필요합니다. Docker를 설치하고 실행했나요?

컴퓨터에서 Docker를 시작해야 합니다.

SAM 로컬 간접 호출 실패: GetLayerVersion 작업을 직접 호출하는 동안 오류(ExpiredTokenException) 발생: 요청에 포함된 보안 토큰이 만료됨

AWS 기본 프로필이 설정되었는지 확인합니다.

보다 일반적인 오류

SAM의 일반적인 오류에 대한 자세한 내용은 AWS SAM CLI troubleshooting을 참조하세요.

실패한 canary 문제 해결

canary가 실패하면 문제 해결을 위해 다음을 확인합니다.

일반 문제 해결

- canary 세부 정보 페이지에서 자세한 정보를 확인할 수 있습니다. CloudWatch 콘솔의 탐색 창에서 [Canaries]를 선택한 다음, canary 이름을 선택하여 canary 세부 정보 페이지를 엽니다. [가용성(Availability)] 탭에서 [SuccessPercent] 지표를 확인하여 문제가 지속적인지 또는 간헐적인지 확인합니다.

  계속해서 [가용성(Availability)] 탭에서 실패한 데이터 요소를 선택하여 실패한 해당 실행에 대한 스크린샷, 로그, 단계 보고서(사용 가능한 경우)를 확인합니다.

  단계가 스크립트의 일부이므로 단계 보고서를 사용할 수 있는 경우 어떤 단계가 실패했는지 확인하고 관련 스크린샷을 살펴봄으로써 고객에게 표시되는 문제를 확인합니다.

  또한 HAR 파일을 확인하여 요청이 하나 이상 실패하는지 확인할 수도 있습니다. 로그를 사용하여 실패한 요청 및 오류를 드릴다운함으로써 더 심층적으로 분석할 수 있습니다. 마지막으로, 이러한 아티팩트를 성공적인 canary 실행의 아티팩트와 비교하여 문제를 정확하게 파악할 수 있습니다.

기본적으로 CloudWatch Synthetics는 UI canary의 각 단계에 대해 스크린샷을 캡처합니다. 그러나 스크립트가 스크린샷을 사용 중지하도록 구성되었을 수 있습니다. 디버깅하는 동안 스크린샷을 다시 사용하도록 설정할 수 있습니다. 마찬가지로 API canary의 경우 디버깅하는 동안 HTTP 요청 및 응답의 헤더 및 본문을 확인할 수 있습니다. 보고서에 이 데이터를 포함하는 방법에 대한 자세한 내용은 executeHttpStep(stepName, requestOptions, [callback], [stepConfig]) 단원을 참조하세요.

- 애플리케이션에 대한 최근 배포가 있는 경우 배포를 롤백한 다음, 나중에 디버깅합니다.
- 엔드포인트에 수동으로 연결하여 동일한 문제를 재현할 수 있는지 확인합니다.

주제

Lambda 환경 업데이트 후 canary 실패

CloudWatch Synthetics canary는 사용자 계정에서 Lambda 함수로 구현됩니다. 이러한 Lambda 함수는 보안 업데이트, 버그 수정 및 기타 개선 사항이 포함된 정규 Lambda 런타임 업데이트에 따라 달라집니다. Lambda는 기존 함수와 역호환되는 런타임 업데이트를 제공하기 위해 노력합니다. 하지만 소프트웨어 패치와 마찬가지로, 드물지만 런타임 업데이트가 기존 함수에 부정적인 영향을 미칠 수 있는 경우도 있습니다. canary가 Lambda 런타임 업데이트의 영향을 받았다고 생각되는 경우 (지원되는 리전에서) Lambda 런타임 관리 수동 모드를 사용하여 Lambda 런타임 버전을 일시적으로 롤백할 수 있습니다. 이렇게 하면 canary 함수가 계속 작동하고 중단을 최소화하여 최신 런타임 버전으로 돌아가기 전에 비호환성을 해결할 시간을 확보할 수 있습니다.

Lambda 런타임 업데이트 후 canary가 실패하는 경우 가장 좋은 해결책은 최신 Synthetics 런타임 중 하나로 업그레이드하는 것입니다. 최신 런타임에 대한 자세한 내용은 Synthetics 런타임 버전 섹션을 참조하세요.

Lambda 런타임 관리 제어가 제공되는 리전에서는 런타임 관리 제어를 위한 수동 모드를 사용하여 canary를 이전 Lambda 관리형 런타임으로 되돌릴 수도 있습니다. 다음 섹션의 아래 단계를 따라 AWS CLI 또는 Lambda 콘솔을 사용하여 수동 모드를 설정할 수 있습니다.

> **Warning**
>
> 런타임 설정을 수동 모드로 변경하면 Lambda 함수는 자동 모드로 되돌아갈 때까지 자동 보안 업데이트를 수신하지 않습니다. 이 기간 동안 Lambda 함수의 보안이 취약해질 수 있습니다.

사전 조건

- jq 설치
- AWS CLI의 최신 버전을 설치합니다. 자세한 내용은 AWS CLI 설치 및 업데이트 지침을 참조하세요.

1단계: Lambda 함수 ARN 가져오기

다음 명령을 실행하여 응답에서 `EngineArn` 필드를 가져옵니다. 이 `EngineArn`은 canary와 관련된 Lambda 함수의 ARN입니다. 다음 단계에서 이 ARN을 사용합니다.

```
aws synthetics get-canary --name my-canary | jq '.Canary.EngineArn'
```

`EngingArn`의 출력 예시:

```
"arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-dc5015c2-db17-4cb5-
afb1-EXAMPLE991:8"
```

2단계: 마지막으로 사용된 정상 Lambda 런타임 버전 ARN 가져오기

canary가 Lambda 런타임 업데이트의 영향을 받았는지 여부를 이해하는 데 도움이 되도록 로그에 나와 있는 Lambda 런타임 버전 ARN의 변경 날짜 및 시간이 canary에 영향을 미친 것으로 확인된 날짜 및 시간인지 확인합니다. 일치하지 않는 경우 문제의 원인이 Lambda 런타임 업데이트가 아닐 수 있습니다.

canary가 Lambda 런타임 업데이트의 영향을 받은 경우 이전에 사용하고 있던 작동 중인 Lambda 런타임 버전의 ARN을 찾아야 합니다. Identifying runtime version changes의 지침을 따라 이전 런타임의 ARN을 찾습니다. 런타임 버전 ARN을 기록하고, 3단계로 진행하여 런타임 관리 구성을 설정합니다.

canary가 아직 Lambda 환경 업데이트의 영향을 받지 않은 경우 현재 사용 중인 Lambda 런타임 버전의 ARN을 찾을 수 있습니다. 다음 명령을 실행하여 응답에서 Lambda 함수의 `RuntimeVersionArn`을 가져옵니다.

```
aws lambda get-function-configuration \
--function-name "arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-
dc5015c2-db17-4cb5-afb1-EXAMPLE991:8" | jq '.RuntimeVersionConfig.RuntimeVersionArn'
```

`RuntimeVersionArn`의 출력 예시:

```
"arn:aws:lambda:us-
west-2::runtime:EXAMPLE647b82f490a45d7ddd96b557b916a30128d9dcab5f4972911ec0f"
```

3단계: Lambda 런타임 관리 구성 업데이트

AWS CLI 또는 Lambda 콘솔을 사용하여 런타임 관리 구성을 업데이트할 수 있습니다.

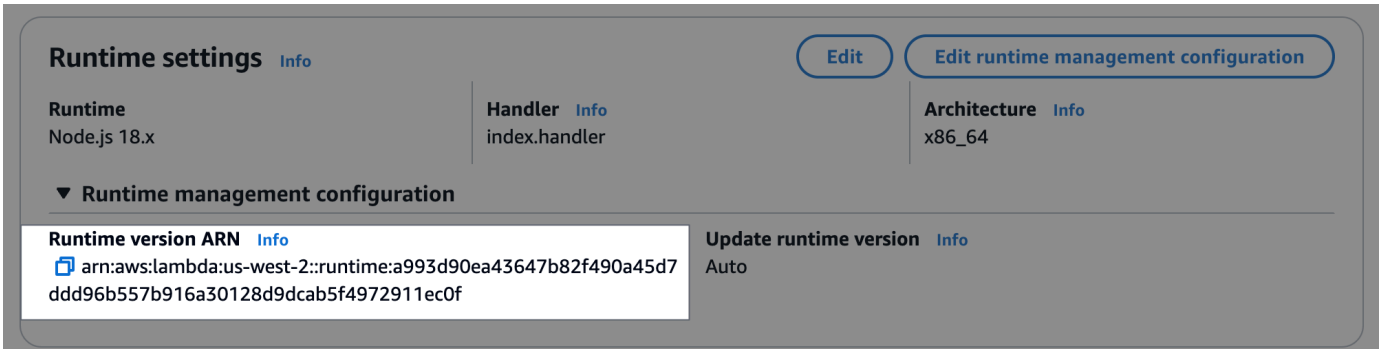
AWS CLI를 사용하여 Lambda 런타임 관리 구성 수동 모드 설정

다음 명령을 입력하여 Lambda 함수의 런타임 관리를 수동 모드로 변경합니다. 1단계에서 찾은 값을 사용하여 *function-name*과 *qualifier*를 각각 Lambda 함수 ARN 및 Lambda 함수 버전 번호로 바꿔야 합니다. 또한 *runtime-version-arn*을 2단계에서 찾은 버전 ARN으로 바꿉니다.

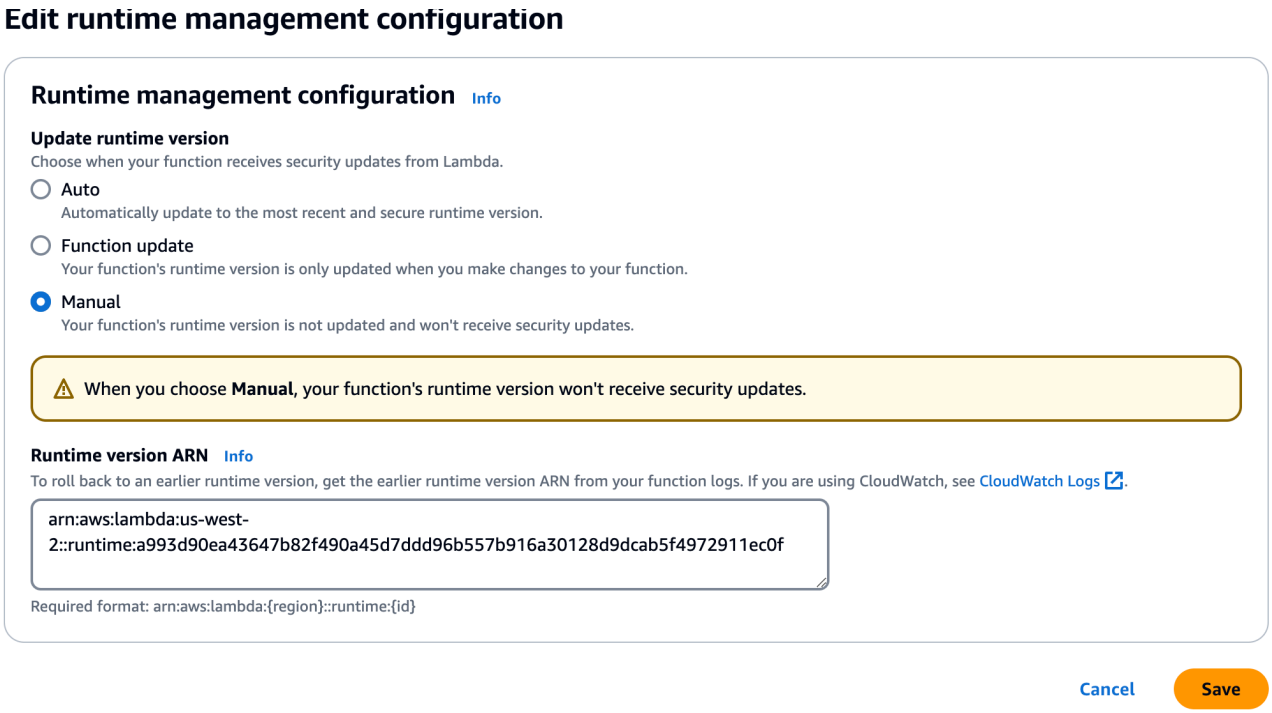
```
aws lambda put-runtime-management-config \
    --function-name "arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-
dc5015c2-db17-4cb5-afb1-EXAMPLE991" \
    --qualifier 8 \
    --update-runtime-on "Manual" \
    --runtime-version-arn "arn:aws:lambda:us-
west-2::runtime:a993d90ea43647b82f490a45d7ddd96b557b916a30128d9dcab5f4972911ec0f"
```

Lambda 콘솔을 사용하여 canary를 수동 모드로 변경

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 버전 탭을 선택하고 ARN에 해당하는 버전 번호 링크를 선택한 다음 코드 탭을 선택합니다.
3. 런타임 설정으로 스크롤하고, 런타임 관리 구성을 확장한 다음 런타임 버전 ARN을 복사합니다.



4. 런타임 관리 구성 편집을 선택하고 수동을 선택한 다음 이전에 복사한 런타임 버전 ARN을 런타임 버전 ARN 필드에 붙여넣습니다. 그런 다음 저장을 선택합니다.



canary가 AWS WAF에 의해 차단됨

AWS WAF에서 canary를 차단하지 않도록 `CloudWatchSynthetics` 문자열을 허용하는 AWS WAF 문자열 일치 조건을 설정합니다. 자세한 내용은 AWS WAF 설명서의 Working with string match conditions를 참조하세요.

요소가 표시될 때까지 대기

로그 및 스크린샷을 분석한 후 스크립트의 요소가 화면에 표시되기를 기다리고 있는데 시간이 초과되는 경우 관련 스크린샷을 확인하여 요소가 페이지에 표시되는지 확인합니다. `xpath`가 올바른지 확인합니다.

Puppeteer 관련 문제는 Puppeteer의 GitHub 페이지 또는 인터넷 게시판을 확인하세요.

노드가 표시되지 않거나 page.click()의 HTMLElement가 아님

노드가 표시되지 않거나 `page.click()`의 `HTMLElement`가 아닌 경우 먼저, 요소를 클릭하는 데 사용 중인 `xpath`를 확인합니다. 또한 요소가 화면 맨 아래에 있는 경우 뷰포트를 조정합니다. CloudWatch Synthetics는 기본적으로 1920 * 1080의 뷰포트를 사용합니다. 브라우저를 시작할 때 또는 Puppeteer 함수 `page.setViewport`를 사용하여 다른 뷰포트를 설정할 수 있습니다.

S3에 아티팩트를 업로드할 수 없음, 예외: S3 버킷 위치를 가져올 수 없음: 액세스가 거부됨

Amazon S3 오류로 인해 canary가 실패한 경우 CloudWatch Synthetics가 권한 문제로 인해 canary에 대해 생성된 스크린샷, 로그 또는 보고서를 업로드할 수 없습니다. 다음을 확인하세요.

- canary의 IAM 역할에 `s3:ListAllMyBuckets` 권한, 올바른 Amazon S3 버킷에 대한 `s3:GetBucketLocation` 권한 및 canary가 아티팩트를 저장하는 버킷에 대한 `s3:PutObject` 권한을 확인합니다. canary가 시각적 모니터링을 수행하는 경우 역할은 버킷에 대한 `s3:GetObject` 권한도 필요합니다. VPC 엔드포인트가 있는 VPC에 canary를 배포하는 경우 Amazon VPC S3 게이트웨이 엔드포인트 정책에서도 이와 동일한 권한이 필요합니다.
- canary가 표준 AWS 관리형 키(기본값) 대신 암호화를 위해 AWS KMS 고객 관리형 키를 사용하는 경우 canary의 IAM 역할은 해당 키를 사용하여 암호화하거나 복호화할 수 있는 권한이 없을 수 있습니다. 자세한 내용은 canary 아티팩트 암호화 단원을 참조하십시오.
- 버킷 정책에서 canary가 사용하는 암호화 메커니즘을 허용하지 않을 수 있습니다. 예를 들어 버킷 정책에서 특정 암호화 메커니즘 또는 KMS 키 사용을 규정한 경우 canary에 대해 동일한 암호화 모드를 선택해야 합니다.

canary가 시각적 모니터링을 수행하는 경우, 자세한 내용은 시각적 모니터링 사용 시 아티팩트 위치 및 암호화 업데이트 섹션을 참조하세요.

오류: 프로토콜 오류(Runtime.callFunctionOn): 대상이 닫혔습니다.

이 오류는 페이지 또는 브라우저가 닫힌 후 일부 네트워크 요청이 있는 경우 나타납니다. 비동기 작업을 기다리는 것을 잊었을 수 있습니다. 스크립트를 실행한 후 CloudWatch Synthetics는 브라우저를 닫습니다. 브라우저가 닫힌 후 비동기 작업을 실행하면 `target closed error`가 발생할 수 있습니다.

canary 실패. 오류: 데이터 요소 없음 - canary가 시간 초과 오류를 표시함

이는 canary 실행이 제한 시간을 초과했음을 의미합니다. CloudWatch Synthetics가 SuccessPercent CloudWatch 지표를 게시하거나 HAR 파일, 로그, 스크린샷과 같은 아티팩트를 업데이트하기 전에 canary 실행이 중지되었습니다. 제한 시간이 너무 낮으면 시간을 늘릴 수 있습니다.

기본적으로 canary 제한 시간 값은 해당 빈도와 같습니다. canary 빈도보다 작거나 같도록 제한 시간 값을 수동으로 조정할 수 있습니다. canary 빈도가 낮으면 빈도를 증가하여 제한 시간을 늘려야 합니다. CloudWatch Synthetics 콘솔을 사용하여 canary를 생성하거나 업데이트할 때 Schedule(일정)에서 빈도와 초과 시간 값을 모두 조정할 수 있습니다.

Lambda 콜드 스타트와 canary 계측 부팅에 걸리는 시간이 허용되려면 canary 시간 초과 값이 15초 이상이어야 합니다.

이 오류가 발생하면 CloudWatch Synthetics 콘솔에서 canary 아티팩트를 볼 수 없습니다. CloudWatch Logs를 사용하여 canary의 로그를 확인할 수 있습니다.

CloudWatch Logs를 사용하여 canary의 로그를 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [로그 그룹(Log groups)]을 선택합니다.
3. 필터 상자에 canary 이름을 입력하여 로그 그룹을 찾습니다. canary의 로그 그룹 이름은 /aws/lambda/cwsyn-***canaryName***-randomId입니다.

내부 엔드포인트에 액세스하려고 시도

canary가 내부 네트워크의 엔드포인트에 액세스하도록 하려면 VPC를 사용하도록 CloudWatch Synthetics를 설정하는 것이 좋습니다. 자세한 내용은 VPC에서 canary 실행 단원을 참조하십시오.

canary 런타임 버전 업그레이드 및 다운그레이드 문제

최근에 런타임 버전 `syn-1.0`에서 이후 버전으로 canary를 업그레이드한 경우 교차 원본 요청 공유(CORS) 문제일 수 있습니다. 자세한 내용은 교차 원본 요청 공유(CORS) 문제 단원을 참조하세요.

최근에 canary를 이전 런타임 버전으로 다운그레이드한 경우 사용 중인 CloudWatch Synthetics 함수를 다운그레이드한 이전 런타임 버전에서 사용할 수 있는지 확인합니다. 예를 들어 `executeHttpStep` 함수는 런타임 버전 `syn-nodejs-2.2` 이상에서 사용할 수 있습니다. 함수의 가용성을 확인하려면 canary 스크립트 작성 단원을 참조하세요.

> **Note**
>
> canary의 런타임 버전을 업그레이드하거나 다운그레이드할 계획이 있다면 먼저, canary를 복제하고 복제된 canary에서 런타임 버전을 업데이트하는 것이 좋습니다. 새 런타임 버전의 복제가 작동하는지 확인한 후에는 원래 canary의 런타임 버전을 업데이트하고 복제를 삭제할 수 있습니다.

교차 원본 요청 공유(CORS) 문제

UI canary에서 일부 네트워크 요청이 `403` 또는 `net::ERR_FAILED`로 실패하는 경우 canary에 활성 추적이 사용 설정되어 있는지 그리고 Puppeteer 함수 `page.setExtraHTTPHeaders`를 사용하여 헤더를 추가했는지 확인합니다. 그렇다면 실패한 네트워크 요청은 교차 원본 요청 공유(CORS) 제한으로 인해 발생했을 수 있습니다. 활성 추적을 사용 중지하거나 추가 HTTP 헤더를 제거하여 이에 해당하는 경우인지 여부를 확인할 수 있습니다.

이런 일이 발생하는 이유는 무엇인가요?

활성 추적을 사용하면 호출을 추적하기 위해 모든 발신 요청에 추가 헤더가 추가됩니다. 추적 헤더를 추가하거나 Puppeteer의 `page.setExtraHTTPHeaders`를 사용하여 추가 헤더를 추가하여 요청 헤더를 수정하면 XMLHttpRequest(XHR) 요청에 대한 CORS 검사가 발생합니다.

활성 추적을 사용 중지하거나 추가 헤더를 제거하지 않으려는 경우 교차 원본 액세스를 허용하도록 웹 애플리케이션을 업데이트하거나 스크립트에서 Chrome 브라우저를 시작할 때 `disable-web-security` 플래그를 사용하여 웹 보안을 사용 중지할 수 있습니다.

CloudWatch Synthetics launch 함수를 사용하여 CloudWatch Synthetics에서 사용하는 시작 파라미터를 재정의하고 추가 `disable-web-security` 플래그 파라미터를 전달할 수 있습니다. 자세한 내용은 Node.js canary 스크립트에 사용할 수 있는 라이브러리 함수 단원을 참조하세요.

> **Note**
>
> 런타임 버전 `syn-nodejs-2.1` 이상을 사용할 경우 CloudWatch Synthetics에서 사용하는 시작 파라미터를 재정의할 수 있습니다.

canary 경쟁 조건 문제

CloudWatch Synthetics를 사용할 때 최상의 경험을 위해 canary용으로 작성된 코드가 멱등성이어야 합니다. 그렇지 않으면 드물기는 하지만 canary가 여러 실행에서 동일한 리소스와 상호 작용할 때 canary 실행 시 경쟁 조건이 발생할 수 있습니다.

VPC에서 canary 문제 해결

VPC에서 canary를 생성하거나 업데이트한 후 문제가 발생할 경우 다음 단원 중 하나를 참고하여 문제를 해결할 수 있습니다.

새 canary가 오류 상태이거나 canary를 업데이트할 수 없음

VPC에서 실행할 canary를 생성한 후 canary가 즉시 오류 상태가 되거나 VPC에서 실행되도록 canary를 업데이트할 수 없는 경우 canary의 역할에 적절한 권한이 없을 수 있습니다. VPC에서 실행하려면 canary에 `ec2:CreateNetworkInterface`, `ec2:DescribeNetworkInterfaces` 및 `ec2:DeleteNetworkInterface` 권한이 있어야 합니다. 이러한 권한은 모두 `AWSLambdaVPCAccessExecutionRole` 관리형 정책에 포함됩니다. 자세한 내용은 실행 역할 및 사용자 권한을 참조하세요.

canary를 생성할 때 이 문제가 발생하면 canary를 삭제하고 새 canary를 만들어야 합니다. CloudWatch 콘솔을 사용하여 새 canary를 생성하는 경우 [액세스 권한(Access Permissions)]에서 [새 역할 생성(Create a new role)]을 선택합니다. canary를 실행하는 데 필요한 모든 권한을 포함하는 새 역할이 생성됩니다.

canary를 업데이트할 때 이 문제가 발생하면 canary를 다시 업데이트하고 필요한 권한이 있는 새 역할을 제공할 수 있습니다.

"No test result returned" 오류

canary에서 “no test result returned(테스트 결과가 반환되지 않음)” 오류가 표시되면 다음 문제 중 하나가 원인일 수 있습니다.

- VPC에서 인터넷에 액세스할 수 없는 경우 VPC 엔드포인트를 사용하여 canary에 CloudWatch 및 Amazon S3에 대한 액세스를 제공해야 합니다. 이러한 엔드포인트 주소를 올바르게 확인할 수 있도록 VPC에서 DNS 확인 및 DNS 호스트 이름 옵션을 활성화해야 합니다. 자세한 내용은 Using DNS with Your VPC 및 Using CloudWatch and CloudWatch Synthetics with interface VPC endpoints를 참조하세요.

- canary는 VPC 내의 프라이빗 서브넷에서 실행해야 합니다. 이를 확인하려면 VPC 콘솔에서 서브넷 페이지를 엽니다. canary를 구성할 때 선택한 서브넷을 확인합니다. 인터넷 게이트웨이(igw-)로 연결되는 경로가 있는 경우 프라이빗 서브넷이 아닙니다.

이러한 문제를 해결하는 데 도움이 되는 canary 로그를 참조하세요.

canary의 로그 이벤트를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. canary 로그 그룹의 이름을 선택합니다. 로그 그룹 이름은 `/aws/lambda/cwsyn-`*`canary-name`*으로 시작합니다.

canary 스크립트 샘플 코드

이 단원에는 CloudWatch Synthetics canary 스크립트에 사용할 수 있는 몇 가지 함수를 보여 주는 코드 샘플이 포함되어 있습니다.

Node.js 및 Puppeteer 샘플

쿠키 설정

웹 사이트는 사용자 지정 기능을 제공하거나 사용자를 추적하기 위해 쿠키를 사용합니다. CloudWatch Synthetics 스크립트에서 쿠키를 설정함으로써 이 사용자 지정 동작을 모방하고 검증할 수 있습니다.

예를 들어 웹 사이트는 재방문 사용자에 대해 [등록(Register)] 링크 대신 [로그인(Login)] 링크를 표시할 수 있습니다.

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const pageLoadBlueprint = async function () {

    let url = "http://smile.amazon.com/";

    let page = await synthetics.getPage();

    // Set cookies.  I found that name, value, and either url or domain are required
 fields.
    const cookies = [{
```

```
      'name': 'cookie1',
      'value': 'val1',
      'url': url
    },{
      'name': 'cookie2',
      'value': 'val2',
      'url': url
    },{
      'name': 'cookie3',
      'value': 'val3',
      'url': url
    }];

    await page.setCookie(...cookies);

    // Navigate to the url
    await synthetics.executeStep('pageLoaded_home', async function (timeoutInMillis =
 30000) {

        var response = await page.goto(url, {waitUntil: ['load', 'networkidle0'],
 timeout: timeoutInMillis});

        // Log cookies for this page and this url
        const cookiesSet = await page.cookies(url);
        log.info("Cookies for url: " + url + " are set to: " +
 JSON.stringify(cookiesSet));
    });

};

exports.handler = async () => {
    return await pageLoadBlueprint();
};
```

디바이스 에뮬레이션

다양한 디바이스를 에뮬레이션하는 스크립트를 작성하여 해당 디바이스에서 페이지가 어떻게 보이고 동작하는지 대략적으로 알 수 있습니다.

다음 샘플은 iPhone 6 디바이스를 에뮬레이션합니다. 에뮬레이션에 대한 자세한 내용은 Puppeteer 설명서의 page.emulate(options)를 참조하세요.

```
var synthetics = require('Synthetics');
```

```
const log = require('SyntheticsLogger');
const puppeteer = require('puppeteer-core');

const pageLoadBlueprint = async function () {

    const iPhone = puppeteer.devices['iPhone 6'];

    // INSERT URL here
    const URL = "https://amazon.com";

    let page = await synthetics.getPage();
    await page.emulate(iPhone);

    //You can customize the wait condition here. For instance,
    //using 'networkidle2' may be less restrictive.
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});
    if (!response) {
        throw "Failed to load page!";
    }

    await page.waitFor(15000);

    await synthetics.takeScreenshot('loaded', 'loaded');

    //If the response status code is not a 2xx success code
    if (response.status() < 200 || response.status() > 299) {
        throw "Failed to load page!";
    }
};

exports.handler = async () => {
    return await pageLoadBlueprint();
};
```

다단계 API canary

이 샘플 코드는 양성 및 음성 테스트 사례에 대해 동일한 API를 테스트하는 두 HTTP 단계로 API canary를 보여 줍니다. 단계 구성이 전달되어 요청 또는 응답 헤더의 보고를 사용 설정합니다. 또한 X-Amz-Security-Token 및 Authorization 헤더를 숨깁니다. 이러한 헤더에 사용자 자격 증명이 포함되어 있기 때문입니다.

이 스크립트를 canary로 사용하면 각 단계 및 관련 HTTP 요청에 관한 세부 정보(예: 단계 통과 또는 실패, 지속 시간 그리고 DNS 조회 시간 및 첫 번째 바이트 시간과 같은 성능 지표)를 볼 수 있습니다. canary 실행에 대한 2xx, 4xx, 5xx의 수를 볼 수 있습니다.

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const apiCanaryBlueprint = async function () {

    // Handle validation for positive scenario
    const validatePositiveCase = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 200 || res.statusCode > 299) {
                throw res.statusCode + ' ' + res.statusMessage;
            }

            let responseBody = '';
            res.on('data', (d) => {
                responseBody += d;
            });

            res.on('end', () => {
                // Add validation on 'responseBody' here if required. For ex, your
 status code is 200 but data might be empty
                resolve();
            });
        });
    };

    // Handle validation for negative scenario
    const validateNegativeCase = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 400) {
                throw res.statusCode + ' ' + res.statusMessage;
            }

            resolve();
        });
    };

    let requestOptionsStep1 = {
        'hostname': 'myproductsEndpoint.com',
```

```
        'method': 'GET',
        'path': '/test/product/validProductName',
        'port': 443,
        'protocol': 'https:'
    };

    let headers = {};
    headers['User-Agent'] = [synthetics.getCanaryUserAgentString(), headers['User-
Agent']].join(' ');

    requestOptionsStep1['headers'] = headers;

    // By default headers, post data and response body are not included in the report
 for security reasons.
    // Change the configuration at global level or add as step configuration for
 individual steps
    let stepConfig = {
        includeRequestHeaders: true,
        includeResponseHeaders: true,
        restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted
 header values do not appear in report generated.
        includeRequestBody: true,
        includeResponseBody: true
    };


    await synthetics.executeHttpStep('Verify GET products API with valid name',
 requestOptionsStep1, validatePositiveCase, stepConfig);

    let requestOptionsStep2 = {
        'hostname': 'myproductsEndpoint.com',
        'method': 'GET',
        'path': '/test/canary/InvalidName(',
        'port': 443,
        'protocol': 'https:'
    };

    headers = {};
    headers['User-Agent'] = [synthetics.getCanaryUserAgentString(), headers['User-
Agent']].join(' ');

    requestOptionsStep2['headers'] = headers;
```

```
    // By default headers, post data and response body are not included in the report
 for security reasons.
    // Change the configuration at global level or add as step configuration for
 individual steps
    stepConfig = {
        includeRequestHeaders: true,
        includeResponseHeaders: true,
        restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted
 header values do not appear in report generated.
        includeRequestBody: true,
        includeResponseBody: true
    };

    await synthetics.executeHttpStep('Verify GET products API with invalid name',
 requestOptionsStep2, validateNegativeCase, stepConfig);

};

exports.handler = async () => {
    return await apiCanaryBlueprint();
};
```

Python 및 Selenium 샘플

다음 Selenium 샘플 코드는 대상 요소가 로드되지 않을 때 사용자 지정 오류 메시지를 표시하며 실패하는 canary입니다.

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver
from aws_synthetics.common import synthetics_logger as logger
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

def custom_selenium_script():
    # create a browser instance
    browser = webdriver.Chrome()
    browser.get('https://www.example.com/')
    logger.info('navigated to home page')
    # set cookie
    browser.add_cookie({'name': 'foo', 'value': 'bar'})
    browser.get('https://www.example.com/')
    # save screenshot
    browser.save_screenshot('signed.png')
```

```
    # expected status of an element
    button_condition = EC.element_to_be_clickable((By.CSS_SELECTOR, '.submit-button'))
    # add custom error message on failure
    WebDriverWait(browser, 5).until(button_condition, message='Submit button failed to
 load').click()
    logger.info('Submit button loaded successfully')
    # browser will be quit automatically at the end of canary run,
    # quit action is not necessary in the canary script
    browser.quit()

# entry point for the canary
def handler(event, context):
    return custom_selenium_script()
```

canary 및 X-Ray 추적

`syn-nodejs-2.0` 이상 런타임을 사용하는 canary에서 활성 AWS X-Ray 추적을 사용하도록 선택할 수 있습니다. 추적이 사용 설정된 경우 브라우저, AWS SDK, HTTP 또는 HTTPS 모듈을 사용하는 canary가 수행한 모든 호출에 대해 추적이 전송됩니다. 추적 기능이 활성화된 canary는 애플리케이션에 대해 활성화된 후 X-Ray 트레이스 맵과 Application Signals 내에 나타납니다.

> **Note**
>
> 아시아태평양(자카르타)에서는 아직 canary의 X-Ray 추적 활성화가 지원되지 않습니다.

canary가 X-Ray 트레이스 맵에 나타날 경우 새로운 클라이언트 노드 유형으로 나타납니다. canary 노드를 마우스로 가리키면 대기 시간, 요청, 장애에 관한 데이터를 확인할 수 있습니다. 또한 canary 노드를 선택하여 페이지 하단에서 더 많은 데이터를 확인할 수도 있습니다. 페이지의 이 영역에서 [Synthetics에서 보기(View in Synthetics)]를 선택하여 CloudWatch Synthetics 콘솔로 이동해 canary에 관한 추가 세부 정보를 보거나 [추적 보기(View Traces)]를 선택하여 이 canary 실행의 추적에 관한 추가 세부 정보를 볼 수 있습니다.

또한 추적이 사용 설정된 canary의 세부 정보 페이지에는 canary 실행의 추적 및 세그먼트에 관한 세부 정보가 포함된 [추적(Tracing)] 탭이 있습니다.

추적을 사용 설정하면 canary 런타임이 2.5%~7%까지 증가합니다.

추적이 사용 설정된 canary는 다음 권한이 있는 역할을 사용해야 합니다. canary를 생성할 때 콘솔을 사용하여 역할을 생성하면 역할에 이러한 권한이 부여됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Sid230934",
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments"
            ],
            "Resource": "*"
        }
    ]
}
```

canary에 의해 생성된 추적에는 요금이 부과됩니다. X-Ray 요금에 대한 자세한 내용은 AWS X-Ray 요금을 참조하세요.

VPC에서 canary 실행

VPC의 엔드포인트와 퍼블릭 내부 엔드포인트에서 canary를 실행할 수 있습니다. VPC에서 canary를 실행하려면 VPC에서 DNS 확인 및 DNS 호스트 이름 옵션을 모두 활성화해야 합니다. 자세한 내용은 VPC에서 DNS 사용하기 단원을 참조하세요.

VPC 엔드포인트에서 canary를 실행할 때는 canary가 CloudWatch에 지표를 전송하고 Amazon S3에 아티팩트를 전송하는 방법을 제공해야 합니다. VPC에 이미 인터넷 액세스가 활성화된 경우 추가 작업이 필요 없습니다. canary는 VPC에서 실행되지만 인터넷에 액세스하여 지표와 아티팩트를 업로드할 수 있습니다.

VPC에 인터넷 액세스가 아직 활성화되지 않은 경우 다음 두 가지 옵션이 있습니다.

- VPC에 인터넷 액세스를 활성화합니다. 자세한 내용은 VPC canary에 대한 인터넷 액세스 제공 섹션을 참조하세요.
- VPC를 비공개로 유지하려는 경우 프라이빗 VPC 엔드포인트를 통해 CloudWatch 및 Amazon S3에 데이터를 전송하도록 canary를 구성할 수 있습니다. 아직 수행하지 않은 경우 CloudWatch에 대한 VPC 엔드포인트(com.amazonaws.*region*.monitoring)와 Amazon S3에 대한 게이트웨이 엔드포인트를 생성해야 합니다. 자세한 내용은 인터페이스 VPC 엔드포인트와 함께 CloudWatch 및 CloudWatch Synthetics 사용하기 및 Amazon S3용 Amazon VPC 엔드포인트를 참조하세요.

VPC canary에 대한 인터넷 액세스 제공

다음 단계에 따라 VPC canary에 인터넷 액세스를 제공하거나 canary에 고정 IP 주소를 할당합니다.

VPC canary에 대한 인터넷 액세스 제공

1. VPC 퍼블릭 서브넷에서 NAT 게이트웨이를 생성합니다. 지침은 NAT 게이트웨이 생성을 참조하세요.
2. canary가 시작된 프라이빗 서브넷의 라우팅 테이블에 새 경로를 추가합니다. 다음을 지정합니다.
   - 대상 주소(Destination)에 **0.0.0.0/0**을 입력합니다.
   - 대상에서 NAT 게이트웨이를 선택한 다음 생성한 NAT 게이트웨이의 ID를 선택합니다.
   - 라우팅 저장을 선택합니다.

   라우팅 테이블에 경로를 추가하는 방법에 대한 자세한 내용은 라우팅 테이블에서 경로 추가 및 제거를 참조하세요.

> **Note**
>
> NAT 게이트웨이에 대한 경로가 활성(active) 상태인지 확인합니다. NAT 게이트웨이가 삭제되었지만 경로를 업데이트하지 않은 경우 해당 경로는 블랙홀 상태입니다. 자세한 내용은 NAT 게이트웨이 작업을 참조하세요.

canary 아티팩트 암호화

CloudWatch Synthetics는 스크린샷, HAR 파일 및 보고서와 같은 canary 아티팩트를 Amazon S3 버킷에 저장합니다. 기본적으로 이러한 아티팩트는 AWS 관리형 키를 사용하여 저장 시 암호화됩니다. 자세한 내용은 Customer keys and AWS keys를 참조하세요.

다른 암호화 옵션을 사용하도록 선택할 수 있습니다. CloudWatch Synthetics는 다음을 지원합니다.

- SSE-S3: Amazon S3 관리형 키를 사용한 서버 측 암호화(SSE).
- SSE-KMS: AWS KMS 고객 관리형 키를 사용한 서버 측 암호화(SSE).

AWS 관리형 키를 통해 기본 암호화 옵션을 사용하고 싶지 않다면 추가 권한이 필요하지 않습니다.

SSE-S3 암호화를 사용하려면 canary를 만들거나 업데이트할 때 암호화 모드로 SSE_S3를 지정합니다. 암호화 모드를 사용하기 위한 추가 권한은 필요하지 않습니다. 더 자세한 내용은 Amazon S3-관리형 암호화 키(SSE-S3)와 함께 서버 측 암호화를 사용해 보호를 참조하세요.

AWS KMS 고객 관리형 키를 사용하려면 canary를 생성하거나 업데이트할 때 암호화 모드로 SSE-KMS를 지정하고 키의 Amazon 리소스 이름(ARN)도 제공합니다. 교차 계정 KMS 키를 사용할 수도 있습니다.

고객 관리형 키를 사용하려면 다음 설정이 필요합니다.

- canary의 IAM 역할에는 키를 사용하여 아티팩트를 암호화할 수 있는 권한이 있어야 합니다. 시각적 모니터링을 사용하는 경우 아티팩트를 복호화할 수 있는 권한도 부여해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {"Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "Your KMS key ARN"
    }
}
```

- IAM 역할에 권한을 추가하는 대신 IAM 역할을 키 정책에 추가할 수 있습니다. 여러 canary에 동일한 역할을 사용하는 경우 이 방법을 고려해야 합니다.

```
{
    "Sid": "Enable IAM User Permissions",
    "Effect": "Allow",
    "Principal": {
        "AWS": "Your synthetics IAM role ARN"
    },
    "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
    ],
    "Resource": "*"
}
```

- 교차 계정 KMS 키를 사용하는 경우 Allowing users in other accounts to use a KMS key를 참조하세요.

고객 관리형 키를 사용할 때 암호화된 canary 아티팩트 보기

canary 아티팩트를 보려면 고객 관리형 키를 업데이트하여 아티팩트를 보는 사용자에게 AWS KMS 복호화 권한을 부여합니다. 또는 아티팩트를 보고 있는 사용자 또는 IAM 역할에 복호화 권한을 추가합니다.

기본적으로 AWS KMS 정책은 KMS 키에 대한 액세스를 허용하도록 계정의 IAM 정책을 사용 설정합니다. 교차 계정 KMS 키를 사용하는 경우 "교차 계정 사용자가 사용자 지정 AWS KMS 키로 암호화된 Amazon S3 객체에 액세스하려고 할 때 액세스 거부 오류가 발생하는 이유는 무엇인가요?"를 참조하세요.

KMS 키로 인한 액세스 거부 문제를 해결하는 방법에 대한 자세한 내용은 키 액세스 문제 해결을 참조하세요.

시각적 모니터링 사용 시 아티팩트 위치 및 암호화 업데이트

시각적 모니터링을 수행하기 위해 CloudWatch Synthetics는 스크린샷을 기준선으로 선택한 실행에서 획득한 기준 스크린샷과 비교합니다. 아티팩트 위치나 암호화 옵션을 업데이트하는 경우 다음 중 하나를 수행해야 합니다.

- IAM 역할에 이전 Amazon S3 위치와 아티팩트의 새 Amazon S3 위치 모두에 대한 충분한 권한이 있는지 확인합니다. 또한 이전 및 새 암호화 방법과 KMS 키 모두에 대한 권한이 있는지 확인합니다.
- 다음 canary 실행을 새 기준으로 선택하여 새 기준을 생성합니다. 이 옵션을 사용하는 경우 IAM 역할에 새 아티팩트 위치 및 암호화 옵션에 대한 충분한 권한이 있는지 확인하기만 하면 됩니다.

다음 실행을 새 기준으로 선택하는 두 번째 옵션을 사용하는 것이 좋습니다. 이렇게 하면 canary에 더 이상 사용하지 않는 아티팩트 위치 또는 암호화 옵션에 대한 종속성을 피할 수 있습니다.

예를 들어 canary에서 아티팩트 위치 A와 KMS 키 K를 사용하여 아티팩트를 업로드한다고 가정합니다. canary를 아티팩트 위치 B 및 KMS 키 L로 업데이트하는 경우 IAM 역할에 아티팩트 위치(A와 B)와 KMS 키(K 및 L) 모두에 대한 권한이 있는지 확인할 수 있습니다. 또는 다음 실행을 새 기준으로 선택하고 canary IAM 역할에 아티팩트 위치 B 및 KMS 키 L에 대한 권한이 있는지 확인할 수 있습니다.

canary 통계 및 세부 정보 보기

canary에 대한 세부 정보를 보고 실행에 대한 통계를 볼 수 있습니다.

canary 실행 결과에 대한 모든 세부 정보를 보려면 충분한 권한이 있는 계정에 로그인해야 합니다. 자세한 내용은 CloudWatch canary에 필요한 역할 및 권한 단원을 참조하세요.

canary 통계 및 세부 정보를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Synthetics canary를 선택합니다.

   생성한 canary에 대한 세부 정보:

   - 상태는 가장 최근 실행을 통과한 canary 수를 시각적으로 보여줍니다.
   - 그룹에는 사용자가 만든 그룹이 표시되고, 실패한 canary 및 경고 canary가 포함된 그룹 수를 표시합니다.
   - 가장 느린 수행자는 가장 성능이 낮은 canary가 있는 그룹과 리전을 표시합니다. 이는 그룹 또는 리전 내의 모든 canary(선택한 기간 동안)의 평균 지속 시간을 합산하고 그룹 또는 리전의 canary 수로 나누어 계산합니다. 가장 느린 그룹에 대한 지표를 선택하면 가장 느린 그룹과 해당 canary만 표시하도록 테이블이 필터링됩니다. 테이블은 평균 기간을 기준으로 정렬됩니다.
   - 페이지 하단 근처에는 모든 canary를 표시하는 테이블이 있습니다. 한 열에는 각 canary에 대해 생성된 경보가 표시됩니다. canary 경보의 이름 지정 표준을 준수하는 경보만 표시됩니다. 이 표준은 `Synthetics-Alarm-`*`canaryName`*`-`*`index`* 입니다. CloudWatch 콘솔의 [Synthetics] 섹션에서 생성하는 canary 경보는 자동으로 이 이름 지정 규칙을 사용합니다. CloudWatch 콘솔의 [경보(Alarms)] 섹션에서 또는 AWS CloudFormation을 사용하여 canary 경보를 생성하는데 이 이름 지정 규칙을 사용하지 않은 경우 경보는 작동하지만 이 목록에 표시되지 않습니다.
3. 단일 canary에 대한 세부 정보를 보려면 canary 테이블에서 canary의 이름을 선택합니다.

   해당 canary에 대한 세부 정보:

   - [가용성(Availability)] 탭에는 이 canary의 최근 실행에 관한 정보가 표시됩니다.

     canary 실행에서 행 중 하나를 선택하여 해당 실행에 대한 세부 정보를 볼 수 있습니다.

     그래프 아래에서 Steps(단계), Screenshot(스크린샷), Logs(로그) 또는 HAR file(HAR 파일)을 선택하여 이러한 유형의 세부 정보를 볼 수 있습니다. canary에 활성 추적이 사용 설정되어 있는 경우 추적(Traces)을 선택하여 canary 실행의 추적 정보를 확인할 수도 있습니다.

     canary 실행의 로그는 S3 버킷 및 CloudWatch Logs에 저장됩니다.

     스크린샷은 고객이 웹 페이지를 보는 방식을 보여 줍니다. HAR 파일(HTTP 아카이브 파일)을 사용하여 웹 페이지에 관한 자세한 성능 데이터를 볼 수 있습니다. 웹 요청 목록을 분석하고 항목에 대한 로드 시간과 같은 성능 문제를 파악할 수 있습니다. 로그 파일은 canary 실행과 웹 페이지 간의 상호 작용 레코드를 보여 주며 오류의 세부 정보를 식별하는 데 사용할 수 있습니다.

canary가 `syn-nodejs-2.0-beta` 런타임 이상을 사용하는 경우 상태 코드, 요청 크기 또는 기간별로 HAR 파일을 정렬할 수 있습니다.

Steps(단계) 탭에는 canary의 단계 목록, 각 단계의 상태, 실패 원인, 단계 실행 후 URL, 스크린샷, 단계 실행 기간이 표시됩니다. HTTP 단계가 있는 API canary의 경우 런타임 `syn-nodejs-2.2` 이상을 사용한다면 단계 및 해당 HTTP 요청을 볼 수 있습니다.

[HTTP 요청(HTTP Requests)] 탭을 선택하여 canary에서 수행한 각 HTTP 요청의 로그를 볼 수 있습니다. 요청 또는 응답 헤더, 응답 본문, 상태 코드, 오류, 성능 타이밍(총 기간, TCP 연결 시간, TLS 핸드셰이크 시간, 첫 번째 바이트 시간, 콘텐츠 전송 시간)을 확인할 수 있습니다. 내부적으로 HTTP/HTTPS 모듈을 사용하는 모든 HTTP 요청이 여기에 캡처됩니다.

기본적으로 API canary에서는 보안상의 이유로 요청 헤더, 응답 헤더, 요청 본문, 응답 본문이 보고서에 포함되지 않습니다. 이러한 값을 포함하도록 선택하는 경우 데이터는 S3 버킷에만 저장됩니다. 보고서에 이 데이터를 포함하는 방법에 대한 자세한 내용은 executeHttpStep(stepName, requestOptions, [callback], [stepConfig]) 단원을 참조하세요.

텍스트, HTML, JSON의 응답 본문 콘텐츠 유형이 지원됩니다. text/HTML, text/plain, application/JSON, application/x-amz-json-1.0과 같은 콘텐츠 유형이 지원됩니다. 압축된 응답은 지원되지 않습니다.

- [모니터링(Monitoring)] 탭에는 이 canary가 게시한 CloudWatch 지표의 그래프가 표시됩니다. 지표에 대한 자세한 내용은 canary가 게시한 CloudWatch 지표 섹션을 참조하세요.

  canary가 게시한 CloudWatch 그래픽 아래에는 canary의 Lambda 코드와 관련된 Lambda 지표의 그래프가 있습니다.
- [구성(Configuration)] 탭에는 canary에 관한 구성 및 일정 정보가 표시됩니다.
- Groups(그룹) 탭에는 이 canary가 연결된 그룹(있는 경우)이 표시됩니다.
- [태그(Tags)] 탭에는 canary와 연결된 태그가 표시됩니다.

canary가 게시한 CloudWatch 지표

canary는 `CloudWatchSynthetics` 네임스페이스의 CloudWatch에 다음 지표를 게시합니다. CloudWatch 지표를 확인하는 방법에 대한 자세한 내용은 사용 가능한 지표 보기 단원을 참조하세요.

지표	설명
`SuccessPercent`	성공하고 실패가 없는 이 canary의 실행 비율입니다.

지표	설명
	유효한 측정기준: CanaryName 유효한 통계: Average 단위: 백분율
Duration	canary 실행의 지속 시간(밀리초)입니다. 유효한 측정기준: CanaryName 유효한 통계: Average 단위: 밀리초
Errors	canary가 전체 스크립트를 실행하지 못한 횟수 유효한 측정기준: CanaryName 유효한 통계: Sum
2xx	200에서 299 사이의 응답 코드로 OK 응답을 반환한 canary가 수행한 네트워크 요청 수입니다. 이 지표는 런타임 버전 syn-nodejs-2.0 이상을 사용하는 UI canary에 대해 보고되며, 런타임 버전 syn-nodejs-2.2 이상을 사용하는 API canary에 대해 보고됩니다. 유효한 측정기준: CanaryName 유효한 통계: Sum 단위: 개

지표	설명
4xx	400에서 499 사이의 응답 코드로 오류 응답을 반환한 canary가 수행한 네트워크 요청 수입니다. 이 지표는 런타임 버전 syn-nodejs-2.0 이상을 사용하는 UI canary에 대해 보고되며, 런타임 버전 syn-nodejs-2.2 이상을 사용하는 API canary에 대해 보고됩니다. 유효한 측정기준: CanaryName 유효한 통계: Sum 단위: 개
5xx	500에서 599 사이의 응답 코드로 장애 응답을 반환한 canary가 수행한 네트워크 요청 수입니다. 이 지표는 런타임 버전 syn-nodejs-2.0 이상을 사용하는 UI canary에 대해 보고되며, 런타임 버전 syn-nodejs-2.2 이상을 사용하는 API canary에 대해 보고됩니다. 유효한 측정기준: CanaryName 유효한 통계: Sum 단위: 개
Failed	실행에 실패한 canary 실행 수입니다. 이러한 실패는 canary 자체와 관련이 있습니다. 유효한 측정기준: CanaryName 유효한 통계: Sum 단위: 개

지표	설명
`Failed requests`	응답 없이 실패한 대상 웹 사이트의 canary가 실행한 HTTP 요청 수입니다. 유효한 측정기준: CanaryName 유효한 통계: Sum 단위: 개
`VisualMonitoringSuccessPercent`	canary 실행 중 기준 스크린샷과 일치한 시각적 비교의 비율입니다. 유효한 측정기준: CanaryName 유효한 통계: Average 단위: 백분율
`VisualMonitoringTotalComparisons`	canary 실행 중에 발생한 총 시각적 비교 수입니다. 유효한 측정기준: CanaryName 단위: 개

> **Note**
>
> Synthetics 라이브러리의 `executeStep()` 또는 `executeHttpStep()` 메서드를 사용하는 canary는 각 단계에 대해 `CanaryName` 및 `StepName` 측정기준과 함께 `SuccessPercent` 및 `Duration` 지표도 게시합니다.

canary 편집 또는 삭제

기존 canary를 편집하거나 삭제할 수 있습니다.

canary 편집

canary를 편집할 때 일정을 변경하지 않은 경우에도 canary를 편집한 시점에 맞춰 일정이 재설정됩니다. 예를 들어 매시간 실행되는 canary가 있는데 해당 canary를 편집한 경우 canary가 편집이 완료된 직후 실행되면 그 후 매시간 실행됩니다.

canary를 편집하거나 업데이트하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Synthetics canary를 선택합니다.
3. canary 이름 옆에 있는 버튼을 선택하고 [작업(Actions)], [편집(Edit)]을 선택합니다.
4. (선택 사항) 이 canary가 스크린샷의 시각적 모니터링을 수행할 때 canary의 다음 실행을 기준으로 설정하려는 경우 [다음 실행을 새 기준으로 설정(Set next run as new baseline)]을 선택합니다.
5. (선택 사항) 이 canary가 스크린샷의 시각적 모니터링을 수행할 때 시각적 모니터링에서 스크린샷을 제거하거나 시각적 비교 중에 무시할 스크린샷의 일부를 지정하려는 경우 [시각적 모니터링(Visual Monitoring)]에서 [기준 편집(Edit Baseline)]을 선택합니다.

   스크린샷이 표시되면 다음 중 하나를 수행할 수 있습니다.

   - 스크린샷을 제거하여 시각적 모니터링에 사용되지 않도록 하려면 [시각적 테스트 기준에서 스크린샷 제거(Remove screenshot from visual test baseline)]를 선택합니다.
   - 시각적 비교 중에 무시할 스크린샷의 일부를 지정하려면 무시할 화면 영역을 클릭하고 끌어서 그립니다. 비교 중에 무시하려는 모든 영역에 대해 이 작업을 수행했다면 [저장(Save)]을 선택합니다.
6. 그 밖에 원하는 대로 canary를 변경하고 [저장(Save)]을 선택합니다.

canary 삭제

canary를 삭제하는 경우 canary에서 사용하고 생성한 다른 리소스도 삭제할지 여부를 선택할 수 있습니다. canary를 삭제하는 경우 다음 항목도 함께 삭제해야 합니다.

- 이 canary에서 사용하는 Lambda 함수 및 계층. 해당 접두사는 `cwsyn-`*`MyCanaryName`*입니다.
- 이 canary에 대해 생성된 CloudWatch 경보. 이러한 경보에는 `Synthetics-Alarm-`*`MyCanaryName`*으로 시작하는 이름이 있습니다. 경보를 삭제하는 방법에 대한 자세한 내용은 CloudWatch 경보 편집 또는 삭제 단원을 참조하세요.
- Amazon S3 객체 및 버킷(예: canary의 결과 위치 및 아티팩트 위치).
- canary에 대해 생성된 IAM 역할. 이러한 역할은 `role/service-role/CloudWatchSyntheticsRole-`*`MyCanaryName`* 이름을 가집니다.

- canary에 대해 생성된 CloudWatch Logs의 로그 그룹. 이러한 로그 그룹은 `/aws/lambda/cwsyn-MyCanaryName-randomId` 이름을 가집니다.

canary를 삭제하기 전에 canary 세부 정보를 보고 이 정보를 기록해 두어야 할 수 있습니다. 이렇게 하면 canary를 삭제한 후 올바른 리소스를 삭제할 수 있습니다.

canary를 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Synthetics canary를 선택합니다.
3. Canary가 현재 `RUNNING` 상태에 있으면 중지해야 합니다. `STOPPED`, `READY(NOT_STARTED)` 또는 `ERROR` 상태에 있는 canary만 삭제할 수 있습니다.

   canary 이름 옆에 있는 버튼을 선택하고 [작업(Actions)], [중지(Stop)]를 선택합니다.
4. canary 이름 옆에 있는 버튼을 선택하고 [작업(Actions)], [삭제(Delete)]를 선택합니다.
5. canary에 대해 생성되고 사용되는 다른 리소스도 삭제할지 여부를 선택합니다. 여기에는 Lambda 함수와 계층, canary의 IAM 역할 및 IAM 정책이 포함됩니다.

   canary의 IAM 역할 및 IAM 정책을 삭제하려면 충분한 권한이 있어야 합니다. 자세한 내용은 CloudWatch Synthetics에 대한 AWS 관리형(미리 정의된) 정책 단원을 참조하십시오.
6. 상자에 **`Delete`**를 입력하고 [삭제(Delete)]를 선택합니다.
7. 이 단원의 앞부분에서 설명한 것과 같이 canary에서 사용하고 canary용으로 생성한 다른 리소스를 삭제합니다.

여러 canary에 대한 런타임 시작, 중지, 삭제 또는 업데이트

한 번의 작업으로 최대 5개의 canary를 중지, 시작, 삭제 또는 업데이트할 수 있습니다. canary의 런타임을 업데이트하면 canary에서 사용하는 언어 및 프레임워크에 사용할 수 있는 최신 런타임으로 canary가 업데이트됩니다.

여러 canary를 선택하고 그 중 일부만 선택한 작업에 유효한 상태인 경우 해당 작업이 유효한 canary에서만 작업이 수행됩니다. 예를 들어, 현재 실행 중인 몇 가지 canary와 실행 중이지 않은 몇 가지 canary를 선택하고 canary를 시작하도록 선택하면 아직 실행 중이 아닌 canary가 시작되고 이미 실행 중인 canary는 영향을 받지 않습니다.

선택한 canary 중 작업에 유효한 canary가 없으면 메뉴에서 해당 작업을 사용할 수 없습니다.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Synthetics canary를 선택합니다.
3. 중지, 시작 또는 삭제하려는 canary의 옆의 확인란을 선택합니다.
4. Actions(작업)를 선택한 다음 Start(시작), Stop(중지), Delete(삭제) 또는 Update Runtime(런타임 업데이트)를 선택합니다.

Amazon EventBridge를 사용하여 canary 이벤트 모니터링

Amazon EventBridge 이벤트 규칙은 canary가 상태를 변경하거나 실행을 완료할 경우 이를 알려줄 수 있습니다. EventBridge는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트의 스트림을 거의 실시간으로 제공합니다. CloudWatch Synthetics는 이러한 이벤트를 '최선을 다해' EventBridge에 전송합니다. 최선을 다한 전달이란 CloudWatch Synthetics가 모든 이벤트를 EventBridge에 전송하려고 시도하지만 드물게 이벤트가 전달되지 않을 수 있음을 의미합니다. EventBridge는 수신한 모든 이벤트를 한 번 이상 처리합니다. 또한 이벤트 리스너는 이벤트가 발생한 순서대로 이벤트를 수신하지 못할 수 있습니다.

> **Note**
>
> Amazon EventBridge는 애플리케이션을 다양한 소스의 데이터와 연결하는 데 사용할 수 있는 이벤트 버스 서비스입니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge란? 단원을 참조하세요.

CloudWatch Synthetics는 canary가 상태를 변경하거나 실행을 완료할 경우 이벤트를 내보냅니다. CloudWatch Synthetics에서 전송한 모든 이벤트 유형과 일치하거나 특정 이벤트 유형과만 일치하는 이벤트 패턴을 포함하는 EventBridge 규칙을 생성할 수 있습니다. canary가 규칙을 트리거하면 EventBridge는 규칙에 정의된 대상 작업을 호출합니다. 이렇게 하면 canary 상태 변경 또는 canary 실행 완료에 대한 응답으로 알림을 전송하고 이벤트 정보를 캡처하며 수정 작업을 수행할 수 있습니다. 예를 들면, 다음 사용 사례에 대한 규칙을 생성할 수 있습니다.

- canary 실행이 실패할 경우 조사
- canary가 `ERROR` 상태가 된 경우 조사
- canary의 수명 주기 추적
- 워크플로의 일부로 canary 실행 성공 또는 실패 모니터링

CloudWatch Synthetics의 이벤트 예

이 단원에서는 CloudWatch Synthetics의 이벤트 예를 설명합니다. 이벤트 형식에 대한 자세한 내용은 EventBridge의 이벤트 및 이벤트 패턴 단원을 참조하세요.

canary 상태 변경

이 이벤트 유형에서 `current-state` 및 `previous-state` 값은 다음일 수 있습니다.

`CREATING` | `READY` | `STARTING` | `RUNNING` | `UPDATING` | `STOPPING` | `STOPPED` | `ERROR`

```
{
                "version": "0",
                "id": "8a99ca10-1e97-2302-2d64-316c5dedfd61",
                "detail-type": "Synthetics Canary Status Change",
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:19:43Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
                                "canary-id": "EXAMPLE-dc5a-4f5f-96d1-989b75a94226",
                                "canary-name": "events-bb-1",
                                "current-state": "STOPPED",
                                "previous-state": "UPDATING",
                                "source-location": "NULL",
                                "updated-on": 1612909161.767,
                                "changed-config": {
                                                "executionArn": {
                                                                "previous-value":
 "arn:aws:lambda:us-east-1:123456789012:function:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:1",
                                                                "current-value":
 "arn:aws:lambda:us-east-1:123456789012:function:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:2"
                                                },
                                                "vpcId": {
                                                                "current-value": "NULL"
                                                },
                                                "testCodeLayerVersionArn": {
                                                                "previous-
value": "arn:aws:lambda:us-east-1:123456789012:layer:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:1",
```

```
                                                                    "current-value":
 "arn:aws:lambda:us-east-1:123456789012:layer:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:2"
                                                }
                                },
                                "message": "Canary status has changed"
                }
}
```

성공적인 canary 실행 완료

```
{
                "version": "0",
                "id": "989EXAMPLE-f4a5-57a7-1a8f-d9cc768a1375",
                "detail-type": "Synthetics Canary TestRun Successful",
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:24:01Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
                                "canary-id": "989EXAMPLE-dc5a-4f5f-96d1-989b75a94226",
                                "canary-name": "events-bb-1",
                                "canary-run-id": "c6c39152-8f4a-471c-9810-989EXAMPLE",
                                "artifact-location": "cw-syn-results-123456789012-us-
east-1/canary/us-east-1/events-bb-1-ec3-28ddbe266797/2021/02/09/22/23-41-200",
                                "test-run-status": "PASSED",
                                "state-reason": "null",
                                "canary-run-timeline": {
                                                "started": 1612909421,
                                                "completed": 1612909441
                                },
                                "message": "Test run result is generated successfully"
                }
}
```

실패한 canary 실행 완료

```
{
                "version": "0",
                "id": "2644b18f-3e67-5ebf-cdfd-bf9f91392f41",
                "detail-type": "Synthetics Canary TestRun Failure",
```

```
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:24:27Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
                                "canary-id": "af3e3a05-dc5a-4f5f-96d1-9989EXAMPLE",
                                "canary-name": "events-bb-1",
                                "canary-run-id": "0df3823e-7e33-4da1-8194-
b04e4d4a2bf6",
                                "artifact-location": "cw-syn-results-123456789012-us-
east-1/canary/us-east-1/events-bb-1-ec3-989EXAMPLE/2021/02/09/22/24-21-275",
                                "test-run-status": "FAILED",
                                "state-reason": "\"Error: net::ERR_NAME_NOT_RESOLVED
 \""
                                "canary-run-timeline": {
                                                "started": 1612909461,
                                                "completed": 1612909467
                                },
                                "message": "Test run result is generated successfully"
                }
}
```

이벤트가 중복되거나 이벤트 순서가 잘못되었을 수 있습니다. 이벤트 순서를 정하려면 `time` 속성을 사용하세요.

EventBridge 규칙 생성을 위한 사전 조건

CloudWatch Synthetics에 대한 EventBridge 규칙을 생성하기 전에 다음을 수행해야 합니다.

- Eventbridge의 이벤트, 규칙, 대상을 숙지해야 합니다.
- EventBridge 규칙에 의해 간접 호출되는 대상을 생성하고 구성해야 합니다. 규칙은 다음을 비롯한 다양한 유형의 대상을 호출할 수 있습니다.
  - Amazon SNS 주제
  - AWS Lambda 함수
  - Kinesis 스트림
  - Amazon SQS 대기열

자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge란? 및 Amazon EventBridge 시작하기 단원을 참조하세요.

EventBridge 규칙 생성(CLI)

다음 예의 단계에서는 us-east-1의 my-canary-name이라는 canary가 실행을 완료하거나 상태를 변경할 때 Amazon SNS 주제를 게시하는 EventBridge 규칙을 생성합니다.

1. 규칙을 생성합니다.

```
aws events put-rule \
  --name TestRule \
  --region us-east-1 \
  --event-pattern "{\"source\": [\"aws.synthetics\"], \"detail\": {\"canary-name\":
 [\"my-canary-name\"]}}"
```

   패턴에서 생략한 속성은 무시됩니다.

2. 규칙 대상으로 주제를 추가합니다.

   - *topic-arn*을 Amazon SNS 주제의 Amazon 리소스 이름(ARN)으로 바꿉니다.

```
aws events put-targets \
  --rule TestRule \
  --targets "Id"="1","Arn"="topic-arn"
```

> **Note**
>
> Amazon EventBridge가 대상 주제를 호출하도록 허용하려면 주제에 리소스 기반 정책을 추가해야 합니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon SNS 권한 섹션을 참조하세요.

자세한 내용은 Amazon EventBridge 사용 설명서에서 EventBridge의 이벤트 및 이벤트 패턴 단원을 참조하세요.

CloudWatch Evidently를 통해 출시 및 A/B 실험 수행

Amazon CloudWatch Evidently를 사용하면 기능을 롤아웃하는 중에 지정된 비율의 사용자에게 제공하여 새 기능을 안전하게 검증할 수 있습니다. 새 기능의 성능을 모니터링해 사용자에게 트래픽을 늘릴 시기를 결정할 수 있습니다. 이를 통해 위험을 줄이고 의도하지 않은 결과를 파악한 후 기능을 완전히 출시할 수 있습니다.

또한 A/B 실험을 수행해 증거와 데이터를 기반으로 기능 설계 결정을 내릴 수 있습니다. 실험은 한 번에 변형을 5개까지 테스트할 수 있습니다. Evidently는 실험 데이터를 수집하고 통계적인 방법을 사용하여 분석합니다. 또한 어떤 변형이 더 잘 수행되는지에 대한 명확한 권장 사항을 제공합니다. 사용자 지향 기능과 백엔드 기능을 모두 테스트할 수 있습니다.

Evidently 요금

Evidently는 Evidently 이벤트 및 Evidently 분석 단위를 기반으로 계정에 요금을 부과합니다. Evidently 이벤트에는 클릭 및 페이지 보기와 같은 데이터 이벤트와 사용자에게 제공할 기능 변형을 결정하는 배정 이벤트가 모두 포함됩니다.

Evidently 분석 단위는 Evidently에서 생성한 규칙에 따라 Evidently 이벤트에서 생성됩니다. 분석 단위는 이벤트에 대한 규칙 일치 횟수입니다. 예를 들어 사용자 클릭 이벤트는 단일 Evidently 분석 단위인 클릭 수를 생성할 수 있습니다. 또 다른 예는 2개의 Evidently 분석 단위인 체크아웃 값과 카트 항목 수를 생성할 수 있는 사용자 체크아웃 이벤트입니다. 요금에 대한 자세한 정보는 Amazon CloudWatch 요금을 참조하세요.

CloudWatch Evidently는 현재 다음 리전에서 사용 가능합니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(오레곤)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(스톡홀름)

주제

Evidently를 사용하는 IAM 정책

CloudWatch Evidently를 완전히 관리하려면 다음 권한을 가진 IAM 사용자 또는 역할로 로그인해야 합니다.

- AmazonCloudWatchEvidentlyFullAccess 정책
- ResourceGroupsandTagEditorReadOnlyAccess 정책

또한 Amazon S3 또는 CloudWatch Logs에 평가 이벤트를 저장하는 프로젝트를 생성할 수 있으려면 다음 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketPolicy",
                "s3:PutBucketPolicy",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:DescribeResourcePolicies",
                "logs:PutResourcePolicy"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

CloudWatch RUM 통합을 위한 추가 권한

또한 Amazon CloudWatch RUM과 통합해서 모니터링을 위해 CloudWatch RUM 지표를 사용하는 Evidently 출시 또는 실험을 관리하려는 경우 AmazonCloudWatchRUMFullAccess 정책이 필요합니다. CloudWatch RUM으로 데이터를 전송하는 웹 클라이언트 권한을 CloudWatch RUM에 부여하도록 IAM 역할을 생성하려면 다음 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchRUMEvidentlyRole-*",
                "arn:aws:iam::*:policy/service-role/CloudWatchRUMEvidentlyPolicy-*"
            ]
        }
    ]
}
```

Evidently에 대한 읽기 전용 액세스 권한

Evidently 데이터를 조회해야 하지만 Evidently 리소스를 만들 필요는 없는 다른 사용자에게 AmazonCloudWatchEvidentlyReadOnlyAccess 정책을 부여할 수 있습니다.

프로젝트, 기능, 출시 및 실험 생성

기능 출시 또는 A/B 실험에서 CloudWatch Evidently를 시작하려면 먼저 프로젝트를 생성합니다. 프로젝트는 리소스의 논리적 그룹입니다. 프로젝트 내에서 테스트하거나 출시하고 싶은 변형이 있는 기능을 생성합니다. 출시 또는 실험을 생성하기 전에 기능을 생성하거나 동시에 생성할 수 있습니다.

주제

새 프로젝트 만들기

다음 단계를 사용하여 새 CloudWatch Evidently 프로젝트를 설정하세요.

새 CloudWatch Evidently 프로젝트 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 프로젝트 만들기를 선택합니다.
4. 프로젝트 이름(Project name)에서 CloudWatch Evidently 콘솔에 있는 이 프로젝트를 식별하는 데 사용할 이름을 입력합니다.

   프로젝트 설명을 추가할 수 있습니다(선택 사항).
5. 평가 이벤트 스토리지(Evaluation event storage)에서 Evidently로 수집한 평가 이벤트를 저장할지 여부를 선택합니다. 이러한 이벤트를 저장하지 않더라도 Evidently는 Evidently 대시보드에서 볼 수 있는 지표 및 기타 실험 데이터를 생성하기 위해 이벤트를 집계합니다. 자세한 내용은 프로젝트 데이터 스토리지 단원을 참조하십시오.
6. Use client-side evaluation(클라이언트측 평가 사용)에서 이 프로젝트에 대해 클라이언트측 평가를 활성화할지 여부를 선택합니다. 클라이언트측 평가를 사용하면 애플리케이션이 EvaluateEature

작업을 호출하는 대신 로컬에서 사용자 세션에 변형을 할당할 수 있습니다. 이를 통해 API 호출로 인한 지연 시간 및 가용성 위험을 줄일 수 있습니다. 자세한 내용은 AWS AppConfig 기반 클라이언트측 평가 사용 단원을 참조하십시오.

클라이언트측 평가로 프로젝트를 생성하려면 `evidently:ExportProjectAsConfiguration` 권한이 있어야 합니다.

클라이언트측 평가를 활성화하는 경우 다음 작업도 수행합니다.

a. 기존 AWS AppConfig 애플리케이션을 사용할지 아니면 새 애플리케이션을 생성할지 선택합니다.

b. 기존 AWS AppConfig 환경을 사용할지 아니면 새 애플리케이션을 생성할지 선택합니다.

AWS AppConfig의 애플리케이션 및 환경에 대한 자세한 내용을 알아보려면 AWS AppConfig 작동 방식을 참조하세요.

7. (선택 사항) 이 프로젝트에 태그를 추가하려면 태그(Tags), 새 태그 추가(Add new tag)를 선택합니다.

그런 다음, 키(Key)에 태그 이름을 입력합니다. 값(Value)에 태그의 선택적 값을 추가할 수 있습니다.

다른 태그를 추가하려면 새 태그 추가(Add new tag)를 다시 선택합니다.

자세한 내용은 AWS 리소스에 태깅을 참조하세요.

8. 프로젝트 만들기를 선택합니다.

AWS AppConfig 기반 클라이언트측 평가 사용

프로젝트에서 AWS AppConfig 기반 클라이언트측 평가(클라이언트측 평가)를 사용할 수 있습니다. 그러면 EvaluateFeature 작업을 호출하여 변형을 할당하는 대신 애플리케이션이 사용자 세션에 변형을 로컬로 할당할 수 있습니다. 이를 통해 API 호출로 인한 지연 시간 및 가용성 위험을 줄일 수 있습니다.

클라이언트측 평가를 사용하려면 AWS AppConfig Lambda 익스텐션을 Lambda 함수에 계층으로 연결하고 환경 변수를 구성합니다. 클라이언트측 평가는 로컬 호스트에서 부 프로세스로 실행됩니다. 그런 다음 `localhost`에 대해 EvaluationFeature 및 PutProjectEvent 작업을 호출할 수 있습니다. 클라이언트측 평가 프로세스는 변형 할당, 캐싱 및 데이터 동기화를 처리합니다. AWS AppConfig에 대한 자세한 내용을 알아보려면 AWS AppConfig 작동 방식을 참조하세요.

AWS AppConfig와 통합할 때 AWS AppConfig 애플리케이션 ID와 AWS AppConfig 환경 ID를 Evidently에 지정합니다. Evidently 프로젝트에서 동일한 애플리케이션 ID와 환경 ID를 사용할 수 있습니다.

클라이언트측 평가가 활성화된 프로젝트를 생성하면 Evidently에서 해당 프로젝트에 대한 AWS AppConfig 구성 프로파일을 생성합니다. 각 프로젝트의 구성 프로파일은 다릅니다.

클라이언트측 평가 액세스 제어

Evidently 클라이언트측 평가는 Evidently의 나머지 부분과 다른 액세스 제어 메커니즘을 사용합니다. 적절한 보안 조치를 구현할 수 있도록 이 내용을 이해하는 것이 좋습니다.

Evidently를 사용하면 사용자가 개별 리소스에 대해 수행할 수 있는 작업을 제한하는 IAM 정책을 생성할 수 있습니다. 예를 들어 사용자가 EvaluateFeature 작업을 수행하지 못하도록 하는 사용자 역할을 생성할 수 있습니다. IAM 정책으로 제어할 수 있는 Evidently 작업에 대한 자세한 내용을 알아보려면 Amazon CloudWatch Evidently에서 정의한 작업을 참조하세요.

클라이언트측 평가 모델은 프로젝트 메타데이터를 사용하는 Evidently 기능의 로컬 평가를 허용합니다. 클라이언트측 평가가 활성화된 프로젝트의 사용자는 로컬 호스트 엔드포인트에 대해 EvaluateFeature API를 호출할 수 있으며 이 API 호출은 Evidently에 도달하지 않으며 Evidently 서비스의 IAM 정책에 의해 인증되지 않습니다. 사용자에게 EvaluateFeature 작업을 사용할 IAM 권한이 없는 경우에도 이 호출은 성공합니다. 그러나 에이전트가 평가 이벤트 또는 사용자 지정 이벤트를 버퍼링하고 데이터를 Evidently 비동기식으로 오프로드하려면 사용자에게 여전히 PutProjectEvents 권한이 필요합니다.

또한 클라이언트측 평가를 사용하는 프로젝트를 생성하려면 사용자에게 `evidently:ExportProjectAsConfiguration` 권한이 있어야 합니다. 이렇게 하면 클라이언트측 평가 중 호출되는 EvaluateFeature 작업에 대한 액세스를 제어하는 데 도움이 됩니다.

주의하지 않으면 클라이언트측 평가 보안 모델이 Evidently의 나머지 부분에 대해 설정한 정책을 손상시킬 수 있습니다. `evidently:ExportProjectAsConfiguration` 권한이 있는 사용자는 클라이언트측 평가가 활성화된 프로젝트를 생성한 다음 IAM 정책에서 EvaluateFeature 작업이 명시적으로 거부된 경우에도 해당 프로젝트에 대한 클라이언트측 평가에 EvaluateFeature 작업을 사용할 수 있습니다.

Lambda 시작하기

Evidently는 현재 AWS Lambda 환경을 사용하여 클라이언트측 평가를 지원합니다. 시작하려면 먼저 사용할 AWS AppConfig 애플리케이션 및 환경을 결정합니다. 기존 애플리케이션과 환경을 선택하거나 새로 생성합니다.

다음 샘플 AWS AppConfig AWS CLI 명령은 애플리케이션과 환경을 생성합니다.

```
aws appconfig create-application --name YOUR_APP_NAME
```

```
aws appconfig create-environment --application-id YOUR_APP_ID --
name YOUR_ENVIRONMENT_NAME
```

그런 다음 이러한 AWS AppConfig 리소스를 사용하여 Evidently 프로젝트를 생성합니다. 자세한 내용은 새 프로젝트 만들기 단원을 참조하십시오.

클라이언트측 평가는 Lambda 계층을 사용하여 Lambda에서 지원됩니다. 이는 AWS AppConfig 서비스에서 생성한 퍼블릭 AWS AppConfig 익스텐션인 AWS-AppConfig-Extension의 일부인 퍼블릭 계층입니다. Lambda 계층에 대한 자세한 내용을 알아보려면 Layer(계층)를 참조하세요.

클라이언트측 평가를 사용하려면 이 계층을 Lambda 함수에 추가하고 권한 및 환경 변수를 구성해야 합니다.

Lambda 함수에 Evidently 클라이언트측 평가 Lambda 계층 추가 및 구성

1. Lambda 함수를 아직 생성하지 않은 경우 함수를 생성합니다.
2. 함수에 클라이언트측 평가 계층을 추가합니다. ARN을 지정합니다. 또는 ARN을 아직 선택하지 않은 경우 AWS 계층 목록에서 ARN을 선택합니다. 자세한 내용을 알아보려면 계층을 사용하도록 함수 구성 및 Available versions of the AWS AppConfig Lambda extension(Lambda 익스텐션의 사용 가능한 버전)을 참조하세요.
3. 다음 내용으로 EvidentlyAppConfigCachingAgentPolicy라는 IAM 정책을 생성하고 함수의 실행 역할에 연결합니다. 자세한 내용을 알아보려면 Lambda 실행 역할을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "appconfig:GetLatestConfiguration",
                "appconfig:StartConfigurationSession",
                "evidently:PutProjectEvents"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

4. Lambda 함수에 필요한 환경 변수 `AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS`를 추가합니다. 이 환경 변수는 Evidently 프로젝트와 AWS AppConfig 리소스 간의 매핑을 지정합니다.

   하나의 Evidently 프로젝트에 이 함수를 사용하는 경우 환경 변수의 값을 `applications/APP_ID/environments/ENVIRONMENT_ID/configurations/PROJECT_NAME`으로 설정합니다.

   여러 Evidently 프로젝트에 이 함수를 사용하는 경우 `applications/APP_ID_1/environments/ENVIRONMENT_ID_1/configurations/PROJECT_NAME_1,applications/APP_ID_2/environments/ENVIRONMENT_ID_2/configurations/PROJECT_NAME_2`와 같이 쉼표를 사용하여 값을 구분합니다.

5. (선택 사항) 기타 환경 변수를 설정합니다. 자세한 내용을 알아보려면 Configuring the AWS AppConfig Lambda extension(Lambda 익스텐션 구성)을 참조하세요.

6. 애플리케이션에서 `EvaluateFeature`를 `localhost`로 전송하여 로컬에서 Evidently 평가를 가져옵니다.

   Python 예제:

```
import boto3
from botocore.config import Config

def lambda_handler(event, context):
    local_client = boto3.client(
        'evidently',
        endpoint_url="http://localhost:2772",
        config=Config(inject_host_prefix=False)
    )
    response = local_client.evaluate_feature(
        project=event['project'],
        feature=event['feature'],
        entityId=event['entityId']
    )
    print(response)
```

   Node.js 예제:

```
const AWS = require('aws-sdk');
const evidently = new AWS.Evidently({
    region: "us-west-2",
    endpoint: "http://localhost:2772",
    hostPrefixEnabled: false
});

exports.handler = async (event) => {

    const evaluation = await evidently.evaluateFeature({
        project: 'John_ETCProject_Aug2022',
        feature: 'Feature_IceCreamFlavors',
        entityId: 'John'
    }).promise()

    console.log(evaluation)
    const response = {
        statusCode: 200,
        body: evaluation,
    };
    return response;
};
```

Kotlin 예제:

```
String localhostEndpoint = "http://localhost:2772/"
public AmazonCloudWatchEvidentlyClient getEvidentlyLocalClient() {
    return AmazonCloudWatchEvidentlyClientBuilder.standard()

 .withEndpointConfiguration(AwsClientBuilder.EndpointConfiguration(localhostEndpoint,
 region))

 .withClientConfiguration(ClientConfiguration().withDisableHostPrefixInjection(true))
        .withCredentials(credentialsProvider)
        .build();
}

AmazonCloudWatchEvidentlyClient evidently = getEvidentlyLocalClient();

// EvaluateFeature via local client.
EvaluateFeatureRequest evaluateFeatureRequest = new
 EvaluateFeatureRequest().builder()
```

```
    .withProject(${YOUR_PROJECT}) //Required.
    .withFeature(${YOUR_FEATURE}) //Required.
    .withEntityId(${YOUR_ENTITY_ID}) //Required.
    .withEvaluationContext(${YOUR_EVAL_CONTEXT}) //Optional: a JSON object of
  attributes that you can optionally pass in as part of the evaluation event sent to
  Evidently.
    .build();

EvaluateFeatureResponse evaluateFeatureResponse =
  evidently.evaluateFeature(evaluateFeatureRequest);

// PutProjectEvents via local client.
PutProjectEventsRequest putProjectEventsRequest = new
  PutProjectEventsRequest().builder()
     .withData(${YOUR_DATA})
     .withTimeStamp(${YOUR_TIMESTAMP})
     .withType(${YOUR_TYPE})
     .build();

PutProjectEvents putProjectEventsResponse =
  evidently.putProjectEvents(putProjectEventsRequest);
```

클라이언트가 Evidently로 데이터를 전송하는 빈도 구성

클라이언트측 평가가 Evidently로 데이터를 전송하는 빈도를 지정하기 위해 선택 사항으로 2개의 환경 변수를 구성할 수 있습니다.

- AWS_APPCONFIG_EXTENSION_EVIDENTLY_EVENT_BATCH_SIZE는 Evidently로 전송하기 전에 일괄 처리할 프로젝트당 이벤트 수를 지정합니다. 유효한 값은 1에서 50 사이의 정수이며 기본값은 40입니다.
- AWS_APPCONFIG_EXTENSION_EVIDENTLY_BATCH_COLLECTION_DURATION은 Evidently로 이벤트를 전송하기 전에 대기할 기간(초)을 지정합니다. 기본값은 30입니다.

문제 해결

다음 정보로 AWS AppConfig 기반 클라이언트측 평가와 함께 CloudWatch Evidently를 사용할 때 발생하는 문제 해결을 돕습니다.

EvaluateFeature 작업 호출 시 오류 발생(BadRequestException): 제공된 경로에 대해 HTTP 메서드가 지원되지 않음

환경 변수가 잘못 구성되었을 수 있습니다. 예를 들어 `AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS` 대신 `EVIDENTLY_CONFIGURATIONS`를 환경 변수 이름으로 사용했을 수 있습니다.

ResourceNotFoundException: 배포를 찾을 수 없음

프로젝트 메타데이터에 대한 업데이트가 AWS AppConfig에 배포되지 않았습니다. 클라이언트측 평가에 사용한 AWS AppConfig 환경에서 활성 배포가 있는지 확인합니다.

ValidationException: 프로젝트에 대한 Evidently 구성 없음

`AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS` 환경 변수가 잘못된 프로젝트 이름으로 구성되었을 수 있습니다.

프로젝트에 기능 추가

CloudWatch Evidently의 기능은 출시하려는 기능이나 변형을 테스트하려는 기능을 나타냅니다.

기능을 추가하려면 먼저 프로젝트를 만들어야 합니다. 자세한 내용은 새 프로젝트 만들기 단원을 참조하십시오.

프로젝트에 기능 추가

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 프로젝트 이름을 선택합니다.
4. 기능 추가(Add feature)를 선택합니다.
5. 기능 이름(Feature name)에 프로젝트 내에서의 기능을 식별하는 데 사용할 이름을 입력합니다.

   기능 설명을 추가할 수 있습니다(선택 사항).
6. 기능 변형(Feature variations)의 변형 유형(Variation type)에서 Boolean, Long, Double 또는 String을 선택합니다. 자세한 내용은 변형 유형 단원을 참조하십시오.
7. 기능에 대해 5개까지 변형을 추가할 수 있습니다. 각 변형에 대한 값(Value)은 선택한 변형 유형(Variation type)에 대해 유효해야 합니다.

변형 중 하나를 기본값으로 지정합니다. 이 변형은 다른 변형과 비교할 기준이며 현재 사용자에게 제공되는 변형이어야 합니다. 또한 이 기능에 대한 출시 또는 실험에 추가되지 않은 사용자에게 제공되는 변형입니다.

8. 샘플 코드(Sample code)를 선택합니다. 코드 예제에서는 변형을 설정하고 사용자 세션을 할당하기 위해 애플리케이션에 무엇을 추가해야 하는지를 보여줍니다. 코드는 JavaScript, Java 및 Python 중에서 선택할 수 있습니다.

   지금 애플리케이션에 코드를 추가할 필요는 없지만, 코드를 추가한 후 출시 또는 실험을 시작해야 합니다.

   자세한 내용은 애플리케이션에 코드 추가 단원을 참조하십시오.

9. (선택 사항) 특정 사용자에게 항상 특정 변형을 표시하도록 지정하려면 재정의(Overrides), 재정의 추가(Add override)를 선택합니다. 그런 다음, 식별자(Identifier)에 사용자 ID, 계정 ID 또는 기타 식별자를 입력해 사용자를 지정하고 어떤 변형을 표시할지 지정합니다.

   이 기능은 자체 테스트 팀 구성원이나 다른 내부 사용자에게 특정 변형을 확인시키고자 할 때 유용할 수 있습니다. 재정의가 할당된 사용자의 세션은 시작 또는 실험 지표에 기여하지 않습니다.

   다시 재정의 추가를 선택해 최대 20명의 사용자에 대해 이 작업을 반복할 수 있습니다.

10. (선택 사항) 이 기능에 태그를 추가하려면 태그(Tags), 새 태그 추가(Add new tag)를 선택합니다.

    그런 다음, 키(Key)에 태그 이름을 입력합니다. 값(Value)에 태그의 선택적 값을 추가할 수 있습니다.

    다른 태그를 추가하려면 새 태그 추가(Add new tag)를 다시 선택합니다.

    자세한 내용은 AWS 리소스에 태깅을 참조하세요.

11. 기능 추가(Add feature)를 선택합니다.

변형 유형

기능을 생성하고 변형을 정의할 때 변형 유형을 선택해야 합니다. 가능한 유형은 다음과 같습니다.

- 불
- 긴 정수
- 배정밀도 부동 소수점 수
- String

변형 유형은 코드에서 다양한 변형이 구별되는 방식을 설정합니다. 변형 유형을 사용하여 CloudWatch Evidently의 구현을 단순화하고 출시 및 실험에서 기능을 수정하는 프로세스를 간소화할 수도 있습니다.

예를 들어 긴 정수 변형 유형으로 기능을 정의하는 경우, 변형을 구분하기 위해 지정하는 정수는 코드에 직접 전달되는 숫자가 될 수 있습니다. 버튼의 픽셀 크기를 테스트하는 것이 한 예일 수 있습니다. 변형 유형의 값은 각 변형에 사용된 픽셀 수가 될 수 있습니다. 각 변형에 대한 코드는 변형 유형 값을 읽어서 버튼 크기로 사용할 수 있습니다. 새 버튼 크기를 테스트하기 위해 다른 코드 변경 없이 변형 값에 사용된 숫자를 변경할 수 있습니다.

기능 내에서 변형 유형에 대한 값을 설정할 때, CloudWatch Evidently를 처음 사용해 보려고 A/A 테스트를 수행하거나 동일한 값을 할당할 다른 이유가 없는 한 여러 변형에 동일한 값을 할당하지 않아야 합니다.

Evidently는 유형으로 JSON에 대한 기본 지원은 제공하지 않지만, 문자열 변형 유형에서 JSON을 전달하고 코드에서 해당 JSON을 구문 분석할 수 있습니다.

세그먼트를 사용하여 대상에 집중

대상 세그먼트를 정의하여 출시 및 실험에 사용할 수 있습니다. 세그먼트는 하나 이상의 특성을 공유하는 대상의 일부입니다. Chrome 브라우저 사용자, 유럽의 사용자, 애플리케이션이 수집하는 다른 기준(예: 나이)을 충족하는 유럽의 Firefox 브라우저 사용자 등을 예로 들 수 있습니다.

실험에서 세그먼트를 사용하면 세그먼트 기준과 일치하는 사용자만 평가하도록 실험 범위가 제한됩니다. 시작 시 하나 이상의 세그먼트를 사용할 때 대상 세그먼트별로 서로 다른 트래픽 분할을 정의할 수 있습니다.

세그먼트 규칙 패턴 구문

세그먼트를 생성하려면 세그먼트 규칙 패턴을 정의합니다. 사용자 세션이 세그먼트에 있는지 여부를 평가하는 데 사용할 속성을 지정합니다. 생성하는 패턴은 Evidently가 사용자 세션에서 찾은 `evaluationContext`의 값과 비교됩니다. 자세한 내용은 EvaluateFeature 사용 단원을 참조하십시오.

세그먼트 규칙 패턴을 생성하려면 패턴을 일치시킬 필드를 지정합니다. 패턴에 `And`, `Or`, `Not`, `Exists` 등의 로직을 사용할 수도 있습니다.

`evaluationContext`가 패턴과 일치하려면 `evaluationContext`가 규칙 패턴의 모든 부분과 일치해야 합니다. Evidently는 `evaluationContext`의 필드 중 규칙 패턴에 포함되지 않은 필드를 무시합니다.

규칙 패턴이 매칭하는 값은 JSON 규칙을 따릅니다. 따옴표(“)로 묶인 문자열, 숫자 및 키워드(`true`, `false` 및 `null`)를 포함할 수 있습니다.

문자열의 경우 Evidently는 대소문자 변환이나 기타 문자열 정규화 없이 정확한 문자별 일치를 사용합니다. 따라서 규칙 일치는 대/소문자를 구분합니다. 예를 들어, `evaluationContext`에 `browser`속성이 포함되어 있지만 규칙 패턴이 `Browser`를 확인하는 경우 일치하지 않습니다.

숫자의 경우 Evidently는 문자열 표현도 사용합니다. 예를 들어 300, 300.0 및 3.0e2는 동일한 것으로 간주되지 않습니다.

`evaluationContext`에 매칭되는 규칙 패턴을 작성할 때는 `TestSegmentPattern` API 또는 `test-segment-pattern` CLI 명령을 사용하여 패턴이 올바른 JSON과 매칭되는지 테스트할 수 있습니다. 자세한 내용은 TestSegmentPattern을 참조하세요.

다음 요약에는 Evidently 세그먼트 패턴에서 사용할 수 있는 모든 비교 연산자가 나와 있습니다.

비교	예	규칙 구문
Null	UserID가 null임	`{ "UserID": [ null ] }`
비어 있음	LastName이 비어 있음	`{ "LastName": [""] }`
같음	Browser가 ‘Chrome’	`{ "Browser": [ "Chrome" ] }`
And	Country가 ‘France’이고 Device가 ‘Mobile’	`{ "Country": [ "France" ], "Device": ["Mobile"] }`

비교	예	규칙 구문
Or(단일 속성의 여러 값)	Browser가 ‘Chrome’ 또는 ‘Firefox’	`{ "Browser": ["Chrome", "Firefox"] }`
Or(다른 속성)	Browser가 ‘Safari’이거나 Device가 ‘Tablet’	`{ "$or": [ {"Browser": ["Safari"]}, {"Device": ["Tablet" }] ] }`
아님	Browser가 ‘Safari’가 아님	`{ "Browser": [ { "anything-but": [ "Safari" ] } ] }`
숫자(같음)	가격은 100임	`{ "Price": [ { "numeric": [ "=", 100 ] } ] }`
숫자(범위)	가격이 10을 초과하고 20보다 작거나 같음	`{ "Price": [ { "numeric": [ ">", 10, "<=", 20 ] } ] }`

비교	예	규칙 구문
존재함	Age 필드가 있음	`{ "Age": [ { "exists": true } ] }`
존재하지 않음	Age 필드가 없음	`{ "Age": [ { "exists": false } ] }`
접두사로 시작함	Region이 United States에 있음	`{ "Region": [ {"prefix": "us-" } ] }`
접미사로 끝남	Location의 접미사가 ‘West’임	`{ "Region": [ {"suffix": "West" } ] }`

세그먼트 규칙 예제

다음의 모든 예제에서는 규칙 패턴에서 사용하는 것과 동일한 필드 레이블 및 값을 사용하여 `evaluationContext`의 값을 전달하는 것으로 가정합니다.

다음 예제는 `Browser`가 Chrome 또는 Firefox이고 `Location`이 US-West인 경우 일치합니다.

```
{
  "Browser": ["Chrome", "Firefox"],
  "Location": ["US-West"]
}
```

다음 예제는 `Browser`가 Chrome 이외의 브라우저고 `Location`이 US로 시작하며 Age 필드가 있는 경우에 매칭됩니다.

```
{
  "Browser": [ {"anything-but": ["Chrome"]}],
  "Location": [{"prefix": "US"}],
  "Age": [{"exists": true}]
}
```

다음 예제는 Location가 Japan이고 Browser가 Safari이거나 Device가 Tablet인 경우에 매칭됩니다.

```
{
 "Location": ["Japan"],
 "$or": [
   {"Browser": ["Safari"]},
   {"Device": ["Tablet"]}
 ]
}
```

세그먼트 생성

세그먼트를 생성한 후에는 프로젝트의 모든 시작 또는 실험에 사용할 수 있습니다.

세그먼트를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. Segments(세그먼트) 탭을 선택합니다.
4. 세그먼트 생성을 선택합니다.
5. Segment name(세그먼트 이름)에 이 세그먼트를 식별하는 데 사용할 이름을 입력합니다.

   설명을 추가할 수도 있습니다.
6. Segment pattern(세그먼트 패턴)에 규칙 패턴을 정의하는 JSON 블록을 입력합니다. 규칙 패턴 구문에 대한 자세한 내용은 세그먼트 규칙 패턴 구문 섹션을 참조하세요.

출시 생성

새 기능 또는 변경 사항을 전체 사용자 중 지정된 비율(%)의 사용자에게 노출하려면 출시를 생성합니다. 그런 다음, 페이지 로드 시간 및 전환 등의 주요 지표를 모니터링한 후 모든 사용자에게 기능을 배포할 수 있습니다.

출시를 추가하려면 먼저 프로젝트를 만들어야 합니다. 자세한 내용은 새 프로젝트 만들기 단원을 참조하십시오.

출시를 추가할 때 이미 생성한 기능을 사용하거나 출시를 생성하는 동안 새 기능을 생성할 수 있습니다.

프로젝트에 출시 추가

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 프로젝트 이름 옆에 있는 버튼을 선택하고 프로젝트 작업(Project actions), 출시 생성(Create launch)을 선택합니다.
4. 출시 이름(Launch name)에서 해당 프로젝트 내에서의 기능을 식별하는 데 사용할 이름을 입력합니다.

   설명을 추가할 수 있습니다(선택 사항).
5. 기존 기능에서 선택(Select from existing features) 또는 새로운 기능 추가(Add new feature)를 선택합니다.

   기존 기능을 사용하는 경우 기능 이름(Feature name)에서 해당 기능을 선택합니다.

   새로운 기능 추가(Add new feature)를 선택하는 경우 다음을 수행합니다.

   a. 기능 이름(Feature name)에서 해당 프로젝트 내에서의 기능을 식별하는 데 사용할 이름을 입력합니다. 설명을 추가할 수 있습니다(선택 사항).

   b. 기능 변형(Feature variations)의 변형 유형(Variation type)에서 Boolean, Long, Double 또는 String을 선택합니다. 자세한 내용은 변형 유형 단원을 참조하십시오.

   c. 기능에 대해 5개까지 변형을 추가할 수 있습니다. 각 변형에 대한 값(Value)은 선택한 변형 유형(Variation type)에 대해 유효해야 합니다.

      변형 중 하나를 기본값으로 지정합니다. 이 변형은 다른 변형과 비교할 기준이며 현재 사용자에게 제공되는 변형이어야 합니다. 실험을 중지하면 해당 기본 변형은 모든 사용자에게 제공됩니다.

   d. 샘플 코드(Sample code)를 선택합니다. 코드 예제에서는 변형을 설정하고 사용자 세션을 할당하기 위해 애플리케이션에 무엇을 추가해야 하는지를 보여줍니다. 코드는 JavaScript, Java 및 Python 중에서 선택할 수 있습니다.

당장 애플리케이션에 코드를 추가할 필요는 없지만, 코드를 추가한 후 출시를 시작해야 합니다.

자세한 내용은 애플리케이션에 코드 추가 단원을 참조하십시오.

6. 출시 구성(Launch configuration)에서 출시를 즉시 시작하거나 나중에 시작하도록 예약할지 여부를 선택합니다.
7. (선택 사항) 정의한 대상 세그먼트에, 일반 대상에 사용할 트래픽 분할 대신 다른 트래픽 분할을 지정하려면 Add Segment Overrides(세그먼트 재정의 추가)를 선택합니다.

   Segment Overrides(세그먼트 재정의)에서 세그먼트를 선택하고 해당 세그먼트에 사용할 트래픽 분할을 정의합니다.

   필요한 경우 Add Segment Override(세그먼트 재정의 추가)를 선택하여 더 많은 세그먼트를 정의하고 트래픽 분할을 정의할 수 있습니다. 출시에는 최대 6개의 세그먼트 재정의를 추가할 수 있습니다.

   자세한 내용은 세그먼트를 사용하여 대상에 집중 단원을 참조하십시오.
8. Traffic configuration(트래픽 구성)에서 세그먼트 재정의와 일치하지 않는 일반 대상의 각 변형에 할당할 트래픽 비율을 선택합니다. 또한 사용자에게 제공되는 변형을 제외하도록 선택할 수 있습니다.

   트래픽 요약(Traffic summary)은 출시에 사용할 수 있는 전체 트래픽 양을 보여줍니다.
9. 나중에 시작하도록 출시 예약을 선택한 경우 출시에 여러 단계를 추가할 수 있습니다. 각 단계에서 변형 제공을 위해 서로 다른 비율을 사용할 수 있습니다. 이렇게 하려면 단계 추가(Add another step)를 선택한 다음, 단계에 대한 일정과 트래픽 비율을 지정합니다. 출시에는 최대 5개 단계를 포함할 수 있습니다.
10. 출시 중에 지표를 사용하여 기능 성능을 추적하려면 지표(Metrics)에서 지표 추가(Add metric)를 선택합니다. CloudWatch RUM 지표 또는 사용자 지정 지표를 사용할 수 있습니다.

    사용자 지정 지표를 사용하려면 여기에서 Amazon EventBridge 규칙을 사용하여 지표를 생성할 수 있습니다. 사용자 지정 지표를 생성하려면 다음을 수행합니다.

    - 사용자 지정 지표(Custom metrics)를 선택하고 지표 이름을 입력합니다.
    - 지표 규칙(Metric rule)의 엔터티 ID(Entity ID)에서 엔티티를 식별하는 방법을 입력합니다. 이는 지표 값을 기록하는 작업을 수행하는 사용자 또는 세션일 수 있습니다. 예를 들면, `userDetails.userID`입니다.
    - 값 키(Value key)에서 지표를 생성하기 위해 추적할 값을 입력합니다.

- 지표의 단위 이름을 입력합니다(선택 사항). 이 단위 이름은 표시 목적으로만 사용되며 Evidently 콘솔의 그래프에 사용됩니다.

이러한 필드를 입력하면 상자에 EventBridge 규칙을 코딩하여 지표를 만드는 방법의 예가 표시됩니다. EventBridge에 대한 자세한 내용은 “ Amazon EventBridge란 무엇인가요?”를 참조하세요.

RUM 지표를 사용하려면 애플리케이션에 대해 이미 RUM 앱 모니터가 설정되어 있어야 합니다. 자세한 내용은 CloudWatch RUM을 사용하도록 애플리케이션 설정 단원을 참조하십시오.

> **Note**
>
> RUM 지표를 사용하고 앱 모니터가 사용자 세션을 100% 샘플링하도록 구성되지 않은 경우, 출시에 참여하는 일부 사용자 세션만 지표를 Evidently로 보냅니다. 출시 지표가 정확하도록 하려면 앱 모니터는 샘플링에 100% 사용자 세션을 사용하는 것이 좋습니다.

11. (선택 사항) 출시에 대해 1개 이상의 지표를 생성하는 경우 기존 CloudWatch 경보를 이 출시와 연결할 수 있습니다. 그렇게 하려면 CloudWatch 경보 연결(Associate CloudWatch alarms)을 선택합니다.

    경보를 출시와 연결할 때 CloudWatch Evidently는 프로젝트 이름 및 출시 이름을 사용하여 경보에 태그를 추가해야 합니다. 이렇게 하면 CloudWatch Evidently가 콘솔의 출시 정보에 올바른 경보를 표시할 수 있습니다.

    CloudWatch Evidently가 이러한 태그를 추가할 것을 확인하려면 이 출시 리소스로 아래 식별된 경보 리소스에 태깅할 Evidently 허용(Allow Evidently to tag the alarm resource identified below with this launch resource)을 선택합니다. 그런 다음, 경보 연결(Associate alarm)을 선택하고 경보 이름을 입력합니다.

    CloudWatch 경보 생성에 대한 자세한 내용은 Amazon CloudWatch 경보 사용 섹션을 참조하세요.

12. (선택 사항) 이 출시에 태그를 추가하려면 태그(Tags)에서 새 태그 추가(Add new tag)를 선택합니다.

    그런 다음, 키(Key)에서 태그 이름을 입력합니다. 값(Value)에 태그의 선택적 값을 추가할 수 있습니다.

    다른 태그를 추가하려면 새 태그 추가(Add new tag)를 다시 선택합니다.

    자세한 내용은 AWS 리소스에 태깅을 참조하세요.

13. 출시 생성(Create launch)을 선택합니다.

실험 생성

실험을 사용하여 여러 버전의 기능 또는 웹 사이트를 테스트하고 실제 사용자 세션에서 데이터를 수집합니다. 이렇게 하면 증거와 데이터를 기반으로 애플리케이션을 선택할 수 있습니다.

실험을 추가하기 전에 프로젝트를 만들어야 합니다. 자세한 내용은 새 프로젝트 만들기 단원을 참조하십시오.

실험을 추가할 때 이미 생성한 기능을 사용하거나 실험을 생성하는 동안 새 기능을 생성할 수 있습니다.

프로젝트에 실험 추가

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 프로젝트 이름 옆에 있는 버튼을 선택하고 프로젝트 작업(Project actions)에서 출시 생성(Create launch)을 선택합니다.
4. 실험 이름(Experiment name)에서 해당 프로젝트 내에서의 기능을 식별하는 데 사용할 이름을 입력합니다.

   설명을 추가할 수 있습니다(선택 사항).
5. 기존 기능에서 선택(Select from existing features) 또는 새로운 기능 추가(Add new feature)를 선택합니다.

   기존 기능을 사용하는 경우 기능 이름(Feature name)에서 해당 기능을 선택합니다.

   새로운 기능 추가(Add new feature)를 선택하는 경우 다음을 수행합니다.

   a. 기능 이름(Feature name)에서 해당 프로젝트 내에서의 기능을 식별하는 데 사용할 이름을 입력합니다. 또한 설명을 입력할 수 있습니다(선택 사항).

   b. 기능 변형(Feature variations)의 변형 유형(Variation type)에서 Boolean, Long, Double 또는 String을 선택합니다. 유형은 각 변형에 사용되는 값 유형을 정의합니다. 자세한 내용은 변형 유형 단원을 참조하십시오.

   c. 기능에 대해 5개까지 변형을 추가할 수 있습니다. 각 변형에 대한 값(Value)은 선택한 변형 유형(Variation type)에 대해 유효해야 합니다.

변형 중 하나를 기본값으로 지정합니다. 이 변형은 다른 변형과 비교할 기준이며 현재 사용자에게 제공되는 변형이어야 합니다. 이 기능을 사용하는 실험을 중지하면 이전 실험에 있던 사용자 비율에 기본 변형이 제공됩니다.

d. 샘플 코드(Sample code)를 선택합니다. 코드 예제에서는 변형을 설정하고 사용자 세션을 할당하기 위해 애플리케이션에 무엇을 추가해야 하는지를 보여줍니다. 코드는 JavaScript, Java 및 Python 중에서 선택할 수 있습니다.

   당장 애플리케이션에 코드를 추가할 필요는 없지만, 코드를 추가한 후 실험을 시작해야 합니다. 자세한 내용은 애플리케이션에 코드 추가 단원을 참조하십시오.

6. 선택적으로, 이 실험을 해당 세그먼트와 일치하는 사용자에게만 적용하려는 경우 Audience(대상)에서 새로 생성한 세그먼트를 선택합니다. 세그먼트에 대한 자세한 내용은 세그먼트를 사용하여 대상에 집중 섹션을 참조하세요.
7. Traffic split for the experiment(실험의 트래픽 분할)에서 실험에 세션을 사용할 선택된 대상의 비율을 지정합니다. 그런 다음, 실험에서 사용하는 다양한 변형에 대한 트래픽을 할당합니다.

   동일한 기능에 대해 출시와 실험이 동시에 실행되고 있는 경우 대상 사용자는 먼저 출시로 이동합니다. 그런 다음, 출시에 지정된 트래픽 비율을 전체 대상에서 가져옵니다. 그 후에 여기서 지정하는 비율은 실험에 사용되는 나머지 대상의 비율입니다. 이후에 남은 트래픽은 모두 기본 변형으로 제공됩니다.
8. 지표(Metrics)에서 실험 중 변평을 평가하는 데 사용할 지표를 선택합니다. 평가에는 1개 이상의 지표를 사용해야 합니다.

   a. 지표 소스(Metric source)에서 CloudWatch RUM 지표 또는 사용자 지정 지표 중 어느 것을 사용할지 여부를 선택합니다.

   b. 지표 이름을 입력합니다. 목표(Goal)에서 더 나은 변형을 나타내기 위해 지표에 더 높은 값을 원할 경우 증가(Increase)를 선택합니다. 더 나은 변형을 나타내기 위해 지표에 더 낮은 값을 원하는 경우 감소(Decrease)를 선택합니다.

   c. 사용자 지정 지표를 사용하고 있다면 여기에서 Amazon EventBridge 규칙을 사용하여 지표를 생성할 수 있습니다. 사용자 지정 지표를 생성하려면 다음을 수행합니다.

      - 지표 규칙(Metric rule)의 엔터티 ID(Entity ID)에서 엔터티를 식별하는 방법을 입력합니다. 이는 지표 값을 기록하는 작업을 수행하는 사용자 또는 세션이 될 수 있습니다. 예를 들면, `userDetails.userID`입니다.
      - 값 키(Value key)에서 지표를 생성하기 위해 추적할 값을 입력합니다.
      - 지표의 단위 이름을 입력합니다(선택 사항). 이 단위 이름은 표시 목적으로만 사용되며 Evidently 콘솔의 그래프에 사용됩니다.

해당 애플리케이션을 모니터링하도록 RUM을 설정한 경우에만 RUM 지표를 사용할 수 있습니다. 자세한 내용은 CloudWatch RUM 사용 단원을 참조하십시오.

> **Note**
>
> RUM 지표를 사용하고 앱 모니터가 사용자 세션을 100% 샘플링하도록 구성되지 않은 경우 실험에 참여하는 일부 사용자 세션만 지표를 Evidently로 보냅니다. 실험 지표가 정확한지 확인하려면 앱 모니터는 샘플링에 100%의 사용자 세션을 사용하는 것이 좋습니다.

d. (선택 사항) 평가할 지표를 더 추가하려면 지표 추가(Add metric)를 선택합니다. 실험하는 동안 최대 3개의 지표를 평가할 수 있습니다.

9. (선택 사항) 이 실험에 사용할 CloudWatch 경보를 생성하려면 CloudWatch 경보(CloudWatch alarms)를 선택합니다. 경보를 통해 각 변형과 기본 변형 간 결과 차이가 지정한 임계값보다 큰지 여부를 모니터링할 수 있습니다. 변형의 성능이 기본 변형보다 좋지 않고 차이가 임계값보다 크면 경보 상태로 전환되어 사용자에게 알립니다.

   여기에서 경보를 생성하면 기본 변형이 아닌 각 변형에 대해 경보 하나가 생성됩니다.

   경보를 생성하는 경우 다음을 지정합니다.

   - 지표 이름(Metric name)에서 경보에 사용할 실험 지표를 선택합니다.
   - 경보 조건(Alarm condition)에서 변형 지표 값을 기본 변형 지표 값과 비교할 때 경보가 경보 상태로 전환되는 조건을 선택합니다. 예를 들어 변형이 제대로 수행되지 않음을 나타내는 높은 숫자가 표시되는 경우 더 큼(Greater) 또는 크거나 같음(Greater/Equal)을 선택합니다. 이는 지표가 페이지 로드 시간을 측정하는 경우 등에 적합합니다.
   - 임계값에 대한 숫자를 입력합니다. 이 값은 `ALARM` 상태로 전환하는 경보가 발생하도록 하는 성능상의 차이 비율입니다.
   - 기간별 평균(Average over period)에서 각 변형에 대한 지표 데이터 집계 양을 선택한 후 비교합니다.

   새 경보 추가(Add new alarm)를 다시 선택해 실험에 더 많은 경보를 추가할 수 있습니다.

   그런 다음, 경보에 대한 알림 설정(Set notifications for the alarm)을 선택하고 Amazon Simple Notification Service 주제를 선택하거나 생성해서 경보 알림을 전송할 수 있습니다. 자세한 내용은 Amazon SNS 알림 설정 섹션을 참조하세요.

10. (선택 사항) 이 실험에 태그를 추가하려면 태그(Tags)에서 새 태그 추가(Add new tag)를 선택합니다.

    그런 다음, 키(Key)에서 태그 이름을 입력합니다. 값(Value)에 태그의 선택적 값을 추가할 수 있습니다.

    다른 태그를 추가하려면 새 태그 추가(Add new tag)를 다시 선택합니다.

    자세한 내용은 AWS 리소스에 태깅을 참조하세요.
11. Create experiment(실험 생성)를 선택합니다.
12. 아직 구축하지 않았다면 애플리케이션에 해당 기능을 구축합니다.
13. 완료를 선택합니다. 사용자가 실험을 시작할 때까지 실험이 시작되지 않습니다.

다음 절차에 따라 단계를 완료하면 실험이 즉시 시작됩니다.

생성한 실험 시작

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 프로젝트 이름을 선택합니다.
4. 실험(Experiments) 탭을 선택합니다.
5. 실험 이름 옆에 있는 버튼을 선택하고 작업(Actions)에서 실험 시작(Start experiment)을 선택합니다.
6. (선택 사항) 실험을 생성할 때 실험 설정한 것을 보거나 수정하려면 실험 설정(Experiment setup)을 선택합니다.
7. 실험을 종료할 시간을 선택합니다.
8. 실험 시작(Start experiment)을 선택합니다.

    실험이 즉시 시작됩니다.

기능, 출시 및 실험 관리

이 섹션의 절차에 따라, 생성한 기능, 출시 및 실험을 관리합니다.

주제

- 기능에 대한 현재 평가 규칙 및 대상 트래픽 보기

- 출시 트래픽 수정
- 출시의 향후 단계 수정
- 실험 트래픽 수정
- 출시 중지
- 실험 중지

기능에 대한 현재 평가 규칙 및 대상 트래픽 보기

CloudWatch Evidently 콘솔을 사용하여 기능의 평가 규칙이 기능의 현재 출시, 실험 및 변형 간에 대상 트래픽을 할당하는 방법을 확인할 수 있습니다.

기능에 대한 대상 트래픽 보기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 기능이 포함된 프로젝트 이름을 선택합니다.
4. 기능(Features) 탭을 선택합니다.
5. 기능 이름을 선택합니다.

   평가 규칙(Evaluation rules) 탭에서 다음과 같이 기능에 대한 대상 트래픽 흐름을 확인할 수 있습니다.

   - 먼저 재정의를 평가합니다. 이 설정은 특정 사용자에게 항상 특정 변형을 제공하도록 지정합니다. 재정의가 할당된 사용자의 세션은 시작 또는 실험 지표에 기여하지 않습니다.
   - 다음으로 남은 트래픽이 있다면 진행 중인 출시에 사용할 수 있습니다. 진행 중인 출시가 있는 경우, 출시(Launches) 섹션의 테이블에 기능 변형 간 출시 이름과 출시 트래픽 분할이 표시됩니다. 출시(Launches) 섹션의 오른쪽에 있는 트래픽(Traffic) 표시기는 (재정의 후) 이 출시에 할당된 사용 가능한 대상의 양을 표시합니다. 출시에 할당되지 않은 나머지 트래픽은 실험(있는 경우)에서 기본 변형으로 흐릅니다.
   - 다음으로 남은 트래픽이 있다면 진행 중인 실험에 사용할 수 있습니다. 진행 중인 실험이 있는 경우 실험(Experiments) 섹션의 테이블에 실험 이름과 진행 상황이 표시됩니다. 실험(Experiments) 섹션의 오른쪽에 있는 트래픽(Traffic) 표시기는 (재정의와 출시 후) 이 실험에 할당된 사용 가능한 대상의 양을 표시합니다. 출시나 실험에 할당되지 않은 나머지 트래픽은 기능의 기본 변형으로 제공됩니다.

출시 트래픽 수정

출시가 진행 중일 때를 포함해 언제든지 출시에 대한 트래픽 할당을 수정할 수 있습니다.

동일한 기능에 대해 출시와 실험이 모두 진행 중인 경우 기능 트래픽이 변경되면 실험 트래픽이 변경됩니다. 이는 실험에 사용할 수 있는 대상이 아직 출시에 할당되지 않은 전체 대상 중 일부이기 때문입니다. 출시 트래픽이 증가하면 실험에 사용할 수 있는 대상이 줄어들고 출시 트래픽을 줄이거나 출시를 종료하면 실험에 사용할 수 있는 대상이 늘어납니다.

출시에 대한 트래픽 할당 수정

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 출시가 포함된 프로젝트 이름을 선택합니다.
4. 출시(Launches) 탭을 선택합니다.
5. 출시 이름을 선택합니다.

   출시 트래픽 수정(Modify launch traffic)을 선택합니다.
6. 제공(Serve)에서 각 변형에 할당할 새 트래픽 비율을 선택합니다. 또한 사용자에게 제공되는 변형을 제외하도록 선택할 수 있습니다. 이러한 값을 변경하면 트래픽 요약(Traffic summary)에서 전체 기능 트래픽에 대한 업데이트된 효과를 확인할 수 있습니다.

   트래픽 요약(Traffic summary)에서는 이 출시에 사용할 수 있는 전체 트래픽 양과 이 출시에 할당된 사용 가능한 트래픽 양을 표시합니다.
7. 수정을 선택합니다.

출시의 향후 단계 수정

아직 진행하지 않은 출시 단계 구성을 수정하고 출시에 단계를 더 추가할 수도 있습니다.

출시 단계를 수정

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 출시가 포함된 프로젝트 이름을 선택합니다.
4. 출시(Launches) 탭을 선택합니다.
5. 출시 이름을 선택합니다.

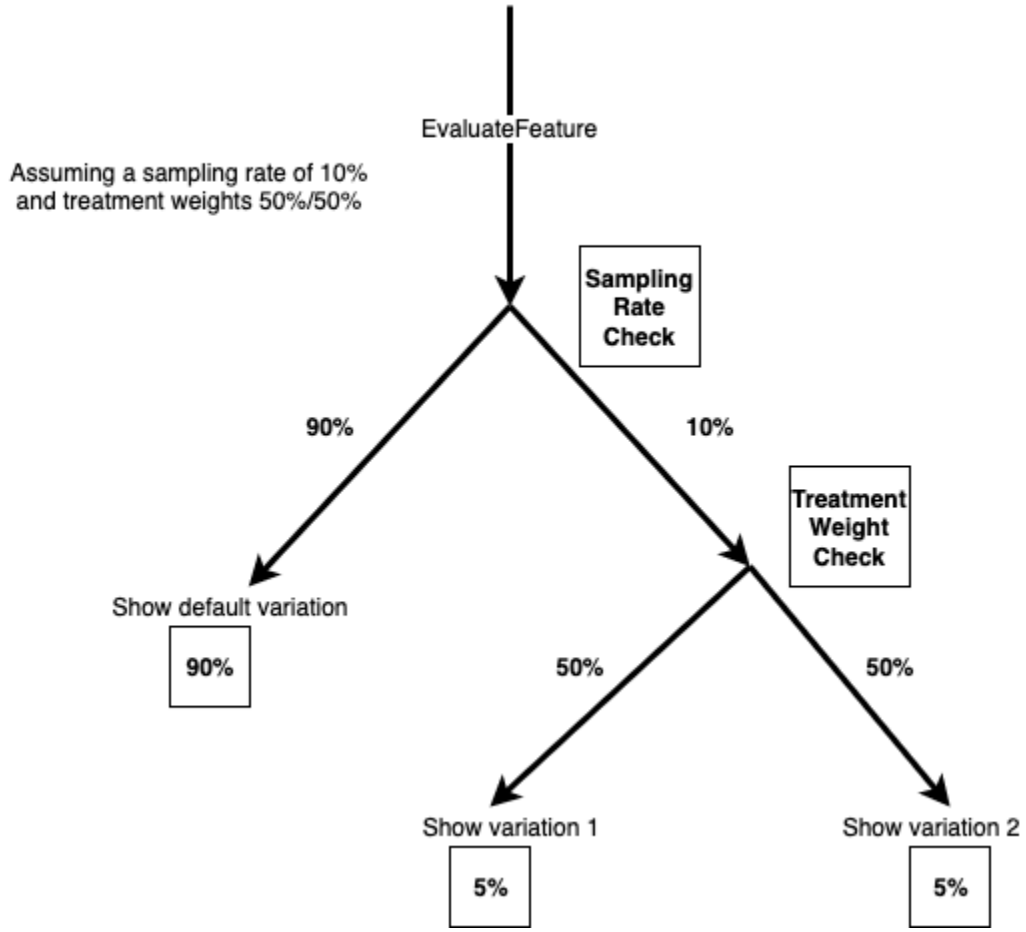
출시 트래픽 수정(Modify launch traffic)을 선택합니다.

6. 출시 예약(Schedule launch)을 선택합니다.
7. 아직 시작되지 않은 단계에 대해 실험에서 사용할 사용 가능한 대상 비율을 수정할 수 있습니다. 또한 해당 트래픽이 변형에 할당되는 방식을 수정할 수도 있습니다.

   단계 추가(Add another step)를 선택해 출시에 더 많은 단계를 추가할 수 있습니다. 출시는 최대 5단계로 진행될 수 있습니다.
8. 수정을 선택합니다.

실험 트래픽 수정

실험이 진행 중일 때를 포함해 언제든지 실험에 대한 샘플링 비율을 수정할 수 있습니다. 하지만 실험이 실행된 후에는 처리 가중치를 업데이트할 수 없습니다. 따라서 실험이 실행된 후 실험에 노출된 총 트래픽은 변경할 수 있지만 각 처리에 대한 상대적 할당량은 변경할 수 없습니다. 진행 중인 실험의 트래픽을 수정하는 경우 편향이 발생하지 않도록 트래픽 할당만 늘리는 것이 좋습니다.



실험에 대한 트래픽 할당 수정

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창의 애플리케이션 모니터링(Application monitoring)에서 Evidently를 선택합니다.
3. 출시가 포함된 프로젝트 이름을 선택합니다.
4. 실험(Experiments) 탭을 선택합니다.
5. 출시 이름을 선택합니다.
6. 실험 트래픽 수정(Modify experiment traffic)을 선택합니다.
7. 비율을 입력하거나 슬라이더를 사용하여 이 실험에 할당할 사용 가능한 트래픽 양을 지정합니다. 사용 가능한 트래픽은 총 대상에서 현재 출시에 할당된 트래픽(있는 경우)을 뺀 값입니다. 출시나 실험에 할당되지 않은 트래픽은 기본 변형으로 제공됩니다.
8. 수정을 선택합니다.

출시 중지

진행 중인 출시를 중지하면 다시 시작할 수 없습니다. 또한 트래픽 할당에 대한 규칙으로 평가되지 않으며 출시에 할당된 트래픽이 있는 경우 해당 기능의 실험에서 대신 사용할 수 있습니다. 그렇지 않으면 출시가 중지된 후 모든 트래픽이 기본 변형으로 제공됩니다.

출시 영구 중지

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 출시가 포함된 프로젝트 이름을 선택합니다.
4. 출시(Launch) 탭을 선택합니다.
5. 출시 이름 왼쪽에 있는 버튼을 선택합니다.
6. 작업(Actions)에서 출시 취소(Cancel launch) 또는 작업(Actions)에서 완료로 표시(Mark as complete)를 선택합니다.

실험 중지

진행 중인 실험을 중지하면 다시 시작할 수 없습니다. 이전에 실험에서 사용된 트래픽 부분은 기본 변형으로 제공됩니다.

실험이 수동으로 중지되지 않고 종료 날짜를 지나면 트래픽은 변경되지 않습니다. 실험에 할당된 트래픽 부분은 여전히 실험으로 이동합니다. 이를 중지하고 대신 실험 트래픽이 기본 변형으로 제공되도록 하려면 실험을 완료로 표시합니다.

실험을 중지할 때 실험을 취소하거나 완료로 표시하도록 선택할 수 있습니다. 취소하면 실험 목록에 취소됨(Cancelled)으로 표시됩니다. 완료로 표시하도록 선택하면 완료됨(Completed)으로 표시됩니다.

실험 영구 중지

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 실험이 포함된 프로젝트 이름을 선택합니다.
4. 실험(Experiments) 탭을 선택합니다.
5. 실험 이름 왼쪽에 있는 버튼을 선택합니다.
6. 작업(Actions)에서 실험 취소(Cancel experiment) 또는 작업(Actions)에서 완료로 표시(Mark as complete)를 선택합니다.

애플리케이션에 코드 추가

CloudWatch Evidently로 작업하려면 애플리케이션에 코드를 추가해 각 사용자 세션에 변형을 할당하고 지표를 Evidently로 보냅니다. CloudWatch Evidently `EvaluateFeature` 작업을 사용하여 사용자 세션에 변형을 할당하고, `PutProjectEvents` 작업을 사용하여 시작 또는 실험에 대한 지표를 계산하는 데 사용할 이벤트를 Evidently로 전송합니다.

변형 또는 사용자 지정 지표를 생성할 때 CloudWatch Evidently 콘솔에서 추가해야 하는 코드의 샘플을 제공합니다.

종합 예제를 보려면 자습서: Evidently 샘플 애플리케이션으로 A/B 테스트 섹션을 참조하세요.

EvaluateFeature 사용

시작 또는 실험에서 기능 변형을 사용하는 경우 애플리케이션은 각 사용자 세션에 변형을 할당하는 EvaluateFeature 작업을 사용합니다. 사용자에게 변형을 할당하는 것은 평가 이벤트입니다. 이 작업을 호출하면 다음을 전달합니다.

- 기능 이름(필수). Evidently는 출시 또는 실험의 기능 평가 규칙에 따라 평가를 처리하고 엔터티에 대한 변형을 선택합니다.
- entityId(필수). 고유 사용자를 나타냅니다.

- evaluationContext(선택 사항). 사용자에 대한 추가 정보를 나타내는 JSON 객체입니다. 세그먼트를 만든 경우, Evidently는 특성 평가 중에 이 값을 사용하여 사용자를 대상의 세그먼트에 매칭합니다. 자세한 내용은 세그먼트를 사용하여 대상에 집중 단원을 참조하십시오.

  다음은 Evidently로 전송할 수 있는 `evaluationContext` 값의 예입니다.

```
{
    "Browser": "Chrome",
    "Location": {
        "Country": "United States",
        "Zipcode": 98007
    }
}
```

고정 평가

CloudWatch Evidently는 "고정" 평가를 사용합니다. `entityId`, 기능, 기능 구성 및 `evaluationContext`의 단일 구성은 항상 동일한 변형 할당을 수신합니다. 엔터티가 재정의에 추가되거나 실험 트래픽이 전화 접속 연결이 될 때만 이 변형 할당이 변경됩니다.

기능 구성에는 다음이 포함되어 있습니다.

- 기능 변형
- 이 기능(있는 경우)에 대해 현재 실행 중인 실험에 대한 변형 구성(각 변형에 할당된 백분율)입니다.
- 이 기능(있는 경우)에 대해 현재 실행 중인 실행에 대한 변형 구성입니다. 변형 구성에는 정의된 세그먼트 재정의(있는 경우)가 포함됩니다.

실험의 트래픽 할당을 늘린 경우 이전에 실험 처리 그룹에 할당된 `entityId`은(는) 계속해서 동일한 처리를 받습니다. 실험에 지정된 변형 구성에 따라 이전에 대조군에 할당되었던 `entityId`를 실험 처리 그룹에 할당할 수 있습니다.

실험의 트래픽 할당이 줄어들면 `entityId`이(가) 처리 그룹에서 대조군으로 이동할 수 있지만 다른 처리 그룹에는 들어가지 않습니다.

PutProjectEvents 사용

Evidently에 대한 사용자 지정 지표를 코딩하려면 PutProjectEvents 작업을 사용합니다. 다음은 간단한 페이로드 예제입니다.

```
{
    "events": [
        {
            "timestamp": {{$timestamp}},
            "type": "aws.evidently.custom",
            "data": "{\"details\": {\"pageLoadTime\": 800.0}, \"userDetails\":
 {\"userId\": \"test-user\"}}"
        }
    ]
}
```

entityIdKey는 entityId거나, userId 같이 다른 이름으로 바꿀 수 있습니다. 실제 이벤트에서 entityId는 사용자 이름, 세션 ID 등이 될 수 있습니다.

```
"metricDefinition":{
    "name": "noFilter",
    "entityIdKey": "userDetails.userId", //should be consistent with jsonValue in
 events "data" fields
    "valueKey": "details.pageLoadTime"
},
```

이벤트가 올바른 시작 또는 실험과 연결되도록 하려면 EvaluateFeature와 PutProjectEvents를 모두 호출할 때 동일한 entityId를 전달해야 합니다. EvaluateFeature 호출 후에 PutProjectEvents를 호출해야 합니다. 그렇지 않으면 데이터가 삭제되고 CloudWatch Evidently에서 사용되지 않습니다.

PutProjectEvents 작업에는 기능 이름이 입력 파라미터로 필요하지 않습니다. 이렇게 하면 여러 실험에서 단일 이벤트를 사용할 수 있습니다. 예를 들어, entityId를 userDetails.userId로 설정하여 EvaluateFeature를 호출한다고 가정합니다. 두 개 이상의 실험이 실행 중인 경우 해당 사용자의 세션에서 단일 이벤트가 각 실험에 대한 지표를 방출하도록 할 수 있습니다. 이를 위해 동일한 entityId를 사용하여 각 실험에 대해 PutProjectEvents를 한 번 호출합니다.

Timing

애플리케이션이 EvaluateFeature를 호출한 후 PutProjectEvents의 지표 이벤트가 해당 평가를 기준으로 귀속되는 1시간의 기간이 있습니다. 1시간 후에 이벤트가 더 이상 발생하는 경우 해당 이벤트는 귀속되지 않습니다.

그러나 초기 호출의 1시간 동안 새 EvaluateFeature 호출에 동일한 entityId가 사용되는 경우 이제 이후 EvaluateFeature 결과가 대신 사용되며 1시간 타이머가 다시 시작됩니다. 이는 이전 고정

평가(Sticky evaluations) 섹션에 설명된 대로 실험 트래픽이 두 할당 간에 전화 접속 연결이 되는 경우와 같은 특정 상황에서만 발생할 수 있습니다.

종합 예제를 보려면 자습서: Evidently 샘플 애플리케이션으로 A/B 테스트 섹션을 참조하세요.

프로젝트 데이터 스토리지

Evidently는 다음과 같은 두 가지 유형의 이벤트를 수집합니다.

- 평가 이벤트는 사용자 세션에 할당된 기능 변형과 관련이 있습니다. Evidently는 이러한 이벤트를 사용하여 지표, 기타 실험 및 출시 데이터를 생성하며, 생성된 결과물은 Evidently 콘솔에서 볼 수 있습니다.

  또한 Amazon CloudWatch Logs 또는 Amazon S3에 이러한 평가 이벤트를 저장하도록 할 수 있습니다.
- 사용자 지정 이벤트는 클릭 및 체크아웃 등의 사용자 작업에서 지표를 생성하는 데 사용됩니다. Evidently는 사용자 지정 이벤트를 저장하는 방법을 제공하지 않습니다. 해당 이벤트를 저장하려면 애플리케이션 코드를 수정해 Evidently 외부의 스토리지 옵션으로 전송해야 합니다.

평가 이벤트 로그 형식

CloudWatch Logs 또는 Amazon S3에서 평가 이벤트 저장을 선택하면 각 평가 이벤트가 다음과 같은 형식의 로그 이벤트로 저장됩니다.

```
{
    "event_timestamp": 1642624900215,
    "event_type": "evaluation",
    "version": "1.0.0",
    "project_arn": "arn:aws:evidently:us-east-1:123456789012:project/petfood",
    "feature": "petfood-upsell-text",
    "variation": "Variation1",
    "entity_id": "7",
    "entity_attributes": {},
    "evaluation_type": "EXPERIMENT_RULE_MATCH",
    "treatment": "Variation1",
    "experiment": "petfood-experiment-2"
}
```

앞의 평가 이벤트 형식에 대한 자세한 내용은 다음과 같습니다.

- 타임스탬프는 밀리초 단위의 UNIX 시간입니다.

- 변형은 이 사용자 세션에 할당된 기능의 변형 이름입니다.
- 엔터티 ID는 문자열입니다.
- 엔터티 속성은 클라이언트 측에서 보낸 임의 값 해시입니다. 예: `entityId`가 파란색 또는 녹색으로 매핑되면 userID, 세션 데이터 또는 상관관계 및 데이터 웨어하우스 관점에서 원하는 항목을 선택적으로 보낼 수 있습니다.

Amazon S3의 평가 이벤트 스토리지에 대한 IAM 정책 및 암호화

Amazon S3를 사용하여 평가 이벤트를 저장하려는 경우 Evidently에서 Amazon S3 버킷에 로그를 게시할 수 있도록 다음과 같은 IAM 정책을 추가해야 합니다. Amazon S3 버킷과 버킷에 포함된 객체는 비공개이며 기본적으로 다른 서비스에 대한 액세스를 허용하지 않기 때문입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::bucket_name/optional_folder/AWSLogs/account_id/
*",
            "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-
control"}}
        },
        {
            "Sid": "AWSLogDeliveryCheck",
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": ["s3:GetBucketAcl", "s3:ListBucket"],
            "Resource": "arn:aws:s3:::bucket_name"
        }
    ]
}
```

Amazon S3에 Evidently 데이터를 저장하는 경우 AWS Key Management Service 키(SSE-KMS)로 서버 측 암호화를 사용하여 데이터를 암호화하도록 선택할 수도 있습니다. 자세한 내용은 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

AWS KMS에서 고객 관리형 키를 사용하는 경우 키에 대한 IAM 정책에 다음을 추가해야 합니다. 이를 통해 Evidently는 버킷에 쓸 수 있습니다.

```
{
    "Sid": "AllowEvidentlyToUseCustomerManagedKey",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "delivery.logs.amazonaws.com"
        ]
    },
   "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

Evidently에서 결과를 계산하는 방법

Amazon CloudWatch Evidently A/B 테스트를 데이터 기반 의사 결정을 위한 도구로 사용할 수 있습니다. A/B 테스트에서 사용자는 대조군(기본 변형이라고도 함)이나 처리군 중 하나(테스트된 변형이라고도 함)에 무작위로 할당됩니다. 예를 들어, 대조군의 사용자는 실험이 시작되기 전과 동일한 방식으로 웹 사이트, 서비스 또는 애플리케이션을 경험할 수 있습니다. 한편, 처리군의 사용자는 변경 사항을 경험할 수 있습니다.

CloudWatch Evidently는 실험에서 최대 5개의 다른 변형을 지원합니다. Evidently는 이러한 변형에 트래픽을 무작위로 할당합니다. 이러한 방식으로 각 그룹의 비즈니스 지표(예: 수익)와 성능 지표(예: 대기 시간)을 추적할 수 있습니다. Evidently는 다음을 수행합니다.

- 처리군을 대조군과 비교합니다. 예를 들어, 새로운 체크아웃 프로세스에 따른 수익 증가 또는 감소 여부를 비교합니다.
- 처리군과 대조군 간에 관찰된 차이가 유의한지 여부를 나타냅니다. 이를 위해 Evidently는 빈도주의 유의 수준과 베이지안 확률의 두 가지 접근 방식을 제공합니다.

빈도주의 접근 방식과 베이지안 접근 방식을 사용하는 이유는 무엇인가요?

처리군이 대조군에 비해 효과가 없는 경우 또는 처리군이 대조군과 동일한 경우(A/A 테스트)를 고려하세요. 데이터에서 처리군과 대조군 사이의 작은 차이가 여전히 관찰됩니다. 이는 테스트 참가자가 웹 사이트, 서비스 또는 애플리케이션의 모든 사용자 중 작은 비율을 차지하는 유한한 사용자 표본으로 구성되기 때문입니다. 빈도주의 유의 수준과 베이지안 확률은 관찰된 차이가 유의한지 아니면 우연에 의한 것인지에 대한 인사이트를 제공합니다.

관찰된 차이가 유의한지 여부를 확인하기 위해 다음 사항을 분명히 고려합니다.

- 차이의 정도
- 테스트의 일부인 샘플의 수
- 데이터 배포 상태

Evidently의 빈도주의 분석

Evidently는 빈도주의 통계의 일반적인 함정인 엿보기의 일반적인 문제를 피하는 순차 테스트를 사용합니다. 엿보기는 진행 중인 A/B 테스트의 결과를 확인하여 이를 중지하고 관찰된 결과를 기반으로 결정을 내리는 방식입니다. 순차 테스트에 대한 자세한 내용을 알아보려면 Project Euclid의 Time-uniform, nonparametric, nonasymptotic confidence sequences(시간 균일, 비모수, 비점근 신뢰 시퀀스)를 참조하세요. (인공신경망 통계. 49 (2) 1055 - 1080, 2021).

Evidently의 결과는 언제든지 유효하기 때문에(항상 유효한 결과) 실험 중에 결과를 엿보고 여전히 적절한 결론을 도출할 수 있습니다. 이렇게 하면 결과가 이미 유의미한 경우 예정된 시간 전에 실험을 중지할 수 있으므로 실험 비용을 일부 줄일 수 있습니다.

Evidently는 테스트된 변형과 대상 지표의 기본 변형 간 차이에 대해 항상 유효한 유의 수준과 항상 유효한 95% 신뢰 구간을 생성합니다. 실험 결과의 Result(결과) 열은 테스트된 변형 성능을 나타내며 다음 중 하나일 수 있습니다.

- Inconclusive(미결정) – 유의 수준이 95% 미만입니다.
- Better(좋음) – 유의 수준이 95% 이상이고 다음 중 하나가 참입니다.
  - 95% 신뢰 구간의 하한이 0보다 높으며 지표가 증가해야 합니다.
  - 95% 신뢰 구간의 상한이 0보다 낮으며 지표가 감소해야 합니다.
- Worse(나쁨) – 유의 수준이 95% 이상이고 다음 중 하나가 참입니다.
  - 95% 신뢰 구간의 상한이 0보다 높으며 지표가 증가해야 합니다.
  - 95% 신뢰 구간의 하한이 0보다 낮으며 지표가 감소해야 합니다.

- Best(최고) - 실험에는 기본 변형 외에 2개 이상의 테스트된 변형이 있으며 다음 조건이 충족됩니다.
  - 변형이 Better(좋음) 지정을 받을 자격이 있습니다.
  - 다음 중 하나가 참이어야 합니다.
    - 95% 신뢰 구간의 하한이 다른 모든 변형의 95% 신뢰 구간 상한보다 높으며 지표가 증가해야 합니다.
    - 95% 신뢰 구간의 상한이 다른 모든 변형의 95% 신뢰 구간 하한보다 낮으며 지표가 감소해야 합니다.

Evidently의 베이지안 분석

베이지안 분석을 사용하면 테스트된 변형의 평균이 기본 변형의 평균보다 크거나 작을 확률을 계산할 수 있습니다. Evidently는 켤레 사전 분포를 사용하여 대상 지표의 평균에 대한 베이지안 추론을 수행합니다. 켤레 사전 분포를 사용하면 베이지안 분석에 필요한 사후 분포를 보다 효율적으로 추론할 수 있습니다.

Evidently는 베이지안 분석 결과를 계산하기 위해 실험이 종료될 때까지 기다립니다. 결과 페이지에는 다음이 표시됩니다.

- 증가 확률 – 테스트된 변형의 지표 평균이 기본 변형의 평균보다 3% 이상 클 확률
- 감수 확률 – 테스트된 변형의 지표 평균이 기본 변형의 평균보다 3% 이상 작을 확률
- 변화가 없을 확률 – 테스트된 변형의 지표 평균이 기본 변형에서 평균의 ±3% 내에 있을 확률

Result(결과) 열은 변형 성능을 나타내며 다음 중 하나일 수 있습니다.

- Better(좋음) - 증가 확률이 90% 이상이고 지표가 증가해야 하거나 감소 확률이 90% 이상이고 지표가 감소해야 합니다.
- Worse(나쁨) - 감소 확률이 90% 이상이고 지표가 증가해야 하거나 증가 확률이 90% 이상이고 지표가 감소해야 합니다.

대시보드에서 출시 결과 보기

실험이 진행 중이거나 완료된 후에 실험의 진행 상황과 지표 결과를 확인할 수 있습니다.

출시 진행 상황 및 결과 확인

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 출시가 포함된 프로젝트 이름을 선택합니다.
4. 출시(Launch) 탭을 선택합니다.
5. 출시 이름을 선택합니다.
6. 각 단계의 출시 단계와 트래픽 할당을 보려면 출시(Launch) 탭을 선택합니다.
7. 시간별 각 변형에 할당된 사용자 세션 수를 확인하고 출시의 각 변형에 대한 성능 지표를 보려면 모니터링(Monitoring) 탭을 선택합니다.

   이 뷰에는 출시 동안 출시 경보가 ALARM 상태로 전환되었는지 여부도 표시됩니다.
8. 이 출시의 변형, 지표, 경보 및 태그를 보려면 구성(Configuration) 탭을 선택합니다.

대시보드에서 실험 결과 보기

실험이 진행 중이거나 완료된 후에 실험의 통계 결과를 확인할 수 있습니다. 실험 결과는 실험을 시작한 후 63일까지 제공됩니다. 그 이후에는 CloudWatch 데이터 보존 정책으로 인해 사용할 수 없습니다.

각 변동에 최소 100개의 이벤트가 있을 때까지 통계 결과가 표시되지 않습니다.

Evidently는 실험이 끝날 때 추가 offline p-값 분석을 수행합니다. 실험 중에 사용된 anytime p-값이 통계적 유의성을 발견하지 못하는 경우에 offline p-값 분석을 통해 통계적 유의성을 발견할 수 있습니다.

CloudWatch Evidently가 실험 결과를 계산하는 방법에 대한 자세한 내용을 알아보려면 Evidently에서 결과를 계산하는 방법 섹션을 참조하세요.

실험 결과 확인

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 실험이 포함된 프로젝트 이름을 선택합니다.
4. 실험(Experiments) 탭을 선택합니다.
5. 실험 이름을 선택한 다음 결과(Results) 탭을 선택합니다.
6. 변형 성능(Variation performance)에는 표시할 실험 통계를 선택할 수 있는 제어 옵션이 있습니다. 둘 이상의 통계를 선택한 경우 Evidently는 각 통계에 대한 그래프와 표를 표시합니다.

   각 그래프와 표에는 지금까지 진행한 실험 결과가 표시됩니다.

각 그래프에는 다음과 같은 결과가 표시될 수 있습니다. 그래프 오른쪽에 있는 제어 기능을 사용하여 다음 항목 중 어떤 항목을 표시할지 결정할 수 있습니다.

- 각 변형에 대해 기록된 사용자 세션 이벤트 수.
- 각 변형에 대해 그래프 상단에서 선택된 지표의 평균값.
- 실험의 통계적 유의성. 그래프의 상단에서 선택한 지표에 대한 차이를 기본 변형 및 다른 변형과 비교합니다.
- 각 변형과 기본 변형 간 선택한 지표의 차이에 대한 95% 신뢰 구간.

표에는 각 변동에 대한 행이 표시됩니다. 기본값이 아닌 각 변형에 대해 Evidently는 통계적으로 유의한 결과를 선언하는 데 충분한 데이터를 수신했는지 여부를 표시합니다. 또한 통계값에서 변형의 개선이 95% 신뢰 수준에 도달했는지 여부도 보여줍니다.

마지막으로 결과(Result) 열에서 Evidently는 이 통계에 따라 어떤 변형이 가장 잘 수행되는지 또는 결과가 결정적이지 않은지에 대한 권장 사항을 제공합니다.

CloudWatch Evidently가 데이터를 수집하고 저장하는 방법

Amazon CloudWatch Evidently는 고객이 실험 및 출시를 실행할 수 있도록 프로젝트 구성과 관련된 데이터를 수집하고 저장합니다. 데이터에는 다음이 포함됩니다.

- 프로젝트, 기능, 출시 및 실험에 관한 메타데이터
- 지표 이벤트
- 평가 데이터

리소스 메타데이터는 Amazon DynamoDB에 저장됩니다. 데이터는 AWS 소유 키를 사용하여 기본적으로 저장 시 암호화됩니다. 이러한 키는 AWS 서비스가 여러 AWS 계정에서 사용하기 위해 소유하고 관리하는 AWS KMS 키 모음입니다. 고객은 이러한 키를 사용하는 것을 보거나, 관리하거나, 감사할 수 없습니다. 또한 고객은 데이터를 암호화하는 키를 보호하기 위해 조치를 취하거나 프로그램을 변경할 필요가 없습니다.

자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS 소유 키 섹션을 참조하세요.

Evidently 지표 이벤트와 평가 이벤트는 고객 소유의 위치로 직접 전달됩니다.

전송 중인 데이터는 HTTPS를 사용하여 자동으로 암호화됩니다. 이 데이터는 고객 소유의 위치로 전달됩니다.

또한 Amazon Simple Storage Service나 Amazon CloudWatch Logs에 평가 이벤트를 저장하도록 선택할 수 있습니다. 이러한 서비스에서 데이터를 보호하는 방법에 대한 자세한 내용은 Amazon S3 기본 버킷 암호화 활성화 및 AWS KMS를 사용하여 CloudWatch Logs의 로그 데이터 암호화를 참조하세요.

데이터 불러오기

CloudWatch Evidently API를 사용하여 데이터를 불러올 수 있습니다. 프로젝트 데이터를 불러오려면 GetProject 또는 ListProjects를 사용합니다.

기능 데이터를 불러오려면 GetFeature 또는 ListFeatures를 사용합니다.

출시 데이터를 불러오려면 GetLaunch 또는 ListLaunches를 사용합니다.

실험 데이터를 불러오려면 GetExperiment, ListExperiments 또는 GetExperimentResults를 사용합니다.

데이터 수정 및 삭제

CloudWatch Evidently API를 사용하여 데이터를 수정하고 삭제할 수 있습니다. 프로젝트 데이터의 경우 UpdateProject 또는 DeleteProject를 사용합니다.

기능 데이터의 경우 UpdateFeature 또는 DeleteFeature를 사용합니다.

출시 데이터의 경우 UpdateLaunch 또는 DeleteLaunch를 사용합니다.

실험 데이터의 경우UpdateExperiment 또는 DeleteExperiment을 사용합니다.

Evidently에 대한 서비스 연결 역할 사용

CloudWatch Evidently는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Evidently에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Evidently에서 사전 정의하며 서비스에서 사용자 대신 다른 AWS 서비스를 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Evidently를 더 쉽게 설정할 수 있습니다. Evidently에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Evidently만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제해야만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 Evidently 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용을 알아보려면 AWS IAM으로 작업하는 서비스를 참조하고 Service-linked roles(서비스 연결 역할) 열에 Yes(예)가 표시된 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

Evidently에 대한 서비스 연결 역할 권한

Evidently는 AWSServiceRoleForCloudWatchEvidently라는 서비스 연결 역할을 사용합니다. 이 역할을 통해 CloudWatch는 고객을 대신하여 연결된 AWS 리소스를 관리할 수 있습니다.

AWSServiceRoleForCloudWatchEvidently 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `CloudWatch Evidently`

AmazonCloudWatchEvidentlyServiceRolePolicy라는 역할 권한 정책은 Evidently가 지정된 리소스에 대해 다음 작업을 완료하도록 허용합니다.

- 작업: Evidently 씩 클라이언트에 대한 `appconfig:StartDeployment`, `appconfig:StopDeployment`, `appconfig:ListDeployments` 및 `appconfig:TagResource`

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 작성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Evidently에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API API에서 Evidently 씩 클라이언트를 사용하기 시작하면 Evidently가 서비스 연결 역할을 생성합니다.

이 서비스 연결 역할을 삭제한 다음 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. Evidently 씩 클라이언트를 사용하기 시작하면 Evidently가 다시 서비스 연결 역할을 생성합니다.

Evidently에 대한 서비스 연결 역할 편집

Evidently에서는 AWSServiceRoleForCloudWatchEvidently 서비스 연결 역할을 편집할 수 없습니다. 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 편집을 참조하세요.

Evidently에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 개체가 없도록 합니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다. 씩 클라이언트를 사용하는 모든 Evidently 프로젝트를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려 할 때 Evidently 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

AWSServiceRoleForCloudWatchEvidently에서 사용하는 Evidently 리소스 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 애플리케이션 모니터링(Application monitoring), Evidently를 선택합니다.
3. 프로젝트 목록에서 씩 클라이언트를 사용한 프로젝트 옆의 확인란을 선택합니다.
4. Project actions(프로젝트 작업), Delete project(프로젝트 삭제)를 선택합니다.

IAM을 사용하여 수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForCloudWatchEvidently 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스에 연결 역할 삭제를 참조하세요.

Evidently 서비스 연결 역할이 지원되는 리전

Evidently는 서비스가 제공되는 모든 리전에서 서비스 연결 역할을 사용하도록 지원합니다. 자세한 내용을 알아보려면 AWS service endpoints(서비스 엔드포인트)를 참조하세요.

CloudWatch Evidently 할당량

CloudWatch Evidently는 다음과 같은 할당량이 있습니다.

Resource	기본 할당량
프로젝트	계정별 리전당 50개 할당량 증가를 요청할 수 있습니다.
세그먼트	계정별 리전당 500개 할당량 증가를 요청할 수 있습니다.
프로젝트당 할당량	• 총 기능 100개 • 총 출시 500개 • 실행 중인 출시 50개 • 총 실험 500개 • 실행 중인 실험 50개 이러한 모든 할당량에 대해 할당량 증가 요청을 할 수 있습니다.
API 할당량(모든 할당량은 리전당 기준임)	• PutProjectEvents: 미국 동부(버니지아 북부), 미국 서부(오레곤) 및 유럽(아일랜드)의 경우 1,000TPS(초당 트랜잭션 수). 기타 모든 리전의 경우 200TPS입니다. • EvaluateFeature: 미국 동부(버니지아 북부), 미국 서부(오레곤) 및 유럽(아일랜드)의 경우 1,000TPS. 기타 모든 리전의 경우 200TPS입니다. • BatchEvaluateFeature: 50TPS • 생성, 읽기, 업데이트, 삭제(CRUD) API: 모든 CRUD API를 합쳐서 10TPS 이러한 모든 할당량에 대해 할당량 증가 요청을 할 수 있습니다.

자습서: Evidently 샘플 애플리케이션으로 A/B 테스트

이 섹션에서는 A/B 테스트를 위해 Amazon CloudWatch Evidently를 사용하는 방법에 대한 자습서를 제공합니다. 이 자습서에서는 간단한 반응 애플리케이션인 Evidently 샘플 애플리케이션입니다. 샘플 앱은 `showDiscount` 기능 여부를 표시하도록 구성됩니다. 이 기능이 사용자에게 표시되면 쇼핑 웹 사이트에 표시된 가격은 20% 할인된 가격으로 표시됩니다.

이 자습서에서는 다른 사용자가 아닌 일부 사용자에게 할인을 표시하는 것 외에도 두 가지 변형에서 페이지 로드 시간 지표를 수집하도록 Evidently를 설정합니다.

> **Warning**
>
> 이 시나리오에서는 프로그래밍 방식 액세스 권한과 장기 보안 인증이 있는 IAM 사용자가 필요하며 이는 보안 위험을 내포합니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다. 필요한 경우 액세스 키를 업데이트할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 액세스 키 업데이트를 참조하세요.

1단계: 샘플 애플리케이션 다운로드

먼저 Evidently 샘플 애플리케이션을 다운로드합니다.

샘플 애플리케이션을 다운로드하려면

1. 다음 Amazon S3 버킷에서 샘플 애플리케이션을 다운로드합니다.

   ```
   https://evidently-sample-application.s3.us-west-2.amazonaws.com/evidently-sample-shopping-app.zip
   ```

2. 패키지의 압축을 풉니다.

2단계: Evidently 엔드포인트 추가 및 자격 증명 설정

그런 다음, 다음 예제와 같이 Evidenty에 대한 리전 및 엔드포인트를 샘플 앱 패키지의 `src` 디렉터리에 있는 `config.js` 파일에 추가합니다.

```
evidently: {
    REGION: "us-west-2",
```

```
    ENDPOINT: "https://evidently.us-west-2.amazonaws.com (https://evidently.us-
west-2.amazonaws.com/)",
},
```

또한 애플리케이션에 CloudWatch Evidenty를 호출할 수 있는 권한이 있는지 확인해야 합니다.

샘플 앱에 Evidently를 호출할 수 있는 권한을 부여하려면

1. AWS 계정에 연동합니다.
2. IAM 사용자를 생성하고 AmazonCloudWatchEvidentlyFullAccess 정책을 이 사용자에 연결합니다.
3. 다음 단계에서 필요하므로 IAM 사용자의 액세스 키 ID 및 비밀 액세스 키를 기록해 둡니다.
4. 다음 예와 같이 이 섹션의 앞부분에서 수정한 동일한 `config.js` 파일에 액세스 키 ID 및 비밀 액세스 키의 값을 입력합니다.

```
credential: {
   accessKeyId: "Access key ID",
   secretAccessKey: "Secret key"
}
```

> **Important**
>
> 이 단계를 사용하여 샘플 앱을 최대한 간단하게 할 수 있습니다. 실제 프로덕션 애플리케이션에 IAM 사용자 자격 증명을 넣지 않는 것이 좋습니다. 대신 인증을 위해 Amazon Cognito를 사용하는 것이 좋습니다. 자세한 내용은 Amazon Cognito와 웹 및 모바일 앱 통합을 참조하세요.

3단계: 기능 평가를 위한 코드 설정

CloudWatch Evidently를 사용하여 기능을 평가할 때는 EvaluateFeature 작업을 사용하여 각 사용자 세션에 대한 기능 변형을 임의로 선택해야 합니다. 이 작업은 실험에서 지정한 비율에 따라 기능의 각 변형에 사용자 세션을 할당합니다.

북스토어 데모 앱에 대한 기능 평가 코드 설정

1. 샘플 앱이 Evidenty를 호출할 수 있도록 클라이언트 빌더를 `src/App.jsx`파일에 추가합니다.

```
import Evidently from 'aws-sdk/clients/evidently';
import config from './config';
```

```
const defaultClientBuilder = (
    endpoint,
    region,
) => {
    const credentials = {
        accessKeyId: config.credential.accessKeyId,
        secretAccessKey: config.credential.secretAccessKey
    }
    return new Evidently({
        endpoint,
        region,
        credentials,
    });
};
```

2. 다음을 const App 코드 섹션에 추가하여 클라이언트를 시작합니다.

```
  if (client == null) {
      client = defaultClientBuilder(
        config.evidently.ENDPOINT,
        config.evidently.REGION,
      );
```

3. 다음 코드를 추가하여 evaluateFeatureRequest를 구성합니다. 이 코드는 이 자습서의 뒷부분에서 권장하는 프로젝트 이름과 기능 이름을 미리 채웁니다. Evidently 콘솔에서 해당 프로젝트와 기능 이름도 지정하면 고유한 프로젝트와 기능 이름을 대체할 수 있습니다.

```
const evaluateFeatureRequest = {
              entityId: id,
              // Input Your feature name
              feature: 'showDiscount',
              // Input Your project name'
              project: 'EvidentlySampleApp',
            };
```

4. 기능 평가를 위해 Evidently를 호출할 코드를 추가합니다. 요청이 전송되면 Evidently가 사용자 세션을 무작위로 할당하여 showDiscount 기능 여부를 확인합니다.

```
client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
    if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
    }
```

```
    getPageLoadTime()
 })
```

4단계: 실험 지표에 대한 코드 설정

사용자 지정 지표의 경우 Evidently의 PutProjectEvents API를 사용하여 Evidently로 지표 결과를 전송합니다. 다음 예제에서는 사용자 지정 지표를 설정하고 실험 데이터를 Evidently로 전송하는 방법을 보여줍니다.

다음 함수를 추가해 페이지 로드 시간을 계산하고 PutProjectEvents를 사용하여 지표 값을 Evidently로 전송합니다. 다음 기능을 Home.tsx에 추가하고 EvaluateFeature API에서 이 함수를 호출합니다.

```
const getPageLoadTime = () => {
    const timeSpent = (new Date().getTime() - startTime.getTime()) * 1.000001;
    const pageLoadTimeData = `{
      "details": {
        "pageLoadTime": ${timeSpent}
      },
      "UserDetails": { "userId": "${id}", "sessionId": "${id}"}
    }`;
    const putProjectEventsRequest = {
      project: 'EvidentlySampleApp',
      events: [
        {
          timestamp: new Date(),
          type: 'aws.evidently.custom',
          data: JSON.parse(pageLoadTimeData)
        },
      ],
    };
    client.putProjectEvents(putProjectEventsRequest).promise();
  }
```

다음은 App.js 파일을 다운로드 한 후 수행한 모든 편집 후의 파일 형식입니다.

```
import React, { useEffect, useState } from "react";
import { BrowserRouter as Router, Switch } from "react-router-dom";
import AuthProvider from "contexts/auth";
import CommonProvider from "contexts/common";
import ProductsProvider from "contexts/products";
```

```
import CartProvider from "contexts/cart";
import CheckoutProvider from "contexts/checkout";
import RouteWrapper from "layouts/RouteWrapper";
import AuthLayout from "layouts/AuthLayout";
import CommonLayout from "layouts/CommonLayout";
import AuthPage from "pages/auth";
import HomePage from "pages/home";
import CheckoutPage from "pages/checkout";
import "assets/scss/style.scss";
import { Spinner } from 'react-bootstrap';

import Evidently from 'aws-sdk/clients/evidently';
import config from './config';

const defaultClientBuilder = (
  endpoint,
  region,
) => {
  const credentials = {
    accessKeyId: config.credential.accessKeyId,
    secretAccessKey: config.credential.secretAccessKey
  }
  return new Evidently({
    endpoint,
    region,
    credentials,
  });
};

const App = () => {
  const [isLoading, setIsLoading] = useState(true);
  const [startTime, setStartTime] = useState(new Date());
  const [showDiscount, setShowDiscount] = useState(false);
  let client = null;
  let id = null;

  useEffect(() => {
    id = new Date().getTime().toString();
    setStartTime(new Date());
    if (client == null) {
      client = defaultClientBuilder(
        config.evidently.ENDPOINT,
        config.evidently.REGION,
      );
```

```
    }
    const evaluateFeatureRequest = {
      entityId: id,
      // Input Your feature name
      feature: 'showDiscount',
      // Input Your project name'
      project: 'EvidentlySampleApp',
    };

    // Launch
    client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
      if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
      }
    });

    // Experiment
    client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
      if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
      }
      getPageLoadTime()
    })

    setIsLoading(false);
  },[]);

  const getPageLoadTime = () => {
    const timeSpent = (new Date().getTime() - startTime.getTime()) * 1.000001;
    const pageLoadTimeData = `{
      "details": {
        "pageLoadTime": ${timeSpent}
      },
      "UserDetails": { "userId": "${id}", "sessionId": "${id}"}
    }`;
    const putProjectEventsRequest = {
      project: 'EvidentlySampleApp',
      events: [
        {
          timestamp: new Date(),
          type: 'aws.evidently.custom',
          data: JSON.parse(pageLoadTimeData)
        },
      ],
```

```
    };
    client.putProjectEvents(putProjectEventsRequest).promise();
  }
  return (
    !isLoading? (
    <AuthProvider>
      <CommonProvider>
        <ProductsProvider>
          <CartProvider>
            <CheckoutProvider>
              <Router>
                <Switch>
                  <RouteWrapper
                    path="/"
                    exact
                    component={() => <HomePage showDiscount={showDiscount}/>}
                    layout={CommonLayout}
                  />
                  <RouteWrapper
                    path="/checkout"
                    component={CheckoutPage}
                    layout={CommonLayout}
                  />
                  <RouteWrapper
                    path="/auth"
                    component={AuthPage}
                    layout={AuthLayout}
                  />
                </Switch>
              </Router>
            </CheckoutProvider>
          </CartProvider>
        </ProductsProvider>
      </CommonProvider>
    </AuthProvider> ) : (
      <Spinner animation="border" />
    )
  );
};

export default App;
```

사용자가 샘플 앱을 방문할 때마다 분석을 위해 사용자 지정 지표가 Evidently로 전송됩니다. Evidently는 각 지표를 분명히 분석하고 Evidently 대시보드에 결과를 실시간으로 표시합니다. 다음 예제에서는 지표 페이로드를 보여줍니다.

```
[ {"timestamp": 1637368646.468, "type": "aws.evidently.custom", "data": "{\"details
\":{\"pageLoadTime\":2058.002058},\"userDetails\":{\"userId\":\"1637368644430\",
\"sessionId\":\"1637368644430\"}}" } ]
```

5단계: 프로젝트, 기능 및 실험 생성

다음으로 CloudWatch Evidently 콘솔에서 프로젝트, 기능 및 실험을 생성합니다.

이 튜토리얼에서 프로젝트, 기능 및 실험 만들기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. 프로젝트 생성(Create project)을 선택하고 필드를 입력합니다. 샘플의 프로젝트 이름에 **EvidentlySampleApp**을 사용하여 올바르게 작동하도록 합니다. 평가 이벤트 스토리지(Evaluation event storage)에 평가 이벤트 저장 안 함(Don't store Evaluation events)을 선택합니다.

   필드에 내용을 입력한 후 프로젝트 생성(Create profile)을 선택합니다.

   자세한 내용은 새 프로젝트 만들기를 참조하세요.
4. 프로젝트를 만든 후 해당 프로젝트에 기능을 생성합니다. **showDiscount** 기능에 이름을 지정합니다. 이 기능에서는 **Boolean** 유형의 2가지 변형을 만듭니다. **False** 값을 가진 첫 번째 변형인 **disable**의 이름을 지정하고 **True** 값을 가진 두 번째 변형인 **enable**의 이름을 지정합니다.

   기능 생성에 대한 자세한 내용은 프로젝트에 기능 추가 섹션을 참조하세요.
5. 기능 생성을 마친 후 프로젝트에서 실험을 생성합니다. **pageLoadTime** 실험 이름을 지정합니다.

   이 실험에서는 테스트 중인 페이지의 페이지 로드 시간을 측정하는 pageLoadTime이라는 사용자 지정 지표를 사용합니다. 실험용 사용자 지정 지표는 Amazon EventBridge를 사용해서 생성됩니다. EventBridge에 대한 자세한 내용은 “ Amazon EventBridge란 무엇인가요?”를 참조하세요.

   사용자 지정 지표를 만들려면 실험을 생성할 때 다음을 수행합니다.

   - 지표(Metrics)의 지표 소스(Metric source)에서 사용자 지정 지표(Custom metrics)를 선택합니다.

- 지표 이름(Metric name)에서 **pageLoadTime**을 입력합니다.
- 목표(Goal)에서 감소(Decrease)를 선택합니다. 이는 기능의 최적 변형을 나타내기 위해 이 지표 값이 낮아야 한다는 것을 나타냅니다.
- 지표 규칙(Metric rule)에서 다음을 입력합니다.
  - Entity ID(개체 ID)에 **UserDetails.userId**를 입력합니다.
  - 값 키(Value key)에서 **details.pageLoadTime**을 입력합니다.
  - 단위(Units)에서 **ms**를 입력합니다.
- 지표 추가(Add metric)를 입력합니다.

대상(Audiences)에서 모든 사용자가 실험에 입력되도록 하려면 100%를 선택합니다. 변형 간 트래픽 분할을 각각 50%로 설정합니다.

그런 다음, 실험을 생성하려면 실험 생성(Create Experiment)을 선택합니다. 실험을 생성한 후, Evidently에 시작하도록 알리기 전까지는 시작하지 않습니다.

6단계: 실험 시작 및 CloudWatch Evidently 테스트

마지막 단계에서는 실험을 시작하고 샘플 앱을 시작합니다.

자습서 실험을 시작하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, Evidently를 선택합니다.
3. EvidentlySampleApp 프로젝트를 선택합니다.
4. 실험(Experiments) 탭을 선택합니다.
5. pageLoadTime 옆에 있는 버튼을 선택하고 작업(Actions)에서 실험 시작(Start experiment)을 선택합니다.
6. 실험을 종료할 시간을 선택합니다.
7. 실험 시작(Start experiment)을 선택합니다.

   실험이 즉시 시작됩니다.

그리고 다음 명령을 사용하여 Evidently 샘플 앱을 시작합니다.

```
npm install -f && npm start
```

앱이 시작되면 테스트 중인 두 가지 기능 변형 중 하나에 할당됩니다. 한 변형에는 "20% 할인"이 표시되고 다른 변형에는 표시되지 않습니다. 다양한 변형을 보려면 페이지를 계속 새로 고칩니다.

> **Note**
>
> Evidently에는 고정 평가가 있습니다. 기능 평가는 결정적입니다. 즉 동일한 `entityId` 및 기능을 사용하면 사용자가 항상 동일한 변형 할당을 받게 됩니다. 변형 할당이 변경되는 유일한 시간은 엔터티가 재정의에 추가되거나 실험 트래픽이 전화 접속 연결이 될 때입니다.
> 그러나 샘플 앱 자습서를 쉽게 사용할 수 있도록 페이지를 새로 고칠 때마다 Evidently가 샘플 앱 기능 평가를 다시 할당하므로 재정의를 추가하지 않고도 두 변형을 모두 경험할 수 있습니다.

문제 해결

npm 버전 6.14.14를 사용하는 것이 좋습니다. 샘플 앱을 구축하거나 시작하는 데 오류가 있고 다른 버전의 npm을 사용하는 경우 다음을 수행합니다.

npm 버전 6.14.14를 설치하려면

1. 브라우저를 사용하여 https://nodejs.org/download/release/v14.17.5/에 연결합니다.
2. node-v14.17.5.pkg를 다운로드하고 이 pkg를 실행하여 npm을 설치합니다.

   `webpack not found` 오류가 나타나는 경우 `evidently-sample-shopping-app` 폴더로 이동하여 다음을 시도합니다.

   a. `package-lock.json` 삭제
   b. `yarn-lock.json` 삭제
   c. `node_modules` 삭제
   d. `package.json`에서 webpack 종속성 삭제
   e. 다음을 실행합니다.

   ```
   npm install -f && npm
   ```

CloudWatch RUM 사용

CloudWatch RUM을 사용하면 실제 사용자 모니터링을 수행하여 실제 사용자 세션에서 웹 애플리케이션 성능에 대한 클라이언트 측 데이터를 거의 실시간으로 수집하고 볼 수 있습니다. 시각화하고 분석할 수 있는 데이터에는 페이지 로드 시간, 클라이언트 측 오류 및 사용자 동작이 포함됩니다. 이 데이터에서 모든 데이터가 함께 집계된 것을 볼 수 있으며 고객이 사용하는 브라우저 및 디바이스별 분석도 확인할 수 있습니다.

수집된 데이터를 사용하여 클라이언트 측 성능 문제를 신속하게 식별하고 디버깅할 수 있습니다. CloudWatch RUM을 사용하면 애플리케이션 성능 이상을 시각화하고 오류 메시지, 스택 추적 및 사용자 세션 등의 관련 디버깅 데이터를 찾을 수 있습니다. 또한 RUM을 사용하여 사용자 수, 위치 정보 및 사용된 브라우저 등 최종 사용자의 영향 범위를 파악할 수 있습니다.

CloudWatch RUM에 대해 수집하는 최종 사용자 데이터는 30일 동안 보존된 다음 자동으로 삭제됩니다. RUM 이벤트를 더 오랫동안 유지하려면 앱 모니터에서 이벤트의 복사본을 계정의 CloudWatch Logs에 보내도록 선택할 수 있습니다. 그런 다음, 해당 로그 그룹의 보존 기간을 조정할 수 있습니다.

RUM을 사용하려면 앱 모니터를 생성해서 일부 정보를 제공합니다. RUM은 사용자가 애플리케이션에 붙여넣을 수 있도록 JavaScript 코드 조각을 생성합니다. 코드 조각은 RUM 웹 클라이언트 코드를 가져옵니다. RUM 웹 클라이언트는 미리 구축된 대시보드에 표시되는 애플리케이션 사용자 세션의 비율에서 데이터를 캡처합니다. 데이터를 수집할 사용자 세션의 비율을 지정할 수 있습니다.

CloudWatch RUM은 애플리케이션 서비스, 클라이언트, Synthetics canary 및 서비스 종속성을 검색하고 모니터링할 수 있는 Application Signals와 통합됩니다. Application Signals를 사용하여 서비스의 목록 또는 시각적 맵을 확인하고, 서비스 수준 목표(SLO)를 기준으로 상태 지표를 확인하고, 더 자세한 문제 해결을 위해 상관관계가 있는 X-Ray 트레이스를 드릴다운할 수 있습니다. Application Signals에서 RUM 클라이언트 페이지 요청을 보려면 앱 모니터를 생성하거나 RUM 웹 클라이언트를 수동으로 구성하여 X-Ray 활성 추적을 켭니다. RUM 클라이언트는 서비스에 연결된 서비스 맵과 직접적으로 호출하는 서비스의 서비스 세부 정보 페이지에 표시됩니다.

RUM 웹 클라이언트는 오픈 소스입니다. 자세한 내용은 CloudWatch RUM 웹 클라이언트를 참조하세요.

성능 고려 사항

이 섹션에서는 CloudWatch RUM 사용 시 성능 고려 사항에 관해 설명합니다.

- 로드 성능 영향: CloudWatch RUM 웹 클라이언트가 JavaScript 모듈로 웹 애플리케이션에 설치되거나 콘텐츠 전송 네트워크(CDN)에서 비동기적으로 웹 애플리케이션에 로드됩니다. 애플리케이션의

로드 프로세스는 차단하지 않습니다. CloudWatch RUM은 애플리케이션 로드 시간에 감지할 수 있는 영향이 미치지 않도록 설계되었습니다.

- 런타임 영향: RUM 웹 클라이언트는 처리를 수행하여 RUM 데이터를 기록하고 CloudWatch RUM 서비스에 전달합니다. 이벤트가 자주 발생하지 않고 처리량이 적기 때문에 CloudWatch RUM은 애플리케이션 성능에 감지할 수 있는 영향을 주지 않도록 설계되었습니다.
- 네트워크 영향: RUM 웹 클라이언트는 정기적으로 CloudWatch RUM 서비스로 데이터를 전송합니다. 데이터는 애플리케이션이 실행되는 동안 정기적으로 전달되며 브라우저가 애플리케이션을 언로드하기 직전에 전달됩니다. 브라우저가 애플리케이션을 언로드하기 직전에 전송된 데이터는 애플리케이션의 언로드 시간에 감지 가능한 영향을 미치지 않도록 설계된 비콘으로 전송됩니다.

RUM 요금

CloudWatch RUM을 사용하면 CloudWatch RUM이 수신하는 모든 RUM 이벤트에 대해 요금이 부과됩니다. RUM 웹 클라이언트를 사용하여 수집된 각 데이터 항목은 RUM 이벤트로 간주됩니다. RUM 이벤트의 예로는 페이지 보기, JavaScript 오류 및 HTTP 오류가 있습니다. 각 앱 모니터에서 수집되는 이벤트 유형에 대한 옵션이 있습니다. 옵션을 활성화하거나 비활성화하여 성능 원격 측정 이벤트, JavaScript 오류, HTTP 오류 및 X-Ray 추적을 수집할 수 있습니다. 이러한 옵션 선택에 대한 자세한 정보는 2단계: 앱 모니터 생성 및 CloudWatch RUM 웹 클라이언트에서 수집한 정보 섹션을 참조하세요. 요금에 대한 자세한 정보는 Amazon CloudWatch 요금을 참조하세요.

리전 가용성

CloudWatch RUM은 현재 다음 리전에서 사용 가능합니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(자카르타)
- 아시아 태평양(뭄바이)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(멜버른)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)

- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스페인)
- 유럽(스톡홀름)
- 유럽(취리히)
- 중동(바레인)
- 중동(UAE)
- 남아메리카(상파울루)

주제

- CloudWatch RUM 문제 해결

CloudWatch RUM을 사용하는 IAM 정책

CloudWatch RUM을 완벽하게 관리하려면 AmazonCloudWatchRUMFullAccess IAM 정책을 포함하는 IAM 사용자 또는 역할로 로그인해야 합니다. 또한 다음과 같은 다른 정책이나 권한이 필요할 수 있습니다.

- 권한 부여를 위해 새 Amazon Cognito 자격 증명 풀을 생성하는 앱 모니터를 만들려면 관리자(Admin) IAM 역할 또는 AdministratorAccess IAM 정책이 있어야 합니다.
- CloudWatch Logs로 데이터를 전송하는 앱 모니터를 생성하려면 다음 권한을 가진 IAM 역할 또는 정책에 로그온해야 합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:PutResourcePolicy"
    ],
    "Resource": [
        "*"
    ]
}
```

CloudWatch RUM 데이터를 조회해야 하지만 CloudWatch RUM 리소스를 만들 필요는 없는 다른 사용자에게 AmazonCloudWatchRUMReadOnlyAccess 정책을 부여할 수 있습니다.

CloudWatch RUM을 사용하도록 애플리케이션 설정

이 섹션의 단계를 사용하여 실제 사용자 세션에서 성능 데이터를 수집하기 위해 CloudWatch RUM을 사용하도록 애플리케이션을 설정합니다.

주제

- 1단계: 애플리케이션에서 데이터를 AWS로 전송하도록 권한 부여
- 2단계: 앱 모니터 생성
- (선택 사항) 3단계: 코드 조각을 수동으로 수정하여 CloudWatch RUM 웹 클라이언트를 구성합니다.
- 4단계: 애플리케이션에 코드 조각 삽입
- 5단계: 사용자 이벤트를 생성한 앱 모니터 설정 테스트

1단계: 애플리케이션에서 데이터를 AWS로 전송하도록 권한 부여

CloudWatch RUM을 사용하려면 애플리케이션에 권한 부여가 있어야 합니다.

권한 부여를 설정할 수 있는 3가지 옵션이 있습니다.

- CloudWatch RUM에서 애플리케이션을 위한 새 Amazon Cognito 자격 증명 풀을 생성할 수 있습니다. 이 방법은 설정하는 데 최소한의 노력이 필요합니다. 이 방법이 기본 옵션입니다.

  자격 증명 풀에는 인증되지 않은 자격 증명이 포함됩니다. 이렇게 하면 CloudWatch RUM 웹 클라이언트가 애플리케이션 사용자를 인증하지 않고도 CloudWatch RUM에 데이터를 전송할 수 있습니다.

  Amazon Cognito 자격 증명 풀에는 연결된 IAM 역할이 있습니다. Amazon Cognito 인증되지 않은 자격 증명을 사용하면 웹 클라이언트가 CloudWatch RUM에 데이터를 전송할 수 있는 권한이 부여된 IAM 역할을 맡을 수 있습니다.
- 기존 Amazon Cognito 자격 증명 풀을 사용합니다. 이 경우 자격 증명 풀에 연결된 IAM 역할도 수정해야 합니다. 인증되지 않은 사용자를 지원하는 자격 증명 풀에 이 옵션을 사용합니다. 동일한 리전의 자격 증명 풀만 사용할 수 있습니다.
- 이미 설정한 기존 자격 증명 공급자의 인증을 사용합니다. 이 경우 자격 증명 공급자로부터 자격 증명을 가져와야 하며 애플리케이션은 이러한 자격 증명을 RUM 웹 클라이언트에 전달해야 합니다.

  인증된 사용자만 지원하는 자격 증명 풀에 이 옵션을 사용합니다.

다음 섹션에서는 이러한 옵션에 대해 상세히 알아봅니다.

CloudWatch RUM에서 새 Amazon Cognito 자격 증명 풀을 생성합니다.

이 옵션은 가장 간단한 설정 옵션이며, 이 옵션을 선택하면 추가 설정 단계가 필요하지 않습니다. 이 옵션을 사용하려면 관리 권한이 있어야 합니다. 자세한 내용은 CloudWatch RUM을 사용하는 IAM 정책 단원을 참조하십시오.

이 옵션을 사용하면 CloudWatch RUM이 다음 리소스를 생성합니다.

- 새로운 Amazon Cognito 자격 증명 풀
- 인증되지 않은 Amazon Cognito 자격 증명. 이를 통해 RUM 웹 클라이언트는 애플리케이션 사용자를 인증하지 않고 IAM 역할을 맡을 수 있습니다.
- RUM 웹 클라이언트가 맡을 IAM 역할. 이 역할에 연결된 IAM 정책을 통해 앱 모니터 리소스가 포함된 `PutRumEvents` API를 사용할 수 있습니다. 즉, RUM 웹 클라이언트가 데이터를 RUM으로 보낼 수 있습니다.

RUM 웹 클라이언트는 Amazon Cognito 자격 증명을 사용하여 AWS 자격 증명을 획득합니다. AWS 자격 증명은 IAM 역할과 연결됩니다. IAM 역할은 AppMonitor 리소스로 PutRumEvents를 사용할 수 있는 권한이 있습니다.

Amazon Cognito는 애플리케이션이 CloudWatch RUM에 데이터를 전송할 수 있도록 필요한 보안 토큰을 전송합니다. CloudWatch RUM이 생성하는 JavaScript 코드 조각에는 인증을 활성화하기 위한 다음 줄이 포함됩니다.

```
   {
    identityPoolId: [identity pool id],  // e.g., 'us-west-2:EXAMPLE4a-66f6-4114-902a-
EXAMPLEbad7'
   }
);
```

기존 Amazon Cognito 자격 증명 풀을 사용

기존 Amazon Cognito 자격 증명 풀을 사용하도록 선택한 경우 애플리케이션을 CloudWatch RUM에 추가할 때 자격 증명 풀을 지정합니다. 풀은 인증되지 않은 자격 증명에 대한 액세스를 활성화할 수 있도록 지원해야 합니다. 동일한 리전의 자격 증명 풀만 사용할 수 있습니다.

또한 이 자격 증명 풀과 관련된 IAM 역할에 연결된 IAM 정책에 다음 권한을 추가해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rum:PutRumEvents",
            "Resource": "arn:aws:rum:[region]:[accountid]:appmonitor/[app monitor
 name]"
        }
    ]
}
```

Amazon Cognito는 애플리케이션이 CloudWatch RUM에 액세스할 수 있도록 필요한 보안 토큰을 전송합니다.

서드 파티 공급자

서드 파티 공급자로부터 프라이빗 인증을 사용하는 경우 자격 증명 공급자로부터 자격 증명을 가져와 AWS로 전달해야 합니다. 이를 수행하는 가장 좋은 방법은 보안 토큰 공급 업체를 이용하는 것입니다.

AWS Security Token Service와 함께 Amazon Cognito를 포함한 모든 보안 토큰 공급 업체를 이용할 수 있습니다. AWS STS에 대한 자세한 내용은 AWS Security Token Service API 참조 소개를 참조하세요.

이 시나리오에서 Amazon Cognito를 토큰 공급 업체로 사용하려면 인증 공급자와 함께 작동하도록 Amazon Cognito를 구성할 수 있습니다. 자세한 내용은 Amazon Cognito 자격 증명 풀 시작(페더레이션 자격 증명)을 참조하세요.

자격 증명 공급자와 함께 작동하도록 Amazon Cognito를 구성한 후에는 다음을 수행해야 합니다.

- 다음 권한을 가진 IAM 역할을 만듭니다. 애플리케이션은 이 역할을 사용하여 AWS에 액세스합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rum:PutRumEvents",
            "Resource": "arn:aws:rum:[region]:[accountID]:appmonitor/[app monitor
 name]"
        }
    ]
}
```

- 애플리케이션에 다음을 추가하여 공급자의 자격 증명을 CloudWatch RUM으로 전달하도록 합니다. 사용자가 애플리케이션에 로그인하고 AWS에 액세스하는 데 사용할 자격 증명을 받은 후 실행되도록 라인을 삽입합니다.

```
cwr('setAwsCredentials', {/* Credentials or CredentialProvider */});
```

AWS JavaScript SDK의 자격 증명 공급자에 대한 자세한 내용은 JavaScript SDK용 v3 개발자 가이드에서 웹 브라우저에서 자격 증명 설정, JavaScript SDK용 v2 개발자 가이드에서 웹 브라우저에서 자격 증명 설정 및 @aws-sdk/credential-providers를 참조하세요.

CloudWatch RUM 웹 클라이언트용 SDK를 사용하여 웹 클라이언트 인증 방법을 구성할 수도 있습니다. 웹 클라이언트 SDK에 대한 자세한 내용은 CloudWatch RUM 웹 클라이언트 SDK를 참조하세요.

2단계: 앱 모니터 생성

애플리케이션에서 CloudWatch RUM을 사용하려면 앱 모니터를 생성합니다. 앱 모니터가 생성되면 RUM은 사용자가 애플리케이션에 붙여넣을 수 있도록 JavaScript 코드 조각을 생성합니다. 코드 조각

은 RUM 웹 클라이언트 코드를 가져옵니다. RUM 웹 클라이언트는 애플리케이션 사용자 세션 비율에서 데이터를 캡처하여 RUM으로 보냅니다.

앱 모니터 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, RUM을 선택합니다.
3. 앱 모니터 추가(Add app monitor)를 선택합니다.
4. 애플리케이션에 대한 정보 및 설정을 입력합니다.

   - 앱 모니터 이름(App monitor name)에서 CloudWatch RUM 콘솔 내에서 이 앱 모니터를 식별하는 데 사용할 이름을 입력합니다.
   - 애플리케이션 도메인(Application domain)에서 애플리케이션에 관리 권한이 있는 최상위 도메인 이름을 입력합니다. URL 도메인 형식이어야 합니다.

     하위 도메인 포함(Include sub domains)을 선택하여 앱 모니터가 최상위 도메인 아래의 모든 하위 도메인에서 데이터를 수집하도록 합니다.
5. RUM 데이터 수집 구성(Configure RUM data collection)에서 앱 모니터에서 다음을 각각 수집할지 여부를 지정합니다.

   - 성능 원격 측정(Performance telemetry): 페이지 로드 및 리소스 로드 시간에 대한 정보를 수집합니다.
   - JavaScript 오류: 애플리케이션에서 발생하는 처리되지 않은 JavaScript 오류에 대한 정보를 수집합니다.
   - HTTP 오류: 애플리케이션에서 발생하는 HTTP 오류에 대한 정보를 수집합니다.

   이러한 옵션을 선택하면 애플리케이션에 대한 자세한 정보가 제공되지만, 더 많은 CloudWatch RUM 이벤트가 생성되므로 더 많은 요금이 발생합니다.

   이 중 하나를 선택하지 않아도 앱 모니터가 세션 시작 이벤트 및 페이지 ID를 수집하여 운영 체제 유형 및 버전, 브라우저 유형 및 버전, 디바이스 유형 및 위치별 분류를 포함하여 애플리케이션을 사용하는 사용자 수를 확인할 수 있습니다.
6. 샘플링된 사용자 세션에서 사용자 ID 및 세션 ID를 수집할 수 있도록 하려면 CloudWatch RUM 웹 클라이언트가 쿠키를 설정할 수 있도록 이 옵션 선택(Check this option to allow the CloudWatch RUM Web Client to set cookies)을 선택합니다. 사용자 ID는 RUM에 의해 임의로 생성됩니다. 자세한 정보는 CloudWatch RUM 웹 클라이언트 쿠키(또는 유사한 기술)을 참조하세요.

7. 세션 샘플(Session samples)에서 RUM 데이터를 수집하는 데 사용할 사용자 세션 비율을 입력합니다. 기본값은 100%입니다. 이 수를 줄이면 데이터가 줄어들지만, 요금도 줄어듭니다. RUM 요금에 대한 자세한 내용은 RUM 요금을 참조하세요.
8. CloudWatch RUM에 대해 수집하는 최종 사용자 데이터는 30일 동안 보존된 다음 자동으로 삭제됩니다. CloudWatch Logs에 RUM 이벤트 복사본을 보관하고 이러한 복사본을 보존하는 기간을 구성하려면 데이터 스토리지(Data storage)에서 애플리케이션 원격 측정 데이터를 CloudWatch Logs 계정에 저장하려면 이 옵션을 선택(Check this option to store your application telemetry data in your CloudWatch Logs account)을 선택합니다. 기본적으로 CloudWatch Logs 로그 그룹은 데이터를 30일 동안 보존합니다. CloudWatch Logs 콘솔에서 보존 기간을 조정할 수 있습니다.
9. 권한 부여(Authorization)에서 새로운 또는 기존 Amazon Cognito 자격 증명 풀을 사용할지 아니면 다른 자격 증명 공급자를 사용할지를 지정합니다. 새 자격 증명 풀을 만드는 것은 다른 설정 단계가 필요하지 않은 가장 간단한 옵션입니다. 자세한 내용은 1단계: 애플리케이션에서 데이터를 AWS로 전송하도록 권한 부여 단원을 참조하십시오.

   새 Amazon Cognito 자격 증명 풀을 생성하려면 관리 권한이 필요합니다. 자세한 내용은 CloudWatch RUM을 사용하는 IAM 정책 섹션을 참조하세요.
10. (선택 사항) 기본적으로 애플리케이션에 RUM 코드 조각을 추가하면 웹 클라이언트는 애플리케이션의 모든 페이지에 있는 HTML 코드에 JavaScript 태그를 삽입하여 사용량을 모니터링합니다. 이를 변경하려면 페이지 구성(Configure pages)을 선택한 다음 이 페이지만 포함(Include only these pages) 또는 이 페이지 제외(Exclude these pages) 중 하나를 선택합니다. 그런 다음, 포함하거나 제외할 페이지를 지정합니다. 포함하거나 제외할 페이지를 지정하려면 전체 URL을 입력합니다. 추가 페이지를 지정하려면 URL 추가(Add URL)를 선택합니다.
11. 앱 모니터에서 샘플링한 사용자 세션의 AWS X-Ray 추적을 사용하려면 Active tracing(활성 추적)을 선택하고 Trace my service with AWS X-Ray(으로 내 서비스 추적)을 선택합니다.

    이 옵션을 선택하면 `XMLHttpRequest` 및 `fetch` 요청이 앱 모니터에서 샘플링하는 사용자 세션 중에 수행됩니다. 그런 다음, RUM 대시보드, X-Ray 트레이스 맵 및 트레이스 세부 정보 페이지에서 이러한 사용자 세션의 트레이스와 세그먼트를 볼 수 있습니다. 이러한 사용자 세션은 애플리케이션에 대해 활성화된 후 Application Signals의 클라이언트 페이지로도 표시됩니다.

    CloudWatch RUM 웹 클라이언트에 대한 추가 구성을 변경함으로써 HTTP 요청에 X-Ray 추적 헤더를 추가하여 AWS 관리형 서비스 다운스트림으로 사용자 세션의 엔드 투 엔드 추적을 활성화할 수 있습니다. 자세한 내용은 X-Ray 종단 간 추적 활성화 단원을 참조하십시오.
12. (선택 사항) 앱 모니터에 태그를 추가하려면 태그(Tags)에서 새 태그 추가(Add new tag)를 선택합니다.

그런 다음, 키(Key)에서 태그 이름을 입력합니다. 값(Value)에 태그의 선택적 값을 추가할 수 있습니다.

다른 태그를 추가하려면 새 태그 추가(Add new tag)를 다시 선택합니다.

자세한 내용은 AWS 리소스에 태깅을 참조하세요.

13. 앱 모니터 추가(Add app monitor)를 선택합니다.
14. 샘플 코드(Sample code) 섹션에서 애플리케이션에 추가하는 데 사용할 코드 조각을 복사할 수 있습니다. JavaScript 또는 TypeScript를 선택하고 NPM을 사용하여 CloudWatch RUM 웹 클라이언트를 JavaScript 모듈로 설치하는 것이 좋습니다.

    또는 콘텐츠 전송 네트워크(CDN)를 사용하는 HTML을 선택하여 CloudWatch RUM 웹 클라이언트를 설치합니다. CDN 사용 시의 단점은 웹 클라이언트가 광고 차단기에 의해 차단되는 경우가 많다는 것입니다.
15. 복사(Copy) 또는 다운로드(Download)를 선택한 다음 완료(Done)를 선택합니다.

(선택 사항) 3단계: 코드 조각을 수동으로 수정하여 CloudWatch RUM 웹 클라이언트를 구성합니다.

애플리케이션에 삽입하기 전에 코드 조각을 수정하여 여러 옵션을 활성화하거나 비활성화할 수 있습니다. 자세한 내용은 CloudWatch RUM 웹 클라이언트 설명서를 참조하세요.

이 섹션에서 설명한 것처럼 세 가지 구성 옵션을 반드시 알고 있어야 합니다.

개인 정보를 포함할 수 있는 리소스 URL 수집 방지

기본적으로 CloudWatch RUM 웹 클라이언트는 애플리케이션에서 다운로드한 리소스 URL을 기록하도록 구성됩니다. 이러한 리소스에는 HTML 파일, 이미지, CSS 파일, JavaScript 파일 등이 포함됩니다. 일부 애플리케이션의 경우 URL에 개인 식별 정보(PII)가 포함될 수 있습니다.

애플리케이션이 여기에 해당한다면 애플리케이션에 삽입하기 전에 코드 조각 구성에서 `recordResourceUrl: false` 설정을 통해 리소스 URL 수집을 비활성화할 것을 강력히 권장합니다.

페이지 보기 수동 기록

기본적으로 웹 클라이언트는 페이지가 처음 로드할 때와 브라우저 기록 API가 호출될 때 페이지 보기를 기록합니다. 기본 페이지 ID는 `window.location.pathname`입니다. 그러나 경우에 따라 이 동작

을 재정의하고 애플리케이션을 계측하여 페이지 보기를 프로그래밍 방식으로 기록할 수도 있습니다. 이렇게 하면 페이지 ID와 기록 시기를 제어할 수 있습니다. 변수 식별자가 있는 URI(예: `/entity/123` 또는 `/entity/456`)가 있는 웹 애플리케이션을 예로 들어 보겠습니다. 기본적으로 CloudWatch RUM은 경로 이름과 일치하는 고유한 페이지 ID를 가진 각 URI에 대해 페이지 보기 이벤트를 생성하지만, 대신 동일한 페이지 ID로 그룹화할 수 있습니다. 이렇게 하려면 `disableAutoPageView` 구성을 사용하여 웹 클라이언트의 페이지 보기 자동화를 사용하지 않도록 설정하고 `recordPageView` 명령을 사용하여 원하는 페이지 ID를 설정합니다. 자세한 내용은 GitHub의 Application-specific Configurations를 참조하세요.

임베디드 스크립트 예:

```
cwr('recordPageView', { pageId: 'entityPageId' });
```

JavaScript 모듈 예:

```
awsRum.recordPageView({ pageId: 'entityPageId' });
```

X-Ray 종단 간 추적 활성화

앱 모니터를 생성할 때 Trace my service with AWS X-Ray(으로 내 서비스 추적)을 선택하면 앱 모니터에서 샘플링한 사용자 세션 도중에 만들어진 `XMLHttpRequest` 및 `fetch` 요청 추적을 사용할 수 있습니다. 그런 다음, CloudWatch RUM 대시보드, X-Ray 트레이스 맵 및 트레이스 세부 정보 페이지에서 이러한 HTTP 요청의 트레이스를 볼 수 있습니다.

기본적으로 이러한 클라이언트 측 추적은 다운스트림 서버 측 추적에 연결되지 않습니다. 클라이언트 측 추적을 서버 측 추적에 연결하고 종단 간 추적을 활성화하려면 웹 클라이언트에서 `addXRayTraceIdHeader` 옵션을 `true`로 설정합니다. 이로 인해 CloudWatch RUM 웹 클라이언트가 HTTP 요청에 X-Ray 추적 헤더를 추가합니다.

다음 코드 블록은 클라이언트 측 추적을 추가하는 예를 보여줍니다. 가독성을 위해 이 샘플에서 일부 구성 옵션이 생략됩니다.

```
<script>
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
        '1.0.0',
        'us-west-2',
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
```

```
        {
            enableXRay: true,
            telemetries: [
                'errors',
                'performance',
                [ 'http', { addXRayTraceIdHeader: true } ]
            ]
        }
    );
</script>
```

> **Warning**
>
> HTTP 요청에 X-Ray 추적 헤더를 추가하도록 CloudWatch RUM 웹 클라이언트를 구성하면 요청이 SigV4로 서명된 경우 cross-origin 리소스 공유(CORS)가 실패하거나 요청의 서명이 무효화될 수 있습니다. 자세한 내용은 CloudWatch RUM 웹 클라이언트 설명서를 참조하세요. 프로덕션 환경에서 클라이언트 측 X-Ray 추적 헤더를 추가하기 전에 애플리케이션을 테스트하는 것을 강력하게 권장합니다.

자세한 내용은 CloudWatch RUM 웹 클라이언트 설명서를 참조하세요.

4단계: 애플리케이션에 코드 조각 삽입

다음으로 이전 섹션에서 생성한 코드 조각을 애플리케이션에 삽입합니다.

> **Warning**
>
> 코드 조각으로 다운로드 및 구성된 웹 클라이언트는 최종 사용자 데이터를 수집하기 위해 쿠키(또는 유사한 기술)를 사용합니다. 코드 조각을 삽입하기 전에 콘솔에서 메타데이터 속성별로 필터링 섹션을 참고하세요.

이전에 생성된 코드 조각이 없는 경우 이미 생성한 코드 조각을 찾으려면 어떻게 해야 한가요? 섹션의 지침에 따라 찾을 수 있습니다.

애플리케이션에 CloudWatch RUM 코드 조각 삽입

1. 복사하거나 다운로드한 코드 조각을 애플리케이션의 <head> 요소의 이전 섹션에 삽입합니다. <body> 요소 또는 기타 <script> 태그 앞에 삽입합니다.

다음은 생성된 코드 조각의 예입니다.

```
<script>
(function (n, i, v, r, s, c, x, z) {
    x = window.AwsRumClient = {q: [], n: n, i: i, v: v, r: r, c: c};
    window[n] = function (c, p) {
        x.q.push({c: c, p: p});
    };
    z = document.createElement('script');
    z.async = true;
    z.src = s;
    document.head.insertBefore(z, document.getElementsByTagName('script')[0]);
})('cwr',
    '194a1c89-87d8-41a3-9d1b-5c5cd3dafbd0',
    '1.0.0',
    'us-east-2',
    'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
    {
        sessionSampleRate: 1,
        identityPoolId: "us-east-2:c90ef0ac-e3b8-4d1a-b313-7e73cfd21443",
        endpoint: "https://dataplane.rum.us-east-2.amazonaws.com",
        telemetries: ["performance", "errors", "http"],
        allowCookies: true,
        enableXRay: false
    });
</script>
```

2. 애플리케이션이 여러 페이지의 웹 애플리케이션인 경우 데이터 수집에 포함할 각 HTML 페이지에 대해 1단계를 반복해야 합니다.

5단계: 사용자 이벤트를 생성한 앱 모니터 설정 테스트

코드 조각을 삽입하고 업데이트된 애플리케이션이 실행 중이면 수동으로 사용자 이벤트를 생성하여 테스트할 수 있습니다. 이 테스트를 하려면 다음을 수행하는 것이 좋습니다. 이 테스트에서는 표준 CloudWatch RUM 요금이 발생합니다.

- 웹 애플리케이션의 페이지 사이를 탐색합니다.
- 서로 다른 브라우저와 디바이스를 사용하여 여러 사용자 세션을 생성합니다.
- 요청을 생성합니다.

- JavaScript 오류를 유발합니다.

일부 이벤트를 생성한 후에는 CloudWatch RUM 대시보드에서 해당 이벤트를 확인합니다. 자세한 내용은 CloudWatch RUM 대시보드 보기 단원을 참조하십시오.

사용자 세션의 데이터가 대시보드에 표시되는 데 최대 15분이 걸릴 수 있습니다.

애플리케이션에서 이벤트를 생성한 후 15분 후에 데이터가 표시되지 않으면 CloudWatch RUM 문제 해결 섹션을 참조하세요.

CloudWatch RUM 웹 클라이언트 구성

애플리케이션은 CloudWatch RUM에서 생성한 코드 조각 중 하나를 사용하여 CloudWatch RUM 웹 클라이언트를 설치할 수 있습니다. 생성된 조각은 NPM을 통한 JavaScript 모듈이나 콘텐츠 전송 네트워크(CDN)의 두 가지 설치 방법을 지원합니다. 최상의 성능을 내려면 NPM 설치 방법을 사용하는 것이 좋습니다. 이 방법의 사용에 대한 자세한 내용은 JavaScript 모듈로 설치를 참조하세요.

CDN 설치 옵션을 사용하는 경우 광고 차단기가 CloudWatch RUM에서 제공하는 기본 CDN을 차단할 수 있습니다. 이렇게 하면 광고 차단기를 설치한 사용자에 대해 애플리케이션 모니터링이 사용 중지됩니다. 따라서 CloudWatch RUM이 포함된 초기 온보딩에만 기본 CDN을 사용하는 것이 좋습니다. 이 문제를 완화하는 방법에 대한 자세한 내용은 애플리케이션 계측을 참조하세요.

코드 조각은 HTML 파일의 <head> 태그에 위치하고, 웹 클라이언트를 다운로드한 다음 모니터링 중인 애플리케이션에 대한 웹 클라이언트를 구성하여 웹 클라이언트를 설치합니다. 코드 조각은 다음과 비슷하게 보이는 자체 실행 함수입니다. 이 예제에서는 가독성을 위해 코드 조각 함수의 본문이 생략되었습니다.

```
<script>
(function(n,i,v,r,s,c,u,x,z){...})(
'cwr',
'00000000-0000-0000-0000-000000000000',
'1.0.0',
'us-west-2',
'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
{ /* Configuration Options Here */ }
);
<script>
```

인수

코드 조각은 6개의 인수를 허용합니다.

- 웹 클라이언트에서 명령을 실행하기 위한 네임스페이스(예: `'cwr'`)
- 앱 모니터의 ID(예: `'00000000-0000-0000-0000-000000000000'`)
- 애플리케이션 버전(예: `'1.0.0'`)
- 앱 모니터의 AWS 리전(예: `'us-west-2'`)
- 웹 클라이언트 URL(예: `'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js'`)
- 애플리케이션별 구성 옵션 자세한 내용은 다음 섹션을 참조하세요.

오류 무시

CloudWatch RUM 웹 클라이언트는 애플리케이션에서 발생하는 모든 유형의 오류를 수신합니다. CloudWatch RUM 대시보드에 표시하지 않으려는 JavaScript 오류가 애플리케이션에서 전달되는 경우, 이러한 오류를 필터링하고 CloudWatch RUM 대시보드에 관련 오류 이벤트만 표시하도록 CloudWatch RUM 웹 클라이언트를 구성할 수 있습니다. 예를 들어 일부 JavaScript 오류에 대한 수정 사항을 이미 식별했고 방대한 오류의 양으로 인해 다른 오류가 가려지는 경우 대시보드에 일부 JavaScript 오류를 표시하지 않도록 선택할 수 있습니다. 또한 서드 파티가 소유한 라이브러리의 소유이기 때문에 수정할 수 없는 오류를 무시할 수도 있습니다.

특정 JavaScript 오류를 필터링하도록 웹 클라이언트를 구성하는 방법에 대한 자세한 내용은 웹 클라이언트 Github 설명서에서 오류의 예를 참조하세요.

구성 옵션

CloudWatch RUM 웹 클라이언트에서 사용할 수 있는 구성 옵션에 대한 자세한 내용은 CloudWatch RUM 웹 클라이언트 설명서를 참조하세요.

지역화

이 섹션에서는 다양한 리전의 애플리케이션과 함께 CloudWatch RUM을 사용하는 전략을 설명합니다.

웹 애플리케이션이 여러 AWS 리전에 배포되어 있습니다.

웹 애플리케이션이 여러 AWS 리전에 배포되는 경우 다음과 같은 세 가지 옵션이 있습니다.

- 하나의 계정으로 하나의 리전에 하나의 앱 모니터를 배포하여 모든 리전에 서비스를 제공합니다.
- 각 리전에 대해 고유한 계정으로 별도의 앱 모니터를 배포합니다.
- 하나의 계정으로 각 리전에 대해 별도의 앱 모니터를 배포할 수 있습니다.

하나의 앱 모니터를 사용하면 모든 데이터가 하나의 시각화로 중앙 집중화되고 모든 로그가 CloudWatch Logs의 동일한 로그 그룹에 기록된다는 이점이 있습니다. 단일 앱 모니터를 사용하면 요청에 대한 추가 지연 시간이 적고 단일 장애 지점이 있습니다.

여러 개의 앱 모니터를 사용하면 단일 장애 지점은 제거되지만 모든 데이터가 하나의 시각화로 결합되지 않습니다.

내 애플리케이션이 배포된 일부 지역에서 CloudWatch RUM이 시작되지 않았습니다.

CloudWatch RUM은 여러 지역에 출시되어 광범위한 지역 서비스를 제공합니다. 사용 가능한 지역에서 CloudWatch RUM을 설정하면 혜택을 누릴 수 있습니다. 최종 사용자는 연결 중인 리전에서 앱 모니터를 설정한 경우 어디에 있더라도 세션에 포함될 수 있습니다.

그러나 CloudWatch RUM은 아직 AWS GovCloud(미국 동부), AWS GovCloud(미국 서부) 또는 중국의 어떤 리전에서도 출시되지 않았습니다. 이러한 지역에서는 CloudWatch RUM으로 데이터를 전송할 수 없습니다.

페이지 그룹 사용

페이지 그룹을 사용하여 애플리케이션의 여러 페이지를 서로 연결함으로써 페이지 그룹에 대한 집계된 분석을 확인할 수 있습니다. 예를 들어 모든 랜딩 페이지의 집계된 페이지 로드 시간을 확인할 수 있습니다.ag

CloudWatch RUM 웹 클라이언트의 페이지 보기 이벤트에 태그를 하나 이상 추가하여 페이지를 페이지 그룹에 넣을 수 있습니다. 다음 예에서는 `/home` 페이지를 `en`이라는 페이지 그룹과 `landing`이라는 페이지 그룹에 넣습니다.

임베디드 스크립트 예

```
cwr('recordPageView', { pageId: '/home', pageTags: ['en', 'landing']});
```

JavaScript 모듈 예

```
awsRum.recordPageView({ pageId: '/home', pageTags: ['en', 'landing']});
```

> **Note**
>
> 페이지 그룹은 여러 페이지에서 분석을 쉽게 집계하기 위한 것입니다. 애플리케이션에 대해 `pageIds`를 정의하고 조작하는 방법에 대한 자세한 내용은 (선택 사항) 3단계: 코드 조각을 수

동으로 수정하여 CloudWatch RUM 웹 클라이언트를 구성합니다.의 페이지 조회 수 수동으로 기록하기 섹션을 참조하세요.

사용자 지정 메타데이터 지정

CloudWatch RUM은 각 이벤트에 추가 데이터를 메타데이터로 첨부합니다. 이벤트 메타데이터는 키-값 페어 형식의 속성으로 구성됩니다. 이러한 속성을 사용하여 CloudWatch RUM 콘솔에서 이벤트를 검색하거나 필터링할 수 있습니다. 기본적으로 CloudWatch RUM은 사용자를 위해 몇 가지 메타데이터를 생성합니다. 기본 메타데이터에 대한 자세한 내용은 RUM 이벤트 메타데이터 섹션을 참조하세요.

CloudWatch RUM 웹 클라이언트를 사용하여 CloudWatch RUM 이벤트에 사용자 지정 메타데이터를 추가할 수도 있습니다. 사용자 지정 메타데이터는 세션 속성과 페이지 속성을 포함할 수 있습니다.

사용자 지정 메타데이터를 추가하려면 버전 1.10.0 이상의 CloudWatch RUM 웹 클라이언트를 사용해야 합니다.

요구 사항 및 구문

각 이벤트는 메타데이터에 최대 10개의 사용자 지정 속성을 포함할 수 있습니다. 사용자 지정 속성의 구문 요구 사항은 다음과 같습니다.

- 키
  - 최대 128자입니다.
  - 영숫자 문자, 콜론(:) 및 밑줄(_)을 포함할 수 있습니다.
  - `aws:`으로 시작할 수 없습니다.
  - 다음 섹션에 나열된 예약 키워드로만 구성될 수 없습니다. 해당 키워드를 더 긴 키 이름의 일부로 사용할 수 있습니다.
- 값
  - 최대 256자입니다.
  - 문자열, 숫자 또는 부울 값이어야 합니다.

예약어

다음 예약 키워드는 전체 키 이름으로 사용할 수 없습니다. `applicationVersion`과 같은 더 긴 키 이름의 일부로 다음 키워드를 사용할 수 있습니다.

- `browserLanguage`

- browserName
- browserVersion
- countryCode
- deviceType
- domain
- interaction
- osName
- osVersion
- pageId
- pageTags
- pageTitle
- pageUrl
- parentPageId
- platformType
- referrerUrl
- subdivisionCode
- title
- url
- version

> **Note**
>
> CloudWatch RUM은 속성에 유효하지 않은 키 또는 값이 포함되어 있거나 이벤트당 사용자 지정 속성 제한 10개에 이미 도달한 경우 RUM 이벤트에서 사용자 지정 속성을 제거합니다.

세션 속성 추가

사용자 지정 세션 속성을 구성하면 해당 속성이 세션의 모든 이벤트에 추가됩니다. CloudWatch RUM 웹 클라이언트 초기화 중 또는 런타임에 addSessionAttributes 명령을 사용하여 세션 속성을 구성합니다.

예를 들어, 애플리케이션의 버전을 세션 속성으로 추가할 수 있습니다. 그런 다음 CloudWatch RUM 콘솔에서 버전별로 오류를 필터링하여 오류율 증가가 애플리케이션의 특정 버전과 관련이 있는지 확인할 수 있습니다.

초기화 시 세션 속성 추가, NPM 예제

굵게 표시된 코드 섹션은 세션 속성을 추가합니다.

```
import { AwsRum, AwsRumConfig } from 'aws-rum-web';

try {
  const config: AwsRumConfig = {
    allowCookies: true,
    endpoint: "https://dataplane.rum.us-west-2.amazonaws.com",
    guestRoleArn: "arn:aws:iam::000000000000:role/RUM-Monitor-us-
west-2-000000000000-00xx-Unauth",
    identityPoolId: "us-west-2:00000000-0000-0000-0000-000000000000",
    sessionSampleRate: 1,
    telemetries: ['errors', 'performance'],
    sessionAttributes: {
        applicationVersion: "1.3.8"
    }
  };

  const APPLICATION_ID: string = '00000000-0000-0000-0000-000000000000';
  const APPLICATION_VERSION: string = '1.0.0';
  const APPLICATION_REGION: string = 'us-west-2';

  const awsRum: AwsRum = new AwsRum(
    APPLICATION_ID,
    APPLICATION_VERSION,
    APPLICATION_REGION,
    config
  );
} catch (error) {
  // Ignore errors thrown during CloudWatch RUM web client initialization
}
```

런타임 시 세션 속성 추가, NPM 예제

```
awsRum.addSessionAttributes({
    applicationVersion: "1.3.8"
```

```
})
```

초기화 시 세션 속성 추가, 임베디드 스크립트 예제

굵게 표시된 코드 섹션은 세션 속성을 추가합니다.

```
<script>
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
        '1.0.0',
        'us-west-2',
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
        {
            sessionSampleRate:1,
            guestRoleArn:'arn:aws:iam::000000000000:role/RUM-Monitor-us-
west-2-000000000000-00xx-Unauth',
            identityPoolId:'us-west-2:00000000-0000-0000-0000-000000000000',
            endpoint:'https://dataplane.rum.us-west-2.amazonaws.com',
            telemetries:['errors','http','performance'],
            allowCookies:true,
            sessionAttributes: {
                applicationVersion: "1.3.8"
            }
        }
    );
</script>
```

런타임 시 세션 속성 추가, 임베디드 스크립트 예제

```
<script>
    function addSessionAttribute() {
        cwr('addSessionAttributes', {
            applicationVersion: "1.3.8"
        })
    }

</script>
```

페이지 속성 추가

사용자 지정 페이지 속성을 구성하면 해당 속성이 현재 페이지의 모든 이벤트에 추가됩니다. CloudWatch RUM 웹 클라이언트 초기화 중 또는 런타임에 `recordPageView` 명령을 사용하여 페이지 속성을 구성합니다.

예를 들어, 페이지 템플릿을 페이지 속성으로 추가할 수 있습니다. 그런 다음 CloudWatch RUM 콘솔에서 페이지 템플릿별로 오류를 필터링하여 오류율 증가가 애플리케이션의 특정 페이지 템플릿과 관련이 있는지 확인할 수 있습니다.

초기화 시 페이지 속성 추가, NPM 예제

굵게 표시된 코드 섹션은 페이지 속성을 추가합니다.

```
const awsRum: AwsRum = new AwsRum(
    APPLICATION_ID,
    APPLICATION_VERSION,
    APPLICATION_REGION,
    { disableAutoPageView:  true // optional }
);
awsRum.recordPageView({
    pageId:'/home',
    pageAttributes: {
      template: 'artStudio'
    }
});
const credentialProvider = new CustomCredentialProvider();
if(awsCreds) awsRum.setAwsCredentials(credentialProvider);
```

런타임 시 페이지 속성 추가, NPM 예제

```
awsRum.recordPageView({
    pageId: '/home',
    pageAttributes: {
        template: 'artStudio'
    }
});
```

초기화 시 페이지 속성 추가, 임베디드 스크립트 예제

굵게 표시된 코드 섹션은 페이지 속성을 추가합니다.

```
<script>
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
        '1.0.0',
        'us-west-2',
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
        {
            disableAutoPageView: true //optional
        }
    );
    cwr('recordPageView', {
       pageId: '/home',
       pageAttributes: {
           template: 'artStudio'
       }
    });
    const awsCreds = localStorage.getItem('customAwsCreds');
    if(awsCreds) cwr('setAwsCredentials', awsCreds)
</script>
```

런타임 시 페이지 속성 추가, 임베디드 스크립트 예제

```
<script>
    function recordPageView() {
        cwr('recordPageView', {
            pageId: '/home',
            pageAttributes: {
                template: 'artStudio'
            }
        });
    }
</script>
```

콘솔에서 메타데이터 속성별로 필터링

CloudWatch RUM 콘솔에서 기본 제공 또는 사용자 지정 메타데이터 속성으로 시각화를 필터링하려면 검색 창을 사용하세요. 검색 창에서 시각화에 적용할 필터 용어를 key=value 형식으로 20개까지 지정할 수 있습니다. 예를 들어 Chrome 브라우저에 대한 데이터만 필터링하려면 browserName=Chrome이라는 필터 용어를 추가할 수 있습니다.

기본적으로 CloudWatch RUM 콘솔은 검색 창의 드롭다운에 표시할 가장 일반적인 속성 키 및 값 100개를 검색합니다. 더 많은 메타데이터 속성을 필터 용어로 추가하려면 검색 창에 전체 속성 키와 값을 입력합니다.

필터는 최대 20개의 필터 용어를 포함할 수 있으며 앱 모니터당 최대 20개의 필터를 저장할 수 있습니다. 필터를 저장하면 Saved filters(저장된 필터) 드롭다운에 저장됩니다. 저장된 필터를 삭제할 수도 있습니다.

사용자 지정 이벤트 전송

CloudWatch RUM은 CloudWatch RUM 웹 클라이언트에서 수집한 정보에 나열된 이벤트를 기록하고 수집합니다. CloudWatch RUM 웹 클라이언트 버전 1.12.0 이상을 사용하는 경우 추가 사용자 지정 이벤트를 정의, 기록 및 전송할 수 있습니다. 정의하는 각 이벤트 유형에 대해 전송할 이벤트 유형 이름과 데이터를 정의합니다. 각 사용자 지정 이벤트 페이로드는 최대 6KB일 수 있습니다.

사용자 지정 이벤트는 앱 모니터에 사용자 지정 이벤트가 활성화된 경우에만 수집됩니다. 앱 모니터의 구성 설정을 업데이트하려면 CloudWatch RUM 콘솔 또는 UpdateAppMonitor API를 사용하세요.

사용자 지정 이벤트를 활성화한 다음 정의하고 전송한 후 검색할 수 있습니다. 사용자 지정 이벤트를 검색하려면 CloudWatch RUM 콘솔의 Events(이벤트) 탭을 사용합니다. 이벤트 유형을 사용하여 검색합니다.

요구 사항 및 구문

사용자 지정 이벤트는 이벤트 유형과 이벤트 세부 정보로 구성됩니다. 다음은 이들에 대한 요구 사항입니다.

- 이벤트 유형
  - 이벤트의 type(유형) 또는 name(이름)일 수 있습니다. 예를 들어 JsError라는 CloudWatch RUM 기본 제공 이벤트 유형의 이벤트 유형은 `com.amazon.rum.js_error_event`입니다.
  - 1~256자여야 합니다.
  - 영숫자, 밑줄, 하이픈 및 마침표의 조합일 수 있습니다.
- 이벤트 세부 정보
  - CloudWatch RUM에 기록하려는 실제 데이터를 포함합니다.
  - 필드와 값으로 구성된 객체여야 합니다.

사용자 지정 이벤트 기록의 예

CloudWatch RUM 웹 클라이언트에서 사용자 지정 이벤트를 기록하는 방법에는 두 가지가 있습니다.

- CloudWatch RUM 웹 클라이언트의 `recordEvent` API를 사용합니다.
- 사용자 지정 플러그인을 사용합니다.

`recordEvent` API, NPM 예제를 사용하여 사용자 지정 이벤트 전송

```
awsRum.recordEvent('my_custom_event', {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
)
```

`recordEvent` API, 임베디드 스크립트 예제를 사용하여 사용자 지정 이벤트 전송

```
cwr('recordEvent', {
    type: 'my_custom_event',
    data: {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
})
```

사용자 지정 플러그인을 사용하여 사용자 지정 이벤트를 전송하는 예

```
// Example of a plugin that listens to a scroll event, and
// records a 'custom_scroll_event' that contains the timestamp of the event.
class MyCustomPlugin implements Plugin {
    // Initialize MyCustomPlugin.
```

```
    constructor() {
        this.enabled;
        this.context;
        this.id = 'custom_event_plugin';
    }
    // Load MyCustomPlugin.
    load(context) {
        this.context = context;
        this.enable();
    }
    // Turn on MyCustomPlugin.
    enable() {
        this.enabled = true;
        this.addEventHandler();
    }
    // Turn off MyCustomPlugin.
    disable() {
        this.enabled = false;
        this.removeEventHandler();
    }
    // Return MyCustomPlugin Id.
    getPluginId() {
        return this.id;
    }
    // Record custom event.
    record(data) {
        this.context.record('custom_scroll_event', data);
    }
    // EventHandler.
    private eventHandler = (scrollEvent: Event) => {
        this.record({timestamp: Date.now()})
    }
    // Attach an eventHandler to scroll event.
    private addEventHandler(): void {
        window.addEventListener('scroll', this.eventHandler);
    }
    // Detach eventHandler from scroll event.
    private removeEventHandler(): void {
        window.removeEventListender('scroll', this.eventHandler);
    }
}
```

CloudWatch RUM 대시보드 보기

CloudWatch RUM을 사용하면 페이지 로드 시간, Apdex 점수, 사용된 브라우저 및 디바이스, 사용자 세션의 지리적 위치, 오류가 있는 세션 등 애플리케이션 성능에 대한 사용자 세션에서 데이터를 수집할 수 있습니다. 이 모든 정보는 대시보드에 표시됩니다.

RUM 대시보드 보기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, RUM을 선택합니다.

   개요(Overview) 탭에는 사용자가 만든 앱 모니터 중 하나에서 수집한 정보가 표시됩니다.

   창의 맨 위 행에는 이 앱 모니터에 대한 다음 정보가 표시됩니다.

   - 페이지 로드 수
   - 평균 페이지 로드 속도
   - Apdex 점수
   - 앱 모니터와 연결된 모든 알람 상태

   애플리케이션 성능 인덱스(Apdex) 점수는 최종 사용자의 만족도를 나타냅니다. 점수는 0(최소 만족)부터 1(가장 만족)까지입니다. 점수는 애플리케이션 성능만을 기준으로 합니다. 사용자에게 애플리케이션을 평가하라는 메시지가 표시되지 않습니다. Apdex 점수에 대한 자세한 내용은 CloudWatch RUM에서 Apdex 점수를 설정하는 방법 섹션을 참조하세요.

   이러한 창 일부에는 데이터를 추가로 검사할 때 사용할 수 있는 링크가 포함되어 있습니다. 이러한 링크 중 하나를 선택하면 디스플레이 상단에 성능, 오류, HTTP 요청, 세션, 이벤트 브라우저 및 디바이스, 사용자 여정 탭이 있는 상세 보기가 표시됩니다.
3. 더 세분화하려면 List view(목록 보기) 탭을 선택한 다음 살펴볼 앱 모니터의 이름을 선택합니다. 그러면 선택한 앱 모니터에 대해 다음 탭이 표시됩니다.

   - Performance(성능) 탭에는 로드 시간, 세션 정보, 요청 정보, 웹 바이탈 및 시간 경과에 따른 페이지 로드를 포함한 페이지 성능 정보가 표시됩니다. 이 보기에는 페이지 로드(Page loads), 요청(Requests) 및위치(Location) 초점 맞추기 간에 보기를 전환하는 제어가 포함되어 있습니다.
   - 오류 및 세션 탭에는 사용자에게 가장 자주 표시되는 오류 메시지와 오류가 가장 많이 발생한 디바이스 및 브라우저를 포함한 Javascript 오류 정보가 표시됩니다. 이 보기에는 오류 히스토그램

과 오류 목록 보기가 포함되어 있습니다. 사용자 및 이벤트 세부 정보를 기준으로 오류 목록을 필터링할 수 있습니다. 오류 메시지를 선택하면 자세한 내용을 볼 수 있습니다.

- HTTP 요청 탭에는 오류가 가장 많이 발생한 요청 URL, 오류가 가장 많이 발생한 장치 및 브라우저를 비롯한 HTTP 요청 정보가 표시됩니다. 이 탭에는 요청 히스토그램, 요청 목록 보기, 네트워크 오류 목록 보기가 포함됩니다. 사용자 및 이벤트 세부 정보별로 목록을 필터링할 수 있습니다. 응답 코드 또는 오류 메시지를 선택하면 각각 요청 또는 네트워크 오류에 대한 자세한 내용을 확인할 수 있습니다.
- 세션 탭에는 세션 지표가 표시됩니다. 이 탭에는 세션 시작 이벤트의 히스토그램과 세션의 목록 보기가 포함되어 있습니다. 이벤트 유형, 사용자 세부 정보, 이벤트 세부 정보별로 세션 목록을 필터링할 수 있습니다. 세션에 대한 자세한 내용을 보려면 sessionId를 선택하세요.
- 이벤트 탭에는 RUM 이벤트의 히스토그램과 이벤트 목록 보기가 표시됩니다. 이벤트 유형, 사용자 세부 정보, 이벤트 세부 정보별로 이벤트 목록을 필터링할 수 있습니다. RUM 이벤트를 선택하면 원시 이벤트를 볼 수 있습니다.
- 브라우저 및 디바이스(Browsers & Devices) 탭에는 애플리케이션에 액세스하기 위한 다양한 브라우저 및 디바이스의 성능 및 사용량 등의 정보가 표시됩니다. 이 보기에는 브라우저 및 디바이스에 초점을 맞추는 보기를 전환하는 제어가 포함되어 있습니다.

  범위를 단일 브라우저로 좁히면 브라우저 버전별로 분류된 데이터가 표시됩니다.
- 사용자 여정(User Journey) 탭에는 고객이 애플리케이션을 탐색하는 데 사용하는 경로가 표시됩니다. 고객이 애플리케이션에 들어가는 위치와 애플리케이션을 종료하는 페이지를 확인할 수 있습니다. 또한 고객이 사용하는 경로와 해당 경로를 따르는 고객의 비율도 확인할 수 있습니다. 노드에서 일시 중지하여 해당 페이지에 대한 세부 정보를 확인할 수 있습니다. 단일 경로를 선택하여 더 쉽게 볼 수 있도록 연결을 강조 표시할 수 있습니다.

4. (선택 사항) 처음 6개의 탭 중 하나에서 페이지 버튼을 누르고 목록에서 페이지 또는 페이지 그룹을 선택합니다. 이렇게 하면 표시된 데이터가 애플리케이션의 단일 페이지 또는 페이지 그룹으로 좁아집니다. 목록의 페이지 또는 페이지 그룹을 즐겨찾기로 표시할 수도 있습니다.

CloudWatch RUM에서 Apdex 점수를 설정하는 방법

Apdex(애플리케이션 성능 인덱스)는 애플리케이션 응답 시간을 보고, 벤치마크 및 평가하기 위한 방법을 정의하는 개방형 표준입니다. Apdex 점수는 시간이 지남에 따라 애플리케이션 성능에 미치는 영향을 이해하고 식별하는 데 도움이 됩니다.

Apdex 점수는 최종 사용자의 만족도 점수 범위는 0(최소 만족)에서 1(가장 만족)까지를 나타냅니다. 점수는 애플리케이션 성능만을 기준으로 합니다. 사용자에게 애플리케이션을 평가하라는 메시지가 표시되지 않습니다.

각 Apdex 점수는 세 가지 임계값 중 하나에 해당합니다. Apdex 임계값과 실제 애플리케이션 응답 시간에 따라 다음과 같은 세 가지 종류의 성능이 있습니다.

- 만족: 실제 애플리케이션 응답 시간이 Apdex 임계값 이하입니다. CloudWatch RUM의 경우 이 임계값은 2,000ms 이하입니다.
- 나쁘지 않음: 실제 애플리케이션 응답 시간이 Apdex 임계값보다 크지만 Apdex 임계값의 4배 이하입니다. CloudWatch RUM의 경우 이 범위는 2,000~8,000ms입니다.
- 실망: 실제 애플리케이션 응답 시간이 Apdex 임계값의 4배보다 큽니다. CloudWatch RUM의 경우 이 범위는 8,000ms 초과입니다.

총 0~1의 Apdex 점수는 다음 공식을 사용하여 계산됩니다.

```
(positive scores + tolerable scores/2)/total scores * 100
```

CloudWatch RUM을 사용하여 수집할 수 있는 CloudWatch 지표

이 섹션의 표에는 CloudWatch RUM을 사용하여 자동으로 수집되는 지표가 나열되어 있습니다. CloudWatch 콘솔에서 이러한 지표에 액세스할 수 있습니다. 자세한 내용은 사용 가능한 지표 보기 단원을 참조하십시오.

필요에 따라 확장 지표를 CloudWatch 또는 CloudWatch Evidently로 전송할 수도 있습니다. 자세한 내용은 확장 지표 단원을 참조하십시오.

이러한 지표는 `AWS/RUM`이라는 지표 네임스페이스에 게시됩니다. 다음 지표는 모두 `application_name` 차원으로 게시됩니다.. 이 차원값은 앱 모니터 이름입니다. 일부 지표는 표에 나열된 대로 추가 차원으로 게시됩니다.

지표	단위	설명
`HttpStatusCodeCount`	개수	응답 상태 코드에 의한 애플리케이션의 HTTP 응답 수입니다. 추가 차원: • `event_details.response.status` 는

지표	단위	설명
		200, 400, 404 등과 같은 응답 상태 코드입니다. • event_type 이벤트의 유형입니다. 현재 이 차원에 대해 사용할 수 있는 유일한 값은 http입니다.
Http4xxCount	개수	4xx 응답 상태 코드가 포함된 애플리케이션의 HTTP 응답 수입니다. 4xx 코드를 생성하는 http_event RUM 이벤트를 기반으로 계산됩니다.
Http5xxCount	개수	5xx 응답 상태 코드가 포함된 애플리케이션의 HTTP 응답 수입니다. 5xx 코드를 생성하는 http_event RUM 이벤트를 기반으로 계산됩니다.
JsErrorCount	개수	수집된 JavaScript 오류 이벤트의 수입니다.

지표	단위	설명
NavigationFrustratedCount	개수	실망 임계값(8000ms)보다 높은 duration을 사용한 탐색 이벤트 수입니다. 탐색 이벤트의 기간은 PerformanceNavigationDuration 지표에서 추적됩니다.
NavigationSatisfiedCount	개수	Apdex 목표(2000ms)보다 작은 duration을 사용한 탐색 이벤트 수입니다. 탐색 이벤트의 기간은 PerformanceNavigationDuration 지표에서 추적됩니다.
NavigationToleratedCount	개수	2000ms~8000ms의 duration을 사용한 탐색 이벤트 수입니다. 탐색 이벤트의 기간은 PerformanceNavigationDuration 지표에서 추적됩니다.

지표	단위	설명
PageViewCount	개수	앱 모니터에서 수집한 페이지 조회 이벤트 수입니다. page_view_event RUM 이벤트를 계산하여 계산됩니다.
PerformanceResourceDuration	밀리초	리소스 이벤트의 duration입니다. 추가 차원: • event_details.file.type 은 스타일시트, 문서, 이미지, 스크립트 또는 글꼴과 같은 리소스 이벤트의 파일 유형입니다. • event_type 이벤트의 유형입니다. 현재 이 차원에 대해 사용할 수 있는 유일한 값은 resource입니다.
PerformanceNavigationDuration	밀리초	탐색 이벤트의 duration입니다.

지표	단위	설명
RumEventPayloadSize	바이트	CloudWatch RUM에서 수집한 모든 이벤트의 크기입니다. 이 지표에 대한 SampleCount 통계를 사용하여 앱 모니터가 수집하고 있는 이벤트 수를 모니터링할 수 있습니다.
SessionCount	개수	앱 모니터에서 수집한 세션 시작 이벤트 수입니다. 다시 말해 시작된 새 세션 수입니다.
WebVitalsCumulativeLayoutShift	None	누적 레이아웃 시프트 이벤트의 값을 추적합니다.
WebVitalsFirstInputDelay	밀리초	첫 번째 입력 지연 이벤트의 값을 추적합니다.
WebVitalsLargestContentfulPaint	밀리초	가장 큰 콘텐츠 페인트 이벤트의 값을 추적합니다.

사용자 지정 지표 및 CloudWatch 및 CloudWatch Evidently로 전송할 수 있는 확장 지표

기본적으로 RUM 앱 모니터는 지표를 CloudWatch로 전송합니다. 이러한 기본 지표와 차원은 CloudWatch RUM을 사용하여 수집할 수 있는 CloudWatch 지표에 나열되어 있습니다.

앱 모니터를 설정하여 지표를 내보낼 수도 있습니다. 앱 모니터는 확장 지표, 사용자 지정 지표 또는 두 가지 모두를 전송할 수 있습니다. CloudWatch나 CloudWatch Evidently, 또는 두 가지 모두로 전송할 수 있습니다.

- 사용자 지정 지표 - 사용자 지정 지표는 사용자가 정의하는 지표입니다. 사용자 지정 지표를 사용하면 모든 지표 이름과 네임스페이스를 사용할 수 있습니다. 지표를 도출하기 위해 사용자 지정 이벤트, 기본 제공 이벤트, 사용자 지정 속성 또는 기본 속성을 사용할 수 있습니다.

  사용자 지정 지표를 CloudWatch와 CloudWatch Evidently 모두에 전송할 수 있습니다.
- 확장 지표 - 기본 CloudWatch RUM 지표를 CloudWatch Evidently로 전송하여 Evidently 실험에 사용할 수 있습니다. 또한 기본 CloudWatch RUM 지표를 추가 측정기준과 함께 CloudWatch로 전송할 수도 있습니다. 이렇게 하면 이러한 지표를 통해 보다 세분화된 보기를 얻을 수 있습니다.

주제

사용자 지정 지표

사용자 지정 지표를 보내려면 AWS API를 사용하거나 콘솔 대신 AWS CLI를 사용해야 합니다. AWS API 사용에 대한 자세한 내용은 PutRumMetricsDestination 및 BatchCreateRumMetricDefinitions를 참조하세요.

하나의 대상에 포함할 수 있는 확장 지표 및 사용자 지정 지표 정의의 최대 개수는 2,000개입니다. 각 대상에 보내는 각 사용자 지정 지표 또는 확장 지표에 대해 측정기준 이름과 측정기준 값의 각 조합이 이 한도에 포함됩니다. 이는 요금을 위한 CloudWatch 사용자 지정 지표로도 계산됩니다.

다음 예에서는 사용자 지정 이벤트에서 파생된 사용자 지정 지표를 만드는 방법을 보여줍니다. 다음은 사용되는 사용자 지정 이벤트의 예입니다.

```
cwr('recordEvent', {
    type: 'my_custom_event',
    data: {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
})
```

이 사용자 지정 이벤트가 주어지면 Chrome 브라우저에서 amazonaws.com URL에 대한 방문 횟수를 계산하는 사용자 지정 지표를 만들 수 있습니다. 다음 정의는 계정의 RUM/CustomMetrics/PageVisits 네임스페이스에 AmazonVisitsCount라는 지표를 생성합니다.

```
{
    "AppMonitorName":"customer-appMonitor-name",
    "Destination":"CloudWatch",
    "MetricDefinitions":[
        {
            "Name":"AmazonVisitsCount",
            "Namespace":"PageVisit",
            "ValueKey":"event_details.visit_count",
            "UnitLabel":"Count",
            "DimensionKeys":{
                "event_details.current_url": "URL"
            },
            "EventPattern":"{\"metadata\":{\"browserName\":[\"Chrome\"]},\"event_type
\":[\"my_custom_event\"],\"event_details\": {\"current_url\": [\"amazonaws.com\"]}}"
        }
    ]
}
```

확장 지표

확장 지표를 설정하면 다음 중 하나 또는 둘 다를 수행할 수 있습니다.

- 기본 CloudWatch RUM 지표를 CloudWatch Evidently로 전송하여 Evidently 실험에 사용할 수 있습니다. PerformanceNavigationDuration, PerformanceResourceDuration, WebVitalsCumulativeLayoutShift, WebVitalsFirstInputDelay 및 WebVitalsLargestContentfulPaint 지표만 Evidently로 전송할 수 있습니다.
- 지표가 보다 세분화된 보기를 제공하도록 기본 CloudWatch RUM 지표를 추가 측정기준과 함께 CloudWatch로 전송합니다. 예를 들어 사용자가 사용하는 특정 브라우저에 대한 지표나 특정 지리적 위치의 사용자에 대한 지표를 볼 수 있습니다.

기본 CloudWatch RUM 지표에 대한 자세한 내용은 CloudWatch RUM을 사용하여 수집할 수 있는 CloudWatch 지표 섹션을 참조하세요.

하나의 대상에 포함할 수 있는 확장 지표 및 사용자 지정 지표 정의의 최대 개수는 2,000개입니다. 각 대상으로 전송하는 각 확장 또는 사용자 지정 지표에 대해 측정기준 이름과 측정기준 값의 각 조합이

이 한도에 대한 확장 지표 수 계산에 포함됩니다. 이는 요금을 위한 CloudWatch 사용자 지정 지표로도 계산됩니다.

CloudWatch로 확장 지표를 전송할 때 CloudWatch RUM 콘솔을 사용하여 확장 지표에 대한 CloudWatch 경보를 생성할 수 있습니다.

확장 지표는 CloudWatch 사용자 지정 지표로 요금이 부과됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하십시오.

앱 모니터에서 전송할 수 있는 모든 지표 이름에 대한 확장 지표에 다음과 같은 차원이 지원됩니다. 이러한 지표 이름은 CloudWatch RUM을 사용하여 수집할 수 있는 CloudWatch 지표에 나열되어 있습니다.

- `BrowserName`

  차원 값의 예: `Chrome`, `Firefox`, `Chrome Headless`
- `CountryCode` 이 차원은 2자리 코드와 함께 ISO-3166 형식을 사용합니다.

  차원 값의 예: `US`, `JP`, `DE`
- `DeviceType`

  차원 값의 예: `desktop`, `mobile`, `tablet`, `embedded`
- `FileType`

  차원 값의 예: `Image`, `Stylesheet`
- `OSName`

  차원 값의 예: `Linux`, `Windows`, `iOS`, `Android`
- `PageId`

콘솔을 사용하여 확장 지표 설정

콘솔을 사용하여 CloudWatch로 확장 지표를 전송하려면 다음 단계를 사용하세요.

CloudWatch Evidently로 확장 지표를 전송하려면 콘솔 대신 AWS API 또는 AWS CLI를 사용해야 합니다. AWS API를 사용하여 확장 지표를 CloudWatch 또는 Clariby로 전송하는 방법에 대한 자세한 내용은 PutRumMetricsDestination 및 BatchCreateRumMetricDefinitions를 참조하세요.

콘솔을 사용하여 RUM 확장 지표를 CloudWatch로 전송하도록 앱 모니터를 설정하려면 다음을 수행하세요.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, RUM을 선택합니다.
3. List view(목록 보기)를 선택한 다음 지표를 전송할 앱 모니터의 이름을 선택합니다.
4. Configuration(구성) 탭을 선택한 다음 RUM extended metrics(RUM 확장 지표)를 선택합니다.
5. Send metrics(지표 전송)를 선택합니다.
6. 추가 차원과 함께 전송할 지표 이름을 하나 이상 선택합니다.
7. 이러한 지표의 차원으로 사용할 요소를 하나 이상 선택합니다. 선택 시 해당 항목이 생성하는 확장 지표 수가 Number of extended metrics(확장 지표 수)에 표시됩니다.

   이 숫자는 선택한 지표 이름의 수에 생성한 서로 다른 차원의 수를 곱하여 계산됩니다. 이 숫자는 요금이 부과되는 사용자 지정 지표의 수를 나타냅니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

   a. 페이지 ID를 차원으로 사용하여 지표를 전송하려면 Browse for page ID(페이지 ID 찾아보기)를 선택한 다음 사용할 페이지 ID를 선택합니다.
   b. 디바이스 유형을 차원으로 사용하여 지표를 전송하려면 Desktop devices(데스크톱 디바이스) 또는 Mobile and tablets(모바일 및 태블릿)를 선택합니다.
   c. 운영 체제를 차원으로 사용하여 지표를 전송하려면 Operating system(운영 체제)에서 운영 체제를 하나 이상 선택합니다.
   d. 브라우저 유형을 차원으로 사용하여 지표를 전송하려면 Browsers(브라우저)에서 브라우저를 하나 이상 선택합니다.
   e. 지리적 위치를 차원으로 사용하여 지표를 전송하려면 Locations(위치)에서 위치를 하나 이상 선택합니다.

      이 앱 모니터가 지표를 보고한 위치만 선택할 수 있는 목록에 표시됩니다.
8. 선택을 마치면 Send metrics(지표 전송)를 선택합니다.
9. (선택 사항) Extended metrics(확장 지표) 목록에서 지표 중 하나를 관찰하는 경보를 생성하려면 해당 지표 행에서 Create alarm(경보 생성)을 선택합니다.

   CloudWatch 경보에 대한 일반적인 정보는 Amazon CloudWatch 경보 사용 섹션을 참조하세요. CloudWatch RUM 확장 지표에 대한 경보를 설정하는 방법에 대한 자습서는 자습서: 확장 지표 생성 및 경보 설정 섹션을 참조하세요.

확장 지표 전송 중지

콘솔을 사용하여 확장 지표 전송을 중지하려면 다음을 수행하세요.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, RUM을 선택합니다.
3. List view(목록 보기)를 선택한 다음 지표를 전송할 앱 모니터의 이름을 선택합니다.
4. Configuration(구성) 탭을 선택한 다음 RUM extended metrics(RUM 확장 지표)를 선택합니다.
5. 전송을 중지할 지표 이름과 차원 조합을 하나 이상 선택합니다. 그런 다음 Actions(작업), Delete(삭제)를 선택합니다.

자습서: 확장 지표 생성 및 경보 설정

이 자습서에서는 CloudWatch로 전송할 확장 지표를 설정하는 방법과 해당 지표에 대한 경보를 설정하는 방법을 설명합니다. 이 자습서에서는 Chrome 브라우저에서 JavaScript 오류를 추적하는 지표를 생성합니다.

이 확장 지표를 설정하고 이에 대한 경보를 설정하려면 다음을 수행하세요.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, RUM을 선택합니다.
3. List view(목록 보기)를 선택한 다음 지표를 전송할 앱 모니터의 이름을 선택합니다.
4. Configuration(구성) 탭을 선택한 다음 RUM extended metrics(RUM 확장 지표)를 선택합니다.
5. Send metrics(지표 전송)를 선택합니다.
6. JSErrorCount를 선택합니다.
7. Browsers(브라우저)에서 Chrome을 선택합니다.

   JSErrorCount와 Chrome의 이러한 조합은 하나의 확장 지표를 CloudWatch로 전송합니다. 지표는 Chrome 브라우저를 사용하는 사용자 세션에 대해서만 JavaScript 오류를 계산합니다. 지표 이름은 JsErrorCount이고 차원 이름은 Browser(브라우저)입니다.
8. Send metrics(지표 전송)를 선택합니다.
9. Extended metrics(확장 지표) 목록에서 Name(이름) 아래에 JsErrorCount가 표시되고 BrowserName 아래에 Chrome이 표시되는 행에서 Create alarm(경보 생성)을 선택합니다.
10. Specify metric and conditions(지표 및 조건 지정)에서 Metric name(지표 이름) 및 BrowserName 필드가 올바른 값으로 미리 채워져 있는지 확인합니다.

11. Statistic(통계)에서 경보에 사용할 통계를 선택합니다. 이러한 유형의 계산 지표에는 Average(평균)가 적합합니다.
12. 기간에 대해 5분을 선택합니다.
13. 조건(Conditions)에서 다음을 수행하십시오:
    - Static(정적)을 선택합니다.
    - Greater(보다 큼)를 선택하여 오류 수가 지정하려는 임곗값보다 높을 때 경보가 ALARM 상태로 전환되도록 지정합니다.
    - than...(큼) 아래에 경보 임계값의 숫자를 입력합니다. 5분 동안 오류 수가 이 숫자를 초과하면 경보가 ALARM 상태로 전환됩니다.
14. (선택 사항) 기본적으로 오류 수가 5분 동안 설정한 임곗값 숫자를 초과하는 즉시 경보가 ALARM 상태로 전환됩니다. 이 숫자가 5분 이상 초과되는 경우에만 경보가 ALARM 상태로 전환되도록 필요에 따라 이를 변경할 수 있습니다.

    이렇게 하려면 Additional configuration(추가 구성)을 선택한 다음 Datapoints to alarm(경보를 보낼 데이터 포인트)에서 경보를 트리거하기 위해 오류 번호가 임계값을 초과해야 하는 5분 기간 수를 지정합니다. 예를 들어, 두 개의 연속된 5분 기간이 임계값을 초과하는 경우에만 경보가 트리거되도록 2개 중 2개를 선택하거나, 3개의 연속된 5분 기간 중 2개가 임계값을 초과하는 경우 경보가 트리거되도록 3개 중 2개를 선택할 수 있습니다.

    이러한 유형의 경보 평가에 대한 자세한 내용은 경보 평가 섹션을 참조하세요.
15. 다음을 선택합니다.
16. Configure actions(작업 구성)에서 경보가 경보 상태로 전환될 때 수행할 작업을 지정합니다. Amazon SNS로 알림을 받으려면 다음을 수행하세요.
    - 알림 추가를 선택합니다.
    - 경보 내를 선택합니다.
    - 기존의 SNS 주제를 선택하거나 새로 생성합니다. 새로 생성하는 경우 이름을 지정하고 이메일 주소를 하나 이상 추가합니다.
17. 다음을 선택합니다.
18. 경보의 이름과 설명(선택 사항)을 입력하고 Next(다음)를 선택합니다.
19. 세부 정보를 검토하고 Create alarm(경보 생성)을 선택합니다.

CloudWatch RUM을 통한 데이터 보호 및 데이터 프라이버시

AWS 공동 책임 모델은 Amazon CloudWatch RUM에 데이터 보호와 데이터 프라이버시를 적용합니다. 이 모델에서 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 이 인프라에서 호스팅되는 콘텐츠에 대한 통제를 유지하는 것은 사용자의 책임입니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그에서 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요. GDPR 요구 사항 준수에 대한 자세한 리소스는 일반 데이터 보호 규정(GDPR) 센터를 참조하세요.

Amazon CloudWatch RUM은 수집하려는 최종 사용자 데이터의 입력에 따라 웹 사이트 또는 웹 애플리케이션 코드에 심을 수 있는 코드 조각을 생성합니다. 코드 조각으로 다운로드 및 구성된 웹 클라이언트는 최종 사용자 데이터 수집을 돕기 위해 쿠키(또는 유사한 기술)를 사용합니다. 쿠키(또는 유사한 기술)의 사용은 특정 관할 지역의 데이터 프라이버시 보호 규정에 따라야 합니다. Amazon CloudWatch RUM을 사용하기 전에 법적으로 적절한 개인 정보 보호 고지를 제공하고 쿠키 사용 및 최종 사용자 데이터 처리(수집 포함)에 필요한 동의를 얻기 위한 모든 관련 법적 요구 사항을 포함하여 관련 법률에 따른 규정 준수 의무를 평가하는 것이 좋습니다. 웹 클라이언트가 쿠키(또는 유사한 기술)를 사용하는 방식과 웹 클라이언트가 수집하는 최종 사용자 데이터에 대한 자세한 내용은 CloudWatch RUM 웹 클라이언트에서 수집한 정보 및 CloudWatch RUM 웹 클라이언트 쿠키(또는 유사한 기술) 섹션을 참조하세요.

자유 형식 필드에 최종 사용자 계정 번호, 메일 주소, 기타 개인 정보와 같은 중요 식별 정보를 절대 입력하지 마세요. Amazon CloudWatch RUM 또는 기타 서비스에 입력하는 모든 데이터는 진단 로그에 포함될 수 있습니다.

CloudWatch RUM 웹 클라이언트 쿠키(또는 유사한 기술)

CloudWatch RUM 웹 클라이언트는 기본적으로 사용자 세션에 대한 특정 데이터를 수집합니다. 웹 클라이언트가 사용자 ID와 세션 ID를 수집하도록 페이지 로드 전반에 지속되는 쿠키를 활성화할 수 있습니다. 사용자 ID는 RUM에 의해 임의로 생성됩니다.

이러한 쿠키가 활성화되면 이 앱 모니터에 대한 RUM 대시보드를 볼 때 RUM은 다음과 같은 유형의 데이터를 표시할 수 있습니다.

- 고유 사용자 수 및 오류를 겪은 여러 사용자 수와 같은 사용자 ID를 기반으로 집계된 데이터.
- 세션 수 및 오류가 발생한 세션 수와 같은 세션 ID를 기반으로 집계된 데이터.
- 사용자 여정인 샘플링된 각 사용자 세션에 포함된 페이지 시퀀스.

> **Important**
>
> 이러한 쿠키(또는 유사한 기술)를 활성화하지 않아도 웹 클라이언트는 브라우저 유형 및 버전, 운영 체제 유형 및 버전, 디바이스 유형 등의 최종 사용자 세션에 대한 특정 정보를 기록합니다. 웹 바이탈, 페이지 보기, 오류가 발생한 페이지와 같은 집계된 페이지별 인사이트를 제공하기 위해 수집됩니다. 데이터 기록에 대한 자세한 내용은 CloudWatch RUM 웹 클라이언트에서 수집한 정보 섹션을 참조하세요.

CloudWatch RUM 웹 클라이언트에서 수집한 정보

이 섹션에서는 PutRumEvents 스키마 즉, CloudWatch RUM을 사용하여 사용자 세션에서 수집할 수 있는 데이터의 구조를 정의합니다.

PutRumEvents 요청은 다음 필드가 있는 데이터 구조를 CloudWatch RUM으로 보냅니다.

- 이 RUM 이벤트 배치의 ID
- 다음을 포함하는 앱 모니터 세부 정보:
  - 앱 모니터 ID
  - 모니터링된 애플리케이션 버전
- 다음을 포함하는 사용자 세부 정보. 이 정보는 앱 모니터에 쿠키가 활성화된 경우에만 수집됩니다.
  - 웹 클라이언트에서 생성된 사용자 ID
  - 세션 ID
- 이 배치의 RUM 이벤트 배열

RUM 이벤트 스키마

각 RUM 이벤트의 구조에는 다음 필드가 포함됩니다.

- 이벤트 ID
- 타임스탬프
- 이벤트 유형
- 사용자 에이전트
- Metadata
- RUM 이벤트 세부 정보

RUM 이벤트 메타데이터

메타데이터에는 페이지 메타데이터, 사용자 에이전트 메타데이터, 지리적 위치 메타데이터 및 도메인 메타데이터가 포함됩니다.

페이지 메타데이터

페이지 메타데이터에는 다음이 포함됩니다.

- 페이지 ID
- 페이지 제목
- 상위 페이지 ID입니다. – 이 정보는 앱 모니터에 쿠키를 사용하도록 설정한 경우에만 수집됩니다.
- 상호 작용 깊이 - 이 정보는 앱 모니터에 쿠키가 사용하도록 설정된 경우에만 수집됩니다.
- 페이지 태그 - 페이지 이벤트에 태그를 추가하여 페이지를 그룹화할 수 있습니다. 자세한 내용은 페이지 그룹 사용 단원을 참조하십시오.

사용자 에이전트 메타데이터

사용자 에이전트 메타데이터에는 다음이 포함됩니다.

- 브라우저 언어
- 브라우저 이름
- 브라우저 버전
- 운영 체제 이름
- 운영 체제 버전
- 디바이스 유형
- 플랫폼 유형

지리적 메타데이터

지리적 메타데이터에는 다음이 포함됩니다.

- 국가 코드
- 세분화 코드

도메인 메타데이터

도메인 메타데이터에는 URL 도메인이 포함됩니다.

RUM 이벤트 세부 정보

이벤트 세부 정보는 이벤트 유형에 따라 다음 유형의 스키마 중 하나를 따릅니다.

세션 시작 이벤트

이 이벤트에는 필드가 없습니다. 이 정보는 앱 모니터에 쿠키가 활성화된 경우에만 수집됩니다.

페이지 보기 스키마

페이지 보기(Page view) 이벤트에는 다음과 같은 속성이 포함됩니다. 웹 클라이언트를 구성하여 페이지 보기 모음을 비활성화할 수 있습니다. 자세한 내용은 CloudWatch RUM 웹 클라이언트 설명서를 참조하세요.

명칭	유형	설명
페이지 ID	String	애플리케이션 내에서 이 페이지를 고유하게 나타내는 ID. 기본적으로 URL 경로입니다.
상위 페이지 ID	String	사용자가 현재 페이지로 이동할 때 사용했던 페이지 ID. 이 정보는 앱 모니터에 쿠키가 활성화된 경우에만 수집됩니다.
상호 작용 깊이	String	이 정보는 앱 모니터에 쿠키가 활성화된 경우에만 수집됩니다.

JavaScript 오류 스키마

에이전트에서 생성된 JavaScript 오류 이벤트에는 다음 속성이 포함됩니다. 웹 클라이언트는 오류 원격 측정을 수집하도록 선택한 경우에만 이러한 이벤트를 수집합니다.

명칭	유형	설명
오류 유형	String	오류 이름(존재하는 경우). 자세한 내용은 Error.prototype.name을 참조하세요.

명칭	유형	설명
		일부 브라우저는 오류 유형을 지원하지 않을 수 있습니다.
오류 메시지	String	오류 메시지 자세한 내용은 Error.prototype.message를 참조하세요. 오류 필드가 없으면 오류 이벤트 메시지입니다. 자세한 내용은 ErrorEvent를 참조하세요. 여러 브라우저에서 오류 메시지가 일관되지 않을 수 있습니다.
스택 추적	String	오류의 스택 추적이 있는 경우 150자로 잘립니다. 자세한 내용은 Error.prototype.stack을 참조하세요. 일부 브라우저는 스택 추적을 지원하지 않을 수 있습니다.

DOM 이벤트 스키마

에이전트에서 생성된 문서 객체 모델(DOM) 이벤트에는 다음과 같은 속성이 포함되어 있습니다. 이러한 이벤트는 기본적으로 수집되지 않습니다. 상호 작용 원격 측정을 활성화하는 경우에만 수집됩니다. 자세한 내용은 CloudWatch RUM 웹 클라이언트 설명서를 참조하세요.

명칭	유형	설명
이벤트	String	클릭, 스크롤 또는 마우스오버와 같은 DOM 이벤트의 유형. 자세한 내용은 이벤트 참조를 참조하세요.
Element	String	DOM 요소 유형
ID 요소	String	이벤트를 생성한 요소에 ID가 있는 경우 이 속성은 해당 ID를 저장합니다. 자세한 내용은 Element.id를 참조하세요.
CSSLocator	String	DOM 요소를 식별하는 데 사용되는 CSS 로케이터입니다.
InteractionId	String	사용자와 UI 간의 상호 작용을 위한 고유 ID입니다.

탐색 이벤트 스키마

탐색 이벤트는 앱 모니터에 성능 원격 측정이 활성화된 경우에만 수집됩니다.

탐색 이벤트는 탐색 타이밍 레벨 1 및 탐색 타이밍 레벨 2 API를 사용합니다. 레벨 2 API는 모든 브라우저에서 지원되지 않으므로 이러한 최신 필드는 선택 사항입니다.

> **Note**
>
> 타임스탬프 지표는 DOMHighResTimestamp를 기반으로 합니다. 레벨 2 API의 경우 모든 타이밍은 기본적으로 `startTime`과 관련되어 있습니다. 하지만 레벨 1의 경우 상대값을 얻기 위해 타임스탬프 지표에서 `navigationStart` 지표를 뺍니다. 모든 타임스탬프 값은 밀리초 단위입니다.

탐색 이벤트에는 다음 속성이 포함됩니다.

명칭	유형	설명	참고
initiatorType	String	성능 이벤트를 시작한 리소스 유형을 나타냅니다.	값: "탐색" 레벨 1: "탐색" 레벨 2: entryData.initiatorType
navigationType	String	탐색 유형을 나타냅니다. 이 속성은 필수가 아닙니다.	값: 값은 다음 중 하나여야 합니다. • `navigate`는 링크를 선택하거나, 브라우저의 주소 표시줄에 URL을 입력하거나, 양식을 제출하거나, `reload`

명칭	유형	설명	참고
			또는 back_forward 이외의 스크립트 작업을 통해 초기화하여 시작하는 탐색입니다. • reload는 브라우저의 재로드 작업 또는 location.reload()를 통한 탐색입니다. • back_forward 는 브라우저의 기록 순회 작업을 통한 탐색입니다. • prerender 는 프리렌더 힌트에 의해 시작된 탐색입니다. 자세한 내용은 프리렌더를

명칭	유형	설명	참고
			참조하세요.
startTime	숫자	이벤트가 트리거되는 시점을 나타냅니다.	값: 0 레벨 1: entryData.navigationStart - entryData.navigationStart 레벨 2: entryData.startTime

명칭	유형	설명	참고
unloadEventStart	숫자	창의 이전 문서가 unload 이벤트가 발생 후 언로드되기 시작한 시간을 나타냅니다.	값: 이전 문서가 없거나 이전 문서 또는 필요한 리디렉션 중 하나가 동일한 원본이 아닌 경우 반환되는 값은 0입니다. 레벨 1: `entryData.unloadEventStart > 0 ? entryData.unloadEventStart - entryData.navigationStart : 0` 레벨 2: entryData.unloadEventStart

명칭	유형	설명	참고
promptForUnload	숫자	문서를 언로드하는 데 걸린 시간입니다. 즉, `unloadEventStart` 및 `unloadEventEnd` 사이의 시간에서 `UnloadEventEnd` 는 언로드 이벤트 핸들러가 완료된 순간을 밀리초로 나타냅니다.	값: 이전 문서가 없거나 이전 문서 또는 필요한 리디렉션 중 하나가 동일한 원본이 아닌 경우 반환되는 값은 0입니다. 레벨 1: entryData.unloadEventEnd - entryData.unloadEventStart 레벨 2: entryData.unloadEventEnd - entryData.unloadEventStart

명칭	유형	설명	참고
redirectCount	숫자	현재 탐색 컨텍스트에서 마지막으로 리디렉션되지 않은 탐색 이후의 리디렉션 수를 나타내는 숫자입니다. 이 속성은 필수가 아닙니다.	값: 리디렉션이 없거나 대상 문서와 같은 원본이 아닌 리디렉션이 있는 경우 반환되는 값은 0입니다. 레벨 1: 사용할 수 없음 레벨 2: entryData.redirectCount

명칭	유형	설명	참고
redirectStart	숫자	첫 번째 HTTP 리디렉션이 시작되는 시간입니다.	값: 리디렉션이 없거나 대상 문서와 같은 원본이 아닌 리디렉션이 있는 경우 반환되는 값은 0입니다. 레벨 1: `entryData.redirectStart > 0 ? entryData.redirectStart - entryData.navigationStart : 0` 레벨 2: entryData.redirectStart

명칭	유형	설명	참고
redirectTime	숫자	HTTP 리디렉션에 걸린 시간입니다. `redirectStart` 및 `redirectEnd` 의 차이점입니다.	레벨 1:: entryData .redirectEnd - entryData .redirectStart 레벨 2:: entryData .redirectEnd - entryData .redirectStart

명칭	유형	설명	참고
workerStart	숫자	이것은 `PerformanceResourceTiming` 인터페이스 속성입니다. 작업자 스레드 작업의 시작을 표시합니다. 이 속성은 필수가 아닙니다.	값: 서비스 작업자 스레드가 이미 실행 중이거나 서비스 작업자 스레드를 시작하기 직전에 이 속성은 `FetchEvent` 디스패치 직전의 시간을 반환합니다. 서비스 작업자가 리소스를 가로채지 않으면 0을 반환합니다. 레벨 1: 사용할 수 없음 레벨 2: entryData.workerStart

명칭	유형	설명	참고
workerTime	숫자	서비스 작업자가 리소스를 가로채면 작업자 스레드 작업에 필요한 시간이 반환됩니다. 이 속성은 필수가 아닙니다.	레벨 1: 사용할 수 없음 레벨 2: `entryData.workerStart > 0 ? entryData.fetchStart - entryData.workerStart : 0`
fetchStart	숫자	브라우저가 HTTP 요청을 사용하여 문서를 가져올 준비가 된 시간입니다. 이는 애플리케이션 캐시를 확인하기 전입니다.	레벨 1: `: entryData.fetchStart > 0 ? entryData.fetchStart - entryData.navigationStart : 0` 레벨 2: entryData.fetchStart

명칭	유형	설명	참고
domainLookupStart	숫자	도메인 조회가 시작되는 시간입니다.	값: 영구 연결이 사용되거나 정보가 캐시 또는 로컬 리소스에 저장되어 있는 경우 값은 `fetchStart` 와 같습니다. 레벨 1: `entryData.domainLookupStart > 0 ? entryData.domainLookupStart - entryData.navigationStart : 0` 레벨 2: entryData.domainLookupStart

명칭	유형	설명	참고
dns	숫자	도메인 조회에 필요한 시간입니다.	값: 리소스 및 DNS 레코드가 캐시된 경우 예상 값은 0입니다. 레벨 1: entryData.domainLookupEnd - entryData.domainLookupStart 레벨 2: entryData.domainLookupEnd - entryData.domainLookupStart
nextHopProtocol	String	리소스를 가져오는 데 사용되는 네트워크 프로토콜을 나타내는 문자열입니다. 이 속성은 필수가 아닙니다.	레벨 1: 사용할 수 없음 레벨 2: entryData.nextHopProtocol

명칭	유형	설명	참고
connectStart	숫자	사용자 에이전트가 문서 검색을 위해 서버에 대한 연결 설정을 시작하기 직전의 시간입니다.	값: RFC2616 영구 연결이 사용되거나 현재 문서가 관련 애플리케이션 캐시 또는 로컬 리소스에서 검색되는 경우 이 속성은 domainLookupEnd 값을 반환합니다. 레벨 1: `entryData.connectStart > 0 ? entryData.connectStart - entryData.navigationStart : 0` 레벨 2: entryData.connectStart

명칭	유형	설명	참고
connect	숫자	전송 연결을 설정하거나 SSL 인증을 수행하는 데 필요한 시간을 측정합니다. 또한 브라우저에서 실행한 동시 요청이 너무 많을 때 걸리는 차단된 시간도 포함됩니다.	레벨 1: entryData .connectEnd - entryData .connectStart 레벨 2: entryData .connectEnd - entryData .connectStart
secureConnectionStart	숫자	현재 페이지의 URL 스키마가 "https"인 경우 이 속성은 사용자 에이전트가 현재 연결을 보호하기 위해 핸드셰이크 프로세스를 시작하기 직전의 시간을 반환합니다. HTTPS를 사용하지 않으면 0을 반환합니다. URL 스키마에 대한 자세한 내용은 URL 표시를 참조하세요.	공식: entryData .secureConnectionStart

명칭	유형	설명	참고
tlsTime	숫자	SSL 핸드셰이크를 완료하는 데 걸린 시간입니다.	레벨 1: `entryData.secureConnectionStart > 0 ? entryData.connectEnd - entryData.secureConnectionStart : 0` 레벨 2: `entryData.secureConnectionStart > 0 ? entryData.connectEnd - entryData.secureConnectionStart : 0`

명칭	유형	설명	참고
requestStart	숫자	사용자 에이전트가 서버나 관련 애플리케이션 캐시 또는 로컬 리소스에서 리소스 요청을 시작하기 직전의 시간입니다.	레벨 1: `: entryData.requestStart > 0 ? entryData.requestStart - entryData.navigationStart : 0` 레벨 2: entryData.requestStart
timeToFirstByte	숫자	요청 후 첫 번째 바이트 정보를 수신하는 데 걸린 시간입니다. 이 시간은 `startTime` 을 기준으로 합니다.	레벨 1: entryData.responseStart - entryData.requestStart 레벨 2: entryData.responseStart - entryData.requestStart

명칭	유형	설명	참고
responseStart	숫자	사용자 에이전트의 HTTP 파서가 관련 애플리케이션 캐시 또는 로컬 리소스 또는 서버에서 응답의 첫 번째 바이트를 수신한 직후의 시간입니다.	레벨 1: `entryData.responseStart > 0 ? entryData.responseStart - entryData.navigationStart : 0` 레벨 2: entryData.responseStart

명칭	유형	설명	참고
responseTime	String	관련 애플리케이션 캐시, 로컬 리소스 또는 서버에서 바이트 형식으로 전체 응답을 수신하는 데 걸린 시간입니다.	레벨 1: `entryData.responseStart > 0 ? entryData.responseEnd - entryData.responseStart : 0` 레벨 2: `entryData.responseStart > 0 ? entryData.responseEnd - entryData.responseStart : 0`

명칭	유형	설명	참고
domIntera ctive	숫자	파서가 기본 문서에서 작업을 완료하고 HTML DOM이 생성되는 시간입니다. 현 시점에서 `Document.readyState` 가 "대화식"으로 바뀌며 해당 `readystatechange` 이벤트가 발생합니다.	레벨 1: `entryData .domInter active > 0 ? entryData .domInter active - entryData .navigati onStart : 0` 레벨 2: entryData .domInter active

명칭	유형	설명	참고
domContentLoadedEventStart	숫자	사용자 에이전트가 현재 문서에서 DOMContentLoaded 이벤트를 실행하기 직전의 시간과 동일한 시간 값을 나타냅니다. DomContentLoad 이벤트는 초기 HTML 문서가 완전히 로드되고 구문 분석되면 발생합니다. 이때 기본 HTML 문서의 구문 분석이 완료되고, 브라우저가 렌더 트리를 구성하기 시작하며, 하위 리소스는 여전히 로드되어야 합니다. 이는 스타일 시트, 이미지 및 하위 프레임이 로드될 때까지 기다리지 않습니다.	레벨 1: `entryData.domContentLoadedEventStart > 0 ? entryData.domContentLoadedEventStart - entryData.navigationStart : 0` 레벨 2: entryData.domContentLoadedEventStart

명칭	유형	설명	참고
domContentLoaded	숫자	렌더 트리 구성의 시작 시간과 종료 시간은 `domContentLoadedEventStart` 및 `domContentLoadedEventEnd` 로 표시됩니다. CloudWatch RUM이 실행을 추적할 수 있습니다. 이 속성은 `domContentLoadedStart` 및 `domContentLoadedEnd` 의 차이점입니다. 이 기간 동안 DOM과 CSSOM이 준비됩니다. 이 속성은 비동기 및 동적으로 생성된 스크립트를 제외하고 스크립트 실행을 기다립니다. 스크립트가 스타일 시트에 의존하는 경우 `domContentLoaded` 는 스타일 시트에서도 기다립니다. 이미지를 기다리지 않습니다. **Note** `domContentLoadedStart` 와 `domContentLoadedEnd` 의 실제 값은 Google Chrome의 네트워크 패널의 `domContentLoaded` 에 근접합니다. 페이지 로딩 프로세스의 시작부터 HTML DOM + CSSOM 렌더링 트리 생성 시간을 나타냅니다. 탐색 지표의 경우 `domContentLoaded` 값은 시작 값과 끝 값의 차이를 나타냅니다. 이 값은 하위 리소스와 렌더링 트리 구성만 다운로드하는 데 필요한 시간입니다.	레벨 2: entryData.domContentLoadedEventEnd - entryData.domContentLoadedEventStart 레벨 2: entryData.domContentLoadedEventEnd - entryData.domContentLoadedEventStart

명칭	유형	설명	참고
domComplete	숫자	브라우저가 현재 문서 준비를 완료하도록 설정하기 바로 전의 시간입니다. 이때 이미지와 같은 하위 리소스의 로드가 완료됩니다. 여기에는 CSS 및 동기식 JavaScript와 같은 차단 콘텐츠를 다운로드하는 데 걸리는 시간이 포함됩니다. 이 시간은 Google Chrome 네트워크 패널의 `loadTime`에 근접합니다.	레벨 1: `entryData.domComplete > 0 ? entryData.domComplete - entryData.navigationStart : 0` 레벨 2: entryData.domComplete
domProcessingTime	숫자	응답과 로드 이벤트 시작 간 총 시간입니다.	레벨 1: entryData.loadEventStart - entryData.responseEnd 레벨 2: entryData.loadEventStart - entryData.responseEnd

명칭	유형	설명	참고
loadEvent Start	숫자	현재 문서의 `load` 이벤트 직전 시간입니다.	레벨 1: `entryData .loadEvent tStart > 0 ? entryData .loadEvent tStart - entryData .navigationStart : 0` 레벨 2: entryData .loadEvent tStart
loadEvent Time	숫자	`loadEventStart` 와 `loadEventEnd` 의 차이점. 이 로드 이벤트를 기다리는 추가 함수 또는 로직은 이 시간 동안 실행됩니다.	레벨 1: entryData .loadEvent tEnd - entryData .loadEvent tStart 레벨 2: entryData .loadEvent tEnd - entryData .loadEvent tStart

명칭	유형	설명	참고
duration	String	기간은 총 페이지 로드 시간입니다. 기본 페이지와 모든 동기식 하위 리소스를 다운로드하고 페이지를 렌더링하는 타이밍을 기록합니다. 스크립트와 같은 비동기 리소스는 나중에 계속 다운로드됩니다. 이는 `loadEventEnd` 및 `startTime` 속성 간 차이점입니다.	레벨 1: entryData .loadEvent End - entryData .navigati onStart 레벨 2: entryData .duration
headerSize	숫자	`transferSize` 및 `encodedBodySize` 간 차이를 반환합니다. 이 속성은 필수가 아닙니다.	레벨 1: 사용할 수 없음 레벨 2: entryData .transferSize - entryData .encodedB odySize 레벨 2: entryData .transferSize - entryData .encodedB odySize

명칭	유형	설명	참고
compressionRatio	숫자	`encodedBodySize` 및 `decodedBodySize` 의 비율. `encodedBodySize` 값은 HTTP 헤더를 제외한 리소스의 압축된 크기입니다. `decodedBodySize` 값은 HTTP 헤더를 제외한 리소스의 압축 해제된 크기입니다. 이 속성은 필수가 아닙니다.	레벨 1: 해당 사항 없음. 레벨 2: `entryData.encodedBodySize > 0 ? entryData.decodedBodySize / entryData.encodedBodySize : 0`
navigationTimingLevel	숫자	탐색 타이밍 API 버전.	값: 1 또는 2

리소스 이벤트 스키마

탐색 이벤트는 앱 모니터에 성능 원격 측정이 활성화된 경우에만 수집됩니다.

타임스탬프 지표는 The DOMHighResTimeStamp typedef를 기반으로 합니다. 레벨 2 API의 경우 모든 타이밍은 기본적으로 `startTime`과 관련됩니다. 하지만 레벨 1 API의 경우 상대값을 얻기 위해 타임스탬프 지표에서 `navigationStart` 지표를 뺍니다. 모든 타임스탬프 값은 밀리초 단위입니다.

에이전트에서 생성된 리소스 이벤트에는 다음 속성이 포함됩니다.

명칭	유형	설명	참고
targetUrl	String	리소스 URL을 반환합니다.	공식: entryData.name

명칭	유형	설명	참고
initiatorType	String	성능 리소스 이벤트를 시작한 리소스 유형을 나타냅니다.	값: "리소스" 공식: entryData .initiatorType
duration	String	`responseEnd` 및 `startTime` 속성 간 차이를 반환합니다. 이 속성은 필수가 아닙니다.	공식: entryData .duration
transferSize	숫자	응답 헤더 필드와 응답 페이로드 본문을 포함하여 가져온 리소스 크기(옥텟 단위)를 반환합니다. 이 속성은 필수가 아닙니다.	공식: entryData .transferSize
fileType	String	대상 URL 패턴에서 파생된 확장입니다.	

가장 큰 콘텐츠 페인트 이벤트 스키마

가장 큰 콘텐츠 페인트 이벤트에는 다음 속성이 포함됩니다.

이러한 이벤트는 활성화된 성능 원격 측정이 앱 모니터에 있는 경우에만 수집됩니다.

명칭	설명		
값	자세한 내용은 웹 바이탈을 참조하세요.		

첫 번째 입력 지연 이벤트

첫 번째 입력 지연 이벤트는 다음 속성을 포함합니다.

이러한 이벤트는 활성화된 성능 원격 측정이 앱 모니터에 있는 경우에만 수집됩니다.

명칭	설명		
값	자세한 내용은 웹 바이탈을 참조하세요.		

누적 레이아웃 시프트 이벤트

누적 레이아웃 시프트 이벤트에는 다음 속성이 포함됩니다.

이러한 이벤트는 활성화된 성능 원격 측정이 앱 모니터에 있는 경우에만 수집됩니다.

명칭	설명		
값	자세한 내용은 웹 바이탈을 참조하세요.		

HTTP 이벤트

HTTP 이벤트에는 다음 속성이 포함될 수 있습니다. `Response` 필드 또는 `Error` 필드 중 하나를 포함하되 두 필드 모두를 포함하지는 않습니다.

이러한 이벤트는 활성화된 HTTP 원격 측정이 앱 모니터에 있는 경우에만 수집됩니다.

명칭	설명
요청	요청 필드에는 다음이 포함됩니다. • `Method` 필드에는 GET, POST 등의 값을 가질 수 있습니다. • URL
응답	응답 필드에는 다음 항목이 포함됩니다. • 2xx, 4xx 또는 5xx 등의 상태 • 상태 텍스트

명칭	설명
오류	응답 필드에는 다음 항목이 포함됩니다. • 유형 • 메시지 • 파일 이름 • 행 번호 • 열 번호 • 스택 추적

X-Ray 추적 이벤트 스키마

이러한 이벤트는 앱 모니터에 X-Ray 추적이 활성화된 경우에만 수집됩니다.

X-Ray 추적 이벤트 스키마에 대한 자세한 내용은 AWS X-Ray 세그먼트 설명서를 참조하세요.

단일 페이지 애플리케이션의 경로 변경 타이밍

기존의 다중 페이지 애플리케이션에서 사용자가 새 콘텐츠를 로드하도록 요청하면 실제로 서버에서 새 HTML 페이지를 요청하게 됩니다. 결과적으로 CloudWatch RUM 웹 클라이언트는 일반 성능 API 지표를 사용하여 로드 시간을 캡처합니다.

하지만 단일 페이지 웹 애플리케이션은 JavaScript와 Ajax를 사용하여 서버에서 새 페이지를 로드하지 않고도 인터페이스를 업데이트합니다. 단일 페이지 업데이트는 브라우저 타이밍 API에 의해 기록되지 않으며, 대신 경로 변경 타이밍을 사용합니다.

CloudWatch RUM은 서버의 전체 페이지 로드와 단일 페이지 업데이트에 대한 모니터링을 모두 지원하지만 다음과 같은 차이점이 있습니다.

- 경로 변경 타이밍의 경우 `tlsTime`, `timeToFirstByte` 등과 같은 브라우저에서 제공하는 지표가 없습니다.
- 경로 변경 타이밍의 경우 `initiatorType` 필드가 `route_change`입니다.

CloudWatch RUM 웹 클라이언트는 경로 변경을 유발할 수 있는 사용자 상호 작용을 수신하며, 이러한 사용자 상호 작용이 기록되면 웹 클라이언트는 타임스탬프를 기록합니다. 그리고 다음 두 가지 조건을 모두 만족할 때 경로 변경 타이밍이 시작됩니다.

- 경로 변경을 수행하는 데 브라우저 기록 API(브라우저 앞으로 및 뒤로 버튼 제외)가 사용되었습니다.
- 경로 변경 감지 시간과 최신 사용자 상호 작용 타임스탬프 간의 차이가 1,000ms 미만입니다. 이는 데이터 스큐를 방지합니다.

그런 다음 경로 변경 타이밍이 시작되면 진행 중인 AJAX 요청 및 DOM 변형이 없는 경우 해당 타이밍이 완료됩니다. 그리고 마지막으로 완료된 활동의 타임스탬프가 완료 타임스탬프로 사용됩니다.

10초 이상 진행 중인 AJAX 요청 또는 DOM 변형이 있는 경우 경로 변경 타이밍이 시간 초과됩니다(기본값). 이 경우 CloudWatch RUM 웹 클라이언트는 더 이상 이 경로 변경에 대한 타이밍을 기록하지 않습니다.

결과적으로, 경로 변경 이벤트의 기간이 다음과 같이 계산됩니다.

```
(time of latest completed activity) - (latest user interaction timestamp)
```

CloudWatch RUM을 사용하는 애플리케이션 관리

이 섹션의 단계에 따라 애플리케이션의 CloudWatch RUM 사용을 관리합니다.

이미 생성한 코드 조각을 찾으려면 어떻게 해야 한가요?

애플리케이션용으로 이미 생성한 CloudWatch RUM 코드 조각을 찾으려면 다음 단계를 수행하세요.

이미 생성한 코드 조각 찾기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, RUM을 선택합니다.
3. 목록 보기(List view)를 선택합니다.
4. 앱 모니터 이름 옆에서 JavaScript 보기(View JavaScript)를 선택합니다
5. JavaScript 코드 조각(JavaScript Snippet) 창에서 클립보드로 복사(Copy to clipboard)를 선택합니다.

애플리케이션 편집

앱 모니터 설정을 변경하려면 다음 단계를 따르세요. 앱 모니터 이름을 제외한 모든 설정을 변경할 수 있습니다.

애플리케이션에서 CloudWatch RUM을 사용하는 방법 편집

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, RUM을 선택합니다.
3. 목록 보기(List view)를 선택합니다.
4. 애플리케이션 이름 옆의 버튼을 선택한 다음 작업(Actions)에서 편집(Edit)을 선택합니다.
5. 애플리케이션 이름을 제외한 모든 설정을 변경할 수 있습니다. 설정에 대한 자세한 내용은 2단계: 앱 모니터 생성 섹션을 참조하세요.
6. 완료하였으면 저장을 선택합니다.

   설정을 변경하면 코드 조각이 변경됩니다. 이제 업데이트된 코드 조각을 애플리케이션에 붙여넣어야 합니다.
7. JavaScript 코드 조각이 생성된 후 클립보드로 복사(Copy to clipboard) 또는 다운로드(Download)를 선택한 다음 완료(Done)를 선택합니다.

   새 설정으로 모니터링을 시작하려면 애플리케이션에 코드 조각을 삽입합니다. 코드 조각을 `<body>` 요소 또는 기타 `<script>` 태그 앞의 애플리케이션 `<head>` 요소에 삽입합니다.

CloudWatch RUM 사용을 중지하거나 앱 모니터 삭제

애플리케이션에서 CloudWatch RUM 사용을 중지하려면 애플리케이션 코드에서 RUM이 생성한 코드 조각을 제거합니다.

RUM 앱 모니터를 삭제하려면 다음 단계를 따르세요.

앱 모니터 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Application Signals, RUM을 선택합니다.
3. 목록 보기(List view)를 선택합니다.
4. 애플리케이션 이름 옆의 버튼을 선택한 다음 작업(Actions)에서 삭제(Delete)를 선택합니다.
5. 확인 필드에 **`Delete`**를 입력한 다음 삭제(Delete)를 선택합니다.
6. 아직 생성하지 않은 경우 애플리케이션 코드에서 CloudWatch RUM 코드 조각을 삭제합니다.

CloudWatch RUM 할당량

CloudWatch RUM은 다음과 같은 할당량이 있습니다.

Resource	기본 할당량
앱 모니터	계정당 20개 할당량 증가를 요청할 수 있습니다.
RUM 수집 속도	초당 PutRumEvents 요청(TPS) 50개 할당량 증가를 요청할 수 있습니다.

CloudWatch RUM 문제 해결

이 섹션에서는 CloudWatch RUM 문제를 해결하는 데 도움이 되는 팁을 제공합니다.

내 애플리케이션에 대한 데이터가 없습니다

먼저 코드 조각이 애플리케이션에 올바르게 삽입되었는지 확인합니다. 자세한 내용은 4단계: 애플리케이션에 코드 조각 삽입 단원을 참조하십시오.

해당 문제가 아닌 경우 아직 애플리케이션에 대한 트래픽이 없는 것일 수 있습니다. 사용자와 동일한 방식으로 애플리케이션에 액세스하여 일부 트래픽을 생성합니다.

내 애플리케이션에 대한 데이터 기록이 중지되었습니다

애플리케이션이 업데이트되었을 수 있으며 이제 CloudWatch RUM 코드 조각이 더 이상 포함되지 않습니다. 애플리케이션 코드를 확인합니다.

또 다른 가능성은 누군가가 코드 조각을 업데이트했지만 업데이트 된 코드 조각을 애플리케이션에 삽입하지 않았을 수도 있습니다. 이미 생성한 코드 조각을 찾으려면 어떻게 해야 한가요?의 지시에 따라 업데이트된 올바른 코드 조각을 찾고 애플리케이션에 붙여넣은 코드 조각과 비교합니다.

네트워크 모니터링

이 섹션의 주제에서는 Amazon CloudWatch Internet Monitor 및 Amazon CloudWatch Network Monitor에서 제공하는 CloudWatch 네트워크 및 인터넷 모니터링 기능을 설명합니다. 이러한 서비스를 통해 AWS에서 호스팅되는 애플리케이션의 네트워크, 인터넷 성능, 가용성에 대한 운영 가시성을 확보할 수 있습니다.

- Internet Monitor는 AWS가 글로벌 네트워킹 공간에서 수집한 연결 데이터를 사용하여 인터넷 트래픽의 성능 및 가용성에 대한 기준선을 계산합니다. 트래픽 패턴 및 상태 이벤트의 글로벌 보기를 보고, 이벤트에 대한 정보를 쉽게 드릴다운할 수 있습니다. 또한 애플리케이션 클라이언트에 영향을 미치는 인터넷 상태 이벤트 관련 알림을 받을 수 있습니다. 게다가 Amazon CloudFront를 사용하거나 여러 AWS 리전를 통해 라우팅함으로써 Internet Monitor에서 제공하는 인사이트를 기반으로 클라이언트 환경의 개선 가능성을 탐색할 수 있습니다.
- Network Monitor는 하이브리드 네트워크 연결의 지연 시간과 패킷 손실을 추적하고 시각화가 가능하도록 완전 관리형 에이전트 접근 방식을 사용합니다. 측정값을 수집하고 Network Monitor에서 애플리케이션에 대한 상태 이벤트 알림을 생성할 수 있도록 하려면 AWS에 호스팅된 리소스에서 온프레미스 대상 IP 주소로 전송되는 프로브를 생성해야 합니다. 네트워크 성능을 모니터링하기 위해 추가 에이전트를 설치할 필요가 없습니다. Internet Monitor와 마찬가지로 알림과 임곗값을 설정하고, 문제를 빠르게 해결하는 데 도움이 되는 정보를 확보한 다음 최종 사용자 환경을 개선하기 위한 조치를 취할 수 있습니다.

주제

Amazon CloudWatch Internet Monitor 사용

Amazon CloudWatch Internet Monitor는 인터넷 문제가 AWS에서 호스트되는 애플리케이션과 최종 사용자 간의 성능 및 가용성에 미치는 영향에 대한 가시성을 제공합니다. 인터넷 문제를 진단하는 데 걸리는 시간을 며칠에서 몇 분으로 단축할 수 있습니다. Internet Monitor는 AWS가 글로벌 네트워킹 공간에서 수집한 연결 데이터를 사용하여 인터넷 트래픽의 성능 및 가용성에 대한 기준선을 계산합니다. 이는 AWS에서 인터넷 가동 시간과 가용성을 모니터링하는 데 사용하는 것과 동일한 데이터입니다. 이러한 측정값을 기준으로 Internet Monitor는 애플리케이션이 실행되는 다양한 지리적 위치에 있는 최종 사용자(클라이언트)에게 심각한 문제가 있는 경우 이를 알려줍니다.

Amazon CloudWatch 콘솔에서는 트래픽 패턴과 상태 이벤트에 대한 글로벌 보기를 볼 수 있으며 다양한 지리적 세부 정보(위치)에서 이벤트에 대한 정보를 쉽게 자세히 분석할 수 있습니다. 영향을 명확하게 시각화하고 영향을 받는 클라이언트 위치 및 네트워크(ASN, 일반적으로 인터넷 서비스 제공업체(ISP))를 정확히 파악할 수 있습니다. Internet Monitor는 인터넷 가용성 또는 성능 문제가 특정 ASN 또는 AWS 네트워크에 의해 발생했다고 판단하는 경우 해당 정보를 제공합니다.

Internet Monitor의 주요 기능

- Internet Monitor는 또한 최종 사용자의 경험을 개선하는 데 도움이 되는 인사이트와 권장 사항을 제안합니다. 다른 서비스를 사용하도록 전환하거나 다른 AWS 리전를 통해 워크로드 트래픽을 다시 라우팅하여 애플리케이션의 예상 지연 시간을 개선하는 방법을 거의 실시간으로 탐색할 수 있습니다.
- Internet Monitor를 사용하면 애플리케이션의 성능과 가용성에 영향을 미치는 요소를 빠르게 식별하여 문제를 추적하고 해결할 수 있습니다.
- Internet Monitor는 인터넷 측정값을 CloudWatch 로그 및 CloudWatch 지표에 게시하여 애플리케이션에 특정한 위치 및 ASN(인터넷 서비스 공급자)에 대한 상태 정보와 함께 CloudWatch 도구 사용을 지원합니다. 원하는 경우, 인터넷 측정값을 Amazon S3에 게시할 수도 있습니다.
- Internet Monitor는 알림을 설정할 수 있도록 Amazon EventBridge로 상태 이벤트를 전송합니다. AWS 네트워크로 인해 문제가 발생하는 경우 AWS에서 문제를 완화하기 위해 수행하는 단계가 포함된 AWS Health Dashboard 알림도 자동으로 수신됩니다.

Internet Monitor 사용 방법

Internet Monitor를 사용하려면 모니터를 생성하고 애플리케이션의 리소스(VPC, Network Load Balancer, CloudFront 배포 또는 WorkSpaces 디렉터리)를 모니터과 연결하여 Internet Monitor가 애플리케이션의 인터넷 트래픽이 어디에 있는지 알 수 있도록 합니다. 그런 다음 Internet Monitor는 도시-네트워크, 즉 클라이언트가 애플리케이션에 액세스하는 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 제공업체(ISP))과 관련된 AWS의 인터넷 측정값을 게시합니다. 자세한 내용은 Amazon CloudWatch Internet Monitor 작동 방식 단원을 참조하십시오. Internet Monitor로 작업을 시작하려면 콘솔을 사용하여 Amazon CloudWatch Internet Monitor 시작하기 섹션을 참조하세요.

내용

- Amazon CloudWatch Internet Monitor 사용 사례 예
- Internet Monitor 교차 계정 관찰성
- 콘솔을 사용하여 Amazon CloudWatch Internet Monitor 시작하기
- Amazon CloudWatch Internet Monitor에서 CLI를 사용하는 예
- Internet Monitor 대시보드로 모니터링 및 최적화
- CloudWatch 도구 및 Internet Monitor 쿼리 인터페이스로 데이터 탐색
- Amazon CloudWatch Internet Monitor를 사용하여 경보 생성
- Amazon EventBridge와 함께 Amazon CloudWatch Internet Monitor 사용
- CloudWatch 로그 및 지표 액세스 오류 문제 해결
- Amazon CloudWatch Internet Monitor를 통한 데이터 보호 및 데이터 프라이버시
- Amazon CloudWatch Internet Monitor에 대한 ID 및 액세스 관리
- Amazon CloudWatch Internet Monitor의 할당량

Amazon CloudWatch Internet Monitor에 대해 AWS 리전 지원

Amazon CloudWatch Internet Monitor에서 지원하는 AWS 리전가 이 섹션에 나와 있습니다. 옵트인 리전을 포함하여 Internet Monitor에서 지원하는 리전의 최신 목록은 Amazon Web Services 일반 참조의 Amazon CloudWatch Internet Monitor endpoints and quotas(Amazon CloudWatch Internet Monitor 엔드포인트 및 할당량)를 참조하세요.

Internet Monitor는 모니터에 대한 데이터를 모니터를 생성한 AWS 리전에만 저장하지만, 모니터에는 여러 리전의 리소스가 포함될 수 있습니다.

리전 이름(옵트인 지원)	지역
아프리카(케이프타운)	af-south-1
아시아 태평양(홍콩)	ap-east-1
아시아 태평양(하이데라바드)	ap-south-2
아시아 태평양(자카르타)	ap-southeast-3
아시아 태평양(멜버른)	ap-southeast-4
유럽(밀라노)	eu-south-1

리전 이름(옵트인 지원)	지역
유럽(스페인)	eu-south-2
유럽(취리히)	eu-central-2
중동(바레인)	me-south-1
중동(UAE)	me-central-1

리전 이름(기본 지원)	지역
미국 동부(오하이오)	us-east-2
미국 동부(버지니아 북부)	us-east-1
미국 서부(캘리포니아 북부)	us-west-1
미국 서부(오레곤)	us-west-2
아시아 태평양(뭄바이)	ap-south-1
아시아 태평양(오사카)	ap-northeast-3
아시아 태평양(서울)	ap-northeast-2
아시아 태평양(싱가포르)	ap-southeast-1
아시아 태평양(시드니)	ap-southeast-2
아시아 태평양(도쿄)	ap-northeast-1
캐나다(중부)	ca-central-1
유럽(프랑크푸르트)	eu-central-1
유럽(아일랜드)	eu-west-1
유럽(런던)	eu-west-2

리전 이름(기본 지원)	지역
유럽(파리)	eu-west-3
유럽(스톡홀름)	eu-north-1
남아메리카(상파울루)	sa-east-1

Amazon CloudWatch Internet Monitor 요금

Amazon CloudWatch Internet Monitor를 사용하면 선불 비용이나 장기 약정이 없습니다. Internet Monitor의 요금은 모니터되는 리소스당 요금과 도시-네트워크당 요금의 두 가지 구성 요소로 이루어져 있습니다. 도시-네트워크는 클라이언트가 애플리케이션 리소스에 액세스하는 위치 및 클라이언트가 리소스에 액세스하는 네트워크(ASN. 예: 인터넷 서비스 제공업체(ISP))를 의미합니다. 생성한 로그와 추가 지표, 대시보드, 경보 또는 인사이트에 대해서도 표준 CloudWatch 요금이 부과됩니다.

모니터를 생성할 때 모니터링할 트래픽의 백분율을 선택합니다. 요금을 관리하기 위해 모니터링할 최대 도시-네트워크 수에 대한 제한을 설정할 수도 있습니다. 모니터를 편집하여 언제든지 모니터링할 트래픽의 백분율 또는 최대 도시-네트워크 제한을 업데이트할 수 있습니다. (계정당 모든 모니터에 걸쳐) 처음 100개의 도시-네트워크가 포함됩니다. 그 후에는 모니터하는 실제 추가 도시-네트워크 수에 대해서만 최대 수까지 비용을 지불하면 됩니다.

처음 100개의 도시-네트워크(계정당 모든 모니터에 걸쳐)에 대해서는 무료로 모니터하는 실제 추가 도시-네트워크 수에 대해서만 최대 수까지 비용을 지불합니다. 도시-네트워크 100개 비용에 해당하는 정액이 월 청구서에서 차감됩니다.

예를 들어 대규모 글로벌 기업에서 인터넷 트래픽을 100% 모니터하고, 하나의 모니터에 하나의 리소스로 도시-네트워크 최대 50,000개를 설정할 수 있습니다. 트래픽이 50,000개의 도시-네트워크에 도달했다고 가정하면 해당 부분의 가격은 약 USD 2,700달러/월입니다. 다른 회사의 경우, 더 적은 지역에서 하나의 모니터에 하나의 리소스와 200개의 도시 네트워크가 있는 경우 이 부분의 요금은 월 13 USD 정도입니다. 자세한 내용은 도시-네트워크 최대 한도 선택 단원을 참조하십시오.

Pricing calculator를 사용하여 다양한 옵션을 시험해 볼 수 있습니다. 요금 옵션을 살펴보려면 CloudWatch Pricing calculator 페이지에서 Internet Monitor까지 아래로 스크롤합니다.

Internet Monitor 및 CloudWatch 요금에 대한 자세한 정보는 Amazon CloudWatch 요금 페이지를 참조하세요.

Amazon CloudWatch Internet Monitor의 구성 요소 및 정의

Amazon CloudWatch Internet Monitor는 다음을 사용하거나 참조합니다.

모니터링

모니터에는 인터넷 성능 및 가용성 측정값을 확인하고 상태 이벤트 알림을 받으려는 단일 애플리케이션에 대한 리소스가 포함됩니다. 애플리케이션에 대한 모니터를 만들 때 애플리케이션에 대한 리소스를 추가하여 Internet Monitor가 모니터링할 도시(위치)를 정의합니다. Internet Monitor는 사용자가 추가하는 애플리케이션 리소스의 트래픽 패턴을 사용하여 애플리케이션과 통신하는 위치 및 ASN(일반적으로 인터넷 서비스 제공업체(ISP))에 대한 인터넷 성능 및 가용성 측정값만 게시할 수 있습니다. 즉, 추가하는 리소스에 따라 Internet Monitor가 모니터하고 측정값을 게시할 도시-네트워크의 범위가 만들어집니다.

모니터에 추가된 리소스('모니터링되는 리소스')

모니터에 추가하는 리소스는 Internet Monitor의 '모니터링되는 리소스'입니다. 즉, 다음과 같습니다.

- 리전에 추가하는 각 VPC는 모니터되는 리소스입니다. VPC를 추가하면 Internet Monitor는 VPC의 모든 인터넷 연결 애플리케이션(예: Amazon EC2 인스턴스, Network Load Balancer 뒤 또는 AWS Fargate 컨테이너에 호스팅되는 애플리케이션)의 트래픽을 모니터링합니다.
- 리전에 추가하는 각 Network Load Balancer는 모니터되는 리소스입니다.
- 리전에 추가하는 각 WorkSpaces 디렉터리는 모니터되는 리소스입니다.
- 추가하는 각 CloudFront 배포는 모니터되는 리소스입니다.

Autonomous System Number(ASN)

Internet Monitor에서 ASN은 일반적으로 Verizon 또는 Comcast와 같은 인터넷 서비스 제공업체(ISP)를 의미합니다. ASN은 클라이언트가 인터넷 애플리케이션에 액세스하는 데 사용하는 네트워크 공급자입니다. Autonomous System(AS)은 하나의 조직에서 모두 관리, 제어 및 감독하는 네트워크 또는 네트워크 모음에 속하는 인터넷 라우팅 가능 인터넷 프로토콜(IP) 접두사의 집합입니다.

도시-네트워크(위치 및 ASN)

도시-네트워크는 클라이언트가 애플리케이션 리소스에 액세스하는 위치(예: 도시)와 클라이언트가 리소스에 액세스하는 ASN(일반적으로 인터넷 서비스 제공업체(ISP))을 의미합니다. 요금 관리를 위해 각 모니터에 대해 Internet Monitor가 모니터링할 수 있는 최대 도시 네트워크 수에 대한 제한을 설정할 수 있습니다. 모니터하는 도시-네트워크의 실제 개수에 대해서만 최대 개수까지만 비용을 지불합니다. 자세한 내용은 도시-네트워크 최대 한도 선택을 참조하세요.

인터넷 측정값

Internet Monitor는 계정의 상위 500개 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 공급업체(ISP))에 대해 5분마다 인터넷 측정값을 CloudWatch 로그의 로그 파일에 게시합니다. 이러한 측정은 애플리케이션의 도시-네트워크에 대한 성능 점수, 가용성 점수, 전송된 바이트(바이트 인 및 바이트 아웃), 왕복 시간을 정량화합니다. 이는 VPC, Network Load Balancer, CloudFront 배포 또는 WorkSpaces 디렉터리와 관련된 도시-네트워크에 대한 측정값입니다. 원하는 경우, 모니터링되는 모든 도시-네트워크(최대 500,000개의 도시-네트워크 서비스 제한)에 대한 인터넷 측정값과 이벤트를 Amazon S3 버킷에 게시하도록 선택할 수 있습니다.

지표

Internet Monitor는 애플리케이션에 대한 글로벌 트래픽과 각 AWS 리전에 대한 글로벌 트래픽에 대한 CloudWatch 지표에 대한 집계 지표를 생성합니다. 자세한 내용은 Amazon CloudWatch Internet Monitor와 함께 CloudWatch 지표 사용 단원을 참조하십시오.

상태 이벤트

Internet Monitor는 상태 이벤트를 생성하여 애플리케이션에 영향을 미치는 특정 문제를 알려줍니다. Internet Monitor는 전 세계의 네트워크 지연 시간 증가와 같은 인터넷 문제를 탐지합니다. 그런 다음 AWS 글로벌 인프라 공간 전체의 기록 인터넷 측정값을 사용하여 현재 문제가 애플리케이션에 미치는 영향을 계산하고 상태 이벤트를 생성합니다. 기본적으로 Internet Monitor는 전체 영향 임계값과 로컬 영향 임계값을 모두 기준으로 상태 이벤트를 생성합니다. 임계값 구성에 대한 자세한 내용은 상태 이벤트 임계값 변경을 참조하세요.

각 상태 이벤트에는 영향을 받는 도시-네트워크에 대한 정보가 포함되어 있습니다. CloudWatch 콘솔에서 또는 AWS SDK 또는 AWS CLI를 사용하여 Internet Monitor API 동작을 통해 상태 이벤트를 볼 수 있습니다. Internet Monitor는 상태 이벤트에 대한 Amazon EventBridge 알림도 전송합니다. 자세한 내용은 Internet Monitor가 상태 이벤트를 생성하고 해결하는 경우를 참조하세요.

인터넷 이벤트

Internet Monitor는 모든 AWS 고객에게 제공되는 인터넷 웨더 맵에 인터넷 이벤트라고 하는 최근의 글로벌 상태 이벤트에 관한 정보를 표시합니다. 인터넷 웨더 맵을 보기 위해 Internet Monitor에서 모니터를 생성할 필요는 없습니다. 상태 이벤트와 달리 인터넷 이벤트는 개별 고객이나 애플리케이션 트래픽에만 국한되지 않습니다. 자세한 내용은 Amazon CloudWatch Internet Monitor의 글로벌 인터넷 웨더 맵 단원을 참조하십시오.

임계값

Internet Monitor는 전체 임계값과 로컬 임계값을 모두 기반으로 상태 이벤트를 생성합니다. 기본 임계값을 변경하고 로컬 임계값 끄기와 같은 다른 옵션을 구성할 수 있습니다. 임계값 구성에 대한 자세한 내용은 상태 이벤트 임계값 변경을 참조하세요.

성능 및 가용성 점수

AWS가 수집하는 데이터를 분석하여 Internet Monitor가 계산하는 예상 기준선과 비교하여 애플리케이션의 성능 및 가용성이 저하된 시점을 탐지할 수 있습니다. 이러한 성능 저하를 쉽게 확인할 수 있도록 Internet Monitor는 정보를 점수로 보고합니다. 성능 점수는 성능 저하가 나타나지 않는 트래픽의 예상 백분율을 나타냅니다. 마찬가지로 가용성 점수는 가용성 저하가 나타나지 않는 트래픽의 예상 백분율을 나타냅니다. 자세한 내용은 AWS가 성능 및 가용성 점수를 계산하는 방법을 참조하세요.

전송된 바이트 및 모니터링된 바이트 전송량

전송된 바이트는 AWS의 애플리케이션과 클라이언트가 애플리케이션에 액세스하는 도시-네트워크(즉, 위치 및 ASN, 일반적으로 인터넷 서비스 공급업체) 간의 수신 및 송신 트래픽의 총 바이트 수입니다. 전송된 모니터 바이트는 유사한 지표지만 모니터되는 트래픽에 대한 바이트만 포함합니다.

왕복 시간

왕복 시간(RTT)은 클라이언트 사용자의 요청이 사용자에게 응답을 반환하는 데 걸리는 시간입니다. RTT가 클라이언트 위치(도시 또는 기타 지역)에 따라 집계되는 경우, 각 클라이언트 위치에서 애플리케이션 트래픽이 얼마나 많이 발생하는지에 따라 가중치가 부여됩니다.

Amazon CloudWatch Internet Monitor의 글로벌 인터넷 웨더 맵

Amazon CloudWatch Internet Monitor는 모든 AWS 고객에게 제공되는 글로벌 인터넷 웨더 맵을 표시합니다. 맵을 보려면 Amazon CloudWatch 콘솔에서 Internet Monitor로 이동합니다.

맵에는 성능 또는 가용성에 문제가 있는 특정 도시 및 네트워크(ASN, 일반적으로 인터넷 서비스 제공업체)를 비롯하여 AWS 고객에게 영향을 미치는 전 세계의 인터넷 이벤트('중단')가 강조 표시됩니다. 인터넷 웨더 맵에는 지난 24시간 동안의 인터넷 이벤트가 포함됩니다.

인터넷 웨더 맵을 보기 위해 Internet Monitor에서 모니터를 생성할 필요는 없습니다. Internet Monitor의 상태 이벤트와 달리 인터넷 이벤트는 개별 고객이나 애플리케이션 트래픽에만 국한되지 않습니다.

인터넷 웨더 맵에서 인터넷 이벤트를 선택하여 해당 이벤트에 관한 세부 정보를 확인할 수 있습니다. 인터넷 이벤트의 경우 시작 시간, 종료 시간(이벤트가 종료된 경우), 현재 상태(활성 또는 해결됨), 중단

문제 유형(가용성 또는 성능)을 확인할 수 있습니다. 인터넷 웨더 맵이 생성되는 방법과 포함된 내용에 관한 자세한 정보는 글로벌 인터넷 웨더 맵 FAQ를 참조하세요.

애플리케이션 트래픽 및 클라이언트 위치와 관련된 세부 정보를 보고 사용하기 위해 Internet Monitor에서 애플리케이션에 맞게 모니터를 쉽게 설정할 수 있습니다. 그러면 현재와 과거의 성능 및 가용성 패턴과 이벤트를 확인할 수 있을 뿐만 아니라 애플리케이션 설치 공간 및 고객에 맞게 조정된 상태 이벤트 알림을 받을 수 있습니다. 인터넷 웨더 맵을 통해 전체적인 정보를 볼 수 있고, 특정 모니터는 정보를 필터링하여 애플리케이션과 관련된 측정 및 세부 정보만 표시합니다. 또한 모니터를 사용하여 과거 지표를 탐색하고 애플리케이션에 대한 클라이언트 경험을 개선하기 위한 권장 사항을 얻을 수 있습니다. 자세한 내용은 콘솔을 사용하여 Amazon CloudWatch Internet Monitor 시작하기을 참조하십시오.

Amazon CloudWatch Internet Monitor 작동 방식

이 섹션에서는 Amazon CloudWatch Internet Monitor의 작동 방식에 대한 정보를 제공합니다. 여기에는 AWS가 인터넷에서 연결 문제를 탐지하는 데 사용하는 데이터를 수집하는 방법과 성능 및 가용성 점수가 계산되는 방법에 대한 설명이 포함되어 있습니다.

목차

Internet Monitor에서 애플리케이션 트래픽 크기에만 초점을 맞추는 방법

Internet Monitor는 다른 도구처럼 전 세계 모든 리전의 웹 사이트를 광범위하게 모니터링하는 대신 AWS 리소스 사용자가 액세스하는 인터넷의 하위 세트만 집중적으로 모니터링합니다. 또한 대기업과 중소기업 모두에 적합한 비용 효율적인 솔루션입니다.

Internet Monitor는 AWS가 내부적으로 활용하는 것과 동일한 강력한 프로브 및 문제 탐지 알고리즘을 사용하며, Internet Monitor에서 상태 이벤트를 생성하여 애플리케이션에 영향을 주는 연결 문제를 알려줍니다. 그런 다음 Internet Monitor는 애플리케이션 리소스를 기반으로 활성 뷰어에서 생성한 트래픽 프로파일을 오버레이하여 결과 성능 및 가용성 맵에 대한 액세스를 제공합니다.

이 정보를 사용하여 Internet Monitor는 관련 이벤트(즉, 활성 시청자가 있는 위치의 이벤트)와 해당 이벤트가 전체 시청자 수에 미치는 영향만 표시합니다. 따라서 이벤트가 미치는 영향은 전 세계 총 트래픽을 기준으로 백분율로 표시됩니다.

Internet Monitor는 각 모니터에 트래픽을 전송하는 상위 500개 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 공급업체(ISP))에 대한 인터넷 측정값을 5분마다 CloudWatch 로그에 게시합니다. 원하는 경우, 모니터링되는 모든 도시-네트워크(최대 500,000개의 도시-네트워크 서비스 제한)에 대한 인터넷 측정값을 Amazon S3 버킷에 게시하도록 선택할 수 있습니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에서 Amazon S3에 인터넷 측정값 게시 단원을 참조하십시오.

Internet Monitor는 다음과 같은 이점이 있습니다.

- Internet Monitor를 사용해도 AWS에서 호스팅되는 애플리케이션에 추가 로드나 비용이 발생하지 않습니다.
- 클라이언트 측 리소스나 애플리케이션에 성능 측정 코드를 포함할 필요가 없습니다.
- 애플리케이션이 연결된 인터넷에서 '라스트 마일' 정보를 비롯하여 성능 및 가용성에 대한 가시성을 얻을 수 있습니다.

Internet Monitor는 AWS 리소스를 기반으로 측정값을 생성하기 때문에 Internet Monitor는 애플리케이션 트래픽에 특정한 이벤트만 생성합니다. 일반적으로 글로벌 인터넷 문제는 보고되지 않습니다. 또한 서비스 위치가 AWS 리전인 경우 방출된 측정값과 이벤트는 리전 수준의 연결을 나타내도록 설계되었으며 최종 사용자 위치와 가용 영역 간의 연결을 정확하게 나타내지 않습니다.

AWS에서 연결 문제를 측정하고 측정값을 계산하는 방법

Amazon CloudWatch Internet Monitor는 Autonomous System Number(ASN)(일반적으로 인터넷 서비스 제공업체(ISP))를 통해 서로 다른 AWS 리전 및 서로 다른 클라이언트 위치에 대한 Amazon CloudFront 접속 지점(POP) 간 인터넷 연결 데이터를 사용합니다. AWS 운영자는 매일 이 연결 데이터를 내부적으로 사용하여 전 세계 인터넷에서 연결 문제를 사전에 탐지합니다.

모든 AWS 리전마다 인터넷의 어느 부분이 지역과 통신하는지 파악하고 다음을 수행합니다.

- 인터넷의 이러한 부분은 30일의 롤링 기간 동안 적극적으로 모니터링됩니다.
- 인바운드 및 아웃바운드 프로빙을 모두 포함하여 네트워크 및 상위 수준 프로토콜 프로브를 모두 사용합니다.

AWS에는 모든 AWS 리전 및 CloudFront 서비스에서 전체 인터넷에 이르는 90번째 백분위수의 지연 시간(성능)과 연결성(가용성)을 측정하는 능동 및 수동 프로브가 있습니다. 서비스와 고객 위치 간의 비정상적인 연결 패턴이 모니터링된 다음, 고객에게 경고로 보고됩니다.

가용성 및 RTT 계산

왕복 시간(RTT)은 사용자의 요청이 사용자에게 응답을 반환하는 데 걸리는 시간입니다. 최종 사용자 위치 전체에서 왕복 시간이 집계되면 값은 각 최종 사용자 위치에서 유도하는 트래픽 양에 따라 가중치가 부여됩니다.

예를 들어, 최종 사용자 위치가 두 개이며 하나는 5ms RTT로 트래픽의 90%를 처리하고 다른 하나는 10ms RTT로 트래픽의 10%를 처리하는 경우 결과는 5.5ms(5ms * 0.9 + 10ms * 0.1)의 집계 RTT입니다.

단, 마지막 마일 지연 시간 측정과 관련된 리소스에는 차이가 있습니다. Internet Monitor 지연 시간 측정의 경우 VPC, Network Load Balancer, WorkSpaces 디렉터리에는 마지막 마일 지연 시간이 포함되지 않습니다.

성능 및 가용성 점수 계산

AWS는 AWS 서비스와 다양한 도시-네트워크(위치 및 ASN) 간의 인터넷 성능 및 가용성에 대한 상당한 과거 데이터를 보유하고 있습니다. Internet Monitor는 데이터에 통계 분석을 적용하여 애플리케이션의 성능 및 가용성이 계산한 예상 기준선과 비교하여 언제 저하되었는지 탐지할 수 있습니다. 이러한 성능 저하를 쉽게 확인할 수 있도록 해당 정보는 성능 점수와 가용성 점수라는 상태 점수의 형태로 사용자에게 보고됩니다.

상태 점수는 서로 다른 세부 수준으로 계산됩니다. 가장 세분화된 수준에서는 도시 또는 대도시 리전과 같은 지리적 리전과 ASN(도시-네트워크)에 대한 상태 점수를 계산합니다. 또한 모니터의 애플리케이션에 대한 개별 건강 점수가 전체 상태 점수 숫자로 롤업됩니다. 특정 지역 또는 서비스 제공업체에 대한 필터링 없이 성능 또는 가용성 점수를 볼 경우 Internet Monitor는 전체 상태 점수를 제공합니다.

전체 상태 점수는 지정된 기간 동안 전체 애플리케이션에 적용됩니다. 애플리케이션의 도시-네트워크 쌍에 대한 애플리케이션의 성능 또는 가용성 점수가 성능 또는 가용성에 대한 해당 상태 이벤트 임계값에 도달하거나 그 이하로 떨어지면 Internet Monitor에서 상태 이벤트를 트리거합니다. 기본적으로 임계값은 전체 성능과 가용성 모두에 대해 95%입니다. 또한 Internet Monitor는 로컬 임계값을 기반으로 (이 옵션이 활성화된 경우 기본적으로 구성한 값 그대로) 상태 이벤트를 생성합니다. 상태 이벤트 임계값 구성에 대해 자세히 알아보려면 상태 이벤트 임계값 변경을 참조하세요.

모니터 및 로그 파일에서 정보를 탐색하여 문제를 조사하고 자세한 내용을 알아볼 때 특정 도시(위치), 네트워크(ASN 또는 인터넷 서비스 제공업체) 또는 두 가지 모두를 기준으로 필터링할 수 있습니다. 따라서 필터를 사용하여 선택한 필터에 따라 다른 도시, ASN 또는 도시-네트워크 쌍에 대한 상태 점수를 확인할 수 있습니다.

- 가용성 점수는 가용성 저하가 나타나지 않는 트래픽의 예상 백분율을 나타냅니다. Internet Monitor는 표시된 총 트래픽 및 가용성 지표 측정값에서 저하를 경험하는 트래픽의 백분율을 추정합니다. 예를 들어 최종 사용자와 서비스 위치 쌍의 가용성 점수 99%는 해당 쌍에 대해 가용성 저하를 경험하는 트래픽의 1%와 같습니다.
- 성능 점수는 성능 저하가 나타나지 않는 트래픽의 백분율을 나타냅니다. 예를 들어 최종 사용자와 서비스 위치 쌍의 가용성 점수 99%는 해당 쌍에 대해 성능 저하를 경험하는 트래픽의 1%와 같습니다.

TTFB 및 RTT(지연 시간) 계산

첫 바이트까지 시간(TTFB)은 클라이언트가 요청을 할 때와 서버로부터 첫 바이트의 정보를 수신할 때 사이의 시간을 의미합니다. AWS TTFB에 대한 계산은 Amazon EC2 또는 Amazon CloudFront에서 Internet Monitor 측정 노드(노드의 마지막 마일 포함)까지 경과한 시간을 측정합니다. 즉, Internet Monitor는 EC2에 대한 TTFB의 경우 사용자에서 Amazon EC2 리전까지의 시간을 측정하고, CloudFront에 대한 TTFB의 경우 사용자에서 CloudFront까지의 시간을 측정합니다.

왕복 시간(RTT)의 경우 Internet Monitor에는 퍼블릭 IP 주소로 매핑된 도시-네트워크(즉, 클라이언트 위치와 ASN, 일반적으로 인터넷 서비스 공급업체)부터 AWS 리전까지의 시간이 포함됩니다. 즉, Internet Monitor는 게이트웨이 또는 VPN 뒤에서 인터넷에 액세스하는 사용자에 대한 마지막 마일 가시성을 제공하지 않습니다.

단, 마지막 마일 지연 시간 측정과 관련된 리소스에는 차이가 있습니다. Internet Monitor 지연 시간 측정의 경우 VPC, Network Load Balancer, WorkSpaces 디렉터리에는 마지막 마일 지연 시간이 포함되지 않습니다.

Internet Monitor는 CloudWatch 대시보드의 트래픽 인사이트 탭에 있는 트래픽 최적화 제안 섹션에 평균 TTFB 정보를 제공하여 성능을 향상시킬 수 있는 애플리케이션의 다양한 설정 옵션을 평가할 수 있도록 도와줍니다.

리전 및 가용 영역 측정 및 집계

Internet Monitor는 측정값을 집계하고 리전 수준에서 해당 영향을 공유하지만, 영향을 계산하는 수준은 가용 영역(AZ)입니다. 즉, 이벤트가 발생했을 때 하나의 AZ만 영향을 받고 대부분의 트래픽이 해당 AZ를 통과하는 경우 트래픽에 미치는 영향을 확인할 수 있습니다. 하지만 동일한 이벤트에서 애플리케이션 트래픽이 영향을 받는 AZ를 통과하지 않으면 영향을 확인할 수 없습니다.

이는 WorkSpaces 디렉터리가 아닌 리소스에만 적용됩니다. WorkSpaces 디렉터리는 리전 수준에서만 측정됩니다.

Internet Monitor의 지리적 위치 정확도

위치 정보의 경우 Internet Monitor는 MaxMind에서 제공하는 IP-지리적 위치 데이터를 사용합니다. Internet Monitor 측정의 위치 정보 정확도는 MaxMind 데이터의 정확도에 따라 달라집니다.

미국 이외의 지역에서는 `Metro` 수준의 측정이 정확하지 않을 수 있습니다.

Internet Monitor가 상태 이벤트를 생성하고 해결하는 경우

Internet Monitor는 설정된 현재 임계값에 따라 모니터하는 애플리케이션 트래픽에 대한 상태 이벤트를 생성하고 종료합니다. Internet Monitor에는 기본 임계값 구성이 있으며 임계값에 대한 고유한 구성을 설정할 수도 있습니다. Internet Monitor는 연결 문제가 애플리케이션에 미치는 전반적인 영향과 애플리케이션에 클라이언트가 있는 로컬 영역에 미치는 영향을 파악하고 임계값을 초과하면 상태 이벤트를 생성합니다.

Internet Monitor는 AWS를 통해 서비스에 제공되는 네트워크 트래픽의 인터넷 성능 및 가용성에 대한 기록 데이터를 기반으로 클라이언트 위치에 대한 연결 문제의 영향을 계산합니다. 클라이언트가 애플리케이션을 사용하는 ASN 및 서비스의 지리적 위치(영향을 받는 도시-네트워크 쌍)를 기반으로 애플리케이션과 관련된 정보를 적용합니다. 위치는 모니터에 추가하는 리소스에 따라 결정됩니다. 그런 다음 Internet Monitor는 통계 분석을 사용하여 성능 및 가용성이 저하되어 애플리케이션의 클라이언트 환경에 영향을 미치는 시점을 탐지합니다.

Internet Monitor가 계산하는 성능 및 가용성 점수는 감소하지 않은 트래픽의 백분율로 표시됩니다. 영향은 이와 반대입니다. 즉, 고객의 최종 사용자에게 얼마나 문제가 되는지를 나타내는 수치입니다. 따라서 예를 들어 글로벌 가용성이 93% 저하하는 경우 해당 영향은 7%입니다.

애플리케이션의 도시-네트워크 쌍에 대한 전 세계 성능 또는 가용성 점수가 성능 또는 가용성에 대한 해당 상태 이벤트 임계값에 도달하거나 그 이하로 떨어지면 Internet Monitor에서 상태 이벤트를 생성합니다. 기본적으로 임계값은 성능과 가용성 모두에 대해 95%입니다. 임계값을 충족하거나 그 이하로 떨어지는 값은 누적되므로 여러 개의 작은 이벤트가 합쳐져 임계값 백분율을 충족하거나 단일 이벤트가 임계값 수준을 충족하거나 그 이하로 떨어질 수 있습니다.

이벤트를 트리거한 성능 또는 가용성 점수가 전체 영향에 대한 해당 상태 이벤트 임계값 백분율 이하인 한, 상태 이벤트는 활성 상태로 유지됩니다. 이벤트를 트리거한 점수 또는 합산 점수가 임계값을 초과하면 Internet Monitor가 상태 이벤트를 해결합니다.

또한 Internet Monitor는 로컬 임계값과 문제가 영향을 미치는 전체 트래픽의 백분율을 기반으로 상태 이벤트를 생성합니다. 로컬 임계값에 대한 옵션을 구성하거나 로컬 임계값을 모두 해제할 수 있습니다.

상태 이벤트 임계값 구성에 대해 자세히 알아보려면 상태 이벤트 임계값 변경을 참조하세요.

상태 이벤트 보고 타이밍

Internet Monitor는 애그리게이터를 사용하여 인터넷 문제에 대한 모든 신호를 수집하고 몇 분 안에 모니터에서 상태 이벤트를 생성합니다.

가능한 경우 Internet Monitor는 상태 이벤트의 원인을 분석하여 AWS 또는 ASN으로 인한 것인지 확인합니다. 상태 이벤트 분석은 이벤트가 해결된 후에도 계속됩니다. Internet Monitor는 최대 1시간 동안 새로운 정보로 이벤트를 업데이트할 수 있습니다.

Internet Monitor는 IPv4 및 IPv6 트래픽에서 작동하는 방식

Internet Monitor는 모든 IP 패밀리를 통해 해당 네트워크에 트래픽을 전송하는 경우 IPv4만을 통해 네트워크의 상태를 측정하고 상태 이벤트와 가용성 및 성능 지표를 표시합니다(IPv4 또는 IPv6). 이중 스택 CloudFront 배포와 같은 이중 스택 리소스에서 트래픽을 서비스하는 경우 Internet Monitor는 IPv4 트래픽이 리소스에 대해 IPv6 트래픽과 동일한 문제가 있는 경우에만 상태 이벤트를 발생시키고 성능 점수 또는 가용성 점수의 하락을 표시합니다.

전체 바이트 수신 및 바이트 발신에 대한 Internet Monitor 지표는 모든 인터넷 트래픽(IPv4 및 IPv6)을 정확하게 반영합니다.

Internet Monitor에서 포함할 도시 네트워크의 하위 집합을 선택하는 방법

모니터로 모니터링하는 도시 네트워크 수의 최대 제한을 설정하거나 모니터링할 트래픽 비율을 선택하면 Internet Monitor에서 최근 가장 많은 트래픽 볼륨을 기준으로 포함(모니터링)할 도시 네트워크를 선택합니다.

예를 들어 도시 네트워크 최대 제한을 100개로 설정하면 Internet Monitor는 최근 1시간 동안 애플리케이션 트래픽을 기반으로 최대 100개의 도시 네트워크를 모니터링합니다. 특히, Internet Monitor에서는 최근 1시간 이전의 1시간 동안 트래픽이 가장 많은 상위 100개 도시 네트워크를 모니터링합니다.

이를 설명하기 위해 현재 시간이 오후 2시 30분이라고 가정합니다. 이 시나리오에서 모니터에 표시되는 트래픽은 오후 1시에서 2시 사이에 캡처되었으며, Internet Monitor에서 상위 100개 도시 네트워크를 결정하는 데 사용하는 트래픽 볼륨은 오후 12시에서 1시 사이에 캡처되었습니다.

글로벌 인터넷 웨더 맵의 생성 방식(자주 묻는 질문)

Amazon CloudWatch Internet Monitor 인터넷 웨더 맵은 Internet Monitor 콘솔에서 인증된 모든 AWS 고객에게 제공됩니다. 이 섹션에는 인터넷 웨더 맵이 생성되는 방법과 사용 방법에 관한 세부 정보가 포함되어 있습니다.

Internet Monitor 웨더 맵이란 무엇인가요?

인터넷 웨더 맵은 전 세계의 인터넷 문제를 시각적으로 보여줍니다. 영향을 받는 클라이언트 위치, 즉 도시와 ASN(일반적으로 인터넷 서비스 제공업체)을 강조 표시합니다. 이 맵에는 최근 전 세계 주요 고객 위치 및 AWS 서비스에 대해 고객의 인터넷 경험에 영향을 미친 가용성 및 성능 문제가 복합적으로 표시되어 있습니다.

맵 데이터의 출처는 어디인가요?

데이터는 능동형 및 수동형 인터넷 탐색의 조합을 기반으로 합니다. Internet Monitor가 데이터를 측정하는 방법을 자세히 알아보려면 AWS의 연결 문제 측정 방법 섹션을 읽어보세요.

맵은 얼마나 자주 업데이트되나요?

인터넷 웨더 맵은 15분마다 업데이트됩니다.

어떤 네트워크에서 중단이 추적되나요?

AWS는 고객이 AWS로의 인터넷 연결에 사용하는 중요한 IP 접두사를 나타내는 전 세계 네트워크를 추적합니다. AWS 네트워크로 송수신되는 트래픽 양이 가장 많은 클라이언트 위치를 대상으로 중단 범위를 파악합니다.

인터넷 이벤트가 맵에 포함되는지 여부는 어떻게 결정되나요?

다음은 인터넷 이벤트가 인터넷 웨더 맵에 포함되는지 여부를 결정하는 데 사용되는 몇 가지 대략적인 기준입니다.

- AWS는 가용성 또는 성능 이벤트가 있는지 감지합니다.
- 예를 들어 이벤트 길이가 5분 미만인 경우 이 이벤트는 무시됩니다.
- 그리고 상위 토커로 분류된 클라이언트 위치에서 이벤트가 발생하면 중단으로 간주됩니다.

인터넷 웨더 맵에는 어떤 임계값이 사용되나요?

인터넷 웨더 맵의 중단 결정 임계값은 고정되어 있지 않습니다. Internet Monitor는 예상하는 값과의 편차를 감지하여 이벤트를 구성하는 요소를 결정합니다. 서비스로 생성한 모니터에 대해 Internet Monitor에서 상태 이벤트를 생성하는 시기를 결정하는 방식을 검토하여 그 방법을 자세히 알아볼 수 있습니다. 모니터를 생성하면 Internet Monitor는 사용자의 애플리케이션 트래픽과 관련된 인터넷 트래픽 상태 측정치를 생성합니다. 또한 Internet Monitor는 애플리케이션의 인터넷 트래픽에 영향을 미치는 문제에 대한 상태 이벤트 경보를 보냅니다.

이 데이터로 무엇을 할 수 있나요?

인터넷 웨더 맵은 지난 24시간 동안 전 세계에서 발생한 주요 인터넷 이벤트를 간략하게 요약합니다. 이를 통해 자신의 인터넷 트래픽을 Internet Monitor에 온보딩할 필요 없이 다양한 인터넷

모니터링 경험을 얻을 수 있습니다. AWS의 인터넷 모니터링 기능을 최대한 활용하고 AWS에서 호스팅되는 애플리케이션 및 서비스에 맞춰 개인화하기 위해 Internet Monitor에서 모니터를 생성할 수 있습니다.

모니터를 생성하면 Internet Monitor를 활성화하여 애플리케이션 클라이언트에 영향을 미치는 특정 인터넷 경로를 식별하고, 클라이언트 경험을 개선하는 데 도움이 되는 기능에 액세스할 수 있습니다. 또한 특히 애플리케이션 트래픽과 클라이언트에 영향을 미치는 새로운 인터넷 문제에 대해 미리 알림을 받을 수 있습니다.

이벤트에 관한 세부 정보는 어떻게 얻을 수 있나요?

맵에서 중단을 클릭하면 이벤트 시작 및 종료 시간, 영향을 받는 도시 및 ASN, 발생한 문제 유형(성능 문제 또는 가용성 문제) 등의 세부 정보를 볼 수 있습니다.

이벤트에 관한 세부 정보를 얻고 애플리케이션 트래픽에 대한 사용자 지정 측정치를 얻으려면 Internet Monitor에서 모니터를 생성합니다.

Amazon CloudWatch Internet Monitor 사용 사례 예

이 섹션에서는 몇 가지 구체적인 예와 함께 자세한 내용이 담긴 블로그 게시물 링크를 설명합니다. 이 예는 Amazon CloudWatch Internet Monitor의 기능을 사용하여 애플리케이션을 모니터하고 사용자 환경을 개선하는 방법을 보여줍니다.

알림 설정 및 취해야 할 조치 결정

Internet Monitor를 사용하면 시간 경과에 따른 평균 인터넷 성능 지표와 도시-네트워크별(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 제공업체) 상태 이벤트에 대한 인사이트를 얻을 수 있습니다. Internet Monitor를 사용하면 Amazon Virtual Private Cloud(VPC), Network Load Balancer, Amazon WorkSpaces 또는 Amazon CloudFront에서 호스트되는 애플리케이션의 최종 사용자 경험에 영향을 미치는 이벤트를 식별할 수 있습니다.

모니터를 만든 후에는 Internet Monitor 상태 이벤트에 대한 알림을 받는 방법에 대한 몇 가지 옵션이 있습니다. 여기에는 상태 이벤트를 필터링하기 위해 이벤트 지표 또는 Amazon EventBridge 규칙을 사용하는 CloudWatch 경보를 기반으로 하는 알림이 포함됩니다. 예를 들어 CloudWatch 로그 그룹에 대한 AWS SMS 알림 또는 업데이트 등 경보에 따라 알림 또는 작업에 대한 다양한 옵션을 선택할 수 있습니다.

자세한 지침이 포함된 예를 보려면 블로그 게시물(Introducing Amazon CloudWatch Internet Monitor)을 참조하세요.

지연 시간 문제를 파악하고 TTFB를 개선하여 멀티플레이어 게임플레이 경험 개선

Internet Monitor를 사용하면 전 세계 클라우드 게임 앱에서 게임 플레이어가 지연 문제를 겪고 있는 위치를 빠르게 파악하고 성능 개선에 대한 인사이트를 얻을 수 있습니다. 현재 가장 많은 플레이어가 첫 바이트까지 걸리는 시간(TTFB)이 가장 느린 위치를 파악하면 지연 시간을 개선하여 대규모 플레이어 기반을 만족시킬 수 있는 방법을 알 수 있습니다.

이제 게임에 다음 EC2 서버를 배포할 준비가 되었으면 Internet Monitor가 지연 시간이 길고 플레이어 수가 많은 지역에서 TTFB를 낮출 수 있다고 제안하는 AWS 리전을 선택하세요.

이 사용 사례에 대한 Internet Monitor 설정 및 사용에 대한 자세한 내용은 블로그 게시물(Using Amazon CloudWatch Internet Monitor for a Better Gaming Experience)을 참조하세요.

Internet Monitor 교차 계정 관찰성

Internet Monitor의 교차 계정 관찰성을 사용하면 단일 AWS 리전 내의 여러 AWS 계정에 걸쳐 있는 애플리케이션을 모니터링할 수 있습니다.

Amazon CloudWatch Observability Access Manager를 사용하여 하나 이상의 AWS 계정을 모니터링 계정으로 설정할 수 있습니다. 모니터링 계정에 싱크를 생성하여 소스 계정의 데이터를 볼 수 있는 기능을 모니터링 계정에 제공합니다. 싱크는 모니터링 계정의 연결 지점을 나타내는 리소스입니다. Internet Monitor의 경우 리소스 연결 지점은 모니터입니다. 싱크를 사용하여 소스 계정에서 모니터링 계정으로의 링크를 만듭니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

필수 리소스

CloudWatch Application Insights 교차 계정 관찰성의 적절한 작동을 위해 CloudWatch Observability Access Manager를 통해 다음 원격 측정 유형이 공유되는지 확인하세요.

- Internet Monitor의 모니터
- Amazon CloudWatch의 지표
- Amazon CloudWatch Logs의 로그 그룹

콘솔을 사용하여 Amazon CloudWatch Internet Monitor 시작하기

Amazon CloudWatch Internet Monitor를 시작하려면 애플리케이션에서 사용하는 AWS 리소스를 추가하고 몇 가지 구성 옵션을 설정하여 Internet Monitor에서 애플리케이션에 대한 모니터를 생성해야 합

니다. 이 장에서는 콘솔에 모니터를 추가하는 절차에 대해 설명합니다. 또한 Internet Monitor의 리소스에 대한 자세한 내용이 포함된 섹션과 모니터에 대해 구성할 수 있거나 구성해야 하는 다양한 옵션에 대한 설명 및 제한 사항이 포함된 추가 섹션이 포함되어 있습니다.

내용

콘솔을 사용하여 Amazon CloudWatch Internet Monitor에서 모니터 생성

애플리케이션에서 사용하는 AWS 리소스를 추가한 다음 몇 가지 구성 옵션을 설정하여 Amazon CloudWatch Internet Monitor에서 애플리케이션에 대한 모니터를 생성합니다. 추가하는 리소스(Amazon Virtual Private Cloud(VPC), Network Load Balancer(NLB), CloudFront 배포 또는 WorkSpaces 디렉터리)는 Internet Monitor가 애플리케이션의 인터넷 트래픽 정보를 매핑하는 데 필요한 정보를 제공합니다. 모니터를 생성한 후 15~30분 정도 기다리고 애플리케이션이 사용되는 위치별로 트래픽 프로필을 생성합니다. 그리고 Internet Monitor 모니터 또는 기타 도구를 사용하여 클라이언트 사용량에 대한 성능 및 가용성을 시각화하고 탐색할 수 있습니다. 이러한 도구는 모니터가 수집하여 CloudWatch 로그 등에 게시한 애플리케이션 트래픽 측정값을 사용하여 인사이트를 제공합니다.

일반적으로 Internet Monitor에서 하나의 애플리케이션에 대해 하나의 모니터를 만드는 것이 가장 간단합니다. 동일한 모니터 내에서 Internet Monitor 로그 파일의 측정값 및 지표를 다른 위치 및 ASN(일반적으로 인터넷 서비스 제공업체) 또는 기타 정보별로 검색하고 정렬할 수 있습니다. 예를 들어 다른 영역의 애플리케이션을 위해 별도의 모니터를 만들 필요가 없습니다.

여기 단계에서는 콘솔을 사용하여 모니터를 설정하는 방법을 안내합니다. Internet Monitor API 작업과 함께 AWS Command Line Interface을(를) 사용하여 모니터를 생성하고, 이벤트를 확인하는 등의 작업을 수행하는 예를 보려면 Amazon CloudWatch Internet Monitor에서 CLI를 사용하는 예 섹션을 참조하세요.

콘솔을 사용하여 모니터 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 애플리케이션 모니터링에서 인터넷 모니터를 선택합니다.
3. 모니터 생성(Create monitor)을 선택합니다.
4. Monitor name(모니터 이름)에 Internet Monitor에서 이 모니터에 사용할 이름을 입력합니다.
5. Add resources(리소스 추가)를 선택한 다음 이 모니터에 사용할 Internet Monitor의 모니터링 경계를 설정할 리소스를 선택합니다.

> **Note**
>
> 다음에 유의하세요.
>
> - Internet Monitor를 사용하여 유의미한 출력을 생성하려면 추가하는 VPC는 인터넷 게이트웨이를 구성하여 인터넷에 연결되어야 합니다.
> - VPC와 CloudFront 배포를 조합하여 추가하거나, WorkSpaces 디렉터리를 추가하거나, Network Load Balancer를 추가할 수 있습니다. Network Load Balancer 또는 WorkSpaces 디렉터리는 다른 유형의 리소스와 함께 추가할 수 없습니다.

6. 모니터링할 인터넷 트래픽의 백분율을 선택합니다.
7. 고급 설정에서 추가 옵션을 지정할 수도 있습니다.

   - 도시 네트워크 최대치에서 Internet Monitor가 트래픽을 모니터링할 도시-네트워크(위치 및 ASN 또는 인터넷 서비스 제공업체) 수에 대한 한도를 선택할 수 있습니다. 모니터를 편집하면 언제든지 변경할 수 있습니다. 도시-네트워크 최대 한도 선택 섹션을 참조하세요.

     기본값으로 재설정하려면 `500000`을 입력합니다.

     도시-네트워크 최대 한도를 설정하면 모니터하기로 선택한 트래픽 백분율과 관계없이 Internet Monitor가 애플리케이션에 대해 모니터하는 도시-네트워크 수에 대한 상한이 설정됩니다.
   - 원하는 경우 Amazon S3 버킷 이름과 사용자 지정 접두사를 지정하여 모니터링되는 모든 도시-네트워크에 대한 인터넷 측정값을 Amazon S3에 게시할 수 있습니다.

     Internet Monitor는 5분마다 애플리케이션에 대한 상위 500개(트래픽 볼륨 기준) 인터넷 측정값을 CloudWatch 로그에 게시합니다. 측정값을 S3에 게시하도록 선택한 경우 측정값은 여전히 CloudWatch 로그에 게시됩니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에서 Amazon S3에 인터넷 측정값 게시 단원을 참조하십시오.

- 필요에 따라 모니터에 대한 태그를 추가할 수 있습니다.

8. 모니터 생성(Create monitor)을 선택합니다.

모니터를 생성한 후에는 언제든지 모니터를 편집하여 애플리케이션 트래픽 백분율을 변경하거나, 최대 도시-네트워크 한도를 업데이트하거나, 리소스를 추가 또는 제거하는 등의 작업을 수행할 수 있습니다. 모니터를 삭제할 수도 있습니다. Internet Monitor 콘솔에서 이러한 태스크를 수행하려면 모니터를 선택한 다음 작업 메뉴에서 옵션을 선택합니다. 모니터 이름은 변경할 수 없습니다.

Internet Monitor 대시보드를 보려면 다음을 수행하세요.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 네트워크 모니터링을 선택한 다음 인터넷 모니터를 선택합니다.

   Monitors(모니터) 탭에는 사용자가 생성한 모니터 목록이 표시됩니다.

특정 모니터에 대한 자세한 내용을 보려면 모니터를 선택합니다.

모니터에 리소스 추가

모니터를 생성할 때 애플리케이션의 리소스(Amazon Virtual Private Cloud(VPC), Network Load Balancer(NLB), Amazon CloudFront 배포 또는 Amazon WorkSpaces 디렉터리)를 모니터과 연결해야 합니다. 그러면 Internet Monitor는 애플리케이션의 인터넷 측 트래픽과 클라이언트의 위치를 파악하여 모니터에 게시할 관련 측정값을 결정하는 트래픽 프로필을 생성하고 유지 관리할 수 있습니다.

Internet Monitor에서 모니터에 다음 유형의 리소스를 '모니터링되는 리소스'로 추가할 수 있습니다. Internet Monitor에서는 하나의 모니터에 여러 유형의 리소스를 함께 추가하는 방식을 지원하지 않습니다.

- VPC: 리전에 추가하는 각 VPC는 모니터되는 리소스입니다. VPC를 추가하면 Internet Monitor는 VPC의 모든 인터넷 연결 애플리케이션(예: Amazon EC2 인스턴스, Network Load Balancer 뒤 또는 AWS Fargate 컨테이너에 호스팅되는 애플리케이션)의 트래픽을 모니터링합니다.
- Network Load Balancer: 추가하는 각 NLB는 모니터되는 리소스입니다.
- CloudFront 배포: 추가하는 각 CloudFront 배포는 모니터되는 리소스입니다.
- WorkSpaces 디렉터리: 리전에 추가하는 각 WorkSpaces 디렉터리는 모니터되는 리소스입니다.

VPC의 트래픽을 모니터링할 때 VPC 뒤의 로드 밸런서에 호스트되는 애플리케이션의 트래픽이 모니터됩니다. 여러 로드 밸런서가 있는 VPC를 모니터링하는 대신 개별 Network Load Balancer 로드 밸런

서에 대한 트래픽을 모니터링하도록 선택할 수 있습니다. 이는 예를 들어 로드 밸런서 수준에서 성능 또는 효율성을 개선하기 위한 기능을 이해하고 구성해야 하는 경우 유용할 수 있습니다. 또는 Network Load Balancer 수준에서 규정 준수 정보가 필요할 수도 있습니다.

Internet Monitor에서 모니터에 리소스를 추가할 때는 다음 사항에 유의하세요.

- Internet Monitor를 사용하여 유의미한 출력을 생성하려면 추가하는 VPC는 인터넷 게이트웨이를 구성하여 인터넷에 연결되어야 합니다.
- Internet Monitor에서는 하나의 모니터에 여러 유형의 리소스를 함께 추가하는 방식을 지원하지 않습니다.

옵트인 리전에 대한 리전별 차이가 있으므로 VPC 또는 NLB를 리소스로 추가할 때 염두에 두어야 합니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에 대해 AWS 리전 지원 단원을 참조하십시오.

또한 마지막 마일 지연 시간 측정과 관련된 리소스에는 차이가 있습니다. Internet Monitor 지연 시간 측정의 경우 VPC, NLB, WorkSpaces 디렉터리에는 마지막 마일 지연 시간이 포함되지 않습니다.

모니터링할 애플리케이션 트래픽 백분율 선택

모니터링할 애플리케이션 트래픽 백분율에 대해 선택하는 적용 범위에 따라 애플리케이션에 대해 모니터되는 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 제공업체) 수가 결정되며, 설정할 수 있는 선택적 도시-네트워크 최대 제한까지 가능합니다.

애플리케이션 트래픽의 100% 미만을 모니터하도록 선택하면 모니터와 관찰성 격차가 존재할 수 있습니다. 트래픽을 모니터링하지 않는 곳에서 Amazon CloudWatch Internet Monitor가 생성하는 상태 이벤트가 있는 경우 이러한 문제를 인식하지 못하기 때문입니다. 또한 애플리케이션에 대한 클라이언트 액세스에 대한 성능 및 가용성 점수 정보에 대한 적용 범위가 줄어들 수 있습니다.

다음 섹션에서는 트래픽 백분율 설정 및 적용 범위를 탐색하고 적용 범위 확대 또는 축소가 미치는 영향을 확인하는 데 필요한 옵션을 설명합니다.

- 애플리케이션 트래픽 백분율 변경 살펴보기
- 다양한 트래픽 백분율 설정에서 모니터되는 도시-네트워크 수 보기

애플리케이션 트래픽 백분율 변경 살펴보기

백분율을 변경할 때 모니터되는 도시-네트워크 수를 확인하여 애플리케이션 트래픽 백분율을 변경할 수 있는 값을 탐색할 수 있습니다. 이 섹션의 절차는 단계별 정보를 제공합니다.

Internet Monitor 콘솔에서 모니터에 대한 애플리케이션 트래픽 백분율을 높이거나 낮추고 그 결과 적용될 도시-네트워크의 예상 수를 확인할 수 있습니다. 이 옵션을 사용하면 트래픽 백분율 변경이 모니터되는 도시-모니터 수에 어떤 영향을 미치는지 빠르게 확인할 수 있습니다. 이를 통해 애플리케이션에 적합한 애플리케이션 트래픽 백분율을 선택하는 데 도움이 될 수 있습니다.

애플리케이션 트래픽 백분율을 늘리거나 줄여 모니터링 범위를 탐색하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 애플리케이션 모니터링에서 인터넷 모니터를 선택합니다.
3. 모니터 목록에서 모니터를 선택합니다.
4. 개요 탭의 모니터링된 트래픽 섹션에서 백분율 그래프를 선택한 다음 모니터링 범위 업데이트를 선택합니다.
5. 트래픽 모니터링 범위 탐색 및 설정 대화 상자에서 화살표를 클릭하여 모니터링할 트래픽의 백분율을 늘리거나 줄입니다. 100% 트래픽을 선택하면 애플리케이션을 모니터링하기 위해 얼마나 많은 도시-네트워크가 모니터링되는지 확인할 수 있습니다.
6. 모니터하는 도시-네트워크 수(여기에서 추정)가 비용에 어떤 영향을 미치는지 자세히 알아보려면 CloudWatch Pricing calculator 링크를 선택한 다음 아래로 스크롤하여 Internet Monitor로 이동하세요.
7. 모니터링할 트래픽의 백분율을 새로 설정하려면 모니터 범위 업데이트를 선택합니다. 또는 현재 적용 범위를 유지하려면 취소를 선택합니다.

다양한 트래픽 백분율 설정에서 모니터되는 도시-네트워크 수 보기

애플리케이션에 대해 모니터링할 도시-네트워크의 수를 다양한 애플리케이션 트래픽 백분율로 확인할 수 있습니다. 이 섹션의 절차는 단계별 정보를 제공합니다.

Internet Monitor 콘솔에서는 지정한 시간 간격 동안 다양한 애플리케이션 트래픽 백분율에 따라 도시-네트워크의 적용 범위가 어떻게 변하는지 보여주는 그래프를 볼 수 있습니다. 이는 특정 트래픽 백분율로 애플리케이션의 모니터링 범위를 하나의 그래프로 시각화하고 비교할 수 있는 빠른 방법입니다.

애플리케이션 트래픽 백분율 및 해당 도시-네트워크 범위 그래프를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 애플리케이션 모니터링에서 인터넷 모니터를 선택합니다.
3. 모니터 목록에서 모니터를 선택합니다.
4. 트래픽 인사이트 탭을 선택하고 아래로 스크롤하여 인터넷 트래픽 그래프로 이동합니다.

5. 트래픽 범위 비교 옵션의 드롭다운 목록에서 하나 이상의 백분율을 선택합니다. 하나 이상의 응용 프로그램 트래픽 백분율을 선택할 수 있으며 모니터링된 총 도시- 네트워크 그래프가 업데이트되어 Internet Monitor가 해당 트래픽 백분율에 대해 제공하는 모니터링 범위를 표시합니다. 트래픽 100%의 도시-네트워크를 선택하면 애플리케이션 모니터링을 위해 전체 범위로 모니터링되는 도시-네트워크 수를 확인할 수 있습니다.

다음 사항에 유의하세요.

- 트래픽 범위는 애플리케이션 트래픽의 이전 1시간 동안의 도시-네트워크 수를 기반으로 계산됩니다. 즉, 모니터링할 트래픽의 특정 비율을 선택한 후에는 아래 교통 범위 비교 그래프에 표시된 것보다 애플리케이션에서 모니터링되는 도시-네트워크 수가 적을 수 있습니다.
- 모든 애플리케이션 트래픽을 모니터링하려면 `TrafficPercentageToMonitor`를 100으로 설정하고 `MaxCityNetworksToMonitor`를 설정하지 마세요. 또는 `MaxCityNetworksToMonitor`를 Internet Monitor의 상한값인 500,000으로 설정할 수 있습니다.
- 도시-네트워크 최대 한도를 설정한 경우 선택한 애플리케이션 트래픽 백분율 옵션과 상관없이 모니터되는 도시-네트워크의 총 수는 이 한도를 초과하지 않습니다.
- 모니터되는 도시-네트워크 수가 비용에 어떤 영향을 미칠 수 있는지 자세히 알아볼 수 있습니다. CloudWatch Pricing calculator 페이지에서 Internet Monitor까지 아래로 스크롤합니다.

모니터링할 트래픽의 백분율을 새로 설정하려면 다른 트래픽 범위 탐색 옵션에서 모니터링 범위 업데이트를 선택합니다. 대화 상자에서 트래픽 백분율을 선택한 다음 모니터 범위 업데이트를 선택합니다.

도시-네트워크 최대 한도 선택

Amazon CloudWatch Internet Monitor는 클라이언트가 애플리케이션 리소스에 액세스하는 일부 또는 모든 위치 그리고 클라이언트가 애플리케이션에 액세스하는 모든 ASN(일반적으로 인터넷 서비스 공급업체), 즉 애플리케이션 인터넷 트래픽을 위한 도시-네트워크에 대한 애플리케이션 트래픽을 모니터링할 수 있습니다. 모니터를 만들 때 모니터링할 애플리케이션 트래픽의 백분율을 선택합니다. 모니터를 편집하여 언제든지 업데이트할 수 있습니다.

트래픽 백분율을 설정하는 것 외에도 모니터하는 도시-네트워크 수에 대한 최대 한도를 설정할 수도 있습니다. 이 섹션에서는 도시-네트워크 한도가 청구 비용을 관리하는 데 어떻게 도움이 되는지 설명하고 설정할 한도를 결정하는 데 도움이 되는 정보와 예시를 제공합니다.

도시-네트워크 수에 설정한 최대 한도는 청구서를 예측할 수 있도록 도와줍니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요. 또한 CloudWatch 요금 계산기를 사용하면 실제로 모니터되

는 도시-네트워크 수의 다양한 값이 청구서에 어떤 영향을 미치는지 알아볼 수 있습니다. 옵션을 살펴보려면 CloudWatch Pricing calculator 페이지에서 Internet Monitor까지 아래로 스크롤합니다.

모니터를 업데이트하고 도시-네트워크 최대 한도를 변경하려면 Internet Monitor 모니터 편집 또는 삭제 섹션을 참조하세요.

도시-네트워크 최대 한도를 기준으로 청구가 작동하는 방식

모니터하는 도시-네트워크 수에 최대 한도를 설정하면 청구서에 예상치 못한 비용이 청구되는 것을 방지할 수 있습니다. 예를 들어 트래픽 패턴이 매우 다양한 경우 유용합니다. 처음 100개의 도시-네트워크가 포함된 이후에는 (계정당 모든 모니터에 걸쳐) 모니터하는 각 도시-네트워크에 대해 청구 비용이 증가합니다. 도시-네트워크 최대 한도를 설정하면 모니터하기로 선택한 트래픽 백분율과 관계없이 Internet Monitor가 애플리케이션에 대해 모니터하는 도시-네트워크 수에 대한 상한이 설정됩니다.

실제로 모니터하는 도시-네트워크의 수에 대해서만 비용을 지불합니다. 선택한 도시-네트워크 최대 한도를 사용하면 Internet Monitor가 모니터로 트래픽을 모니터링할 때 포함할 수 있는 총합에 대한 상한을 설정할 수 있습니다. 모니터를 편집하여 언제든지 최대 한도를 변경할 수 있습니다.

옵션을 살펴보려면 CloudWatch Pricing calculator 페이지에서 Internet Monitor까지 아래로 스크롤합니다. Internet Monitor 요금에 대한 자세한 내용은 Amazon CloudWatch 요금 페이지의 Internet Monitor 섹션을 참조하세요.

도시-네트워크 최대 한도를 선택하는 방법

선택할 도시-네트워크 최대 한도를 결정하는 데 도움이 되도록 애플리케이션에 대해 모니터하려는 트래픽의 양을 고려하세요. Internet Monitor 지표(`CityNetworksMonitored`, `TrafficMonitoredPercent`, `CityNetworksFor`*`NN`*`PercentTraffic` 지표 중 하나 이상)는 모니터를 생성한 후 트래픽 사용량 및 적용 범위를 분석하는 데 도움이 될 수 있습니다. 여기서 *NN*은 25, 50, 90, 95, 99, 100 중 하나에 해당하는 백분율 값입니다. 이러한 지표 및 기타 모든 Internet Monitor 지표에 대한 정의를 검토하려면 Amazon CloudWatch Internet Monitor와 함께 CloudWatch 지표 사용 섹션을 참조하세요.

인터넷 트래픽 커버리지에 대한 개요 그래프를 보려면 CloudWatch 대시보드의 트래픽 인사이트 탭으로 이동하여 인터넷 트래픽 그래프 섹션에서 트래픽 범위에 대한 옵션 비교 옵션을 선택하세요. 이 섹션에 표시된 그래프에는 애플리케이션에 대해 모니터되는 실제 도시-네트워크 수와 드롭다운 목록에서 선택한 다양한 애플리케이션 트래픽 백분율에 대한 그래프 선이 표시됩니다. 자세한 내용은 애플리케이션 트래픽 백분율 설정을 참조하세요.

옵션을 더 자세히 살펴보려면 다음 예에 설명된 대로 Internet Monitor 지표를 사용하면 됩니다. 이 예는 원하는 애플리케이션 인터넷 트래픽 범위의 폭에 따라 가장 적합한 최대 도시-네트워크 제한을 선택하

는 방법을 보여줍니다. CloudWatch 지표의 Internet Monitor 지표에 대한 쿼리를 사용하면 애플리케이션 인터넷 트래픽 범위에 대해 자세히 파악할 수 있습니다.

도시-네트워크 최대 한도 결정 예

예를 들어 모니터링 최대 한도를 100개의 도시-네트워크로 설정하고 2,637개의 도시-네트워크에서 클라이언트가 애플리케이션에 액세스한다고 가정하겠습니다. CloudWatch 지표에서 다음과 같은 Internet Monitor 지표가 반환되는 것을 볼 수 있습니다.

```
CityNetworksMonitored 100
TrafficMonitoredPercent  12.5
CityNetworksFor90PercentTraffic  2143
CityNetworksFor100PercentTraffic  2637
```

이 예에서 현재 인터넷 트래픽의 12.5%를 모니터링하고 있으며 최대 제한이 100개 도시-네트워크로 설정되어 있음을 알 수 있습니다. 트래픽의 90%를 모니터하려는 경우 다음 지표에서 이에 대한 정보를 제공합니다. 즉, `CityNetworksFor90PercentTraffic`은 90% 범위를 위해 2,143개의 도시-네트워크를 모니터해야 한다는 것을 나타냅니다. 이렇게 하려면 모니터를 업데이트하고 최대 도시-네트워크 한도를 2,143개로 설정하면 됩니다.

마찬가지로 애플리케이션에 대해 100% 인터넷 트래픽을 모니터링하고 싶다고 가정하겠습니다. 다음 지표인 `CityNetworksFor100PercentTraffic`은 이를 위해 최대 도시-네트워크 제한을 2,637로 설정하도록 모니터를 업데이트해야 함을 나타냅니다.

이제 도시-네트워크를 최대 5,000개로 설정하면 2,637개를 초과하므로 다음과 같은 지표가 반환됩니다.

```
CityNetworksMonitored 2637
TrafficMonitoredPercent  100
CityNetworksFor90PercentTraffic  2143
CityNetworksFor100PercentTraffic  2637
```

이러한 지표를 통해 한도가 높을수록 인터넷 트래픽의 100%에 해당하는 2,637개 도시-네트워크를 모두 모니터링한다는 것을 알 수 있습니다.

Amazon CloudWatch Internet Monitor에서 Amazon S3에 인터넷 측정값 게시

Amazon CloudWatch Internet Monitor가 모니터에서 모니터되는 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 제공업체)로의 인터넷 연결 트래픽에 대한 인터넷 측정값을 도시-

네트워크 서비스 한도 최대 500,000개까지 Amazon S3에 게시하도록 선택할 수 있습니다. Internet Monitor는 각 모니터의 상위 500개(트래픽 볼륨 기준) 도시-네트워크에 대해 5분마다 인터넷 측정값을 CloudWatch 로그에 자동으로 게시합니다. S3에 게시하는 측정값에는 CloudWatch 로그에 게시되는 상위 500개 측정값이 포함됩니다.

S3에 게시하는 옵션을 선택하고 모니터를 생성하거나 업데이트할 때 측정값을 게시할 버킷을 지정할 수 있습니다. Internet Monitor에서 버킷을 지정하려면 먼저 S3에서 버킷이 이미 생성되어 있어야 합니다. S3에 게시되는 인터넷 측정값의 서비스 한도는 500,000개의 도시-네트워크입니다. Internet Monitor는 인터넷 측정값을 버킷에 저장되는 일련의 압축된 로그 파일 개체인 이벤트로 S3에 게시합니다.

Internet Monitor가 측정값을 게시할 S3 버킷을 만들 때는 CloudWatch Logs에서 제공하는 권한 지침을 따라야 합니다. 이렇게 하면 Internet Monitor가 S3에 직접 로그를 게시할 수 있으며, 필요한 경우 AWS에서 로그를 수신하는 로그 그룹과 관련된 리소스 정책을 생성하고 변경할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 가이드에서 CloudWatch Logs로 전송된 로그를 참조하세요.

게시된 로그 파일은 압축된 상태입니다. Amazon S3 콘솔을 사용하여 로그 파일을 열면 압축이 해제되고 인터넷 측정 이벤트가 표시됩니다. 파일을 다운로드한 경우 이벤트를 보려면 압축을 해제해야 합니다.

Amazon Athena를 사용하여 로그 파일에서 인터넷 측정값을 쿼리할 수도 있습니다. Amazon Athena는 표준 SQL을 사용하여 Amazon S3에서 데이터를 더 쉽게 분석할 수 있는 대화형 쿼리 서비스입니다. 자세한 내용은 Amazon Athena를 사용하여 Amazon S3 로그 파일에서 인터넷 측정값 쿼리 단원을 참조하십시오.

Internet Monitor 모니터 사용

Amazon CloudWatch Internet Monitor 모니터를 생성한 후 사용하는 방법에는 여러 가지가 있습니다. 예를 들어 CloudWatch 대시보드에서 정보를 보고, AWS Command Line Interface를 사용하여 정보를 얻고, 상태 알림을 설정할 수 있습니다.

모니터는 애플리케이션 및 구성 기본 설정에 대한 정보를 제공하므로 Internet Monitor는 이벤트에 게시할 측정 및 지표를 사용자 지정할 수 있습니다. Internet Monitor는 AWS에 대한 글로벌 인프라 공간에서 측정값을 수집합니다. 이러한 측정값은 전 세계에서 수집된 엄청난 양의 네트워크 성능 및 가용성 정보입니다. Internet Monitor는 애플리케이션에 추가하는 리소스의 정보를 사용하여 애플리케이션이 활성화되어 있는 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 공급업체(ISP))로 범위가 지정된 성능 및 가용성 측정값을 게시합니다. 따라서 가용성, 성능, 모니터링된 바이트 전송량, 왕복 시간 등 Internet Monitor 대시보드와 CloudWatch 로그의 측정값 및 지표는 클라이언트 위치 및 ASN에 따라 다릅니다.

또한 Internet Monitor는 성능 및 가용성에 이상이 있는 시점도 확인합니다. 기본적으로 Internet Monitor는 트래픽에 AWS가 클라이언트 위치의 각 소스-대상 쌍에 대해 수집한 가용성 및 성능 측정값을 오버레이하여 성능 또는 가용성이 눈에 띄게 저하되는 시점을 확인합니다. 애플리케이션의 위치 및 범위에서 성능이 크게 저하되면 Internet Monitor는 상태 이벤트를 생성하고 문제에 대한 정보를 모니터에 게시합니다.

모니터를 생성한 후에는 다음과 같은 방법으로 모니터를 사용하여 Internet Monitor가 제공하는 정보에 액세스하거나 알림을 받을 수 있습니다.

- CloudWatch 대시보드를 사용하여 성능, 가용성, 상태 이벤트를 보고 탐색하고, 애플리케이션의 기록 데이터를 탐색하고, 더 나은 성능을 위해 애플리케이션을 구성하는 새로운 방법에 대한 인사이트를 얻을 수 있습니다. 자세한 내용은 다음을 참조하세요.
  - Amazon CloudWatch Internet Monitor에서 실시간 성능 및 가용성 추적(Overview(개요) 탭)
  - Amazon CloudWatch Internet Monitor에서 기록 데이터 필터링 및 보기(기록 탐색기 탭)
  - Amazon CloudWatch Internet Monitor에서 애플리케이션 성능 개선을 위한 인사이트 얻기(트래픽 인사이트 탭)
- 상태 이벤트 임계값을 구성하여 Internet Monitor가 애플리케이션에 상태 이벤트를 생성하도록 트리거하는 요인을 변경하세요. 전체 임계값과 로컬(도시-네트워크) 임계값을 구성할 수 있습니다. 자세히 알아보려면 상태 이벤트 임계값 변경을 참조하세요.
- Internet Monitor API 동작과 함께 AWS CLI 명령을 사용하여 트래픽 프로필 정보를 보고, 측정값을 보고, 상태 이벤트를 나열하는 등의 작업을 수행할 수 있습니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에서 CLI를 사용하는 예 단원을 참조하세요.
- CloudWatch Contributor Insights, CloudWatch Metrics 탐색기 및 CloudWatch Logs Insights와 같은 표준 CloudWatch 도구를 사용하여 CloudWatch의 데이터를 시각화합니다. 자세한 내용은 CloudWatch 도구 및 Internet Monitor 쿼리 인터페이스로 데이터 탐색 단원을 참조하세요.
- S3에 측정값 게시를 사용 설정한 경우 S3 로그와 함께 Athena를 사용하여 애플리케이션에 대한 Internet Monitor 인터넷 측정값에 액세스하고 분석할 수 있습니다.
- Internet Monitor가 상태 이벤트가 있음을 확인하면 알려주는 Amazon EventBridge 알림을 생성합니다. 자세한 내용은 Amazon EventBridge와 함께 Amazon CloudWatch Internet Monitor 사용 단원을 참조하세요.
- Internet Monitor가 AWS 네트워크에 의해 문제가 발생했다고 판단하면 AWS Health Dashboard 알림을 자동으로 수신합니다. 알림에는 문제를 완화하기 위해 AWS에서 취하고 있는 조치가 포함되어 있습니다.

Internet Monitor 모니터 편집 또는 삭제

Amazon CloudWatch Internet Monitor에서 모니터를 생성한 후 작업 메뉴를 사용하여 모니터를 편집하거나 삭제할 수 있습니다. 예를 들어 모니터를 편집하여 다음과 같은 작업을 수행할 수 있습니다.

- 모니터링할 애플리케이션 트래픽의 백분율 변경
- 도시-네트워크 최대 한도 설정 또는 업데이트
- 가용성 또는 성능 점수에 대한 상태 이벤트 임계값 변경
- 리소스 추가 또는 제거
- Amazon S3에 게시 이벤트 사용 설정 또는 업데이트

모니터를 삭제할 수도 있습니다. 모니터를 생성한 후에는 모니터 이름을 변경할 수 없다는 점에 유의하세요.

모니터를 변경하거나 모니터를 삭제하려면 다음 절차 중 하나를 사용하세요.

모니터를 편집하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 애플리케이션 모니터링에서 인터넷 모니터를 선택합니다.
3. 모니터를 선택한 다음 작업 메뉴를 선택합니다.
4. 모니터 업데이트를 선택합니다.
5. 원하는 대로 업데이트합니다. 예를 들어 모니터링할 트래픽의 백분율을 변경하려면 모니터링할 애플리케이션 트래픽 아래에서 백분율을 선택하거나 입력합니다.
6. 업데이트를 선택합니다.

모니터를 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 애플리케이션 모니터링에서 인터넷 모니터를 선택합니다.
3. 모니터를 선택한 다음 작업 메뉴를 선택합니다.
4. 비활성화를 선택합니다.
5. 작업 메뉴를 다시 선택한 다음 삭제를 선택합니다.

업데이트할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요.

- Internet Monitor에서 추가하는 리소스에 대해 자세히 알아보려면 모니터에 리소스 추가 섹션을 참조하세요.
- 애플리케이션 트래픽 백분율에 대해 자세히 알아보려면 모니터링할 애플리케이션 트래픽 백분율 선택 섹션을 참조하세요.
- 상태 이벤트 임계값 변경에 대해 자세히 알아보려면 상태 이벤트 임계값 변경을 참조하세요.
- 도시-네트워크 최대 한도에 대해 자세히 알아보려면 도시-네트워크 최대 한도 선택 섹션을 참조하세요.
- S3에 이벤트를 게시하도록 선택하는 방법에 대해 자세히 알아보려면 Amazon CloudWatch Internet Monitor에서 Amazon S3에 인터넷 측정값 게시 섹션을 참조하세요.

Amazon VPC를 사용하여 Amazon CloudWatch Internet Monitor 모니터 추가 또는 생성

AWS Management Console에서 Amazon Virtual Private Cloud VPC를 생성할 때 Amazon CloudWatch Internet Monitor에서 해당 VPC에 대한 모니터링을 설정하도록 선택할 수 있습니다. VPC를 기존 모니터에 추가하거나 Amazon VPC 콘솔에서 VPC용 새 모니터를 생성할 수 있습니다.

VPC와 함께 Internet Monitor를 사용하면 가용성, 성능, 모니터링된 바이트 전송량, 애플리케이션의 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 제공업체)에 대한 왕복 시간 등의 측정값 및 지표를 보고 평가할 수 있습니다. 또한 Internet Monitor는 성능 및 가용성에 이상이 있는 시점도 확인하고 모니터에 상태 이벤트를 생성하며, 사용자는 이에 대한 알림을 받을 수 있습니다. 모니터를 사용하여 클라이언트의 애플리케이션 사용 경험을 관리하고 개선하는 방법에 대한 자세한 내용은 Internet Monitor 모니터 사용 섹션을 참조하세요.

> **Important**
>
> 모니터를 생성하거나 모니터에 VPC를 추가하려면 적절한 권한이 있어야 합니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에 대한 ID 및 액세스 관리 단원을 참조하십시오.

기존 모니터에 VPC 추가

AWS Management Console에서 VPC를 생성할 때 Amazon CloudWatch Internet Monitor가 기존 모니터에 새 VPC를 추가하도록 선택할 수 있습니다. VPC를 추가한 후 몇 분 정도 기다리면 VPC에 대한 지표가 Internet Monitor 콘솔에 표시되기 시작합니다.

언제든지 모니터를 편집하여 VPC를 제거하거나 다른 VPC 또는 다른 리소스를 추가할 수 있습니다. 모니터링 중인 트래픽의 비율을 변경하거나 다른 사항을 변경할 수도 있습니다. 모니터에서 VPC를 제거

하기로 선택하면 클라이언트에서 해당 VPC로 향하는 트래픽은 더 이상 Internet Monitor에서 모니터링되지 않습니다.

모니터 업데이트에 대한 자세한 내용은 Internet Monitor 모니터 편집 또는 삭제 섹션을 참조하세요.

VPC용 모니터 생성

VPC용 모니터를 생성하기로 선택한 경우, 모니터 생성 마법사가 생성 단계를 안내합니다. 모니터를 생성할 때 VPC를 모니터링되는 리소스로 추가합니다. 원하는 경우 애플리케이션에 대해 모니터링할 클라이언트 트래픽의 비율을 선택할 수도 있습니다(기본값은 100%).

콘솔을 사용하여 Amazon CloudWatch Internet Monitor에서 모니터 생성에서 해당 정보를 검토하고 자세히 알아볼 수 있습니다.

요금

Amazon CloudWatch Internet Monitor에서는 사용한 만큼만 지불하면 됩니다. Internet Monitor의 요금은 모니터되는 리소스당 요금과 도시-네트워크당 요금의 두 가지 구성 요소로 이루어져 있습니다. 도시-네트워크는 클라이언트가 애플리케이션 리소스에 액세스하는 위치 및 클라이언트가 리소스에 액세스하는 네트워크(ASN. 예: 인터넷 서비스 제공업체(ISP))를 의미합니다.

요금 예시를 포함한 자세한 내용은 Amazon CloudWatch Internet Monitor 요금을 참조하세요.

VPC 모니터링 중지

Internet Monitor를 통한 VPC 리소스 모니터링을 중단하려면 Internet Monitor 콘솔에서 다음을 수행합니다.

모니터에서 리소스 제거하기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 애플리케이션 모니터링에서 인터넷 모니터를 선택합니다.
3. 모니터를 선택한 다음 작업 메뉴를 선택합니다.
4. 모니터 업데이트를 선택합니다.
5. 추가된 리소스에서 리소스 제거를 선택합니다.
6. 제거할 VPC를 선택한 다음 제거를 선택합니다.
7. 업데이트를 선택합니다.

CloudFront를 사용하여 Amazon CloudWatch Internet Monitor 모니터 추가 또는 생성

Amazon CloudFront 콘솔의 배포에 대한 지표 대시보드에서 Amazon CloudWatch Internet Monitor의 배포에 대한 추가 모니터링을 설정할 수 있습니다. 배포를 기존 모니터에 추가하거나 배포에 사용할 새 모니터를 생성할 수 있습니다.

CloudFront 배포와 함께 Internet Monitor를 사용하면 가용성, 성능, 모니터링된 바이트 전송량, 애플리케이션의 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 제공업체)에 대한 왕복 시간 등의 측정값 및 지표를 보고 평가할 수 있습니다. 또한 Internet Monitor는 성능 및 가용성에 이상이 있는 시점도 확인하고 모니터에 상태 이벤트를 생성하며, 사용자는 이에 대한 알림을 받을 수 있습니다. 모니터를 사용하여 클라이언트의 애플리케이션 사용 경험을 관리하고 개선하는 방법에 대한 자세한 내용은 Internet Monitor 모니터 사용 섹션을 참조하세요.

> **Important**
>
> 모니터를 생성하거나 기존 모니터에 배포를 추가하려면 적절한 권한이 있어야 합니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에 대한 ID 및 액세스 관리 단원을 참조하십시오.

기존 모니터에 배포판 추가

AWS Management Console의 CloudFront 지표 대시보드에서 Internet Monitor가 기존 모니터에 배포를 직접 추가하도록 선택할 수 있습니다. 배포를 추가한 후 몇 분 정도 기다리면 배포에 대한 지표가 Internet Monitor 콘솔에 표시되기 시작합니다.

언제든지 모니터를 편집하여 배포를 제거하거나 다른 배포 또는 다른 리소스를 추가할 수 있습니다. 모니터링 중인 트래픽의 비율을 변경하거나 다른 사항을 변경할 수도 있습니다. 모니터에서 배포를 제거하기로 선택하면 클라이언트에서 해당 배포로 향하는 트래픽은 더 이상 Internet Monitor에서 모니터링되지 않습니다.

모니터 업데이트에 대한 자세한 내용은 Internet Monitor 모니터 편집 또는 삭제 섹션을 참조하세요.

배포용 모니터 생성

배포용 모니터를 생성하기로 선택한 경우, 모니터 생성 마법사가 생성 단계를 안내합니다. 모니터를 생성할 때 배포를 모니터링되는 리소스로 추가합니다. 원하는 경우 애플리케이션에 대해 모니터링할 클라이언트 트래픽의 비율을 선택할 수도 있습니다(기본값은 100%).

콘솔을 사용하여 Amazon CloudWatch Internet Monitor에서 모니터 생성에서 해당 정보를 검토하고 자세히 알아볼 수 있습니다.

요금

Amazon CloudWatch Internet Monitor에서는 사용한 만큼만 지불하면 됩니다. Internet Monitor의 요금은 모니터되는 리소스당 요금과 도시-네트워크당 요금의 두 가지 구성 요소로 이루어져 있습니다. 도시-네트워크는 클라이언트가 애플리케이션 리소스에 액세스하는 위치 및 클라이언트가 리소스에 액세스하는 네트워크(ASN. 예: 인터넷 서비스 제공업체(ISP))를 의미합니다.

요금 예시를 포함한 자세한 내용은 Amazon CloudWatch Internet Monitor 요금을 참조하세요.

배포 모니터링 중지

Internet Monitor를 통한 배포 리소스 모니터링을 중단하려면 Internet Monitor 콘솔에서 다음을 수행합니다.

모니터에서 리소스 제거하기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 애플리케이션 모니터링에서 인터넷 모니터를 선택합니다.
3. 모니터를 선택한 다음 작업 메뉴를 선택합니다.
4. 모니터 업데이트를 선택합니다.
5. 추가된 리소스에서 리소스 제거를 선택합니다.
6. 제거할 배포를 선택한 다음 제거를 선택합니다.
7. 업데이트를 선택합니다.

Amazon CloudWatch Internet Monitor에서 CLI를 사용하는 예

이 섹션에는 Amazon CloudWatch Internet Monitor 운영에 AWS Command Line Interface를 사용하는 예가 포함되어 있습니다.

시작하기 전에 모니터하려는 Amazon Virtual Private Cloud(VPC), Network Load Balancers, Amazon CloudFront 배포 또는 Amazon WorkSpaces 디렉터리가 있는 동일한 AWS 계정으로 AWS CLI를 사용하려면 로그인해야 합니다. Internet Monitor는 계정 간 리소스 액세스를 지원하지 않습니다. AWS CLI 사용에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하세요. Amazon CloudWatch Internet Monitor에서 API 작업을 사용하는 방법에 대한 자세한 내용은 Amazon CloudWatch Internet Monitor API 참조 안내서를 참조하세요.

주제

- 모니터 생성

- 모니터 세부 정보 보기
- 상태 이벤트 나열
- 특정 상태 이벤트 보기
- 모니터 목록 보기
- 모니터 편집
- 모니터 삭제

모니터 생성

Internet Monitor에서 모니터를 생성할 때 이름을 제공하고 리소스를 모니터와 연결하여 애플리케이션의 인터넷 트래픽 위치를 표시합니다. 모니터링할 애플리케이션 트래픽의 양을 정의하는 트래픽 백분율을 지정합니다. 또한 모니터되는 도시-네트워크, 즉 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 공급업체(ISP))의 수에 따라 모니터 대상도 결정됩니다. 또한 애플리케이션 리소스를 모니터링할 최대 도시-네트워크 수에 대한 한도를 설정하여 요금을 관리할 수도 있습니다. 자세한 내용은 도시-네트워크 최대 한도 선택 단원을 참조하십시오.

마지막으로 애플리케이션에 대한 모든 인터넷 측정값을 Amazon S3에 게시할지 여부를 선택할 수 있습니다. 상위 500개 도시-네트워크(트래픽 양 기준)에 대한 인터넷 측정값은 Internet Monitor별 CloudWatch 로그에 자동으로 게시되지만, 모든 측정값을 S3에도 게시하도록 선택할 수 있습니다.

AWS CLI를 통해 모니터를 생성하려면 `create-monitor` 명령을 사용합니다. 다음 명령은 트래픽을 100% 모니터하지만 최대 도시-네트워크 제한을 10,000으로 설정하고, VPC 리소스를 추가하고, 인터넷 측정값을 Amazon S3에 게시하도록 선택하는 모니터를 생성합니다.

> **Note**
>
> Internet Monitor는 각 모니터에 트래픽을 전송하는 상위 500개 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 공급업체(ISP))에 대한 인터넷 측정값을 5분마다 CloudWatch 로그에 게시합니다. 원하는 경우, 모니터링되는 모든 도시-네트워크(최대 500,000개의 도시-네트워크 서비스 제한)에 대한 인터넷 측정값을 Amazon S3 버킷에 게시하도록 선택할 수 있습니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에서 Amazon S3에 인터넷 측정값 게시 단원을 참조하십시오.

```
aws internetmonitor --create-monitor monitor-name "TestMonitor" \
    --traffic-percentage-to-monitor 100 \
    --max-city-networks-to-monitor 10000 \
```

```
    --resources "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889" \
    --internet-measurements-log-delivery
 S3Config="{BucketName=MyS3Bucket,LogDeliveryStatus=ENABLED}"
```

```
{
"Arn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
 "Status": "ACTIVE"
}
```

> **Note**
>
> 모니터 이름은 변경할 수 없습니다.

모니터 세부 정보 보기

AWS CLI가 있는 모니터에 대한 정보를 보려면 `get-monitor` 명령을 사용합니다.

```
aws internetmonitor get-monitor --monitor-name "TestMonitor"
```

```
{
    "ClientLocationType": "city",
    "CreatedAt": "2022-09-22T19:27:47Z",
    "ModifiedAt": "2022-09-22T19:28:30Z",
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "MonitorName": "TestMonitor",
    "ProcessingStatus": "OK",
    "ProcessingStatusInfo": "The monitor is actively processing data",
    "Resources": [
        "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889"
    ],
    "MaxCityNetworksToMonitor": 10000,
    "Status": "ACTIVE"
}
```

상태 이벤트 나열

애플리케이션의 인터넷 트래픽 성능이 저하되면 Internet Monitor는 모니터에 상태 이벤트를 생성합니다. AWS CLI를 사용하여 현재 상태 이벤트 목록을 보려면 `list-health-events` 명령을 사용합니다.

```
aws internetmonitor list-health-events --monitor-name "TestMonitor"
```

```
{
    "HealthEvents": [
        {
            "EventId": "2022-06-20T01-05-05Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:15:14Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.21,
            "ClientLocations": [
                {
                    "City": "Lockport",
                    "PercentOfClientLocationImpacted": 60.370000000000005,
                    "PercentOfTotalTraffic": 2.01,
                    "Country": "United States",
                    "Longitude": -78.6913,
                    "AutonomousSystemNumber": 26101,
                    "Latitude": 43.1721,
                    "Subdivision": "New York",
                    "NetworkName": "YAHOO-BF1"
                }
            ],
            "StartedAt": "2022-06-20T01:05:05Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-05-05Z/latency"
        },
        {
            "EventId": "2022-06-20T01-17-56Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:30:23Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.29,
            "ClientLocations": [
```

```
            {
                "City": "Toronto",
                "PercentOfClientLocationImpacted": 75.32,
                "PercentOfTotalTraffic": 1.05,
                "Country": "Canada",
                "Longitude": -79.3623,
                "AutonomousSystemNumber": 14061,
                "Latitude": 43.6547,
                "Subdivision": "Ontario",
                "CausedBy": {
                    "Status": "ACTIVE",
                    "Networks": [
                        {
                            "AutonomousSystemNumber": 16509,
                            "NetworkName": "Amazon.com"
                        }
                    ],
                    "NetworkEventType": "AWS"
                },
                "NetworkName": "DIGITALOCEAN-ASN"
            },
            {
                "City": "Lockport",
                "PercentOfClientLocationImpacted": 22.91,
                "PercentOfTotalTraffic": 2.01,
                "Country": "United States",
                "Longitude": -78.6913,
                "AutonomousSystemNumber": 26101,
                "Latitude": 43.1721,
                "Subdivision": "New York",
                "NetworkName": "YAHOO-BF1"
            },
            {
                "City": "Hangzhou",
                "PercentOfClientLocationImpacted": 2.88,
                "PercentOfTotalTraffic": 0.7799999999999999,
                "Country": "China",
                "Longitude": 120.1612,
                "AutonomousSystemNumber": 37963,
                "Latitude": 30.2994,
                "Subdivision": "Zhejiang",
                "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
            }
        ],
```

```
            "StartedAt": "2022-06-20T01:17:56Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-17-56Z/latency"
        },
        {
            "EventId": "2022-06-20T01-34-20Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:35:04Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.15,
            "ClientLocations": [
                {
                    "City": "Lockport",
                    "PercentOfClientLocationImpacted": 39.45,
                    "PercentOfTotalTraffic": 2.01,
                    "Country": "United States",
                    "Longitude": -78.6913,
                    "AutonomousSystemNumber": 26101,
                    "Latitude": 43.1721,
                    "Subdivision": "New York",
                    "NetworkName": "YAHOO-BF1"
                },
                {
                    "City": "Toronto",
                    "PercentOfClientLocationImpacted": 29.770000000000003,
                    "PercentOfTotalTraffic": 1.05,
                    "Country": "Canada",
                    "Longitude": -79.3623,
                    "AutonomousSystemNumber": 14061,
                    "Latitude": 43.6547,
                    "Subdivision": "Ontario",
                    "CausedBy": {
                        "Status": "ACTIVE",
                        "Networks": [
                            {
                                "AutonomousSystemNumber": 16509,
                                "NetworkName": "Amazon.com"
                            }
                        ],
```

```
                        "NetworkEventType": "AWS"
                    },
                    "NetworkName": "DIGITALOCEAN-ASN"
                },
                {
                    "City": "Hangzhou",
                    "PercentOfClientLocationImpacted": 2.88,
                    "PercentOfTotalTraffic": 0.7799999999999999,
                    "Country": "China",
                    "Longitude": 120.1612,
                    "AutonomousSystemNumber": 37963,
                    "Latitude": 30.2994,
                    "Subdivision": "Zhejiang",
                    "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
                }
            ],
            "StartedAt": "2022-06-20T01:34:20Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-34-20Z/latency"
        }
    ]
}
```

특정 상태 이벤트 보기

CLI를 사용하여 특정 상태 이벤트에 대한 자세한 정보를 보려면 모니터 이름과 상태 이벤트 ID를 사용하여 `get-health-event` 명령을 실행합니다.

```
aws internetmonitor get-monitor --monitor-name "TestMonitor" --event-id "health-event/
TestMonitor/2021-06-03T01:02:03Z/latency"
```

```
{
    "EventId": "2022-06-20T01-34-20Z/latency",
    "Status": "RESOLVED",
    "EndedAt": "2022-06-20T01:35:04Z",
    "ServiceLocations": [
        {
            "Name": "us-east-1"
        }
    ],
    "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor/
health-event/2022-06-20T01-34-20Z/latency",
```

```
    "LastUpdatedAt": "2022-06-20T01:35:04Z",
    "ClientLocations": [
        {
            "City": "Lockport",
            "PercentOfClientLocationImpacted": 39.45,
            "PercentOfTotalTraffic": 2.01,
            "Country": "United States",
            "Longitude": -78.6913,
            "AutonomousSystemNumber": 26101,
            "Latitude": 43.1721,
            "Subdivision": "New York",
            "NetworkName": "YAHOO-BF1"
        },
        {
            "City": "Toronto",
            "PercentOfClientLocationImpacted": 29.770000000000003,
            "PercentOfTotalTraffic": 1.05,
            "Country": "Canada",
            "Longitude": -79.3623,
            "AutonomousSystemNumber": 14061,
            "Latitude": 43.6547,
            "Subdivision": "Ontario",
            "CausedBy": {
                "Status": "ACTIVE",
                "Networks": [
                    {
                        "AutonomousSystemNumber": 16509,
                        "NetworkName": "Amazon.com"
                    }
                ],
                "NetworkEventType": "AWS"
            },
            "NetworkName": "DIGITALOCEAN-ASN"
        },
        {
            "City": "Shenzhen",
            "PercentOfClientLocationImpacted": 4.07,
            "PercentOfTotalTraffic": 0.61,
            "Country": "China",
            "Longitude": 114.0683,
            "AutonomousSystemNumber": 37963,
            "Latitude": 22.5455,
            "Subdivision": "Guangdong",
            "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
```

```
        },
        {
            "City": "Hangzhou",
            "PercentOfClientLocationImpacted": 2.88,
            "PercentOfTotalTraffic": 0.7799999999999999,
            "Country": "China",
            "Longitude": 120.1612,
            "AutonomousSystemNumber": 37963,
            "Latitude": 30.2994,
            "Subdivision": "Zhejiang",
            "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
        }
    ],
    "StartedAt": "2022-06-20T01:34:20Z",
    "ImpactType": "PERFORMANCE",
    "PercentOfTotalTrafficImpacted": 1.15
}
```

모니터 목록 보기

CLI를 사용하여 계정의 모든 모니터 목록을 보려면 `list-monitors` 명령을 실행합니다.

```
aws internetmonitor list-monitors
```

```
{
    "Monitors": [
        {
            "MonitorName": "TestMonitor",
            "ProcessingStatus": "OK",
            "Status": "ACTIVE"
        }
    ],
    "NextToken": " zase12"
}
```

모니터 편집

CLI를 사용하여 모니터에 대한 정보를 업데이트하려면 `update-monitor` 명령을 사용하고 업데이트할 모니터의 이름을 지정합니다. 모니터링할 트래픽의 백분율, 모니터링할 최대 도시-네트워크 수 한도를 업데이트하고, Internet Monitor가 트래픽 모니터에 사용하는 리소스를 추가 또는 제거하고, 모니터

상태를 ACTIVE에서 INACTIVE 또는 그 반대로 변경할 수 있습니다. 모니터 이름은 변경할 수 없습니다.

update-monitor 호출에 대한 응답은 MonitorArn 및 Status만 반환합니다.

다음 예에서는 update-monitor 명령을 사용하여 모니터링할 최대 도시-네트워크 수를 50000으로 변경하는 방법을 보여 줍니다.

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" --max-city-networks-to-
monitor 50000
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": " ACTIVE "
}
```

다음 예제에서는 리소스를 추가하고 제거하는 방법을 보여줍니다.

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" \
    --resources-to-add "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889" \
    --resources-to-remove "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-2222444455556666"
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": "ACTIVE"
}
```

다음 예제에서는 update-monitor 명령을 사용하여 모니터 상태를 INACTIVE로 변경하는 방법을 보여줍니다.

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" --status "INACTIVE"
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": "INACTIVE"
}
```

모니터 삭제

delete-monitor 명령을 사용하여 CLI로 모니터를 삭제할 수 있습니다. 먼저 모니터를 비활성으로 설정해야 합니다. 이렇게 하려면 update-monitor 명령을 사용하여 상태를 INACTIVE로 변경합니다. get-monitor 명령을 사용하고 상태를 확인하여 모니터가 비활성 상태인지 확인합니다.

모니터 상태가 INACTIVE이면 CLI를 사용하여 delete-monitor 명령을 실행하여 모니터를 삭제할 수 있습니다. 성공적인 delete-monitor 호출에 대한 응답은 비어 있습니다.

```
aws internetmonitor delete-monitor --monitor-name "TestMonitor"
```

```
{}
```

Internet Monitor 대시보드로 모니터링 및 최적화

이 섹션의 정보에서는 AWS 애플리케이션의 인터넷 트래픽 및 설정에 대한 시각화 및 인사이트를 얻기 위해 Amazon CloudWatch Internet Monitor 대시보드에서 정보를 필터링하고 보는 방법을 설명합니다.

애플리케이션의 인터넷 성능 및 가용성을 모니터하는 모니터를 생성한 후 Amazon CloudWatch Internet Monitor는 클라이언트 위치-네트워크(도시-네트워크) 쌍에 대한 인터넷 측정값이 포함된 CloudWatch 로그를 게시하고 애플리케이션, 각 AWS 리전 리전 및 엣지 로케이션에 대한 트래픽에 대해 집계된 CloudWatch 지표를 게시합니다. 여러 가지 방법으로 Internet Monitor의 이 정보에서 작업 지향 제안을 필터링하고, 탐색하고, 얻을 수 있습니다.

시작하려면 CloudWatch 콘솔의 애플리케이션 모니터링에서 인터넷 모니터를 선택합니다.

이 섹션에서는 주로 AWS Management Console을 사용하여 Internet Monitor 지표를 필터링하고 보는 방법에 대해 설명합니다. 또는 AWS CLI 또는 SDK에서 Internet Monitor API 작업을 사용하여 CloudWatch 로그 파일에 저장된 Internet Monitor 이벤트로 직접 작업할 수 있습니다. 자세한 내용은 모니터 및 측정값 정보 사용을 참조하세요. API 작업 사용에 대한 자세한 내용은 Amazon CloudWatch Internet Monitor에서 CLI를 사용하는 예 및 Amazon CloudWatch Internet Monitor API Reference를 참조하세요.

Internet Monitor 대시보드에는 세 개의 탭이 있습니다.

- Overview(개요) 탭에서 애플리케이션에 대한 현재 및 기록 성능 및 가용성 정보와 클라이언트 위치에 영향을 미치는 상태 이벤트를 볼 수 있습니다.
- 다음 탭인 기록 탐색기에서 위치, ASN, 날짜 등으로 필터링하고 그래프를 사용하여 시간 경과에 따른 인터넷 트래픽 지표를 시각화할 수 있습니다.

- 트래픽 인사이트 탭에서 사용자 지정 가능한 여러 방법으로 요약된 주요 모니터 트래픽에 대한 정보를 보는 것 외에도 다양한 위치 및 ASN 쌍의 성능을 개선하기 위해 최적화된 설정에 대한 제안을 얻을 수 있습니다. Internet Monitor는 트래픽을 라우팅하는 방법이나 사용하는 AWS 리소스를 변경할 때 트래픽 패턴과 과거 성능을 기준으로 애플리케이션의 성능 향상을 예측합니다. 또한 모니터링에 대해 선택한 애플리케이션 트래픽 백분율에 따라 모니터링 범위에 포함된 도시-네트워크의 수를 비교하는 그래프를 볼 수 있습니다.

또한 Internet Monitor는 트래픽에 대한 측정값과 함께 로그 파일을 생성하고 게시하므로 콘솔의 다른 CloudWatch 도구를 사용하여 CloudWatch Contributor Insights, CloudWatch Metrics 및 CloudWatch Logs Insights를 포함하여 Internet Monitor에서 게시한 데이터를 추가로 시각화할 수 있습니다. 자세한 내용은 CloudWatch 도구 및 Internet Monitor 쿼리 인터페이스로 데이터 탐색 단원을 참조하십시오.

다음 섹션에서 Internet Monitor를 사용하여 성능 및 가용성 측정값을 탐색하는 방법을 알아보세요.

주제

- Amazon CloudWatch Internet Monitor에서 실시간 성능 및 가용성 추적(Overview(개요) 탭)
- Amazon CloudWatch Internet Monitor에서 기록 데이터 필터링 및 보기(기록 탐색기 탭)
- Amazon CloudWatch Internet Monitor에서 애플리케이션 성능 개선을 위한 인사이트 얻기(트래픽 인사이트 탭)

Amazon CloudWatch Internet Monitor에서 실시간 성능 및 가용성 추적(Overview(개요) 탭)

CloudWatch 콘솔의 Internet Monitor 아래에 있는 개요 탭을 사용하여 모니터가 추적하는 트래픽의 성능 및 가용성에 대한 높은 수준의 보기를 확인하세요. 또한 이 탭에는 애플리케이션의 글로벌 트래픽과 상태 이벤트의 위치 및 영향을 시각화하는 데 도움이 되는 트래픽 클러스터가 포함된 인터넷 트래픽 개요 맵이 표시됩니다.

상태 점수

상태 점수 그래프는 글로벌 트래픽에 대한 성능 및 가용성 정보를 보여 줍니다. AWS에서는 다양한 ASN 및 AWS 서비스에 대한 지리적 위치 간 네트워크 트래픽 관련 인터넷 성능 및 가용성에 대한 상당한 과거 데이터를 보유하고 있습니다. Internet Monitor는 AWS가 글로벌 네트워킹 공간에서 수집한 연결 데이터를 사용하여 인터넷 트래픽의 성능 및 가용성에 대한 기준선을 계산합니다. 이는 AWS에서 인터넷 가동 시간과 가용성을 모니터링하는 데 사용하는 것과 동일한 데이터입니다.

이러한 측정값을 기준선으로 사용하여 Internet Monitor는 기준선에 비해 언제 애플리케이션의 성능과 가용성이 저하되는지 탐지할 수 있습니다. 이러한 저하를 더 쉽게 확인할 수 있도록 해당 정보를 성능 점수 및 가용성 점수로 보고합니다. 자세한 내용은 CloudWatch 도구 및 Internet Monitor 쿼리 인터페이스로 데이터 탐색 단원을 참조하십시오.

상태 점수 그래프에는 선택한 기간 동안 발생한 상태 이벤트가 포함됩니다. 상태 이벤트가 있으면 그래프의 성능 또는 가용성 라인에서 저하를 확인할 수 있습니다. 이벤트를 선택하면 이벤트가 지속된 기간을 보여주는 날짜 및 시간 정보와 함께 추가 세부 정보와 밴드가 그래프에 표시됩니다.

각 데이터 포인트에 대한 로그 파일에 직접 액세스하여 이러한 지표를 볼 수도 있습니다. Actions(작업) 메뉴에서 View CloudWatch Logs(CloudWatch Logs 보기)를 선택합니다.

인터넷 트래픽 개요

인터넷 트래픽 개요 맵은 사용자가 애플리케이션에 액세스하는 위치 및 ASN과 관련된 인터넷 트래픽 및 상태 이벤트를 표시합니다. 맵에서 회색으로 표시된 국가는 애플리케이션에 대한 트래픽이 포함된 국가입니다.

맵의 각 원은 사용자가 선택한 기간 내 영역의 상태 이벤트를 나타냅니다. Internet Monitor는 AWS에서 호스트되는 리소스 중 하나와 사용자가 애플리케이션에 액세스하는 도시-네트워크 간의 연결과 관련하여 특정 임계값에서 문제를 탐지하면 상태 이벤트를 생성합니다. 맵에서 원을 선택하면 해당 위치의 상태 이벤트에 대한 추가 세부 정보가 표시됩니다. 또한 상태 이벤트가 있는 클러스터의 경우 맵 아래의 Health events(상태 이벤트) 테이블에서 자세한 정보를 볼 수 있습니다.

Internet Monitor는 이벤트가 애플리케이션에 중대한 글로벌 영향을 미친다고 판단할 때 모니터에 상태 이벤트를 생성합니다. 선택한 기간 동안 클라이언트 위치의 트래픽에 미치는 영향 임계값을 초과하는 상태 이벤트가 없는 경우 맵이 비어 있습니다. 자세한 내용은 Internet Monitor가 상태 이벤트를 생성하고 해결하는 경우를 참조하세요.

상태 이벤트 임계값 변경

Internet Monitor가 애플리케이션의 상태 이벤트를 생성하는 방법 및 시기와 관련된 몇 가지 옵션을 구성할 수 있습니다. 변경하려면 임계값 업데이트를 선택합니다.

Internet Monitor를 트리거하여 상태 이벤트를 생성하는 전체 임계값을 변경할 수 있습니다. 성능 점수 및 가용성 점수 모두에 대한 기본 상태 이벤트 임계값은 95%입니다. 즉, 애플리케이션의 전체 성능 또는 가용성 점수가 95% 이하로 떨어지면 Internet Monitor에서 상태 이벤트를 생성합니다. 전체 임계값의 경우, 상태 이벤트는 하나의 큰 문제 또는 여러 개의 작은 문제가 결합되어 트리거될 수 있습니다.

또한 전체 영향 수준의 백분율과 결합된 로컬, 즉 도시-네트워크 임계값을 변경할 수 있으며, 이 임계값이 합쳐지면 상태 이벤트가 트리거됩니다. 하나 이상의 도시-네트워크(위치 및 ASN, 일반적으

로 ISP)에 대해 점수가 임계값 아래로 떨어질 때 상태 이벤트를 생성하는 임계값을 설정하면 예를 들어 트래픽이 적은 위치에서 언제 문제가 발생하는지에 대한 인사이트를 얻을 수 있습니다.

추가 로컬 임계값 옵션은 가용성 또는 성능 점수에 대한 로컬 임계값과 함께 작동합니다. 두 번째 요소는 Internet Monitor가 로컬 임계값에 따라 상태 이벤트를 생성하기 전에 영향을 받아야 하는 전체 트래픽의 백분율입니다.

전체 트래픽 및 로컬 트래픽에 대한 임계값 옵션을 구성하여 애플리케이션 사용량과 필요에 맞게 상태 이벤트 생성 빈도를 미세 조정할 수 있습니다. 로컬 임계값을 더 낮게 설정하면 일반적으로 애플리케이션 및 설정한 다른 임계값 구성 값에 따라 더 많은 상태 이벤트가 생성된다는 점에 유의하세요.

요약하면 다음과 같은 방법으로 성능 점수, 가용성 점수 또는 두 가지 모두에 대한 상태 이벤트 임계값을 구성할 수 있습니다.

- 상태 이벤트 트리거를 위한 다양한 글로벌 임계값을 선택하세요.
- 상태 이벤트 트리거를 위한 다양한 로컬 임계값을 선택하세요. 이 옵션을 사용하면 Internet Monitor가 이벤트를 생성하기 전에 초과해야 하는 전체 애플리케이션에 미치는 영향의 백분율을 변경할 수도 있습니다.
- 로컬 임계값에 기반을 둔 상태 이벤트 트리거를 끄거나 로컬 임계값 옵션을 사용하도록 설정합니다.

성능 점수, 가용성 점수 또는 둘 다에 대한 옵션을 구성할 수도 있습니다. 여러 옵션을 조합하여 구성하거나 옵션 중 하나만 구성할 수 있습니다.

성능 점수, 가용성 점수 또는 둘 다에 대한 임계값 및 기타 구성 옵션을 업데이트하려면 다음과 같이 하세요.

임계값 구성 옵션을 변경하려면

1. AWS Management Console에서 CloudWatch로 이동한 다음 왼쪽 탐색 창에서 Internet Monitor를 선택합니다.
2. 개요 탭의 상태 이벤트 타임라인 섹션에서 임계값 업데이트를 선택합니다.
3. 대화 상자 페이지가 열리면 Internet Monitor가 상태 이벤트를 생성하도록 트리거하는 임계값 및 기타 옵션에 대해 원하는 새 값과 옵션을 선택합니다. 다음 중 무엇이든 수행할 수 있습니다.
   - 가용성 점수 임계값, 성능 점수 임계값 또는 둘 다에 대해 새 값을 선택합니다.

     각 설정 섹션의 그래프에는 애플리케이션의 현재 임계값 설정과 가용성 또는 성능에 대한 실제 최근 상태 이벤트 점수가 표시됩니다. 일반적인 값을 확인하면 임계값을 변경할 수 있는 값에 대한 아이디어를 얻을 수 있습니다.

팁: 그래프를 더 크게 보고 기간을 변경하려면 그래프의 오른쪽 상단 모서리에 있는 확장기를 선택하세요.

- 가용성 또는 성능 또는 둘 다에 대한 로컬 임계값을 설정하거나 해제하도록 선택합니다. 옵션을 사용하도록 설정하면 Internet Monitor에서 상태 이벤트를 생성할 시점에 대한 임계값 및 영향 수준을 설정할 수 있습니다.

4. 임계값 옵션을 구성한 후에는 상태 이벤트 임계값 업데이트를 선택하여 업데이트를 저장합니다.

상태 이벤트가 작동하는 방식에 대해 자세히 알아보려면 When Internet Monitor creates and resolves health events를 참조하세요.

상태 이벤트 테이블

상태 이벤트 테이블에는 상태 이벤트의 영향을 받은 클라이언트 위치가 이벤트에 대한 정보와 함께 나열됩니다. 테이블에는 다음과 같은 열이 포함되어 있습니다.

	설명
클라이언트 위치	이벤트의 영향을 받아 지연 시간이 증가하거나 가용성이 감소한 최종 사용자의 위치입니다. Internet Monitor의 클라이언트 위치 정확도에 대해 자세히 알아보려면 Internet Monitor의 지리적 위치 정보 및 정확도를 참조하세요.
트래픽 영향	지연 시간 증가 또는 가용성 감소에서 이벤트로 인해 발생하는 영향의 정도입니다. 지연 시간의 경우 이 클라이언트 네트워크를 사용하여 이 클라이언트 위치에서 이 AWS 위치로 전송되는 일반적인 트래픽 성능과 비교하여 이벤트 중에 증가한 지연 시간의 백분율입니다.
클라이언트 네트워크	트래픽이 통과한 네트워크입니다. 일반적으로 이는 네트워크 트래픽에 대한 인터넷 서비스 제공업체(ISP) 또는 Autonomous System Number(ASN)입니다.

	설명
AWS 위치	네트워크 트래픽의 AWS 위치로, AWS 리전 또는 엣지 로케이션일 수 있습니다.
영향 유형	상태 이벤트에 대한 영향의 유형입니다. 상태 이벤트는 일반적으로 지연 시간 증가(성능 문제) 또는 도달 가능성(가용성 문제)으로 인해 발생합니다. 영향 유형을 클릭하여 장애의 원인을 확인할 수도 있습니다. 가능하다면 Internet Monitor는 상태 이벤트의 원인을 분석하여 AWS 또는 ASN(인터넷 서비스 제공업체)에 의해 발생한 것인지 확인합니다. 이 분석은 이벤트가 해결된 후에도 계속됩니다. Internet Monitor는 최대 1시간 동안 새로운 정보로 이벤트를 업데이트할 수 있습니다.

상태 이벤트 테이블에서 클라이언트 위치 중 하나를 선택하면 해당 위치의 상태 이벤트에 대한 자세한 내용을 볼 수 있습니다. 예를 들어 이벤트가 시작된 시간, 종료된 시간 및 로컬 트래픽 영향을 확인할 수 있습니다.

네트워크 경로 시각화

완료된 장애 분석은 네트워크 경로 시각화 아래에 전체 네트워크 경로가 표시됩니다. 전체 경로는 AWS 위치와 클라이언트 사이의 상태 이벤트에 대한 애플리케이션의 네트워크 경로를 따라 클라이언트-위치 쌍에 대한 각 노드를 표시합니다.

Internet Monitor에서 장애의 원인이 확인되면 빨간색 점선으로 표시됩니다. 장애는 일반적으로 인터넷 서비스 제공업체(ISP)와 같은 ASN이나 AWS에 의해 발생할 수 있습니다. 장애의 원인이 여러 가지인 경우 여러 노드에 원이 표시됩니다.

Amazon CloudWatch Internet Monitor에서 기록 데이터 필터링 및 보기(기록 탐색기 탭)

CloudWatch 콘솔의 Internet Monitor 아래에 있는 기록 탐색기 탭을 사용하여 CloudWatch 로그에 있는 애플리케이션의 데이터를 필터링하고 볼 수 있습니다. Internet Monitor는 가용성, 성능, 전송된 모니터

바이트(또는 클라이언트 연결 수, WorkSpaces 디렉터리에만 해당), 모니터되는 AWS 리전의 도시-네트워크의 왕복 시간에 대한 측정값을 애플리케이션과 관련된 CloudWatch 로그에 게시합니다.

> **Note**
>
> Internet Monitor는 각 모니터에 트래픽을 전송하는 상위 500개(트래픽 양 기준) 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 공급업체(ISP))에 대해 5분마다 인터넷 측정값을 CloudWatch 로그에 게시합니다. 원하는 경우, 모니터링되는 모든 도시-네트워크(최대 500,000개의 도시-네트워크 서비스 제한)에 대한 인터넷 측정값을 Amazon S3 버킷에 게시하도록 선택할 수 있습니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에서 Amazon S3에 인터넷 측정값 게시 단원을 참조하십시오.

애플리케이션의 데이터 탐색을 시작하려면 기간을 선택합니다. 그런 다음 특정 지리적 위치(예: 도시) 및 선택 사항으로 다른 필터를 선택합니다. Internet Monitor는 애플리케이션 트래픽의 경우 도시-네트워크에 대해 게시한 인터넷 측정 로그의 데이터에 필터를 적용합니다. 그런 다음 시간 경과에 따른 애플리케이션의 성능 점수, 가용성 점수, 전송된 모니터링된 바이트(VPC, Network Load Balancer, CloudFront 배포의 경우) 또는 클라이언트 연결 수(WorkSpaces 디렉터리의 경우), 왕복 시간(RTT)을 보여주는 데이터 그래프를 볼 수 있습니다.

그래프 아래의 All events(모든 이벤트) 테이블에는 애플리케이션 트래픽에 대해 필터가 반환하는 상태 이벤트가 각 이벤트에 대한 정보와 함께 표시됩니다. 여기에는 다음 열이 포함됩니다.

	설명
이벤트 시작	상태 이벤트가 시작된 시간입니다.
상태 표시기	이벤트가 아직 활성 상태인지 아니면 해결되었는지입니다.
클라이언트 위치	이벤트의 영향을 받아 지연 시간이 증가하거나 성능이 감소한 최종 사용자의 위치입니다. Internet Monitor의 클라이언트 위치 정확도에 대해 자세히 알아보려면 Internet Monitor의 지리적 위치 정보 및 정확도를 참조하세요.
트래픽 영향	상태 이벤트의 위치에 대한 이벤트의 가중치 영향입니다. 예를 들어 클라이언트 위치에서 클라

	설명
	이언트 ASN(일반적으로 인터넷 서비스 제공업체(ISP)를 통해 AWS 위치로 전송되는 트래픽의 일반적인 성능과 비교하여 지연 시간에 미치는 영향이 있습니다. 마찬가지로 가용성에 영향을 미치는 이벤트의 경우 클라이언트 ASN을 통해 AWS 위치에 대한 클라이언트 위치의 일반적인 가용성과 비교하여 가용성에 미치는 영향을 확인할 수 있습니다.
이벤트 기간	이벤트가 지속된 기간입니다. 상태 이벤트가 애플리케이션 클라이언트 위치의 총 5% 이상에 더 이상 영향을 미치지 않으면 Internet Monitor가 상태 이벤트를 종료합니다.
클라이언트 ISP	네트워크 트래픽의 통신 사업자인 ASN(일반적으로 인터넷 서비스 제공업체(ISP))입니다.
서비스 위치	이 서비스 위치는 네트워크 트래픽이 발생한 곳으로 AWS 리전 또는 인터넷 엣지 로케이션일 수 있습니다.

또는 각 데이터 포인트에 대한 로그에 직접 액세스하여 애플리케이션의 측정값을 확인할 수 있습니다. Actions(작업) 메뉴에서 View CloudWatch Logs(CloudWatch Logs 보기)를 선택합니다. 측정 이벤트가 생성되면 계정에 게시되므로 이를 기반으로 다른 CloudWatch 대시보드나 경보도 만들 수 있습니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에서 애플리케이션 성능 개선을 위한 인사이트 얻기(트래픽 인사이트 탭) 및 Amazon CloudWatch Internet Monitor를 사용하여 경보 생성 섹션을 참조하세요.

Internet Monitor 측정값 및 지표를 탐색 및 분석하고 이를 기반으로 대시보드 및 경보를 생성하는 것 외에도 Internet Monitor를 사용하여 애플리케이션의 성능을 개선할 방법을 파악할 수 있습니다. Traffic insights(트래픽 인사이트) 탭에는 옵션을 탐색하는 데 도움이 되는 여러 가지 방법이 있습니다. 자세한 내용은 트래픽 인사이트 탭에서 트래픽 최적화 제안을 참조하세요. 또한 Internet Monitor사용 사례 장에서 구체적인 예를 확인할 수 있습니다.

Amazon CloudWatch Internet Monitor에서 애플리케이션 성능 개선을 위한 인사이트 얻기(트래픽 인사이트 탭)

CloudWatch 콘솔의 Internet Monitor 아래에 있는 트래픽 인사이트 탭을 사용하여 애플리케이션의 상위 트래픽(볼륨 기준)에 대한 요약 정보를 확인하세요. 여러 가지 방법으로 애플리케이션 트래픽을 필터링하고 정렬할 수 있습니다. 그런 다음 아래로 스크롤하여 애플리케이션에 대한 다양한 설정 조합을 선택하여 Internet Monitor가 가장 빠른 첫 바이트 전송 시간(TTFB) 성능을 얻기 위해 제안하는 최상의 대안을 확인하세요.

Internet Monitor는 각 모니터에 트래픽을 전송하는 상위 500개(트래픽 양 기준) 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 공급업체(ISP))에 대해 5분마다 인터넷 측정값을 CloudWatch 로그에 게시합니다. 원하는 경우, 모니터링되는 모든 도시-네트워크(최대 500,000개의 도시-네트워크 서비스 제한)에 대한 인터넷 측정값을 Amazon S3 버킷에 게시하도록 선택할 수 있습니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에서 Amazon S3에 인터넷 측정값 게시 단원을 참조하십시오.

주요 트래픽 요약

먼저 특정 기간 동안 애플리케이션의 전체 트래픽 및 성능에 대한 개략적인 요약을 클라이언트 위치별로 필터링하여 볼 수 있습니다. 또한 트래픽 양에 따라 상위(또는 하위) 클라이언트 위치에 대한 애플리케이션의 성능을 여러 가지 방법으로 필터링 및 정렬하여 확인할 수 있습니다. 예를 들어 세분성(도시, 구, 국가 또는 대도시 지역), 총 트래픽, 평균 첫 바이트 시간(TTFB), 기타 요인별로 정렬할 수 있습니다.

Internet Monitor의 클라이언트 위치 정확도에 대해 자세히 알아보려면 Internet Monitor의 지리적 위치 정보 및 정확도를 참조하세요.

> **Note**
>
> 사용하는 필터는 전체 페이지에 적용되므로 총 트래픽에 대한 요약 그래프 및 정보에 포함되는 도시-네트워크와 다음에 나오는 트래픽 최적화 제안 섹션에 포함되는 도시-네트워크에도 영향을 미칩니다.

트래픽 최적화 제안

트래픽 최적화 제안 섹션에는 트래픽에 대해 모니터되는 도시-네트워크(위치 및 ASN, 인터넷 서비스 제공업체)의 필터링된 세트와 각 네트워크에 대한 총 클라이언트 트래픽이 표시됩니다. 테이블의 항목은 페이지 상단의 트래픽 인사이트에서 애플리케이션 트래픽에 대해 선택한 필터를 기

반으로 합니다. 기본값은 트래픽 기준 상위 10개 도시입니다. 각 고유한 도시-네트워크 쌍에 대한 항목이 있기 때문에 일반적으로 테이블에 10개 이상의 행이 표시됩니다. 즉, 클라이언트가 애플리케이션에 액세스하는 위치(도시)와 ASN(네트워크 제공업체)의 조합(예: 미국 텍사스주 댈러스와 Comcast)에 대해 각각 하나의 행이 있습니다.

> **Note**
>
> 모니터하는 모든 도시-네트워크에 대한 트래픽 최적화 제안을 보려면 CloudWatch Insights에서 직접 쿼리를 실행하세요. 이 페이지에서 도시-네트워크 목록을 제한하는 지리적 세분성 필터를 포함하지 않는 샘플 쿼리는 Amazon CloudWatch Internet Monitor와 함께 CloudWatch Logs Insights 사용 섹션을 참조하세요.

이 섹션에서는 Amazon EC2, CloudFront 또는 둘 다 등 다양한 옵션을 선택합니다. 이를 통해 현재 첫 번째 바이트까지의 시간(TTFB)과 비교하여 다양한 AWS 리전의 해당 서비스와 함께 애플리케이션을 사용할 때 클라이언트에 대한 예상 평균 TTFB 값이 무엇인지 확인할 수 있습니다. TTFB 계산에 대한 자세한 내용은 TTFB 및 지연 시간에 대한 AWS 계산을 참조하세요.

다양한 옵션을 선택한 다음 표에서 결과를 확인하면 클라이언트 성능을 향상할 설정 및 배포 계획을 시작할 수 있습니다. 데이터를 표시할 수 없는 경우 열에 값 대신 대시(-)가 표시될 수 있습니다. 성능을 향상하는 방법에 대한 구체적인 예제를 검토하려면 Using Amazon CloudWatch Internet Monitor for a Better Gaming Experience를 참조하세요.

예를 들어 우선 특정 도시-네트워크(클라이언트 위치 및 ASN 쌍)에 대해 EC2나 CloudFront 옵션 중 하나 또는 둘 다를 선택하여 실험해 보세요. 테이블에 나열된 각 도시-네트워크에 대해 Internet Monitor는 현재 설정과 비교하여 해당 옵션을 사용한 (특정 AWS 리전을 통한) 트래픽 라우팅 선택을 기반으로 TTFB의 잠재적인 성능 향상을 보여줍니다. (완전성을 위해 이 표에는 이미 최적화된 경로도 포함되어 있습니다.) 예를 들어 `us-east-1`을 통해 EC2 라우팅을 사용하는 경우 예상 평균 TTFB가 50ms인 반면, `us-west-2`를 통해 EC2 라우팅을 사용하는 경우 TTFB가 100ms인 현재 설정과 비교하여 예상 평균 TTFB가 50ms로 표시될 수 있습니다. 따라서 `us-west-2`를 통해 라우팅하는 것을 고려할 수 있습니다.

다른 예로, EC2를 선택한 후 한 클라이언트 위치와 ASN에 대해 측정 가능한 성능 차이가 없음을 확인한 다음 동일한 리전으로 CloudFront를 선택하면 TTFB가 다소 낮아진다는 것을 알 수 있습니다. 이는 애플리케이션 앞에 CloudFront 배포를 추가하면 성능이 향상될 수 있으며 이 클라이언트 위치 및 ASN에 대해 시도해 볼 가치가 있음을 시사합니다.

CloudWatch 도구 및 Internet Monitor 쿼리 인터페이스로 데이터 탐색

Amazon CloudWatch Internet Monitor 대시보드를 사용하여 애플리케이션에 대한 성능 및 가용성을 시각화하는 것 외에도 Internet Monitor가 생성하는 데이터를 심층 분석하는 데 사용할 수 있는 몇 가지 방법이 있습니다. 이러한 방법에는 CloudWatch 로그 파일에 저장된 Internet Monitor 데이터와 함께 CloudWatch 도구를 사용하고 Internet Monitor 쿼리 인터페이스를 사용하는 것이 포함됩니다. 사용할 수 있는 도구로 CloudWatch Logs Insights, CloudWatch Metrics, CloudWatch Contributor Insights, Amazon Athena 등이 있습니다. 필요에 따라 대시보드뿐만 아니라 이러한 도구 일부 또는 전체를 사용하여 Internet Monitor 데이터를 탐색할 수 있습니다.

Internet Monitor는 애플리케이션 및 각 AWS 리전으로 전송되는 트래픽에 대한 CloudWatch 지표를 집계하고 총 트래픽 영향, 가용성, 왕복 시간과 같은 데이터를 포함합니다. 이 데이터는 CloudWatch Logs에 게시되며 Internet Monitor 쿼리 인터페이스에서도 사용할 수 있습니다. 지리적 세분성과 각 데이터에 대해 탐색할 수 있는 기타 측면에 대한 세부 정보는 다양합니다.

Amazon CloudWatch Internet Monitor는 5분 간격으로 모니터용 데이터를 게시한 다음 여러 가지 방법으로 데이터를 사용할 수 있도록 합니다. 다음 표에는 Internet Monitor 데이터에 액세스하는 시나리오가 나열되어 있으며 각 시나리오에 대해 수집되는 데이터의 특징이 설명되어 있습니다.

기능	CloudWatch Logs	S3로 내보내기	쿼리 인터페이스	CloudWatch 대시보드
기본적으로 사용됩니다.	예	아니요	예	예
데이터가 수집되는 도시-네트워크 수	상위 500개(아래 참고 참조)	모두	모두	모두
데이터 보존	사용자 제어	사용자 제어	30일	30일
데이터 수집 대상 지리적 세부 정보	전체(도시-네트워크, 메트로+네트워크, 세분+네트워크, 국가+네트워크)	도시-네트워크	전체(도시-네트워크, 메트로+네트워크, 세분+네트워크, 국가+네트워크)	전체(도시-네트워크, 메트로+네트워크, 세분+네트워크, 국가+네트워크)

기능	CloudWatch Logs	S3로 내보내기	쿼리 인터페이스	CloudWatch 대시보드
데이터 쿼리 및 필터링 방법	Amazon CloudWatch Internet Monitor와 함께 CloudWatch Logs Insights 사용	Amazon Athena를 사용하여 Amazon S3 로그 파일에서 인터넷 측정값 쿼리	Amazon CloudWatch Internet Monitor 쿼리 인터페이스 사용	Internet Monitor 대시보드로 모니터링 및 최적화

참고: 도시-네트워크의 경우 상위 500개 측정치, 메트로+네트워크는 상위 250개, 세분+네트워크는 상위 100개, 국가+네트워크는 상위 50개의 측정값이 캡처됩니다.

이 장에서는 CloudWatch 도구 또는 Internet Monitor 쿼리 인터페이스를 사용하여 데이터를 쿼리하고 탐색하는 방법을 각 방법의 예시와 함께 설명합니다.

내용

Amazon CloudWatch Internet Monitor와 함께 CloudWatch Logs Insights 사용

Amazon CloudWatch Internet Monitor는 가용성 및 왕복 시간에 대한 세분화된 측정값을 CloudWatch 로그에 게시하며, CloudWatch 로그 인사이트 쿼리를 사용하여 특정 도시 또는 지역(클라이언트 위치), 클라이언트 ASN(ISP) 및 AWS 소스 위치에 대한 로그의 하위 세트를 필터링할 수 있습니다.

Internet Monitor의 클라이언트 위치 정확도에 대해 자세히 알아보려면 Internet Monitor의 지리적 위치 정보 및 정확도를 참조하세요.

이 섹션의 예를 통해 자체 애플리케이션 트래픽 측정 및 지표에 대해 자세히 알아볼 수 있는 CloudWatch 로그 인사이트 쿼리를 생성할 수 있습니다. CloudWatch Logs Insights에서 이 예를 사용하는 경우 *monitorName*을 자신의 모니터 이름으로 바꾸세요.

트래픽 최적화 제안 보기

Internet Monitor의 트래픽 인사이트 탭에서 위치별로 필터링된 트래픽 최적화 제안을 볼 수 있습니다. 해당 탭의 트래픽 최적화 제안 섹션에 표시되는 것과 동일한 정보를 보려면 위치 세분성 필터를 사용하지 않고 다음 CloudWatch Logs Insights 쿼리를 사용하세요.

1. AWS Management Console에서 CloudWatch Logs Insights로 이동합니다.
2. Log Group(로그 그룹)에서 `/aws/internet-monitor/`*`monitorName`*`/byCity`와 `/aws/internet-monitor/`*`monitorName`*`/byCountry`를 선택한 다음 시간 범위를 지정합니다.
3. 다음 쿼리를 추가하고 실행합니다.

```
fields @timestamp,
clientLocation.city as @city, clientLocation.subdivision as @subdivision,
 clientLocation.country as @country,
`trafficInsights.timeToFirstByte.currentExperience.serviceName` as @serviceNameField,
concat(@serviceNameField, ` (`, `serviceLocation`, `)`) as @currentExperienceField,
concat(`trafficInsights.timeToFirstByte.ec2.serviceName`, ` (`,
 `trafficInsights.timeToFirstByte.ec2.serviceLocation`, `)`) as @ec2Field,
`trafficInsights.timeToFirstByte.cloudfront.serviceName` as @cloudfrontField,
concat(`clientLocation.networkName`, ` (AS`, `clientLocation.asn`, `)`) as @networkName
| filter ispresent(`trafficInsights.timeToFirstByte.currentExperience.value`)
| stats avg(`trafficInsights.timeToFirstByte.currentExperience.value`) as @averageTTFB,
avg(`trafficInsights.timeToFirstByte.ec2.value`) as @ec2TTFB,
avg(`trafficInsights.timeToFirstByte.cloudfront.value`) as @cloudfrontTTFB,
sum(`bytesIn` + `bytesOut`) as @totalBytes,
latest(@ec2Field) as @ec2,
latest(@currentExperienceField) as @currentExperience,
latest(@cloudfrontField) as @cloudfront,
count(*) by @networkName, @city, @subdivision, @country
| display @city, @subdivision, @country, @networkName, @totalBytes, @currentExperience,
 @averageTTFB, @ec2, @ec2TTFB, @cloudfront, @cloudfrontTTFB
| sort @totalBytes desc
```

인터넷 가용성 및 RTT(p50, p90, p95) 보기

트래픽에 대한 인터넷 가용성 및 왕복 시간(p50, p90, p95)을 보려면 다음 CloudWatch Logs Insights 쿼리를 사용하세요.

최종 사용자 지역: 미국 일리노이주 시카고

최종 사용자 네트워크(ASN): AS7018

AWS 서비스 위치: 미국 동부(버지니아 북부) 리전

다음 방법으로 로그를 볼 수 있습니다.

1. AWS Management Console에서 CloudWatch Logs Insights로 이동합니다.
2. Log Group(로그 그룹)에서 `/aws/internet-monitor/`*`monitorName`*`/byCity`와 `/aws/internet-monitor/`*`monitorName`*`/byCountry`를 선택한 다음 시간 범위를 지정합니다.
3. 다음 쿼리를 추가하고 실행합니다.

이 쿼리는 선택한 기간 동안 일리노이주 시카고에 있는 AS7018에서 미국 동부(버지니아 북부) 리전으로 연결한 사용자의 모든 성능 데이터를 반환합니다.

```
fields @timestamp,
internetHealth.availability.experienceScore as availabilityExperienceScore,
internetHealth.availability.percentageOfTotalTrafficImpacted as
 percentageOfTotalTrafficImpacted,
internetHealth.performance.experienceScore as performanceExperienceScore,
internetHealth.performance.roundTripTime.p50 as roundTripTimep50,
internetHealth.performance.roundTripTime.p90 as roundTripTimep90,
internetHealth.performance.roundTripTime.p95 as roundTripTimep95
 | filter clientLocation.country == `United States`
 and clientLocation.city == `Chicago`
 and serviceLocation == `us-east-1`
 and clientLocation.asn == 7018
```

자세한 내용은 CloudWatch Logs Insights를 사용한 로그 분석을 참조하세요.

Amazon CloudWatch Internet Monitor와 함께 Contributor Insights 사용

CloudWatch Contributor Insights는 애플리케이션의 상위 클라이언트 위치 및 네트워크(ASN 또는 인터넷 서비스 공급업체)를 식별하는 데 도움이 됩니다. 다음 샘플 Contributor Insights 규칙으로 Amazon CloudWatch Internet Monitor에서 유용한 규칙을 시작합니다. 자세한 내용은 Contributor Insights 규칙 생성 단원을 참조하십시오.

Internet Monitor의 클라이언트 위치 정확도에 대해 자세히 알아보려면 Internet Monitor의 지리적 위치 정보 및 정확도를 참조하세요.

> **Note**
>
> Internet Monitor는 5분마다 데이터를 게시하므로 Contributor Insights 규칙을 설정한 후 그래프를 보려면 기간을 5분으로 조정해야 합니다.

가용성 영향의 영향을 받는 상위 위치 및 ASN 보기

가용성 저하로 인해 영향을 받는 상위 클라이언트 위치 및 ASN을 보려면 구문 편집기에서 다음 Contributor Insights 규칙을 사용하면 됩니다. *monitor-name*을 사용자의 모니터 이름으로 바꿉니다.

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.awsInternetHealth.availability.percentageOfTotalTrafficImpacted"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
    ]
}
```

지연 시간 영향의 영향을 받는 상위 클라이언트 위치 및 ASN 보기

왕복 시간(지연 시간) 증가의 영향을 받는 상위 클라이언트 위치 및 ASN을 보려면 구문 편집기에서 다음 Contributor Insights 규칙을 사용하세요. *monitor-name*을 사용자의 모니터 이름으로 바꿉니다.

```
{
```

```
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [            {
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.awsInternetHealth.performance.percentageOfTotalTrafficImpacted"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
    ]
}
```

총 트래픽 백분율에 따라 영향을 받는 상위 클라이언트 위치 및 ASN 보기

총 트래픽 백분율에 따라 영향을 받는 상위 클라이언트 위치 및 ASN을 보려면 구문 편집기에서 다음 Contributor Insights 규칙을 사용하세요. *monitor-name*을 사용자의 모니터 이름으로 바꿉니다.

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
```

```
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.percentageOfTotalTraffic"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
    ]
}
```

Amazon CloudWatch Internet Monitor와 함께 CloudWatch 지표 사용

Amazon CloudWatch Internet Monitor는 성능, 가용성, 왕복 시간 및 처리량(초당 바이트)에 대한 지표를 포함하여 계정에 지표를 게시하며, 이 지표는 CloudWatch 콘솔의 CloudWatch 지표에서 확인할 수 있습니다. 모니터에 대한 모든 지표를 찾으려면 CloudWatch 지표 대시보드에서 사용자 지정 네임스페이스 `AWS/InternetMonitor`를 참조하세요.

지표는 모니터의 VPC, Network Load Balancer, CloudFront 배포 또는 WorkSpaces 디렉터리에 대한 모든 인터넷 트래픽과 모니터되는 각 AWS 리전 및 인터넷 엣지 로케이션에 대한 모든 트래픽에 대해 집계됩니다. 리전은 서비스 위치로 정의되며 모든 위치 또는 `us-east-1`과 같은 특정 리전일 수 있습니다.

참고: 도시-네트워크는 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 제공업체(ISP))입니다.

Internet Monitor는 다음 지표를 제공합니다.

지표	설명
PerformanceScore	성능 점수는 성능 저하가 나타나지 않는 트래픽의 예상 백분율을 나타냅니다.
AvailabilityScore	가용성 점수는 가용성 저하가 나타나지 않는 트래픽의 예상 백분율을 나타냅니다.
BytesIn	모든 애플리케이션 도시-네트워크에서 애플리케이션 인터넷 트래픽으로 전송된 바이트 수입니다.

지표	설명
BytesOut	모든 애플리케이션 도시-네트워크에서 애플리케이션 인터넷 트래픽에서 전송된 바이트 수입니다.
BytesInMonitored	모니터되는 도시-네트워크에서 애플리케이션 인터넷 트래픽으로 전송된 바이트 수입니다.
BytesOutMonitored	모니터되는 도시-네트워크에서 애플리케이션 인터넷 트래픽에서 전송된 바이트 수입니다.
왕복 시간(RTT)	AWS 리전, ASN(일반적으로 인터넷 서비스 제공업체(ISP)) 및 VPC, Network Load Balancer, CloudFront 배포 또는 WorkSpaces 디렉터리와 관련된 위치(예: 도시) 간의 왕복 시간입니다.
CityNetworksMonitored	애플리케이션 인터넷 트래픽을 모니터하는 도시-네트워크 Internet Monitor의 수입니다. 이는 모니터의 최대 도시-네트워크로 설정한 상한을 초과할 수 없습니다.
TrafficMonitoredPercent	Internet Monitor가 모니터링하고 있는 도시-네트워크로 표시(포함됨)되는 이 모니터링에 대한 총 애플리케이션 인터넷 트래픽의 백분율입니다. 클라이언트가 모니터에 설정한 최대 도시-네트워크 한도보다 더 많은 도시-네트워크에서 애플리케이션에 액세스하는 경우 이 수치는 100 미만(즉, 100% 미만)입니다.
CityNetworksFor100PercentTraffic	Internet Monitor에서 애플리케이션 인터넷 트래픽의 100%를 모니터하려는 경우 도시-네트워크 최대 한도로 설정해야 하는 수입니다.
CityNetworksFor99PercentTraffic	Internet Monitor에서 애플리케이션 인터넷 트래픽의 99%를 모니터하려는 경우 도시-네트워크 최대 한도로 설정해야 하는 수입니다.

지표	설명
CityNetworksFor95PercentTraffic	Internet Monitor에서 애플리케이션 인터넷 트래픽의 95%를 모니터하려는 경우 도시-네트워크 최대 한도로 설정해야 하는 수입니다.
CityNetworksFor90PercentTraffic	Internet Monitor에서 애플리케이션 인터넷 트래픽의 90%를 모니터하려는 경우 도시-네트워크 최대 한도로 설정해야 하는 수입니다.
CityNetworksFor75PercentTraffic	Internet Monitor에서 애플리케이션 인터넷 트래픽의 75%를 모니터하려는 경우 도시-네트워크 최대 한도로 설정해야 하는 수입니다.
CityNetworksFor50PercentTraffic	Internet Monitor에서 애플리케이션 인터넷 트래픽의 50%를 모니터하려는 경우 도시-네트워크 최대 한도로 설정해야 하는 수입니다.
CityNetworksFor25PercentTraffic	Internet Monitor에서 애플리케이션 인터넷 트래픽의 25%를 모니터하려는 경우 도시-네트워크 최대 한도로 설정해야 하는 수입니다.

> **Note**
>
> 모니터의 도시-네트워크 최대값을 선택하는 데 도움이 되도록 이러한 여러 지표의 사용 사례를 보려면 도시-네트워크 최대값 선택을 참조하세요.

자세한 내용은 Amazon CloudWatch 지표 사용 단원을 참조하십시오.

Amazon Athena를 사용하여 Amazon S3 로그 파일에서 인터넷 측정값 쿼리

Amazon Athena를 사용하여 Amazon CloudWatch Internet Monitor가 Amazon S3 버킷에 게시하는 인터넷 측정값을 쿼리하고 확인할 수 있습니다. Internet Monitor에는 애플리케이션의 인터넷 측정값을 모니터하는 도시-네트워크(클라이언트 위치 및 ASN, 일반적으로 인터넷 서비스 제공업체(ISP))의 인터넷 대상 트래픽에 대한 S3 버킷에 게시하는 옵션이 있습니다. S3에 대한 측정값 게시 여부와 관계없이 Internet Monitor는 각 모니터의 상위 500개(트래픽 볼륨 기준) 도시-네트워크에 대해 5분마다 인터넷 측정값을 CloudWatch 로그에 자동으로 게시합니다.

이 장에서는 S3 로그 파일에 있는 인터넷 측정값에 대해 Athena에서 테이블을 만드는 방법에 대한 단계와 측정값의 다양한 보기를 볼 수 있는 예 쿼리를 제공합니다. 예를 들어 지연 시간 영향에 따라 영향을 받은 상위 10개 도시-네트워크에 대해 쿼리할 수 있습니다.

Amazon Athena를 사용하여 Internet Monitor에서 인터넷 측정값 테이블 생성

Internet Monitor S3 로그 파일과 함께 Athena를 사용하려면 먼저 인터넷 측정값에 대한 테이블을 생성하세요.

이 절차의 단계에 따라 S3 로그 파일을 기반으로 Athena에서 테이블을 생성합니다. 그러면 테이블에서 다음 예 인터넷 측정값 쿼리와 같은 Athena 쿼리를 실행하여 측정에 대한 정보를 얻을 수 있습니다.

Athena 테이블을 만들려면

1. https://console.aws.amazon.com/athena/에서 Athena 콘솔을 엽니다.
2. Athena 쿼리 편집기에서 쿼리 문을 입력하여 Internet Monitor 인터넷 측정값이 포함된 테이블을 생성합니다. 위치 파라미터의 값을 Internet Monitor 인터넷 측정값이 저장되는 S3 버킷의 위치로 바꿉니다.

```
CREATE EXTERNAL TABLE internet_measurements (
    version INT,
    timestamp INT,
    clientlocation STRING,
    servicelocation STRING,
    percentageoftotaltraffic DOUBLE,
    bytesin INT,
    bytesout INT,
    clientconnectioncount INT,
    internethealth STRING,
    trafficinsights STRING
)
PARTITIONED BY (year STRING, month STRING, day STRING)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION
's3://bucket_name/bucket_prefix/AWSLogs/account_id/internetmonitor/AWS_Region/'
TBLPROPERTIES ('skip.header.line.count' = '1');
```

3. 데이터를 읽을 파티션을 생성하는 문을 입력합니다. 예를 들어 다음 쿼리는 지정된 날짜와 위치에 대해 단일 파티션을 생성합니다.

```
ALTER TABLE internet_measurements
ADD PARTITION (year = 'YYYY', month = 'MM', day = 'dd')
```

```
LOCATION
's3://bucket_name/bucket_prefix/AWSLogs/account_id/internetmonitor/AWS_Region/YYYY/
MM/DD';
```

4. Run(실행)을 선택합니다.

인터넷 측정값에 대한 Athena 문 예시

다음은 테이블을 생성하는 문의 예입니다.

```
CREATE EXTERNAL TABLE internet_measurements (
    version INT,
    timestamp INT,
    clientlocation STRING,
    servicelocation STRING,
    percentageoftotaltraffic DOUBLE,
    bytesin INT,
    bytesout INT,
    clientconnectioncount INT,
    internethealth STRING,
    trafficinsights STRING
)
PARTITIONED BY (year STRING, month STRING, day STRING)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://internet-measurements/TestMonitor/AWSLogs/1111222233332/internetmonitor/
us-east-2/'
TBLPROPERTIES ('skip.header.line.count' = '1');
```

다음은 데이터를 읽을 파티션을 생성하는 문의 예입니다.

```
ALTER TABLE internet_measurements
ADD PARTITION (year = '2023', month = '04', day = '07')
LOCATION 's3://internet-measurements/TestMonitor/AWSLogs/1111222233332/internetmonitor/
us-east-2/2023/04/07/'
```

Internet Monitor에서 인터넷 측정값에 사용할 샘플 Amazon Athena 쿼리

이 섹션에는 Amazon Athena에서 Amazon S3에 게시된 애플리케이션의 인터넷 측정값에 대한 정보를 얻기 위해 사용할 수 있는 쿼리 예가 포함되어 있습니다.

영향을 받는 상위 10개(총 트래픽 백분율 기준) 클라이언트 위치 및 ASN 쿼리

이 Athena 쿼리를 실행하여 영향을 받는 상위 10개(총 트래픽 백분율 기준) 도시-네트워크, 즉 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 제공업체)을 반환합니다.

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(percentageoftotaltraffic) as percentageoftotaltraffic
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageoftotaltraffic desc
limit 10
```

영향을 받는 상위 10개(가용성 기준) 클라이언트 위치 및 ASN 쿼리

이 Athena 쿼리를 실행하여 영향을 받는 상위 10개(총 트래픽 백분율 기준) 도시-네트워크, 즉 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 제공업체)을 반환합니다.

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(
        cast(
            json_extract_scalar(
                internetHealth,
                '$.availability.percentageoftotaltrafficimpacted'
            )
        as double )
    ) as percentageOfTotalTrafficImpacted
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageOfTotalTrafficImpacted desc
limit 10
```

영향을 받는 상위 10개(지연 시간 기준) 클라이언트 위치 및 ASN 쿼리

이 Athena 쿼리를 실행하여 영향을 받는(지연 시간 영향에 따라) 상위 10개 도시-네트워크, 즉 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 제공업체)을 반환합니다.

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(
```

```
        cast(
            json_extract_scalar(
                internetHealth,
                '$.performance.percentageoftotaltrafficimpacted'
            )
        as double )
    ) as percentageOfTotalTrafficImpacted
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageOfTotalTrafficImpacted desc
limit 10
```

클라이언트 위치 및 ASN의 트래픽 하이라이트 쿼리

이 Athena 쿼리를 실행하여 도시-네트워크, 즉 클라이언트 위치 및 ASN(일반적으로 인터넷 서비스 제공업체)에 대한 가용성 점수, 성능 점수, 첫 바이트까지의 시간 등 트래픽 하이라이트를 반환합니다.

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.subdivision') as subdivision,
    json_extract_scalar(clientLocation, '$.country') as country,
    avg(cast(json_extract_scalar(internetHealth, '$.availability.experiencescore') as
 double)) as availabilityScore,
    avg(cast(json_extract_scalar(internetHealth, '$.performance.experiencescore') as
 double)) performanceScore,
    avg(cast(json_extract_scalar(trafficinsights,
 '$.timetofirstbyte.currentexperience.value') as double)) as averageTTFB,
    sum(bytesIn) as bytesIn,
    sum(bytesOut) as bytesOut,
    sum(bytesIn + bytesOut) as totalBytes
FROM internet_measurements
where json_extract_scalar(clientLocation, '$.city') != 'N/A'
GROUP BY
json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.subdivision'),
    json_extract_scalar(clientLocation, '$.country')
ORDER BY totalBytes desc
limit 100
```

Athena 사용에 대한 자세한 내용은 Amazon Athena 사용 설명서를 참조하세요.

Amazon CloudWatch Internet Monitor 쿼리 인터페이스 사용

Amazon CloudWatch Internet Monitor 쿼리 인터페이스를 사용하면 AWS 애플리케이션의 인터넷 트래픽을 더 자세히 파악할 수 있습니다. 쿼리 인터페이스를 사용하려면 선택한 데이터 필터를 사용하여 쿼리를 생성한 다음 쿼리를 실행하여 Internet Monitor 데이터의 하위 집합을 반환해야 합니다. 쿼리가 반환하는 데이터를 탐색하면 애플리케이션이 인터넷에서 어떻게 작동하는지 파악할 수 있습니다.

가용성 및 성능 점수, 전송된 바이트, 왕복 시간, 첫 번째 바이트까지 걸리는 시간(TTFB) 등 Internet Monitor가 모니터로 캡처하는 모든 지표를 쿼리하고 탐색할 수 있습니다.

Internet Monitor는 쿼리 인터페이스를 사용하여 Internet Monitor 콘솔 대시보드에서 탐색할 수 있는 데이터를 제공합니다. 대시보드(기록 탐색기 탭 또는 트래픽 인사이트 탭)의 검색 옵션을 사용하여 애플리케이션의 인터넷 데이터를 쿼리하고 필터링할 수 있습니다.

대시보드가 제공하는 것보다 더 유연하게 데이터를 탐색하고 필터링하려면 Internet Monitor API 작업을 AWS Command Line Interface 또는 AWS SDK와 함께 사용하여 쿼리 인터페이스를 직접 사용할 수 있습니다. 이 섹션에서는 쿼리 인터페이스에서 사용할 수 있는 쿼리 유형과 데이터의 하위 집합을 생성하여 애플리케이션의 인터넷 트래픽에 대한 인사이트를 얻기 위해 지정할 수 있는 필터를 소개합니다.

주제

쿼리 인터페이스 사용 방법

쿼리 유형을 선택한 다음 필터 값을 지정하여 쿼리 인터페이스로 쿼리를 생성하고, 로그 파일 데이터의 원하는 특정 하위 집합을 반환합니다. 그런 다음 데이터 하위 집합을 사용하여 추가 필터링 및 정렬, 보고서 생성 등의 작업을 수행할 수 있습니다.

쿼리 프로세스는 다음과 같습니다.

1. 쿼리를 실행하면 Internet Monitor가 쿼리의 고유한 `query ID`를 반환합니다. 이 섹션에서는 사용 가능한 쿼리 유형과 쿼리의 데이터를 필터링하는 옵션을 설명합니다. 작동 방식을 이해하려면 쿼리 예제에 대한 섹션을 검토할 수 있습니다.

2. GetQueryResults API 작업을 통해 모니터 이름과 함께 쿼리 ID를 지정하여 쿼리에 대한 데이터 결과를 반환합니다. 각 쿼리 유형은 서로 다른 데이터 필드 세트를 반환합니다. 자세히 알아보려면 쿼리 결과 가져오기를 참조하세요.

쿼리 인터페이스는 다음과 같은 세 가지 쿼리 유형을 제공합니다. 각 쿼리 유형은 다음과 같이 로그 파일의 트래픽에 대한 다양한 정보 세트를 반환합니다.

- 측정: 가용성 점수, 성능 점수, 총 트래픽, 왕복 시간을 5분 간격으로 제공합니다.
- 상위 위치: 모니터링 중인 상위 위치 및 ASN 조합에 대한 가용성 점수, 성능 점수, 총 트래픽, TTFB(첫 번째 바이트까지 시간) 정보를 트래픽 볼륨별로 제공합니다.
- 주요 위치 세부 정보: Amazon CloudFront용 TTFB, 현재 구성, 최고 성능의 Amazon EC2 구성을 1시간 간격으로 제공합니다.

각 쿼리 유형에서 다음 기준 중 하나 이상을 지정하여 데이터를 더 많이 필터링할 수 있습니다.

- AWS 위치: AWS 위치의 경우 CloudFront 또는 AWS 리전(`us-east-2`, `us-west-2`) 등과 같이 지정할 수 있습니다.
- ASN: 일반적으로 인터넷 서비스 제공업체(ISP) 인 ASN을 지정합니다.
- 클라이언트 위치: 위치는 도시, 대도시, 구역 또는 국가를 지정합니다.
- 지역: 일부 쿼리에서 `geo`를 지정합니다. 이는 `Top locations` 쿼리 유형을 사용하는 쿼리에는 필요하지만 다른 쿼리 유형에는 허용되지 않습니다. 필터 파라미터에서 `geo` 지정 시기를 이해하려면 쿼리 예제 섹션을 참조하세요.

데이터 필터링에 사용할 수 있는 연산자는 `EQUALS` 및 `NOT_EQUALS`입니다. 매개 변수 필터링에 대한 자세한 내용은 FilterParameter API 작업을 참조하세요.

쿼리 인터페이스 작업에 대한 자세한 내용은 Amazon CloudWatch Internet Monitor API 참조 안내서의 다음 API 작업을 참조하세요.

- 쿼리를 생성하고 실행하려면 StartQuery API 작업을 참조하세요.
- 쿼리를 중지하려면 StopQuery API 작업을 참조하세요.
- 생성한 쿼리의 데이터를 반환하려면 GetQueryResults API 작업을 참조하세요.
- 쿼리 상태를 검색하려면 GetQueryStatus API 작업을 참조하세요.

쿼리 예제

모니터의 로그 파일에서 필터링된 데이터 세트를 검색하는 데 사용할 수 있는 쿼리를 생성하려면 StartQuery API 작업을 사용합니다. 쿼리 유형을 지정하고 쿼리의 매개 변수를 필터링합니다. 그런 다음 Internet Monitor 쿼리 인터페이스 API 작업을 사용하여 해당 쿼리를 사용한 쿼리 결과를 가져오면 작업하려는 데이터의 하위 집합이 검색됩니다.

쿼리 유형과 필터 매개변수의 작동 방식을 설명하기 위해 몇 가지 예를 살펴보겠습니다.

예 1

한 도시를 제외하고 특정 국가에 대한 모니터의 모든 로그 파일 데이터를 검색한다고 가정해 보겠습니다. 다음 예제는 이 시나리오의 `StartQuery` 작업으로 생성하 수 있는 쿼리의 필터 매개변수를 보여줍니다.

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "MEASUREMENTS"
   FilterParameters: [
      {
       Field: "country",
       Operator: "EQUALS",
       Values: ["Germany"]
      },
      {
       Field: "city",
       Operator: "NOT_EQUALS",
       Values: ["Berlin"]
      },
    ]
}
```

예제 2

또 다른 예로 상위 위치를 대도시별로 확인한다고 가정해 보겠습니다. 이 시나리오에서는 다음 예제 쿼리를 사용할 수 있습니다.

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
```

```
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATIONS"
   FilterParameters: [
      {
       Field: "geo",
       Operator: "EQUALS",
       Values: ["metro"]
      },
    ]
}
```

예 3

이제 로스앤젤레스 대도시 지역의 주요 도시-네트워크 조합을 확인한다고 가정해 보겠습니다. 이렇게 하려면 geo=city을 지정하고 metro을 로스앤젤레스로 설정합니다. 이제 쿼리는 전체 상위 메트로+네트워크 대신 로스앤젤레스 대도시 지역의 상위 도시-네트워크를 반환합니다.

사용할 수 있는 예제 쿼리는 다음과 같습니다.

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATIONS"
   FilterParameters: [
      {
       Field: "geo",
       Operator: "EQUALS",
       Values: ["city"]
      },
      {
       Field: "metro",
       Operator: "EQUALS",
       Values: ["Los Angeles"]
      }
    ]
}
```

예 4

마지막으로 특정 세분화(예: 미국의 주)에 대한 TTFB 데이터를 검색한다고 가정해 보겠습니다.

다음은 이 시나리오의 예제 쿼리입니다.

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATION_DETAILS"
   FilterParameters: [
      {
       Field: "subdivision",
       Operator: "EQUALS",
       Values: ["California"]
      },
    ]
}
```

쿼리 결과 가져오기

쿼리를 정의한 후 다른 Internet Monitor API 작업인 GetQueryResults를 실행하여 쿼리와 함께 결과 세트를 반환할 수 있습니다. GetQueryResults를 실행할 때 모니터 이름과 함께 정의한 쿼리의 쿼리 ID를 지정합니다. GetQueryResults가 지정된 쿼리의 데이터를 결과 집합으로 검색합니다.

쿼리 실행 시 GetQueryResults을 사용하여 결과를 확인하기 전에 쿼리 실행이 완료되었는지 확인하세요. GetQueryStatus API 작업을 사용하여 쿼리가 완료되었는지 확인할 수 있습니다. 쿼리의 Status가 SUCCEEDED인 경우 결과를 검토할 수 있습니다.

쿼리가 완료되면 다음 정보를 결과를 검토하는 데 사용할 수 있습니다. 쿼리를 생성할 때 사용하는 각 쿼리 유형에는 다음 목록에 설명된 대로 로그 파일의 고유한 데이터 필드 세트가 포함됩니다.

측정

measurements 쿼리 유형은 다음과 같은 데이터를 반환합니다.

timestamp, availability, performance, bytes_in, bytes_out, rtt_p50, rtt_p90, rtt_p95

상위 위치

top locations 쿼리 유형은 위치별로 데이터를 그룹화하고 기간 동안의 평균 데이터를 제공합니다. 반환되는 데이터에는 다음이 포함됩니다.

aws_location, city, metro, subdivision, country, asn, availability, performance, bytes_in, bytes_out, current_fbl, best_ec2, best_ec2_region, best_cf_fbl

참고로 city, metro, subdivision는 geo 필드에서 해당 위치 유형을 선택한 경우에만 반환됩니다. geo에 지정한 위치 유형에 따라 다음과 같은 위치 필드가 반환됩니다.

```
city = city, metro, subdivision, country
metro = metro, subdivision, country
subdivision = subdivision, country
country = country
```

상위 위치 세부 정보

`top locations details` 쿼리 유형은 시간별로 그룹화된 데이터를 반환합니다. 이 쿼리는 다음과 같은 데이터를 반환합니다.

`timestamp, current_service, current_fbl, best_ec2_fbl, best_ec2_region, best_cf_fbl`

`GetQueryResults` API 작업을 실행하면 Internet Monitor가 응답으로 다음을 반환합니다.

- 쿼리가 반환하는 결과를 포함하는 데이터 문자열 배열입니다. 이 정보는 `Fields` 필드에 따라 정렬된 배열로 반환되며 API 호출에서도 반환됩니다. `Fields` 필드를 사용하여 `Data` 리포지토리의 정보를 파싱한 다음 용도에 맞게 추가로 필터링하거나 정렬할 수 있습니다.
- 쿼리가 (`Data` 필드 응답에서) 데이터를 반환한 필드를 나열하는 필드 배열입니다. 배열의 각 항목은 이름-데이터 유형 쌍입니다(예: `availability_score-float`).

문제 해결

쿼리 인터페이스 API 작업을 사용할 때 오류가 반환되는 경우 Amazon CloudWatch Internet Monitor를 사용에 필요한 권한이 있는지 확인하세요. 다음 권한이 활성화되어 있는지 확인하세요.

```
internetmonitor:StartQuery
internetmonitor:GetQueryStatus
internetmonitor:GetQueryResults
internetmonitor:StopQuery
```

이러한 권한은 콘솔의 Internet Monitor 대시보드를 사용하기 위한 권장 AWS Identity and Access Management 정책에 포함되어 있습니다. 자세한 내용은 Amazon CloudWatch Internet Monitor에 대한 IAM 권한 단원을 참조하십시오.

Amazon CloudWatch Internet Monitor를 사용하여 경보 생성

다른 Amazon CloudWatch 지표와 마찬가지로 Amazon CloudWatch Internet Monitor 지표를 기반으로 Amazon CloudWatch 경보를 생성할 수 있습니다.

예를 들어 Internet Monitor 지표 `PerformanceScore`를 기반으로 경보를 생성하고 지표가 선택한 값보다 낮을 때 알림을 보내도록 구성할 수 있습니다. 다른 CloudWatch 지표와 동일한 지침에 따라 Internet Monitor 지표에 대한 경보를 구성합니다.

다음은 경보를 생성하기 위해 선택할 수 있는 Internet Monitor 지표의 예입니다.

- PerformanceScore
- AvailabilityScore
- RoundtripTime

Internet Monitor에 사용할 수 있는 모든 지표를 확인하려면 Amazon CloudWatch Internet Monitor와 함께 CloudWatch 지표 사용 섹션을 참조하세요.

다음 절차에서는 PerformanceScore에 경보를 설정하는 예를 설명합니다. 먼저 CloudWatch 대시보드에서 지표로 이동합니다. 그런 다음 표준 CloudWatch 단계에서 선택한 임계값에 따라 경보를 생성하고 알림을 설정하거나 다른 옵션을 선택합니다.

CloudWatch 지표에서 PerformanceScore에 대한 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 지표를 선택한 다음 모든 지표를 선택합니다.
3. `AWS/InternetMonitor`를 선택하여 Internet Monitor를 필터링할 수 있습니다.
4. MeasurementSource, MonitorName을 선택합니다.
5. 목록에서 PerformanceScore를 선택합니다.
6. GraphedMetrics 탭의 작업에서 벨 아이콘을 선택하여 정적 임계값을 기반으로 경보를 생성합니다.

이제 표준 CloudWatch 단계에 따라 경보 옵션을 선택합니다. 예를 들어 PerformanceScore가 특정 임계값 미만인 경우 Amazon SNS 메시지로 알림을 받도록 선택할 수 있습니다. 대시보드에 경보를 추가할 수도 있습니다.

다음 사항에 유의하세요.

- Internet Monitor 지표는 일반적으로 20분 이내에 계산 및 게시됩니다.
- Internet Monitor 지표를 기반으로 경보를 생성하 경우 경보의 룩백 기간을 설정할 때 게시 전의 짧은 지연 시간을 고려해야 합니다. 룩백 기간을 최소 25분으로 설정하여 평가 기간을 구성하는 것이 좋습니다.

Internet Monitor와 함께 CloudWatch 경보를 사용하는 방법에 대한 자세한 내용은 다음 블로그 게시물 인터넷 가시성 향상을 위한 Amazon CloudWatch Internet Monitor 사용을 참조하세요.

CloudWatch 경보를 생성하는 옵션에 대한 자세한 내용은 정적 임곗값을 기반으로 CloudWatch 경보 생성 섹션을 참조하세요.

Amazon EventBridge와 함께 Amazon CloudWatch Internet Monitor 사용

Amazon CloudWatch Internet Monitor가 네트워킹 문제에 대해 생성하는 상태 이벤트는 Amazon EventBridge를 통해 게시되므로 애플리케이션에 대한 최종 사용자 경험 저하에 대한 알림을 전송할 수 있습니다.

EventBridge를 사용하여 Internet Monitor 상태 이벤트로 다음 지침을 따르세요.

EventBridge에서 Internet Monitor에 대한 규칙을 설정하려면 다음을 수행하세요.

1. AWS Management Console의 EventBridge에서 Rules(규칙)를 선택한 다음 이름과 설명을 입력합니다. Default(기본) 이벤트 버스에 대해 규칙을 생성합니다.
2. 2단계에서 이벤트 소스로 기타를 선택한 다음 이벤트 패턴에서 다음 소스와 일치하도록 합니다.

```
{
  "source": ["aws.internetmonitor"]
}
```

3. 3단계에서 대상에 대해 AWS Service와 CloudWatch Logs Group(CloudWatch Logs 그룹)을 선택한 다음 기존 로그 그룹을 선택하거나 새 로그 그룹을 생성합니다.
4. 원하는 태그를 추가한 다음 규칙을 생성합니다. 그러면 선택한 CloudWatch Logs 그룹이 EventBridge의 이벤트로 채워집니다.

이벤트 패턴과 함께 EventBridge 규칙이 작동하는 방식에 대한 자세한 내용은 EventBridge 사용 안내서에서 Amazon EventBridge event patterns를 참조하세요.

CloudWatch 로그 및 지표 액세스 오류 문제 해결

일부 기능을 지원하려면 Amazon CloudWatch Internet Monitor가 로그와 지표를 비롯한 특정 Amazon CloudWatch 리소스와 상호 작용해야 합니다. Internet Monitor가 액세스가 필요한 CloudWatch 리소스에 액세스할 수 없는 경우 Internet Monitor는 모니터에 대해 `FAULT_ACCESS_CLOUDWATCH` 상태 코드를 설정합니다.

모니터가 `FAULT_ACCESS_CLOUDWATCH` 상태가 되는 데에는 여러 가지 이유가 있습니다. 다음 섹션에는 이러한 오류의 가능한 원인과 문제 해결 단계 제안이 나열되어 있습니다.

Internet Monitor가 사용자 계정의 CloudWatch 로그에 액세스할 수 없음

Internet Monitor는 모니터가 추적하는 애플리케이션 트래픽에 대한 진단 로그를 게시합니다. 이 로그를 `/aws/internet-monitor/`*`monitor_name`*`/[byCity|byMetro|bySubdivision|byCountry]`의 CloudWatch Logs 내 로그 그룹에 게시합니다. Internet Monitor가 로그 그룹에 액세스하지 못했습니다.

오류 상태 및 잠재적 해결 방법:

- PutLogEvents 제한 오류: 모니터의 로그를 CloudWatch에 게시하려고 할 때 Internet Monitor 서비스가 제한되었을 수 있습니다. 계정의 제한 한도를 검토하고 필요한 경우 한도 증가를 요청합니다.
- 로그 그룹을 찾을 수 없음: 모니터를 비활성했다가 다시 활성화합니다. 모니터를 활성화하면 로그 그룹 생성이 다시 시작되어 문제가 해결될 수 있습니다.
- PutLogEvents 액세스 거부 오류: AWS에 지원을 문의합니다.
- PutLogEvents 알 수 없는 오류 또는 일반 오류: AWS에 지원을 문의합니다.

Internet Monitor가 사용자 계정의 CloudWatch 지표에 액세스할 수 없음

Internet Monitor는 모니터가 추적하는 애플리케이션 트래픽에 대한 특정 CloudWatch 지표를 제공합니다. Internet Monitor가 이러한 지표를 CloudWatch에 전달하려고 할 때 오류가 발생했습니다.

오류 상태 및 잠재적 해결 방법:

- PutMetricData 제한 오류: 모니터의 지표를 CloudWatch에 게시하려고 할 때 Internet Monitor 서비스가 제한되었을 수 있습니다. 계정의 제한 한도를 검토하고 필요한 경우 한도 증가를 요청합니다.
- PutMetricData 액세스 거부 오류: AWS에 지원을 문의합니다.
- PutMetricData 알 수 없는 오류 또는 일반 오류: AWS에 지원을 문의합니다.

Amazon CloudWatch Internet Monitor를 통한 데이터 보호 및 데이터 프라이버시

AWS 공동 책임 모델은 Amazon CloudWatch Internet Monitor에 데이터 보호와 데이터 프라이버시를 적용합니다. 이 모델에서 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 이 인프라에서 호스팅되는 콘텐츠에 대한 통제를 유지하는 것은 사용자의 책임입니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그에서 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요. GDPR 요구 사항 준수에 대한 자세한 리소스는 일반 데이터 보호 규정(GDPR) 센터를 참조하세요.

자유 형식 필드에 최종 사용자 계정 번호, 메일 주소, 기타 개인 정보와 같은 중요 식별 정보를 절대 입력하지 마세요. Amazon CloudWatch Internet Monitor 또는 기타 서비스에 입력하는 모든 데이터는 진단 로그에 포함될 수 있습니다.

Amazon CloudWatch Internet Monitor에 대한 ID 및 액세스 관리

AWS Identity and Access Management(IAM)는 관리자가 AWS리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 Internet Monitor 리소스를 사용할 수 있도록 인증(로그인)하고 권한을 부여(권한 보유)할 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

Important

2023년 2월 24일자 Internet Monitor 리소스 변경
2023년 2월 24일 이전에 Internet Monitor 리소스를 포함하는 IAM 정책을 만든 경우 Internet Monitor 리소스 및 리소스 유형에 대한 다음 변경 사항에 유의하세요.

- HealthEvents 리소스의 이름이 HealthEvent로 변경되었습니다.
- HealthEvent 리소스의 ARN 및 Regex 형식이 업데이트되었습니다.
- Monitor 리소스의 ARN 및 Regex 형식이 업데이트되었습니다.
- 이제 GetHealthEvent 작업에 대한 리소스 수준 권한은 HealthEvent 리소스 유형에서만 지원됩니다. 모니터 리소스에서는 지원되지 않습니다.
- 모니터 리소스 유형에 대한 TagResource, UntagResource, ListTagsForResource가 필수 항목으로 업데이트되었습니다.

Internet Monitor에서 AWS 리소스에 대한 액세스를 관리하기 위해 정책에 지정할 수 있는 작업, 리소스 및 조건 키에 대한 자세한 내용은 Amazon CloudWatch Internet Monitor에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

내용

Amazon CloudWatch Internet Monitor와 IAM의 작동 방식

IAM을 사용하여 Internet Monitor에 대한 액세스를 관리하기 전에 Internet Monitor에서 사용할 수 있는 IAM 기능에 대해 알아보세요.

AWS 서비스가 대부분의 IAM 기능과 어떻게 작동하는지에 대한 유사한 상위 수준 보기를 보여주는 표를 보려면 IAM 사용 가이드에서 AWS IAM으로 작업하는 서비스를 참조하세요.

Amazon CloudWatch Internet Monitor에서 사용할 수 있는 IAM 기능

IAM 특성	Internet Monitor 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACLs	아니요
ABAC(정책 내 태그)	부분

IAM 특성	Internet Monitor 지원
임시 보안 인증	예
보안 주체 권한	예
서비스 역할	아니요
서비스 링크 역할	예

Internet Monitor에 대한 ID 기반 정책

ID 기반 정책 지원	예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업와 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Internet Monitor 내 리소스 기반 정책

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다.

Internet Monitor에 대한 정책 조치

정책 작업 지원	예

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 Action요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWSAPI 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

Internet Monitor 작업 목록을 보려면 서비스 승인 참조에서 Amazon CloudWatch Internet Monitor에서 정의한 작업을 참조하세요.

Internet Monitor의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
internetmonitor
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "internetmonitor:action1",
      "internetmonitor:action2"
         ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "internetmonitor:Describe*"
```

Internet Monitor에 대한 정책 리소스

정책 리소스 지원	예

서비스 승인 참조에서 Internet Monitor와 관련된 다음 정보를 볼 수 있습니다.

- Internet Monitor의 리소스 유형 및 ARN의 목록을 보려면 Amazon CloudWatch Internet Monitor에서 정의한 리소스를 참조하세요.

- 각 리소스의 ARN으로 지정할 수 있는 작업에 대해 알아보려면 Amazon CloudWatch Internet Monitor에서 정의한 작업을 참조하세요.

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 타입을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Internet Monitor의 정책 조건 키

서비스별 정책 조건 키 지원	예

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

Condition 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 Condition요소를 지정하거나 단일 Condition요소에서 여러 키를 지정하는 경우 AWS는 논리적 AND태스크를 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS는 논리적 OR태스크를 사용하여 조건을 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용자 가이드의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Internet Monitor 조건 키 목록을 보려면 서비스 승인 참조에서 Amazon CloudWatch Internet Monitor에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon CloudWatch Internet Monitor에서 정의한 작업을 참조하세요.

Internet Monitor의 ACL

ACL 지원	아니요

액세스 제어 목록(ACLs)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는 지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Internet Monitor를 사용한 ABAC

ABAC(정책 내 태그) 지원	부분

Internet Monitor는 정책의 태그를 부분적으로 지원합니다. 하나의 리소스인 모니터에 대한 태그 지정을 지원합니다.

Internet Monitor에서 태그를 사용하려면 AWS Command Line Interface 또는 AWS SDK를 사용하세요. AWS Management Console에서는 Internet Monitor에 대한 태그 지정이 지원되지 않습니다.

일반적으로 정책에서 태그를 사용하는 방법에 대해 자세히 알아보려면 다음 정보를 검토하세요.

ABAC(속성 기반 액세스 제어)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그를 기반으로 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 타입에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 타입에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 정보는 IAM 사용 설명서의 ABAC란 무엇인가요?를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Internet Monitor에서 임시 보안 인증 정보 사용

임시 보안 인증 지원	예

일부 AWS 서비스는 임시 보안 인증을 사용하여 로그인할 때 작동하지 않습니다. 임시 보안 인증으로 작동하는 AWS 서비스를 비롯한 추가 정보는 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 참조하세요.

사용자 이름과 암호를 제외한 다른 방법을 사용하여 AWS Management Console에 로그인하면 임시 보안 인증을 사용하는 것입니다. 예를 들어 회사의 Single Sign-On(SSO) 링크를 사용하여 AWS에 액세스하면 해당 프로세스에서 자동으로 임시 보안 인증을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 정보는 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 보안 인증을 수동으로 만들 수 있습니다 그런 다음 이러한 임시 보안 인증을 사용하여 AWS에 액세스할 수 있습니다. AWS에서는 장기 액세스 키를 사용하는 대신 임시 보안 인증을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 인증 섹션을 참조하세요.

Internet Monitor의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Internet Monitor의 서비스 역할

서비스 역할 지원	아니요

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임하는 IAM role(IAM 역할)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용자 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조합니다.

Internet Monitor의 서비스 연결 역할

서비스 링크 역할 지원	예

서비스 링크 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

Internet Monitor의 서비스 연결 역할에 대한 자세한 내용은 Amazon CloudWatch Internet Monitor용 서비스 연결 역할 섹션을 참조하세요.

AWS에서 일반적으로 서비스 연결 역할을 생성하거나 관리하는 방법에 대한 자세한 내용은 AWS IAM으로 작업하는 서비스를 참조하세요. 서비스 연결 역할 열에서 Yes가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

Amazon CloudWatch Internet Monitor에 대한 AWS 관리형 정책

AWS 관리형 정책은 AWS에서 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. 만약 AWS가 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 보안 주체 엔터티(사용자, 그룹 및 역할)에도 업데이트가 적용됩니다. 새로운 AWS 서비스를 시작하거나 새로운 API 작업을 기존 서비스에 이용하는 경우 AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 설명서의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: CloudWatchInternetMonitorServiceRolePolicy

이 정책은 AWSServiceRoleForInternetMonitor라는 서비스 연결 역할에 연결되어 Internet Monitor가 사용자 계정에 있는 Amazon Virtual Private Cloud 리소스 또는 Network Load Balancer 등의 리소

스에 액세스하도록 허용하여 사용자가 모니터를 생성할 때 선택할 수 있도록 합니다. 자세한 내용은 Amazon CloudWatch Internet Monitor용 서비스 연결 역할 단원을 참조하십시오.

Amazon CloudWatch Internet Monitor에 대한 IAM 권한

Amazon CloudWatch Internet Monitor에서 모니터 및 데이터 사용을 위한 작업에 액세스하려면 사용자에게 올바른 권한이 있어야 합니다.

Amazon CloudWatch의 보안에 대한 자세한 내용은 Amazon CloudWatch의 Identity and Access Management 섹션을 참조하세요.

Amazon CloudWatch Internet Monitor의 읽기 전용 액세스 권한

Amazon CloudWatch Internet Monitor의 모니터링 및 데이터 사용을 위한 읽기 전용 작업에 액세스하려면 사용자가 다음 권한이 있는 사용자 또는 역할로 로그인해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "internetmonitor:Get*",
                "internetmonitor:List*",
                "internetmonitor:StartQuery",
                "internetmonitor:StopQuery",
                "logs:DescribeLogGroups",
                "logs:GetQueryResults",
                "logs:StartQuery",
                "logs:StopQuery"
                ],
            "Resource": "*"
        }
    ]
}
```

Amazon CloudWatch Internet Monitor의 전체 액세스 권한

Amazon CloudWatch Internet Monitor에서 모니터를 생성하고 Internet Monitor에서 모니터 및 데이터 사용을 위한 작업에 대한 전체 액세스 권한을 가지려면 사용자가 다음 권한이 있는 사용자 또는 역할로 로그인해야 합니다.

- Internet Monitor와 연결된 서비스 연결 역할을 생성할 수 있는 권한. 자세한 내용은 Amazon CloudWatch Internet Monitor용 서비스 연결 역할 단원을 참조하십시오.
- Internet Monitor의 모니터 및 데이터에 대한 전체 액세스를 가능하게 하는 작업 권한

> **Note**
>
> 더 제한적인 자격 증명 기반 권한 정책을 생성하는 경우 해당 정책이 적용되는 사용자는 Internet Monitor에서 모니터와 데이터를 생성하고 작업할 수 있는 전체 액세스 권한을 갖지 못할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "internetmonitor:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
internetmonitor.amazonaws.com/AWSServiceRoleForInternetMonitor",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "internetmonitor.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
internetmonitor.amazonaws.com/AWSServiceRoleForInternetMonitor"
        },
```

```
        {
            "Action": [
                "ec2:DescribeVpcs",
                "elasticloadbalancing:DescribeLoadBalancers",
                "workspaces:DescribeWorkspaceDirectories",
                "cloudfront:GetDistribution"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Amazon CloudWatch Internet Monitor용 서비스 연결 역할

Amazon CloudWatch Internet Monitor는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Internet Monitor에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Internet Monitor에서 사전 정의하며, 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

Internet Monitor에서 서비스 연결 역할 권한을 정의하므로, 달리 정의되지 않은 한 Internet Monitor만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Internet Monitor 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-linked role) 열에 예(Yes)가 있는 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Internet Monitor에 대한 서비스 연결 역할 권한

Internet Monitor는 AWSServiceRoleForInternetMonitor라는 서비스 연결 역할을 사용합니다. 이 역할을 사용하면 Internet Monitor가 Amazon 가상 사설 클라우드 리소스, Amazon CloudFront 배포, Amazon WorkSpaces 디렉터리, Network Load Balancer와 같은 계정의 리소스에 액세스할 수 있으므로 모니터를 만들 때 이러한 리소스를 선택할 수 있습니다.

서비스 연결 역할은 관리형 정책 `CloudWatchInternetMonitorServiceRolePolicy`를 사용합니다.

AWSServiceRoleForInternetMonitor 서비스 연결 역할은 다음 서비스를 신뢰하여 역할을 위임합니다.

- `internetmonitor.amazonaws.com`

이 정책의 권한을 보려면 AWS 관리형 정책 참조의 CloudWatchInternetMonitorServiceRolePolicy를 확인하세요.

Internet Monitor의 서비스 연결 역할 생성

Internet Monitor의 서비스 연결 역할을 수동으로 생성할 필요가 없습니다. 모니터를 처음 생성하면 Internet Monitor가 자동으로 AWSServiceRoleForInternetMonitor를 생성합니다.

자세한 정보는 IAM 사용 설명서의 서비스 연결 역할 생성 섹션을 참조하십시오.

Internet Monitor의 서비스 연결 역할 편집

Internet Monitor가 계정에 서비스 연결 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Internet Monitor의 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

Internet Monitor의 모니터에서 리소스를 제거한 다음 모니터를 삭제한 후 서비스 연결 역할 AWSServiceRoleForInternetMonitor를 삭제할 수 있습니다.

> **Note**
>
> 삭제하려는 역할이 Internet Monitor 서비스에서 사용되고 있는 경우 삭제에 실패할 수 있습니다. 이 경우 몇 분 정도 기다린 후 다시 시도하세요.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForInternetMonitor 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Internet Monitor 서비스 연결 역할 업데이트

Internet Monitor 서비스 연결 역할의 AWS 관리형 정책인 AWSServiceRoleForInternetMonitor 업데이트에 대해서는 AWS 관리형 정책에 대한 CloudWatch 업데이트를 참조하세요. CloudWatch의 관리되

는 정책 변경에 대한 자동 알림을 받으려면 CloudWatch 문서 기록 페이지에서 RSS 피드를 구독하세요.

Amazon CloudWatch Internet Monitor의 할당량

Amazon CloudWatch Internet Monitor의 할당량은 다음과 같습니다.

Resource	기본 할당량
리전당 모니터 수	50
모니터당 리소스 수	50
해결된 Internet Monitor 상태 이벤트가 유지되는 일 수	400

Amazon CloudWatch Network Monitor 사용

Amazon CloudWatch Network Monitor를 사용하면 AWS 호스팅된 애플리케이션을 온프레미스 대상에 연결하는 네트워크의 성능을 파악하고 몇 분 내에 네트워크 성능 저하의 원인을 파악할 수 있습니다. Network Monitor는 AWS에서 완전히 관리됩니다. 따라서 네트워크 성능을 모니터링하기 위해 추가 에이전트를 설치할 필요가 없습니다. 하이브리드 네트워크 연결의 패킷 손실과 지연 시간을 빠르게 시각화하고, 알림과 임곗값을 설정한 다음, 최종 사용자의 네트워크 환경을 개선하기 위한 조치를 취할 수 있습니다.

Network Monitor는 네트워크 성능에 대한 실시간 인사이트를 원하는 네트워크 사업자와 애플리케이션 개발자를 위한 것입니다.

주요 기능

- Network Monitor를 사용하면 지속적인 실시간 패킷 손실 및 지연 시간 지표로 변화하는 하이브리드 네트워크 환경을 벤치마킹할 수 있습니다.
- AWS Direct Connect를 사용하여 연결하면 Network Monitor는 CloudWatch 계정에 AWS 네트워크 상태 지표를 써서 네트워크 성능 저하를 신속하게 진단합니다. 이 지표는 네트워크 성능 저하가 AWS 내에서 발생했는지 확인하기 위한 확률적 점수를 제공합니다.
- Network Monitor는 완전관리형 에이전트 접근 방식을 통해 원활한 모니터링을 제공하므로 VPC나 온프레미스에 에이전트를 설치할 필요가 없습니다. 시작하려면 VPC 서브넷과 온프레미스 IP 주소만 지정하면 됩니다.

- Network Monitor는 CloudWatch 지표에 지표를 게시합니다. 대시보드를 생성하여 지표를 확인하고 애플리케이션별 지표에 대해 실행 가능한 임곗값과 경보를 생성할 수 있습니다.

자세한 내용은 the section called “Network Monitor 작동 방식”를 참조하세요.

Network Monitor 용어 및 구성 요소

- 모니터 - 모니터에는 네트워크 성능 및 가용성 측정값을 보려는 리소스와 상태 이벤트 경고를 받으려는 리소스가 표시됩니다. 애플리케이션용 모니터를 생성할 때 AWS 호스팅 리소스를 네트워크 소스로 추가합니다. 그런 다음 Network Monitor는 AWS 호스팅 리소스와 대상 IP 주소 사이에 가능한 모든 프로브 목록을 생성합니다.
- 프로브 - 프로브는 AWS 호스팅 리소스에서 온프레미스 대상 IP 주소로 전송되는 트래픽입니다. Network Monitor 지표는 모니터에 구성된 모든 프로브에 대해 CloudWatch 계정에 기록됩니다.
- AWS 네트워크 소스 - 네트워크 모니터 프로브의 원래 AWS 소스로, 모든 VPC의 서브넷이 됩니다.
- 대상 - AWS 네트워크 소스에 대한 온프레미스 네트워크의 대상입니다. 대상은 온프레미스 IP 주소, 네트워크 프로토콜, 포트 및 네트워크 패킷 크기의 조합입니다. IPv4 주소와 IPv6 주소 모두 지원됩니다.

Network Monitor 제한 및 요구 사항

- Network Monitor는 최대 4개의 대상 IP 주소와 모니터당 최대 24개의 프로브를 지원합니다.
- 계정별로 리전당 최대 100개의 모니터가 있을 수 있습니다.
- 모니터 서브넷은 모니터와 동일한 계정에서 소유해야 합니다.
- Network Monitor는 AWS 네트워크 문제가 발생한 경우 자동 네트워크 장애 조치를 제공하지 않습니다.
- 생성하는 각 프로브에 대해 요금이 부과됩니다. 가격에 대한 자세한 내용은 the section called “요금”을 참조하세요.

Amazon CloudWatch Network Monitor 작동 방식

Network Monitor는 에이전트가 필요 없는 완전관리형 솔루션을 제공하므로 모니터링이 더욱 쉬워집니다. AWS 호스팅 리소스에 모니터를 생성하면 AWS는 왕복 시간 및 패킷 손실 측정을 수행하기 위해 백그라운드에서 모든 인프라를 생성하고 관리합니다. 결과적으로 AWS 인프라 내에 에이전트를 설치하거나 제거할 필요 없이 모니터링을 신속하게 확장할 수 있습니다.

Network Monitor는 AWS 리전의 모든 흐름을 광범위하게 모니터링하는 대신 AWS 호스팅 리소스의 흐름이 이동하는 경로에 대한 모니터링에 중점을 둡니다. 워크로드가 여러 가용 영역에 분산되어 있는 경우 Network Monitor는 각 프라이빗 서브넷의 경로를 모니터링할 수 있습니다.

Network Monitor는 모니터를 생성할 때 설정된 집계 간격을 기반으로 Amazon CloudWatch 계정에 왕복 시간 및 패킷 손실 지표를 게시합니다. CloudWatch를 사용하여 각 모니터에 대해 개별 지연 시간 및 패킷 손실 임곗값을 설정할 수도 있습니다. 예를 들어, 패킷 손실에 민감한 워크로드의 패킷 손실 평균이 정적 0.1% 임곗값보다 높으면 알려주는 경보를 생성할 수 있습니다. CloudWatch 이상 탐지를 사용하여 원하는 범위를 벗어나는 패킷 손실 또는 지연 시간 지표에 대해 경보를 보낼 수도 있습니다.

가용성 및 성능 측정

Network Monitor는 AWS 리소스에서 온프레미스 대상으로 주기적인 활성 프로브를 전송합니다. 모니터를 생성할 때 다음을 지정합니다.

- 집계 간격. CloudWatch가 측정된 결과를 수신하는 시간(초)입니다. 30초 또는 60초 간격입니다. 모니터에 대해 선택한 집계 기간은 해당 모니터의 모든 프로브에 적용됩니다.
- 프로브 프로토콜. 모니터에 추가되는 각 프로브는 인터넷 제어 메시지 프로토콜(ICMP) 또는 전송 제어 프로토콜(TCP) 프로토콜을 사용해야 합니다. 자세한 내용은 the section called “통신 프로토콜” 섹션을 참조하세요.
- 패킷 크기. AWS 호스팅 리소스와 단일 프로브의 대상 간에 전송되는 각 패킷의 크기(바이트)입니다. 모니터의 각 프로브는 고유한 패킷 크기를 가질 수 있습니다.

지표의 경우

- 밀리초 단위로 측정되는 왕복 시간 지표는 성능 측정치를 측정 및 기록하고 프로브가 대상 IP 주소로 전송되고 관련 응답을 수신하는 데 걸리는 시간을 기록합니다.
- 패킷 손실 지표는 전송된 총 패킷의 비율을 측정하고 관련 응답을 받지 못한 전송 프로브 수를 기록합니다. 이는 해당 패킷이 네트워크 경로를 따라 사실상 손실되었음을 의미합니다.

지원되는 통신 프로토콜

ICMP 기반 프로브는 AWS 호스팅 리소스의 ICMP 에코 요청을 대상 주소로 전달하고 대상 주소에서 다시 ICMP 에코 응답을 예상합니다. Network Monitor는 ICMP 에코 요청 및 응답 메시지에 대한 정보를 사용하여 왕복 시간 및 패킷 손실 지표를 계산합니다.

TCP 기반 프로브는 AWS 호스트 리소스에서 대상 주소 및 포트로 TCP SYN 패킷을 전달하고 대상 주소 및 포트에서 다시 TCP SYN+ACK 또는 RST 패킷을 예상합니다. Network Monitor는 TCP SYN 및

TCP SYN+ACK 또는 RST 메시지에 대한 정보를 사용하여 왕복 시간 및 패킷 손실 지표를 계산합니다. 또한 Network Monitor는 소스 TCP 포트를 주기적으로 전환하여 네트워크 적용 범위를 늘리므로 패킷 손실을 탐지할 가능성이 높아집니다.

AWS 네트워크 상태 지표

Network Monitor는 AWS Direct Connect를 통해 연결된 대상의 네트워크 성능 및 가용성에 대한 정보를 제공하는 네트워크 상태 지표(NHI)를 게시합니다. 지표는 모니터가 배포된 AWS 호스팅 리소스에서 Direct Connect 위치까지 AWS 제어 네트워크 경로의 상태를 통계적으로 측정한 것입니다.

Network Monitor는 이상 탐지를 사용하여 네트워크 경로에 따른 가용성 저하 또는 성능 저하를 계산합니다.

> **Note**
>
> 새 모니터를 생성하거나, 프로브를 추가하거나, 프로브를 다시 활성화할 때마다 AWS가 이상 탐지를 수행하기 위해 데이터를 수집할 수 있도록 해당 모니터에 대한 NHI가 몇 시간씩 지연됩니다.

Network Monitor는 NHI 상태 지표를 제공하기 위해 네트워크 경로를 시뮬레이션하는 트래픽의 패킷 손실 및 왕복 지연 시간 지표뿐만 아니라 AWS 샘플 데이터 세트 전반에 걸쳐 통계적 상관 관계를 적용합니다. 지표는 1 또는 0의 두 변수 중 하나일 수 있습니다. 값 1은 Network Monitor가 AWS 제어 네트워크 경로 내에서 네트워크 성능 저하를 관찰했음을 나타냅니다. 값 0은 Network Monitor가 경로를 따라 네트워크 성능 저하를 관찰하지 않았음을 나타냅니다. 이를 통해 네트워크 문제를 보다 신속하게 해결할 수 있습니다. NHI 지표에 알림을 설정하여 네트워크 경로를 따라 진행 중인 문제에 대한 알림을 받을 수 있습니다.

IPv4 및 IPv6 주소 지원

Network Monitor는 IPv4 또는 IPv6 네트워크를 통한 가용성 및 성능 지표를 제공하며 이중 스택 VPC에서 IPv4 또는 IPv6 주소를 모니터링할 수 있습니다. Network Monitor는 동일한 모니터 내에서 IPv4 및 IPv6 대상을 모두 구성하는 것을 허용하지 않지만 IPv4 전용 및 IPv6 전용 대상에 대해 별도로 생성할 수 있습니다.

리전 가용성

Network Monitor는 현재 다음 AWS 리전에서 사용할 수 있습니다.

지역	
아시아 태평양(홍콩)	ap-east-1
아시아 태평양(뭄바이)	ap-south-1
아시아 태평양(서울)	ap-northeast-2
아시아 태평양(싱가포르)	ap-southeast-1
아시아 태평양(시드니)	ap-southeast-2
아시아 태평양(도쿄)	ap-northeast-1
캐나다 서부(캘거리)	ca-west-1
유럽(프랑크푸르트)	eu-central-1
유럽(아일랜드)	eu-west-1
유럽(런던)	eu-west-2

지역	
유럽(파리)	eu-west-3
유럽(스톡홀름)	eu-north-1
중동(바레인)	me-south-1
남아메리카(상파울루)	sa-east-1
미국 동부(버지니아 북부)	us-east-1
미국 동부(오하이오)	us-east-2
미국 서부(캘리포니아 북부)	us-west-1
미국 서부(오레곤)	us-west-2

Network Monitor 생성

다음 단계에서는 모니터를 생성한 다음 필요한 프로브를 추가하는 방법을 설명합니다. 프로브의 경우 소스 서브넷을 선택하고 모니터당 최대 24개의 프로브에 대해 최대 4개의 대상 IP 주소를 선택합니다. Amazon CloudWatch 콘솔을 사용하거나 명령줄 또는 API를 사용하여 모니터를 생성할 수 있습니다.

주제

- 콘솔을 사용하여 Network Monitor 생성

- 명령줄 또는 API를 사용하여 Network Monitor 생성

콘솔을 사용하여 Network Monitor 생성

다음 단계에서는 Amazon CloudWatch 콘솔을 사용하여 모니터를 생성하는 방법을 설명합니다. 소스 서브넷을 선택한 다음, 대상을 최대 4개까지 추가하여 모니터당 최대 24개의 프로브를 생성할 수 있습니다. Amazon CloudWatch 콘솔을 사용하거나 명령줄 또는 SDK를 사용하여 모니터를 생성할 수 있습니다.

> **Important**
>
> 이 단계는 모두 한 번에 완료할 수 있도록 설계되었습니다. 나중에 계속하기 위해 진행 중인 작업을 저장할 수 없습니다.

모니터 세부 정보 정의

모니터를 생성하는 첫 번째 단계는 기본 세부 정보를 정의하는 것입니다. 여기에는 모니터에 이름을 지정하고 집계 기간을 정의하는 작업이 포함됩니다. 모니터에 선택적 태그를 추가할 수 있습니다.

모니터 세부 정보 정의

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.
2. 모니터 생성(Create monitor)을 선택합니다.
3. 모니터 이름에 이 모니터에 사용할 이름을 입력합니다.
4. 집계 기간에서 CloudWatch에 지표를 전송할 빈도를 선택합니다. 사용 가능한 집계 기간은 다음과 같습니다.

   - 30초
   - 60초

   > **Note**
   >
   > 집계 기간이 짧을수록 네트워크 문제를 더 빨리 탐지할 수 있지만 선택한 집계 기간이 결제 구조에 영향을 미칠 수 있습니다. 요금에 대한 자세한 정보는 Amazon CloudWatch 요금 페이지를 참조하세요.

5. (선택 사항) 태그 섹션에서 이 리소스를 식별하는 데 도움이 되는 키와 값 페어를 추가하여 특정 정보를 검색하거나 필터링할 수 있습니다.

   1. 새 태그 추가를 선택합니다.
   2. 키 이름과 관련 값을 입력합니다.
   3. 새 태그 추가를 선택하여 새 태그를 추가합니다.

      새 태그 추가를 선택하여 여러 태그를 추가하거나 제거를 선택하여 태그를 제거할 수 있습니다.
   4. 태그를 모니터와 연결하려면 모니터에서 생성한 프로브에 태그 추가를 선택한 상태로 유지합니다. 이렇게 하면 모니터 프로브에 태그가 추가되므로 태그 기반 인증 또는 측정을 사용하는 경우 유용할 수 있습니다.
6. 다음을 선택하여 the section called “소스 및 대상 선택”으로 진행합니다.

소스 및 대상 선택

네트워크 모니터는 네트워크가 운영되는 리전의 VPC 및 관련 서브넷의 AWS 소스를 사용합니다. 모니터 대상은 온프레미스 IP 주소, 네트워크 프로토콜, 포트 및 네트워크 패킷 크기의 조합입니다.

소스와 대상의 조합을 프로브라고 합니다. 서브넷당 최대 4개의 프로브, 모니터당 최대 총 24개의 프로브를 가질 수 있습니다.

> **Important**
>
> 이 단계는 모두 한 번에 완료할 수 있도록 설계되었습니다. 나중에 계속하기 위해 진행 중인 작업을 저장할 수 없습니다.

소스 및 대상 선택

1. AWS 네트워크 소스에서 모니터에 포함할 서브넷을 하나 이상 선택합니다. 단일 VPC를 선택하여 해당 VPC 내의 모든 서브넷을 선택하거나 특정 서브넷을 선택할 수 있습니다. 선택한 VPC와 서브넷이 네트워크 모니터의 소스가 됩니다.
2. 대상 1에는 온프레미스 네트워크의 대상 IP 주소를 입력합니다. IPv4 주소와 IPv6 주소 모두 지원됩니다.
3. 고급 설정(Advanced Settings)을 선택합니다.
4. 고객 관리형 대상에 대해 네트워크 프로토콜을 선택합니다. 다음 중 하나일 수 있습니다.

   - ICMP

- TCP

5. 프로토콜이 TCP인 경우 다음 정보를 입력합니다. 그렇지 않은 경우 다음 단계로 건너뜁니다.

   1. 네트워크가 연결하는 데 사용하는 포트를 입력합니다. 포트는 1에서 65,535 사이의 숫자여야 합니다.
   2. 패킷 크기를 입력합니다. 이는 소스와 대상 사이의 프로브에서 전송되는 각 패킷의 크기(바이트)입니다. 패킷 크기는 56에서 8,500 사이의 숫자여야 합니다.

6. 대상 추가를 선택하여 이 모니터에 다른 온프레미스 대상을 추가합니다. 추가하려는 각 대상에 대해 이러한 단계를 반복합니다.
7. 완료되면 다음을 선택하여 프로브를 확인합니다.

프로브 확인

프로브를 확인하면 모니터의 네트워크 프로브 조합을 검토할 수 있습니다. 이 페이지에는 선택한 소스와 대상의 가능한 모든 조합이 표시됩니다. 예를 들어, 6개의 소스 서브넷과 4개의 대상 IP가 있는 경우 총 24개의 프로브 조합이 가능합니다.

> **Important**
>
> - 이 단계는 모두 한 번에 완료할 수 있도록 설계되었습니다. 나중에 계속하기 위해 진행 중인 작업을 저장할 수 없습니다.
> - 프로브 확인 페이지에는 프로브가 유효한지 여부가 표시되지 않습니다. 따라서 이 페이지를 자세히 검토하고 유효하지 않은 프로브를 삭제하는 것이 좋습니다. 유효하지 않은 프로브를 제거하지 않으면 이에 대한 요금이 부과될 수 있습니다.

모니터 프로브 확인

1. 사전 조건: the section called “소스 및 대상 선택”
2. 프로브 확인 페이지에서 소스 및 대상 조합 목록을 검토합니다.
3. 모니터에서 제거하려는 프로브를 하나 이상 선택한 다음, 제거를 선택합니다.

> **Note**
>
> 삭제를 확인하는 메시지가 표시되지 않습니다. 프로브가 삭제된 후에는 다시 설정해야 합니다. 네트워크 모니터 페이지의 네트워크 모니터 섹션에서 프로브를 모니터에 다시 추가

할 수 있습니다. 자세한 내용은 the section called “모니터에 프로브 추가” 단원을 참조하십시오.

4. 모니터를 생성하기 전에 다음을 선택하여 모니터 세부 정보를 검토합니다.

검토 및 생성

모니터와 프로브를 생성하는 마지막 단계는 모니터와 프로브의 세부 정보를 검토하는 것입니다. 이 시점에서 모든 정보를 변경할 수 있습니다. 검토를 마치고 모니터를 생성하고 지표를 추적하기 시작하면 모든 프로브에 대한 요금이 청구됩니다.

Important

- 이 단계는 모니터와 프로브를 생성할 때 한 번에 모두 완료되도록 설계되었습니다. 나중에 계속하기 위해 진행 중인 작업을 저장할 수 없습니다.
- 섹션을 편집하기로 선택한 경우 편집 중인 시점부터 모니터 생성을 단계별로 진행해야 합니다. 하지만 후속 단계를 다시 생성할 필요는 없습니다. 이러한 페이지는 이전에 입력된 정보를 유지합니다.

모니터 검토 및 생성

1. 프로브 검토 및 생성 페이지에서 변경하려는 섹션의 편집을 선택합니다.
2. 해당 섹션에서 변경을 수행합니다.
3. 다음을 선택합니다.
4. 다음을 수행합니다.
   - 추가 모니터 페이지에서 원하는 대로 변경하고 검토 및 생성 페이지로 돌아올 때까지 다음을 선택합니다.
   - 변경이 필요한 다른 페이지가 없으면 검토 및 생성 페이지로 돌아갈 때까지 다음을 선택합니다.
5. 모니터 생성(Create monitor)을 선택합니다.

   네트워크 모니터 페이지에는 네트워크 모니터 섹션의 모니터 생성의 현재 상태가 표시됩니다. 모니터를 생성하는 동안 상태는 보류 중입니다. 상태가 활성으로 변경되면 모니터 대시보드에 액세스하여 CloudWatch 지표를 볼 수 있습니다.

모니터 대시보드 작업에 대한 자세한 내용은 the section called “네트워크 모니터 대시보드” 섹션을 참조하세요.

> **Note**
>
> 새로 추가된 네트워크 모니터에서 네트워크 지표 수집을 시작하는 데 몇 분 정도 걸릴 수 있습니다.

명령줄 또는 API를 사용하여 Network Monitor 생성

명령줄 또는 API를 사용하여 네트워크 모니터를 보고 생성합니다.

명령줄 또는 API를 사용하여 네트워크 모니터 생성

1. create-monitor를 사용하여 네트워크 모니터를 생성합니다.
2. create-probe를 사용하여 네트워크 모니터 프로브를 생성합니다.

Network Monitor 모니터 및 프로브 작업

Amazon CloudWatch 콘솔을 사용하거나 명령줄 또는 API를 사용하여 모니터와 프로브로 다음 작업을 수행할 수 있습니다.

주제:

- 모니터 편집
- 모니터 삭제
- 프로브 활성화 또는 비활성화
- 모니터에 프로브 추가
- 프로브 편집
- 프로브 삭제
- 명령줄 또는 API를 사용하여 리소스에 태그 지정 또는 태그 해제

모니터 편집

이름 변경, 새 집계 기간 설정, 태그 추가 또는 제거 등 Network Monitor에 대한 모든 정보를 편집할 수 있습니다. 모니터 정보를 변경해도 연결된 프로브는 변경되지 않습니다. Amazon CloudWatch 콘솔을 사용하거나 명령줄 또는 API를 사용하여 모니터를 편집할 수 있습니다.

콘솔을 사용하여 모니터 편집

CloudWatch 콘솔을 사용하여 모니터를 편집합니다.

콘솔을 사용하여 모니터 편집

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.
2. 네트워크 모니터 섹션에서 편집하려는 모니터를 선택합니다.
3. 모니터 대시보드 페이지에서 편집을 선택합니다.
4. 모니터 이름에 모니터의 새 이름을 입력합니다.
5. 집계 기간에서 CloudWatch에 지표를 전송할 빈도를 선택합니다. 유효 기간은 다음과 같습니다.
   - 30초
   - 60초

   > **Note**
   > 집계 기간이 짧을수록 네트워크 문제를 더 빨리 탐지할 수 있지만 선택한 집계 기간이 결제 구조에 영향을 미칠 수 있습니다. 요금에 대한 자세한 정보는 Amazon CloudWatch 요금 페이지를 참조하세요.

6. (선택 사항) 태그 섹션에서 이 리소스를 식별하는 데 도움이 되는 키와 값 페어를 추가하여 특정 정보를 검색하거나 필터링할 수 있습니다. 현재 키의 값만 변경할 수도 있습니다.
   1. 새 태그 추가를 선택합니다.
   2. 키 이름과 관련 값을 입력합니다.
   3. 새 태그 추가를 선택하여 새 태그를 추가합니다.

      새 태그 추가를 선택하여 여러 태그를 추가하거나 제거를 선택하여 태그를 제거할 수 있습니다.

4. 태그를 모니터와 연결하려면 모니터에서 생성한 프로브에 태그 추가를 선택한 상태로 유지합니다. 이렇게 하면 모니터 프로브에 태그가 추가되므로 태그 기반 인증 또는 측정을 사용하는 경우 유용할 수 있습니다.

7. Save changes(변경 사항 저장)를 선택합니다.

CLI 또는 API를 사용하여 모니터 편집

명령줄 또는 API를 사용하여 모니터를 보고 편집합니다.

명령줄 또는 API를 사용하여 모니터 편집

1. 모니터 이름을 모르는 경우 list-monitors를 사용하여 모니터 목록을 가져옵니다. 편집할 모니터의 이름을 기록해 둡니다.
2. 이전 단계의 모니터 이름을 사용하여 edit-monitor를 사용합니다.

모니터 삭제

모니터를 삭제하려면 먼저 모니터 상태와 상관없이 해당 모니터와 연결된 모든 프로브를 비활성화하거나 삭제해야 합니다. 모니터가 비활성화되거나 삭제된 후에는 해당 모니터 프로브에 대한 요금이 더 이상 청구되지 않습니다. 삭제된 모니터는 복원할 수 없습니다. Amazon CloudWatch 콘솔을 사용하거나 명령줄 또는 API를 사용하여 모니터를 삭제할 수 있습니다.

프로브가 삭제되거나 비활성화될 수 있지만 CloudWatch는 여전히 15일 동안 지표를 보존합니다.

콘솔을 사용하여 모니터 삭제

CloudWatch 콘솔을 사용하여 모니터를 삭제합니다.

콘솔을 사용하여 모니터 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.
2. 네트워크 모니터 섹션에서 삭제하려는 모니터를 선택합니다.
3. 작업을 선택한 후 삭제를 선택합니다.
4. 활성 프로브가 있는 경우 프로브를 비활성화하라는 메시지가 나타납니다. 프로브 비활성화를 선택합니다.

> **Note**
>
> 프로브 비활성화를 선택한 후에는 이 작업을 취소할 수 없습니다. 그러나 비활성화된 프로브가 모니터에서 제거되지는 않습니다. 나중에 프로브를 다시 활성화할 수 있습니다. the section called “프로브 활성화 또는 비활성화” 섹션을 참조하세요.

5. 확인 필드에 **`confirm`**을 입력한 다음, 삭제를 선택합니다.

명령줄 또는 API를 사용하여 모니터 삭제

명령줄 또는 API를 사용하여 모니터를 삭제합니다.

명령줄 또는 API를 사용하여 네트워크 모니터 삭제

1. 삭제할 모니터의 이름이 필요합니다. 이름을 모르는 경우 list-monitors를 사용하여 모니터 목록을 가져옵니다. 삭제할 모니터의 이름을 기록해 둡니다.
2. 모니터에 프로브가 있는지 확인합니다. 이전 단계의 모니터 이름과 함께 get-monitor를 사용합니다. 그러면 해당 모니터와 연결된 모든 프로브 목록이 반환됩니다.
3. 모니터에 프로브가 포함된 경우 먼저 해당 프로브를 비활성으로 설정하거나 삭제해야 합니다.
   - 프로브를 비활성으로 설정하려면 update-probe를 사용하고 상태를 `INACTIVE`로 설정합니다.
   - 프로브를 삭제하려면 delete-probe를 사용합니다.
4. 프로브가 `INACTIVE`로 설정되거나 삭제되면 delete-monitor를 사용하여 모니터를 삭제합니다. 비활성 프로브는 삭제되지 않습니다.

프로브 활성화 또는 비활성화

필요에 따라 모니터 프로브를 활성화하거나 비활성화할 수 있습니다. 현재 사용하고 있지 않지만 나중에 다시 사용하고 싶을 경우 프로브를 비활성화할 수 있습니다. 프로브를 비활성화하면 다시 설정하는 데 시간을 들일 필요가 없습니다. 비활성화된 프로브에 대해서는 요금이 청구되지 않습니다.

Amazon CloudWatch 콘솔을 사용하거나 명령줄 또는 API를 사용하여 모니터의 상태를 변경할 수 있습니다.

콘솔을 사용하여 활성 또는 비활성으로 프로브 설정

CloudWatch 콘솔을 사용하여 프로브를 활성 또는 비활성으로 설정합니다.

콘솔을 사용하여 활성 또는 비활성으로 프로브 설정

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.
2. 모니터 세부 정보 탭을 선택합니다.
3. 프로브 섹션에서 활성화하거나 비활성화하려는 프로브를 선택합니다.
4. 작업을 선택한 다음, 활성화 또는 비활성화를 선택합니다.

> **Note**
>
> 비활성화된 프로브를 다시 활성화하면 해당 프로브에 대한 요금이 청구되기 시작합니다.

명령줄 또는 API를 사용하여 활성 또는 비활성으로 프로브 설정

명령줄 또는 API를 사용하여 프로브를 활성 또는 비활성으로 설정하거나 비활성화합니다. 단일 프로브에만 이 명령을 사용할 수 있습니다.

명령줄 또는 API를 사용하여 활성 또는 비활성으로 프로브 설정

1. 모니터 이름을 모르는 경우 list-monitors를 사용하여 모니터 목록을 가져옵니다. 프로브 상태를 변경하려는 모니터의 이름을 기록해 둡니다.
2. 이전 단계의 모니터 이름과 함께 get-monitor를 사용합니다. 그러면 해당 모니터와 연결된 모든 프로브 목록이 반환됩니다. 상태를 변경하려는 프로브의 프로브 ID를 기록해 둡니다.
3. update-probe를 사용하고 상태를 `ACTIVE` 또는 `INACTIVE`로 변경하려는 프로브를 설정합니다.

모니터에 프로브 추가

기존 모니터에 프로브를 추가할 수 있습니다. 모니터에 프로브를 추가하면 결제 구조가 업데이트되어 새 프로브가 추가되었음을 표시합니다.

콘솔을 사용하여 모니터에 프로브 추가

콘솔을 사용하여 모니터에 프로브 추가

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.
2. 네트워크 모니터 섹션에서 다음 중 하나를 수행합니다.

- 프로브를 추가할 모니터의 이름 링크를 선택합니다. 모니터 세부 정보 탭을 선택한 다음, 프로브 섹션에서 프로브 추가를 선택합니다.
- 모니터 확인란을 선택하고 작업을 선택한 다음, 프로브 추가를 선택합니다.

3. 프로브 추가 페이지에서 다음을 수행합니다.

   1. AWS 네트워크 소스에서 모니터에 추가할 서브넷을 선택합니다.

      > **Note**
      >
      > 한 번에 하나의 프로브만 추가할 수 있으며 모니터당 최대 4개의 프로브를 추가할 수 있습니다.

   2. 온프레미스 네트워크의 대상 IP 주소를 입력합니다. IPv4 주소와 IPv6 주소 모두 지원됩니다.
   3. 고급 설정(Advanced Settings)을 선택합니다.
   4. 대상의 네트워크 프로토콜을 선택합니다. ICMP 또는 TCP일 수 있습니다.
   5. 프로토콜이 TCP인 경우 다음 정보를 입력합니다. 그렇지 않은 경우 다음 단계로 건너뜁니다.
      - 네트워크가 연결하는 데 사용하는 포트를 입력합니다. 포트는 1에서 65,535 사이의 숫자여야 합니다.
      - 패킷 크기를 입력합니다. 이는 소스와 대상 사이의 프로브를 따라 전송되는 각 패킷의 크기(바이트)입니다. 패킷 크기는 56에서 8,500 사이의 숫자여야 합니다.

4. (선택 사항) 태그 섹션에서 이 리소스를 식별하는 데 도움이 되는 키와 값 페어를 추가하여 특정 정보를 검색하거나 필터링할 수 있습니다.

   1. 새 태그 추가를 선택합니다.
   2. 키 이름과 관련 값을 입력합니다.
   3. 새 태그 추가를 선택하여 새 태그를 추가합니다.

      새 태그 추가를 선택하여 여러 태그를 추가하거나 제거를 선택하여 태그를 제거할 수 있습니다.

5. 프로브 추가를 선택합니다.

   프로브가 활성화되는 동안에는 상태가 보류 중으로 표시됩니다. 프로브가 활성 상태가 되는 데 몇 분 정도 걸릴 수 있습니다.

명령줄 또는 API를 사용하여 모니터에 프로브 추가

명령줄 또는 API를 사용하여 모니터에 프로브를 추가합니다. 이 명령은 한 번에 하나의 프로브를 추가하는 데만 사용할 수 있습니다.

명령줄 또는 API를 사용하여 모니터에 프로브 추가

1. 모니터 이름을 모르는 경우 list-monitors를 사용하여 모니터 목록을 가져옵니다. 프로브를 추가하려는 모니터의 이름을 기록해 둡니다.
2. create-probe를 사용하여 모니터에 프로브를 추가합니다.

프로브 편집

해당 프로브의 활성화 여부에 관계없이 현재 프로브에 대한 모든 정보를 변경할 수 있습니다. Amazon CloudWatch 콘솔을 사용하거나 명령줄 또는 API를 사용하여 프로브를 편집할 수 있습니다.

콘솔을 사용하여 프로브 편집

콘솔을 사용하여 프로브 편집

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.

   이름 링크를 선택하여 모니터 대시보드를 엽니다.
2. 모니터 세부 정보 탭을 선택합니다.
3. 프로브 섹션에서 편집하려는 프로브의 링크를 선택합니다.
4. 프로브 대시보드 페이지에서 편집을 선택하거나 작업을 선택한 다음, 편집을 선택합니다.
5. 프로브 편집 페이지에서 새 대상 프로브 IP 주소를 입력합니다. IPv4 주소와 IPv6 주소 모두 지원됩니다.
6. 고급 설정(Advanced Settings)을 선택합니다.
7. 네트워크 프로토콜을 선택합니다. ICMP 또는 TCP일 수 있습니다.
8. 프로토콜이 TCP인 경우 다음 정보를 입력합니다. 그렇지 않은 경우 다음 단계로 건너뜁니다.

   - 네트워크가 연결하는 데 사용하는 포트를 입력합니다. 포트는 1에서 65,535 사이의 숫자여야 합니다.
   - 패킷 크기를 입력합니다. 이는 소스와 대상 사이의 프로브를 따라 전송되는 각 패킷의 크기(바이트)입니다. 패킷 크기는 56에서 8,500 사이의 숫자여야 합니다.

9. (선택 사항) 태그 섹션에서 이 리소스를 식별하는 데 도움이 되는 키와 값 페어를 추가하여 특정 정보를 검색하거나 필터링할 수 있습니다.

    1. 새 태그 추가를 선택합니다.
    2. 키 이름과 관련 값을 입력합니다.
    3. 새 태그 추가를 선택하여 새 태그를 추가합니다.

    새 태그 추가를 선택하여 여러 태그를 추가하거나 제거를 선택하여 태그를 제거할 수 있습니다.

10. Save changes(변경 사항 저장)를 선택합니다.

명령줄 또는 API를 사용하여 프로브 편집

명령줄을 사용하여 모니터 프로브를 편집합니다. 단일 프로브에만 이 명령을 사용할 수 있습니다.

명령줄 또는 API를 사용하여 프로브 편집

1. 모니터 이름을 모르는 경우 list-monitors를 사용하여 모니터 목록을 가져옵니다. 프로브 상태를 변경하려는 모니터의 이름을 기록해 둡니다.
2. 이전 단계의 모니터 이름과 함께 get-monitor를 사용합니다. 그러면 해당 모니터와 연결된 모든 프로브 목록이 반환됩니다. 편집하려는 프로브의 프로브 ID를 기록해 둡니다.
3. update-probe를 사용하여 프로브의 정보를 변경합니다.

프로브 삭제

프로브가 나중에 다시 필요하지 않을 것임을 알고 있는 경우 프로브를 비활성화하는 대신 삭제할 수 있습니다. 삭제된 프로브는 복구할 수 없으며 대신 다시 생성해야 합니다. 프로브가 삭제되면 해당 프로브에 대한 결제가 중지됩니다. Amazon CloudWatch 콘솔이나 명령줄 또는 API를 사용하여 프로브를 삭제할 수 있습니다.

콘솔을 사용하여 프로브 삭제

콘솔을 사용하여 프로브 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.
2. 네트워크 모니터 섹션에서 이름 링크를 선택하여 모니터 대시보드를 엽니다.
3. 모니터 세부 정보 탭을 선택합니다.

4. 모니터 확인란을 선택하고 작업을 선택한 다음, 삭제를 선택합니다.
5. 프로브 삭제 대화 상자에서 삭제를 선택하여 프로젝트 삭제를 확인합니다.
6. 삭제를 선택하여 프로브 삭제를 확인합니다.

   프로브 섹션의 프로브 상태에 삭제 중이 표시됩니다. 삭제된 후에는 프로브 섹션에서 프로브가 제거됩니다.

명령줄 또는 API를 사용하여 프로브 삭제

명령줄 또는 API를 사용하여 프로브를 삭제합니다. 단일 프로브에만 이 명령을 사용할 수 있습니다.

명령줄 또는 API를 사용하여 활성 또는 비활성으로 프로브 설정

1. 모니터 이름을 모르는 경우 list-monitors를 사용하여 모니터 목록을 가져옵니다. 삭제하려는 프로브가 있는 모니터의 이름을 기록해 둡니다.
2. 이전 단계의 모니터 이름과 함께 get-monitor를 사용합니다. 그러면 해당 모니터와 연결된 모든 프로브 목록이 반환됩니다. 삭제하려는 프로브의 프로브 ID를 기록해 둡니다.
3. delete-probe를 사용합니다.

명령줄 또는 API를 사용하여 리소스에 태그 지정 또는 태그 해제

명령줄 또는 CLI를 사용하여 리소스 태그를 추가하거나 업데이트할 수 있습니다.

명령줄 또는 API를 사용하여 네트워크 모니터 태그 업데이트

- 리소스의 태그를 나열하려면 list-tags-for-resources를 사용합니다.
- 리소스에 태그를 지정하려면 tag-resource를 사용합니다.
- 리소스의 태그를 해제하려면 untag-resource를 사용합니다.

네트워크 모니터 대시보드

Amazon CloudWatch 네트워크 모니터 대시보드를 사용하여 AWS 네트워크 상태를 보고 왕복 시간과 패킷 손실을 조사할 수 있습니다. 모니터와 개별 프로브 모두에 대해 이러한 지표를 볼 수 있습니다.

네트워크 모니터 대시보드

- 모니터 대시보드
- 프로브 대시보드

프로브 경보

다른 Amazon CloudWatch 지표와 마찬가지로 Amazon CloudWatch Network Monitor 지표를 기반으로 Amazon CloudWatch 경보를 생성할 수 있습니다. 생성한 모든 경보는 경보가 트리거될 때 네트워크 모니터 대시보드에 있는 모니터 세부 정보 섹션의 프로브 상태 열에 표시됩니다. 상태는 정상 또는 경보 내로 표시됩니다. 프로브에 대한 상태가 표시되지 않으면 해당 프로브에 대한 경보가 생성되지 않은 것입니다.

예를 들어 Network Monitor 지표 `PacketLoss`를 기반으로 경보를 생성하고 지표가 선택한 값보다 높을 때 알림을 보내도록 구성할 수 있습니다. 다른 CloudWatch 지표와 동일한 지침에 따라 Network Monitor 지표에 대한 경보를 구성합니다.

네트워크 모니터에 대한 CloudWatch 경보를 생성할 때 `AWS/NetworkMonitor`에서 다음 지표를 사용할 수 있습니다.

- HealthIndicator
- PacketLoss
- 왕복 시간(RTT)

네트워크 모니터 경보를 생성하는 단계는 the section called “정적 임곗값을 기반으로 경보 생성” 섹션을 참조하세요.

지표 기간 설정

두 대시보드의 지표와 이벤트는 현재 시간을 기준으로 계산된 기본 시간인 2시간을 사용합니다. 다음 사전 설정 중 하나를 사용하도록 기본값을 변경할 수 있습니다.

- 1h - 1시간
- 2h - 2시간
- 1d - 1일
- 1w - 1주일

사용자 지정 기간을 설정할 수도 있습니다. 사용자 지정을 선택하고 절대 또는 상대 시간을 선택한 다음, 기간을 원하는 시간으로 설정합니다. 상대 시간은 CloudWatch 기본값에 따라 오늘 날짜로부터 15일 전까지만 지원합니다.

또한 UTC 시간대 또는 현지 시간대를 기준으로 차트에 표시되는 시간을 선택할 수 있습니다.

모니터 대시보드

Amazon CloudWatch 네트워크 모니터 대시보드를 사용하여 AWS 네트워크 상태를 보고 왕복 시간과 패킷 손실을 조사할 수 있습니다. 네트워크 모니터에는 모니터와 프로브 모두에 대한 대시보드가 있습니다.

모니터 대시보드에 액세스

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.
2. 네트워크 모니터 섹션에서 이름 링크를 선택하여 모니터 대시보드를 엽니다.

개요

개요 페이지에 모니터에 대한 다음 정보가 표시됩니다.

- AWS 네트워크 상태 - AWS 네트워크 상태는 AWS 네트워크의 전체 상태만 표시합니다. 상태는 정상 또는 성능 저하됨입니다. 정상 상태이면 AWS 네트워크에서 어떤 문제도 관찰되지 않은 것입니다. 성능 저하됨 상태이면 AWS 네트워크에서 문제가 관찰된 것입니다. 이 섹션의 상태 표시줄에는 기본 시간인 1시간 동안의 네트워크 상태가 표시됩니다. 상태 표시줄의 아무 지점이나 가리키면 자세한 내용을 볼 수 있습니다.
- 프로브 트래픽 요약 - 모니터의 소스 AWS 서브넷과 대상 IP 주소 간의 현재 트래픽 상태를 표시합니다. 프로브 트래픽 요약에는 다음이 표시됩니다.
  - 알람 프로브 - 이 숫자는 성능 저하 상태에 있는 프로브 수를 나타냅니다. 경보로 설정한 지표가 트리거되면 경보가 트리거됩니다. 네트워크 모니터 지표 경보에 대한 자세한 내용은 the section called “프로브 경보” 섹션을 참조하세요.
  - 패킷 손실 - 소스 서브넷과 대상 IP 주소 사이에서 손실된 패킷 수입니다. 이는 전송된 총 패킷의 백분율로 표시됩니다.
  - 왕복 시간 - 소스 서브넷의 패킷이 대상 IP 주소에 도달한 후 다시 돌아오는 데 걸리는 시간(밀리초)입니다.

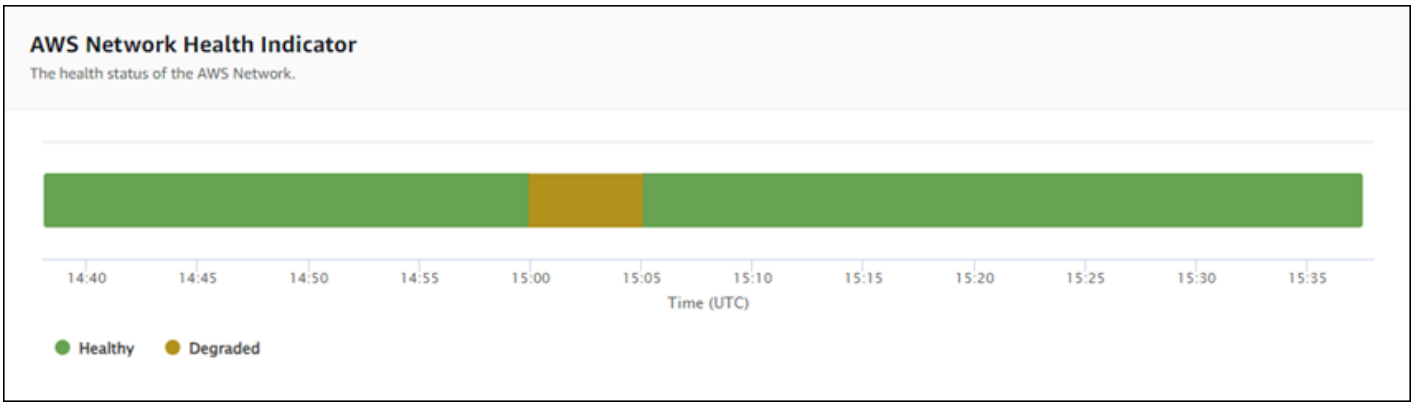
대화형 그래프로 데이터가 표시되어 세부 정보를 볼 수 있습니다.

기본적으로 현재 날짜 및 시간을 기준으로 계산된 2시간 기간에 대한 데이터가 표시됩니다. 그러나 요구 사항에 맞게 범위를 변경할 수 있습니다. 자세한 내용은 the section called “지표 기간 설정” 단원을 참조하십시오.

지표 추적

네트워크 모니터 대시보드에는 모니터와 프로브가 그래픽으로 표시됩니다. 다음과 같은 그래프를 사용할 수 있습니다.

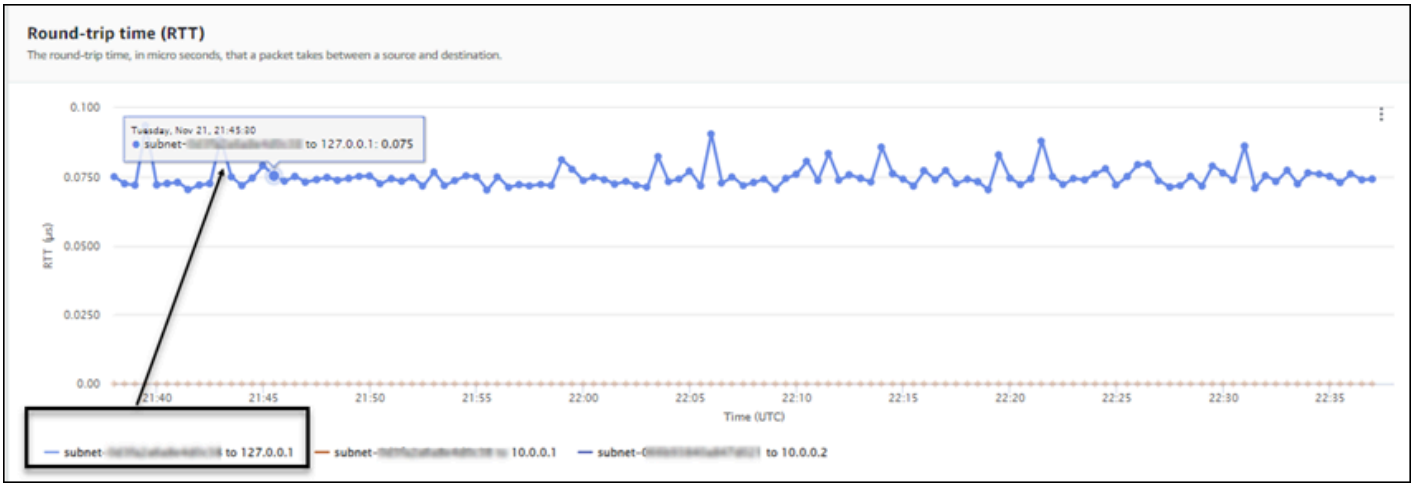
- AWS 네트워크 상태 지표 - 지정된 기간 동안의 AWS 네트워크 상태를 나타냅니다. 상태는 정상 또는 성능 저하됨입니다. 다음 예제에서는 15:00 UTC부터 15:05 UTC까지 AWS 네트워크의 성능 저하됨 상태임을 알 수 있습니다. 15:05 후 네트워크가 정상 상태로 돌아왔습니다. 그래프의 아무 부분이나 가리키면 추가 세부 정보를 볼 수 있습니다.



> **Note**
>
> 네트워크 상태 지표는 프로브의 상태를 나타내지 않고 AWS 네트워크의 상태만 나타냅니다.

- 패킷 손실 - 이 그래프에는 모니터에 있는 각 프로브의 패킷 손실율을 나타내는 고유한 선이 표시됩니다. 페이지 하단의 범례에는 모니터에 있는 각 프로브가 고유성에 따라 색상으로 구분되어 표시됩니다. 이 차트에서 프로브를 가리키면 소스 서브넷, 대상 IP 및 패킷 손실률이 표시됩니다. 다음 예제에서는 서브넷에서 IP 주소 127.0.0.1로의 프로브에 대해 패킷 손실 경보가 설정되었습니다. 프로브의 패킷 손실 임곗값이 초과될 때 경보가 트리거되었습니다. 그래프를 가리키면 프로브 소스와 대상이 표시되며, 11월 21일 02:41:30에 이 프로브에 대해 30.97%의 패킷 손실이 발생했음을 알 수 있습니다.

- 왕복 시간 - 이 그래프에는 각 프로브에 대한 선이 표시되어 각 프로브의 왕복 시간을 보여줍니다. 페이지 하단의 범례에는 모니터에 있는 각 프로브가 고유성에 따라 색상으로 구분되어 표시됩니다. 이 차트에서 프로브를 가리키면 소스 서브넷, 대상 IP 주소 및 왕복 시간이 표시됩니다. 다음 예제는 11월 21일 화요일 21:45:30에 서브넷에서 IP 주소 127.0.0.1까지의 프로브 왕복 시간이 0.075초임을 보여줍니다.



모니터 세부 정보

모니터 세부 정보 페이지에는 프로브를 포함하여 모니터에 대한 세부 정보가 표시됩니다. 이 페이지에서 태그를 관리하거나 프로브를 추가할 수 있습니다. 이 페이지는 다음 세 섹션으로 구분됩니다.

- 모니터 세부 정보 - 이 페이지에서는 모니터에 대한 세부 정보를 제공합니다. 이 섹션의 정보는 편집할 수 없습니다. 하지만 역할 이름 링크를 선택하여 Network Monitor 서비스 연결 역할의 세부 정보를 볼 수 있습니다.
- 프로브 - 이 섹션에는 모니터와 연결된 모든 프로브의 목록이 표시됩니다. VPC 또는 서브넷 ID 링크를 선택하여 Amazon VPC 콘솔에서 VPC 또는 서브넷 세부 정보를 엽니다. 프로브를 활성화하거나 비활성화하는 등 프로브를 수정할 수도 있습니다. 자세한 내용은 the section called “모니터 및 프로브 사용” 단원을 참조하십시오.

  프로브 섹션에는 해당 모니터에 설정된 각 프로브에 대해 프로브 ID, VPC ID, 서브넷 ID, IP 주소, 프로토콜, 프로브 상태(활성 또는 비활성)를 포함한 정보가 표시됩니다. 프로브에 대한 경보를 설정한 경우 해당 경보의 현재 상태가 표시됩니다. 정상은 지표 이벤트가 경보를 트리거하지 않았음을 나타냅니다. 경보 내는 CloudWatch에서 설정한 지표가 경보를 트리거했음을 나타냅니다. 프로브 상태가 표시되지 않으면 CloudWatch 경보가 설정되지 않은 것입니다. 생성할 수 있는 네트워크 모니터 프로브 경보 유형에 대한 자세한 내용은 the section called “프로브 경보” 섹션을 참조하세요.
- 태그 - 모니터의 현재 태그를 봅니다. 태그 관리를 선택하여 태그를 추가하거나 제거할 수 있습니다. 그러면 프로브 편집 페이지가 열립니다. 태그 편집에 대한 자세한 내용은 the section called “모니터 편집” 섹션을 참조하세요.

프로브 대시보드

Amazon CloudWatch Network Monitor 대시보드를 사용하여 AWS 네트워크 상태와 특정 프로브의 특정 왕복 시간 및 패킷 손실에 대한 정보를 볼 수 있습니다. 개요와 프로브 세부 정보의 두 가지 프로브 대시보드가 있습니다.

CloudWatch 경보를 생성하여 패킷 손실 및 왕복 시간 지표 임곗값을 설정할 수 있습니다. 지표에 대한 임곗값에 도달하면 CloudWatch 경보가 이를 알려줍니다. 프로브 경보 생성에 대한 자세한 내용은 the section called “프로브 경보” 섹션을 참조하세요.

프로브 대시보드에 액세스

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 네트워크 모니터링에서 네트워크 모니터를 선택합니다.
2. 네트워크 모니터 섹션에서 이름 링크를 선택하여 모니터 대시보드를 엽니다.

3. ID 링크를 선택하여 해당 프로브에 대한 대시보드를 봅니다.

개요

개요 페이지에 프로브에 대한 다음 정보가 표시됩니다.

- AWS 네트워크 상태 지표 세부 정보 - AWS 네트워크의 전체 상태만 제공합니다. 상태는 정상 또는 성능 저하됨입니다. 성능 저하됨 상태는 AWS 네트워크에 문제가 있음을 나타내며 프로브에 문제가 있는지 여부를 나타내지는 않습니다.
- 패킷 손실 - 소스 서브넷에서 이 프로브의 대상 IP 주소 사이에서 손실된 패킷 수입니다.
- 왕복 시간 - 소스 서브넷의 패킷이 대상 IP 주소에 도달한 후 다시 돌아오는 데 걸린 시간(밀리초)입니다.

프로브 세부 정보

프로브 세부 정보 페이지에는 프로브에 대한 세부 정보가 표시됩니다. 이 페이지에서 프로브를 편집할 수 있습니다. 자세한 내용은 the section called “모니터 및 프로브 사용” 단원을 참조하십시오.

- 프로브 세부 정보 - 이 페이지는 프로브에 대한 일반 정보를 제공합니다. 이 섹션의 정보는 편집할 수 없습니다.
- 프로브 소스 및 대상 - 이 섹션에는 프로브에 대한 세부 정보가 표시됩니다. VPC 또는 서브넷 ID 링크를 선택하여 Amazon VPC 콘솔에서 VPC 또는 서브넷 세부 정보를 엽니다. 프로브를 활성화하거나 비활성화하는 등 프로브를 수정할 수도 있습니다.
- 태그 - 모니터의 현재 태그를 봅니다. 태그 관리를 선택하여 태그를 추가하거나 제거할 수 있습니다. 그러면 프로브 편집 페이지가 열립니다. 태그 편집에 대한 자세한 내용은 the section called “프로브 편집 ” 섹션을 참조하세요.

Network Monitor 할당량

네트워크 모니터 할당량은 다음과 같습니다.

할당량	기본값	조정 가능
AWS 리전당 계정별 최대 모니터 수	100	예
모니터당 최대 프로브 수	24	예

할당량	기본값	조정 가능
모니터당 서브넷별 최대 프로브 수	4	예

네트워크 모니터의 데이터 보안 및 데이터 보호

AWS에서 클라우드 보안은 가장 중요합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 사용자의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드의 보안 - AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호합니다. AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS규정 준수 프로그램의 일환으로 보안 효과를 정기적으로 테스트하고 검증합니다. Amazon CloudWatch Network Monitor에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 AWS 범위 내 서비스를 참조하세요.
- 클라우드 내 보안 – 귀하의 책임은 귀하가 사용하는 AWS서비스에 의해 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 CloudWatch Network Monitor를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 CloudWatch Network Monitor를 구성하는 방법을 보여줍니다. 또한 CloudWatch Network Monitor 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

- Amazon CloudWatch Network Monitor의 데이터 보호
- Amazon CloudWatch Network Monitor의 인프라 보안

Amazon CloudWatch Network Monitor의 데이터 보호

AWS 공동 책임 모델은 Amazon CloudWatch Network Monitor의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과

관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그에서 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터를 보호하려면 AWS 계정보안 인증 정보를 보호하고 AWS IAM Identity Center또는 AWS Identity and Access Management(IAM)를 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신하세요. TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정하세요.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용하세요.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 FIPS(Federal Information Processing Standard) 140-2를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 CloudWatch Network Monitor 또는 기타 AWS 서비스 서비스에서 콘솔, API, AWS CLI 또는 AWS SDK를 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 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

Amazon CloudWatch Network Monitor의 인프라 보안

관리형 서비스인 Amazon CloudWatch Network Monitor는 Amazon Web Services: Overview of Security Processes 백서에 설명된 AWS 글로벌 네트워크 보안 절차에 따라 보호됩니다.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 CloudWatch Network Monitor에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

Amazon CloudWatch Network Monitor에 대한 자격 증명 및 액세스 관리

AWS Identity and Access Management(IAM)는 AWS 리소스에 대한 관리자의 액세스를 안전하게 제어하는 데 도움이 되는 AWS 서비스입니다. IAM 관리자는 CloudWatch Network Monitor 리소스를 사용하도록 인증(로그인) 및 권한(권한 있음)을 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다. IAM의 기능을 사용하면 보안 자격 증명을 공유하지 않고도 다른 사용자, 서비스 및 애플리케이션이 AWS 리소스를 완전히 또는 제한된 방식으로 사용할 수 있습니다.

기본적으로 IAM 사용자는 AWS 리소스를 생성, 확인 또는 수정할 수 있는 권한이 없습니다. IAM 사용자가 글로벌 네트워크와 같은 리소스에 액세스하고 작업을 수행할 수 있도록 하려면 다음을 수행해야 합니다.

- 사용자에게 필요한 API 작업 및 특정 리소스를 사용할 권한을 부여하는 IAM 정책을 생성합니다.
- 그런 다음 정책을 IAM 사용자 또는 사용자가 속한 그룹에 연결합니다.

사용자 또는 사용자 그룹에 정책을 연결하면 해당 정책은 지정된 리소스에서 지정된 작업을 수행할 권한을 사용자에게 허용하거나 거부합니다.

조건 키

`Condition` 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 선택 사항입니다. 같음, 미만 등의 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 빌드할 수 있습니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM JSON 정책 요소: Condition 연산자를 참조하세요.

한 문에서 여러 `Condition`요소를 지정하거나 단일 `Condition`요소에서 여러 키를 지정하는 경우 AWS는 논리적 `AND`태스크를 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS는 논리적 `OR`태스크를 사용하여 조건을 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다.

태그를 CloudWatch Network Monitor 리소스에 연결하거나 요청을 통해 태그를 Cloud WAN에 전달할 수 있습니다. 태그를 기반으로 액세스를 제어하려면 `aws:ResourceTag/key-name`,

`aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에서 태그 정보를 제공합니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM JSON 정책 요소: Condition을 참조하세요.

모든 AWS 전역 조건 키를 보려면 AWS Identity and Access Management 사용 설명서의 AWS 글로벌 조건 컨텍스트 키를 참조하세요.

코어 네트워크 리소스에 태그 지정

태그는 사용자 또는 AWS가 AWS 리소스에 할당하는 메타데이터 레이블입니다. 각 태그는 키와 값으로 구성됩니다. 사용자가 할당하는 태그에 대해 키와 값을 정의합니다. 예를 들어 키를 `purpose`로 정의하고 리소스 하나의 값을 `test`로 정의할 수 있습니다. 태그는 다음을 지원합니다.

- AWS 리소스를 식별하고 정리합니다. 많은 AWS 서비스가 태그 지정을 지원하므로 다른 서비스의 리소스에 동일한 태그를 할당하여 해당 리소스의 관련 여부를 나타낼 수 있습니다.
- AWS 리소스에 대한 액세스를 제어합니다. 자세한 내용은 AWS Identify and Access Management 사용 설명서의 태그를 사용한 AWS 리소스 액세스 제어를 참조하세요.

Amazon CloudWatch Network Monitor와 IAM의 작동 방식

IAM을 사용하여 CloudWatch Network Monitor에 대한 액세스를 관리하기 전에 CloudWatch Network Monitor와 함께 사용할 수 있는 IAM 기능을 알아보세요.

Amazon CloudWatch Network Monitor에서 사용할 수 있는 IAM 기능

IAM 특성	CloudWatch 네트워크 모니터 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACLs	아니요
ABAC(정책 내 태그)	부분

IAM 특성	CloudWatch 네트워크 모니터 지원
임시 보안 인증	예
보안 주체 권한	예
서비스 역할	아니요
서비스 링크 역할	예

CloudWatch Network Monitor 및 기타 AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Amazon CloudWatch 네트워크 모니터에 대한 자격 증명 기반 정책

ID 기반 정책 지원	예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업와 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

CloudWatch Network Monitor에 대한 자격 증명 기반 정책 예시

CloudWatch Network Monitor 자격 증명 기반 정책 예시를 보려면 Amazon CloudWatch에 대한 자격 증명 기반 정책 예시 섹션을 참조하세요.

CloudWatch Network Monitor 내 리소스 기반 정책

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 또는 AWS 서비스가 포함될 수 있습니다.

계정 간 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 엔터티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 서로 다른 AWS 계정에 있는 경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에도 리소스 액세스 권한을 부여해야만 합니다. 개체에 자격 증명 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

CloudWatch Network Monitor에 대한 정책 작업

정책 작업 지원	예

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWSAPI 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

CloudWatch Network Monitor 작업 목록을 보려면 서비스 승인 참조의 Amazon CloudWatch Network Monitor에서 정의한 작업을 참조하세요.

CloudWatch Network Monitor의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
networkmonitor
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
```

```
      "networkmonitor:action1",
      "networkmonitor:action2"
         ]
```

CloudWatch Network Monitor 자격 증명 기반 정책 예시를 보려면 Amazon CloudWatch에 대한 자격 증명 기반 정책 예시 섹션을 참조하세요.

CloudWatch Network Monitor에 대한 정책 리소스

정책 리소스 지원	예

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 타입을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

CloudWatch Network Monitor 리소스 유형 및 해당 ARN 목록을 보려면 서비스 승인 참조의 Amazon CloudWatch Network Monitor에서 정의한 리소스 유형을 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon CloudWatch Network Monitor에서 정의한 작업을 참조하세요.

CloudWatch Network Monitor에 대한 정책 조건 키

서비스별 정책 조건 키 지원	예

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 Condition요소를 지정하거나 단일 Condition요소에서 여러 키를 지정하는 경우 AWS는 논리적 AND태스크를 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS는 논리적 OR태스크를 사용하여 조건을 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용자 가이드의 AWS 전역 조건 컨텍스트 키를 참조하세요.

CloudWatch Network Monitor 조건 키 목록을 보려면 서비스 승인 참조의 Amazon CloudWatch Network Monitor에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon CloudWatch Network Monitor에서 정의한 작업을 참조하세요.

CloudWatch Network Monitor의 ACL

ACL 지원	아니요

액세스 제어 목록(ACLs)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는 지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

CloudWatch 네트워크 모니터가 포함된 ABAC

ABAC(정책 내 태그) 지원	부분

속성 기반 액세스 제어(ABAC)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그를 기반으로 액세스를 제어하려면 aws:ResourceTag/*key-name*, aws:RequestTag/*key-name* 또는 aws:TagKeys 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 타입에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 타입에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 정보는 IAM 사용 설명서의 ABAC란 무엇인가요?를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

CloudWatch Network Monitor에서 임시 보안 인증 정보 사용

임시 보안 인증 지원	예

일부 AWS 서비스는 임시 보안 인증을 사용하여 로그인할 때 작동하지 않습니다. 임시 보안 인증으로 작동하는 AWS 서비스를 비롯한 추가 정보는 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 참조하세요.

사용자 이름과 암호를 제외한 다른 방법을 사용하여 AWS Management Console에 로그인하면 임시 보안 인증을 사용하는 것입니다. 예를 들어 회사의 Single Sign-On(SSO) 링크를 사용하여 AWS에 액세스하면 해당 프로세스에서 자동으로 임시 보안 인증을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 정보는 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 보안 인증을 수동으로 만들 수 있습니다 그런 다음 이러한 임시 보안 인증을 사용하여 AWS에 액세스할 수 있습니다. AWS에서는 장기 액세스 키를 사용하는 대신 임시 보안 인증을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 인증 섹션을 참조하세요.

CloudWatch Network Monitor의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

CloudWatch 네트워크 모니터에 대한 서비스 역할

서비스 역할 지원	아니요

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임하는 IAM role(IAM 역할)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 CloudWatch Network Monitor 기능이 중단될 수 있습니다. CloudWatch Network Monitor에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집하세요.

CloudWatch Network Monitor에 서비스 연결 역할 사용

서비스 링크 역할 지원	예

서비스 링크 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는 AWS서비스 단원을 참조하세요. 서비스 연결 역할 열에서 `Yes`가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

CloudWatch Network Monitor에 대한 자격 증명 기반 정책 예시

기본적으로 사용자 및 역할에는 CloudWatch Network Monitor 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 AWSAPI를 사용해 태스크를 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 포함하여 CloudWatch Network Monitor에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon CloudWatch Network Monitor에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- CloudWatch Network Monitor 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- CloudWatch Network Monitor 자격 증명 및 액세스 문제 해결

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 CloudWatch Network Monitor 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS managed policies(관리형 정책) 또는 AWS managed policies for job functions(직무에 대한 관리형 정책)를 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 Policies and permissions in IAM(IAM의 정책 및 권한)을 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 CloudFormation와 같이, 특정 AWS 서비스를 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하tpdy.

- 다중 인증(MFA) 필요 – AWS 계정 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 설정합니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 Configuring MFA-protected API access(MFA 보호 API 액세스 구성)를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

CloudWatch Network Monitor 콘솔 사용

Amazon CloudWatch Network Monitor 콘솔에 액세스하려면 최소한의 권한 세트가 있어야 합니다. 이러한 권한은 AWS 계정에서 CloudWatch Network Monitor 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다. 최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 엔터티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 CloudWatch Network Monitor 콘솔을 여전히 사용할 수 있도록 하려면 CloudWatch Network Monitor *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책을 엔터티에 추가합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예시는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI나 AWSAPI를 사용하여 프로그래밍 방식으로 이 태스크를 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": "*"
        }
    ]
}
```

CloudWatch Network Monitor 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 CloudWatch Network Monitor 및 IAM 작업 시 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- CloudWatch Network Monitor에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 AWS 계정 외부의 사람이 내 CloudWatch Network Monitor 리소스에 액세스하도록 허용하고 싶음

CloudWatch Network Monitor에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojacksonIAM 사용자가 콘솔을 사용하여 가상 *my-example-widget*리소스에 대한 세부 정보를 보려고 하지만 가상 networkmonitor:*GetWidget*권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 networkmonitor:GetWidget on resource: my-example-widget
```

이 경우 networkmonitor:*GetWidget*작업을 사용하여 *my-example-widget*리소스에 액세스할 수 있도록 mateojackson사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 CloudWatch Network Monitor에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예시 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 CloudWatch Network Monitor에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사람이 내 CloudWatch Network Monitor 리소스에 액세스하도록 허용하고 싶음

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- CloudWatch Network Monitor에서 이러한 기능의 지원 여부를 알아보려면 Amazon CloudWatch와 함께 IAM을 사용하는 방법 섹션을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하세요.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 연동)을 참조하세요.

- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

CloudWatch Network Monitor에 대한 AWS 관리형 정책

사용자, 그룹 또는 역할에 권한을 추가할 때 정책을 직접 작성하는 것보다 AWS관리형 정책을 사용하는 것이 더욱 편리합니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하려면 시간과 전문 지식이 필요합니다. 빨리 시작하려면 AWS관리형 정책을 사용할 수 있습니다. 이러한 정책은 일반적인 사용 사례에 적용되며 AWS계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 정보는 IAM 사용 설명서에서 AWS관리형 정책을 참조하세요.

AWS 서비스 유지 관리 및 AWS관리형 정책 업데이트입니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스에서 때때로 추가 권한을 AWS 관리형 정책에 추가하여 새로운 기능을 지원합니다. 이 타입의 업데이트는 정책이 연결된 모든 보안 인증(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새로운 기능이 시작되거나 새 태스크를 사용할 수 있을 때 AWS 관리형 정책에 업데이트됩니다. 서비스는 AWS관리형 정책에서 권한을 제거하지 않기 때문에 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한 AWS(은)는 여러 서비스의 직무에 대한 관리형 정책을 지원합니다. 예를 들어 `ReadOnlyAccess`AWS 관리형 정책은 모든 AWS및 리소스에 대한 읽기 전용 액세스 권한을 제공합니다. 서비스에서 새 기능을 시작하면 AWS가 새 작업 및 리소스에 대한 읽기 전용 권한을 추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한 AWS관리형 정책을 참조하세요.

AWS 관리형 정책: CloudWatchNetworkMonitorServiceRolePolicy

`CloudWatchNetworkMonitorServiceRolePolicy`는 서비스가 사용자를 대신하여 작업을 수행하고 CloudWatch Network Monitor와 연결된 리소스에 액세스할 수 있도록 허용하는 서비스 연결 역할에 연결됩니다. 이 정책을 IAM 자격 증명에 연결할 수 없습니다. 자세한 내용은 the section called “서비스 연결 역할” 단원을 참조하십시오.

AWS 관리형 정책에 대한 CloudWatch Network Monitor 업데이트

아래에는 CloudWatch Network Monitor의 AWS 관리형 정책 업데이트에 관한 세부 정보가 나와 있습니다. 이 서비스가 2023년 11월에 해당 변경 사항을 추적하기 시작한 이후부터 설명되어 있습니다.

변경 사항	설명	날짜
CloudWatchNetworkMonitorServiceRolePolicy: 새 정책	CloudWatch 네트워크 모니터에 추가된 새 정책입니다.	2023년 11월 27일
the section called “AWSServiceRoleForNetworkMonitor”. 새 역할	CloudWatch 네트워크 모니터에 추가된 새 역할입니다.	2023년 11월 27일

CloudWatch Network Monitor에 대한 IAM 권한

Amazon CloudWatch Network Monitor를 사용하려면 사용자에게 올바른 권한이 있어야 합니다.

Amazon CloudWatch의 보안에 대한 자세한 내용은 Amazon CloudWatch의 Identity and Access Management 섹션을 참조하세요.

모니터를 보는 데 필요한 권한

AWS Management Console에서 Amazon CloudWatch Network Monitor의 모니터를 보려면 다음 권한이 있는 사용자 또는 역할로 로그인해야 합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "networkmonitor:Get*",
                "networkmonitor:List*"
                ],
            "Resource": "*"
        }
    ]
}
```

모니터를 생성하는 데 필요한 권한

Amazon CloudWatch Network Monitor에서 모니터를 만들려면 사용자에게 Network Monitor와 연결된 서비스 연결 역할을 생성할 권한이 있어야 합니다. 서비스 연결 역할에 대한 자세한 내용은 CloudWatch Network Monitor에 서비스 연결 역할 사용 섹션을 참조하세요.

AWS Management Console에서 Amazon CloudWatch Network Monitor의 모니터를 생성하려면 사용자가 다음 정책에 포함된 권한이 있는 역할 또는 사용자로 로그인해야 합니다.

> **Note**
>
> 더욱 엄격하게 제한되는 자격 증명 기반 권한 정책을 만들면 해당 정책이 적용되는 사용자는 모니터를 생성할 수 없습니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "networkmonitor:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
networkmonitor.amazonaws.com/AWSServiceRoleForNetworkMonitor",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "networkmonitor.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:GetRole",
                "iam:PutRolePolicy"
```

```
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
networkmonitor.amazonaws.com/AWSServiceRoleForNetworkMonitor"
        },
        {
            "Action": [
                "ec2:CreateSecurityGroup",
                "ec2:CreateNetworkInterface",
                "ec2:CreateTags"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

CloudWatch Network Monitor에 서비스 연결 역할 사용

Amazon CloudWatch Network Monitor는 다른 AWS 서비스를 직접적으로 자동 호출하는 데 필요한 권한에 다음 서비스 연결 역할을 사용합니다.

- AWSServiceRoleForNetworkMonitor

AWSServiceRoleForNetworkMonitor

CloudWatch Network Monitoring은 AWSServiceRoleForNetworkMonitor라는 서비스 연결 역할을 사용하여 CloudWatch 네트워크 모니터를 업데이트하고 관리합니다.

AWSServiceRoleForNetworkMonitor 서비스 연결 역할은 그 역할을 위임하기 위해 다음 서비스를 신뢰합니다.

- networkmonitor.amazonaws.com

CloudWatchNetworkMonitorServiceRolePolicy는 서비스 연결 역할에 연결되어 서비스가 계정의 VPC 및 EC2 리소스에 액세스하고 생성된 네트워크 모니터를 관리할 수 있는 액세스 권한을 부여합니다.

권한 그룹화

정책은 다음 권한 집합으로 그룹화됩니다.

- **cloudwatch** - 서비스 보안 주체가 CloudWatch 리소스에 네트워크 모니터링 지표를 게시할 수 있도록 합니다.
- **ec2** - 서비스 보안 주체가 사용자 계정의 VPC와 서브넷을 설명하여 모니터와 프로브를 만들거나 업데이트할 수 있도록 합니다. 또한 서비스 보안 주체가 보안 그룹, 네트워크 인터페이스 및 관련 권한을 생성, 수정 및 삭제하여 모니터링 트래픽을 엔드포인트로 전송하도록 모니터 또는 프로브를 구성할 수 있습니다.

정책에 대한 자세한 내용은 the section called “AWS 관리형 정책” 섹션을 참조하세요.

다음은 CloudWatchNetworkMonitorServiceRolePolicy를 보여줍니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "PublishCw",
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": "AWS/NetworkMonitor"
    }
   }
  },
  {
   "Sid": "DescribeAny",
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribeNetworkInterfaceAttribute",
    "ec2:DescribeVpcs",
    "ec2:DescribeNetworkInterfacePermissions",
    "ec2:DescribeSubnets",
    "ec2:DescribeSecurityGroups"
   ],
   "Resource": "*"
  },
  {
   "Sid": "DeleteModifyEc2Resources",
   "Effect": "Allow",
   "Action": [
```

```
    "ec2:AuthorizeSecurityGroupEgress",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterfacePermission",
    "ec2:RevokeSecurityGroupEgress",
    "ec2:ModifyNetworkInterfaceAttribute",
    "ec2:DeleteNetworkInterface",
    "ec2:DeleteSecurityGroup"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*:*:security-group/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/ManagedByCloudWatchNetworkMonitor": "true"
    }
   }
  }
 ]
}
```

서비스 연결 역할 생성

`AWSServiceRoleForNetworkMonitor`

`AWSServiceRoleForNetworkMonitor` 역할을 수동으로 생성할 필요가 없습니다.

- CloudWatch 네트워크 모니터는 첫 번째 네트워크 모니터를 생성할 때 `AWSServiceRoleForNetworkMonitor` 역할을 생성합니다. 이 역할은 이후에 생성하는 모든 모니터에 적용됩니다.

서비스 연결 역할을 자동으로 생성하려면 사용자에게 필수 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

서비스 연결 역할 편집

IAM을 사용하여 `AWSServiceRoleForNetworkMonitor` 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

서비스 연결 역할 삭제

이제 CloudWatch Network Monitor를 사용할 필요가 없는 경우 `AWSServiceRoleForNetworkMonitor` 역할을 삭제하는 것이 좋습니다.

네트워크 모니터를 삭제한 후에만 이 서비스 연결 역할을 삭제할 수 있습니다. 네트워크 모니터 삭제에 대한 자세한 내용은 네트워크 모니터 삭제를 참조하세요.

IAM 콘솔, IAM CLI 또는 IAM API를 사용하여 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

`AWSServiceRoleForNetworkMonitor` CloudWatch Network Monitor를 삭제하면 새 모니터를 생성할 때 역할이 다시 생성됩니다.

CloudWatch Network Monitor 서비스 연결 역할을 지원하는 리전

CloudWatch Network Monitor에서는 서비스를 사용할 수 있는 모든 AWS 리전 리전에서 서비스 연결 역할을 지원합니다. 자세한 내용은 AWS 일반 참조의 AWS 엔드포인트를 참조하세요.

서비스 연결 역할 삭제

이제 CloudWatch Network Monitor를 사용할 필요가 없는 경우 `AWSServiceRoleForNetworkMonitor` 역할을 삭제하는 것이 좋습니다.

네트워크 모니터를 삭제한 후에만 이 서비스 연결 역할을 삭제할 수 있습니다. 네트워크 모니터 삭제에 대한 자세한 내용은 네트워크 모니터 삭제를 참조하세요.

IAM 콘솔, IAM CLI 또는 IAM API를 사용하여 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

`AWSServiceRoleForNetworkMonitor` CloudWatch Network Monitor를 삭제하면 새 모니터를 생성할 때 역할이 다시 생성됩니다.

요금

Amazon CloudWatch Network Monitor를 사용하면 선불 비용이나 장기 약정이 없습니다. 네트워크 모니터 요금에는 다음 두 가지 구성 요소가 있습니다.

- 모니터링되는 리소스당 시간당 요금
- CloudWatch 지표 요금

네트워크 모니터를 생성할 때 모니터링할 리소스를 네트워크 모니터와 연결합니다. 네트워크 모니터의 경우 Amazon Virtual Private Cloud(VPC)의 서브넷입니다. 모니터링되는 각 리소스를 사용하면 VPC의 각 서브넷에서 4개의 대상에 대해 최대 4개의 프로브를 생성할 수 있습니다. 요금 관리에 도움이 되도록 모니터링되는 리소스 수를 줄여 서브넷 범위와 온프레미스 IP 범위를 조정할 수 있습니다.

요금에 대한 자세한 정보는 Amazon CloudWatch 요금 페이지를 참조하세요.

인프라 모니터링

이 섹션의 주제에서는 AWS 리소스에 대한 운영 가시성을 확보하는 데 도움이 되는 CloudWatch 기능에 대해 설명합니다.

주제

Container Insights

CloudWatch Container Insights를 사용해 컨테이너화된 애플리케이션 및 마이크로서비스의 지표 및 로그를 수집하고 집계하며 요약할 수 있습니다. Container Insights는 Amazon Elastic Container Service(Amazon ECS), Amazon Elastic Kubernetes Service(Amazon EKS), Amazon EC2의 Kubernetes 플랫폼에서 사용할 수 있습니다. Container Insights는 Amazon ECS와 Amazon EKS 모두에 대해 AWS Fargate에 배포된 클러스터에서 지표를 수집하는 것을 지원합니다.

CloudWatch는 CPU, 메모리, 디스크, 네트워크와 같은 많은 리소스에 대한 지표를 자동으로 수집합니다. 또한 Container Insights는 컨테이너 재시작 오류 같은 진단 정보를 제공하여 문제를 격리하고 신속하게 해결할 수 있도록 도와줍니다. Container Insights가 수집하는 지표에 대해 CloudWatch 경보를 설정할 수도 있습니다.

Container Insights는 임베디드 지표 형식을 사용하여 데이터를 '성능 로그 이벤트'로 수집합니다. 이러한 성능 로그 이벤트는 카디널리티가 높은 데이터를 대규모로 수집 및 저장할 수 있게 하는 구조화된 JSON 스키마를 사용하는 항목입니다. 이 데이터를 기반으로 CloudWatch는 클러스터, 노드, 포드, 태스크 및 서비스 수준에서 집계된 지표를 CloudWatch 지표로 생성합니다. Container Insights가 수집하는 지표는 CloudWatch 자동 대시보드에서 사용할 수 있으며 CloudWatch 콘솔의 지표 섹션에서도 볼 수 있습니다. 컨테이너 작업이 일정 시간 동안 실행될 때까지는 지표가 표시되지 않습니다.

컨테이너 인사이트를 배포하면 성능 로그 이벤트에 대한 로그 그룹이 자동으로 생성됩니다. 이 로그 그룹을 직접 생성할 필요는 없습니다.

Container Insights 비용 관리에 도움이 되도록 CloudWatch는 로그 데이터에서 가능한 모든 지표를 자동으로 생성하지 않습니다. 그러나 CloudWatch Logs Insights를 사용하여 원시 성능 로그 이벤트를 분석하면 추가 지표 및 추가 세부 수준을 확인할 수 있습니다.

Container Insights의 원래 버전에서는 수집된 지표와 로그가 사용자 지정 지표로 청구됩니다. Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 사용하면 Container Insights 지표 및 로그는 저장된 지표나 수집된 로그별로 요금이 부과되는 대신 관찰당 요금이 부과됩니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

Amazon EKS 및 Kubernetes에서 Container Insights는 컨테이너화된 버전의 CloudWatch 에이전트를 사용하여 클러스터에서 실행 중인 모든 컨테이너를 검색합니다. 이어서 성능 스택의 모든 계층에서 성능 데이터를 수집합니다.

Container Insights는 자체적으로 수집하는 로그 및 지표에 대해 AWS KMS key를 이용한 암호화를 지원합니다. 이 암호화를 활성화하려면 Container Insights 데이터를 수신하는 로그 그룹에 대한 AWS KMS 암호화를 수동으로 활성화해야 합니다. 이렇게 하면 Container Insights가 제공된 KMS 키를 사용하여 이 데이터를 암호화합니다. 대칭 키만 지원됩니다. 비대칭 KMS 키를 사용하여 로그 그룹을 암호화하지 마세요.

자세한 내용은 AWS KMS를 사용하여 CloudWatch Logs의 로그 데이터 암호화 단원을 참조하세요.

Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights

2023년 11월 6일, Container Insights의 새 버전이 출시되었습니다. 이 버전은 Amazon EC2에서 실행되는 Amazon EKS 클러스터의 향상된 관찰 기능을 지원하며 이러한 클러스터에서 더 자세한 지표를 수집할 수 있습니다. 설치 후에는 Amazon EKS 클러스터에 대한 상세한 인프라 텔레메트리 및 컨테이너 로그를 자동으로 수집합니다. 그런 다음 즉시 사용할 수 있는 엄선된 대시보드를 사용하여 애플리케이션 및 인프라 텔레메트리를 자세히 살펴볼 수 있습니다.

Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights는 컨테이너 수준까지 세분화된 상태, 성능 및 상태 지표와 컨트롤 플레인 지표를 수집합니다. 추가 지표 및 차원의 수집에 대한 자세한 내용은 Amazon EKS 및 Kubernetes Container Insights 지표 섹션을 참조하세요.

2023년 11월 6일 이후에 Amazon EC2의 Amazon EKS 클러스터에 CloudWatch 에이전트를 사용하여 Container Insights를 설치한 경우 Amazon EKS에 대한 관찰성이 향상된 Container Insights를 갖게 됩니다. 그렇지 않으면 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights로 업그레이드의 지침에 따라 Amazon EKS 클러스터를 새 버전으로 업그레이드할 수 있습니다.

Container Insights는 CloudWatch 크로스 계정 관찰성을 지원합니다. 단일 모니터링 계정을 사용하면 단일 리전 내의 여러 AWS 계정에 걸쳐 있는 애플리케이션을 모니터링하고 문제를 해결할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights는 Windows 워커 노드도 지원합니다.

Amazon EKS의 향상된 관찰 기능 포함된 Container Insights는 Fargate에서 지원되지 않습니다.

> **Note**
>
> Container Insights 콘솔로 이동하여 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights로 업그레이드할 수 있는 클러스터가 있는지 확인할 수 있습니다. 이렇게 하려면 CloudWatch 콘솔의 탐색 창에서 Insights, Container Insights를 선택합니다. Container Insights 콘솔에는 업그레이드할 수 있는 Amazon EKS 클러스터가 있는지 여부를 알려주는 배너와 업그레이드 페이지 링크가 표시됩니다.

지원하는 플랫폼

Container Insights는 Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, Amazon EC2 인스턴스의 Kubernetes 플랫폼에서 사용할 수 있습니다.

- Amazon ECS의 경우 Container Insights는 Linux 인스턴스와 Windows Server 인스턴스 모두에서 클러스터, 태스크 및 서비스 수준의 지표를 수집합니다. Linux 인스턴스에서만 인스턴스 레벨에서 지표를 수집할 수 있습니다.

  Amazon ECS의 경우 네트워크 지표는 `bridge` 네트워크 모드와 `awsvpc` 네트워크 모드의 컨테이너에만 제공됩니다. `host` 네트워크 모드의 컨테이너에는 제공되지 않습니다.
- Amazon Elastic Kubernetes Service 및 Amazon EC2 인스턴스의 Kubernetes 플랫폼의 경우 Container Insights는 Linux 인스턴스에서만 지원됩니다.

CloudWatch 에이전트 컨테이너 이미지

Amazon은 Amazon Elastic Container Registry의 CloudWatch 에이전트 컨테이너 이미지를 제공합니다. 자세한 내용은 Amazon ECR의 cloudwatch-agent를 참조하세요.

지원되는 리전

Amazon ECS의 Container Insights는 다음 리전에서 지원됩니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(자카르타)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(도쿄)
- 아시아 태평양(시드니)
- 캐나다 서부(캘거리)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스페인)
- 유럽(스톡홀름)
- 유럽(취리히)
- 중동(바레인)
- 중동(UAE)
- 남아메리카(상파울루)
- AWS GovCloud(미국 동부)

- AWS GovCloud(미국 서부)
- 중국(베이징)
- 중국(닝샤)

Amazon EKS 및 Kubernetes 지원 리전

Amazon EKS 및 Kubernetes의 Container Insights는 다음 리전에서 지원됩니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 아시아 태평양(홍콩)
- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 유럽(스톡홀름)
- 중동(바레인)
- 남아메리카(상파울루)
- AWS GovCloud(미국 동부)
- AWS GovCloud(미국 서부)

Container Insights 설정

Amazon ECS, Amazon EKS, Kubernetes에 대한 Container Insights 설정 프로세스는 서로 다릅니다.

주제

Amazon ECS에서 Container Insights 설정

다음 옵션 중 하나 또는 둘 다를 사용하여 Amazon ECS 클러스터에서 Container Insights를 사용 설정할 수 있습니다.

- AWS Management Console 또는 AWS CLI를 사용하여 클러스터 레벨, 작업 레벨 및 서비스 레벨 지표 수집을 시작합니다.
- CloudWatch 에이전트를 대몬(daemon) 서비스로 배포하여 Amazon EC2 인스턴스에서 호스트되는 클러스터에서 인스턴스 수준 지표 수집을 시작하세요.

주제

클러스터 및 서비스 수준 지표를 위해 Amazon ECS에서 Container Insights 설정

신규 및 기존 Amazon ECS 클러스터에서 Container Insights를 사용 설정할 수 있습니다. Container Insights는 클러스터, 작업 및 서비스 레벨에서 지표를 수집합니다. Amazon ECS 콘솔 또는 AWS CLI를 사용하여 Container Insights를 활성화할 수 있습니다.

Amazon EC2 인스턴스에서 Amazon ECS를 사용 중인데 Container Insights에서 네트워크 및 스토리지 지표를 수집하려는 경우 Amazon ECS 에이전트 버전 1.29를 포함한 AMI를 사용하여 해당 인스턴스를 시작합니다. 에이전트 버전 업데이트에 대한 자세한 내용은 Amazon ECS 컨테이너 에이전트 업데이트 단원을 참조하세요.

AWS CLI를 사용하여 계정에 생성된 새로운 Amazon ECS 클러스터에 대해 Container Insights를 사용 설정할 수 있는 계정 수준 권한을 설정할 수 있습니다. 이를 위해 다음 명령을 입력합니다.

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

> **Note**
>
> Amazon ECS Container Insights 지표에 사용하는 고객 관리 AWS KMS 키가 CloudWatch와 함께 작동하도록 아직 구성되지 않은 경우, CloudWatch Logs에서 암호화된 로그를 허용하도록 키 정책을 업데이트해야 합니다. 또한 사용자 고유의 AWS KMS 키를 `/aws/ecs/containerinsights/ClusterName/performance` 아래의 로그 그룹과 연결해야 합니다. 자세한 내용은 AWS Key Management Service를 사용하여 CloudWatch Logs의 로그 데이터 암호화 섹션을 참조하세요.

기존 Amazon ECS 클러스터에서 Container Insights 설정

기존 Amazon ECS 클러스터에서 Container Insights를 사용 설정하려면 다음 명령을 입력합니다. 다음 명령이 작동하려면 AWS CLI 버전 1.16.200 이상을 실행 중이어야 합니다.

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=enabled
```

신규 Amazon ECS 클러스터에서 Container Insights 설정

새로운 Amazon ECS 클러스터에서 Container Insights를 사용 설정할 수 있는 방법이 두 가지 있습니다. 기본적으로 새로운 모든 클러스터가 Container Insights에 대해 사용 설정되도록 Amazon ECS를 구성할 수 있습니다. 그렇지 않으면 새 클러스터를 생성할 때 활성화할 수 있습니다.

AWS Management Console 사용

모든 새 클러스터에서 기본적으로 또는 생성할 때 개별 클러스터에서 컨테이너 인사이트를 설정할 수 있습니다.

모든 새 클러스터에서 컨테이너 인사이트를 기본적으로 사용하도록 설정하려면

1. https://console.aws.amazon.com/ecs/v2에서 콘솔을 엽니다.
2. 탐색 페이지에서 Account Settings(계정 설정)를 선택합니다.

3. 업데이트를 선택합니다.
4. 클러스터에서 기본적으로 CloudWatch Container Insights를 사용하려면 CloudWatch Container Insights에서 CloudWatch Container Insights를 선택하거나 선택 취소합니다.
5. Save changes(변경 사항 저장)를 선택합니다.

모든 새 클러스터에 기본적으로 Container Insights를 활성화하기 위해 앞에 나온 절차를 사용하지 않은 경우 Container Insights가 활성화된 상태에서 다음 단계에 따라 클러스터를 생성하세요.

Container Insights가 켜진 클러스터를 생성하려면

1. https://console.aws.amazon.com/ecs/v2에서 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 클러스터(Clusters) 페이지에서 클러스터 생성(Create cluster)을 선택합니다.
4. 클러스터 구성(Cluster configuration) 아래의 클러스터 이름(Cluster name)에 고유한 이름을 입력합니다.

   이름은 최대 255자(대/소문자), 숫자 및 하이픈을 포함할 수 있습니다.
5. Container Insights를 켜려면 모니터링을 확장한 다음 Container Insights 사용을 켭니다.

이제 클러스터에서 작업 정의 생성, 작업 실행, 서비스 시작을 할 수 있습니다. 자세한 내용은 다음을 참조하십시오.

- 태스크 정의 생성하기
- 태스크 실행
- 서비스 생성

AWS CLI를 사용하여 신규 Amazon ECS 클러스터에서 Container Insights 설정

모든 새 클러스터에서 Container Insights를 기본적으로 활성화하려면 다음 명령을 입력합니다.

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

모든 새로운 클러스터에 기본적으로 Container Insights를 활성화하기 위해 앞에 나온 명령을 사용하지 않은 경우, Container Insights 활성화되어 있을 때 새로운 클러스터를 생성하기 위해 다음 명령을 입력합니다. 다음 명령이 작동하려면 AWS CLI 버전 1.16.200 이상을 실행 중이어야 합니다.

```
aws ecs create-cluster --cluster-name myCICluster --settings
 "name=containerInsights,value=enabled"
```

Amazon ECS 클러스터에서 Container Insights 사용 중지

기존 Amazon ECS 클러스터에서 Container Insights를 사용 중지하려면 다음 명령을 입력합니다.

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=disabled
```

AWS Distro for OpenTelemetry를 사용하여 Amazon ECS에서 Container Insights 설정

AWS Distro for OpenTelemetry를 사용하여 Amazon ECS 클러스터에서 CloudWatch Container Insights를 설정하려는 경우 이 단원을 참조하세요. AWS Distro for Open Telemetry에 대한 자세한 내용은 AWS Distro for OpenTelemetry를 참조하세요.

다음 단계에서는 Amazon ECS를 실행 중인 클러스터가 이미 있다고 가정합니다. Amazon ECS와 함께 AWS Distro for Open Telemetry를 사용하고 이러한 용도로 Amazon ECS 클러스터를 설정하는 방법에 대한 자세한 내용은 Amazon Elastic Container Service에서 AWS Distro for OpenTelemetry Collector 설정을 참조하세요.

1단계: 태스크 역할 생성

첫 번째 단계는 AWS OpenTelemetry Collector가 사용할 클러스터에서 태스크 역할을 생성하는 것입니다.

AWS Distro for OpenTelemetry에 대한 태스크 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택한 후 정책 생성을 선택합니다.
3. [JSON] 탭을 선택하고 다음 정책을 복사합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
```

```
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        }
    ]
}
```

4. 정책 검토를 선택합니다.
5. 이름에 **AWSDistroOpenTelemetryPolicy**를 입력한 다음, [정책 생성(Create policy)]을 선택합니다.
6. 왼쪽 탐색 창에서 [역할(Roles)]을 선택한 다음, [역할 생성(Create role)]을 선택합니다.
7. 서비스 목록에서 [Elastic Container Service]를 선택합니다.
8. 페이지 하단에서 [Elastic Container Service 태스크(Elastic Container Service Task)]를 선택한 후 [다음: 권한(Next: Permissions)]을 선택합니다.
9. 정책 목록에서 AWSDistroOpenTelemetryPolicy를 검색합니다.
10. [AWSDistroOpenTelemetryPolicy] 옆의 확인란을 선택합니다.
11. [다음: 태그(Next: Tags)]를 선택한 후 [다음: 검토(Next: Review)]를 선택합니다.
12. [역할 이름(Role name)]에 **AWSOpenTelemetryTaskRole**을 입력한 다음, [역할 생성(Create role)]을 선택합니다.

2단계: 태스크 실행 역할 생성

다음 단계는 AWS OpenTelemetry Collector에 대한 태스크 실행 역할을 생성하는 것입니다.

AWS Distro for OpenTelemetry에 대한 태스크 실행 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [역할(Roles)]을 선택한 다음, [역할 생성(Create role)]을 선택합니다.
3. 서비스 목록에서 [Elastic Container Service]를 선택합니다.
4. 페이지 하단에서 [Elastic Container Service 태스크(Elastic Container Service Task)]를 선택한 후 [다음: 권한(Next: Permissions)]을 선택합니다.
5. 정책 목록에서 AmazonECSTaskExecutionRolePolicy를 검색한 다음, [AmazonECSTaskExecutionRolePolicy] 옆의 확인란을 선택합니다.

6. 정책 목록에서 CloudWatchLogsFullAccess를 검색한 다음, [CloudWatchLogsFullAccess] 옆의 확인란을 선택합니다.
7. 정책 목록에서 AmazonSSMReadOnlyAccess를 검색한 다음, [AmazonSSMReadOnlyAccess] 옆의 확인란을 선택합니다.
8. [다음: 태그(Next: Tags)]를 선택한 후 [다음: 검토(Next: Review)]를 선택합니다.
9. [역할 이름(Role name)]에 **`AWSOpenTelemetryTaskExecutionRole`**을 입력한 다음, [역할 생성(Create role)]을 선택합니다.

3단계: 태스크 정의 생성

다음 단계는 태스크 정의를 생성하는 것입니다.

AWS Distro for OpenTelemetry에 대한 태스크 정의를 생성하려면

1. https://console.aws.amazon.com/ecs/v2에서 콘솔을 엽니다.
2. 탐색 창에서 태스크 정의(Task definitions)를 선택합니다.
3. 새 태스크 정의 생성(Create new Task Definition), 새 태스크 정의 생성(Create new Task Definition)을 선택합니다.
4. 태스크 정의 패밀리(Task definition family)에서 태스크 정의에 대해 고유한 이름을 지정합니다.
5. 컨테이너를 구성한 후 다음을 선택합니다.
6. 지표 및 로깅에서 지표 수집 사용을 선택합니다.
7. 다음을 선택합니다.
8. Create를 선택합니다.

Amazon ECS와 함께 AWS OpenTelemetry Collector를 사용하는 방법에 대한 자세한 내용은 Amazon Elastic Container Service에서 AWS Distro for OpenTelemetry Collector 설정을 참조하세요.

4단계: 태스크 실행

마지막 단계는 생성한 태스크를 실행하는 것입니다.

AWS Distro for OpenTelemetry에 대한 태스크를 실행하려면

1. https://console.aws.amazon.com/ecs/v2에서 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [태스크 정의(Task Definitions)]를 선택한 다음, 방금 생성한 태스크를 선택합니다.

3. 작업, 배포, 태스크 실행을 선택합니다.
4. 배포(Deploy), 태스크 실행(Run task)을 선택합니다.
5. 컴퓨팅 옵션 섹션의 기존 클러스터에서 클러스터를 선택합니다.
6. 생성(Create)을 선택합니다.
7. 그런 다음, CloudWatch 콘솔에서 새 지표를 확인할 수 있습니다.
8. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
9. 왼쪽 탐색 창에서 Metrics를 선택합니다.

   ECS/ContainerInsights 네임스페이스가 표시됩니다. 해당 네임스페이스를 선택하면 8개의 지표가 표시됩니다.

CloudWatch 에이전트를 배포하여 Amazon ECS의 EC2 인스턴스 수준 지표 수집

CloudWatch 에이전트를 배포하여 EC2 인스턴스에서 호스트되는 Amazon ECS 클러스터에서 인스턴스 수준 지표를 수집하려면 기본 구성으로 빠른 시작 설정을 사용하거나 에이전트를 수동으로 설치해 사용자 지정하면 됩니다.

두 방법 모두 EC2 시작 유형으로 배포된 Amazon ECS 클러스터가 하나 이상 있고 CloudWatch 에이전트 컨테이너가 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 수 있어야 합니다. IMDS에 대한 자세한 내용은 인스턴스 메타데이터 및 사용자 데이터를 참조하세요.

이러한 방법은 AWS CLI도 설치되어 있다고 가정합니다. 또한 다음 절차에서 명령을 실행하려면 IAMFullAccess 및 AmazonECS_FullAccess 정책이 있는 계정 또는 역할에 로그인해야 합니다.

주제

- AWS CloudFormation을 사용한 빠른 설정
- 수동 설치 및 사용자 지정

AWS CloudFormation을 사용한 빠른 설정

빠른 설정을 사용하려면 다음 명령을 입력하여 AWS CloudFormation을 사용해 에이전트를 설치합니다. *`cluster-name`* 및 *`cluster-region`*을 Amazon ECS 클러스터의 이름 및 리전으로 바꿉니다.

이 명령은 IAM 역할인 CWAgentECSTaskRole 및 CWAgentECSExecutionRole을 생성합니다. 이러한 역할이 계정에 이미 있는 경우 명령을 입력할 때 `ParameterKey=CreateIAMRoles,ParameterValue=True` 대신

ParameterKey=CreateIAMRoles,ParameterValue=False를 사용합니다. 그렇지 않으면 명령이 실패합니다.

```
ClusterName=cluster-name
Region=cluster-region
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
    --template-body file://cwagent-ecs-instance-metric-cfn.json \
    --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
                 ParameterKey=CreateIAMRoles,ParameterValue=True \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${Region}
```

(대안) 자체 IAM 역할 사용

CWAgentECSTaskRole 및 CWAgentECSExecutionRole 역할 대신 고유한 사용자 지정 ECS 작업 역할 및 ECS 작업 실행 역할을 사용하려면 먼저 ECS 작업 역할로 사용할 역할에 CloudWatchAgentServerPolicy가 연결되어 있는지 확인합니다. 또한 ECS 작업 실행 역할로 사용할 역할에 CloudWatchAgentServerPolicy 및 AmazonECSTaskExecutionRolePolicy 정책이 모두 연결되어 있는지 확인합니다. 이어서 다음 명령을 입력합니다. 명령에서 *task-role-arn*을 사용자 지정 ECS 작업 역할의 ARN으로 바꾸고 *execution-role-arn*을 사용자 지정 ECS 작업 실행 역할의 ARN으로 바꿉니다.

```
ClusterName=cluster-name
Region=cluster-region
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
    --template-body file://cwagent-ecs-instance-metric-cfn.json \
    --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
                 ParameterKey=TaskRoleArn,ParameterValue=${TaskRoleArn} \
                 ParameterKey=ExecutionRoleArn,ParameterValue=${ExecutionRoleArn} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${Region}
```

빠른 설정 문제 해결

AWS CloudFormation 스택의 상태를 확인하려면 다음 명령을 입력합니다.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stacks --stack-name CWAgentECS-$ClusterName-$Region --
region $Region
```

CREATE_COMPLETE 또는 CREATE_IN_PROGRESS 이외의 StackStatus가 표시되면 스택 이벤트를 확인하여 오류를 찾습니다. 다음 명령을 입력합니다.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack-events --stack-name CWAgentECS-$ClusterName-$Region
 --region $Region
```

cwagent 데몬 서비스의 상태를 확인하려면 다음 명령을 입력합니다. 출력에서 runningCount가 deployment 섹션의 desiredCount와 같은 것을 볼 수 있습니다. 같지 않은 경우 출력에서 failures 섹션을 확인합니다.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs describe-services --services cwagent-daemon-service --cluster $ClusterName --
region $Region
```

CloudWatch Logs 콘솔을 사용하여 에이전트 로그를 확인할 수도 있습니다. /ecs/ecs-cwagent-daemon-service 로그 그룹을 찾습니다.

CloudWatch 에이전트의 AWS CloudFormation 스택 삭제

AWS CloudFormation 스택을 삭제해야 하는 경우 다음 명령을 입력합니다.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation delete-stack --stack-name CWAgentECS-${ClusterName}-${Region} --
region ${Region}
```

수동 설치 및 사용자 지정

이 단원의 단계에 따라 CloudWatch 에이전트를 수동으로 배포하여 EC2 인스턴스에서 호스트되는 Amazon ECS 클러스터에서 인스턴스 수준 지표를 수집합니다.

필요한 IAM 역할 및 정책

두 가지 IAM 역할이 필요합니다. 아직 존재하지 않는 경우 생성해야 합니다. 이러한 역할에 대한 자세한 내용은 IAM roles for Tasks(태스크에 대한 IAM 역할)와 Amazon ECS Task Execution Role(Amazon ECS 태스크 실행 역할)을 참조하세요.

- CloudWatch 에이전트가 지표를 게시하는 데 사용하는 'ECS 태스크 역할'. 이 역할이 이미 있는 경우 `CloudWatchAgentServerPolicy` 정책이 연결되어 있는지 확인해야 합니다.
- Amazon ECS 에이전트가 CloudWatch 에이전트를 시작하는 데 사용하는 'ECS 태스크 실행 역할'. 이 역할이 이미 있는 경우 `AmazonECSTaskExecutionRolePolicy` 및 `CloudWatchAgentServerPolicy` 정책이 연결되어 있는지 확인해야 합니다.

이러한 역할이 아직 없는 경우 다음 명령을 사용하여 역할을 생성하고 필요한 정책을 연결할 수 있습니다. 이 첫 번째 명령은 ECS 작업 역할을 생성합니다.

```
aws iam create-role --role-name CWAgentECSTaskRole \
    --assume-role-policy-document "{\"Version\": \"2012-10-17\",\"Statement\": [{\"Sid
\": \"\",\"Effect\": \"Allow\",\"Principal\": {\"Service\": \"ecs-tasks.amazonaws.com
\"},\"Action\": \"sts:AssumeRole\"}]}"
```

위 명령을 입력한 후 명령 출력에서 `Arn` 값을 "TaskRoleArn"으로 기록해 둡니다. 나중에 태스크 정의를 생성할 때 사용해야 합니다. 이어서 다음 명령을 입력하여 필요한 정책을 연결합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
CloudWatchAgentServerPolicy \
    --role-name CWAgentECSTaskRole
```

다음 명령은 ECS 작업 실행 역할을 생성합니다.

```
aws iam create-role --role-name CWAgentECSExecutionRole \
    --assume-role-policy-document "{\"Version\": \"2012-10-17\",\"Statement\": [{\"Sid
\": \"\",\"Effect\": \"Allow\",\"Principal\": {\"Service\": \"ecs-tasks.amazonaws.com
\"},\"Action\": \"sts:AssumeRole\"}]}"
```

위 명령을 입력한 후 명령 출력에서 `Arn` 값을 "ExecutionRoleArn"으로 기록해 둡니다. 나중에 태스크 정의를 생성할 때 사용해야 합니다. 이어서 다음 명령을 입력하여 필요한 정책을 연결합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
CloudWatchAgentServerPolicy \
```

```
    --role-name CWAgentECSExecutionRole

aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy \
    --role-name CWAgentECSExecutionRole
```

태스크 정의 생성 및 데몬 서비스 시작

태스크 정의를 생성하고 이를 사용하여 CloudWatch 에이전트를 데몬 서비스로 시작합니다. 작업 정의를 생성하려면 다음 명령을 입력합니다. 첫 번째 줄에서 자리 표시자를 배포의 실제 값으로 바꿉니다. *logs-region*은 CloudWatch Logs가 있는 리전이고 *cluster-region*은 클러스터가 있는 리전입니다. *task-role-arn*은 사용 중인 ECS 태스크 역할의 ARN이고 *execution-role-arn*은 ECS 태스크 실행 역할의 ARN입니다.

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
Region=cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-
ecs-instance-metric/cwagent-ecs-instance-metric.json \
    | sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|
${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
    | xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-json
```

이어서 다음 명령을 실행하여 데몬 서비스를 시작합니다. *cluster-name* 및 *cluster-region*을 Amazon ECS 클러스터의 이름 및 리전으로 바꿉니다.

> **Important**
>
> 이 명령을 실행하기 전에 모든 용량 공급자 전략을 제거하세요. 그러지 않으면 명령이 작동하지 않습니다.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs create-service \
    --cluster ${ClusterName} \
    --service-name cwagent-daemon-service \
    --task-definition ecs-cwagent-daemon-service \
    --scheduling-strategy DAEMON \
```

```
    --region ${Region}
```

`An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent` 오류 메시지가 표시되면 `cwagent-daemon-service`라는 데몬 서비스를 이미 생성한 것입니다. 다음 명령을 예로 사용하여 먼저 해당 서비스를 삭제해야 합니다.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs delete-service \
    --cluster ${ClusterName} \
    --service cwagent-daemon-service \
    --region ${Region} \
    --force
```

(선택 사항) 고급 구성

선택적으로 SSM을 사용하여 EC2 인스턴스에서 호스트되는 Amazon ECS 클러스터의 CloudWatch 에이전트에 대한 다른 구성 옵션을 지정할 수 있습니다. 이러한 옵션은 다음과 같습니다.

- `metrics_collection_interval` – CloudWatch 에이전트가 지표를 수집하는 빈도(초)입니다. 기본값은 60입니다. 범위는 1~172,000입니다.
- `endpoint_override` – (선택 사항) 로그를 전송할 다른 엔드포인트를 지정합니다. VPC의 클러스터에서 게시하는 중이고 로그 데이터를 VPC 엔드포인트로 이동하려는 경우 이 작업을 원할 수 있습니다.

  `endpoint_override`의 값은 URL인 문자열이어야 합니다.
- `force_flush_interval` – 로그가 서버로 전송되기 전에 메모리 버퍼에 남아 있는 최대 시간(초)을 지정합니다. 이 필드에 대한 설정과 상관없이 버퍼 내 로그의 크기가 1MB에 도달하면 로그가 즉시 서버로 전송됩니다. 기본 값은 5초입니다.
- `region` – 기본적으로 에이전트는 Amazon ECS 컨테이너 인스턴스가 있는 리전과 동일한 리전에 지표를 게시합니다. 이를 무시하려면 여기에서 다른 리전을 지정합니다. 예: `"region" : "us-east-1"`

다음은 사용자 지정된 구성의 예입니다.

```
{
    "agent": {
```

```
        "region": "us-east-1"
    },
    "logs": {
        "metrics_collected": {
            "ecs": {
                "metrics_collection_interval": 30
            }
        },
        "force_flush_interval": 5
    }
}
```

Amazon ECS 컨테이너에서 CloudWatch 에이전트 구성을 사용자 지정하려면

1. AmazonSSMReadOnlyAccess 정책이 Amazon ECS 태스크 실행 역할에 연결되어 있는지 확인합니다. 이렇게 하려면 다음 명령을 입력합니다. 이 예에서는 Amazon ECS 태스크 실행 역할이 CWAgentECSExecutionRole이라고 가정합니다. 다른 역할을 사용하는 경우 다음 명령에서 해당 역할 이름으로 바꿉니다.

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   AmazonSSMReadOnlyAccess \
           --role-name CWAgentECSExecutionRole
   ```

2. 앞의 예제와 비슷한 사용자 지정된 구성 파일을 생성합니다. 이 파일의 이름을 `/tmp/ecs-cwagent-daemon-config.json`으로 지정합니다.
3. 다음 명령을 실행하여 이 구성을 파라미터 스토어에 넣습니다. *`cluster-region`*을 Amazon ECS 클러스터의 리전으로 바꿉니다. 이 명령을 실행하려면 AmazonSSMFullAccess 정책이 있는 사용자 또는 역할에 로그인해야 합니다.

   ```
   Region=cluster-region
   aws ssm put-parameter \
       --name "ecs-cwagent-daemon-service" \
       --type "String" \
       --value "`cat /tmp/ecs-cwagent-daemon-config.json`" \
       --region $Region
   ```

4. 작업 정의 파일을 로컬 파일(예 `/tmp/cwagent-ecs-instance-metric.json`)로 다운로드합니다.

   ```
   curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
   insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/
   ```

```
cwagent-ecs-instance-metric/cwagent-ecs-instance-metric.json -o /tmp/cwagent-ecs-
instance-metric.json
```

5. 작업 정의 파일을 수정합니다. 다음 섹션을 삭제합니다.

```
"environment": [
                {
                    "name": "USE_DEFAULT_CONFIG",
                    "value": "True"
                }
            ],
```

해당 섹션을 다음으로 바꿉니다.

```
"secrets": [
                {
                    "name": "CW_CONFIG_CONTENT",
                    "valueFrom": "ecs-cwagent-daemon-service"
                }
            ],
```

6. 다음 단계에 따라 에이전트를 데몬 서비스로 다시 시작합니다.

   a. 다음 명령을 실행합니다.

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
Region=cluster-region
cat /tmp/cwagent-ecs-instance-metric.json \
    | sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|
${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
    | xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-
json
```

   b. 데몬 서비스를 시작하려면 다음 명령을 실행합니다. *cluster-name* 및 *cluster-region*을 Amazon ECS 클러스터의 이름 및 리전으로 바꿉니다.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs create-service \
    --cluster ${ClusterName} \
```

```
    --service-name cwagent-daemon-service \
    --task-definition ecs-cwagent-daemon-service \
    --scheduling-strategy DAEMON \
    --region ${Region}
```

`An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent` 오류 메시지가 표시되면 `cwagent-daemon-service`라는 데몬 서비스를 이미 생성한 것입니다. 다음 명령을 예로 사용하여 먼저 해당 서비스를 삭제해야 합니다.

```
ClusterName=cluster-name
Region=Region
aws ecs delete-service \
    --cluster ${ClusterName} \
    --service cwagent-daemon-service \
    --region ${Region} \
    --force
```

AWS Distro for OpenTelemetry를 배포하여 Amazon ECS 클러스터의 EC2 인스턴스 수준 지표 수집

이 단원의 단계에 따라 AWS Distro for OpenTelemetry를 사용하여 Amazon ECS 클러스터의 EC2 인스턴스 수준 지표를 수집할 수 있습니다. AWS Distro for OpenTelemetry에 대한 자세한 내용은 AWS Distro for OpenTelemetry를 참조하세요.

다음 단계에서는 Amazon ECS를 실행 중인 클러스터가 이미 있다고 가정합니다. 이 클러스터는 EC2 시작 유형으로 배포되어야 합니다. Amazon ECS와 함께 AWS Distro for Open Telemetry를 사용하고 이러한 용도로 Amazon ECS 클러스터를 설정하는 방법에 대한 자세한 내용은 Amazon Elastic Container Service에서 ECS EC2 인스턴스 수준 지표에 대해 AWS Distro for OpenTelemetry Collector 설정을 참조하세요.

주제

- AWS CloudFormation을 사용한 빠른 설정
- 수동 설치 및 사용자 지정

AWS CloudFormation을 사용한 빠른 설정

EC2에서 Amazon ECS용 AWS Distro for OpenTelemetry Collector를 설치하기 위한 AWS CloudFormation 템플릿 파일을 다운로드합니다. 다음 curl 명령을 실행합니다.

```
curl -O https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/
deployment-template/ecs/aws-otel-ec2-instance-metrics-daemon-deployment-cfn.yaml
```

템플릿 파일을 다운로드한 후 파일을 열고 *PATH_TO_CloudFormation_TEMPLATE*을 템플릿 파일을 저장한 경로로 바꿉니다. 그런 후, 다음 명령과 같이 다음 파라미터를 내보내고 AWS CloudFormation 명령을 실행합니다.

- Cluster_Name – Amazon ECS 클러스터 이름
- AWS_Region – 데이터를 전송할 리전
- PATH_TO_CloudFormation_TEMPLATE – AWS CloudFormation 템플릿 파일을 저장한 경로
- command – AWS Distro for OpenTelemetry Collector가 Amazon EC2에서 Amazon ECS의 인스턴스 수준 지표를 수집하도록 하려면 이 파라미터에 `--config=/etc/ecs/otel-instance-metrics-config.yaml`을 지정해야 합니다.

```
ClusterName=Cluster_Name
Region=AWS_Region
command=--config=/etc/ecs/otel-instance-metrics-config.yaml
aws cloudformation create-stack --stack-name AOCECS-${ClusterName}-${Region} \
--template-body file://PATH_TO_CloudFormation_TEMPLATE \
--parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
ParameterKey=CreateIAMRoles,ParameterValue=True \
ParameterKey=command,ParameterValue=${command} \
--capabilities CAPABILITY_NAMED_IAM \
--region ${Region}
```

이 명령을 실행한 후 Amazon ECS 콘솔을 사용하여 태스크가 실행 중인지 확인합니다.

빠른 설정 문제 해결

AWS CloudFormation 스택의 상태를 확인하려면 다음 명령을 입력합니다.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack --stack-name AOCECS-$ClusterName-$Region --region
 $Region
```

`StackStatus`의 값이 `CREATE_COMPLETE` 또는 `CREATE_IN_PROGRESS`가 아닌 경우 스택 이벤트를 확인하여 오류를 찾습니다. 다음 명령을 입력합니다.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack-events --stack-name AOCECS-$ClusterName-$Region --
region $Region
```

AOCECS 데몬 서비스의 상태를 확인하려면 다음 명령을 입력합니다. 출력에서 runningCount가 배포 섹션의 desiredCount와 같은 것을 확인해야 합니다. 같지 않으면 출력에서 실패 섹션을 확인합니다.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs describe-services --services AOCECS-daemon-service --cluster $ClusterName --
region $Region
```

CloudWatch Logs 콘솔을 사용하여 에이전트 로그를 확인할 수도 있습니다. /aws/ecs/containerinsights/{ClusterName}/performance 로그 그룹을 찾습니다.

수동 설치 및 사용자 지정

이 단원의 단계에 따라 AWS Distro for OpenTelemetry를 수동으로 배포하여 Amazon EC2 인스턴스에서 호스트되는 Amazon ECS 클러스터에서 인스턴스 수준 지표를 수집합니다.

1단계: 필요한 역할 및 정책

두 가지 IAM 역할이 필요합니다. 아직 존재하지 않는 경우 생성해야 합니다. 이러한 역할에 대한 자세한 내용은 IAM 정책 생성 및 IAM 역할 생성을 참조하세요.

2단계: 태스크 정의 생성

태스크 정의를 생성하고 이를 사용하여 AWS Distro for OpenTelemetry를 데몬 서비스로 시작합니다.

태스크 정의 템플릿을 사용하여 태스크 정의를 생성하려면 AWS OTel Collector를 사용하여 EC2 인스턴스의 ECS EC2 태스크 정의 생성의 지침을 따르세요.

Amazon ECS 콘솔을 사용하여 태스크 정의를 생성하려면 AWS 콘솔을 통해 Amazon ECS EC2 인스턴스 지표에 대한 태스크 정의를 생성하여 AWS OTel Collector 설치의 지침을 따르세요.

3단계: 데몬 서비스 시작

AWS Distro for OpenTelemetry를 데몬 서비스로 시작하려면 데몬 서비스를 사용하여 Amazon Elastic Container Service(Amazon ECS)에서 태스크 실행의 지침을 따르세요.

(선택 사항) 고급 구성

선택적으로 SSM을 사용하여 Amazon EC2 인스턴스에서 호스트되는 Amazon ECS 클러스터의 AWS Distro for OpenTelemetry에 대한 다른 구성 옵션을 지정할 수 있습니다. 구성 파일 생성에 대한 자세한 내용은 사용자 지정 OpenTelemetry 구성을 참조하세요. 구성 파일에서 사용할 수 있는 옵션에 대한 자세한 내용은 AWS Container Insights Receiver를 참조하세요.

CloudWatch Logs에 로그를 전송하도록 FireLens 설정

Amazon ECS용 FireLens를 통해 태스크 정의 파라미터를 사용하여 로그를 Amazon CloudWatch Logs로 경로 지정함으로써 로그를 저장 및 분석할 수 있습니다. FireLens는 Fluent Bit 및 Fluentd와 함께 작동합니다. AWS for Fluent Bit 이미지를 제공하므로 이를 사용하거나 자체 Fluent Bit 또는 Fluentd 이미지를 사용할 수 있습니다. FireLens 구성을 사용한 Amazon ECS 태스크 정의 생성은 AWS SDK, AWS CLI, AWS Management Console을 통해 지원됩니다. CloudWatch Logs에 대한 자세한 내용은 CloudWatch Logs란? 단원을 참조하세요.

Amazon ECS용 FireLens를 사용할 때 고려할 주요 사항이 있습니다. 자세한 내용은 고려 사항 단원을 참조하세요.

AWS for Fluent Bit 이미지를 찾으려면 AWS for Fluent Bit 이미지 사용 단원을 참조하세요.

FireLens 구성을 사용하는 태스크 정의를 생성하려면 FireLens 구성을 사용하는 태스크 정의 생성 단원을 참조하세요.

예

다음 태스크 정의 예에서는 로그를 CloudWatch Logs 로그 그룹에 전달하는 로그 구성을 지정하는 방법을 보여 줍니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 Amazon CloudWatch Logs란? 섹션을 참조하세요.

로그 구성 옵션에서 로그 그룹 이름과 이 그룹이 속한 리전을 지정합니다. 사용자를 대신하여 Fluent Bit가 로그 그룹을 생성하게 하려면 `"auto_create_group":"true"`를 지정합니다. 또한 필터링을 지원하는 로그 스트림 접두사로 태스크 ID를 지정할 수도 있습니다. 자세한 정보는 Fluent Bit Plugin for CloudWatch Logs를 참조하세요.

```
{
 "family": "firelens-example-cloudwatch",
 "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
 "containerDefinitions": [
  {
   "essential": true,
   "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-bit:latest",
```

```
   "name": "log_router",
   "firelensConfiguration": {
    "type": "fluentbit"
   },
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-group": "firelens-container",
     "awslogs-region": "us-west-2",
     "awslogs-create-group": "true",
     "awslogs-stream-prefix": "firelens"
    }
   },
   "memoryReservation": 50
   },
   {
    "essential": true,
    "image": "nginx",
    "name": "app",
    "logConfiguration": {
     "logDriver":"awsfirelens",
     "options": {
     "Name": "cloudwatch",
     "region": "us-west-2",
     "log_key": "log",
                                  "log_group_name": "/aws/ecs/containerinsights/
$(ecs_cluster)/application",
     "auto_create_group": "true",
     "log_stream_name": "$(ecs_task_id)"
    }
   },
   "memoryReservation": 100
  }
 ]
}
```

Amazon EKS 및 Kubernetes에서 Container Insights 설정

Container Insights는 Amazon EKS 버전 1.23 이상에서 지원됩니다. 빠른 시작 설치 방법은 버전 1.24 이상에서만 지원됩니다.

Amazon EKS 또는 Kubernetes에서 Container Insights를 설정하는 전반적인 프로세스는 다음과 같습니다.

1. 필요한 사전 조건을 갖추었는지 확인합니다.
2. 클러스터의 Amazon CloudWatch Observability EKS 추가 기능, CloudWatch 에이전트 또는 AWS Distro for OpenTelemetry를 설정하여 CloudWatch에 지표를 전송합니다.

> **Note**
>
> Amazon EKS의 향상된 관찰 기능과 함께 Container Insights를 사용하려면 Amazon CloudWatch Observability EKS 애드온 또는 CloudWatch 에이전트를 사용해야 합니다. 이 버전의 Container Insights에 대한 자세한 정보는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights 섹션을 참조하세요.
> Fargate에서 Container Insights를 사용하려면 AWS Distro for OpenTelemetry를 사용해야 합니다. Amazon EKS의 향상된 관찰 기능 포함된 Container Insights는 Fargate에서 지원되지 않습니다.

> **Note**
>
> Container Insights는 이제 Amazon EKS 클러스터의 Windows 워커 노드를 지원합니다. Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights는 Windows에서도 지원됩니다. Windows의 Container Insights 활성화에 대한 자세한 정보는 Container Insights와 함께 CloudWatch 에이전트를 사용하고 관찰 기능 활성화 섹션을 참조하세요.

   CloudWatch Logs에 로그를 전송하도록 Fluent Bit 또는 Fluentd를 설정합니다. (Amazon CloudWatch Observability EKS 애드온을 설치하면 기본적으로 활성화됩니다.)

   이러한 단계를 빠른 시작 설정의 일부로 동시에 수행하거나(CloudWatch 에이전트를 사용하는 경우), 별도로 수행할 수 있습니다.
3. (선택 사항) Amazon EKS 제어 영역 로깅을 설정합니다.
4. (선택 사항) CloudWatch 에이전트를 클러스터의 StatsD 엔드포인트로 설정하여 CloudWatch에 StatsD 지표를 전송합니다.
5. (선택 사항) App Mesh Envoy 액세스 로그를 사용 설정합니다.

Container Insights의 원래 버전에서는 수집된 지표와 로그가 사용자 지정 지표로 청구됩니다. Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 사용하면 Container Insights 지표 및 로그는 저

장된 지표나 수집된 로그별로 요금이 부과되는 대신 관찰당 요금이 부과됩니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

주제

- 사전 조건 확인
- Container Insights와 함께 CloudWatch 에이전트를 사용하고 관찰 기능 활성화
- AWS Distro for OpenTelemetry 사용
- CloudWatch Logs에 로그 전송
- Amazon EKS 및 Kubernetes에서 Container Insights 업데이트 또는 삭제

사전 조건 확인

Amazon EKS 또는 Kubernetes에서 Container Insights를 설치하기 전에 다음을 확인합니다. 이러한 사전 조건은 Amazon EKS 클러스터에서 Container Insights를 설정하는 데 CloudWatch 에이전트를 사용하든 AWS Distro for OpenTelemetry를 사용하든 상관없이 적용됩니다.

- Amazon EKS 및 Kubernetes의 Container Insights를 지원하는 리전 중 하나에 노드가 연결되어 작동하는 Amazon EKS 또는 Kubernetes 클러스터가 있습니다. 지원되는 리전 목록은 Container Insights 단원을 참조하세요.
- `kubectl`을 설치하여 실행하고 있습니다. 자세한 내용은 Amazon EKS 사용 설명서의 `kubectl` 설치 단원을 참조하세요.
- Amazon EKS를 사용하는 대신 AWS에서 실행되는 Kubernetes를 사용하는 경우 다음 사전 조건도 충족해야 합니다.
  - Kubernetes 클러스터가 역할 기반 액세스 제어(RBAC)를 지원하는지 확인합니다. 자세한 내용은 Kubernetes 참조 문서의 RBAC 승인 사용을 참조하세요.
  - 여러분의 Kubelet에서는 Webhook 인증 모드를 지원해왔습니다. 자세한 내용은 Kubernetes 참조 문서의 Kubelet 인증/권한 부여를 참조하세요.

또한 Amazon EKS 작업자 노드가 지표 및 로그를 CloudWatch에 전송할 수 있도록 IAM 권한을 부여해야 합니다. 이렇게 하는 방법은 두 가지입니다.

- 작업자 노드의 IAM 역할에 정책을 연결합니다. 이는 Amazon EKS 클러스터와 다른 Kubernetes 클러스터 모두에 적용됩니다.
- 클러스터 서비스 계정의 IAM 역할을 사용하고 이 역할에 정책을 연결합니다. 이는 Amazon EKS 클러스터에만 적용됩니다.

첫 번째 옵션은 전체 노드의 CloudWatch에 권한을 부여하며, 서비스 계정의 IAM 역할을 사용하면 CloudWatch에 적절한 DaemonSet 포드에 대한 액세스 권한만 부여됩니다.

작업자 노드의 IAM 역할에 정책 연결

다음 단계에 따라 작업자 노드의 IAM 역할에 정책을 연결합니다. 이는 Amazon EKS 클러스터와 Amazon EKS 외부의 Kubernetes 클러스터 모두에 적용됩니다.

작업자 노드의 IAM 역할에 필요한 정책을 연결하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 작업자 노드 인스턴스 중 하나를 선택하고 설명에서 IAM 역할을 선택합니다.
3. IAM 역할 페이지에서 [정책 연결(Attach policies)]을 선택합니다.
4. 정책 목록에서 CloudWatchAgentServerPolicy 옆의 확인란을 선택합니다. 필요한 경우 검색 상자를 사용하여 이 정책을 찾습니다.
5. 정책 연결을 선택합니다.

Amazon EKS 외부에서 Kubernetes 클러스터를 실행하는 경우 작업자 노드에 연결된 IAM 역할이 아직 없을 수 있습니다. 이렇게 없는 경우에는 먼저, IAM 역할을 인스턴스에 연결한 다음, 이전 단계에서 설명한 대로 정책을 추가해야 합니다. 인스턴스에 역할을 연결하는 방법에 대한 자세한 내용은 'Windows 인스턴스용 Amazon EC2 사용 설명서'의 인스턴스에 IAM 역할 연결 단원을 참조하세요.

Amazon EKS 외부에서 Kubernetes 클러스터를 실행 중인데 지표에서 EBS 볼륨 ID를 수집하려는 경우 인스턴스에 연결된 IAM 역할에 다른 정책을 추가해야 합니다. 다음 내용을 인라인 정책으로 추가합니다. 자세한 내용은 'IAM 사용 설명서'의 IAM 자격 증명 권한 추가 및 제거 단원을 참조하세요.

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

IAM 서비스 계정 역할 사용

이 방법은 Amazon EKS 클러스터에서만 작동합니다.

IAM 서비스 계정 역할을 사용하여 CloudWatch에 권한을 부여하려면

1. 아직 설정하지 않았다면 클러스터에서 서비스 계정의 IAM 역할을 사용 설정합니다. 자세한 내용은 클러스터의 서비스 계정에 대한 IAM 역할 활성화를 참조하세요.
2. 아직 구성하지 않은 경우 IAM 역할을 사용하도록 서비스 계정을 구성합니다. 자세한 내용을 알아보려면 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성을 참조하세요.

   역할을 생성할 경우 역할에 대해 생성한 정책 외에도 CloudWatchAgentServerPolicy IAM 정책을 역할에 연결합니다. 또한 이 역할에 연결된 Kubernetes 서비스 계정은 CloudWatch 및 Fluent Bit daemonsets가 다음 단계에서 배포될 `amazon-cloudwatch` 네임스페이스에 생성되어야 합니다
3. 아직 연결하지 않았다면 IAM 역할을 클러스터의 서비스 계정과 연결합니다. 자세한 내용을 알아보려면 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성을 참조하세요.

Container Insights와 함께 CloudWatch 에이전트를 사용하고 관찰 기능 활성화

CloudWatch 에이전트를 사용하여 Amazon EKS 클러스터 또는 Kubernetes 클러스터에 Container Insights를 설정하려면 다음 섹션 중 하나의 지침을 따르세요. 빠른 시작 지침은 Amazon EKS 버전 1.24 이상에서만 지원됩니다.

> **Note**
>
> 다음 섹션 중 하나의 지침에 따라 Container Insights를 설치할 수 있습니다. 세 가지 지침을 모두 따를 필요는 없습니다.

주제

- Amazon CloudWatch Observability EKS 추가 기능 설치
- Amazon EKS 및 Kubernetes에서 Container Insights의 빠른 시작 설정
- 클러스터 지표를 수집하도록 CloudWatch 에이전트 설정

Amazon CloudWatch Observability EKS 추가 기능 설치

Amazon EKS 추가 기능을 사용하여 Amazon EKS의 관찰 기능이 향상된 Container Insights를 설치할 수 있습니다. 추가 기능은 CloudWatch 에이전트를 설치하여 클러스터에서 인프라 지표를 전송하고,

Fluent Bit를 설치하여 컨테이너 로그를 전송하고, CloudWatch Application Signals를 활성화하여 애플리케이션 성능 텔레메트리를 전송합니다.

Amazon EKS 추가 기능 버전 1.5.0 이상을 사용하는 경우 클러스터의 Linux 및 Windows 워커 노드 모두에서 Container Insights가 활성화됩니다. Amazon EKS의 Windows에서는 Application Signals가 현재 지원되지 않습니다.

Amazon EKS 추가 기능은 Amazon EKS 대신 Kubernetes를 실행하는 클러스터에서는 지원되지 않습니다.

Amazon CloudWatch Observability EKS 추가 기능에 대한 자세한 내용은 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 CloudWatch 에이전트 설치 섹션을 참조하세요.

Amazon CloudWatch Observability EKS 추가 기능 설치

1. 먼저 워커 노드에 CloudWatchAgentServerPolicy IAM 정책을 연결하여 필요한 권한을 설정합니다. 이를 위해 다음 명령을 입력합니다. *my-worker-node-role*을 Kubernetes 워커 노드에서 사용하는 IAM 역할로 대체합니다.

```
aws iam attach-role-policy \
--role-name my-worker-node-role \
--policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
```

2. 다음 명령을 입력하여 추가 기능을 설치합니다.

```
aws eks create-addon --cluster-name my-cluster-name --addon-name amazon-cloudwatch-observability
```

Amazon EKS 및 Kubernetes에서 Container Insights의 빠른 시작 설정

> **Important**
>
> Amazon EKS 클러스터에 Container Insights를 설치하는 경우 이 섹션의 지침을 따르는 대신 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 설치하는 것이 좋습니다. 또한 가속화된 컴퓨팅 네트워크를 검색하려면 Amazon CloudWatch Observability EKS 추가 기능을 사용해야 합니다. 자세한 정보와 지침은 Amazon CloudWatch Observability EKS 추가 기능 설치단원을 참조하세요.

Container Insights 설정을 완료하려면 이 단원의 빠른 시작 지침을 따르세요. Amazon EKS 클러스터에 설치하고 2023년 11월 6일 또는 그 이후에 이 섹션의 지침을 사용하는 경우 Amazon EKS의 관찰 기능이 향상된 Container Insights를 클러스터에 설치합니다.

> **Important**
>
> 이 단원의 단계를 완료하기 전에 먼저, IAM 권한을 포함한 사전 조건을 확인해야 합니다. 자세한 내용은 사전 조건 확인 단원을 참조하십시오.

또는 클러스터 지표를 수집하도록 CloudWatch 에이전트 설정 및 CloudWatch Logs에 로그 전송 단원의 지침을 따를 수 있습니다. 이러한 단원에서는 CloudWatch 에이전트가 Amazon EKS 및 Kubernetes와 작동하는 방식에 대한 추가 구성 세부 정보를 제공하지만, 추가 설치 단계를 수행해야 합니다.

Container Insights의 원래 버전에서는 수집된 지표와 로그가 사용자 지정 지표로 청구됩니다. Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 사용하면 Container Insights 지표 및 로그는 저장된 지표나 수집된 로그별로 요금이 부과되는 대신 관찰당 요금이 부과됩니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

> **Note**
>
> Amazon은 이제 Container Insights의 기본 로그 솔루션으로 상당한 성능 향상을 제공하는 Fluent Bit를 출시했습니다. 따라서 Fluentd 대신 Fluent Bit를 사용하는 것이 좋습니다.

CloudWatch 에이전트 운영자 및 Fluent Bit를 사용한 빠른 시작

Fluent Bit에는 두 가지 구성, 즉 최적화된 버전 및 Fluentd와 더 유사한 환경을 제공하는 버전이 있습니다. 빠른 시작 구성은 최적화된 버전을 사용합니다. Fluentd 호환 구성에 대한 자세한 내용은 Fluent Bit를 DaemonSet로 설정하여 CloudWatch Logs에 로그 전송 섹션을 참조하세요.

CloudWatch 에이전트 운영자는 Amazon EKS 클러스터에 설치되는 추가 컨테이너입니다. OpenTelemetry Operator for Kubernetes를 따라 모델링됩니다. 운영자는 클러스터에서 Kubernetes 리소스의 수명 주기를 관리합니다. Amazon EKS 클러스터에 CloudWatch 에이전트, DCGM Exporter(NVIDIA), AWS Neuron Monitor를 설치하고 관리합니다. Fluent Bit와 Windows용 CloudWatch 에이전트는 운영자가 관리하지 않아도 Amazon EKS 클러스터에 직접 설치됩니다.

보다 안전하고 기능이 풍부한 인증 기관 솔루션을 위해 CloudWatch 에이전트 운영자는 cert-manager를 요구하는데, 이는 Kubernetes에서의 TLS 인증서 관리를 위해 널리 채택되고 있는 솔루션입니다.

cert-manager를 사용하면 이러한 인증서를 획득, 갱신, 관리 및 사용하는 프로세스가 간소화됩니다. 인증서가 유효하고 최신 상태인지 확인하고 만료되기 전에 구성된 시간에 인증서 갱신을 시도합니다. cert-manager는 또한 AWS Certificate Manager Private Certificate Authority를 포함하여 지원되는 다양한 소스에서 인증서 발급을 용이하게 합니다.

빠른 시작을 사용하여 Container Insights 배포

1. 클러스터에 아직 설치되지 않은 경우 cert-manager를 설치합니다. 자세한 내용은 cert-manager Installation을 참조하세요.
2. 다음 명령을 입력하여 사용자 지정 리소스 정의(CRD)를 설치합니다.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/main/k8s-quickstart/cwagent-custom-resource-definitions.yaml | kubectl
 apply --server-side -f -
```

3. 다음 명령을 입력하여 운영자를 설치합니다. *my-cluster-name*을 Amazon EKS 또는 Kubernetes 클러스터의 이름으로 바꾸고, *my-cluster-region*을 로그가 게시되는 리전의 이름으로 바꿉니다. AWS 아웃바운드 데이터 전송 비용을 줄이기 위해 클러스터가 배포되는 리전과 동일한 리전을 사용하는 것이 좋습니다.

```
ClusterName=my-cluster-name
RegionName=my-cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
{{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
 apply -f -
```

   예를 들어 `MyCluster`라는 클러스터에서 Container Insights를 배포하고 로그 및 지표를 미국 서부(오레곤)에 게시하려면 다음 명령을 입력합니다.

```
ClusterName='MyCluster'
RegionName='us-west-2'
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
{{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
 apply -f -
```

Container Insights에서 마이그레이션

Amazon EKS 클러스터에 Container Insights가 이미 구성되어 있고 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights로 마이그레이션하려는 경우 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights로 업그레이드 섹션을 참조하세요.

Container Insights 삭제

빠른 시작 설정을 사용한 후 Container Insights를 제거하려면 다음 명령을 입력합니다.

```
ClusterName=my-cluster-name
RegionName=my-cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
{{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
 delete -f -
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/main/k8s-quickstart/cwagent-custom-resource-definitions.yaml | kubectl delete
 -f -
```

클러스터 지표를 수집하도록 CloudWatch 에이전트 설정

> **Important**
>
> Amazon EKS 클러스터에 Container Insights를 설치하는 경우 이 섹션의 지침을 따르는 대신 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 설치하는 것이 좋습니다. 자세한 정보와 지침은 Amazon CloudWatch Observability EKS 추가 기능 설치단원을 참조하세요.

Container Insights를 설정하여 지표를 수집하려면 Amazon EKS 및 Kubernetes에서 Container Insights의 빠른 시작 설정의 절차를 따르거나 이 단원의 절차를 따르면 됩니다. 다음 단계에서는 클러스터에서 지표를 수집할 수 있도록 CloudWatch 에이전트를 설정합니다.

Amazon EKS 클러스터에 설치하고 2023년 11월 6일 또는 그 이후에 이 섹션의 지침을 사용하는 경우 Amazon EKS의 관찰 기능이 향상된 Container Insights를 클러스터에 설치합니다.

1단계: CloudWatch의 네임스페이스 생성

다음 단계를 통해 CloudWatch에 대해 `amazon-cloudwatch`라는 Kubernetes 네임스페이스를 생성합니다. 이 네임스페이스를 이미 생성했다면 이 단계를 건너뛸 수 있습니다.

CloudWatch의 네임스페이스를 생성하려면

- 다음 명령을 입력합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

2단계: 클러스터에서 서비스 계정 생성

아직 서비스 계정이 없다면 다음 단계를 통해 CloudWatch 에이전트의 서비스 계정을 생성합니다.

CloudWatch 에이전트의 서비스 계정을 생성하려면

- 다음 명령을 입력합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
```

이전 단계를 따르지 않았지만 사용하려는 CloudWatch 에이전트에 대한 서비스 계정이 이미 있는 경우에는 다음 규칙이 있는지 확인해야 합니다. 뿐만 아니라 Container Insights 설치를 위한 나머지 단계에서 `cloudwatch-agent` 대신 이 서비스 계정의 이름을 사용해야 합니다.

```
rules:
  - apiGroups: [""]
    resources: ["pods", "nodes", "endpoints"]
    verbs: ["list", "watch"]
  - apiGroups: [ "" ]
    resources: [ "services" ]
    verbs: [ "list", "watch" ]
  - apiGroups: ["apps"]
    resources: ["replicasets", "daemonsets", "deployments", "statefulsets"]
    verbs: ["list", "watch"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["list", "watch"]
  - apiGroups: [""]
    resources: ["nodes/proxy"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes/stats", "configmaps", "events"]
    verbs: ["create", "get"]
```

```
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["cwagent-clusterleader"]
    verbs: ["get","update"]
  - nonResourceURLs: ["/metrics"]
    verbs: ["get", "list", "watch"]
```

3단계: CloudWatch 에이전트에 대한 ConfigMap 생성

다음 단계를 통해 CloudWatch 에이전트에 대한 ConfigMap을 생성합니다.

CloudWatch 에이전트에 대한 ConfigMap을 생성하려면

1. 다음 명령을 실행하여 kubectl 클라이언트 호스트로 ConfigMap YAML을 다운로드합니다.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-configmap.yaml
   ```

2. 다운로드한 YAML 파일을 다음과 같이 편집합니다.

   - cluster_name – kubernetes 섹션에서 {{cluster_name}}을 클러스터 이름으로 바꿉니다. {{}} 문자를 제거합니다. 또는 Amazon EKS 클러스터를 사용하는 경우 "cluster_name" 필드 및 값을 삭제할 수 있습니다. 그렇게 하면 CloudWatch 에이전트가 Amazon EC2 태그에서 클러스터 이름을 감지합니다.

3. (선택 사항) 다음과 같이 모니터링 요구사항에 따라 ConfigMap을 추가로 변경합니다.

   - metrics_collection_interval – kubernetes 섹션에서 에이전트가 지표를 수집하는 빈도를 지정할 수 있습니다. 기본값은 60초입니다. Kubelet의 기본 cadvisor 수집 간격은 15초이기 때문에 이 값을 15초 미만으로 설정해서는 안 됩니다.
   - endpoint_override – logs 섹션에서 기본 엔드포인트를 재정의하려는 경우 CloudWatch Logs 엔드포인트를 지정할 수 있습니다. VPC의 클러스터에서 게시 중인 데이터를 VPC 종단점으로 이동시키고 싶은 경우에 재정의를 원할 수 있습니다.
   - force_flush_interval – logs 섹션에서 로그 이벤트를 CloudWatch Logs에 게시하기 전에 배치 처리하는 간격을 지정할 수 있습니다. 기본값은 5초입니다.
   - region – 기본적으로 에이전트는 작업자 노드가 있는 리전에 지표를 게시합니다. 이를 재정의하기 위해 "region":"us-west-2"처럼 agent 섹션에서 region 필드를 추가할 수 있습니다.
   - statsd 섹션 - CloudWatch Logs 에이전트가 클러스터의 각 작업자 노드에서 StatsD 리스너로도 실행되도록 하려는 경우 다음 예와 같이 statsd 섹션을 metrics 섹션에 추가할 수 있습니다.

이 섹션의 다른 StatsD 옵션에 대한 자세한 내용은 StatsD를 사용하여 사용자 지정 지표 검색 단원을 참조하세요.

```
"metrics": {
  "metrics_collected": {
    "statsd": {
      "service_address":":8125"
    }
  }
}
```

JSON 섹션에 대한 전체 예는 다음과 같습니다.

```
{
    "agent": {
        "region": "us-east-1"
    },
    "logs": {
        "metrics_collected": {
            "kubernetes": {
                "cluster_name": "MyCluster",
                "metrics_collection_interval": 60
            }
        },
        "force_flush_interval": 5,
        "endpoint_override": "logs.us-east-1.amazonaws.com"
    },
    "metrics": {
        "metrics_collected": {
            "statsd": {
                "service_address": ":8125"
            }
        }
    }
}
```

4. 다음 명령을 실행하여 클러스터에서 ConfigMap을 생성합니다.

```
kubectl apply -f cwagent-configmap.yaml
```

4단계: DaemonSet로 CloudWatch 에이전트 배포

CloudWatch 에이전트의 설치를 완료하고 컨테이너 지표 수집을 시작하려면 다음 단계를 따르세요.

CloudWatch 에이전트를 DaemonSet로 배포하려면

1. • 클러스터에서 StatsD를 사용하려면 다음 명령을 입력하세요.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-
cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/
deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-
daemonset.yaml
```

   • StatsD를 사용하지 않으려면 다음 절차를 따르세요.

   a. 다음 명령을 실행하여 `kubectl` 클라이언트 호스트에 DaemonSet YAML을 다운로드합니다.

```
curl -O  https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-
mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

   b. 다음과 같이 `cwagent-daemonset.yaml` 파일에서 `port` 섹션의 주석 처리를 해제합니다.

```
ports:
  - containerPort: 8125
    hostPort: 8125
    protocol: UDP
```

   c. 다음 명령을 실행하여 클러스터에서 CloudWatch 에이전트를 배포합니다.

```
kubectl apply -f cwagent-daemonset.yaml
```

   d. 다음 명령을 실행하여 클러스터의 Windows 노드에서 CloudWatch 에이전트를 배포합니다. StatsD 리스너는 Windows의 CloudWatch 에이전트에서 지원되지 않습니다.

```
kubectl apply -f cwagent-daemonset-windows.yaml
```

2. 다음 명령을 실행하여 에이전트가 배포되었는지 확인합니다.

```
kubectl get pods -n amazon-cloudwatch
```

완료되면 CloudWatch 에이전트는 /aws/containerinsights/*Cluster_Name*/performance라는 로그 그룹을 생성하고 이 로그 그룹에 성능 로그 이벤트를 전송합니다. 또한 StatsD 리스너로서 에이전트를 설정하는 경우, 에이전트는 애플리케이션 Pod가 예약된 노드의 IP 주소를 통해 포트 8125에서 StatsD 지표를 수신합니다.

문제 해결

에이전트에서 배포가 올바르게 되지 않으면 다음을 수행해 보세요.

- 다음 명령을 실행하여 Pod 목록을 가져옵니다.

```
kubectl get pods -n amazon-cloudwatch
```

- 다음 명령을 실행하고 출력 하단에서 이벤트를 확인합니다.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- 다음 명령을 실행하여 로그를 확인합니다.

```
kubectl logs pod-name  -n amazon-cloudwatch
```

AWS Distro for OpenTelemetry 사용

AWS Distro for OpenTelemetry Collector를 사용하여 Amazon EKS 클러스터에서 지표를 수집하도록 Container Insights를 설정할 수 있습니다. AWS Distro for OpenTelemetry에 대한 자세한 내용은 AWS Distro for OpenTelemetry를 참조하세요.

> **Important**
>
> AWS Distro for OpenTelemetry를 사용하여 설치하는 경우 Container Insights는 설치하지만 Amazon EKS의 관찰 기능이 향상된 Container Insights는 얻을 수 없습니다. Amazon EKS의 향상된 관찰 기능을 통해 Container Insights에서 지원되는 세부 지표는 수집할 수 없습니다.

Container Insights를 설정하는 방법은 클러스터가 Amazon EC2 인스턴스 또는 AWS Fargate (Fargate) 중 어디에서 호스팅되는지에 따라 결정됩니다.

Amazon EC2에서 호스팅되는 Amazon EKS 클러스터

사전 조건 충족을 아직 확인하지 않은 경우 필요한 IAM 역할을 포함한 사전 조건을 충족했는지 확인합니다. 자세한 내용은 사전 조건 확인 단원을 참조하십시오.

Amazon은 Amazon EC2에서 Amazon Elastic Kubernetes Service 모니터링을 설정하는 데 사용할 수 있는 Helm 차트를 제공합니다. 이 모니터링은 지표의 경우 AWS Distro for OpenTelemetry(ADOT) Collector를 사용하고 로그의 경우 Fluent Bit를 사용합니다. 따라서 Helm 차트는 Amazon EC2에서 Amazon EKS를 사용하고 CloudWatch Container Insights로 전송하기 위해 지표 및 로그를 수집하려는 고객에게 유용합니다. 이 Helm 차트에 대한 자세한 내용은 Amazon CloudWatch Container Insights에 대한 EC2 지표 및 로그의 EKS용 ADOT Helm 차트를 참조하세요.

또는 이 섹션에 있는 지침을 사용할 수 있습니다.

먼저, 다음 명령을 입력하여 AWS Distro for OpenTelemetry Collector를 DaemonSet로 배포합니다.

```
curl https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/
deployment-template/eks/otel-container-insights-infra.yaml |
kubectl apply -f -
```

Collector가 실행 중인지 확인하려면 다음 명령을 입력합니다.

```
kubectl get pods -l name=aws-otel-eks-ci -n aws-otel-eks
```

이 명령의 출력에 `Running` 상태의 포드가 여러 개 포함되어 있다면 Collector가 실행 중이며 클러스터에서 지표를 수집 중입니다. Collector는 `aws/containerinsights/`*`cluster-name`*`/performance`라는 로그 그룹을 생성하고 이 그룹에 성능 로그 이벤트를 전송합니다.

CloudWatch에서 Container Insights 지표를 보는 방법에 대한 자세한 내용은 Container Insights 지표 보기 단원을 참조하세요.

AWS는 이 시나리오에 대한 GitHub 문서도 제공했습니다. Container Insights에서 게시한 지표 및 로그를 사용자 지정하려면 https://aws-otel.github.io/docs/getting-started/container-insights/eks-infra를 참조하세요.

Fargate에서 호스팅되는 Amazon EKS 클러스터

Fargate의 Amazon EKS 클러스터에 배포된 워크로드에서 시스템 지표를 수집하여 CloudWatch Container Insights로 전송하도록 ADOT Collector를 구성 및 배포하는 방법에 대한 지침은 AWS Distro for OpenTelemetry 설명서의 Container Insights EKS Fargate를 참조하세요.

CloudWatch Logs에 로그 전송

컨테이너에서 Amazon CloudWatch Logs로 로그를 전송하려면 Fluent Bit 또는 Fluentd를 사용하면 됩니다. 자세한 내용은 Fluent Bit 및 Fluentd를 참조하세요.

Fluentd를 아직 사용하고 있지 않다면 다음과 같은 이유로 Fluent Bit를 사용하는 것이 좋습니다.

- Fluent Bit는 Fluentd보다 리소스 공간이 더 작고 메모리 및 CPU와 같은 리소스를 더 효율적으로 사용합니다. 더 자세한 비교는 Fluent Bit와 Fluentd 성능 비교 단원을 참조하세요.
- Fluent Bit 이미지는 AWS에서 개발 및 유지 관리합니다. 이를 통해 AWS는 훨씬 더 빠르게 새로운 Fluent Bit 이미지 기능을 채택하고 문제에 대응할 수 있습니다.

주제

- Fluent Bit와 Fluentd 성능 비교
- Fluent Bit를 DaemonSet로 설정하여 CloudWatch Logs에 로그 전송
- (선택 사항) Fluentd를 DaemonSet로 설정하여 CloudWatch Logs에 로그 전송
- (선택 사항) Amazon EKS 제어 영역 로깅 설정
- (선택 사항) App Mesh Envoy 액세스 로그 사용 설정
- (선택 사항) 대규모 클러스터에 Use_Kubelet 기능 활성화

Fluent Bit와 Fluentd 성능 비교

다음 표는 메모리 및 CPU 사용량에서 Fluent Bit가 Fluentd보다 뛰어난 성능 이점이 있음을 보여 줍니다. 다음 수치는 참조용일 뿐이며 환경에 따라 바뀔 수 있습니다.

초당 로그	Fluentd CPU 사용량	Fluentd 호환 구성의 Fluent Bit CPU 사용량	최적화된 구성의 Fluent Bit CPU 사용량
100	0.35 vCPU	0.02 vCPU	0.02 vCPU
1,000	0.32 vCPU	0.14 vCPU	0.11 vCPU
5,000	0.85 vCPU	0.48 vCPU	0.30 vCPU
10,000	0.94 vCPU	0.60 vCPU	0.39 vCPU

초당 로그	Fluentd 메모리 사용량	Fluentd 호환 구성의 Fluent Bit 메모리 사용량	최적화된 구성의 Fluent Bit 메모리 사용량
100	153MB	46MB	37MB
1,000	270MB	45MB	40MB
5,000	320MB	55MB	45MB
10,000	375MB	92MB	75MB

Fluent Bit를 DaemonSet로 설정하여 CloudWatch Logs에 로그 전송

다음 단원을 통해 Fluent Bit를 배포하여 컨테이너에서 CloudWatch Logs로 로그를 전송할 수 있습니다.

주제

- 이미 Fluentd를 사용 중인 경우 차이점
- Fluent Bit 설정
- 여러 줄 로그 지원
- (선택 사항) Fluent Bit의 로그 볼륨 축소
- 문제 해결
- 대시보드

이미 Fluentd를 사용 중인 경우 차이점

이미 Fluentd를 사용하여 컨테이너에서 CloudWatch Logs로 로그를 전송하고 있다면 이 단원을 읽고 Fluentd와 Fluent Bit의 차이점을 확인하세요. Container Insights와 함께 Fluentd를 아직 사용하고 있지 않다면 Fluent Bit 설정으로 건너뛸 수 있습니다.

다음과 같이 Fluent Bit에 대한 기본 구성을 두 가지 제공합니다.

- Fluent Bit 최적화 구성 - Fluent Bit 모범 사례에 맞춰 조정된 구성입니다.
- Fluentd 호환 구성 - 가능한 한 Fluentd 동작에 맞춰 조정한 구성입니다.

다음 목록에서는 Fluentd와 각 Fluent Bit 구성 간의 차이점을 자세히 설명합니다.

- 로그 스트림 이름의 차이점 - Fluent Bit 최적화 구성을 사용하는 경우 로그 스트림 이름이 달라집니다.

  `/aws/containerinsights/Cluster_Name/application`에서

  - Fluent Bit 최적화 구성은 로그를 *kubernetes-nodeName*-`application.var.log.containers.`*kubernetes-podName*_*kubernetes-namespace*_*kubernetes-container-name*-*kubernetes-containerID*에 전송합니다.
  - Fluentd는 로그를 *kubernetes-podName*_*kubernetes-namespace*_*kubernetes-containerName*_*kubernetes-containerID*에 전송합니다.

  `/aws/containerinsights/Cluster_Name/host`에서

  - Fluent Bit 최적화 구성은 로그를 *kubernetes-nodeName*.*host-log-file*에 전송합니다.
  - Fluentd는 로그를 *host-log-file*-*Kubernetes-NodePrivateIp*에 전송합니다.

  `/aws/containerinsights/Cluster_Name/dataplane`에서

  - Fluent Bit 최적화 구성은 로그를 *kubernetes-nodeName*.*dataplaneServiceLog*에 전송합니다.
  - Fluentd는 로그를 *dataplaneServiceLog*-*Kubernetes-nodeName*에 전송합니다.
- Container Insights가 작성하는 `kube-proxy` 및 `aws-node` 로그 파일은 서로 다른 위치에 있습니다. Fluentd 구성에서는 `/aws/containerinsights/Cluster_Name/application`에 있습니다. Fluent Bit 최적화 구성에서는 `/aws/containerinsights/Cluster_Name/dataplane`에 있습니다.
- `pod_name` 및 `namespace_name`과 같은 대부분의 메타데이터는 Fluent Bit와 Fluentd에서 동일하지만, 다음은 다릅니다.
  - Fluent Bit 최적화 구성은 `docker_id`를 사용하며 Fluentd는 `Docker.container_id`를 사용합니다.
  - 두 Fluent Bit 구성은 모두 다음 메타데이터를 사용하지 않습니다. 즉, 다음 메타데이터는 Fluentd에만 있습니다. `container_image_id`, `master_url`, `namespace_id`, `namespace_labels`

Fluent Bit 설정

Fluent Bit를 설정하여 컨테이너에서 로그를 수집하려면 Amazon EKS 및 Kubernetes에서 Container Insights의 빠른 시작 설정의 절차를 따르거나 이 단원의 절차를 따르면 됩니다.

어느 방법이든 클러스터 노드에 첨부된 IAM 역할에 충분한 권한이 있어야 합니다. Amazon EKS 클러스터를 실행하는 데 필요한 권한에 대한 자세한 내용은 Amazon EKS 사용 설명서의 Amazon EKS IAM 정책, 역할 및 권한 단원을 참조하세요.

다음 단계에서는 Fluent Bit를 daemonSet로 설정하여 CloudWatch Logs에 로그를 전송합니다. 이 단계를 완료하면 Fluent Bit가 다음 로그 그룹을 생성합니다(아직 없는 경우).

> **Important**
>
> Container Insights에 이미 FluentD가 구성되어 있고 FluentD DaemonSet가 예상대로 실행되지 않는 경우(containerd 런타임을 사용하는 경우 발생할 수 있음), Fluent Bit를 설치하기 전에 Fluent Bit를 제거해야 Fluent Bit가 FluentD 오류 로그 메시지를 처리하지 않습니다. 그러지 않으면 Fluent Bit를 성공적으로 설치한 후 즉시 FluentD를 제거해야 합니다. Fluent Bit를 설치한 후 Fluentd를 제거하면 이 마이그레이션 과정에서 로깅을 계속 유지할 수 있습니다. CloudWatch Logs로 로그를 전송하는 데 Fluent Bit 또는 FluentD 중 하나만 필요합니다.

로그 그룹 이름	로그 소스
/aws/containerinsights/*Cluster_Name* /application	/var/log/containers 의 모든 로그 파일
/aws/containerinsights/*Cluster_Name* /host	/var/log/dmesg , /var/log/secure 및 /var/log/messages 에서의 로그
/aws/containerinsights/*Cluster_Name* /dataplane	kubelet.service , kubeproxy.service 및 docker.service 에 대한 /var/log/journal 에서의 로그.

Fluent Bit를 설치하여 컨테이너에서 CloudWatch Logs로 로그를 전송하려면

1. amazon-cloudwatch라는 네임스페이스가 아직 없는 경우 다음 명령을 입력하여 네임스페이스를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

2. 다음 명령을 실행하여 로그를 전송할 클러스터 이름 및 리전이 포함된 cluster-info라는 ConfigMap을 생성합니다. *cluster-name* 및 *cluster-region*을 클러스터의 이름 및 리전으로 바꿉니다.

```
ClusterName=cluster-name
RegionName=cluster-region
FluentBitHttpPort='2020'
FluentBitReadFromHead='Off'
[[ ${FluentBitReadFromHead} = 'On' ]] && FluentBitReadFromTail='Off'||
 FluentBitReadFromTail='On'
[[ -z ${FluentBitHttpPort} ]] && FluentBitHttpServer='Off' ||
 FluentBitHttpServer='On'
kubectl create configmap fluent-bit-cluster-info \
--from-literal=cluster.name=${ClusterName} \
--from-literal=http.server=${FluentBitHttpServer} \
--from-literal=http.port=${FluentBitHttpPort} \
--from-literal=read.head=${FluentBitReadFromHead} \
--from-literal=read.tail=${FluentBitReadFromTail} \
--from-literal=logs.region=${RegionName} -n amazon-cloudwatch
```

이 명령에서 플러그 인 지표 모니터링을 위한 FluentBitHttpServer는 기본적으로 활성화되어 있습니다. 명령에서 이를 비활성화하려면 명령의 세 번째 줄을 FluentBitHttpPort=''(빈 문자열)로 변경합니다.

또한 기본적으로 Fluent Bit는 테일에서 로그 파일을 읽으며 배포된 후 새 로그만 캡처합니다. 반대를 원하는 경우 FluentBitReadFromHead='On'으로 설정하면 파일 시스템의 모든 로그를 수집합니다.

3. 다음 명령 중 하나를 실행하여 Fluent Bit 데몬 세트를 다운로드한 후 클러스터에 배포합니다.

   - Linux 컴퓨터에 대한 Fluent Bit 최적화 구성을 원하는 경우 이 명령을 실행합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-
cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/
deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-
bit.yaml
```

   - Windows 컴퓨터에 대한 Fluent Bit 최적화 구성을 원하는 경우 이 명령을 실행합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-
cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/
```

```
deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit-windows.yaml
```

- Linux 컴퓨터를 사용 중이고 Fluentd와 더 유사한 Fluent Bit 구성을 원하는 경우 이 명령을 실행합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit-compatible.yaml
```

> **Important**
>
> Fluent Bit DaemonSet 구성은 기본적으로 로그 수준을 INFO로 설정하므로 CloudWatch Logs 수집 비용이 높아질 수 있습니다. 로그 수집 볼륨과 비용을 줄이려면 로그 수준을 오류로 변경하면 됩니다.
> 로그 볼륨을 줄이는 방법에 대한 자세한 내용은 (선택 사항) Fluent Bit의 로그 볼륨 축소 섹션을 참조하세요.

4. 다음 명령을 입력하여 배포를 검증합니다. 각 노드에는 fluent-bit-*라는 포드가 하나 있어야 합니다.

```
kubectl get pods -n amazon-cloudwatch
```

위 단계는 클러스터에 다음 리소스를 생성합니다.

- `amazon-cloudwatch` 네임스페이스의 `Fluent-Bit`이라는 서비스 계정. 이 서비스 계정은 Fluent Bit daemonSet를 실행하는 데 사용됩니다. 자세한 내용은 Kubernetes 참조 문서의 서비스 계정 관리를 참조하세요.
- `amazon-cloudwatch` 네임스페이스의 `Fluent-Bit-role`이라는 클러스터 역할. 이 클러스터 역할은 `Fluent-Bit` 서비스 계정에 대해 Pod 로그에서 `get`, `list` 및 `watch` 권한을 부여합니다. 자세한 내용은 Kubernetes 참조 문서의 API 개요를 참조하세요.
- `amazon-cloudwatch` 네임스페이스의 `Fluent-Bit-config`이라는 ConfigMap. 이 ConfigMap에는 Fluent Bit에서 사용할 구성이 포함되어 있습니다. 자세한 내용은 Kubernetes 작업 설명서의 ConfigMap을 사용하도록 Pod 구성을 참조하세요.

Fluent Bit 설정을 확인하려면 다음 단계를 따르세요.

Fluent Bit 설정 확인

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. Fluent Bit를 배포한 리전에 있는지 확인합니다.
4. 리전의 로그 그룹 목록을 확인합니다. 다음과 같은 모양이어야 합니다.
   - `/aws/containerinsights/Cluster_Name/application`
   - `/aws/containerinsights/Cluster_Name/host`
   - `/aws/containerinsights/Cluster_Name/dataplane`
5. 이러한 로그 그룹 중 하나로 이동하여 로그 스트림의 마지막 이벤트 시간을 확인합니다. 해당 시간이 Fluent Bit를 배포한 시점을 기준으로 최근인 경우 설정이 확인됩니다.

   `/dataplane` 로그 그룹을 생성하는 데 약간의 지연이 있을 수 있습니다. 이러한 로그 그룹은 Fluent Bit가 해당 로그 그룹에 대한 로그 전송을 시작할 때만 생성되기 때문에 약간의 지연 현상은 정상입니다.

여러 줄 로그 지원

여러 줄 로그와 함께 Fluent Bit를 사용하는 방법에 대한 자세한 내용은 Fluent Bit 설명서의 다음 섹션을 참조하세요.

- 여러 줄 구문 분석
- 여러 줄 및 컨테이너(v1.8)
- 여러 줄 코어(v1.8)
- 테일 입력에는 항상 여러 줄 사용

(선택 사항) Fluent Bit의 로그 볼륨 축소

기본적으로 CloudWatch에 Fluent Bit 애플리케이션 로그와 Kubernetes 메타데이터가 전송됩니다. CloudWatch에 전송되는 데이터의 볼륨을 줄이려면 이러한 데이터 원본 중 하나 또는 둘 모두가 CloudWatch에 전송되지 못하게 막으면 됩니다.

Fluent Bit 애플리케이션 로그를 중지하려면 `Fluent-Bit.yaml` 파일에서 다음 섹션을 제거합니다.

```
[INPUT]
```

```
        Name                tail
        Tag                 application.*
        Path                /var/log/containers/fluent-bit*
        Parser              docker
        DB                  /fluent-bit/state/flb_log.db
        Mem_Buf_Limit       5MB
        Skip_Long_Lines     On
        Refresh_Interval    10
```

Kubernetes 메타데이터를 제거하여 CloudWatch로 전송되는 로그 이벤트에 추가되지 못하게 하려면 `Fluent-Bit.yaml` 파일의 `application-log.conf` 섹션에 다음 필터를 추가합니다. *<Metadata_1>* 및 유사한 필드를 실제 메타데이터 식별자로 바꾸세요.

```
application-log.conf: |
    [FILTER]
        Name                nest
        Match               application.*
        Operation           lift
        Nested_under        kubernetes
        Add_prefix          Kube.

    [FILTER]
        Name                modify
        Match               application.*
        Remove              Kube.<Metadata_1>
        Remove              Kube.<Metadata_2>
        Remove              Kube.<Metadata_3>

    [FILTER]
        Name                nest
        Match               application.*
        Operation           nest
        Wildcard            Kube.*
        Nested_under        kubernetes
        Remove_prefix       Kube.
```

문제 해결

이러한 로그 그룹이 표시되지 않으며 리전이 올바른지 살펴보고 있는 경우 Fluent Bit daemonSet 포드의 로그를 확인하여 오류를 찾습니다.

다음 명령을 실행하여 상태가 `Running`인지 확인합니다.

```
kubectl get pods -n amazon-cloudwatch
```

로그에 IAM 권한과 관련된 오류가 있다면 클러스터 노드에 연결된 IAM 역할을 확인합니다. Amazon EKS 클러스터를 실행하는 데 필요한 권한에 대한 자세한 내용은 Amazon EKS 사용 설명서의 Amazon EKS IAM 정책, 역할 및 권한 단원을 참조하세요.

Pod 상태가 `CreateContainerConfigError`이면 다음 명령을 실행하여 정확한 오류를 가져옵니다.

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

대시보드

대시보드를 생성하여 실행 중인 각 플러그 인의 지표를 모니터링할 수 있습니다. 출력 오류 및 재시도/실패 비율 관련 데이터뿐만 아니라 입력 및 출력 바이트 관련 데이터, 레코드 처리 속도 관련 데이터를 확인할 수 있습니다. 이러한 지표를 보려면 Amazon EKS 및 Kubernetes 클러스터용 Prometheus 지표 수집과 함께 CloudWatch 에이전트를 설치해야 합니다. 대시보드 설정 방법에 대한 자세한 내용은 Amazon EKS 및 Kubernetes 클러스터에 Prometheus 지표 수집과 함께 CloudWatch 에이전트 설치 단원을 참조하세요.

> **Note**
>
> 이 대시보드를 설정하려면 먼저, Prometheus 지표에 대한 Container Insights를 설정해야 합니다. 자세한 내용은 Container Insights Prometheus 지표 모니터링 단원을 참조하십시오.

Fluent Bit Prometheus 지표에 대한 대시보드를 생성하려면

1. 환경 변수를 만들어서 다음 줄의 오른쪽에 있는 값을 배포와 일치하도록 바꿉니다.

   ```
   DASHBOARD_NAME=your_cw_dashboard_name
   REGION_NAME=your_metric_region_such_as_us-west-1
   CLUSTER_NAME=your_kubernetes_cluster_name
   ```

2. 다음 명령을 실행하여 대시보드를 생성합니다.

   ```
   curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
   insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
   prometheus/sample_cloudwatch_dashboards/fluent-bit/cw_dashboard_fluent_bit.json \
   | sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
   ```

```
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| xargs -0 aws cloudwatch put-dashboard --dashboard-name ${DASHBOARD_NAME} --
dashboard-body
```

(선택 사항) Fluentd를 DaemonSet로 설정하여 CloudWatch Logs에 로그 전송

> **Warning**
>
> Fluentd에 대한 Container Insights 지원은 현재 유지 관리 모드에 있습니다. 즉, AWS는 Fluentd에 대한 추가 업데이트를 제공하지 않으며 가까운 시일 내에 Fluentd를 사용 중지할 계획입니다. 또한 Container Insights의 현재 Fluentd 구성은 최신 개선 사항 및 보안 패치가 없는 이전 버전의 Fluentd 이미지인 `fluent/fluentd-kubernetes-daemonset:v1.10.3-debian-cloudwatch-1.0`을 사용하고 있습니다. 오픈 소스 커뮤니티에서 지원하는 최신 Fluentd 이미지는 fluentd-kubernetes-daemonset를 참조하세요.
> 가능하면 Container Insights와 함께 FluentBit를 사용하도록 마이그레이션하는 것이 좋습니다. FluentBit를 Container Insights의 로그 전달자로 사용하면 상당한 성능 향상을 얻을 수 있습니다.
> 자세한 내용은 Fluent Bit를 DaemonSet로 설정하여 CloudWatch Logs에 로그 전송 및 이미 Fluentd를 사용 중인 경우 차이점 섹션을 참조하세요.

Fluentd를 설정하여 컨테이너에서 로그를 수집하려면 Amazon EKS 및 Kubernetes에서 Container Insights의 빠른 시작 설정의 절차를 따르거나 이 섹션의 절차를 따르면 됩니다. 다음 단계에서는 FluentD를 DaemonSet로 설정하여 CloudWatch Logs에 로그를 전송합니다. 이 단계가 완료되고 로그 그룹이 아직 존재하지 않으면 Fluentd가 다음과 같이 로그 그룹을 생성합니다.

로그 그룹 이름	로그 소스
`/aws/containerinsights/`*`Cluster_Name`*`/application`	`/var/log/containers` 의 모든 로그 파일
`/aws/containerinsights/`*`Cluster_Name`*`/host`	`/var/log/dmesg` , `/var/log/secure` 및 `/var/log/messages` 에서의 로그
`/aws/containerinsights/`*`Cluster_Name`*`/dataplane`	`kubelet.service` , `kubeproxy.service` 및 `docker.service` 에 대한 `/var/log/journal` 에서의 로그.

1단계: CloudWatch의 네임스페이스 생성

다음 단계를 통해 CloudWatch에 대해 `amazon-cloudwatch`라는 Kubernetes 네임스페이스를 생성합니다. 이 네임스페이스를 이미 생성했다면 이 단계를 건너뛸 수 있습니다.

CloudWatch의 네임스페이스를 생성하려면

- 다음 명령을 입력합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

2단계: Fluentd 설치

Fluentd를 다운로드하여 이 프로세스를 시작합니다. 이러한 단계를 완료하면 배포된 에이전트에서 클러스터에 대해 다음과 같은 리소스가 생성됩니다.

- `amazon-cloudwatch` 네임스페이스의 `fluentd`이라는 서비스 계정. 이 서비스 계정은 Fluentd DaemonSet를 실행하는 데 사용됩니다. 자세한 내용은 Kubernetes 참조 문서의 서비스 계정 관리를 참조하세요.
- `amazon-cloudwatch` 네임스페이스의 `fluentd`이라는 클러스터 역할. 이 클러스터 역할은 `fluentd` 서비스 계정에 대해 Pod 로그에서 `get`, `list` 및 `watch` 권한을 부여합니다. 자세한 내용은 Kubernetes 참조 문서의 API 개요를 참조하세요.
- `amazon-cloudwatch` 네임스페이스의 `fluentd-config`이라는 ConfigMap. 이 ConfigMap에는 Fluentd에서 사용되는 구성이 포함되어 있습니다. 자세한 내용은 Kubernetes 작업 설명서의 ConfigMap을 사용하도록 Pod 구성을 참조하세요.

Fluentd를 설치하려면

1. 클러스터 이름과 로그가 전송될 AWS 리전을 포함하여 `cluster-info`이라는 ConfigMap을 생성합니다. 다음 명령을 실행하여 클러스터와 리전 이름으로 자리표시자를 업데이트합니다.

```
kubectl create configmap cluster-info \
--from-literal=cluster.name=cluster_name \
--from-literal=logs.region=region_name -n amazon-cloudwatch
```

2. 다음 명령을 실행하여 Fluentd DaemonSet를 다운로드한 후 클러스터에 배포합니다. 올바른 아키텍처의 컨테이너 이미지를 사용하고 있는지 확인합니다. 매니페스트 예는 x86 인스턴스에서만 작동하므로 클러스터에 Advanced RISC Machine(ARM) 인스턴스가 있는 경우 `CrashLoopBackOff`를 입력합니다. Fluentd DaemonSet에는 여러 기본 이미지에 대해 하나의 태그를 사용하고 컨테이너 런타임이 올바른 이미지를 가져오도록 하는 공식 다중 아키텍처 도커 이미지가 없습니다. Fluentd ARM 이미지는 `arm64` 접미사가 있는 다른 태그를 사용합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluentd/fluentd.yaml
```

> **Note**
>
> Fluentd 구성을 최적화하고 Kubernetes API 엔드포인트에 대한 Fluentd API 요청의 영향을 최소화하기 위한 최근 변경으로 인해 Kubernetes 필터에 대한 'Watch' 옵션이 기본적으로 사용 중지되었습니다. 자세한 내용은 fluent-plugin-kubernetes_metadata_filter를 참조하세요.

3. 다음 명령을 실행하여 배포를 확인합니다. 각 노드에는 `fluentd-cloudwatch-*`라는 이름의 Pod가 하나 포함되어야 합니다.

```
kubectl get pods -n amazon-cloudwatch
```

3단계: Fluentd 설정 확인

다음 단계를 이용하여 Fluentd 설정을 확인합니다.

Container Insights를 위한 Fluentd 설정을 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그 그룹을 선택합니다. 컨테이너에 Fluentd를 배포한 리전에 있는지 확인합니다.

   해당 리전의 로그 그룹 목록에서 다음을 확인해야 합니다.

   - `/aws/containerinsights/Cluster_Name/application`
   - `/aws/containerinsights/Cluster_Name/host`
   - `/aws/containerinsights/Cluster_Name/dataplane`

이러한 로그 그룹을 보면 Fluentd 설정이 확인됩니다.

여러 줄 로그 지원

2019년 8월 19일에 Fluentd에서 수집한 로그에 대한 여러 줄 로그 지원을 추가했습니다.

기본적으로 여러 줄 로그 항목 스타터는 공백이 포함되지 않은 임의의 문자입니다. 이는 공백이 포함되지 않은 문자로 시작하는 모든 로그 행이 새로운 여러 줄 로그 항목으로 간주됨을 뜻합니다.

자체 애플리케이션 로그에서 다른 여러 줄 스타터를 사용하는 경우 `fluentd.yaml` 파일에서 두 가지 사항을 변경하여 이를 지원할 수 있습니다.

먼저 `fluentd.yaml`의 `containers` 섹션에 있는 `exclude_path` 필드에 로그 파일의 경로 이름을 추가하여 기본 여러 줄 지원에서 해당 로그를 제외합니다. 다음은 예입니다.

```
<source>
  @type tail
  @id in_tail_container_logs
  @label @containers
  path /var/log/containers/*.log
  exclude_path ["full_pathname_of_log_file*", "full_pathname_of_log_file2*"]
```

그 다음 로그 파일에 대한 블록을 `fluentd.yaml` 파일에 추가합니다. 아래 예는 타임스탬프 정규 표현식을 여러 줄 스타터로 사용하는 CloudWatch 에이전트의 로그 파일에 사용됩니다. 이 블록을 복사하여 `fluentd.yaml`에 추가할 수 있습니다. 표시된 줄을 변경하여 사용하려는 애플리케이션 로그 파일 이름과 여러 줄 스타터를 반영하게 합니다.

```
<source>
  @type tail
  @id in_tail_cwagent_logs
  @label @cwagentlogs
  path /var/log/containers/cloudwatch-agent*
  pos_file /var/log/cloudwatch-agent.log.pos
  tag *
  read_from_head true
 <parse>
    @type json
    time_format %Y-%m-%dT%H:%M:%S.%NZ
```

```
 </parse>
</source>
```

```
<label @cwagentlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_cwagent
  </filter>

  <filter **>
    @type record_transformer
    @id filter_cwagent_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
 </filter>

  <filter **>
    @type concat
    key log
    multiline_start_regexp /^\d{4}[-/]\d{1,2}[-/]\d{1,2}/
    separator ""
    flush_interval 5
    timeout_label @NORMAL
 </filter>

 <match **>
    @type relabel
    @label @NORMAL
 </match>
</label>
```

(선택 사항) Fluentd의 로그 볼륨 축소

기본적으로 CloudWatch에 Fluentd 애플리케이션 로그와 Kubernetes 메타데이터가 전송됩니다. CloudWatch에 전송되는 데이터의 볼륨을 줄이려면 이러한 데이터 원본 중 하나 또는 둘 모두가 CloudWatch에 전송되지 못하게 막으면 됩니다.

Fluentd 애플리케이션 로그를 중단하려면 `fluentd.yaml` 파일에서 다음 섹션을 제거하세요.

```
<source>
```

```
  @type tail
  @id in_tail_fluentd_logs
  @label @fluentdlogs
  path /var/log/containers/fluentd*
  pos_file /var/log/fluentd.log.pos
  tag *
  read_from_head true
   <parse>
   @type json
   time_format %Y-%m-%dT%H:%M:%S.%NZ
  </parse>
</source>

<label @fluentdlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_fluentd
  </filter>

  <filter **>
    @type record_transformer
    @id filter_fluentd_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
   </filter>

   <match **>
     @type relabel
     @label @NORMAL
   </match>
</label>
```

Kubernetes 메타데이터를 제거하여 CloudWatch로 전송되는 로그 이벤트에 추가되지 못하게 하려면 `fluentd.yaml` 파일의 `record_transformer` 섹션에 한 줄을 추가합니다. 이 메타데이터를 제거하려는 로그 소스에서 다음 줄을 추가합니다.

```
remove_keys $.kubernetes.pod_id, $.kubernetes.master_url,
 $.kubernetes.container_image_id, $.kubernetes.namespace_id
```

예:

```
<filter **>
  @type record_transformer
  @id filter_containers_stream_transformer
  <record>
    stream_name ${tag_parts[3]}
  </record>
  remove_keys $.kubernetes.pod_id, $.kubernetes.master_url,
 $.kubernetes.container_image_id, $.kubernetes.namespace_id
</filter>
```

문제 해결

이러한 로그 그룹을 확인하지 않고 올바른 리전을 살펴보고 있는 경우, 오류를 찾으려면 Fluentd DaemonSet Pod에 대한 로그를 확인합니다.

다음 명령을 실행하여 상태가 Running인지 확인합니다.

```
kubectl get pods -n amazon-cloudwatch
```

이전 명령의 결과에서 fluentd-cloudwatch로 시작하는 Pod 이름을 적어둡니다. 다음 명령에서 이 Pod 이름을 사용합니다.

```
kubectl logs pod_name -n amazon-cloudwatch
```

로그에 IAM 권한과 관련된 오류가 있다면 클러스터 노드에 연결된 IAM 역할을 확인합니다. Amazon EKS 클러스터를 실행하는 데 필요한 권한에 대한 자세한 내용은 Amazon EKS 사용 설명서의 Amazon EKS IAM 정책, 역할 및 권한 단원을 참조하세요.

Pod 상태가 CreateContainerConfigError이면 다음 명령을 실행하여 정확한 오류를 가져옵니다.

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

포드 상태가 CrashLoopBackOff인 경우 Fluentd 컨테이너 이미지의 아키텍처가 Fluentd를 설치한 시점의 노드와 동일한지 확인합니다. 클러스터에 x86 및 ARM64 노드가 모두 있는 경우 kubernetes.io/arch 레이블을 사용하여 이미지를 올바른 노드에 배치할 수 있습니다. 자세한 내용은 kuberntes.io/arch를 참조하세요.

(선택 사항) Amazon EKS 제어 영역 로깅 설정

Amazon EKS를 사용하는 경우 선택적으로 Amazon EKS 제어 영역 로깅을 사용 설정하여 Amazon EKS 제어 영역에서 CloudWatch Logs로 직접 감사 및 진단 로그를 제공할 수 있습니다. 자세한 내용은 Amazon EKS 제어 영역 로깅 단원을 참조하세요.

(선택 사항) App Mesh Envoy 액세스 로그 사용 설정

Container Insights Fluentd를 설정하여 App Mesh Envoy 액세스 로그를 CloudWatch Logs에 전송할 수 있습니다. 자세한 내용을 알아보려면 Logging(로깅)을 참조하세요.

Envoy 액세스 로그를 CloudWatch Logs에 전송하려면

1. 클러스터에서 Fluentd를 설정합니다. 자세한 내용은 (선택 사항) Fluentd를 DaemonSet로 설정하여 CloudWatch Logs에 로그 전송 단원을 참조하십시오.
2. 가상 노드에 대한 Envoy 액세스 로그를 구성합니다. 지침은 Logging(로깅)을 참조하세요. 각 가상 노드에서 로그 경로를 **`/dev/stdout`**로 구성해야 합니다.

작업을 완료하면 envoy 액세스 로그가 `/aws/containerinsights/`*`Cluster_Name`*`/application` 로그 그룹으로 전송됩니다.

(선택 사항) 대규모 클러스터에 Use_Kubelet 기능 활성화

기본적으로 Use_Kubelet 기능은 FluentBit Kubernetes 플러그 인에서 비활성화되어 있습니다. 이 기능을 활성화하면 API 서버에 대한 트래픽이 줄어들고 API 서버에 병목 현상이 발생하는 문제를 완화할 수 있습니다. 대규모 클러스터에 해당 기능을 활성화하는 것이 좋습니다.

Use_Kubelet을 활성화하려면 먼저 노드와 노드 및 프록시 권한을 ClusterRole 구성에 추가합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: fluent-bit-role
rules:
  - nonResourceURLs:
      - /metrics
    verbs:
      - get
  - apiGroups: [""]
    resources:
      - namespaces
      - pods
```

```
      - pods/logs
      - nodes
      - nodes/proxy
    verbs: ["get", "list", "watch"]
```

DaemonSet 구성에서 이 기능을 사용하려면 호스트 네트워크 액세스 권한이 필요합니다. amazon/aws-for-fluent-bit의 이미지 버전이 2.12.0 이상이거나 fluent bit 이미지 버전이 1.7.2 이상이어야 합니다.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluent-bit
  namespace: amazon-cloudwatch
  labels:
    k8s-app: fluent-bit
    version: v1
    kubernetes.io/cluster-service: "true"
spec:
  selector:
    matchLabels:
      k8s-app: fluent-bit
  template:
    metadata:
      labels:
        k8s-app: fluent-bit
        version: v1
        kubernetes.io/cluster-service: "true"
    spec:
      containers:
      - name: fluent-bit
        image: amazon/aws-for-fluent-bit:2.19.0
        imagePullPolicy: Always
        env:
            - name: AWS_REGION
              valueFrom:
                configMapKeyRef:
                  name: fluent-bit-cluster-info
                  key: logs.region
            - name: CLUSTER_NAME
              valueFrom:
                configMapKeyRef:
                  name: fluent-bit-cluster-info
```

```
                key: cluster.name
          - name: HTTP_SERVER
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: http.server
          - name: HTTP_PORT
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: http.port
          - name: READ_FROM_HEAD
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: read.head
          - name: READ_FROM_TAIL
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: read.tail
          - name: HOST_NAME
            valueFrom:
              fieldRef:
                fieldPath: spec.nodeName
          - name: HOSTNAME
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.name
          - name: CI_VERSION
            value: "k8s/1.3.8"
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 500m
            memory: 100Mi
        volumeMounts:
        # Please don't change below read-only permissions
        - name: fluentbitstate
          mountPath: /var/fluent-bit/state
        - name: varlog
          mountPath: /var/log
```

```
          readOnly: true
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
        - name: fluent-bit-config
          mountPath: /fluent-bit/etc/
        - name: runlogjournal
          mountPath: /run/log/journal
          readOnly: true
        - name: dmesg
          mountPath: /var/log/dmesg
          readOnly: true
      terminationGracePeriodSeconds: 10
      hostNetwork: true
      dnsPolicy: ClusterFirstWithHostNet
      volumes:
      - name: fluentbitstate
        hostPath:
          path: /var/fluent-bit/state
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
      - name: fluent-bit-config
        configMap:
          name: fluent-bit-config
      - name: runlogjournal
        hostPath:
          path: /run/log/journal
      - name: dmesg
        hostPath:
          path: /var/log/dmesg
      serviceAccountName: fluent-bit
      tolerations:
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      - operator: "Exists"
        effect: "NoExecute"
      - operator: "Exists"
        effect: "NoSchedule"
```

쿠버네티스 플러그 인 구성은 다음과 유사합니다.

```
[FILTER]
        Name                kubernetes
        Match               application.*
        Kube_URL            https://kubernetes.default.svc:443
        Kube_Tag_Prefix     application.var.log.containers.
        Merge_Log           On
        Merge_Log_Key       log_processed
        K8S-Logging.Parser  On
        K8S-Logging.Exclude Off
        Labels              Off
        Annotations         Off
        Use_Kubelet         On
        Kubelet_Port        10250
        Buffer_Size         0
```

Amazon EKS 및 Kubernetes에서 Container Insights 업데이트 또는 삭제

이 단원의 절차를 통해 CloudWatch 에이전트 컨테이너 이미지를 업데이트하거나 Amazon EKS 또는 Kubernetes 클러스터에서 Container Insights를 제거할 수 있습니다.

주제

- Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights로 업그레이드
- CloudWatch 에이전트 컨테이너 이미지 업데이트
- Container Insights의 CloudWatch 에이전트 및 Fluent Bit 삭제

Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights로 업그레이드

> **Important**
>
> Amazon EKS 클러스터에 Container Insights를 업그레이드 또는 설치하는 경우 이 섹션의 지침을 따르는 대신 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 설치하는 것이 좋습니다. 또한 가속화된 컴퓨팅 지표를 검색하려면 Amazon CloudWatch Observability EKS 추가 기능을 사용해야 합니다. 자세한 정보와 지침은 Amazon CloudWatch Observability EKS 추가 기능 설치단원을 참조하세요.

Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights는 Container Insights의 최신 버전입니다. Amazon EKS를 실행하는 클러스터에서 세부 지표를 수집하고, 애플리케이션 및 인프라 텔레메트리를 자세히 살펴볼 수 있도록 즉시 사용할 수 있는 선별된 대시보드를 제공합니다. 이 버전의 Container Insights에 대한 자세한 정보는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights 섹션을 참조하세요.

Amazon EKS 클러스터에 원래 버전의 Container Insights를 설치한 후 관찰 기능이 향상된 새 버전으로 업그레이드하려면 이 섹션의 지침을 따르세요.

> **Important**
>
> 이 섹션의 단계를 완료하기 전에 먼저, cert-manager를 포함한 사전 조건을 확인해야 합니다. 자세한 내용은 CloudWatch 에이전트 운영자 및 Fluent Bit를 사용한 빠른 시작 단원을 참조하십시오.

Amazon EKS 클러스터를 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights로 업그레이드

1. 다음 명령을 입력하여 CloudWatch 에이전트 운영자를 설치합니다. *my-cluster-name*을 Amazon EKS 또는 Kubernetes 클러스터의 이름으로 바꾸고, *my-cluster-region*을 로그가 게시되는 리전의 이름으로 바꿉니다. AWS 아웃바운드 데이터 전송 비용을 줄이기 위해 클러스터가 배포되는 리전과 동일한 리전을 사용하는 것이 좋습니다.

   ```
   ClusterName=my-cluster-name
   RegionName=my-cluster-region
   curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
   container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
   {{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
    apply -f -
   ```

   리소스 충돌로 인해 장애가 발생하는 경우 이는 CloudWatch 에이전트와 Fluent Bit 및 ServiceAccount, ClusterRole, ClusterRoleBinding 등 Fluent Bit 구성 요소가 클러스터에 설치되어 있기 때문일 수 있습니다. CloudWatch 에이전트 운영자가 CloudWatch 에이전트 및 연결된 구성 요소를 설치하려고 할 때 콘텐츠의 변경 사항이 탐지되면 기본적으로 클러스터의 리소스 상태를 덮어쓰지 않도록 설치 또는 업데이트가 실패합니다. 이전에 클러스터에 설치한 Container Insights 설정을 포함하여 기존 CloudWatch 에이전트를 삭제한 다음 CloudWatch 에이전트 운영자를 설치하는 것이 좋습니다.
2. (선택 사항) 기존 사용자 지정 Fluent Bit 구성을 적용하려면 Fluent Bit 데몬셋과 연결된 Configmap을 업데이트해야 합니다. CloudWatch 에이전트 운영자는 Fluent Bit의 기본 구성을 제공하며, 사

용자는 필요에 따라 기본 구성을 재정의 또는 수정할 수 있습니다. 사용자 지정 구성을 적용하려면 다음 단계를 따릅니다.

a. 다음 명령을 입력하여 기존 구성을 엽니다.

```
kubectl edit cm fluent-bit-config -n amazon-cloudwatch
```

b. 파일을 변경한 다음 `:wq`를 입력하여 파일을 저장하고 편집 모드를 종료합니다.

c. 다음 명령을 입력하여 Fluent Bit를 다시 시작합니다.

```
kubectl rollout restart fluent-bit -n amazon-cloudwatch
```

CloudWatch 에이전트 컨테이너 이미지 업데이트

> **Important**
>
> Amazon EKS 클러스터에 Container Insights를 업그레이드 또는 설치하는 경우 이 섹션의 지침을 따르는 대신 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 설치하는 것이 좋습니다. 또한 가속화된 컴퓨팅 지표를 검색하려면 Amazon CloudWatch Observability EKS 추가 기능 또는 CloudWatch 에이전트 운영자를 사용해야 합니다. 자세한 정보와 지침은 Amazon CloudWatch Observability EKS 추가 기능 설치단원을 참조하세요.

컨테이너 이미지를 최신 버전으로 업데이트해야 하는 경우 이 단원의 절차를 따르세요.

컨테이너 이미지를 업데이트하려면

1. 다음 명령을 입력하여 `amazoncloudwatchagent` 사용자 지정 리소스 정의(CRD)가 이미 존재하는지 확인합니다.

```
kubectl get crds amazoncloudwatchagents.cloudwatch.aws.amazon.com -n amazon-
cloudwatch
```

   이 명령에서 CRD가 누락되었다는 오류를 반환하는 경우 클러스터에는 CloudWatch 에이전트 운영자로 구성된 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights가 없는 것입니다. 이 경우 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights로 업그레이드 섹션을 참조하세요.

2. 다음 명령을 입력하여 최신 `cwagent-version.yaml` 파일을 적용합니다.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/main/k8s-quickstart/cwagent-version.yaml | kubectl apply -f -
```

Container Insights의 CloudWatch 에이전트 및 Fluent Bit 삭제

Amazon EKS용 CloudWatch Observability 추가 기능 설치를 사용하여 Container Insights를 설치한 경우, 다음 명령을 입력하여 Container Insights와 CloudWatch 에이전트를 삭제할 수 있습니다.

> **Note**
>
> Amazon EKS 추가 기능은 이제 Windows 워커 노드의 Container Insights를 지원합니다. Amazon EKS 추가 기능을 삭제하면 Windows용 Container Insights도 삭제됩니다.

```
aws eks delete-addon –cluster-name my-cluster –addon-name amazon-cloudwatch-
observability
```

또는 CloudWatch 에이전트 및 Fluent Bit와 관련된 모든 리소스를 삭제하려면 다음 명령을 입력합니다. 이 명령에서 *My_Cluster_Name*은 Amazon EKS 또는 Kubernetes 클러스터의 이름이며, *My_Region*은 로그가 게시되는 리전의 이름입니다.

```
ClusterName=My_Cluster_Name
RegionName=My-Region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
{{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
 delete -f -
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/main/k8s-quickstart/cwagent-custom-resource-definitions.yaml | kubectl delete
 -f -
```

Container Insights 지표 보기

Container Insights를 설정하고 지표를 수집한 후에는 CloudWatch 콘솔에서 해당 지표를 볼 수 있습니다.

대시보드에 Container Insights 지표를 표시하려면 Container Insights 설정을 완료해야 합니다. 자세한 내용은 Container Insights 설정 단원을 참조하십시오.

이 절차에서는 Container Insights가 수집된 로그 데이터에서 자동으로 생성하는 지표를 보는 방법을 설명합니다. 이 단원의 나머지 부분에서는 데이터를 심층적으로 분석하고 CloudWatch Logs Insights를 사용하여 더 세분화된 수준에서 더 많은 지표를 확인하는 방법을 설명합니다.

Container Insights 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 인사이트를 선택한 다음 Container Insights를 선택합니다.
3. Container Insights 아래의 드롭다운 상자에서 성능 모니터링을 선택합니다.
4. 맨 위 근처의 드롭다운 상자를 사용하여 보려는 리소스 유형과 특정 리소스를 선택합니다.

Container Insights가 수집하는 지표에 대해 CloudWatch 경보를 설정할 수 있습니다. 자세한 내용은 Amazon CloudWatch 경보 사용 섹션을 참조하세요.

> **Note**
>
> 컨테이너화된 애플리케이션을 모니터링하도록 CloudWatch Application Insights를 이미 설정한 경우 Application Insights 대시보드가 Container Insights 대시보드 아래에 표시됩니다. 아직 Application Insights를 활성화하지 않은 경우 Container Insights 대시보드의 성능 보기 아래에 있는 Application Insights 자동 구성(Auto-configure Application Insights)을 선택해 활성화할 수 있습니다.
>
> Application Insights 및 컨테이너화 애플리케이션에 대한 자세한 내용은 Amazon ECS 및 Amazon EKS 리소스 모니터링을 위한 Application Insights 활성화 섹션을 참조하세요.

상위 기여자 보기

Container Insights 성능 모니터링의 일부 보기에서는 메모리나 CPU 또는 가장 최근의 활성 리소스를 기준으로 상위 기여자를 확인할 수도 있습니다. 페이지 상단 근처의 드롭다운 상자에서 다음 대시보드 중 하나를 선택할 경우 사용할 수 있습니다.

- ECS 서비스
- ECS 태스크
- EKS 네임스페이스
- EKS 서비스
- EKS 포드

이러한 유형의 리소스 중 하나를 살펴볼 때 페이지 하단에 처음에 CPU 사용량별로 정렬된 테이블이 표시됩니다. 메모리 사용량 또는 최근 활동별로 정렬되도록 변경할 수 있습니다. 테이블의 행 중 하나에 대해 자세히 보려면 해당 행 옆의 확인란을 선택한 다음, [작업(Actions)]을 선택하고 [작업(Actions) 메뉴의 옵션 중 하나를 선택합니다.

CloudWatch Logs Insights를 사용하여 Container Insights 데이터 보기

Container Insights는 임베디드 지표 형식을 사용한 성능 로그 이벤트를 사용하여 지표를 수집합니다. 로그는 CloudWatch Logs에 저장됩니다. CloudWatch는 CloudWatch 콘솔에서 볼 수 있는 로그에서 여러 지표를 자동으로 생성합니다. CloudWatch Logs Insights 쿼리를 사용하면 수집된 성능 데이터를 더 심층적으로 분석할 수도 있습니다.

CloudWatch Logs Insights에 대한 자세한 내용은 CloudWatch Logs Insights를 사용한 로그 데이터 분석 단원을 참조하세요. 쿼리에서 사용할 수 있는 로그 필드에 대한 자세한 내용은 Amazon EKS 및 Kubernetes의 Container Insights 성능 로그 이벤트 단원을 참조하세요.

CloudWatch Logs Insights를 사용하여 컨테이너 지표 데이터를 쿼리하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Insights를 선택합니다.

   화면 상단 근처에 쿼리 편집기가 있습니다. CloudWatch Logs Insights를 처음 열면 이 상자에는 최신 로그 이벤트 20개를 반환하는 기본 쿼리가 포함되어 있습니다.
3. 쿼리 편집기 위의 상자에서 쿼리할 Container Insights 로그 그룹을 선택합니다. 작업할 다음 예제 쿼리에서는 로그 그룹 이름이 performance로 끝나야 합니다.

   로그 그룹을 선택하면 CloudWatch Logs Insights가 로그 그룹의 데이터에서 필드를 자동으로 감지하고 오른쪽 창의 [검색된 필드(Discovered fields)]에 해당 필드를 표시합니다. 또한 이 로그 그룹의 로그 이벤트를 시간의 흐름에 따라 보여주는 막대 그래프도 표시합니다. 이 막대 그래프에서는 테이블에 표시된 이벤트뿐만 아니라 쿼리 및 시간 범위와 일치하는 로그 그룹 내 이벤트의 분포를 보여줍니다.
4. 쿼리 편집기에서 기본 쿼리를 다음 쿼리로 바꾸고 쿼리 실행을 선택합니다.

   ```
   STATS avg(node_cpu_utilization) as avg_node_cpu_utilization by NodeName
   | SORT avg_node_cpu_utilization DESC
   ```

   이 쿼리는 노드 목록을 평균적인 노드 CPU 이용률에 따라 정렬하여 보여줍니다.

5. 또 다른 예를 시도하려면 쿼리를 다음 쿼리로 바꾸고 쿼리 실행을 선택합니다. 추가 샘플 쿼리는 이 페이지 후반부에 나열되어 있습니다.

```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by
 PodName
| SORT avg_number_of_container_restarts DESC
```

   이 쿼리는 Pod 목록을 평균적인 컨테이너 재시작 횟수에 따라 정렬하여 보여줍니다.
6. 또 다른 쿼리를 시도하고 싶은 경우에는 화면 오른쪽의 목록에 필드를 포함시킬 수 있습니다. 쿼리 구문에 대한 자세한 내용은 CloudWatch Logs Insights 쿼리 구문 단원을 참조하세요.

리소스 목록을 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 리소스를 선택합니다.
3. 기본 보기는 Container Insights에서 모니터링하는 리소스 목록과 이러한 리소스에 대해 설정한 경보입니다. 리소스의 시각적 맵을 보려면 맵 보기를 선택합니다.
4. 맵 보기에서 맵의 리소스 위에 포인터를 일시 중지하여 해당 리소스에 대한 기본 지표를 볼 수 있습니다. 리소스를 선택하여 리소스에 대한 자세한 그래프를 볼 수 있습니다.

사용 사례: Amazon ECS 컨테이너의 태스크 수준 지표 보기

다음 예에서는 CloudWatch Logs Insights를 사용하여 Container Insights 로그를 더 심층적으로 분석하는 방법을 보여 줍니다. 더 많은 예는 Amazon ECS의 Amazon CloudWatch Container Insights 소개 블로그를 참조하세요.

Container Insights는 세분화된 태스크 수준에서 지표를 자동으로 생성하지 않습니다. 다음 쿼리는 CPU 및 메모리 사용량에 대한 태스크 수준 지표를 표시합니다.

```
stats avg(CpuUtilized) as CPU, avg(MemoryUtilized) as Mem by TaskId, ContainerName
| sort Mem, CPU desc
```

Container Insights에 대한 기타 샘플 쿼리

평균적인 컨테이너 재시작 횟수에 따라 정렬된 Pod 목록

```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by PodName
```

```
| SORT avg_number_of_container_restarts DESC
```

요청된 Pod와 실행 중인 Pod 간 비교

```
fields @timestamp, @message
| sort @timestamp desc
| filter Type="Pod"
| stats min(pod_number_of_containers) as requested,
 min(pod_number_of_running_containers) as running, ceil(avg(pod_number_of_containers-
pod_number_of_running_containers)) as pods_missing by kubernetes.pod_name
| sort pods_missing desc
```

클러스터 노드 실패 횟수

```
stats avg(cluster_failed_node_count) as CountOfNodeFailures
| filter Type="Cluster"
| sort @timestamp desc
```

컨테이너 이름별 애플리케이션 로그 오류

```
stats count() as countoferrors by kubernetes.container_name
| filter stream="stderr"
| sort countoferrors desc
```

Container Insights에서 수집한 지표

Container Insights는 Amazon ECS 및 Amazon ECS의 AWS Fargate에 대한 하나의 지표 세트와 Amazon EKS, Amazon EKS의 AWS Fargate 및 Kubernetes에 대한 다른 세트를 수집합니다.

컨테이너 작업이 일정 시간 동안 실행될 때까지는 지표가 표시되지 않습니다.

주제

- Amazon ECS Container Insights 지표
- Amazon EKS 및 Kubernetes Container Insights 지표

Amazon ECS Container Insights 지표

아래 표에는 Container Insights가 Amazon ECS용으로 수집하는 지표 및 측정기준이 나와 있습니다. 이러한 지표는 ECS/ContainerInsights 네임스페이스에 있습니다. 자세한 내용은 지표 단원을 참조하십시오.

콘솔에 Container Insights 지표가 보이지 않는 경우, Container Insights 설정을 완료했는지 확인합니다. Container Insights 설정이 완료되기 전에는 지표가 나타나지 않습니다. 자세한 내용은 Container Insights 설정 단원을 참조하십시오.

다음 지표는 클러스터 및 서비스 수준 지표를 위해 Amazon ECS에서 Container Insights 설정의 단계를 완료하면 사용할 수 있습니다.

지표 이름	측정기준	설명
`ContainerInstanceCount`	ClusterName	클러스터에 등록된 Amazon ECS 에이전트를 실행하는 EC2 인스턴스의 수입니다. 이 지표는 클러스터에서 Amazon ECS 작업을 실행하는 컨테이너 인스턴스에 대해서만 수집됩니다. Amazon ECS 작업이 없는 빈 컨테이너 인스턴스에 대해서는 수집되지 않습니다. 단위: 수

`CpuUtilized`	`TaskDefinitionFamily` , `ClusterName`
	
	`ServiceName` , `ClusterName`
	
	`ClusterName`
		사용 중인 측정기준 세트로 지정된 리소스의 작업에서 사용하는 CPU 단위입니다.
		
		이 지표는 작업 정의에 정의된 CPU 예약이 있는 작업에 대해서만 수집됩니다.
		
		단위: 없음

`CpuReserved`	`TaskDefinitionFamily` , `ClusterName`	사용 중인 측정기준 세트에서 지정한 리소스의 작

지표 이름	측정기준	설명

	`ServiceName` , `ClusterName`
	
	`ClusterName`
		업에서 예약된 CPU 단위입니다.
		
		이 지표는 작업 정의에 정의된 CPU 예약이 있는 작업에 대해서만 수집됩니다.
		
		단위: 없음

`DeploymentCount`	`ServiceName` , `ClusterName`	Amazon ECS 서비스의 배포 수입니다. 단위: 수
`DesiredTaskCount`	`ServiceName` , `ClusterName`	Amazon ECS 서비스에 대해 원하는 태스크 수입니다. 단위: 수

`EBSFilesystemSize`	`VolumeName` , `TaskDefinitionFamily` , `ClusterName`
	
	`TaskDefinitionFamily` , `ClusterName`
	
	`ServiceName` , `ClusterName`
		사용 중인 차원으로 지정된 리소스에 할당된 Amazon EBS 파일 시스템 스토리지의 총량(GB)
		
		이 지표는 플랫폼 버전 `1.4.0`을(를) 사용하는 Fargate에서 실행되는 Amazon ECS 인프라 또는 컨테이너 에이전트 버전 `1.79.0` 이상을 사용하는 Amazon EC2 인스턴스에서 실행되는 작업에만 사용할 수 있습니다.
		
		단위: 기가바이트(GB)

지표 이름	측정기준	설명
EBSFilesystemUtilized	VolumeName , TaskDefinitionFamily , ClusterName TaskDefinitionFamily , ClusterName ServiceName , ClusterName	사용 중인 차원으로 지정된 리소스에서 사용하는 Amazon EBS 파일 시스템 스토리지의 총량(GB) 이 지표는 플랫폼 버전 1.4.0을(를) 사용하는 Fargate에서 실행되는 Amazon ECS 인프라 또는 컨테이너 에이전트 버전 1.79.0 이상을 사용하는 Amazon EC2 인스턴스에서 실행되는 작업에만 사용할 수 있습니다. Fargate에서 실행되는 작업의 경우 Fargate는 Fargate만 사용하는 디스크 공간을 예약합니다. Fargate가 사용하는 공간에는 비용이 들지 않지만 df와 같은 도구를 사용하면 이 추가 스토리지를 확인할 수 있습니다. 단위: 기가바이트(GB)

지표 이름	측정기준	설명
EphemeralStorageReserved [1]	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	사용 중인 차원으로 지정된 리소스의 임시 스토리지에서 예약된 바이트 수입니다. 임시 스토리지는 컨테이너 루트 파일 시스템과 컨테이너 이미지 및 작업 정의에 정의된 모든 바인드 마운트 호스트 볼륨에 사용됩니다. 임시 스토리지의 양은 실행 중인 작업에서 변경할 수 없습니다. 이 지표는 Fargate Linux 플랫폼 버전 1.4.0 이상에서 실행되는 작업에만 사용할 수 있습니다. 단위: 기가바이트(GB)

지표 이름	측정기준	설명
EphemeralStorageUtilized [1]	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	사용 중인 차원으로 지정된 리소스의 임시 스토리지에서 사용된 바이트 수입니다. 임시 스토리지는 컨테이너 루트 파일 시스템과 컨테이너 이미지 및 작업 정의에 정의된 모든 바인드 마운트 호스트 볼륨에 사용됩니다. 임시 스토리지의 양은 실행 중인 작업에서 변경할 수 없습니다. 이 지표는 Fargate Linux 플랫폼 버전 1.4.0 이상에서 실행되는 작업에만 사용할 수 있습니다. 단위: 기가바이트(GB)
MemoryUtilized	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	사용 중인 측정기준 세트로 지정된 리소스의 작업에서 사용 중인 메모리입니다. 이 지표는 작업 정의에 정의된 메모리 예약이 있는 작업에 대해서만 수집됩니다. 단위: 메가바이트

지표 이름	측정기준	설명
MemoryReserved	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	사용 중인 측정기준 세트에서 지정한 리소스의 작업에서 예약된 메모리입니다. 이 지표는 작업 정의에 정의된 메모리 예약이 있는 작업에 대해서만 수집됩니다. 단위: 메가바이트
NetworkRxBytes	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	사용중인 측정기준에서 지정한 리소스에서 수신된 바이트의 숫자입니다. 이 지표는 Docker 런타임에서 가져옵니다. 이 지표는 awsvpc 또는 bridge 네트워크 모드를 사용하는 태스크의 컨테이너에 대해서만 제공됩니다. 단위: 바이트/초

지표 이름	측정기준	설명
NetworkTxBytes	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	사용중인 측정기준에서 지정한 리소스에서 전송된 바이트의 숫자입니다. 이 지표는 Docker 런타임에서 가져옵니다. 이 지표는 awsvpc 또는 bridge 네트워크 모드를 사용하는 태스크의 컨테이너에 대해서만 제공됩니다. 단위: 바이트/초
PendingTaskCount	ServiceName , ClusterName	현재 PENDING 상태인 작업의 숫자입니다. 단위: 수
RunningTaskCount	ServiceName , ClusterName	현재 RUNNING 상태인 작업의 숫자입니다. 단위: 수
ServiceCount	ClusterName	클러스터의 서비스 숫자입니다. 단위: 수

지표 이름	측정기준	설명
StorageReadBytes	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	사용 중인 측정기준에서 지정한 리소스의 인스턴스에 있는 스토리지에서 읽힌 바이트의 숫자입니다. 스토리지 디바이스의 읽기 바이트는 여기에 포함되지 않습니다. 이 지표는 Docker 런타임에서 가져옵니다. 단위: 바이트
StorageWriteBytes	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	사용 중인 측정기준에서 지정한 리소스의 스토리지에서 쓰여진 바이트의 숫자입니다. 이 지표는 Docker 런타임에서 가져옵니다. 단위: 바이트
TaskCount	ClusterName	클러스터에서 실행 중인 태스크의 수입니다. 단위: 수
TaskSetCount	ServiceName , ClusterName	서비스의 작업 세트 숫자입니다. 단위: 수

> **Note**
>
> EphemeralStorageReserved 및 EphemeralStorageUtilized 지표는 Fargate Linux 플랫폼 버전 1.4.0 이상에서 실행되는 작업에만 사용할 수 있습니다.

Fargate는 디스크 공간을 예약합니다. Fargate에서만 사용됩니다. 이에 대한 요금은 청구되지 않습니다. 이러한 지표에는 표시되지 않습니다. 그러나 `df` 등의 다른 도구에서는 이 추가 스토리지를 볼 수 있습니다.

다음 지표는 CloudWatch 에이전트를 배포하여 Amazon ECS의 EC2 인스턴스 수준 지표 수집의 단계를 완료하면 사용할 수 있습니다.

지표 이름	측정기준	설명
`instance_cpu_limit`	`ClusterName`	클러스터의 단일 EC2 인스턴스에 할당할 수 있는 최대 CPU 단위 수입니다. 단위: 없음

`instance_cpu_reserved_capacity`	`ClusterName`
	
	`InstanceId` , `ContainerInstanceId` , `ClusterName`
		클러스터의 단일 EC2 인스턴스에서 현재 예약 중인 CPU의 비율입니다.
		
		단위: 백분율

`instance_cpu_usage_total`	`ClusterName`	클러스터의 단일 EC2 인스턴스에서 사용 중인 CPU 단위 수입니다. 단위: 없음

`instance_cpu_utilization`	`ClusterName`
	
	`InstanceId` , `ContainerInstanceId` , `ClusterName`
		클러스터의 단일 EC2 인스턴스에서 사용 중인 CPU 단위의 총 비율입니다.
		
		단위: 백분율

지표 이름	측정기준	설명
instance_filesystem_utilization	ClusterName InstanceId , ContainerInstanceId , ClusterName	클러스터의 단일 EC2 인스턴스에서 사용 중인 파일 시스템 용량의 총 비율입니다. 단위: 백분율
instance_memory_limit	ClusterName	이 클러스터의 단일 EC2 인스턴스에 할당할 수 있는 최대 메모리 양(바이트)입니다. 단위: 바이트
instance_memory_reserved_capacity	ClusterName InstanceId , ContainerInstanceId , ClusterName	클러스터의 단일 EC2 인스턴스에서 현재 예약 중인 메모리의 비율입니다. 단위: 백분율
instance_memory_utilization	ClusterName InstanceId , ContainerInstanceId , ClusterName	클러스터의 단일 EC2 인스턴스에서 사용 중인 메모리의 총 비율입니다. 단위: 백분율
instance_memory_working_set	ClusterName	클러스터의 단일 EC2 인스턴스에서 사용 중인 메모리의 양(바이트)입니다. 단위: 바이트

지표 이름	측정기준	설명
instance_network_total_bytes	ClusterName	클러스터의 단일 EC2 인스턴스에서 네트워크를 통해 전송 및 수신된 초당 총 바이트 수입니다. 단위: 바이트/초
instance_number_of_running_tasks	ClusterName	클러스터의 단일 EC2 인스턴스에서 실행 중인 작업 수입니다. 단위: 수

Amazon EKS 및 Kubernetes Container Insights 지표

아래 표에는 Container Insights가 Amazon EKS 및 쿠버네티스용으로 수집하는 지표 및 측정 기준이 나와 있습니다. 이러한 지표는 ContainerInsights 네임스페이스에 있습니다. 자세한 내용은 지표 단원을 참조하십시오.

콘솔에 Container Insights 지표가 보이지 않는 경우, Container Insights 설정을 완료했는지 확인합니다. Container Insights 설정이 완료되기 전에는 지표가 나타나지 않습니다. 자세한 내용은 Container Insights 설정 단원을 참조하십시오.

Amazon EKS 추가 기능 버전 1.5.0 이상 또는 CloudWatch 에이전트 버전 1.300035.0을 사용 중인 경우 다음 표에 나열된 대부분의 지표가 Linux와 Windows 노드 모두에 대해 수집됩니다. 표의 지표 이름 열을 참조하여 Windows에서 수집되지 않는 지표를 확인하세요.

Container Insights의 원래 버전에서 지표는 사용자 지정 지표로 청구됩니다. Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 사용하면 Container Insights 지표는 저장된 지표나 수집된 로그별로 요금이 부과되는 대신 관찰당 요금이 부과됩니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

> **Note**
>
> Windows에서는 호스트 프로세스 컨테이너에 대해 pod_network_rx_bytes 및 pod_network_tx_bytes 등의 네트워크 지표는 수집되지 않습니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
cluster_failed_node_count	ClusterName		클러스터의 실패한 작업자 노드의 숫자입니다. '노드 조건' 문제를 겪고 있는 경우 노드가 실패한 것으로 간주됩니다. 자세한 내용은 Kubernetes 설명서에서 조건을 참조하세요.
cluster_node_count	ClusterName		클러스터의 작업자 노드의 총 숫자입니다.
namespace_number_of_running_pods	Namespace ClusterName ClusterName		사용 중인 측정기준에서 지정한 리소스의 네임스페이스당 실행 중인 Pod 숫자입니다.
node_cpu_limit	ClusterName	ClusterName , InstanceId , NodeName	클러스터에서 단일 노드에 할당할 수 있는 최대 CPU 단위 숫자입니다.
node_cpu_reserved_capacity	NodeName, ClusterName , InstanceId ClusterName		kubelet, kube-proxy, Docker 등 노드 구성 요소에 예약된 CPU 단위의 비율입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
			공식: `node_cpu_request / node_cpu_limit` **Note** `node_cpu_request` 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.
`node_cpu_usage_total`	`ClusterName`	`ClusterName` , `InstanceId` , `NodeName`	클러스터의 노드에서 사용 중인 CPU 단위의 숫자입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_cpu_utilization	NodeName, ClusterName , InstanceId ClusterName		클러스터의 노드에서 사용 중인 CPU 단위의 총 백분율입니다. 공식: node_cpu_usage_total / node_cpu_limit

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_filesystem_utilization	NodeName, ClusterName , InstanceId ClusterName		클러스터에서 노드에 사용하는 파일 시스템 용량의 총 백분율입니다. 공식: node_filesystem_usage / node_filesystem_capacity **Note** node_filesystem_usage 및 node_filesystem_capacity 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
			원을 참조하십시오.
node_memory_limit	ClusterName	ClusterName , InstanceId , NodeName	클러스터에서 단일 노드로 할당될 수 있는 최대 메모리의 양(바이트)입니다.
node_filesystem_inodes 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다. Windows에서는 제공되지 않습니다.		ClusterName ClusterName , InstanceId , NodeName	노드의 총 아이노드(사용 및 미사용) 수입니다.
node_filesystem_inodes_free 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다. Windows에서는 제공되지 않습니다.		ClusterName ClusterName , InstanceId , NodeName	노드의 미사용 아이노드 수입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_memory_reserved_capacity	NodeName, ClusterName , InstanceId ClusterName		클러스터의 노드에서 현재 사용 중인 메모리의 비율입니다. 공식: node_memory_request / node_memory_limit **Note** node_memory_request 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_memory_utilization	NodeName, ClusterName , InstanceId ClusterName		한 개 또는 여러 개의 노드에서 현재 사용 중인 메모리의 비율입니다. 노드 메모리 사용량을 노드 메모리 제한으로 나눈 백분율입니다. 공식: node_memory_working_set / node_memory_limit 입니다.
node_memory_working_set	ClusterName	ClusterName , InstanceId , NodeName	클러스터의 노드 작업 세트에서 사용하는 메모리의 양(바이트)입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_network_total_bytes	NodeName, ClusterName , InstanceId ClusterName		클러스터에서 노드당 네트워크를 통해 전송 및 수신된 초당 바이트의 합계 수치입니다. 공식: node_network_rx_bytes + node_network_tx_bytes **Note** node_network_rx_bytes 및 node_network_tx_bytes 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_number_of_running_containers	NodeName, ClusterName , InstanceId ClusterName		클러스터에서 노드당 실행 중인 컨테이너의 숫자입니다.
node_number_of_running_pods	NodeName, ClusterName , InstanceId ClusterName		클러스터에서 노드당 실행 중인 Pod 숫자입니다.
node_status_allocatable_pods 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	할당 가능한 리소스를 기준으로 노드에 할당할 수 있는 포드 수이며 이는 시스템 대몬(daemon) 예약 및 하드 제거 임곗값을 고려한 후 노드 용량의 나머지 부분으로 정의됩니다.
node_status_capacity_pods 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	용량에 따라 노드에 할당할 수 있는 포드 수입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_status_condition_ready 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	노드 상태 조건 Ready이 참인지 여부를 나타냅니다.
node_status_condition_memory_pressure 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	노드 상태 조건 MemoryPressure 이 참인지 여부를 나타냅니다.
node_status_condition_pid_pressure 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	노드 상태 조건 PIDPressure 이 참인지 여부를 나타냅니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_status_condition_disk_pressure 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	노드 상태 조건 OutOfDisk 이 참인지 여부를 나타냅니다.
node_status_condition_unknown 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	노드 상태 조건 중 알 수 없는 상태가 있는지 여부를 나타냅니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_interface_network_rx_dropped 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	노드의 네트워크 인터페이스에서 수신한 후 삭제한 패킷 수입니다.
node_interface_network_tx_dropped 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , InstanceId , NodeName	전송될 예정이었으나 노드의 네트워크 인터페이스에서 삭제된 패킷 수입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
node_diskio_io_service_bytes_total 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다. Windows에서는 제공되지 않습니다.		ClusterName ClusterName , InstanceId , NodeName	노드의 모든 I/O 작업에서 전송된 총 바이트 수입니다.
node_diskio_io_serviced_total 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다. Windows에서는 제공되지 않습니다.		ClusterName ClusterName , InstanceId , NodeName	노드의 총 I/O 작업 횟수입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_cpu_reserved_capacity	PodName, 네임스페이스, ClusterName ClusterName	ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , Service	클러스터에서 Pod별로 예약된 CPU 용량입니다. 공식: pod_cpu_request / node_cpu_limit **Note** pod_cpu_request 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_cpu_utilization	PodName, 네임스페이스, ClusterName 네임스페이스, ClusterName 서비스, 네임스페이스, ClusterName ClusterName	ClusterName , Namespace , PodName, FullPodName	Pod에서 사용 중인 CPU 단위의 비율입니다. 공식: pod_cpu_usage_total / node_cpu_limit **Note** pod_cpu_usage_total 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_cpu_utilization_over_pod_limit	PodName, 네임스페이스, ClusterName 네임스페이스, ClusterName 서비스, 네임스페이스, ClusterName ClusterName	ClusterName , Namespace , PodName, FullPodName	포드 제한을 기준으로 포드에서 사용 중인 CPU 단위의 백분율입니다. 공식: pod_cpu_usage_total / pod_cpu_limit **Note** pod_cpu_usage_total 및 pod_cpu_limit 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_memory_reserved_capacity	PodName, 네임스페이스, ClusterName ClusterName	ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , Service	포드에 예약된 메모리의 비율입니다. 공식: pod_memory_request / node_memory_limit **Note** pod_memory_request 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_memory_utilization	PodName, 네임스페이스, ClusterName 네임스페이스, ClusterName 서비스, 네임스페이스, ClusterName ClusterName	ClusterName , Namespace , PodName, FullPodName	한 개 또는 여러 개의 Pod에서 현재 사용 중인 메모리의 비율입니다. 공식: pod_memory_working_set / node_memory_limit **Note** pod_memory_working_set 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_memory_utilization_over_pod_limit	PodName, 네임스페이스, ClusterName 네임스페이스, ClusterName 서비스, 네임스페이스, ClusterName ClusterName	ClusterName , Namespace , PodName, FullPodName	포드 제한을 기준으로 포드에서 사용 중인 메모리의 백분율입니다. 포드의 컨테이너에 메모리 제한이 정의되지 않은 경우 이 지표는 표시되지 않습니다. 공식: pod_memory_working_set / pod_memory_limit **Note** pod_memory_working_set 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
			원을 참조하십시오.
pod_network_rx_bytes	PodName, 네임스페이스, ClusterName 네임스페이스, ClusterName 서비스, 네임스페이스, ClusterName ClusterName	ClusterName , Namespace , PodName, FullPodName	Pod에서 네트워크를 통해 수신 중인 초당 바이트 수입니다. 공식: sum(pod_interface_network_rx_bytes) **Note** pod_interface_network_rx_bytes 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_network_tx_bytes	PodName, 네임스페이스, ClusterName 네임스페이스, ClusterName 서비스, 네임스페이스, ClusterName ClusterName	ClusterName , Namespace , PodName, FullPodName	Pod에서 네트워크를 통해 전송 중인 초당 바이트 수입니다. 공식: sum(pod_interface_network_tx_bytes) **Note** pod_interface_network_tx_bytes 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`pod_cpu_request`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			포드에 대한 CPU 요청입니다.
			
			공식: `sum(container_cpu_request)`
			
			Note
			`pod_cpu_request` 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_memory_request 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드에 대한 메모리 요청량입니다. 공식: sum(container_memory_request) **Note** pod_memory_request 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`pod_cpu_limit`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			포드의 컨테이너에 대해 정의된 CPU 한도입니다. 포드의 컨테이너에 CPU 제한이 정의되지 않은 경우 이 지표는 표시되지 않습니다.
			
			공식: `sum(container_cpu_limit)`
			
			Note
			`pod_cpu_limit` 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`pod_memory_limit`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			포드의 컨테이너에 대해 정의된 메모리 한도입니다. 포드의 컨테이너에 메모리 제한이 정의되지 않은 경우 이 지표는 표시되지 않습니다.
			
			공식: `sum(container_memory_limit)`
			
			Note
			`pod_cpu_limit` 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_status_failed 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드의 모든 컨테이너가 종료되었으며, 하나 이상의 컨테이너가 0이 아닌 상태로 종료되었거나 시스템에 의해 종료되었음을 나타냅니다.
pod_status_ready 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드의 모든 컨테이너가 Container Ready 조건에 도달하여 준비가 완료되었음을 나타냅니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_status_running 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드의 모든 컨테이너가 실행 중임을 나타냅니다.
pod_status_scheduled 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드가 노드에 예약되었음을 나타냅니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_status_unknown 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드의 상태를 가져올 수 없음을 나타냅니다.
pod_status_pending 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	클러스터에서 포드를 수락했지만 하나 이상의 컨테이너가 아직 준비되지 않았음을 나타냅니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_status_succeeded 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드의 모든 컨테이너가 성공적으로 종료되었으며 다시 시작되지 않음을 나타냅니다.
pod_number_of_containers 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드 사양에 정의된 컨테이너 수를 보고합니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_number_of_running_containers 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	현재 Running 상태에 있는 포드 내 컨테이너 수를 보고합니다.
pod_container_status_terminated 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드에서 Terminated 상태에 있는 컨테이너 수를 보고합니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_container_status_running 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드에서 Running 상태에 있는 컨테이너 수를 보고합니다.
pod_container_status_waiting 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드에서 Waiting 상태에 있는 컨테이너 수를 보고합니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
pod_interface_network_rx_dropped 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	포드의 네트워크 인터페이스에서 수신된 후 삭제된 패킷 수입니다.
pod_interface_network_tx_dropped 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	전송될 예정이었으나 포드에서 삭제된 패킷 수입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`container_cpu_utilization`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			컨테이너에서 사용 중인 CPU 단위의 비율입니다.
			
			공식: `container_cpu_usage_total / node_cpu_limit`
			
			Note
			`container_cpu_utilization` 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`container_cpu_utilization_over_container_limit`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			컨테이너 제한을 기준으로 컨테이너에서 사용 중인 CPU 단위의 비율입니다. 컨테이너에 CPU 제한이 정의되지 않은 경우 이 지표는 표시되지 않습니다.
			
			공식: `container_cpu_usage_total / container_cpu_limit`
			
			Note
			`container_cpu_utilization_over_container_limit` 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
			관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`container_memory_utilization`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			컨테이너에서 사용 중인 메모리 단위의 비율입니다.
			
			공식: `container_memory_working_set / node_memory_limit`
			
			Note
			`container_memory_utilization` 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드 단원을 참조하십시오.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`container_memory_utilization_over_container_limit`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			컨테이너 제한을 기준으로 컨테이너에서 사용 중인 메모리 단위의 비율입니다. 컨테이너에 메모리 제한이 정의되지 않은 경우 이 지표는 표시되지 않습니다.
			
			공식: `container_memory_working_set / container_memory_limit`
			
			Note
			`container_memory_utilization_over_container_limit` 는 지표로 직접 보고되지는 않지만 성능 로그 이벤트의 필드입니다. 자세한 내용은 Amazon EKS 및 Kubernetes에 대한 성능

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
			로그 이벤트의 관련 필드 단원을 참조하십시오.
container_memory_failures_total 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다. Windows에서는 제공되지 않습니다.		ClusterName PodName, Namespace , ClusterName , ContainerName PodName, Namespace , ClusterName , ContainerName , FullPodName	컨테이너에서 발생한 메모리 할당 실패 횟수입니다.
pod_number_of_container_restarts	PodName, Namespace , ClusterName		Pod의 컨테이너 재시작 총 횟수입니다.
service_number_of_running_pods	서비스, Namespace , ClusterName ClusterName		클러스터에서 단일 또는 복수의 서비스를 실행하는 Pod의 숫자입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`replicas_desired`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
			워크로드 사양에 정의된 워크로드에 필요한 포드 수입니다.

`replicas_ready`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
			준비 상태에 도달한 워크로드의 포드 수입니다.

`status_replicas_available`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
			워크로드에 사용할 수 있는 포드 수입니다. 워크로드 사양에 정의된 대로 `minReadySeconds` 준비가 되면 포드를 사용할 수 있습니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`status_replicas_unavailable`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
			사용할 수 없는 워크로드의 포드 수입니다. 워크로드 사양에 정의된 대로 `minReadySeconds` 준비가 되면 포드를 사용할 수 있습니다. 이 기준을 충족하지 않으면 포드를 사용할 수 없습니다.

`apiserver_storage_objects`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName` , `resource`
			마지막 확인 당시 etcd에 저장된 객체 수입니다.

`apiserver_request_total`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName` , `code`, `verb`
			Kubernetes API 서버에 대한 총 API 요청 수입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`apiserver_request_duration_seconds`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName` , `verb`
			Kubernetes API 서버에 대한 API 요청의 응답 지연 시간입니다.

`apiserver_admission_controller_admission_duration_seconds`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName` , `operation`
			승인 컨트롤러 지연 시간(초)입니다. 승인 컨트롤러는 Kubernetes API 서버에 대한 요청을 가로채는 코드입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`rest_client_request_duration_seconds`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName`, `operation`
			Kubernetes API 서버를 호출하는 클라이언트가 경험한 응답 지연 시간입니다. 이 지표는 실험용이며 Kubernetes의 향후 릴리스에서 변경될 수 있습니다.

`rest_client_requests_total`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName`, `code`, `method`
			클라이언트가 Kubernetes API 서버에 요청한 총 API 요청 수입니다. 이 지표는 실험용이며 Kubernetes의 향후 릴리스에서 변경될 수 있습니다.

`etcd_request_duration_seconds`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName`, `operation`
			Etcd에 대한 API 호출의 응답 지연 시간입니다. 이 지표는 실험용이며 Kubernetes의 향후 릴리스에서 변경될 수 있습니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`apiserver_storage_size_bytes`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName`, `endpoint`
			물리적으로 할당된 스토리지 데이터베이스 파일의 크기(바이트)입니다. 이 지표는 실험용이며 Kubernetes의 향후 릴리스에서 변경될 수 있습니다.

`apiserver_longrunning_requests`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName`, `resource`
			Kubernetes API 서버에 대한 활성 장기 실행 요청 수입니다.

`apiserver_current_inflight_requests`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName`, `request_kind`
			Kubernetes API 서버에서 처리 중인 요청 수입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`apiserver_admission_webhook_admission_duration_seconds`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName , name`
			승인 웹후크 지연 시간(초)입니다. 승인 웹후크는 승인 요청을 수신하고 이를 이용해 무언가를 수행하는 HTTP 콜백입니다.

`apiserver_admission_step_admission_duration_seconds`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName , operation`
			승인 하위 단계 지연 시간(초)입니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명
apiserver_requested_deprecated_apis 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , group	Kubernetes API 서버에서 더 이상 사용되지 않는 API에 대한 요청 수입니다.
apiserver_request_total_5XX 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , code, verb	Kubernetes API 서버에 대한 요청 중 5XX HTTP 응답 코드로 응답한 요청 수입니다.
apiserver_storage_list_duration_seconds 이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다		ClusterName ClusterName , resource	Etcd의 객체를 나열하는 응답 지연 시간입니다. 이 지표는 실험용이며 Kubernetes의 향후 릴리스에서 변경될 수 있습니다.

지표 이름	모든 버전의 Container Insights를 사용한 측정 기준	Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights의 추가 측정 기준	설명

`apiserver_current_inqueue_requests`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName , request_kind`
			Kubernetes API 서버에서 대기열에 있는 요청 수입니다. 이 지표는 실험용이며 Kubernetes의 향후 릴리스에서 변경될 수 있습니다.

`apiserver_flowcontrol_rejected_requests_total`
 	
이 지표는 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 통해서만 제공됩니다
		`ClusterName`
		
		`ClusterName , reason`
			API 우선순위 및 공정성 하위 시스템에서 거부한 요청 수입니다. 이 지표는 실험용이며 Kubernetes의 향후 릴리스에서 변경될 수 있습니다.

NVIDIA GPU 지표

Amazon EKS의 향상된 관찰성을 사용하여 CloudWatch 에이전트 버전 `1.300034.0`부터 Container Insights는 기본적으로 EKS 워크로드에서 NVIDIA GPU 지표를 수집합니다. CloudWatch 에이전트를 설치할 때는 CloudWatch Observability EKS 추가 기능 버전 `v1.3.0-eksbuild.1` 이상을 사용해야 합니다. 자세한 내용은 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 CloudWatch 에이전트 설치 단원을 참조하십시오. 이렇게 수집된 NVIDIA GPU 지표는 이 섹션의 표에 나열되어 있습니다.

Container Insights로 NVIDIA GPU 지표를 수집하려면 다음 사전 요구 사항을 충족해야 합니다.

- Amazon CloudWatch Observability EKS 추가 기능 버전 `v1.3.0-eksbuild.1` 이상을 사용하여 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 사용해야 합니다.
- 쿠버네티스용 NVIDIA 디바이스 플러그인을 클러스터에 설치해야 합니다.
- NVIDIA 컨테이너 툴킷은 클러스터의 노드에 설치해야 합니다. 예를 들어 Amazon EKS 최적화된 가속화 AMI는 필수 구성 요소로 구축됩니다.

CloudWatch 에이전트 구성 파일 처음의 `accelerated_compute_metrics` 옵션을 `false`로 설정하여 NVIDIA GPU 지표 수집을 옵트아웃할 수 있습니다. 자세한 내용과 옵트아웃 예시는 (선택 사항) 추가 구성 단원을 참조하십시오.

지표 이름	측정기준	설명

`container_gpu_memory_total`	`ClusterName`
	
	`ClusterName` , `Namespace` , `PodName`, `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName` , `GpuDevice`
		컨테이너에 할당된 GPU의 총 프레임 버퍼 바이트 규모.

`container_gpu_memory_used`	`ClusterName`
	
	`ClusterName` , `Namespace` , `PodName`, `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName` , `GpuDevice`
		컨테이너에 할당된 GPU에서 사용된 프레임 버퍼의 바이트.

지표 이름	측정기준	설명
container_gpu_memory_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	컨테이너에 할당된 GPU의 프레임 버퍼 사용률.
container_gpu_power_draw	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	컨테이너에 할당된 GPU의 전력 와트 사용량.

지표 이름	측정기준	설명
container_gpu_temperature	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	컨테이너에 할당된 GPU의 섭씨 온도.
container_gpu_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	컨테이너에 할당된 GPU의 활용률.
node_gpu_memory_total	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	노드에 할당된 GPU의 총 프레임 버퍼 바이트 규모.

지표 이름	측정기준	설명
node_gpu_memory_used	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	노드에 할당된 GPU에서 사용된 프레임 버퍼의 바이트.
node_gpu_memory_utilization	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	노드에 할당된 GPU의 프레임 버퍼 사용률.
node_gpu_power_draw	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	노드에 할당된 GPU의 전력 와트 사용량.
node_gpu_temperature	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	노드에 할당된 GPU의 섭씨 온도.

지표 이름	측정기준	설명
node_gpu_utilization	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	노드에 할당된 GPU의 활용률.
pod_gpu_memory_total	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	포드에 할당된 GPU의 총 프레임 버퍼 바이트 규모.

지표 이름	측정기준	설명
pod_gpu_memory_used	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	포드에 할당된 GPU에서 사용된 프레임 버퍼의 바이트.
pod_gpu_memory_utilization	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	포드에 할당된 GPU의 프레임 버퍼 사용률.

지표 이름	측정기준	설명
pod_gpu_power_draw	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	포드에 할당된 GPU의 전력 와트 사용량.
pod_gpu_temperature	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	포드에 할당된 GPU의 섭씨 온도.

지표 이름	측정기준	설명
pod_gpu_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	포드에 할당된 GPU의 활용률.

AWS Trainium 및 AWS Inferentia의 AWS Neuron 지표

CloudWatch 에이전트의 1.300036.0 버전부터 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights는 기본적으로 AWS Trainium 및 AWS Inferentia 액셀러레이터로부터 가속화된 컴퓨팅 지표를 수집합니다. CloudWatch 에이전트를 설치할 때는 CloudWatch Observability EKS 추가 기능 버전 v1.5.0-eksbuild.1 이상을 사용해야 합니다. 추가 기능에 대한 자세한 내용은 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 CloudWatch 에이전트 설치 섹션을 참조하세요. AWS Trainium에 대한 자세한 내용은 AWS Trainium을 참조하세요. AWS Inferentia에 대한 자세한 내용은 AWS Inferentia를 참조하세요.

Container Insights로 AWS Neuron 지표를 수집하려면 다음 사전 조건을 충족해야 합니다.

- Amazon CloudWatch Observability EKS 추가 기능 버전 v1.5.0-eksbuild.1 이상을 사용하여 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 사용해야 합니다.
- Neuron 드라이버는 클러스터의 노드에 설치해야 합니다.
- Neuron 디바이스 플러그인은 클러스터에 설치해야 합니다. 예를 들어 Amazon EKS 최적화된 가속화 AMI는 필수 구성 요소로 구축됩니다.

이렇게 수집된 지표는 이 섹션의 표에 나열되어 있습니다. 지표는 AWS Trainium, AWS Inferentia, AWS Inferentia2에 대해 수집됩니다.

CloudWatch 에이전트는 Neuron 모니터에서 이러한 지표를 수집하고 필요한 Kubernetes 리소스 상관관계를 수행하여 포드 및 컨테이너 수준에서 지표를 제공합니다.

지표 이름	측정기준	설명
container_neuroncore_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	컨테이너에 할당된 NeuronCore의 캡처된 기간 동안의 NeuronCore 사용률. 단위: 백분율
container_neuroncore_memory_usage_constants	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	NeuronCore의 교육 도중 컨테이너에 할당된 상수(또는 추론 중 가중치)에 사용되는 디바이스 메모리의 양입니다. 단위: 바이트
container_neuroncore_memory_usage_model_code	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName	컨테이너에 할당된 NeuronCore가 모델의 실행 코드에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트

지표 이름	측정기준	설명
	ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	
container_neuroncore_memory_usage_model_shared_scratchpad	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	컨테이너에 할당된 NeuronCore가 모델의 공유되는 스크래치패드에 사용하는 디바이스 메모리의 양입니다. 이 메모리 영역은 모델용입니다. 단위: 바이트
container_neuroncore_memory_usage_runtime_memory	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	컨테이너에 할당된 NeuronCore에서 Neuron 런타임에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트

지표 이름	측정기준	설명
container_neuroncore_memory_usage_tensors	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	컨테이너에 할당된 NeuronCore에서 텐서에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트
container_neuroncore_memory_usage_total	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	컨테이너에 할당된 NeuronCore에서 사용하는 총 메모리의 양입니다. 단위: 바이트

지표 이름	측정기준	설명
container_neurondevice_hw_ecc_events_total	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice	노드에 있는 Neuron 디바이스의 온칩 SRAM 및 디바이스 메모리에 대해 수정 및 수정되지 않은 ECC 이벤트의 수입니다. 단위: 수
pod_neuroncore_utilization	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	포드에 할당된 NeuronCore의 캡처된 기간 동안의 NeuronCore 사용률. 단위: 백분율

지표 이름	측정기준	설명
pod_neuroncore_memory_usage_constants	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	NeuronCore의 교육 도중 포드에 할당된 상수(또는 추론 중 가중치)에 사용되는 디바이스 메모리의 양입니다. 단위: 바이트
pod_neuroncore_memory_usage_model_code	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	포드에 할당된 NeuronCore가 모델의 실행 코드에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트

지표 이름	측정기준	설명
pod_neuroncore_memory_usage_model_shared_scratchpad	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	포드에 할당된 NeuronCore가 모델의 공유되는 스크래치패드에 사용하는 디바이스 메모리의 양입니다. 이 메모리 영역은 모델용입니다. 단위: 바이트
pod_neuroncore_memory_usage_runtime_memory	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	포드에 할당된 NeuronCore에서 Neuron 런타임에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트

지표 이름	측정기준	설명
pod_neuroncore_memory_usage_tensors	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	포드에 할당된 NeuronCore에서 텐서에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트
pod_neuroncore_memory_usage_total	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	포드에 할당된 NeuronCore에서 사용하는 총 메모리의 양입니다. 단위: 바이트

지표 이름	측정기준	설명
pod_neurondevice_hw_ecc_events_total	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice	포드에 할당된 Neuron 디바이스의 온칩 SRAM 및 디바이스 메모리에 대해 수정 및 수정되지 않은 ECC 이벤트의 수입니다. 단위: 바이트
node_neuroncore_utilization	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	노드에 할당된 NeuronCore의 캡처된 기간 동안의 NeuronCore 사용률. 단위: 백분율
node_neuroncore_memory_usage_constants	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	NeuronCore의 교육 도중 노드에 할당된 상수(또는 추론 중 가중치)에 사용되는 디바이스 메모리의 양입니다. 단위: 바이트

지표 이름	측정기준	설명
node_neuroncore_memory_usage_model_code	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	노드에 할당된 NeuronCore가 모델의 실행 코드에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트
node_neuroncore_memory_usage_model_shared_scratchpad	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	노드에 할당된 NeuronCore가 모델의 공유되는 스크래치패드에 사용하는 디바이스 메모리의 양입니다. 이 메모리 영역은 모델용입니다. 단위: 바이트
node_neuroncore_memory_usage_runtime_memory	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	노드에 할당된 NeuronCore에서 Neuron 런타임에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트
node_neuroncore_memory_usage_tensors	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	노드에 할당된 NeuronCore에서 텐서에 사용하는 디바이스 메모리의 양입니다. 단위: 바이트

지표 이름	측정기준	설명
node_neuroncore_memory_usage_total	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	노드에 할당된 NeuronCore에서 사용하는 총 메모리의 양입니다. 단위: 바이트
node_neuron_execution_errors_total	ClusterName ClusterName , InstanceId , NodeName	노드의 총 실행 오류 수입니다. 이 값은 CloudWatch 에이전트에서 generic, numerical , transient , model, runtime, hardware 유형의 오류를 집계하여 계산됩니다. 단위: 수
node_neurondevice_runtime_memory_used_bytes	ClusterName ClusterName , InstanceId , NodeName	노드의 Neuron 디바이스 메모리 사용량(바이트)의 총합입니다. 단위: 바이트
node_neuron_execution_latency	ClusterName ClusterName , InstanceId , NodeName	Neuron 런타임으로 측정한 노드에서의 실행 지연 시간(초)입니다. 단위: 초
node_neurondevice_hw_ecc_events_total	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , NodeName, NeuronDevice	노드에 있는 Neuron 디바이스의 온칩 SRAM 및 디바이스 메모리에 대해 수정 및 수정되지 않은 ECC 이벤트의 수입니다. 단위: 수

AWS Elastic Fabric Adapter(EFA) 지표

CloudWatch 에이전트 `1.300037.0` 버전부터 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights 는 Linux 인스턴스의 Amazon EKS 클러스터로부터 AWS Elastic Fabric Adapter(EFA) 지표를 수집합니다. CloudWatch 에이전트를 설치할 때는 CloudWatch Observability EKS 추가 기능 버전 `v1.5.2-eksbuild.1` 이상을 사용해야 합니다. 추가 기능에 대한 자세한 내용은 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 CloudWatch 에이전트 설치 섹션을 참조하세요. AWS Elastic Fabric Adapter에 대한 자세한 내용은 Elastic Fabric Adapter를 참조하세요.

Container Insights로 AWS Elastic Fabric Adapter 지표를 수집하려면 다음 사전 조건을 충족해야 합니다.

- Amazon CloudWatch Observability EKS 추가 기능 버전 `v1.5.2-eksbuild.1` 이상을 사용하여 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 사용해야 합니다.
- EFA 디바이스 플러그인은 클러스터에 설치해야 합니다. 자세한 내용은 GitHub의 aws-efa-k8s-device-plugin을 참조하세요.

수집된 지표 목록은 다음 표에 나와 있습니다.

지표 이름	측정기준	설명

`container_efa_rx_bytes`	`ClusterName`
	
	`ClusterName` , `Namespace` , `PodName`, `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName` , `EfaDevice`
		컨테이너에 할당된 EFA 디바이스에서 수신하는 초당 바이트 수입니다.
		
		단위: 바이트/초

`container_efa_tx_bytes`	`ClusterName`
	
	`ClusterName` , `Namespace` , `PodName`, `ContainerName`
		컨테이너에 할당된 EFA 디바이스에서 송신하는 초당 바이트 수입니다.
		
		단위: 바이트/초

지표 이름	측정기준	설명
	ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	
container_efa_rx_dropped	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	컨테이너에 할당된 EFA 디바이스에서 수신 후 삭제된 패킷 수입니다. 단위: 개수/초
container_efa_rdma_read_bytes	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	컨테이너에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 읽기 작업을 사용하여 수신하는 초당 바이트 수입니다. 단위: 바이트/초

지표 이름	측정기준	설명
container_efa_rdma_write_bytes	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	컨테이너에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 읽기 작업을 사용하여 송신하는 초당 바이트 수입니다. 단위: 바이트/초
container_efa_rdma_write_recv_bytes	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	컨테이너에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 쓰기 작업 도중 수신하는 초당 바이트 수입니다. 단위: 바이트/초

지표 이름	측정기준	설명
pod_efa_rx_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	포드에 할당된 EFA 디바이스에서 수신하는 초당 바이트 수입니다. 단위: 바이트/초
pod_efa_tx_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	포드에 할당된 EFA 디바이스에서 송신하는 초당 바이트 수입니다. 단위: 바이트/초

지표 이름	측정기준	설명
pod_efa_rx_dropped	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	포드에 할당된 EFA 디바이스에서 수신 후 삭제된 패킷 수입니다. 단위: 개수/초
pod_efa_rdma_read_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	포드에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 읽기 작업을 사용하여 수신하는 초당 바이트 수입니다. 단위: 바이트/초

지표 이름	측정기준	설명
pod_efa_rdma_write_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	포드에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 읽기 작업을 사용하여 송신하는 초당 바이트 수입니다. 단위: 바이트/초
pod_efa_rdma_write_recv_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	포드에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 쓰기 작업 도중 수신하는 초당 바이트 수입니다. 단위: 바이트/초

지표 이름	측정기준	설명
node_efa_rx_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	노드에 할당된 EFA 디바이스에서 수신하는 초당 바이트 수입니다. 단위: 바이트/초
node_efa_tx_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	노드에 할당된 EFA 디바이스에서 송신하는 초당 바이트 수입니다. 단위: 바이트/초
node_efa_rx_dropped	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	노드에 할당된 EFA 디바이스에서 수신 후 삭제된 패킷 수입니다. 단위: 개수/초
node_efa_rdma_read_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	노드에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 읽기 작업을 사용하여 수신하는 초당 바이트 수입니다. 단위: 바이트/초

지표 이름	측정기준	설명
pod_efa_rdma_write_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	포드에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 읽기 작업을 사용하여 송신하는 초당 바이트 수입니다. 단위: 바이트/초
node_efa_rdma_write_recv_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	노드에 할당된 EFA 디바이스에서 원격 직접 메모리 액세스 쓰기 작업 도중 수신하는 초당 바이트 수입니다. 단위: 바이트/초

Container Insights 성능 로그 참조

이 단원에는 Container Insights가 성능 로그 이벤트를 사용하여 지표를 수집하는 방법에 관한 참조 정보가 포함되어 있습니다. 컨테이너 인사이트를 배포하면 성능 로그 이벤트에 대한 로그 그룹이 자동으로 생성됩니다. 이 로그 그룹을 직접 생성할 필요는 없습니다.

주제

- Amazon ECS의 Container Insights 성능 로그 이벤트
- Amazon EKS 및 Kubernetes의 Container Insights 성능 로그 이벤트
- Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드

Amazon ECS의 Container Insights 성능 로그 이벤트

다음은 Container Insights가 Amazon ECS에서 수집하는 성능 로그 이벤트의 예입니다.

이러한 로그는 CloudWatch Logs의 `/aws/ecs/containerinsights/`*`CLUSTER_NAME`*`/performance`라는 로그 그룹에 있습니다. 해당 로그 그룹 내에서 각 컨테이너 인스턴스에는 `AgentTelemetry-`*`CONTAINER_INSTANCE_ID`*라는 로그 스트림이 있습니다.

`{ $.Type = "Container" }`와 같은 쿼리를 사용하여 이러한 로그를 쿼리하면 모든 컨테이너 로그 이벤트를 볼 수 있습니다.

유형: 컨테이너

```
{
 "Version":"0",
 "Type":"Container",
 "ContainerName":"sleep",
 "TaskId":"7ac4dfba69214411b4783a3b8189c9ba",
 "TaskDefinitionFamily":"sleep360",
 "TaskDefinitionRevision":"1",
 "ContainerInstanceId":"0d7650e6dec34c1a9200f72098071e8f",
 "EC2InstanceId":"i-0c470579dbcdbd2f3",
 "ClusterName":"MyCluster",
 "Image":"busybox",
 "ContainerKnownStatus":"RUNNING",
 "Timestamp":1623963900000,
 "CpuUtilized":0.0,
 "CpuReserved":10.0,
 "MemoryUtilized":0,
 "MemoryReserved":10,
 "StorageReadBytes":0,
 "StorageWriteBytes":0,
 "NetworkRxBytes":0,
 "NetworkRxDropped":0,
 "NetworkRxErrors":0,
 "NetworkRxPackets":14,
 "NetworkTxBytes":0,
 "NetworkTxDropped":0,
 "NetworkTxErrors":0,
 "NetworkTxPackets":0
}
```

유형: 작업

```
{
    "Version": "0",
    "Type": "Task",
    "TaskId": "7ac4dfba69214411b4783a3b8189c9ba",
    "TaskDefinitionFamily": "sleep360",
    "TaskDefinitionRevision": "1",
    "ContainerInstanceId": "0d7650e6dec34c1a9200f72098071e8f",
```

```
    "EC2InstanceId": "i-0c470579dbcdbd2f3",
    "ClusterName": "MyCluster",
    "AccountID": "637146863587",
    "Region": "us-west-2",
    "AvailabilityZone": "us-west-2b",
    "KnownStatus": "RUNNING",
    "LaunchType": "EC2",
    "PullStartedAt": 1623963608201,
    "PullStoppedAt": 1623963610065,
    "CreatedAt": 1623963607094,
    "StartedAt": 1623963610382,
    "Timestamp": 1623963900000,
    "CpuUtilized": 0.0,
    "CpuReserved": 10.0,
    "MemoryUtilized": 0,
    "MemoryReserved": 10,
    "StorageReadBytes": 0,
    "StorageWriteBytes": 0,
    "NetworkRxBytes": 0,
    "NetworkRxDropped": 0,
    "NetworkRxErrors": 0,
    "NetworkRxPackets": 14,
    "NetworkTxBytes": 0,
    "NetworkTxDropped": 0,
    "NetworkTxErrors": 0,
    "NetworkTxPackets": 0,
    "EBSFilesystemUtilized": 10,
    "EBSFilesystemSize": 20,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "CpuUtilized",
                    "Unit": "None"
                },
                {
                    "Name": "CpuReserved",
                    "Unit": "None"
                },
                {
                    "Name": "MemoryUtilized",
                    "Unit": "Megabytes"
                },
```

```
                {
                    "Name": "MemoryReserved",
                    "Unit": "Megabytes"
                },
                {
                    "Name": "StorageReadBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "StorageWriteBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "NetworkRxBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "NetworkTxBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "EBSFilesystemSize",
                    "Unit": "Gigabytes"
                },
                {
                    "Name": "EBSFilesystemUtilzed",
                    "Unit": "Gigabytes"
                }
            ],
            "Dimensions": [
                ["ClusterName"],
                [
                    "ClusterName",
                    "TaskDefinitionFamily"
                ]
            ]
        }
    ]
}
```

유형: 서비스

```
{
```

```
    "Version": "0",
    "Type": "Service",
    "ServiceName": "myCIService",
    "ClusterName": "myCICluster",
    "Timestamp": 1561586460000,
    "DesiredTaskCount": 2,
    "RunningTaskCount": 2,
    "PendingTaskCount": 0,
    "DeploymentCount": 1,
    "TaskSetCount": 0,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "DesiredTaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "RunningTaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "PendingTaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "DeploymentCount",
                    "Unit": "Count"
                },
                {
                    "Name": "TaskSetCount",
                    "Unit": "Count"
                }
            ],
            "Dimensions": [
                [
                    "ServiceName",
                    "ClusterName"
                ]
            ]
        }
    ]
```

```
}
```

유형: 볼륨

```
{
    "Version": "0",
    "Type": "Volume",
    "TaskDefinitionFamily": "myCITaskDef",
    "TaskId": "7ac4dfba69214411b4783a3b8189c9ba",
    "ClusterName": "myCICluster",
    "ServiceName": "myCIService",
    "VolumeId": "vol-1233436545ff708cb",
    "InstanceId": "i-0c470579dbcdbd2f3",
    "LaunchType": "EC2",
    "VolumeName": "MyVolumeName",
    "EBSFilesystemUtilized": 10,
    "EBSFilesystemSize": 20,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "EBSFilesystemSize",
                    "Unit": "Gigabytes"
                },
                {
                    "Name": "EBSFilesystemUtilzed",
                    "Unit": "Gigabytes"
                }
            ],
            "Dimensions": [
                ["ClusterName"],
                [
                    "VolumeName",
                    "TaskDefinitionFamily",
                    "ClusterName"
                ],
                [
                    "ServiceName",
                    "ClusterName"
                ]
            ]
        }
```

```
    ]
}
```

유형: 클러스터

```
{
    "Version": "0",
    "Type": "Cluster",
    "ClusterName": "myCICluster",
    "Timestamp": 1561587300000,
    "TaskCount": 5,
    "ContainerInstanceCount": 5,
    "ServiceCount": 2,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "TaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "ContainerInstanceCount",
                    "Unit": "Count"
                },
                {
                    "Name": "ServiceCount",
                    "Unit": "Count"
                }
            ],
            "Dimensions": [
                [
                    "ClusterName"
                ]
            ]
        }
    ]
}
```

Amazon EKS 및 Kubernetes의 Container Insights 성능 로그 이벤트

다음은 Container Insights가 Amazon EKS 및 Kubernetes 클러스터에서 수집하는 성능 로그 이벤트의 예입니다.

유형: 노드

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "node_network_total_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "node_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_reserved_capacity"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_pods"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_containers"
        }
      ],
      "Dimensions": [
        [
          "NodeName",
          "InstanceId",
          "ClusterName"
        ]
      ],
```

```
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "node_network_total_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "node_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_reserved_capacity"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_pods"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_containers"
        },
        {
          "Name": "node_cpu_usage_total"
        },
        {
          "Name": "node_cpu_limit"
        },
        {
          "Unit": "Bytes",
          "Name": "node_memory_working_set"
        },
        {
          "Unit": "Bytes",
```

```
          "Name": "node_memory_limit"
        }
      ],
      "Dimensions": [
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "/proc",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567096682364",
  "Type": "Node",
  "Version": "0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_cpu_limit": 4000,
  "node_cpu_request": 1130,
  "node_cpu_reserved_capacity": 28.249999999999996,
  "node_cpu_usage_system": 33.794636630852764,
  "node_cpu_usage_total": 136.47852169244098,
  "node_cpu_usage_user": 71.67075111567326,
  "node_cpu_utilization": 3.4119630423110245,
  "node_memory_cache": 3103297536,
  "node_memory_failcnt": 0,
  "node_memory_hierarchical_pgfault": 0,
  "node_memory_hierarchical_pgmajfault": 0,
  "node_memory_limit": 16624865280,
  "node_memory_mapped_file": 406646784,
  "node_memory_max_usage": 4230746112,
  "node_memory_pgfault": 0,
  "node_memory_pgmajfault": 0,
  "node_memory_request": 1115684864,
```

```
  "node_memory_reserved_capacity": 6.7109407818311055,
  "node_memory_rss": 798146560,
  "node_memory_swap": 0,
  "node_memory_usage": 3901444096,
  "node_memory_utilization": 6.601302600149552,
  "node_memory_working_set": 1097457664,
  "node_network_rx_bytes": 35918.392817386324,
  "node_network_rx_dropped": 0,
  "node_network_rx_errors": 0,
  "node_network_rx_packets": 157.67565245448117,
  "node_network_total_bytes": 68264.20276554905,
  "node_network_tx_bytes": 32345.80994816272,
  "node_network_tx_dropped": 0,
  "node_network_tx_errors": 0,
  "node_network_tx_packets": 154.21455923431654,
  "node_number_of_running_containers": 16,
  "node_number_of_running_pods": 13
}
```


유형: NodeFS


```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_filesystem_utilization"
        }
      ],
      "Dimensions": [
        [
          "NodeName",
          "InstanceId",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
```

```
    }
  ],
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097939726",
  "Type": "NodeFS",
  "Version": "0",
  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_filesystem_available": 17298395136,
  "node_filesystem_capacity": 21462233088,
  "node_filesystem_inodes": 10484720,
  "node_filesystem_inodes_free": 10367158,
  "node_filesystem_usage": 4163837952,
  "node_filesystem_utilization": 19.400767547940255
}
```

유형: NodeDiskIO

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeDiskIO",
  "Version": "0",
```

```
  "device": "/dev/nvme0n1",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_diskio_io_service_bytes_async": 9750.505814277016,
  "node_diskio_io_service_bytes_read": 0,
  "node_diskio_io_service_bytes_sync": 230.6174506688036,
  "node_diskio_io_service_bytes_total": 9981.123264945818,
  "node_diskio_io_service_bytes_write": 9981.123264945818,
  "node_diskio_io_serviced_async": 1.153087253344018,
  "node_diskio_io_serviced_read": 0,
  "node_diskio_io_serviced_sync": 0.03603397666700056,
  "node_diskio_io_serviced_total": 1.1891212300110185,
  "node_diskio_io_serviced_write": 1.1891212300110185
}
```


유형: NodeNet


```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeNet",
  "Version": "0",
  "interface": "eni972f6bfa9a0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_interface_network_rx_bytes": 3163.008420864309,
  "node_interface_network_rx_dropped": 0,
  "node_interface_network_rx_errors": 0,
  "node_interface_network_rx_packets": 16.575629266820258,
  "node_interface_network_total_bytes": 3518.3935157426017,
  "node_interface_network_tx_bytes": 355.385094878293,
  "node_interface_network_tx_dropped": 0,
```

```
  "node_interface_network_tx_errors": 0,
  "node_interface_network_tx_packets": 3.9997714100370625
}
```

유형: Pod

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_rx_bytes"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_tx_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization_over_pod_limit"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization_over_pod_limit"
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ],
```

```
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_reserved_capacity"
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "pod_number_of_container_restarts"
        }
```

```
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567097351092",
  "Type": "Pod",
  "Version": "0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "pod_cpu_limit": 200,
```

```
  "pod_cpu_request": 200,
  "pod_cpu_reserved_capacity": 5,
  "pod_cpu_usage_system": 1.4504841104992765,
  "pod_cpu_usage_total": 5.817016867430125,
  "pod_cpu_usage_user": 1.1281543081661038,
  "pod_cpu_utilization": 0.14542542168575312,
  "pod_cpu_utilization_over_pod_limit": 2.9085084337150624,
  "pod_memory_cache": 8192,
  "pod_memory_failcnt": 0,
  "pod_memory_hierarchical_pgfault": 0,
  "pod_memory_hierarchical_pgmajfault": 0,
  "pod_memory_limit": 104857600,
  "pod_memory_mapped_file": 0,
  "pod_memory_max_usage": 25268224,
  "pod_memory_pgfault": 0,
  "pod_memory_pgmajfault": 0,
  "pod_memory_request": 104857600,
  "pod_memory_reserved_capacity": 0.6307275170893897,
  "pod_memory_rss": 22777856,
  "pod_memory_swap": 0,
  "pod_memory_usage": 25141248,
  "pod_memory_utilization": 0.10988455961791709,
  "pod_memory_utilization_over_pod_limit": 17.421875,
  "pod_memory_working_set": 18268160,
  "pod_network_rx_bytes": 9880.697124714186,
  "pod_network_rx_dropped": 0,
  "pod_network_rx_errors": 0,
  "pod_network_rx_packets": 107.80005532263283,
  "pod_network_total_bytes": 10158.829201483635,
  "pod_network_tx_bytes": 278.13207676944796,
  "pod_network_tx_dropped": 0,
  "pod_network_tx_errors": 0,
  "pod_network_tx_packets": 1.146027574644318,
  "pod_number_of_container_restarts": 0,
  "pod_number_of_containers": 1,
  "pod_number_of_running_containers": 1,
  "pod_status": "Running"
}
```


유형: PodNet


```
{
```

```
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097351092",
  "Type": "PodNet",
  "Version": "0",
  "interface": "eth0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "pod_interface_network_rx_bytes": 9880.697124714186,
  "pod_interface_network_rx_dropped": 0,
  "pod_interface_network_rx_errors": 0,
  "pod_interface_network_rx_packets": 107.80005532263283,
  "pod_interface_network_total_bytes": 10158.829201483635,
  "pod_interface_network_tx_bytes": 278.13207676944796,
  "pod_interface_network_tx_dropped": 0,
  "pod_interface_network_tx_errors": 0,
  "pod_interface_network_tx_packets": 1.146027574644318
```

```
}
```

유형: 컨테이너

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-
sample",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567097399912",
  "Type": "Container",
  "Version": "0",
  "container_cpu_limit": 200,
  "container_cpu_request": 200,
  "container_cpu_usage_system": 1.87958283771964,
  "container_cpu_usage_total": 6.159993652997942,
  "container_cpu_usage_user": 1.6707403001952357,
  "container_cpu_utilization": 0.15399984132494854,
  "container_memory_cache": 8192,
  "container_memory_failcnt": 0,
  "container_memory_hierarchical_pgfault": 0,
  "container_memory_hierarchical_pgmajfault": 0,
  "container_memory_limit": 104857600,
  "container_memory_mapped_file": 0,
  "container_memory_max_usage": 24580096,
  "container_memory_pgfault": 0,
  "container_memory_pgmajfault": 0,
  "container_memory_request": 104857600,
  "container_memory_rss": 22736896,
  "container_memory_swap": 0,
  "container_memory_usage": 24453120,
  "container_memory_utilization": 0.10574541028701798,
  "container_memory_working_set": 17580032,
```

```
  "container_status": "Running",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id":
 "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "number_of_container_restarts": 0
}
```

유형: ContainerFS

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
```

```
  "Timestamp": "1567097399912",
  "Type": "ContainerFS",
  "Version": "0",

  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id":
 "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  }
}
```

유형: 클러스터

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "cluster_node_count"
        },
        {
          "Unit": "Count",
          "Name": "cluster_failed_node_count"
```

```
        }
      ],
      "Dimensions": [
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097534160",
  "Type": "Cluster",
  "Version": "0",
  "cluster_failed_node_count": 0,
  "cluster_node_count": 3
}
```

유형: ClusterService

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "service_number_of_running_pods"
        }
      ],
      "Dimensions": [
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
```

```
    }
  ],
  "ClusterName": "myCICluster",
  "Namespace": "amazon-cloudwatch",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097534160",
  "Type": "ClusterService",
  "Version": "0",
  "kubernetes": {
    "namespace_name": "amazon-cloudwatch",
    "service_name": "cloudwatch-agent-statsd"
  },
  "service_number_of_running_pods": 1
}
```

유형: ClusterNamespace

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "namespace_number_of_running_pods"
        }
      ],
      "Dimensions": [
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Namespace": "amazon-cloudwatch",
```

```
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097594160",
  "Type": "ClusterNamespace",
  "Version": "0",
  "kubernetes": {
    "namespace_name": "amazon-cloudwatch"
  },
  "namespace_number_of_running_pods": 7
}
```

Amazon EKS 및 Kubernetes에 대한 성능 로그 이벤트의 관련 필드

Amazon EKS 및 Kubernetes의 경우 컨테이너화된 CloudWatch 에이전트가 데이터를 성능 로그 이벤트로 내보냅니다. 이를 통해 CloudWatch는 카디널리티가 높은 데이터를 수집하고 저장할 수 있습니다. CloudWatch는 성능 로그 이벤트의 데이터를 사용하여 세분화된 세부 정보를 잃지 않고도 클러스터, 노드 및 포드 수준에서 집계된 CloudWatch 지표를 생성합니다.

다음 표에는 이러한 성능 로그 이벤트에서 Container Insights 지표 데이터의 수집과 관련된 필드가 나와 있습니다. CloudWatch Logs Insights를 사용해 이러한 필드 중 어떤 것에 대해서든 쿼리를 수행하여 데이터를 수집하거나 문제를 조사할 수 있습니다. 자세한 내용은 CloudWatch Logs Insights를 사용한 로그 데이터 분석 단원을 참조하세요.

유형	로그 필드	소스	공식 또는 참고
Pod	pod_cpu_utilization	계산 완료	공식: pod_cpu_usage_total / node_cpu_limit
Pod	pod_cpu_usage_total pod_cpu_usage_total 는 밀리코어 단위로 보고됩니다.	cadvisor	
Pod	pod_cpu_limit	계산 완료	공식: sum(conta

유형	로그 필드	소스	공식 또는 참고
			iner_cpu_limit) sum(container_cpu_limit) 에는 이미 완료된 포드가 포함되어 있습니다. Pod의 어떤 컨테이너에도 CPU 한도가 정의되어 있지 않은 경우에는 로그 이벤트에 이 필드가 나타나지 않습니다. 여기에는 init 컨테이너가 포함되어 있습니다.
Pod	pod_cpu_request	계산 완료	공식: sum(container_cpu_request) container_cpu_request 이 설정되어 있지 않을 수도 있습니다. 설정된 것만 합계에 포함됩니다.

유형	로그 필드	소스	공식 또는 참고
Pod	pod_cpu_utilization_over_pod_limit	계산 완료	공식: pod_cpu_usage_total / pod_cpu_limit
Pod	pod_cpu_reserved_capacity	계산 완료	공식: pod_cpu_request / node_cpu_limit
Pod	pod_memory_utilization	계산 완료	공식: pod_memory_working_set / node_memory_limit 노드 메모리 제한에 대한 포드 메모리 사용량의 백분율입니다.
Pod	pod_memory_working_set	cadvisor	

유형	로그 필드	소스	공식 또는 참고
Pod	pod_memory_limit	계산 완료	공식: sum(container_memory_limit) Pod의 어떤 컨테이너에도 메모리 한도가 정의되어 있지 않은 경우에는 로그 이벤트에 이 필드가 나타나지 않습니다. 여기에는 init 컨테이너가 포함되어 있습니다.
Pod	pod_memory_request	계산 완료	공식: sum(container_memory_request) container_memory_request 이 설정되어 있지 않을 수도 있습니다. 설정된 것만 합계에 포함됩니다.

유형	로그 필드	소스	공식 또는 참고
Pod	pod_memory_utilization_over_pod_limit	계산 완료	공식: pod_memory_working_set / pod_memory_limit Pod의 어떤 컨테이너에도 메모리 한도가 정의되어 있지 않은 경우에는 로그 이벤트에 이 필드가 나타나지 않습니다. 여기에는 init 컨테이너가 포함되어 있습니다.
Pod	pod_memory_reserved_capacity	계산 완료	공식: pod_memory_request / node_memory_limit

유형	로그 필드	소스	공식 또는 참고
Pod	pod_network_tx_bytes	계산 완료	공식: sum(pod_interface_network_tx_bytes) 이 데이터는 Pod별로 모든 네트워크 인스턴스에서 사용할 수 있습니다. CloudWatch 에이전트는 합계를 계산하고 지표 추출 규칙을 추가합니다.
Pod	pod_network_rx_bytes	계산 완료	공식: sum(pod_interface_network_rx_bytes)
Pod	pod_network_total_bytes	계산 완료	공식: pod_network_rx_bytes + pod_network_tx_bytes

유형	로그 필드	소스	공식 또는 참고
PodNet	pod_interface_network_rx_bytes	cadvisor	이 데이터는 Pod 네트워크 인터페이스에 대한 초당 네트워크 rx 바이트입니다.
PodNet	pod_interface_network_tx_bytes	cadvisor	이 데이터는 Pod 네트워크 인터페이스에 대한 초당 네트워크 tx 바이트입니다.
컨테이너	container_cpu_usage_total	cadvisor	
컨테이너	container_cpu_limit	cadvisor	설정되어 있지 않을 수도 있습니다. 설정이 되지 않은 경우에는 표시되지 않습니다.
컨테이너	container_cpu_request	cadvisor	설정되어 있지 않을 수도 있습니다. 설정이 되지 않은 경우에는 표시되지 않습니다.
컨테이너	container_memory_working_set	cadvisor	

유형	로그 필드	소스	공식 또는 참고
컨테이너	container_memory_limit	pod	설정되어 있지 않을 수도 있습니다. 설정이 되지 않은 경우에는 표시되지 않습니다.
컨테이너	container_memory_request	pod	설정되어 있지 않을 수도 있습니다. 설정이 되지 않은 경우에는 표시되지 않습니다.
노드	node_cpu_utilization	계산 완료	공식: node_cpu_usage_total / node_cpu_limit
노드	node_cpu_usage_total	cadvisor	
노드	node_cpu_limit	/proc	

유형	로그 필드	소스	공식 또는 참고
노드	node_cpu_request	계산 완료	공식: sum(pod_cpu_request) cronjobs의 경우 node_cpu_request 에는 완성된 포드의 요청이 포함되어 있습니다. 이로 인해 node_cpu_reserved_capacity 의 값이 높아질 수 있습니다.
노드	node_cpu_reserved_capacity	계산 완료	공식: node_cpu_request / node_cpu_limit
노드	node_memory_utilization	계산 완료	공식: node_memory_working_set / node_memory_limit
노드	node_memory_working_set	cadvisor	
노드	node_memory_limit	/proc	

유형	로그 필드	소스	공식 또는 참고
노드	node_memory_request	계산 완료	공식: sum(pod_memory_request)
노드	node_memory_reserved_capacity	계산 완료	공식: node_memory_request / node_memory_limit
노드	node_network_rx_bytes	계산 완료	공식: sum(node_interface_network_rx_bytes)
노드	node_network_tx_bytes	계산 완료	공식: sum(node_interface_network_tx_bytes)
노드	node_network_total_bytes	계산 완료	공식: node_network_rx_bytes + node_network_tx_bytes
노드	node_number_of_running_pods	Pod 목록	
노드	node_number_of_running_containers	Pod 목록	

유형	로그 필드	소스	공식 또는 참고
NodeNet	node_interface_network_rx_bytes	cadvisor	이 데이터는 작업자 노드 네트워크 인터페이스에 대한 초당 네트워크 rx 바이트입니다.
NodeNet	node_interface_network_tx_bytes	cadvisor	이 데이터는 작업자 노드 네트워크 인터페이스에 대한 초당 네트워크 tx 바이트입니다.
NodeFS	node_filesystem_capacity	cadvisor	
NodeFS	node_filesystem_usage	cadvisor	
NodeFS	node_filesystem_utilization	계산 완료	공식: node_filesystem_usage / node_filesystem_capacity 이 데이터는 디스크 이름별로 사용할 수 있습니다.
클러스터	cluster_failed_node_count	API 서버	
클러스터	cluster_node_count	API 서버	

유형	로그 필드	소스	공식 또는 참고
Service	service_number_of_running_pods	API 서버	
Namespace	namespace_number_of_running_pods	API 서버	

지표 계산 예

이 섹션에는 앞의 표에서 몇몇 값들이 어떻게 계산되었는지를 보여주는 예제가 포함되어 있습니다.

클러스터가 다음 상태에 있다고 가정해 보겠습니다.

```
Node1
   node_cpu_limit = 4
   node_cpu_usage_total = 3

   Pod1
     pod_cpu_usage_total = 2

     Container1
        container_cpu_limit = 1
        container_cpu_request = 1
        container_cpu_usage_total = 0.8

     Container2
        container_cpu_limit = null
        container_cpu_request = null
        container_cpu_usage_total = 1.2

   Pod2
     pod_cpu_usage_total = 0.4

     Container3
        container_cpu_limit = 1
        container_cpu_request = 0.5
        container_cpu_usage_total = 0.4

Node2
   node_cpu_limit = 8
   node_cpu_usage_total = 1.5
```

```
Pod3
  pod_cpu_usage_total = 1

  Container4
     container_cpu_limit = 2
     container_cpu_request = 2
     container_cpu_usage_total = 1
```

다음 표에는 이 데이터를 사용하여 Pod CPU 지표를 계산하는 방법이 나와 있습니다.

지표	공식	Pod1	Pod2	Pod3
pod_cpu_utilization	pod_cpu_usage_total / node_cpu_limit	2 / 4 = 50%	0.4 / 4 = 10%	1 / 8 = 12.5%
pod_cpu_utilization_over_pod_limit	pod_cpu_usage_total / sum(container_cpu_limit)	Container 2 에 대한 CPU 한도가 정의되어 있지 않기 때문에 해당 사항 없음	0.4 / 1 = 40%	1 / 2 = 50%
pod_cpu_reserved_capacity	sum(container_cpu_request) / node_cpu_limit	(1 + 0) / 4 = 25%	0.5 / 4 = 12.5%	2 / 8 = 25%

다음 표에는 이 데이터를 사용하여 노드 CPU 지표를 계산하는 방법이 나와 있습니다.

지표	공식	노드 1	노드 2
node_cpu_utilization	node_cpu_usage_total / node_cpu_limit	3 / 4 = 75%	1.5 / 8 = 18.75%
node_cpu_reserved_capacity	sum(pod_cpu_request) / node_cpu_limit	1.5 / 4 = 37.5%	2 / 8 = 25%

Container Insights Prometheus 지표 모니터링

Prometheus에 대한 CloudWatch Container Insights 모니터링은 컨테이너화된 시스템 및 워크로드에서 Prometheus 지표 검색을 자동화합니다. Prometheus는 오픈 소스 시스템 모니터링 및 알림 도구 키트입니다. 자세한 내용은 Prometheus 설명서의 Prometheus란 무엇입니까?를 참조하세요..

Prometheus 지표 검색은 Amazon EC2 인스턴스에서 실행되는 Amazon Elastic Container Service, Amazon Elastic Kubernetes Service 및 Kubernetes 클러스터에 대해 지원됩니다. Prometheus 카운터, 게이지 및 요약 지표 유형이 수집됩니다. 히스토그램 지표에 대한 지원은 향후 릴리스에서 계획되어 있습니다.

Amazon ECS 및 Amazon EKS 클러스터의 경우 EC2 및 Fargate 시작 유형이 모두 지원됩니다. Container Insights는 여러 워크로드에서 지표를 자동으로 수집합니다. 어느 워크로드에서든 지표를 수집하도록 Container Insights를 구성할 수 있습니다.

Prometheus를 오픈 소스 및 개방형 표준 방법으로 채택하여 CloudWatch에서 사용자 지정 지표를 수집할 수 있습니다. Prometheus가 지원되는 CloudWatch 에이전트는 Prometheus 지표를 검색 및 수집하여 애플리케이션 성능 저하 및 장애에 대한 모니터링, 문제 해결 및 경보를 더 빠르게 수행합니다. 이를 통해 관측 기능을 향상시키는 데 필요한 모니터링 도구의 수도 줄어듭니다.

Container Insight Prometheus 지원에는 수집, 저장 및 분석을 포함한 지표 및 로그의 종량제가 포함됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

일부 워크로드에 대해 사전 구축된 대시보드

Container Insights Prometheus 솔루션에는 이 단원에 나열된 인기 있는 워크로드에 대해 사전 구축된 대시보드가 포함되어 있습니다. 이러한 워크로드에 대한 샘플 구성은 (선택 사항) Prometheus 지표 테스트를 위한 컨테이너화된 Amazon ECS 워크로드 샘플 설정 및 (선택 사항) Prometheus 지표 테스트를 위한 컨테이너화된 Amazon EKS 워크로드 샘플 설정 단원을 참조하세요.

또한 에이전트 구성 파일을 편집하여 다른 컨테이너화된 서비스 및 애플리케이션에서 Prometheus 지표를 수집하도록 Container Insights를 구성할 수도 있습니다.

Amazon EC2 인스턴스에서 실행되는 Amazon EKS 클러스터 및 Kubernetes 클러스터용으로 사전 구축된 대시보드가 있는 워크로드:

- AWS App Mesh
- NGINX
- Memcached

- Java/JMX
- HAProxy

Amazon ECS 클러스터용으로 사전 구축된 대시보드가 있는 워크로드:

- AWS App Mesh
- Java/JMX
- NGINX
- NGINX Plus

Amazon ECS 클러스터에서 Prometheus 지표 수집 설정 및 구성

Amazon ECS 클러스터에서 Prometheus 지표를 수집하려면 CloudWatch 에이전트를 수집기로 사용하거나 AWS Distro for OpenTelemetry Collector를 사용하면 됩니다. AWS Distro for OpenTelemetry Collector 사용에 대한 자세한 내용은 https://aws-otel.github.io/docs/getting-started/container-insights/ecs-prometheus를 참조하세요.

다음 단원에서는 CloudWatch 에이전트를 수집기로 사용하여 Prometheus 지표를 검색하는 방법을 설명합니다. Amazon ECS를 실행 중인 클러스터에 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설치하고, 선택적으로 추가 대상을 스크레이프하도록 에이전트를 구성할 수 있습니다. 또한 이 단원에서는 Prometheus 모니터링을 통해 테스트하는 데 사용할 샘플 워크로드를 설정하기 위한 선택적 튜토리얼도 제공합니다.

Amazon ECS의 Container Insights는 Prometheus 지표에 대한 다음 시작 유형 및 네트워크 모드 조합을 지원합니다.

Amazon ECS 시작 유형	지원되는 네트워크 모드
EC2(Linux)	브리지, 호스트, awsvpc
Fargate	awsvpc

VPC 보안 그룹 요구 사항

Prometheus 워크로드의 보안 그룹 수신 규칙은 프라이빗 IP로 Prometheus 지표를 스크레이프하기 위해 CloudWatch 에이전트에 대한 Prometheus 포트를 열어야 합니다.

CloudWatch 에이전트의 보안 그룹 송신 규칙은 CloudWatch 에이전트가 프라이빗 IP로 Prometheus 워크로드의 포트에 연결할 수 있도록 허용해야 합니다.

주제

- Amazon ECS 클러스터에 Prometheus 지표 수집과 함께 CloudWatch 에이전트 설치
- 추가 Prometheus 소스 스크레이핑 및 해당 지표 가져오기
- (선택 사항) Prometheus 지표 테스트를 위한 컨테이너화된 Amazon ECS 워크로드 샘플 설정

Amazon ECS 클러스터에 Prometheus 지표 수집과 함께 CloudWatch 에이전트 설치

이 단원에서는 Amazon ECS를 실행 중인 클러스터에서 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설정하는 방법을 설명합니다. 이렇게 에이전트를 설정하면 에이전트가 해당 클러스터에서 실행 중인 다음 워크로드에 대한 지표를 자동으로 스크레이프하고 가져옵니다.

- AWS App Mesh
- Java/JMX

추가 Prometheus 워크로드 및 소스에서 지표를 스크레이프하고 가져오도록 에이전트를 구성할 수도 있습니다.

IAM 역할 설정

CloudWatch 에이전트 태스크 정의의 두 IAM 역할이 필요합니다. Container Insights가 이러한 역할을 자동으로 생성하도록 AWS CloudFormation 스택에서 **`CreateIAMRoles=True`**를 지정하면 올바른 권한을 가진 역할이 생성됩니다. 역할을 직접 생성하거나 기존 역할을 사용하려는 경우 다음 역할 및 권한이 필요합니다.

- CloudWatch 에이전트 ECS 태스크 역할 - CloudWatch 에이전트 컨테이너는 이 역할을 사용합니다. 여기에는 CloudWatchAgentServerPolicy 정책 및 다음과 같은 읽기 전용 권한을 포함하는 고객 관리형 정책이 포함되어야 합니다.
  - `ec2:DescribeInstances`
  - `ecs:ListTasks`
  - `ecs:ListServices`
  - `ecs:DescribeContainerInstances`
  - `ecs:DescribeServices`
  - `ecs:DescribeTasks`

  - `ecs:DescribeTaskDefinition`
- CloudWatch 에이전트 ECS 태스크 실행 역할 - Amazon ECS가 컨테이너를 시작하고 실행하는 데 필요한 역할입니다. 태스크 실행 역할에 AmazonSSMReadOnlyAccess, AmazonECSTaskExecutionRolePolicy 및 CloudWatchAgentServerPolicy 정책이 연결되어 있는지 확인합니다. Amazon ECS에서 사용할 더 민감한 데이터를 저장하려는 경우 민감한 데이터 지정 단원을 참조하세요.

AWS CloudFormation을 사용하여 Prometheus 모니터링이 포함된 CloudWatch 에이전트 설치

AWS CloudFormation을 사용하여 Amazon ECS 클러스터에 대해 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설치합니다. 다음 목록에는 AWS CloudFormation 템플릿에서 사용할 파라미터가 나와 있습니다.

- ECSClusterName - 대상 Amazon ECS 클러스터를 지정합니다.
- CreateIAMRoles - Amazon ECS 태스크 역할 및 Amazon ECS 태스크 실행 역할의 새 역할을 생성하려면 **`True`**를 지정합니다. 기존 역할을 재사용하려면 **`False`**를 지정합니다.
- TaskRoleName - CreateIAMRoles에서 **`True`**를 지정한 경우 이 파라미터는 새 Amazon ECS 태스크 역할에 사용할 이름을 지정합니다. CreateIAMRoles에서 **`False`**를 지정한 경우 이 파라미터는 Amazon ECS 태스크 역할로 사용할 기존 역할을 지정합니다.
- ExecutionRoleName - CreateIAMRoles에서 **`True`**를 지정한 경우 이 파라미터는 새 Amazon ECS 태스크 실행 역할에 사용할 이름을 지정합니다. CreateIAMRoles에서 **`False`**를 지정한 경우 이 파라미터는 Amazon ECS 태스크 실행 역할로 사용할 기존 역할을 지정합니다.
- ECSNetworkMode - EC2 시작 유형을 사용하는 경우 여기에서 네트워크 모드를 지정합니다. **`bridge`** 또는 **`host`**여야 합니다.
- ECSLaunchType - **`fargate`** 또는 **`EC2`**를 지정합니다.
- SecurityGroupID - ECSNetworkMode가 **`awsvpc`**인 경우 여기에서 보안 그룹 ID를 지정합니다.
- SubnetID - ECSNetworkMode가 **`awsvpc`**인 경우 여기에서 서브넷 ID를 지정합니다.

샘플 명령

이 단원에는 다양한 시나리오에서 Prometheus 모니터링과 함께 Container Insights를 설치하기 위한 샘플 AWS CloudFormation 명령이 포함되어 있습니다.

브리지 네트워크 모드에서 Amazon ECS 클러스터의 AWS CloudFormation 스택 생성

```
export AWS_PROFILE=your_aws_config_profile_eg_default
```

```
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_NETWORK_MODE=bridge
export CREATE_IAM_ROLES=True
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

호스트 네트워크 모드에서 Amazon ECS 클러스터의 AWS CloudFormation 스택 생성

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_NETWORK_MODE=host
export CREATE_IAM_ROLES=True
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name


curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
```

```
                ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

awsvpc 네트워크 모드에서 Amazon ECS 클러스터의 AWS CloudFormation 스택 생성

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_LAUNCH_TYPE=EC2
export CREATE_IAM_ROLES=True
export ECS_CLUSTER_SECURITY_GROUP=your_security_group_eg_sg-xxxxxxxxxx
export ECS_CLUSTER_SUBNET=your_subnet_eg_subnet-xxxxxxxxxx
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-${ECS_LAUNCH_TYPE}-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSLaunchType,ParameterValue=${ECS_LAUNCH_TYPE} \
                 ParameterKey=SecurityGroupID,ParameterValue=
${ECS_CLUSTER_SECURITY_GROUP} \
                 ParameterKey=SubnetID,ParameterValue=${ECS_CLUSTER_SUBNET} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

awsvpc 네트워크 모드에서 Fargate 클러스터의 AWS CloudFormation 스택 생성

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_LAUNCH_TYPE=FARGATE
export CREATE_IAM_ROLES=True
export ECS_CLUSTER_SECURITY_GROUP=your_security_group_eg_sg-xxxxxxxxxx
export ECS_CLUSTER_SUBNET=your_subnet_eg_subnet-xxxxxxxxxx
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-${ECS_LAUNCH_TYPE}-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSLaunchType,ParameterValue=${ECS_LAUNCH_TYPE} \
                 ParameterKey=SecurityGroupID,ParameterValue=
${ECS_CLUSTER_SECURITY_GROUP} \
                 ParameterKey=SubnetID,ParameterValue=${ECS_CLUSTER_SUBNET} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

AWS CloudFormation 스택에서 생성한 AWS 리소스

다음 표에는 AWS CloudFormation을 사용하여 Amazon ECS 클러스터에서 Prometheus 모니터링과 함께 Container Insights를 설정할 때 생성되는 AWS 리소스가 나와 있습니다.

리소스 유형	리소스 이름	설명
AWS::SSM::Parameter	AmazonCloudWatch-CWAgentConfig-$*ECS_CLUSTER_NAME* -$*ECS_LAUNCH_TYPE* -$*ECS_NETWORK_MODE*	이 리소스는 기본 App Mesh 및 Java/JMX 임베디드 지표 형식 정의가 있는 CloudWatch 에이전트입니다.

리소스 유형	리소스 이름	설명
AWS::SSM::Parameter	AmazonCloudWatch-PrometheusConfigName-$*ECS_CLUSTER_NAME* -$*ECS_LAUNCH_TYPE* -$*ECS_NETWORK_MODE*	이 리소스는 Prometheus 스크레이핑 구성입니다.
AWS::IAM::Role	$ECS_TASK_ROLE_NAME.	Amazon ECS 태스크 역할입니다. `CREATE_IAM_ROLES` 에서 **`True`**를 지정한 경우에만 생성됩니다.
AWS::IAM::Role	${ECS_EXECUTION_ROLE_NAME}	Amazon ECS 태스크 실행 역할입니다. `CREATE_IAM_ROLES` 에서 **`True`**를 지정한 경우에만 생성됩니다.
AWS::ECS::TaskDefinition	cwagent-prometheus-$*ECS_CLUSTER_NAME* -$*ECS_LAUNCH_TYPE* -$*ECS_NETWORK_MODE*	
AWS::ECS::Service	cwagent-prometheus-replica-service-$*ECS_LAUNCH_TYPE* -$*ECS_NETWORK_MODE*	

Prometheus 모니터링이 포함된 CloudWatch 에이전트의 AWS CloudFormation 스택 삭제

Amazon ECS 클러스터에서 CloudWatch 에이전트를 삭제하려면 다음 명령을 입력합니다.

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export CLOUDFORMATION_STACK_NAME=your_cloudformation_stack_name

aws cloudformation delete-stack \
--stack-name ${CLOUDFORMATION_STACK_NAME} \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

추가 Prometheus 소스 스크레이핑 및 해당 지표 가져오기

Prometheus 모니터링이 포함된 CloudWatch 에이전트는 Prometheus 지표를 스크레이프하는 데 두 가지 구성이 필요합니다. 하나는 Prometheus 설명서의 <scrape_config>에 설명된 표준 Prometheus 구성을 위한 것입니다. 다른 하나는 CloudWatch 에이전트 구성을 위한 것입니다.

Amazon ECS 클러스터의 경우 구성은 다음과 같이 Amazon ECS 태스크 정의의 보안 암호를 통해 AWS Systems Manager의 파라미터 스토어와 통합됩니다.

- 보안 암호 `PROMETHEUS_CONFIG_CONTENT`는 Prometheus 스크레이프 구성을 위한 것입니다.
- 보안 암호 `CW_CONFIG_CONTENT`는 CloudWatch 에이전트 구성을 위한 것입니다.

추가 Prometheus 지표 소스를 스크레이프하고 해당 지표를 CloudWatch에 가져오려면 Prometheus 스크레이프 구성과 CloudWatch 에이전트 구성을 모두 수정한 다음, 업데이트된 구성으로 에이전트를 다시 배포합니다.

VPC 보안 그룹 요구 사항

Prometheus 워크로드의 보안 그룹 수신 규칙은 프라이빗 IP로 Prometheus 지표를 스크레이프하기 위해 CloudWatch 에이전트에 대한 Prometheus 포트를 열어야 합니다.

CloudWatch 에이전트의 보안 그룹 송신 규칙은 CloudWatch 에이전트가 프라이빗 IP로 Prometheus 워크로드의 포트에 연결할 수 있도록 허용해야 합니다.

Prometheus 스크레이프 구성

CloudWatch 에이전트는 Prometheus 설명서의 <scrape_config>에 설명된 대로 표준 Prometheus 스크레이프 구성을 지원합니다. 이 섹션을 편집하여 이 파일에 이미 있는 구성을 업데이트하고 Prometheus 스크레이핑 대상을 더 추가할 수 있습니다. 기본적으로 샘플 구성 파일에는 다음과 같은 글로벌 구성 줄이 포함되어 있습니다.

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
```

- scrape_interval - 대상을 스크레이프하는 빈도를 정의합니다.
- scrape_timeout - 스크레이프 요청 시간이 초과되기 전에 대기할 시간을 정의합니다.

작업 수준에서 이러한 설정에 다른 값을 정의하여 전역 구성을 재정의할 수도 있습니다.

Prometheus 스크레이핑 작업

CloudWatch 에이전트 YAML 파일에는 일부 기본 스크레이핑 작업이 이미 구성되어 있습니다. 예를 들어 `cwagent-ecs-prometheus-metric-for-bridge-host.yaml`과 같은 Amazon ECS의 YAML 파일에서 기본 스크레이핑 작업은 `ecs_service_discovery` 섹션에서 구성됩니다.

```
"ecs_service_discovery": {
                  "sd_frequency": "1m",
                  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
                  "docker_label": {
                  },
                  "task_definition_list": [
                    {
                      "sd_job_name": "ecs-appmesh-colors",
                      "sd_metrics_ports": "9901",
                      "sd_task_definition_arn_pattern": ".*:task-definition\/.*-
ColorTeller-(white):[0-9]+",
                      "sd_metrics_path": "/stats/prometheus"
                    },
                    {
                      "sd_job_name": "ecs-appmesh-gateway",
                      "sd_metrics_ports": "9901",
                      "sd_task_definition_arn_pattern": ".*:task-definition/.*-
ColorGateway:[0-9]+",
                      "sd_metrics_path": "/stats/prometheus"
                    }
                  ]
                }
```

이러한 각 기본 대상은 스크레이프되고 지표는 임베디드 지표 형식을 사용하여 로그 이벤트로 CloudWatch에 전송됩니다. 자세한 내용은 로그 내에 지표 포함 단원을 참조하십시오.

Amazon ECS 클러스터의 로그 이벤트는 /aws/ecs/containerinsights/***cluster_name***/prometheus 로그 그룹에 저장됩니다.

각 스크레이핑 작업은 이 로그 그룹의 서로 다른 로그 스트림에 포함됩니다.

새 스크레이핑 대상을 추가하려면 YAML 파일의 `ecs_service_discovery` 섹션 아래 `task_definition_list` 섹션에 새 항목을 추가하고 에이전트를 다시 시작합니다. 이 프로세스의 예는 새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Prometheus API 서버 지표 단원을 참조하세요.

Prometheus에 대한 CloudWatch 에이전트 구성

CloudWatch 에이전트 구성 파일에는 `metrics_collected` 아래에 Prometheus 스크레이핑 구성에 대한 `prometheus` 섹션이 있습니다. 여기에는 다음 구성 옵션이 포함됩니다.

- cluster_name - 로그 이벤트에서 레이블로 추가할 클러스터 이름을 지정합니다. 이 필드는 선택 사항입니다. 생략하는 경우 에이전트가 Amazon ECS 클러스터 이름을 감지할 수 있습니다.
- log_group_name - 스크레이프한 Prometheus 지표의 로그 그룹 이름을 지정합니다. 이 필드는 선택 사항입니다. 생략하는 경우 CloudWatch는 Amazon ECS 클러스터의 로그에 /aws/ecs/containerinsights/***cluster_name***/prometheus를 사용합니다.
- prometheus_config_path - Prometheus 스크레이프 구성 파일 경로를 지정합니다. 이 필드의 값이 `env:`로 시작하는 경우 Prometheus 스크레이프 구성 파일 내용이 컨테이너의 환경 변수에서 검색됩니다. 이 값은 변경하지 마세요.
- ecs_service_discovery - Amazon ECS Prometheus 대상 자동 검색 기능의 구성을 지정하는 섹션입니다. Prometheus 대상을 검색하기 위해 두 모드, 즉 컨테이너의 Docker 레이블을 기반으로 하는 검색 또는 Amazon ECS 태스크 정의 ARN 정규 표현식을 기반으로 하는 검색이 지원됩니다. 두 모드를 함께 사용할 수 있습니다. CloudWatch 에이전트는 {private_ip}:{port}/{metrics_path}를 기반으로 검색된 대상의 중복을 제거합니다.

  `ecs_service_discovery` 섹션에는 다음 필드가 포함될 수 있습니다.

  - `sd_frequency`는 Prometheus Exporter를 검색하는 빈도입니다. 숫자와 단위 접미사를 지정합니다. 예를 들어 1분마다 한 번의 경우 `1m` 또는 30초마다 한 번의 경우 `30s`입니다. 유효한 단위 접미사는 `ns`, `us`, `ms`, `s`, `m`, `h`입니다.

    이 필드는 선택 사항입니다. 기본값은 60초(1분)입니다.
  - `sd_target_cluster`는 자동 검색의 대상 Amazon ECS 클러스터 이름입니다. 이 필드는 선택 사항입니다. 기본값은 CloudWatch 에이전트가 설치된 Amazon ECS 클러스터의 이름입니다.
  - `sd_cluster_region`은 대상 Amazon ECS 클러스터의 리전입니다. 이 필드는 선택 사항입니다. 기본값은 CloudWatch 에이전트가 설치된 Amazon ECS 클러스터의 리전입니다.
  - `sd_result_file`은 Prometheus 대상 결과의 YAML 파일 경로입니다. Prometheus 스크레이프 구성은 이 파일을 참조합니다.
  - `docker_label`은 Docker 레이블 기반 서비스 검색을 위한 구성을 지정하는 데 사용할 수 있는 선택적 섹션입니다. 이 섹션을 생략하면 Docker 레이블 기반 검색이 사용되지 않습니다. 이 섹션에는 다음 필드가 포함될 수 있습니다.

  - `sd_port_label`은 Prometheus 지표에 대한 컨테이너 포트를 지정하는 컨테이너의 Docker 레이블 이름입니다. 기본 값은 `ECS_PROMETHEUS_EXPORTER_PORT`입니다. 컨테이너에 이 Docker 레이블이 없다면 CloudWatch 에이전트는 이 필드를 건너뜁니다.
  - `sd_metrics_path_label`은 Prometheus 지표 경로를 지정하는 컨테이너의 Docker 레이블 이름입니다. 기본 값은 `ECS_PROMETHEUS_METRICS_PATH`입니다. 컨테이너에 이 도커 레이블이 없다면 에이전트는 기본 경로 `/metrics`를 가정합니다.
  - `sd_job_name_label`은 Prometheus 스크레이프 작업 이름을 지정하는 컨테이너의 Docker 레이블 이름입니다. 기본 값은 `job`입니다. 컨테이너에 이 Docker 레이블이 없다면 CloudWatch 에이전트는 Prometheus 스크레이프 구성의 작업 이름을 사용합니다.
- `task_definition_list`는 태스크 정의 기반 서비스 검색의 구성을 지정하는 데 사용할 수 있는 선택적 섹션입니다. 이 섹션을 생략하면 태스크 정의 기반 검색이 사용되지 않습니다. 이 섹션에는 다음 필드가 포함될 수 있습니다.
  - `sd_task_definition_arn_pattern`은 검색할 Amazon ECS 태스크 정의를 지정하는 데 사용할 패턴입니다. 이는 정규 표현식입니다.
  - `sd_metrics_ports`는 Prometheus 지표에 대한 containerPort를 나열합니다. containerPort를 세미콜론으로 구분합니다.
  - `sd_container_name_pattern`은 Amazon ECS 태스크 컨테이너 이름을 지정합니다. 이는 정규 표현식입니다.
  - `sd_metrics_path`는 Prometheus 지표 경로를 지정합니다. 이 필드를 생략하면 에이전트는 기본 경로 `/metrics`를 가정합니다.
  - `sd_job_name`은 Prometheus 스크레이프 작업 이름을 지정합니다. 이 필드를 생략하면 CloudWatch 에이전트는 Prometheus 스크레이프 구성의 작업 이름을 사용합니다.
- `service_name_list_for_tasks`는 서비스 이름 기반 검색의 구성을 지정하는 데 사용할 수 있는 선택적 섹션입니다. 이 섹션을 생략하면 서비스 이름 기반 검색이 사용되지 않습니다. 이 섹션에는 다음 필드가 포함될 수 있습니다.
  - `sd_service_name_pattern`은 태스크를 검색할 Amazon ECS 서비스를 지정하는 데 사용할 패턴입니다. 이는 정규 표현식입니다.
  - `sd_metrics_ports`는 Prometheus 지표에 대한 `containerPort`를 나열합니다. 여러 `containerPorts`를 세미콜론으로 구분합니다.
  - `sd_container_name_pattern`은 Amazon ECS 태스크 컨테이너 이름을 지정합니다. 이는 정규 표현식입니다.
  - `sd_metrics_path`는 Prometheus 지표 경로를 지정합니다. 이 필드를 생략하면 에이전트는 기본 경로 `/metrics`를 가정합니다.

    - sd_job_name은 Prometheus 스크레이프 작업 이름을 지정합니다. 이 필드를 생략하면 CloudWatch 에이전트는 Prometheus 스크레이프 구성의 작업 이름을 사용합니다.
- metric_declaration - 생성할 임베디드 지표 형식이 있는 로그 배열을 지정하는 섹션입니다. CloudWatch 에이전트가 기본적으로 가져오는 각 Prometheus 소스에 대한 metric_declaration 섹션이 있습니다. 이러한 섹션에는 각각 다음 필드가 포함됩니다.
    - label_matcher는 source_labels에 나열된 레이블의 값을 확인하는 정규 표현식입니다. 일치하는 지표는 CloudWatch에 전송된 임베디드 지표 형식에 포함할 수 있습니다.

      source_labels에 여러 레이블이 지정된 경우 label_matcher의 정규 표현식에 ^ 또는 $ 문자를 사용하지 않는 것이 좋습니다.
    - source_labels는 label_matcher 줄에 의해 확인되는 레이블의 값을 지정합니다.
    - label_separator는 여러 source_labels가 지정된 경우 label_matcher 줄에 사용할 구분 기호를 지정합니다. 기본값은 ;입니다. 다음 예에서 label_matcher 줄에 이 기본값이 사용된 것을 볼 수 있습니다.
    - metric_selectors는 수집하여 CloudWatch에 보낼 지표를 지정하는 정규 표현식입니다.
    - dimensions는 선택한 각 지표의 CloudWatch 측정기준으로 사용할 레이블 목록입니다.

다음 metric_declaration 예를 참조하세요.

```
"metric_declaration": [
  {
     "source_labels":[ "Service", "Namespace"],
     "label_matcher":"(.*node-exporter.*|.*kube-dns.*);kube-system$",
     "dimensions":[
        ["Service", "Namespace"]
     ],
     "metric_selectors":[
        "^coredns_dns_request_type_count_total$"
     ]
  }
]
```

이 예에서는 다음 조건이 충족될 경우 임베디드 지표 형식 섹션을 로그 이벤트로 전송하도록 구성합니다.

- Service의 값에 node-exporter 또는 kube-dns가 포함되어 있습니다.
- Namespace의 값이 kube-system입니다.

- Prometheus `coredns_dns_request_type_count_total` 지표에 `Service`와 `Namespace` 레이블이 모두 포함되어 있습니다.

전송되는 로그 이벤트에는 다음과 같은 강조 표시된 섹션이 포함됩니다.

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"coredns_dns_request_type_count_total"
            }
         ],
         "Dimensions":[
            [
               "Namespace",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "Namespace":"kube-system",
   "Service":"kube-dns",
   "coredns_dns_request_type_count_total":2562,
   "eks_amazonaws_com_component":"kube-dns",
   "instance":"192.168.61.254:9153",
   "job":"kubernetes-service-endpoints",
   ...
}
```

Amazon ECS 클러스터의 자동 검색에 대한 자세한 가이드

Prometheus는 <scrape_config>에 설명된 대로 수십 가지의 동적 서비스 검색 메커니즘을 제공합니다. 그러나 Amazon ECS에 대한 기본 제공 서비스 검색은 없습니다. CloudWatch 에이전트가 이 메커니즘을 추가합니다.

Amazon ECS Prometheus 서비스 검색을 사용 설정하면 CloudWatch 에이전트는 Amazon ECS 및 Amazon EC2 프런트엔드에 대해 다음 API 호출을 주기적으로 수행하여 대상 ECS 클러스터에서 실행 중인 ECS 태스크의 메타데이터를 검색합니다.

```
EC2:DescribeInstances
```

```
ECS:ListTasks
ECS:ListServices
ECS:DescribeContainerInstances
ECS:DescribeServices
ECS:DescribeTasks
ECS:DescribeTaskDefinition
```

CloudWatch 에이전트는 메타데이터를 사용하여 ECS 클러스터 내에서 Prometheus 대상을 스캔합니다. CloudWatch 에이전트는 다음과 같은 세 가지 서비스 검색 모드를 지원합니다.

- 컨테이너 Docker 레이블 기반 서비스 검색
- 이 예에서는 ECS 태스크 정의 ARN 정규 표현식 기반 서비스 검색을 사용 설정합니다.
- ECS 서비스 이름 정규 표현식 기반 서비스 검색

모든 모드를 함께 사용할 수 있습니다. CloudWatch 에이전트는 `{private_ip}:{port}/{metrics_path}`를 기반으로 검색된 대상의 중복을 제거합니다.

검색된 모든 대상은 CloudWatch 에이전트 컨테이너 내의 `sd_result_file` 구성 필드에서 지정한 결과 파일에 작성됩니다. 다음은 샘플 결과 파일입니다.

```
- targets:
  - 10.6.1.95:32785
  labels:
    __metrics_path__: /metrics
    ECS_PROMETHEUS_EXPORTER_PORT: "9406"
    ECS_PROMETHEUS_JOB_NAME: demo-jar-ec2-bridge-dynamic
    ECS_PROMETHEUS_METRICS_PATH: /metrics
    InstanceType: t3.medium
    LaunchType: EC2
    SubnetId: subnet-123456789012
    TaskDefinitionFamily: demo-jar-ec2-bridge-dynamic-port
    TaskGroup: family:demo-jar-ec2-bridge-dynamic-port
    TaskRevision: "7"
    VpcId: vpc-01234567890
    container_name: demo-jar-ec2-bridge-dynamic-port
    job: demo-jar-ec2-bridge-dynamic
- targets:
  - 10.6.3.193:9404
  labels:
    __metrics_path__: /metrics
    ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_B: "9404"
```

```
    ECS_PROMETHEUS_JOB_NAME: demo-tomcat-ec2-bridge-mapped-port
    ECS_PROMETHEUS_METRICS_PATH: /metrics
    InstanceType: t3.medium
    LaunchType: EC2
    SubnetId: subnet-123456789012
    TaskDefinitionFamily: demo-tomcat-ec2-bridge-mapped-port
    TaskGroup: family:demo-jar-tomcat-bridge-mapped-port
    TaskRevision: "12"
    VpcId: vpc-01234567890
    container_name: demo-tomcat-ec2-bridge-mapped-port
    job: demo-tomcat-ec2-bridge-mapped-port
```

이 결과 파일을 Prometheus 파일 기반 서비스 검색과 직접 통합할 수 있습니다. Prometheus 파일 기반 서비스 검색에 대한 자세한 내용은 <file_sd_config>를 참조하세요.

결과 파일이 `/tmp/cwagent_ecs_auto_sd.yaml`에 작성되었다고 가정합니다. 다음 Prometheus 스크레이프 구성은 이 파일을 사용합니다.

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: cwagent-ecs-file-sd-config
    sample_limit: 10000
    file_sd_configs:
      - files: [ "/tmp/cwagent_ecs_auto_sd.yaml" ]
```

CloudWatch 에이전트는 검색된 대상에 대해 다음과 같은 레이블도 추가합니다.

- `container_name`
- `TaskDefinitionFamily`
- `TaskRevision`
- `TaskGroup`
- `StartedBy`
- `LaunchType`
- `job`
- `__metrics_path__`
- Docker 레이블

클러스터에 EC2 시작 유형이 있는 경우 다음과 같은 레이블 세 가지가 추가됩니다.

- `InstanceType`
- `VpcId`
- `SubnetId`

> **Note**
>
> 정규 표현식 `[a-zA-Z_][a-zA-Z0-9_]*`와 일치하지 않는 Docker 레이블은 필터링되어 제외됩니다. 이는 Prometheus 설명서에 있는 구성 파일의 `label_name`에 나열된 Prometheus 규칙과 일치합니다.

ECS 서비스 검색 구성 예

이 단원에는 ECS 서비스 검색을 보여 주는 예가 포함되어 있습니다.

예 1

```
"ecs_service_discovery": {
  "sd_frequency": "1m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
  }
}
```

이 예에서는 Docker 레이블 기반 서비스 검색을 사용 설정합니다. CloudWatch 에이전트는 1분마다 한 번씩 ECS 태스크의 메타데이터를 쿼리하고 검색된 대상을 CloudWatch 에이전트 컨테이너 내의 `/tmp/cwagent_ecs_auto_sd.yaml` 파일에 씁니다.

`docker_label` 섹션에 있는 `sd_port_label`의 기본값은 `ECS_PROMETHEUS_EXPORTER_PORT`입니다. ECS 태스크의 실행 중인 컨테이너에 `ECS_PROMETHEUS_EXPORTER_PORT` Docker 레이블이 있는 경우 CloudWatch 에이전트는 해당 값을 `container port`로 사용하여 컨테이너의 노출된 포트를 모두 스캔합니다. 일치하는 항목이 있다면 매핑된 호스트 포트와 컨테이너의 프라이빗 IP를 사용하여 `private_ip:host_port` 형식으로 Prometheus Exporter 대상을 구성합니다.

`docker_label` 섹션에 있는 `sd_metrics_path_label`의 기본값은 `ECS_PROMETHEUS_METRICS_PATH`입니다. 컨테이너에 이 Docker 레이블이 있는 경우 해당 값을

__metrics_path__로 사용합니다. 컨테이너에 이 레이블이 없다면 기본값인 /metrics를 사용합니다.

docker_label 섹션에 있는 sd_job_name_label의 기본값은 job입니다. 컨테이너에 이 Docker 레이블이 있는 경우 해당 값은 대상에 대한 레이블 중 하나로 추가되어 Prometheus 구성에 지정된 기본 작업 이름을 대체합니다. 이 Docker 레이블의 값은 CloudWatch Logs 로그 그룹의 로그 스트림 이름으로 사용됩니다.

예제 2

```
"ecs_service_discovery": {
  "sd_frequency": "15s",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
    "sd_port_label": "ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_A",
    "sd_job_name_label": "ECS_PROMETHEUS_JOB_NAME"
  }
}
```

이 예에서는 Docker 레이블 기반 서비스 검색을 사용 설정합니다. CloudWatch 에이전트는 15초마다 ECS 태스크의 메타데이터를 쿼리하고 검색된 대상을 CloudWatch 에이전트 컨테이너 내의 /tmp/cwagent_ecs_auto_sd.yaml 파일에 씁니다. ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_A의 Docker 레이블이 있는 컨테이너를 스캔합니다. Docker 레이블 ECS_PROMETHEUS_JOB_NAME의 값을 작업 이름으로 사용합니다.

예 3

```
"ecs_service_discovery": {
  "sd_frequency": "5m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "task_definition_list": [
    {
      "sd_job_name": "java-prometheus",
      "sd_metrics_path": "/metrics",
      "sd_metrics_ports": "9404; 9406",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*javajmx.*:[0-9]+"
    },
    {
      "sd_job_name": "envoy-prometheus",
      "sd_metrics_path": "/stats/prometheus",
      "sd_container_name_pattern": "^envoy$",
      "sd_metrics_ports": "9901",
```

```
      "sd_task_definition_arn_pattern": ".*:task-definition/.*appmesh.*:23"
    }
  ]
}
```

이 예에서는 ECS 태스크 정의 ARN 정규 표현식 기반 서비스 검색을 사용 설정합니다. CloudWatch 에이전트는 5분마다 ECS 태스크의 메타데이터를 쿼리하고 검색된 대상을 CloudWatch 에이전트 컨테이너 내의 `/tmp/cwagent_ecs_auto_sd.yaml` 파일에 씁니다.

다음과 같이 두 태스크 정의 ARN 정규 표현식 섹션이 정의됩니다.

- 첫 번째 섹션의 경우 ECS 태스크 정의 ARN에 `javajmx`가 있는 ECS 태스크가 컨테이너 포트 스캔에 대해 필터링됩니다. 이러한 ECS 태스크 내의 컨테이너가 9404 또는 9406의 컨테이너 포트를 노출하는 경우 컨테이너의 프라이빗 IP와 함께 매핑된 호스트 포트를 사용하여 Prometheus Exporter 대상을 생성합니다. `sd_metrics_path` 값은 `__metrics_path__`를 `/metrics`로 설정합니다. 따라서 CloudWatch 에이전트는 `private_ip:host_port/metrics`에서 Prometheus 지표를 스크레이프하고, 스크레이프한 지표를 CloudWatch Logs 로그 그룹 `/aws/ecs/containerinsights/cluster_name/prometheus`의 `java-prometheus` 로그 스트림에 전송합니다.
- 두 번째 섹션의 경우 ECS 태스크 정의 ARN에 `appmesh`가 있고 `version`이 `:23`인 ECS 태스크가 컨테이너 포트 스캔에 대해 필터링됩니다. `envoy`라는 이름의 컨테이너가 `9901`의 컨테이너 포트를 노출하는 경우 컨테이너의 프라이빗 IP와 함께 매핑된 호스트 포트를 사용하여 Prometheus Exporter 대상을 생성합니다. 이러한 ECS 태스크 내의 값은 9404 또는 9406의 컨테이너 포트를 노출하며, 컨테이너의 프라이빗 IP와 함께 매핑된 호스트 포트를 사용하여 Prometheus Exporter 대상을 생성합니다. `sd_metrics_path` 값은 `__metrics_path__`를 `/stats/prometheus`로 설정합니다. 따라서 CloudWatch 에이전트는 `private_ip:host_port/stats/prometheus`에서 Prometheus 지표를 스크레이프하고, 스크레이프한 지표를 CloudWatch Logs 로그 그룹 `/aws/ecs/containerinsights/cluster_name/prometheus`의 `envoy-prometheus` 로그 스트림에 전송합니다.

예 4

```
"ecs_service_discovery": {
  "sd_frequency": "5m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "service_name_list_for_tasks": [
    {
      "sd_job_name": "nginx-prometheus",
      "sd_metrics_path": "/metrics",
```

```
      "sd_metrics_ports": "9113",
      "sd_service_name_pattern": "^nginx-.*"
    },
    {
      "sd_job_name": "haproxy-prometheus",
      "sd_metrics_path": "/stats/metrics",
      "sd_container_name_pattern": "^haproxy$",
      "sd_metrics_ports": "8404",
      "sd_service_name_pattern": ".*haproxy-service.*"
    }
  ]
}
```

이 예에서는 ECS 서비스 이름 정규 표현식 기반 서비스 검색을 사용 설정합니다. CloudWatch 에이전트는 5분마다 ECS 서비스의 메타데이터를 쿼리하고 검색된 대상을 CloudWatch 에이전트 컨테이너 내의 `/tmp/cwagent_ecs_auto_sd.yaml` 파일에 씁니다.

다음과 같이 두 서비스 이름 정규 표현식 섹션이 정의됩니다.

- 첫 번째 섹션의 경우 이름이 정규 표현식 `^nginx-.*`와 일치하는 ECS 서비스와 연결된 ECS 태스크가 컨테이너 포트 스캔에 대해 필터링됩니다. 이러한 ECS 태스크 내의 컨테이너가 9113의 컨테이너 포트를 노출하는 경우 컨테이너의 프라이빗 IP와 함께 매핑된 호스트 포트를 사용하여 Prometheus Exporter 대상을 생성합니다. `sd_metrics_path` 값은 `__metrics_path__`를 `/metrics`로 설정합니다. 따라서 CloudWatch 에이전트는 `private_ip:host_port/metrics`에서 Prometheus 지표를 스크레이프하고, 스크레이프한 지표를 CloudWatch Logs 로그 그룹 `/aws/ecs/containerinsights/cluster_name/prometheus`의 `nginx-prometheus` 로그 스트림에 전송합니다.
- 두 번째 섹션의 경우 이름이 정규 표현식 `.*haproxy-service.*`와 일치하는 ECS 서비스와 연결된 ECS 태스크가 컨테이너 포트 스캔에 대해 필터링됩니다. `haproxy`라는 이름의 컨테이너가 8404의 컨테이너 포트를 노출하는 경우 컨테이너의 프라이빗 IP와 함께 매핑된 호스트 포트를 사용하여 Prometheus Exporter 대상을 생성합니다. `sd_metrics_path` 값은 `__metrics_path__`를 `/stats/metrics`로 설정합니다. 따라서 CloudWatch 에이전트는 `private_ip:host_port/stats/metrics`에서 Prometheus 지표를 스크레이프하고, 스크레이프한 지표를 CloudWatch Logs 로그 그룹 `/aws/ecs/containerinsights/cluster_name/prometheus`의 `haproxy-prometheus` 로그 스트림에 전송합니다.

예 5

```
"ecs_service_discovery": {
```

```
  "sd_frequency": "1m30s",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
    "sd_port_label": "MY_PROMETHEUS_EXPORTER_PORT_LABEL",
    "sd_metrics_path_label": "MY_PROMETHEUS_METRICS_PATH_LABEL",
    "sd_job_name_label": "MY_PROMETHEUS_METRICS_NAME_LABEL"
  }
  "task_definition_list": [
    {
      "sd_metrics_ports": "9150",
      "sd_task_definition_arn_pattern": "*memcached.*"
    }
  ]
}
```

이 예에서는 두 ECS 서비스 검색 모드를 모두 사용하도록 설정합니다. CloudWatch 에이전트는 90초마다 ECS 태스크의 메타데이터를 쿼리하고 검색된 대상을 CloudWatch 에이전트 컨테이너 내의 `/tmp/cwagent_ecs_auto_sd.yaml` 파일에 씁니다.

Docker 기반 서비스 검색 구성의 경우:

- Docker 레이블 `MY_PROMETHEUS_EXPORTER_PORT_LABEL`이 있는 ECS 태스크가 Prometheus 포트 스캔에 대해 필터링됩니다. 대상 Prometheus 컨테이너 포트는 `MY_PROMETHEUS_EXPORTER_PORT_LABEL` 레이블의 값으로 지정합니다.
- `__metrics_path__`에 Docker 레이블 `MY_PROMETHEUS_EXPORTER_PORT_LABEL`의 값을 사용합니다. 컨테이너에 이 Docker 레이블이 없다면 기본값인 `/metrics`를 사용합니다.
- Docker 레이블 `MY_PROMETHEUS_EXPORTER_PORT_LABEL`의 값을 작업 레이블로 사용합니다. 컨테이너에 이 Docker 레이블이 없다면 Prometheus 구성에 정의된 작업 이름을 사용합니다.

ECS 태스크 정의 ARN 정규 표현식 기반 서비스 검색 구성의 경우:

- ECS 태스크 정의 ARN에 `memcached`가 있는 ECS 태스크가 컨테이너 포트 스캔에 대해 필터링됩니다. 대상 Prometheus 컨테이너 포트는 `sd_metrics_ports`에 정의된 대로 9150입니다. 기본 지표 경로인 `/metrics`를 사용합니다. Prometheus 구성에 정의된 작업 이름을 사용합니다.

(선택 사항) Prometheus 지표 테스트를 위한 컨테이너화된 Amazon ECS 워크로드 샘플 설정

CloudWatch Container Insights에서 Prometheus 지표 지원을 테스트하기 위해 다음과 같은 컨테이너화된 워크로드 중 하나 이상을 설정할 수 있습니다. Prometheus가 지원되는 CloudWatch 에이

전트는 이러한 각 워크로드에서 지표를 자동으로 수집합니다. 기본적으로 수집되는 지표를 보려면 CloudWatch 에이전트가 수집하는 Prometheus 지표 단원을 참조하세요.

주제

- Amazon ECS 클러스터의 App Mesh 워크로드 샘플
- Amazon ECS 클러스터의 Java/JMX 워크로드 샘플
- Amazon ECS 클러스터의 NGINX 워크로드 샘플
- Amazon ECS 클러스터의 NGINX Plus 워크로드 샘플
- 새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Amazon ECS의 Memcached
- Amazon ECS Fargate에서 Redis Prometheus 지표를 스크레이프하기 위한 튜토리얼

Amazon ECS 클러스터의 App Mesh 워크로드 샘플

Amazon ECS의 Prometheus 워크로드 샘플에서 지표를 수집하려면 클러스터에서 Container Insights를 실행 중이어야 합니다. Container Insights 설치에 대한 자세한 내용은 Amazon ECS에서 Container Insights 설정 단원을 참조하세요.

먼저, 이 시연에 따라 Amazon ECS 클러스터에 샘플 컬러 앱을 배포합니다. 완료하면 포트 9901에 App Mesh Prometheus 지표가 노출됩니다.

다음으로, 다음 단계에 따라 컬러 앱을 설치한 동일한 Amazon ECS 클러스터에 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설치합니다. 이 단원의 단계에서는 브리지 네트워크 모드에서 CloudWatch 에이전트를 설치합니다.

시연에서 설정한 환경 변수인 `ENVIRONMENT_NAME`, `AWS_PROFILE`, `AWS_DEFAULT_REGION`은 다음 단계에서도 사용됩니다.

테스트를 위해 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설치하려면

1. 다음 명령을 입력하여 AWS CloudFormation 템플릿을 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

2. 다음 명령을 입력하여 네트워크 모드를 설정합니다.

```
export ECS_CLUSTER_NAME=${ENVIRONMENT_NAME}
```

```
export ECS_NETWORK_MODE=bridge
```

3. 다음 명령을 입력하여 AWS CloudFormation 스택을 생성합니다.

```
aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=True \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=CWAgent-Prometheus-
TaskRole-${ECS_CLUSTER_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=CWAgent-Prometheus-
ExecutionRole-${ECS_CLUSTER_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

4. (선택 사항) AWS CloudFormation 스택이 생성되면 CREATE_COMPLETE 메시지가 표시됩니다. 해당 메시지가 표시되기 전에 상태를 확인하려면 다음 명령을 입력합니다.

```
aws cloudformation describe-stacks \
--stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--query 'Stacks[0].StackStatus' \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

문제 해결

시연 단계에서는 jq를 사용하여 AWS CLI의 출력 결과를 구문 분석합니다. jq 설치에 대한 자세한 내용은 jq를 참조하세요. jq가 올바르게 구문 분석할 수 있도록 다음 명령을 사용하여 AWS CLI의 기본 출력 형식을 JSON으로 설정합니다.

```
$ aws configure
```

[기본 출력 형식(`Default output format`)] 응답에 **`json`**을 입력합니다.

Prometheus 모니터링이 포함된 CloudWatch 에이전트 제거

테스트를 마쳤으면 다음 명령을 입력하여 AWS CloudFormation 스택을 삭제함으로써 CloudWatch 에이전트를 제거합니다.

```
aws cloudformation delete-stack \
--stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

Amazon ECS 클러스터의 Java/JMX 워크로드 샘플

JMX Exporter는 JMX mBeans를 스크레이프하여 노출할 수 있는 공식 Prometheus 익스포터입니다. 자세한 내용은 prometheus/jmx_exporter를 참조하세요.

Prometheus가 지원되는 CloudWatch 에이전트는 Amazon ECS 클러스터의 서비스 검색 구성을 기반으로 Java/JMX Prometheus 지표를 스크레이프합니다. 지표를 다른 포트 또는 metrics_path에 노출하도록 JMX Exporter를 구성할 수 있습니다. 포트 또는 경로를 변경하는 경우 CloudWatch 에이전트 구성의 기본 `ecs_service_discovery` 섹션을 업데이트하세요.

Amazon ECS의 Prometheus 워크로드 샘플에서 지표를 수집하려면 클러스터에서 Container Insights를 실행 중이어야 합니다. Container Insights 설치에 대한 자세한 내용은 Amazon ECS에서 Container Insights 설정 단원을 참조하세요.

Amazon ECS 클러스터의 Java/JMX 샘플 워크로드를 설치하려면

1. 이 단원의 절차에 따라 Docker 이미지를 생성합니다.
   - 예: Prometheus 지표가 있는 Java Jar 애플리케이션 Docker 이미지
   - 예: Prometheus 지표가 있는 Apache Tomcat Docker 이미지
2. Amazon ECS 태스크 정의 파일에서 다음과 같은 두 Docker 레이블을 지정합니다. 그러면 클러스터에서 태스크 정의를 Amazon ECS 서비스 또는 Amazon ECS 태스크로 실행할 수 있습니다.
   - `ECS_PROMETHEUS_EXPORTER_PORT`를 Prometheus 지표가 노출되는 containerPort를 가리키도록 설정합니다.
   - `Java_EMF_Metrics`를 `true`으로 설정합니다. CloudWatch 에이전트는 이 플래그를 사용하여 로그 이벤트에 임베디드 지표 형식을 생성합니다.

   다음은 그 예제입니다.

```
{
  "family": "workload-java-ec2-bridge",
  "taskRoleArn": "{{task-role-arn}}",
```

```
"executionRoleArn": "{{execution-role-arn}}",
"networkMode": "bridge",
"containerDefinitions": [
  {
    "name": "tomcat-prometheus-workload-java-ec2-bridge-dynamic-port",
    "image": "your_docker_image_tag_for_tomcat_with_prometheus_metrics",
    "portMappings": [
      {
        "hostPort": 0,
        "protocol": "tcp",
        "containerPort": 9404
      }
    ],
    "dockerLabels": {
      "ECS_PROMETHEUS_EXPORTER_PORT": "9404",
      "Java_EMF_Metrics": "true"
    }
  }
],
"requiresCompatibilities": [
  "EC2"  ],
"cpu": "256",
"memory": "512"
}
```

AWS CloudFormation 템플릿에서 CloudWatch 에이전트의 기본 설정은 Docker 레이블 기반 서비스 검색과 태스크 정의 ARN 기반 서비스 검색을 모두 사용 설정합니다. 이러한 기본 설정을 보려면 CloudWatch 에이전트 YAML 구성 파일의 65번 코드 줄을 참조하세요. ECS_PROMETHEUS_EXPORTER_PORT 레이블이 있는 컨테이너는 Prometheus 스크레이핑에 대해 지정된 컨테이너 포트를 기반으로 자동 검색됩니다.

또한 CloudWatch 에이전트의 기본 설정에는 동일한 파일의 112번 코드 줄에서 찾아볼 수 있는 Java/JMX의 metric_declaration 설정도 있습니다. 대상 컨테이너의 모든 Docker 레이블은 Prometheus 지표의 레이블로 추가되고 CloudWatch Logs에 전송됩니다. Java_EMF_Metrics="true"의 Docker 레이블이 있는 Java/JMX 컨테이너의 경우 임베디드 지표 형식이 생성됩니다.

Amazon ECS 클러스터의 NGINX 워크로드 샘플

NGINX Prometheus Exporter는 NGINX 데이터를 Prometheus 지표로 스크레이프하고 노출할 수 있습니다. 이 예에서는 Amazon ECS의 NGINX 역방향 프록시 서비스와 함께 Exporter를 사용합니다.

NGINX Prometheus Exporter에 대한 자세한 내용은 Github의 nginx-prometheus-exporter를 참조하세요. NGINX 역방향 프록시에 대한 자세한 내용은 Github의 ecs-nginx-reverse-proxy를 참조하세요.

Prometheus가 지원되는 CloudWatch 에이전트는 Amazon ECS 클러스터의 서비스 검색 구성을 기반으로 NGINX Prometheus 지표를 스크레이프합니다. 지표를 다른 포트 또는 경로에 노출하도록 NGINX Prometheus Exporter를 구성할 수 있습니다. 포트 또는 경로를 변경하는 경우 CloudWatch 에이전트 구성 파일의 `ecs_service_discovery` 섹션을 업데이트하세요.

Amazon ECS 클러스터의 NGINX 역방향 프록시 샘플 워크로드 설치

다음 단계에 따라 NGINX 역방향 프록시 샘플 워크로드를 설치합니다.

Docker 이미지 생성

NGINX 역방향 프록시 샘플 워크로드의 Docker 이미지를 생성하려면

1. NGINX 역방향 프록시 리포지토리에서 다음 폴더를 다운로드합니다. https://github.com/awslabs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/
2. app 디렉터리를 찾아 해당 디렉터리에서 이미지를 구축합니다.

   ```
   docker build -t web-server-app ./path-to-app-directory
   ```

3. NGINX용 사용자 지정 이미지를 구축합니다. 먼저, 다음과 같은 두 파일이 포함된 디렉터리를 만듭니다.

   - 샘플 Dockerfile:

     ```
     FROM nginx
     COPY nginx.conf /etc/nginx/nginx.conf
     ```

   - https://github.com/awslabs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/에서 수정한 `nginx.conf` 파일:

     ```
     events {
       worker_connections 768;
     }

     http {
       # Nginx will handle gzip compression of responses from the app server
       gzip on;
       gzip_proxied any;
       gzip_types text/plain application/json;
     ```

```
  gzip_min_length 1000;

  server{
    listen 8080;
    location /stub_status {
        stub_status   on;
    }
  }

  server {
    listen 80;

    # Nginx will reject anything not matching /api
    location /api {
      # Reject requests with unsupported HTTP method
      if ($request_method !~ ^(GET|POST|HEAD|OPTIONS|PUT|DELETE)$) {
        return 405;
      }

      # Only requests matching the whitelist expectations will
      # get sent to the application server
      proxy_pass http://app:3000;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection 'upgrade';
      proxy_set_header Host $host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_cache_bypass $http_upgrade;
    }
  }
}
```

> **Note**
>
> nginx-prometheus-exporter가 지표를 스크레이프하도록 구성된 동일한 포트에서 stub_status를 사용하도록 설정해야 합니다. 태스크 정의 예에서 nginx-prometheus-exporter는 포트 8080에서 지표를 스크레이프하도록 구성됩니다.

4. 새 디렉터리의 파일에서 이미지를 구축합니다.

```
docker build -t nginx-reverse-proxy ./path-to-your-directory
```

5. 나중에 사용할 수 있도록 새 이미지를 이미지 리포지토리에 업로드합니다.

Amazon ECS에서 NGINX 및 웹 서버 앱을 실행하는 태스크 정의 생성

다음으로, 태스크 정의를 설정합니다.

이 태스크 정의를 사용하면 NGINX Prometheus 지표를 수집하고 내보낼 수 있습니다. NGINX 컨테이너는 앱의 입력을 추적하고, `nginx.conf`에 설정된 대로 해당 데이터를 포트 8080에 노출합니다. NGINX Prometheus Exporter 컨테이너는 이러한 지표를 스크레이프하며 CloudWatch에서 사용할 수 있도록 포트 9113에 게시합니다.

NGINX 샘플 Amazon ECS 워크로드의 태스크 정의를 설정하려면

1. 다음 내용이 포함된 태스크 정의 JSON 파일을 생성합니다. *your-customized-nginx-iamge*를 사용자 지정 NGINX 이미지의 이미지 URI로 바꾸고 *your-web-server-app-image*를 웹 서버 앱 이미지의 이미지 URI로 바꿉니다.

```
{
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "your-customized-nginx-image",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "links": [
        "app"
      ]
    },
    {
      "name": "app",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 256,
      "essential": true
    },
```

```
    {
      "name": "nginx-prometheus-exporter",
      "image": "docker.io/nginx/nginx-prometheus-exporter:0.8.0",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "command": [
        "-nginx.scrape-uri",
        "http://nginx:8080/stub_status"
    ],
    "links":[
      "nginx"
    ],
      "portMappings":[
        {
          "containerPort": 9113,
          "protocol": "tcp"
        }
      ]
    }
  ],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx-sample-stack"
}
```

2. 다음 명령을 입력하여 태스크 정의를 등록합니다.

```
aws ecs register-task-definition --cli-input-json file://path-to-your-task-definition-json
```

3. 다음 명령을 입력하여 태스크를 실행할 서비스를 생성합니다.

   서비스 이름을 변경해서는 안 됩니다. 태스크를 시작한 서비스의 이름 패턴을 사용해 태스크를 검색하는 구성을 사용하여 CloudWatch 에이전트 서비스를 실행합니다. 예를 들어 CloudWatch 에이전트가 이 명령으로 시작된 태스크를 찾도록 하려면 `sd_service_name_pattern` 값을 `^nginx-service$`로 지정할 수 있습니다. 다음 단원에서 더 자세히 설명합니다.

```
aws ecs create-service \
 --cluster your-cluster-name \
 --service-name nginx-service \
 --task-definition nginx-sample-stack:1 \
```

```
 --desired-count 1
```

NGINX Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트 구성

마지막 단계는 NGINX 지표를 스크레이프하도록 CloudWatch 에이전트를 구성하는 것입니다. 이 예에서 CloudWatch 에이전트는 서비스 이름 패턴과 Exporter가 NGINX에 대한 Prometheus 지표를 노출하는 포트 9113을 통해 태스크를 검색합니다. 태스크를 검색하고 지표를 사용할 수 있게 되면 CloudWatch 에이전트는 수집된 지표를 로그 스트림 nginx-prometheus-exporter에 게시하기 시작합니다.

NGINX 지표를 스크레이프하도록 CloudWatch 에이전트를 구성하려면

1. 다음 명령을 입력하여 필요한 YAML 파일의 최신 버전을 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

2. 텍스트 편집기를 사용하여 파일을 열고 `resource:CWAgentConfigSSMParameter` 섹션에서 `value` 키의 전체 CloudWatch 에이전트 구성을 찾습니다. 그런 다음, `ecs_service_discovery` 섹션에서 다음 `service_name_list_for_tasks` 섹션을 추가합니다.

```
"service_name_list_for_tasks": [
  {
    "sd_job_name": "nginx-prometheus-exporter",
    "sd_metrics_path": "/metrics",
    "sd_metrics_ports": "9113",
    "sd_service_name_pattern": "^nginx-service$"
   }
],
```

3. 동일한 파일의 `metric_declaration` 섹션에 다음 섹션을 추가하여 NGINX 지표를 허용합니다. 이때 기존의 들여쓰기 패턴을 따라야 합니다.

```
{
  "source_labels": ["job"],
  "label_matcher": ".*nginx.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName"]],
```

```
  "metric_selectors": [
    "^nginx_.*$"
  ]
},
```

4. 이 클러스터에 CloudWatch 에이전트를 아직 배포하지 않은 경우 8단계로 건너뜁니다.

   AWS CloudFormation을 사용하여 Amazon ECS 클러스터에 CloudWatch 에이전트를 이미 배포한 경우 다음 명령을 입력하여 변경 세트를 생성할 수 있습니다.

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name nginx-scraping-support
```

5. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

6. 새로 생성한 변경 세트인 nginx-scraping-support를 검토합니다. CWAgentConfigSSMParameter 리소스에 적용된 변경 사항 하나가 표시되어야 합니다. 변경 세트를 실행하고 다음 명령을 입력하여 CloudWatch 에이전트 태스크를 다시 시작합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. 10초 정도 기다린 후 다음 명령을 입력합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

8. 클러스터에 처음으로 Prometheus 지표 수집이 포함된 CloudWatch 에이전트를 설치하는 경우 다음 명령을 입력합니다.

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

NGINX 지표 및 로그 보기

이제 수집 중인 NGINX 지표를 볼 수 있습니다.

샘플 NGINX 워크로드에 대한 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 클러스터가 실행되고 있는 리전에서 왼쪽 탐색 창의 [지표(Metrics)]를 선택합니다. ContainerInsights/Prometheus 네임스페이스를 찾아 지표를 확인합니다.
3. CloudWatch Logs 이벤트를 보려면 탐색 창에서 [로그 그룹(Log groups)]을 선택합니다. 이벤트는 로그 그룹 /aws/containerinsights/***your_cluster_name***/prometheus의 로그 스트림 *nginx-prometheus-exporter*에 있습니다.

Amazon ECS 클러스터의 NGINX Plus 워크로드 샘플

NGINX Plus는 NGINX의 상용 버전입니다. 사용하려면 라이선스가 있어야 합니다. 자세한 내용은 NGINX Plus를 참조하세요.

NGINX Prometheus Exporter는 NGINX 데이터를 Prometheus 지표로 스크레이프하고 노출할 수 있습니다. 이 예에서는 Amazon ECS의 NGINX Plus 역방향 프록시 서비스와 함께 Exporter를 사용합니다.

NGINX Prometheus Exporter에 대한 자세한 내용은 Github의 nginx-prometheus-exporter를 참조하세요. NGINX 역방향 프록시에 대한 자세한 내용은 Github의 ecs-nginx-reverse-proxy를 참조하세요.

Prometheus가 지원되는 CloudWatch 에이전트는 Amazon ECS 클러스터의 서비스 검색 구성을 기반으로 NGINX Plus Prometheus 지표를 스크레이프합니다. 지표를 다른 포트 또는 경로에 노출하도록 NGINX Prometheus Exporter를 구성할 수 있습니다. 포트 또는 경로를 변경하는 경우 CloudWatch 에이전트 구성 파일의 `ecs_service_discovery` 섹션을 업데이트하세요.

Amazon ECS 클러스터의 NGINX Plus 역방향 프록시 샘플 워크로드 설치

다음 단계에 따라 NGINX 역방향 프록시 샘플 워크로드를 설치합니다.

Docker 이미지 생성

NGINX Plus 역방향 프록시 샘플 워크로드의 Docker 이미지를 생성하려면

1. NGINX 역방향 프록시 리포지토리에서 다음 폴더를 다운로드합니다. https://github.com/awslabs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/
2. app 디렉터리를 찾아 해당 디렉터리에서 이미지를 구축합니다.

   ```
   docker build -t web-server-app ./path-to-app-directory
   ```

3. NGINX Plus용 사용자 지정 이미지를 구축합니다. NGINX Plus용 이미지를 구축하기 전에 먼저, `nginx-repo.key`라는 키와 라이선스가 있는 NGINX Plus의 SSL 인증서인 `nginx-repo.crt`를 가져와야 합니다. 디렉터리를 만들어서 그 안에 `nginx-repo.key` 및 `nginx-repo.crt` 파일을 저장합니다.

   방금 만든 디렉터리에서 다음과 같은 두 파일을 생성합니다.

   - 다음 내용이 포함된 샘플 Dockerfile: 이 Docker 파일은 https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-docker/#docker_plus_image에서 제공한 샘플 파일에서 채택한 것입니다. 중요한 변경 사항은 다음 단계에서 생성될 `nginx.conf`라는 별도의 파일을 로드한다는 것입니다.

```
FROM debian:buster-slim

LABEL maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>“

# Define NGINX versions for NGINX Plus and NGINX Plus modules
# Uncomment this block and the versioned nginxPackages block in the main RUN
# instruction to install a specific release
# ENV NGINX_VERSION 21
# ENV NJS_VERSION 0.3.9
# ENV PKG_RELEASE 1~buster

# Download certificate and key from the customer portal (https://cs.nginx.com
 (https://cs.nginx.com/))
# and copy to the build context
COPY nginx-repo.crt /etc/ssl/nginx/
COPY nginx-repo.key /etc/ssl/nginx/
# COPY nginx.conf /etc/ssl/nginx/nginx.conf

RUN set -x \
# Create nginx user/group first, to be consistent throughout Docker variants
&& addgroup --system --gid 101 nginx \
&& adduser --system --disabled-login --ingroup nginx --no-create-home --home /
nonexistent --gecos "nginx user" --shell /bin/false --uid 101 nginx \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y ca-
certificates gnupg1 \
&& \
NGINX_GPGKEY=573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62; \
found=''; \
for server in \
ha.pool.sks-keyservers.net (http://ha.pool.sks-keyservers.net/) \
hkp://keyserver.ubuntu.com:80 \
hkp://p80.pool.sks-keyservers.net:80 \
pgp.mit.edu (http://pgp.mit.edu/) \
; do \
echo "Fetching GPG key $NGINX_GPGKEY from $server"; \
apt-key adv --keyserver "$server" --keyserver-options timeout=10 --recv-keys
 "$NGINX_GPGKEY" && found=yes && break; \
done; \
test -z "$found" && echo >&2 "error: failed to fetch GPG key $NGINX_GPGKEY" &&
 exit 1; \
apt-get remove --purge --auto-remove -y gnupg1 && rm -rf /var/lib/apt/lists/* \
# Install the latest release of NGINX Plus and/or NGINX Plus modules
```

```
# Uncomment individual modules if necessary
# Use versioned packages over defaults to specify a release
&& nginxPackages=" \
nginx-plus \
# nginx-plus=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-xslt \
# nginx-plus-module-xslt=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-geoip \
# nginx-plus-module-geoip=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-image-filter \
# nginx-plus-module-image-filter=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-perl \
# nginx-plus-module-perl=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-njs \
# nginx-plus-module-njs=${NGINX_VERSION}+${NJS_VERSION}-${PKG_RELEASE} \
" \
&& echo "Acquire::https::plus-pkgs.nginx.com::Verify-Peer \"true\";" >> /etc/apt/
apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::Verify-Host \"true\";" >> /etc/apt/
apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::SslCert \"/etc/ssl/nginx/nginx-
repo.crt\";" >> /etc/apt/apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::SslKey \"/etc/ssl/nginx/nginx-
repo.key\";" >> /etc/apt/apt.conf.d/90nginx \
&& printf "deb https://plus-pkgs.nginx.com/debian buster nginx-plus\n" > /etc/
apt/sources.list.d/nginx-plus.list \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y \
$nginxPackages \
gettext-base \
curl \
&& apt-get remove --purge --auto-remove -y && rm -rf /var/lib/apt/lists/* /etc/
apt/sources.list.d/nginx-plus.list \
&& rm -rf /etc/apt/apt.conf.d/90nginx /etc/ssl/nginx

# Forward request logs to Docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log \
&& ln -sf /dev/stderr /var/log/nginx/error.log

COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

STOPSIGNAL SIGTERM
```

```
CMD ["nginx", "-g", "daemon off;"]
```

- https://github.com/awslabs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/nginx에서 수정한 nginx.conf 파일

```
events {
  worker_connections 768;
}

http {
  # Nginx will handle gzip compression of responses from the app server
  gzip on;
  gzip_proxied any;
  gzip_types text/plain application/json;
  gzip_min_length 1000;

  upstream backend {
    zone name 10m;
    server app:3000    weight=2;
    server app2:3000    weight=1;
  }

  server{
    listen 8080;
    location /api {
      api write=on;
    }
  }

  match server_ok {
    status 100-599;
  }

  server {
    listen 80;
    status_zone zone;
    # Nginx will reject anything not matching /api
    location /api {
      # Reject requests with unsupported HTTP method
      if ($request_method !~ ^(GET|POST|HEAD|OPTIONS|PUT|DELETE)$) {
        return 405;
      }
```

```
      # Only requests matching the whitelist expectations will
      # get sent to the application server
      proxy_pass http://backend;
      health_check uri=/lorem-ipsum match=server_ok;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection 'upgrade';
      proxy_set_header Host $host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_cache_bypass $http_upgrade;
    }
  }
}
```

4. 새 디렉터리의 파일에서 이미지를 구축합니다.

```
docker build -t nginx-plus-reverse-proxy ./path-to-your-directory
```

5. 나중에 사용할 수 있도록 새 이미지를 이미지 리포지토리에 업로드합니다.

Amazon ECS에서 NGINX Plus 및 웹 서버 앱을 실행하는 태스크 정의 생성

다음으로, 태스크 정의를 설정합니다.

이 태스크 정의를 사용하면 NGINX Plus Prometheus 지표를 수집하고 내보낼 수 있습니다. NGINX 컨테이너는 앱의 입력을 추적하고, `nginx.conf`에 설정된 대로 해당 데이터를 포트 8080에 노출합니다. NGINX Prometheus Exporter 컨테이너는 이러한 지표를 스크레이프하며 CloudWatch에서 사용할 수 있도록 포트 9113에 게시합니다.

NGINX 샘플 Amazon ECS 워크로드의 태스크 정의를 설정하려면

1. 다음 내용이 포함된 태스크 정의 JSON 파일을 생성합니다. *your-customized-nginx-plus-image*를 사용자 지정 NGINX Plus 이미지의 이미지 URI로 바꾸고 *your-web-server-app-image*를 웹 서버 앱 이미지의 이미지 URI로 바꿉니다.

```
{
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "your-customized-nginx-plus-image",
      "memory": 256,
```

```
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "links": [
        "app",
        "app2"
      ]
    },
    {
      "name": "app",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 128,
      "essential": true
    },
    {
      "name": "app2",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 128,
      "essential": true
    },
    {
      "name": "nginx-prometheus-exporter",
      "image": "docker.io/nginx/nginx-prometheus-exporter:0.8.0",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "command": [
        "-nginx.plus",
        "-nginx.scrape-uri",
         "http://nginx:8080/api"
    ],
    "links":[
      "nginx"
    ],
      "portMappings":[
        {
          "containerPort": 9113,
```

```
          "protocol": "tcp"
        }
      ]
    }
  ],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx-plus-sample-stack"
}
```

2. 태스크 정의를 등록합니다.

```
aws ecs register-task-definition --cli-input-json file://path-to-your-task-definition-json
```

3. 다음 명령을 입력하여 태스크를 실행할 서비스를 생성합니다.

```
aws ecs create-service \
 --cluster your-cluster-name \
 --service-name nginx-plus-service \
 --task-definition nginx-plus-sample-stack:1 \
 --desired-count 1
```

서비스 이름을 변경해서는 안 됩니다. 태스크를 시작한 서비스의 이름 패턴을 사용해 태스크를 검색하는 구성을 사용하여 CloudWatch 에이전트 서비스를 실행합니다. 예를 들어 CloudWatch 에이전트가 이 명령으로 시작된 태스크를 찾도록 하려면 `sd_service_name_pattern` 값을 `^nginx-plus-service$`로 지정할 수 있습니다. 다음 단원에서 더 자세히 설명합니다.

NGINX Plus Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트 구성

마지막 단계는 NGINX 지표를 스크레이프하도록 CloudWatch 에이전트를 구성하는 것입니다. 이 예에서 CloudWatch 에이전트는 서비스 이름 패턴과 Exporter가 NGINX에 대한 Prometheus 지표를 노출하는 포트 9113을 통해 태스크를 검색합니다. 태스크를 검색하고 지표를 사용할 수 있게 되면 CloudWatch 에이전트는 수집된 지표를 로그 스트림 nginx-prometheus-exporter에 게시하기 시작합니다.

NGINX 지표를 스크레이프하도록 CloudWatch 에이전트를 구성하려면

1. 다음 명령을 입력하여 필요한 YAML 파일의 최신 버전을 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-
bridge-host.yaml
```

2. 텍스트 편집기를 사용하여 파일을 열고 `resource:CWAgentConfigSSMParameter` 섹션에서 `value` 키의 전체 CloudWatch 에이전트 구성을 찾습니다. 그런 다음, `ecs_service_discovery` 섹션에서 다음 `service_name_list_for_tasks` 섹션을 추가합니다.

```
"service_name_list_for_tasks": [
  {
    "sd_job_name": "nginx-plus-prometheus-exporter",
    "sd_metrics_path": "/metrics",
    "sd_metrics_ports": "9113",
    "sd_service_name_pattern": "^nginx-plus.*"
   }
],
```

3. 동일한 파일의 `metric_declaration` 섹션에 다음 섹션을 추가하여 NGINX Plus 지표를 허용합니다. 이때 기존의 들여쓰기 패턴을 따라야 합니다.

```
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName"]],
  "metric_selectors": [
    "^nginxplus_connections_accepted$",
    "^nginxplus_connections_active$",
    "^nginxplus_connections_dropped$",
    "^nginxplus_connections_idle$",
    "^nginxplus_http_requests_total$",
    "^nginxplus_ssl_handshakes$",
    "^nginxplus_ssl_handshakes_failed$",
    "^nginxplus_up$",
    "^nginxplus_upstream_server_health_checks_fails$"
  ]
},
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
```

```
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName",
 "upstream"]],
  "metric_selectors": [
    "^nginxplus_upstream_server_response_time$"
  ]
},
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName", "code"]],
  "metric_selectors": [
    "^nginxplus_upstream_server_responses$",
    "^nginxplus_server_zone_responses$"
  ]
},
```

4. 이 클러스터에 CloudWatch 에이전트를 아직 배포하지 않은 경우 8단계로 건너뜁니다.

   AWS CloudFormation을 사용하여 Amazon ECS 클러스터에 CloudWatch 에이전트를 이미 배포한 경우 다음 명령을 입력하여 변경 세트를 생성할 수 있습니다.

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name nginx-plus-scraping-support
```

5. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

6. 새로 생성한 변경 세트인 nginx-plus-scraping-support를 검토합니다. CWAgentConfigSSMParameter 리소스에 적용된 변경 사항 하나가 표시되어야 합니다. 변경 세트를 실행하고 다음 명령을 입력하여 CloudWatch 에이전트 태스크를 다시 시작합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. 10초 정도 기다린 후 다음 명령을 입력합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

8. 클러스터에 처음으로 Prometheus 지표 수집이 포함된 CloudWatch 에이전트를 설치하는 경우 다음 명령을 입력합니다.

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

NGINX Plus 지표 및 로그 보기

이제 수집 중인 NGINX Plus 지표를 볼 수 있습니다.

샘플 NGINX 워크로드에 대한 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 클러스터가 실행되고 있는 리전에서 왼쪽 탐색 창의 [지표(Metrics)]를 선택합니다. ContainerInsights/Prometheus 네임스페이스를 찾아 지표를 확인합니다.
3. CloudWatch Logs 이벤트를 보려면 탐색 창에서 [로그 그룹(Log groups)]을 선택합니다. 이벤트는 로그 그룹 /aws/containerinsights/***your_cluster_name***/prometheus의 로그 스트림 *nginx-plus-prometheus-exporter*에 있습니다.

새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Amazon ECS의 Memcached

이 튜토리얼에서는 EC2 시작 유형의 Amazon ECS 클러스터에서 샘플 Memcached 애플리케이션의 Prometheus 지표를 스크레이프하는 실습을 소개합니다. CloudWatch 에이전트는 ECS 태스크 정의 기반 서비스 검색을 통해 Memcached Prometheus Exporter 대상을 자동 검색합니다.

Memcached는 범용 분산 메모리 캐싱 시스템으로, 흔히 외부 데이터 소스(예: 데이터베이스 또는 API)를 읽어야 하는 횟수를 줄이기 위해 RAM에 데이터 및 객체를 캐싱하여 동적 데이터베이스 기반 웹 사이트의 속도를 높이는 데 사용됩니다. 자세한 내용은 Memcached란 무엇입니까?를 참조하세요.

memchached_exporter(Apache 라이선스 2.0)는 공식 Prometheus Exporter 중 하나입니다. 기본적으로 memcache_exporter는 `/metrics.`의 포트 0.0.0.0:9150에서 제공됩니다.

이 튜토리얼에서는 다음과 같은 두 Docker Hub 리포지토리의 Docker 이미지를 사용합니다.

- Memcached
- prom/memcached-exporter

사전 조건

Amazon ECS의 Prometheus 워크로드 샘플에서 지표를 수집하려면 클러스터에서 Container Insights를 실행 중이어야 합니다. Container Insights 설치에 대한 자세한 내용은 Amazon ECS에서 Container Insights 설정 단원을 참조하세요.

주제

- Amazon ECS EC2 클러스터 환경 변수 설정
- 샘플 Memcached 워크로드 설치
- Memcached Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트 구성

- Memcached 지표 보기

Amazon ECS EC2 클러스터 환경 변수 설정

Amazon ECS EC2 클러스터 환경 변수를 설정하려면

1. 아직 설치하지 않은 경우 Amazon ECS CLI를 설치합니다. 자세한 내용은 Amazon ECS CLI 설치 단원을 참조하세요.
2. 새 Amazon ECS 클러스터 이름 및 리전을 설정합니다. 예:

```
ECS_CLUSTER_NAME=ecs-ec2-memcached-tutorial
AWS_DEFAULT_REGION=ca-central-1
```

3. (선택 사항) 샘플 Memcached 워크로드 및 CloudWatch 에이전트를 설치하려는 EC2 시작 유형의 Amazon ECS 클러스터가 아직 없는 경우 다음 명령을 입력하여 클러스터를 생성할 수 있습니다.

```
ecs-cli up --capability-iam --size 1 \
--instance-type t3.medium \
--cluster $ECS_CLUSTER_NAME \
--region $AWS_REGION
```

이 명령의 예상 결과는 다음과 같습니다.

```
WARN[0000] You will not be able to SSH into your EC2 instances without a key pair.
INFO[0000] Using recommended Amazon Linux 2 AMI with ECS Agent 1.44.4 and Docker
 version 19.03.6-ce
INFO[0001] Created cluster                               cluster=ecs-ec2-memcached-
tutorial region=ca-central-1
INFO[0002] Waiting for your cluster resources to be created...
INFO[0002] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
INFO[0063] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
INFO[0124] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
VPC created: vpc-xxxxxxxxxxxxxxxxx
Security Group created: sg-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxx
Cluster creation succeeded.
```

샘플 Memcached 워크로드 설치

Prometheus 지표를 노출하는 샘플 Memcached 워크로드를 설치하려면

1. 다음 명령을 입력하여 Memcached AWS CloudFormation 템플릿을 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/sample_traffic/memcached/memcached-traffic-sample.yaml
```

2. 다음 명령을 입력하여 Memcached용으로 생성할 IAM 역할 이름을 설정합니다.

```
MEMCACHED_ECS_TASK_ROLE_NAME=memcached-prometheus-demo-ecs-task-role-name
MEMCACHED_ECS_EXECUTION_ROLE_NAME=memcached-prometheus-demo-ecs-execution-role-name
```

3. 다음 명령을 입력하여 샘플 Memcached 워크로드를 설치합니다. 이 샘플은 `host` 네트워크 모드에서 워크로드를 설치합니다.

```
MEMCACHED_ECS_NETWORK_MODE=host

aws cloudformation create-stack --stack-name Memcached-Prometheus-Demo-ECS-
$ECS_CLUSTER_NAME-EC2-$MEMCACHED_ECS_NETWORK_MODE \
    --template-body file://memcached-traffic-sample.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=ECSNetworkMode,ParameterValue=
$MEMCACHED_ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=
$MEMCACHED_ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$MEMCACHED_ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

AWS CloudFormation 스택은 다음과 같은 네 개의 리소스를 생성합니다.

- ECS 태스크 역할 1개
- ECS 태스크 실행 역할 1개
- Memcached 태스크 정의 1개
- Memcached 서비스 1개

Memcached 태스크 정의에서는 다음과 같이 두 컨테이너가 정의됩니다.

- 기본 컨테이너는 단순한 Memcached 애플리케이션을 실행하고 액세스를 위해 포트 11211을 엽니다.
- 다른 컨테이너는 Redis Exporter 프로세스를 실행하여 포트 9150에서 Prometheus 지표를 노출합니다. 이는 CloudWatch 에이전트가 검색하고 스크레이프할 컨테이너입니다.

Memcached Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트 구성

Memcached Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트를 구성하려면

1. 다음 명령을 입력하여 최신 버전의 `cwagent-ecs-prometheus-metric-for-awsvpc.yaml`을 다운로드합니다.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml
   ```

2. 텍스트 편집기를 사용하여 파일을 열고 `resource:CWAgentConfigSSMParameter` 섹션에서 `value` 키 뒤에 있는 전체 CloudWatch 에이전트 구성을 찾습니다.

   그런 다음, `ecs_service_discovery` 섹션에서 다음 구성을 `task_definition_list` 섹션에 추가합니다.

   ```
   {
       "sd_job_name": "ecs-memcached",
       "sd_metrics_ports": "9150",
       "sd_task_definition_arn_pattern": ".*:task-definition/memcached-prometheus-demo.*:[0-9]+"
   },
   ```

   `metric_declaration` 섹션의 경우 기본 설정은 어느 Memcached 지표도 허용하지 않습니다. Memcached 지표를 허용하려면 다음 섹션을 추가합니다. 이때 기존의 들여쓰기 패턴을 따라야 합니다.

   ```
   {
     "source_labels": ["container_name"],
     "label_matcher": "memcached-exporter-.*",
     "dimensions": [["ClusterName", "TaskDefinitionFamily"]],
   ```

```
  "metric_selectors": [
    "^memcached_current_(bytes|items|connections)$",
    "^memcached_items_(reclaimed|evicted)_total$",
    "^memcached_(written|read)_bytes_total$",
    "^memcached_limit_bytes$",
    "^memcached_commands_total$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "memcached-exporter-.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily","status","command"],
 ["ClusterName", "TaskDefinitionFamily","command"]],
  "metric_selectors": [
    "^memcached_commands_total$"
  ]
},
```

3. AWS CloudFormation에서 Amazon ECS 클러스터에 CloudWatch 에이전트를 이미 배포한 경우 다음 명령을 입력하여 변경 세트를 생성할 수 있습니다.

```
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name memcached-scraping-support
```

4. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
5. 새로 생성한 변경 세트인 `memcached-scraping-support`를 검토합니다. `CWAgentConfigSSMParameter` 리소스에 적용된 변경 사항 하나가 표시되어야 합니다. 변경 세트를 실행하고 다음 명령을 입력하여 CloudWatch 에이전트 태스크를 다시 시작합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

6. 10초 정도 기다린 후 다음 명령을 입력합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. 클러스터에 처음으로 Prometheus 지표 수집이 포함된 CloudWatch 에이전트를 설치하는 경우 다음 명령을 입력합니다.

```
ECS_NETWORK_MODEE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

Memcached 지표 보기

이 튜토리얼에서는 CloudWatch의 ECS/ContainerInsights/Prometheus 네임스페이스에 다음 지표를 전송합니다. CloudWatch 콘솔을 사용하여 해당 네임스페이스의 지표를 볼 수 있습니다.

지표 이름	측정기준	
memcached_current_items	ClusterName , TaskDefinitionFamily	
memcached_current_connections	ClusterName , TaskDefinitionFamily	
memcached_limit_bytes	ClusterName , TaskDefinitionFamily	
memcached_current_bytes	ClusterName , TaskDefinitionFamily	
memcached_written_bytes_total	ClusterName , TaskDefinitionFamily	
memcached_read_bytes_total	ClusterName , TaskDefinitionFamily	
memcached_items_evicted_total	ClusterName , TaskDefinitionFamily	
memcached_items_reclaimed_total	ClusterName , TaskDefinitionFamily	
memcached_commands_total	ClusterName , TaskDefinitionFamily ClusterName , TaskDefinitionFamily, 명령 ClusterName , TaskDefinitionFamily, 상태, 명령	

> **Note**
>
> [명령(command)] 측정기준의 값은 `delete`, `get`, `cas`, `set`, `decr`, `touch`, `incr` 또는 `flush`일 수 있습니다.
> [상태(status)] 측정기준의 값은 `hit`, `miss` 또는 `badval`일 수 있습니다.

또한 Memcached Prometheus 지표에 대한 CloudWatch 대시보드를 생성할 수도 있습니다.

Memcached Prometheus 지표에 대한 대시보드를 생성하려면

1. 환경 변수를 만들어서 아래의 값을 배포와 일치하도록 바꿉니다.

```
DASHBOARD_NAME=your_memcached_cw_dashboard_name
ECS_TASK_DEF_FAMILY=memcached-prometheus-demo-$ECS_CLUSTER_NAME-EC2-
$MEMCACHED_ECS_NETWORK_MOD
```

2. 다음 명령을 입력하여 대시보드를 생성합니다.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/ecs-task-definition-templates/deployment-mode/
replica-service/cwagent-prometheus/sample_cloudwatch_dashboards/memcached/
cw_dashboard_memcached.json \
| sed "s/{{YOUR_AWS_REGION}}/$AWS_REGION/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/$ECS_CLUSTER_NAME/g" \
| sed "s/{{YOUR_TASK_DEF_FAMILY}}/$ECS_TASK_DEF_FAMILY/g" \
| xargs -0 aws cloudwatch put-dashboard --dashboard-name ${DASHBOARD_NAME} --region
 $AWS_REGION --dashboard-body
```

Amazon ECS Fargate에서 Redis Prometheus 지표를 스크레이프하기 위한 튜토리얼

이 튜토리얼에서는 Amazon ECS Fargate 클러스터에서 샘플 Redis 애플리케이션의 Prometheus 지표를 스크레이프하는 실습을 소개합니다. CloudWatch 에이전트는 컨테이너의 Docker 레이블을 기반으로 하는 Prometheus 지표 지원을 통해 Redis Prometheus Exporter 대상을 자동 검색합니다.

Redis(https://redis.io/)는 데이터베이스, 캐시 및 메시지 브로커로 사용되는 오픈 소스(BSD 라이선스), 인 메모리 구조 데이터 스토어입니다. 자세한 내용은 redis를 참조하세요.

redis_exporter(MIT License 라이선스)는 지정된 포트(기본값: 0.0.0.0:9121)에서 Redis Prometheus 지표를 노출하는 데 사용됩니다. 자세한 내용은 redis_exporter를 참조하세요.

이 튜토리얼에서는 다음과 같은 두 Docker Hub 리포지토리의 Docker 이미지를 사용합니다.

- redis
- redis_exporter

사전 조건

Amazon ECS의 Prometheus 워크로드 샘플에서 지표를 수집하려면 클러스터에서 Container Insights를 실행 중이어야 합니다. Container Insights 설치에 대한 자세한 내용은 Amazon ECS에서 Container Insights 설정 단원을 참조하세요.

주제

- Amazon ECS Fargate 클러스터 환경 변수 설정
- Amazon ECS Fargate 클러스터의 네트워크 환경 변수 설정
- 샘플 Redis 워크로드 설치
- Redis Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트 구성
- Redis 지표 보기

Amazon ECS Fargate 클러스터 환경 변수 설정

Amazon ECS Fargate 클러스터 환경 변수를 설정하려면

1. 아직 설치하지 않은 경우 Amazon ECS CLI를 설치합니다. 자세한 내용은 Amazon ECS CLI 설치 단원을 참조하세요.
2. 새 Amazon ECS 클러스터 이름 및 리전을 설정합니다. 예:

```
ECS_CLUSTER_NAME=ecs-fargate-redis-tutorial
AWS_DEFAULT_REGION=ca-central-1
```

3. (선택 사항) 샘플 Redis 워크로드 및 CloudWatch 에이전트를 설치하려는 Amazon ECS Fargate 클러스터가 아직 없는 경우 다음 명령을 입력하여 클러스터를 생성할 수 있습니다.

```
ecs-cli up --capability-iam \
--cluster $ECS_CLUSTER_NAME \
--launch-type FARGATE \
--region $AWS_DEFAULT_REGION
```

이 명령의 예상 결과는 다음과 같습니다.

```
INFO[0000] Created cluster   cluster=ecs-fargate-redis-tutorial region=ca-central-1
INFO[0001] Waiting for your cluster resources to be created...
INFO[0001] Cloudformation stack status   stackStatus=CREATE_IN_PROGRESS
VPC created: vpc-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxx
Cluster creation succeeded.
```

Amazon ECS Fargate 클러스터의 네트워크 환경 변수 설정

Amazon ECS Fargate 클러스터의 네트워크 환경 변수를 설정하려면

1. Amazon ECS 클러스터의 VPC 및 서브넷 ID를 설정합니다. 이전 절차에서 새 클러스터를 생성한 경우 마지막 명령의 결과에 다음과 같은 값이 표시됩니다. 그렇지 않으면 Redis와 함께 사용할 기존 클러스터의 ID를 사용합니다.

   ```
   ECS_CLUSTER_VPC=vpc-xxxxxxxxxxxxxxxxx
   ECS_CLUSTER_SUBNET_1=subnet-xxxxxxxxxxxxxxxxx
   ECS_CLUSTER_SUBNET_2=subnet-xxxxxxxxxxxxxxxxx
   ```

2. 이 튜토리얼에서는 Amazon ECS 클러스터 VPC의 기본 보안 그룹에 Redis 애플리케이션 및 CloudWatch 에이전트를 설치합니다. 기본 보안 그룹은 동일한 보안 그룹 내의 모든 네트워크 연결을 허용하므로 CloudWatch 에이전트가 Redis 컨테이너에 노출된 Prometheus 지표를 스크레이프할 수 있습니다. 실제 프로덕션 환경에서는 Redis 애플리케이션 및 CloudWatch 에이전트의 전용 보안 그룹을 생성하고 이에 대한 사용자 지정 권한을 설정할 수 있습니다.

   다음 명령을 입력하여 기본 보안 그룹 ID를 가져옵니다.

   ```
   aws ec2 describe-security-groups \
   --filters Name=vpc-id,Values=$ECS_CLUSTER_VPC  \
   --region $AWS_DEFAULT_REGION
   ```

   그다음에는 다음 명령을 입력하여 Fargate 클러스터 기본 보안 그룹 변수를 설정하고 *my-default-security-group*을 이전의 명령에서 찾은 값으로 바꿉니다.

   ```
   ECS_CLUSTER_SECURITY_GROUP=my-default-security-group
   ```

샘플 Redis 워크로드 설치

Prometheus 지표를 노출하는 샘플 Redis 워크로드를 설치하려면

1. 다음 명령을 입력하여 Redis AWS CloudFormation 템플릿을 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/sample_traffic/redis/redis-traffic-sample.yaml
```

2. 다음 명령을 입력하여 Redis용으로 생성할 IAM 역할 이름을 설정합니다.

```
REDIS_ECS_TASK_ROLE_NAME=redis-prometheus-demo-ecs-task-role-name
REDIS_ECS_EXECUTION_ROLE_NAME=redis-prometheus-demo-ecs-execution-role-name
```

3. 다음 명령을 입력하여 샘플 Redis 워크로드를 설치합니다.

```
aws cloudformation create-stack --stack-name Redis-Prometheus-Demo-ECS-
$ECS_CLUSTER_NAME-fargate-awsvpc \
    --template-body file://redis-traffic-sample.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                 ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET_1 \
                 ParameterKey=TaskRoleName,ParameterValue=$REDIS_ECS_TASK_ROLE_NAME
 \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$REDIS_ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_DEFAULT_REGION
```

AWS CloudFormation 스택은 다음과 같은 네 개의 리소스를 생성합니다.

- ECS 태스크 역할 1개
- ECS 태스크 실행 역할 1개
- Redis 태스크 정의 1개
- Redis 서비스 1개

Redis 태스크 정의에서는 다음과 같이 두 컨테이너가 정의됩니다.

- 기본 컨테이너는 단순한 Redis 애플리케이션을 실행하고 액세스를 위해 포트 6379를 엽니다.
- 다른 컨테이너는 Redis Exporter 프로세스를 실행하여 포트 9121에서 Prometheus 지표를 노출합니다. 이는 CloudWatch 에이전트가 검색하고 스크레이프할 컨테이너입니다. 다음 Docker 레이블을 정의하여 CloudWatch 에이전트가 해당 레이블을 기반으로 이 컨테이너를 검색할 수 있도록 합니다.

```
ECS_PROMETHEUS_EXPORTER_PORT: 9121
```

Redis Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트 구성

Redis Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트를 구성하려면

1. 다음 명령을 입력하여 최신 버전의 `cwagent-ecs-prometheus-metric-for-awsvpc.yaml`을 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml
```

2. 텍스트 편집기를 사용하여 파일을 열고 `resource:CWAgentConfigSSMParameter` 섹션에서 `value` 키 뒤에 있는 전체 CloudWatch 에이전트 구성을 찾습니다.

   여기에 나와 있는 `ecs_service_discovery` 섹션에서는 Redis ECS 태스크 정의에서 정의한 Docker 레이블과 일치하는 `ECS_PROMETHEUS_EXPORTER_PORT`를 기반으로 하는 기본 설정으로 `docker_label` 기반 서비스 검색이 사용 설정됩니다. 따라서 이 섹션에서는 아무것도 변경할 필요가 없습니다.

```
ecs_service_discovery": {
  "sd_frequency": "1m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
*  "docker_label": {
  },*
  ...
```

   `metric_declaration` 섹션의 경우 기본 설정은 어느 Redis 지표도 허용하지 않습니다. Redis 지표를 허용하려면 다음 섹션을 추가합니다. 이때 기존의 들여쓰기 패턴을 따라야 합니다.

```
{
  "source_labels": ["container_name"],
```

```
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily"]],
  "metric_selectors": [
    "^redis_net_(in|out)put_bytes_total$",
    "^redis_(expired|evicted)_keys_total$",
    "^redis_keyspace_(hits|misses)_total$",
    "^redis_memory_used_bytes$",
    "^redis_connected_clients$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily","cmd"]],
  "metric_selectors": [
    "^redis_commands_total$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily","db"]],
  "metric_selectors": [
    "^redis_db_keys$"
  ]
},
```

3. AWS CloudFormation에서 Amazon ECS 클러스터에 CloudWatch 에이전트를 이미 배포한 경우 다음 명령을 입력하여 변경 세트를 생성할 수 있습니다.

```
ECS_LAUNCH_TYPE=FARGATE
CREATE_IAM_ROLES=True
ECS_CLUSTER_SUBNET=$ECS_CLUSTER_SUBNET_1
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
$ECS_CLUSTER_NAME-$ECS_LAUNCH_TYPE-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSLaunchType,ParameterValue=$ECS_LAUNCH_TYPE \
```

```
                ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET \
                ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --change-set-name redis-scraping-support
```

4. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
5. 새로 생성한 변경 세트인 `redis-scraping-support`를 검토합니다. `CWAgentConfigSSMParameter` 리소스에 적용된 변경 사항 하나가 표시되어야 합니다. 변경 세트를 실행하고 다음 명령을 입력하여 CloudWatch 에이전트 태스크를 다시 시작합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-$ECS_LAUNCH_TYPE-awsvpc \
--region ${AWS_DEFAULT_REGION}
```

6. 10초 정도 기다린 후 다음 명령을 입력합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-$ECS_LAUNCH_TYPE-awsvpc \
--region ${AWS_DEFAULT_REGION}
```

7. 클러스터에 처음으로 Prometheus 지표 수집이 포함된 CloudWatch 에이전트를 설치하는 경우 다음 명령을 입력합니다.

```
ECS_LAUNCH_TYPE=FARGATE
CREATE_IAM_ROLES=True
ECS_CLUSTER_SUBNET=$ECS_CLUSTER_SUBNET_1
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
$ECS_CLUSTER_NAME-$ECS_LAUNCH_TYPE-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSLaunchType,ParameterValue=$ECS_LAUNCH_TYPE \
```

```
                ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET \
                ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION}
```

Redis 지표 보기

이 튜토리얼에서는 CloudWatch의 ECS/ContainerInsights/Prometheus 네임스페이스에 다음 지표를 전송합니다. CloudWatch 콘솔을 사용하여 해당 네임스페이스의 지표를 볼 수 있습니다.

지표 이름	측정기준	
redis_net_input_bytes_total	ClusterName, TaskDefinitionFamily	
redis_net_output_bytes_total	ClusterName, TaskDefinitionFamily	
redis_expired_keys_total	ClusterName, TaskDefinitionFamily	
redis_evicted_keys_total	ClusterName, TaskDefinitionFamily	
redis_keyspace_hits_total	ClusterName, TaskDefinitionFamily	
redis_keyspace_misses_total	ClusterName, TaskDefinitionFamily	

지표 이름	측정기준	
redis_memory_used_bytes	ClusterName, TaskDefinitionFamily	
redis_connected_clients	ClusterName, TaskDefinitionFamily	
redis_commands_total	ClusterName , TaskDefinitionFamily , cmd	
redis_db_keys	ClusterName , TaskDefinitionFamily , db	

> **Note**
>
> [cmd] 측정기준의 값은 append, client, command, config, dbsize, flushall, get, incr, info, latency 또는 slowlog일 수 있습니다.
> [db] 측정기준의 값은 db0~db15일 수 있습니다.

또한 Redis Prometheus 지표에 대한 CloudWatch 대시보드를 생성할 수도 있습니다.

Redis Prometheus 지표에 대한 대시보드를 생성하려면

1. 환경 변수를 만들어서 아래의 값을 배포와 일치하도록 바꿉니다.

```
DASHBOARD_NAME=your_cw_dashboard_name
ECS_TASK_DEF_FAMILY=redis-prometheus-demo-$ECS_CLUSTER_NAME-fargate-awsvpc
```

2. 다음 명령을 입력하여 대시보드를 생성합니다.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/redis/cw_dashboard_redis.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| sed "s/{{YOUR_NAMESPACE}}/${NAMESPACE}/g" \
```

Amazon EKS 및 Kubernetes 클러스터에서 Prometheus 지표 수집 설정 및 구성

Amazon EKS 또는 Kubernetes를 실행하는 클러스터에서 Prometheus 지표를 수집하려면 CloudWatch 에이전트를 수집기로 사용하거나 AWS Distro for OpenTelemetry Collector를 사용하면 됩니다. AWS Distro for OpenTelemetry Collector 사용에 대한 자세한 내용은 https://aws-otel.github.io/docs/getting-started/container-insights/eks-prometheus를 참조하세요.

다음 단원에서는 CloudWatch 에이전트를 사용하여 Prometheus 지표를 수집하는 방법을 설명합니다. Amazon EKS 또는 Kubernetes를 실행 중인 클러스터에 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설치하는 방법과 추가 대상을 스크레이프하도록 에이전트를 구성하는 방법을 설명합니다. 또한 Prometheus 모니터링을 통해 테스트하는 데 사용할 샘플 워크로드를 설정하기 위한 선택적 튜토리얼도 제공합니다.

주제

Amazon EKS 및 Kubernetes 클러스터에 Prometheus 지표 수집과 함께 CloudWatch 에이전트 설치

이 단원에서는 Amazon EKS 또는 Kubernetes를 실행 중인 클러스터에서 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설정하는 방법을 설명합니다. 이렇게 에이전트를 설정하면 에이전트가 해당 클러스터에서 실행 중인 다음 워크로드에 대한 지표를 자동으로 스크레이프하고 가져옵니다.

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy
- Fluent Bit

추가 Prometheus 워크로드 및 소스를 스크레이프하고 가져오도록 에이전트를 구성할 수도 있습니다.

다음 단계에 따라 Prometheus 지표 수집용 CloudWatch 에이전트를 설치하기 전에 먼저, Amazon EKS에서 클러스터가 실행 중이거나 Amazon EC2 인스턴스에서 Kubernetes 클러스터가 실행 중이어야 합니다.

VPC 보안 그룹 요구 사항

Prometheus 워크로드의 보안 그룹 수신 규칙은 프라이빗 IP로 Prometheus 지표를 스크레이프하기 위해 CloudWatch 에이전트에 대한 Prometheus 포트를 열어야 합니다.

CloudWatch 에이전트의 보안 그룹 송신 규칙은 CloudWatch 에이전트가 프라이빗 IP로 Prometheus 워크로드의 포트에 연결할 수 있도록 허용해야 합니다.

주제

- Amazon EKS 및 Kubernetes 클러스터에 Prometheus 지표 수집과 함께 CloudWatch 에이전트 설치
- 추가 Prometheus 소스 스크레이핑 및 해당 지표 가져오기
- (선택 사항) Prometheus 지표 테스트를 위한 컨테이너화된 Amazon EKS 워크로드 샘플 설정

Amazon EKS 및 Kubernetes 클러스터에 Prometheus 지표 수집과 함께 CloudWatch 에이전트 설치

이 단원에서는 Amazon EKS 또는 Kubernetes를 실행 중인 클러스터에서 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설정하는 방법을 설명합니다. 이렇게 에이전트를 설정하면 에이전트가 해당 클러스터에서 실행 중인 다음 워크로드에 대한 지표를 자동으로 스크레이프하고 가져옵니다.

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy
- Fluent Bit

추가 Prometheus 워크로드 및 소스를 스크레이프하고 가져오도록 에이전트를 구성할 수도 있습니다.

다음 단계에 따라 Prometheus 지표 수집용 CloudWatch 에이전트를 설치하기 전에 먼저, Amazon EKS에서 클러스터가 실행 중이거나 Amazon EC2 인스턴스에서 Kubernetes 클러스터가 실행 중이어야 합니다.

VPC 보안 그룹 요구 사항

Prometheus 워크로드의 보안 그룹 수신 규칙은 프라이빗 IP로 Prometheus 지표를 스크레이프하기 위해 CloudWatch 에이전트에 대한 Prometheus 포트를 열어야 합니다.

CloudWatch 에이전트의 보안 그룹 송신 규칙은 CloudWatch 에이전트가 프라이빗 IP로 Prometheus 워크로드의 포트에 연결할 수 있도록 허용해야 합니다.

주제

- IAM 역할 설정
- Prometheus 지표를 수집하기 위한 CloudWatch 에이전트 설치

IAM 역할 설정

첫 번째 단계는 클러스터에 필요한 IAM 역할을 설정하는 것입니다. 다음과 같은 두 가지 방법으로 설정할 수 있습니다.

- '서비스 역할'이라고도 하는 서비스 계정의 IAM 역할을 설정합니다. 이 방법은 EC2 시작 유형과 Fargate 시작 유형 모두에 적용됩니다.
- 클러스터에 사용되는 IAM 역할에 IAM 정책을 추가합니다. 이는 EC2 시작 유형에만 적용됩니다.

서비스 역할 설정(EC2 시작 유형 및 Fargate 시작 유형)

서비스 역할을 설정하려면 다음 명령을 입력합니다. *MyCluster*를 클러스터 이름으로 바꿉니다.

```
eksctl create iamserviceaccount \
 --name cwagent-prometheus \
--namespace amazon-cloudwatch \
 --cluster MyCluster \
--attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
--approve \
--override-existing-serviceaccounts
```

클러스터의 IAM 역할에 정책 추가(EC2 시작 유형만 해당)

Prometheus 지원을 위해 클러스터에서 IAM 정책을 설정하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 Instances(인스턴스)를 선택합니다.
3. 클러스터에 대한 IAM 역할 이름의 접두사를 찾아야 합니다. 이렇게 하려면 클러스터에 있는 인스턴스 이름 옆의 확인란을 선택하고 작업, 인스턴스 설정, IAM 역할 연결/바꾸기를 선택합니다. 그런 다음, `eksctl-dev303-workshop-nodegroup`과 같은 IAM 역할의 접두사를 복사합니다.
4. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
5. 탐색 창에서 역할을 선택합니다.
6. 검색 상자를 사용하여 이 절차의 앞부분에서 복사한 접두사를 찾은 다음 해당 역할을 선택합니다.

7. 정책 연결을 선택합니다.
8. 검색 상자를 사용하여 CloudWatchAgentServerPolicy를 찾습니다. CloudWatchAgentServerPolicy 옆의 확인란을 선택하고 정책 연결을 선택합니다.

Prometheus 지표를 수집하기 위한 CloudWatch 에이전트 설치

지표를 수집하려면 클러스터에 CloudWatch 에이전트를 설치해야 합니다. Amazon EKS 클러스터와 Kubernetes 클러스터의 에이전트 설치 방법은 다릅니다.

Prometheus가 지원되는 CloudWatch 에이전트의 이전 버전 삭제

클러스터에 Prometheus가 지원되는 CloudWatch 에이전트 버전을 이미 설치한 경우 다음 명령을 입력하여 해당 버전을 삭제해야 합니다. 이는 Prometheus가 지원되는 이전 버전의 에이전트에만 필요합니다. Prometheus가 지원되지 않는 Container Insights를 사용하는 CloudWatch 에이전트는 삭제할 필요가 없습니다.

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

EC2 시작 유형의 Amazon EKS 클러스터에 CloudWatch 에이전트 설치

Amazon EKS 클러스터에 Prometheus가 지원되는 CloudWatch 에이전트를 설치하려면 다음 단계를 따릅니다.

Amazon EKS 클러스터에 Prometheus가 지원되는 CloudWatch 에이전트를 설치하려면

1. 다음 명령을 입력하여 `amazon-cloudwatch` 네임스페이스가 이미 생성되었는지 확인합니다.

   ```
   kubectl get namespace
   ```

2. `amazon-cloudwatch`가 결과에 표시되지 않으면 다음 명령을 입력하여 생성합니다.

   ```
   kubectl create namespace amazon-cloudwatch
   ```

3. 기본 구성으로 에이전트를 배포하고 에이전트가 설치된 AWS 리전으로 데이터를 전송하도록 하려면 다음 명령을 입력합니다.

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
   container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
   service/cwagent-prometheus/prometheus-eks.yaml
   ```

에이전트가 대신 다른 리전으로 데이터를 전송하도록 하려면 다음 단계를 따르세요.

a. 다음 명령을 입력하여 에이전트에 대한 YAML 파일을 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
service/cwagent-prometheus/prometheus-eks.yaml
```

b. 텍스트 편집기로 파일을 열고 파일의 `cwagentconfig.json` 블록을 검색합니다.

c. 강조 표시된 선을 추가하여 원하는 리전을 지정합니다.

```
cwagentconfig.json: |
    {
      "agent": {
        "region": "us-east-2"
      },
      "logs": { ...
```

d. 파일을 저장하고, 업데이트된 파일을 사용하여 에이전트를 배포합니다.

```
kubectl apply -f prometheus-eks.yaml
```

Fargate 시작 유형의 Amazon EKS 클러스터에 CloudWatch 에이전트 설치

Fargate 시작 유형의 Amazon EKS 클러스터에 Prometheus가 지원되는 CloudWatch 에이전트를 설치하려면 다음 단계를 따릅니다.

Fargate 시작 유형의 Amazon EKS 클러스터에 Prometheus가 지원되는 CloudWatch 에이전트를 설치하려면

1. 다음 명령을 입력하여 CloudWatch 에이전트의 Fargate 프로파일을 생성하여 클러스터 내에서 실행될 수 있도록 합니다. *MyCluster*를 클러스터 이름으로 바꿉니다.

```
eksctl create fargateprofile --cluster MyCluster \
--name amazon-cloudwatch \
--namespace amazon-cloudwatch
```

2. CloudWatch 에이전트를 설치하려면 다음 명령을 입력합니다. *MyCluster*를 클러스터 이름으로 바꿉니다. 이 이름은 에이전트가 수집한 로그 이벤트를 저장하는 로그 그룹 이름에 사용되며, 에이전트에서 수집한 지표에 대한 측정기준으로도 사용됩니다.

   *region*을 지표를 전송할 리전의 이름으로 바꿉니다. 예를 들면 `us-west-1`입니다.

   ```
   curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml |
   sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" |
   kubectl apply -f -
   ```

Kubernetes 클러스터에 CloudWatch 에이전트 설치

Kubernetes를 실행하는 클러스터에 Prometheus가 지원되는 CloudWatch 에이전트를 설치하려면 다음 명령을 입력합니다.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml |
sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" |
kubectl apply -f -
```

*MyCluster*를 클러스터 이름으로 바꿉니다. 이 이름은 에이전트가 수집한 로그 이벤트를 저장하는 로그 그룹 이름에 사용되며, 에이전트에서 수집한 지표에 대한 측정기준으로도 사용됩니다.

*##*을 지표를 전송할 AWS 리전의 이름으로 바꿉니다. 예를 들면 **us-west-1**입니다.

에이전트가 실행 중인지 확인

Amazon EKS 클러스터와 Kubernetes 클러스터 모두에서 다음 명령을 입력하여 에이전트가 실행 중인지 확인할 수 있습니다.

```
kubectl get pod -l "app=cwagent-prometheus" -n amazon-cloudwatch
```

결과에 `Running` 상태의 단일 CloudWatch 에이전트 포드가 포함되어 있다면 에이전트가 실행 중이며 Prometheus 지표를 수집하고 있는 것입니다. 기본적으로 CloudWatch 에이전트는 1분마다 App Mesh, NGINX, Memcached, Java/JMX, HAProxy에 대한 지표를 수집합니다. 지표에 대한 자세한 내용은 CloudWatch 에이전트가 수집하는 Prometheus 지표 단원을 참조하세요. CloudWatch에서 Prometheus 지표를 확인하는 방법에 대한 지침은 Prometheus 지표 보기 단원을 참조하세요.

또한 다른 Prometheus Exporter에서 지표를 수집하도록 CloudWatch 에이전트를 구성할 수도 있습니다. 자세한 내용은 추가 Prometheus 소스 스크레이핑 및 해당 지표 가져오기 단원을 참조하십시오.

추가 Prometheus 소스 스크레이핑 및 해당 지표 가져오기

Prometheus 모니터링이 포함된 CloudWatch 에이전트는 Prometheus 지표를 스크레이프하는 데 두 가지 구성이 필요합니다. 하나는 Prometheus 설명서의 <scrape_config>에 설명된 표준 Prometheus 구성을 위한 것입니다. 다른 하나는 CloudWatch 에이전트 구성을 위한 것입니다.

Amazon EKS 클러스터에서 구성은 `prometheus-eks.yaml`(EC2 시작 유형의 경우) 또는 `prometheus-eks-fargate.yaml`(Fargate 시작 유형의 경우)에 다음과 같은 두 config 맵으로 정의됩니다.

- `name: prometheus-config` 섹션에는 Prometheus 스크레이핑 설정이 포함되어 있습니다.
- `name: prometheus-cwagentconfig` 섹션에는 CloudWatch 에이전트에 대한 구성이 포함되어 있습니다. 이 섹션을 사용하여 CloudWatch가 Prometheus 지표를 수집하는 방법을 구성할 수 있습니다. 예를 들어 CloudWatch에 가져올 지표를 지정하고 해당 지표의 측정기준을 정의합니다.

Amazon EC2 인스턴스에서 실행되는 Kubernetes 클러스터에서 구성은 `prometheus-k8s.yaml` YAML 파일에 다음과 같은 두 config 맵으로 정의됩니다.

- `name: prometheus-config` 섹션에는 Prometheus 스크레이핑 설정이 포함되어 있습니다.
- `name: prometheus-cwagentconfig` 섹션에는 CloudWatch 에이전트에 대한 구성이 포함되어 있습니다.

추가 Prometheus 지표 소스를 스크레이프하고 해당 지표를 CloudWatch에 가져오려면 Prometheus 스크레이프 구성과 CloudWatch 에이전트 구성을 모두 수정한 다음, 업데이트된 구성으로 에이전트를 다시 배포합니다.

VPC 보안 그룹 요구 사항

Prometheus 워크로드의 보안 그룹 수신 규칙은 프라이빗 IP로 Prometheus 지표를 스크레이프하기 위해 CloudWatch 에이전트에 대한 Prometheus 포트를 열어야 합니다.

CloudWatch 에이전트의 보안 그룹 송신 규칙은 CloudWatch 에이전트가 프라이빗 IP로 Prometheus 워크로드의 포트에 연결할 수 있도록 허용해야 합니다.

Prometheus 스크레이프 구성

CloudWatch 에이전트는 Prometheus 설명서의 <scrape_config>에 설명된 대로 표준 Prometheus 스크레이프 구성을 지원합니다. 이 섹션을 편집하여 이 파일에 이미 있는 구성을 업데이트하고 Prometheus 스크레이핑 대상을 더 추가할 수 있습니다. 기본적으로 샘플 구성 파일에는 다음과 같은 글로벌 구성 줄이 포함되어 있습니다.

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
```

- scrape_interval - 대상을 스크레이프하는 빈도를 정의합니다.
- scrape_timeout - 스크레이프 요청 시간이 초과되기 전에 대기할 시간을 정의합니다.

작업 수준에서 이러한 설정에 다른 값을 정의하여 전역 구성을 재정의할 수도 있습니다.

Prometheus 스크레이핑 작업

CloudWatch 에이전트 YAML 파일에는 일부 기본 스크레이핑 작업이 이미 구성되어 있습니다. 예를 들어 `prometheus-eks.yaml`에서 기본 스크레이핑 작업은 `scrape_configs` 섹션의 `job_name` 줄에서 구성됩니다. 이 파일에서 다음 기본 `kubernetes-pod-jmx` 섹션에서는 JMX Exporter 지표를 스크레이프합니다.

```
   - job_name: 'kubernetes-pod-jmx'
      sample_limit: 10000
      metrics_path: /metrics
      kubernetes_sd_configs:
      - role: pod
      relabel_configs:
      - source_labels: [__address__]
        action: keep
        regex: '.*:9404$'
      - action: labelmap
        regex: __meta_kubernetes_pod_label_(.+)
      - action: replace
        source_labels:
        - __meta_kubernetes_namespace
        target_label: Namespace
      - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: pod_name
```

```
      - action: replace
        source_labels:
        - __meta_kubernetes_pod_container_name
        target_label: container_name
      - action: replace
        source_labels:
        - __meta_kubernetes_pod_controller_name
        target_label: pod_controller_name
      - action: replace
        source_labels:
        - __meta_kubernetes_pod_controller_kind
        target_label: pod_controller_kind
      - action: replace
        source_labels:
        - __meta_kubernetes_pod_phase
        target_label: pod_phase
```

이러한 각 기본 대상은 스크레이프되고 지표는 임베디드 지표 형식을 사용하여 로그 이벤트로 CloudWatch에 전송됩니다. 자세한 내용은 로그 내에 지표 포함 단원을 참조하십시오.

Amazon EKS 및 Kubernetes 클러스터의 로그 이벤트는 CloudWatch Logs의 /aws/containerinsights/***cluster_name***/prometheus 로그 그룹에 저장됩니다. Amazon ECS 클러스터의 로그 이벤트는 /aws/ecs/containerinsights/***cluster_name***/prometheus 로그 그룹에 저장됩니다.

각 스크레이핑 작업은 이 로그 그룹의 서로 다른 로그 스트림에 포함됩니다. 예를 들어 Prometheus 스크레이핑 작업 `kubernetes-pod-appmesh-envoy`는 App Mesh에 대해 정의됩니다. Amazon EKS 및 Kubernetes 클러스터의 모든 App Mesh Prometheus 지표는 /aws/containerinsights/***cluster_name***>prometheus/kubernetes-pod-appmesh-envoy/라는 로그 스트림에 전송됩니다.

새 스크레이핑 대상을 추가하려면 YAML 파일의 `scrape_configs` 섹션에 새 `job_name` 섹션을 추가하고 에이전트를 다시 시작합니다. 이 프로세스의 예는 새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Prometheus API 서버 지표 단원을 참조하세요.

Prometheus에 대한 CloudWatch 에이전트 구성

CloudWatch 에이전트 구성 파일에는 `metrics_collected` 아래에 Prometheus 스크레이핑 구성에 대한 `prometheus` 섹션이 있습니다. 여기에는 다음 구성 옵션이 포함됩니다.

- cluster_name - 로그 이벤트에서 레이블로 추가할 클러스터 이름을 지정합니다. 이 필드는 선택 사항입니다. 생략하는 경우 에이전트가 Amazon EKS 또는 Kubernetes 클러스터 이름을 감지할 수 있습니다.

- log_group_name - 스크레이프한 Prometheus 지표의 로그 그룹 이름을 지정합니다. 이 필드는 선택 사항입니다. 생략하는 경우 CloudWatch는 Amazon EKS 및 Kubernetes 클러스터의 로그에 /aws/containerinsights/***cluster_name***/prometheus를 사용합니다.
- prometheus_config_path - Prometheus 스크레이프 구성 파일 경로를 지정합니다. 이 필드의 값이 `env:`로 시작하는 경우 Prometheus 스크레이프 구성 파일 내용이 컨테이너의 환경 변수에서 검색됩니다. 이 값은 변경하지 마세요.
- ecs_service_discovery - Amazon ECS Prometheus 서비스 검색을 위한 구성을 지정하는 섹션입니다. 자세한 내용은 Amazon ECS 클러스터의 자동 검색에 대한 자세한 가이드 단원을 참조하십시오.

  `ecs_service_discovery` 섹션에는 다음 필드가 포함될 수 있습니다.

  - `sd_frequency`는 Prometheus Exporter를 검색하는 빈도입니다. 숫자와 단위 접미사를 지정합니다. 예를 들어 1분마다 한 번의 경우 `1m` 또는 30초마다 한 번의 경우 `30s`입니다. 유효한 단위 접미사는 `ns`, `us`, `ms`, `s`, `m`, `h`입니다.

    이 필드는 선택 사항입니다. 기본값은 60초(1분)입니다.
  - `sd_target_cluster`는 자동 검색의 대상 Amazon ECS 클러스터 이름입니다. 이 필드는 선택 사항입니다. 기본값은 CloudWatch 에이전트가 설치된 Amazon ECS 클러스터의 이름입니다.
  - `sd_cluster_region`은 대상 Amazon ECS 클러스터의 리전입니다. 이 필드는 선택 사항입니다. 기본값은 CloudWatch 에이전트가 설치된 Amazon ECS 클러스터의 리전입니다.
  - `sd_result_file`은 Prometheus 대상 결과의 YAML 파일 경로입니다. Prometheus 스크레이프 구성은 이 파일을 참조합니다.
  - `docker_label`은 Docker 레이블 기반 서비스 검색을 위한 구성을 지정하는 데 사용할 수 있는 선택적 섹션입니다. 이 섹션을 생략하면 Docker 레이블 기반 검색이 사용되지 않습니다. 이 섹션에는 다음 필드가 포함될 수 있습니다.
    - `sd_port_label`은 Prometheus 지표에 대한 컨테이너 포트를 지정하는 컨테이너의 Docker 레이블 이름입니다. 기본 값은 `ECS_PROMETHEUS_EXPORTER_PORT`입니다. 컨테이너에 이 Docker 레이블이 없다면 CloudWatch 에이전트는 이 필드를 건너뜁니다.
    - `sd_metrics_path_label`은 Prometheus 지표 경로를 지정하는 컨테이너의 Docker 레이블 이름입니다. 기본 값은 `ECS_PROMETHEUS_METRICS_PATH`입니다. 컨테이너에 이 도커 레이블이 없다면 에이전트는 기본 경로 `/metrics`를 가정합니다.
    - `sd_job_name_label`은 Prometheus 스크레이프 작업 이름을 지정하는 컨테이너의 Docker 레이블 이름입니다. 기본 값은 `job`입니다. 컨테이너에 이 Docker 레이블이 없다면 CloudWatch 에이전트는 Prometheus 스크레이프 구성의 작업 이름을 사용합니다.

- task_definition_list는 태스크 정의 기반 서비스 검색의 구성을 지정하는 데 사용할 수 있는 선택적 섹션입니다. 이 섹션을 생략하면 태스크 정의 기반 검색이 사용되지 않습니다. 이 섹션에는 다음 필드가 포함될 수 있습니다.
  - sd_task_definition_arn_pattern은 검색할 Amazon ECS 태스크 정의를 지정하는 데 사용할 패턴입니다. 이는 정규 표현식입니다.
  - sd_metrics_ports는 Prometheus 지표에 대한 containerPort를 나열합니다. containerPort를 세미콜론으로 구분합니다.
  - sd_container_name_pattern은 Amazon ECS 태스크 컨테이너 이름을 지정합니다. 이는 정규 표현식입니다.
  - sd_metrics_path는 Prometheus 지표 경로를 지정합니다. 이 필드를 생략하면 에이전트는 기본 경로 /metrics를 가정합니다.
  - sd_job_name은 Prometheus 스크레이프 작업 이름을 지정합니다. 이 필드를 생략하면 CloudWatch 에이전트는 Prometheus 스크레이프 구성의 작업 이름을 사용합니다.
- metric_declaration - 생성할 임베디드 지표 형식이 있는 로그 배열을 지정하는 섹션입니다. CloudWatch 에이전트가 기본적으로 가져오는 각 Prometheus 소스에 대한 metric_declaration 섹션이 있습니다. 이러한 섹션에는 각각 다음 필드가 포함됩니다.
  - label_matcher는 source_labels에 나열된 레이블의 값을 확인하는 정규 표현식입니다. 일치하는 지표는 CloudWatch에 전송된 임베디드 지표 형식에 포함할 수 있습니다.

    source_labels에 여러 레이블이 지정된 경우 label_matcher의 정규 표현식에 ^ 또는 $ 문자를 사용하지 않는 것이 좋습니다.
  - source_labels는 label_matcher 줄에 의해 확인되는 레이블의 값을 지정합니다.
  - label_separator는 여러 source_labels가 지정된 경우 label_matcher 줄에 사용할 구분 기호를 지정합니다. 기본값은 ;입니다. 다음 예에서 label_matcher 줄에 이 기본값이 사용된 것을 볼 수 있습니다.
  - metric_selectors는 수집하여 CloudWatch에 보낼 지표를 지정하는 정규 표현식입니다.
  - dimensions는 선택한 각 지표의 CloudWatch 측정기준으로 사용할 레이블 목록입니다.

다음 metric_declaration 예를 참조하세요.

```
"metric_declaration": [
  {
     "source_labels":[ "Service", "Namespace"],
     "label_matcher":"(.*node-exporter.*|.*kube-dns.*);kube-system",
     "dimensions":[
```

```
        ["Service", "Namespace"]
     ],
     "metric_selectors":[
        "^coredns_dns_request_type_count_total$"
     ]
  }
]
```

이 예에서는 다음 조건이 충족될 경우 임베디드 지표 형식 섹션을 로그 이벤트로 전송하도록 구성합니다.

- `Service`의 값에 `node-exporter` 또는 `kube-dns`가 포함되어 있습니다.
- `Namespace`의 값이 `kube-system`입니다.
- Prometheus `coredns_dns_request_type_count_total` 지표에 `Service`와 `Namespace` 레이블이 모두 포함되어 있습니다.

전송되는 로그 이벤트에는 다음과 같은 강조 표시된 섹션이 포함됩니다.

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"coredns_dns_request_type_count_total"
            }
         ],
         "Dimensions":[
            [
               "Namespace",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "Namespace":"kube-system",
   "Service":"kube-dns",
   "coredns_dns_request_type_count_total":2562,
   "eks_amazonaws_com_component":"kube-dns",
   "instance":"192.168.61.254:9153",
   "job":"kubernetes-service-endpoints",
```

```
    ...
}
```

새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Prometheus API 서버 지표

Kubernetes API 서버는 기본적으로 엔드포인트에 Prometheus 지표를 표시합니다. Kubernetes API 서버 스크레이핑 구성에 대한 공식 예제는 Github에 있습니다.

다음 튜토리얼에서는 아래 단계를 수행하여 Kubernetes API 서버 지표를 CloudWatch로 가져오는 방법을 보여 줍니다.

- CloudWatch 에이전트 YAML 파일에 Kubernetes API 서버의 Prometheus 스크레이핑 구성 추가
- CloudWatch 에이전트 YAML 파일에서 임베디드 지표 형식 지표 정의 구성
- (선택 사항) Kubernetes API 서버 지표에 대한 CloudWatch 대시보드 생성

> **Note**
>
> Kubernetes API 서버는 게이지, 카운터, 히스토그램 및 요약 지표를 표시합니다. 이 Prometheus 지표 지원 릴리스에서는 CloudWatch가 게이지, 카운터 및 요약 유형의 지표만 가져옵니다.

CloudWatch에서 Kubernetes API 서버 Prometheus 지표 수집을 시작하려면

1. 다음 명령 중 하나를 입력하여 `prometheus-eks.yaml`, `prometheus-eks-fargate.yaml` 또는 `prometheus-k8s.yaml` 파일의 최신 버전을 다운로드합니다.

   EC2 시작 유형의 Amazon EKS 클러스터의 경우 다음 명령을 입력합니다.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
   ```

   Fargate 시작 유형의 Amazon EKS 클러스터의 경우 다음 명령을 입력합니다.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml
   ```

Amazon EC2 인스턴스에서 실행되는 Kubernetes 클러스터의 경우 다음 명령을 입력합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
```

2. 텍스트 편집기로 파일을 열고 prometheus-config 섹션을 찾은 후 이 섹션 안에 다음 섹션을 추가합니다. 그리고 변경 사항을 저장합니다.

```
    # Scrape config for API servers
    - job_name: 'kubernetes-apiservers'
      kubernetes_sd_configs:
        - role: endpoints
          namespaces:
            names:
              - default
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels: [__meta_kubernetes_service_name,
__meta_kubernetes_endpoint_port_name]
        action: keep
        regex: kubernetes;https
      - action: replace
        source_labels:
        - __meta_kubernetes_namespace
        target_label: Namespace
      - action: replace
        source_labels:
        - __meta_kubernetes_service_name
        target_label: Service
```

3. 텍스트 편집기에서 YAML 파일을 열어 둔 상태로 cwagentconfig.json 섹션을 찾습니다. 다음 하위 섹션을 추가하고 변경 사항을 저장합니다. 이 섹션에서는 API 서버 지표를 CloudWatch 에이전트 허용 목록에 넣습니다. 다음과 같은 세 유형의 API 서버 지표가 허용 목록에 추가됩니다.

   - etcd 객체 카운트
   - API 서버 등록 컨트롤러 지표

- API 서버 요청 지표

```
{"source_labels": ["job", "resource"],
  "label_matcher": "^kubernetes-apiservers;(services|daemonsets.apps|
deployments.apps|configmaps|endpoints|secrets|serviceaccounts|replicasets.apps)",
  "dimensions": [["ClusterName","Service","resource"]],
  "metric_selectors": [
  "^etcd_object_counts$"
  ]
},
{"source_labels": ["job", "name"],
   "label_matcher": "^kubernetes-apiservers;APIServiceRegistrationController$",
   "dimensions": [["ClusterName","Service","name"]],
   "metric_selectors": [
   "^workqueue_depth$",
   "^workqueue_adds_total$",
   "^workqueue_retries_total$"
  ]
},
{"source_labels": ["job","code"],
  "label_matcher": "^kubernetes-apiservers;2[0-9]{2}$",
  "dimensions": [["ClusterName","Service","code"]],
  "metric_selectors": [
   "^apiserver_request_total$"
  ]
},
{"source_labels": ["job"],
  "label_matcher": "^kubernetes-apiservers",
  "dimensions": [["ClusterName","Service"]],
  "metric_selectors": [
  "^apiserver_request_total$"
  ]
},
```

4. 클러스터에 Prometheus가 지원되는 CloudWatch 에이전트를 이미 배포한 경우 다음 명령을 입력하여 해당 에이전트를 삭제해야 합니다.

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

5. 다음 명령 중 하나를 입력하여 업데이트된 구성으로 CloudWatch 에이전트를 배포합니다. EC2 시작 유형의 Amazon EKS 클러스터의 경우 다음 명령을 입력합니다.

```
kubectl apply -f prometheus-eks.yaml
```

Fargate 시작 유형의 Amazon EKS 클러스터의 경우 다음 명령을 입력합니다. *MyCluster* 및 *region*을 배포와 일치하는 값으로 바꿉니다.

```
cat prometheus-eks-fargate.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

Kubernetes 클러스터의 경우 다음 명령을 입력합니다. *MyCluster* 및 *region*을 배포와 일치하는 값으로 바꿉니다.

```
cat prometheus-k8s.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

이 작업을 완료하면 kubernetes-apiservers라는 새 로그 스트림이 /aws/containerinsights/***cluster_name***/prometheus 로그 그룹에 나타납니다. 이 로그 스트림은 다음과 같은 임베디드 지표 형식 정의가 있는 로그 이벤트를 포함해야 합니다.

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"apiserver_request_total"
            }
         ],
         "Dimensions":[
            [
               "ClusterName",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "ClusterName":"my-cluster-name",
   "Namespace":"default",
```

```
    "Service":"kubernetes",
    "Timestamp":"1592267020339",
    "Version":"0",
    "apiserver_request_count":0,
    "apiserver_request_total":0,
    "code":"0",
    "component":"apiserver",
    "contentType":"application/json",
    "instance":"192.0.2.0:443",
    "job":"kubernetes-apiservers",
    "prom_metric_type":"counter",
    "resource":"pods",
    "scope":"namespace",
    "verb":"WATCH",
    "version":"v1"
}
```

CloudWatch 콘솔의 ContainerInsights/Prometheus 네임스페이스에서 지표를 볼 수 있습니다. 또한 선택적으로 Prometheus Kubernetes API 서버 지표에 대한 CloudWatch 대시보드를 생성할 수도 있습니다.

(선택 사항) Kubernetes API 서버 지표에 대한 대시보드 생성

대시보드에서 Kubernetes API 서버 지표를 보려면 먼저, 이전 단원의 단계를 완료하여 CloudWatch에서 이러한 지표 수집을 시작해야 합니다.

Kubernetes API 서버 지표에 대한 대시보드를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 올바른 AWS 리전을 선택했는지 확인합니다.
3. 탐색 창에서 대시보드를 선택합니다.
4. 대시보드 생성을 선택합니다. 새 대시보드의 이름을 입력하고 대시보드 생성을 선택합니다.
5. 이 대시보드에 추가에서 취소를 선택합니다.
6. 작업, 소스 보기/편집을 선택합니다.
7. Kubernetes API 대시보드 소스라는 JSON 파일을 다운로드합니다.
8. 텍스트 편집기로 다운로드한 JSON 파일을 열고, 다음과 같이 변경합니다.

    - 모든 {{YOUR_CLUSTER_NAME}} 문자열을 정확한 클러스터 이름으로 바꿉니다. 텍스트 앞이나 뒤에 공백을 추가하지 않도록 하세요.

- 모든 {{YOUR_AWS_REGION}} 문자열을 지표가 수집되는 리전의 이름으로 바꿉니다. 예: us-west-2. 텍스트 앞이나 뒤에 공백을 추가하지 않도록 하세요.

9. JSON blob 전체를 복사하여 CloudWatch 콘솔의 텍스트 상자에 붙여넣어 이미 상자에 있는 내용을 바꿉니다.
10. 업데이트, 대시보드 저장을 선택합니다.

(선택 사항) Prometheus 지표 테스트를 위한 컨테이너화된 Amazon EKS 워크로드 샘플 설정

CloudWatch Container Insights에서 Prometheus 지표 지원을 테스트하기 위해 다음과 같은 컨테이너화된 워크로드 중 하나 이상을 설정할 수 있습니다. Prometheus가 지원되는 CloudWatch 에이전트는 이러한 각 워크로드에서 지표를 자동으로 수집합니다. 기본적으로 수집되는 지표를 보려면 CloudWatch 에이전트가 수집하는 Prometheus 지표 단원을 참조하세요.

이러한 워크로드를 설치하려면 다음 명령을 입력하여 Helm 3.x를 설치해야 합니다.

```
brew install helm
```

자세한 내용은 Helm을 참조하세요.

주제

- Amazon EKS 및 Kubernetes의 AWS App Mesh 샘플 워크로드 설정
- Amazon EKS 및 Kubernetes에서 샘플 트래픽이 포함된 NGINX 설정
- Amazon EKS 및 Kubernetes에서 지표 익스포터가 포함된 Memcached 설정
- Amazon EKS 및 Kubernetes에서 Java/JMX 샘플 워크로드 설정
- Amazon EKS 및 Kubernetes에서 지표 익스포터가 포함된 HAProxy 설정
- 새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Amazon EKS 및 Kubernetes 클러스터의 Redis

Amazon EKS 및 Kubernetes의 AWS App Mesh 샘플 워크로드 설정

CloudWatch Container Insights의 Prometheus 지원은 AWS App Mesh를 지원합니다. 다음 단원에서는 App Mesh를 설정하는 방법을 설명합니다.

CloudWatch Container Insights는 App Mesh Envoy 액세스 로그도 수집할 수 있습니다. 자세한 내용은 (선택 사항) App Mesh Envoy 액세스 로그 사용 설정 단원을 참조하십시오.

주제

- EC2 시작 유형의 Amazon EKS 클러스터 또는 Kubernetes 클러스터에서 AWS App Mesh 샘플 워크로드 설정
- Fargate 시작 유형의 Amazon EKS 클러스터에서 AWS App Mesh 샘플 워크로드 설정

EC2 시작 유형의 Amazon EKS 클러스터 또는 Kubernetes 클러스터에서 AWS App Mesh 샘플 워크로드 설정

EC2 시작 유형의 Amazon EKS를 실행하는 클러스터 또는 Kubernetes 클러스터에서 App Mesh를 설정할 경우 다음 지침을 따르세요.

IAM 권한 구성

Amazon EKS 또는 Kubernetes 노드 그룹의 IAM 역할에 AWSAppMeshFullAccess 정책을 추가해야 합니다. Amazon EKS에서 이 노드 그룹 이름은 `eksctl-integ-test-eks-prometheus-NodeInstanceRole-ABCDEFHIJKL`과 유사합니다. Kubernetes에서는 `nodes.integ-test-kops-prometheus.k8s.local`과 유사하게 보일 수 있습니다.

App Mesh 설치

App Mesh Kubernetes 컨트롤러를 설치하려면 App Mesh 컨트롤러의 지침을 따르세요.

샘플 애플리케이션 설치

aws-app-mesh-examples에는 여러 Kubernetes App Mesh 시연이 포함되어 있습니다. 이 튜토리얼에서는 http 라우팅이 들어오는 요청을 일치시키기 위해 어떻게 헤더를 사용하는지 보여 주는 샘플 컬러 애플리케이션을 설치합니다.

샘플 App Mesh 애플리케이션을 사용하여 Container Insights를 테스트하려면

1. 다음 지침에 따라 애플리케이션을 설치합니다. https://github.com/aws/aws-app-mesh-examples/tree/main/walkthroughs/howto-k8s-http-headers
2. 다음과 같이 curler 포드를 시작하여 트래픽을 생성합니다.

   ```
   kubectl -n default run -it curler --image=tutum/curl /bin/bash
   ```

3. HTTP 헤더를 변경하여 다양한 엔드포인트를 curl합니다. 다음과 같이 curl 명령을 여러 번 실행합니다.

   ```
   curl -H "color_header: blue" front.howto-k8s-http-headers.svc.cluster.local:8080/;
    echo;
   ```

```
curl -H "color_header: red" front.howto-k8s-http-headers.svc.cluster.local:8080/;
 echo;

curl -H "color_header: yellow" front.howto-k8s-http-
headers.svc.cluster.local:8080/; echo;
```

4. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
5. 클러스터가 실행되고 있는 AWS 리전에서 왼쪽 탐색 창의 [지표(Metrics)]를 선택합니다. 지표는 ContainerInsights/Prometheus 네임스페이스에 있습니다.
6. CloudWatch Logs 이벤트를 보려면 탐색 창에서 [로그 그룹(Log groups)]을 선택합니다. 이벤트는 로그 그룹 `/aws/containerinsights/your_cluster_name/prometheus` 의 로그 스트림 `kubernetes-pod-appmesh-envoy`에 있습니다.

App Mesh 테스트 환경 삭제

App Mesh 및 샘플 애플리케이션의 사용을 마쳤으면 다음 명령을 사용하여 불필요한 리소스를 삭제합니다. 다음 명령을 입력하여 샘플 애플리케이션을 삭제합니다.

```
cd aws-app-mesh-examples/walkthroughs/howto-k8s-http-headers/
kubectl delete -f _output/manifest.yaml
```

다음 명령을 입력하여 App Mesh 컨트롤러를 삭제합니다.

```
helm delete appmesh-controller -n appmesh-system
```

Fargate 시작 유형의 Amazon EKS 클러스터에서 AWS App Mesh 샘플 워크로드 설정

Fargate 시작 유형의 Amazon EKS를 실행하는 클러스터에서 App Mesh를 설정할 경우 다음 지침을 따르세요.

IAM 권한 구성

IAM 권한을 설정하려면 다음 명령을 입력합니다. *MyCluster*를 클러스터 이름으로 바꿉니다.

```
eksctl create iamserviceaccount --cluster MyCluster \
 --namespace howto-k8s-fargate \
 --name appmesh-pod \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshEnvoyAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapDiscoverInstanceAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess \
```

```
--attach-policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapFullAccess \
--override-existing-serviceaccounts \
--approve
```

App Mesh 설치

App Mesh Kubernetes 컨트롤러를 설치하려면 App Mesh 컨트롤러의 지침을 따르세요. Fargate 시작 유형의 Amazon EKS에 대한 지침을 따라야 합니다.

샘플 애플리케이션 설치

aws-app-mesh-examples에는 여러 Kubernetes App Mesh 시연이 포함되어 있습니다. 이 튜토리얼에서는 Fargate 시작 유형의 Amazon EKS 클러스터에서 작동하는 샘플 컬러 애플리케이션을 설치합니다.

샘플 App Mesh 애플리케이션을 사용하여 Container Insights를 테스트하려면

1. 다음 지침에 따라 애플리케이션을 설치합니다. https://github.com/aws/aws-app-mesh-examples/tree/main/walkthroughs/howto-k8s-fargate

   해당 지침에서는 올바른 Fargate 프로파일을 사용하여 새 클러스터를 생성한다고 가정합니다. 이미 설정한 Amazon EKS 클러스터를 사용하려는 경우 다음 명령을 사용하여 해당 클러스터를 이 데모에 맞게 설정할 수 있습니다. *MyCluster*를 클러스터 이름으로 바꿉니다.

   ```
   eksctl create iamserviceaccount --cluster MyCluster \
    --namespace howto-k8s-fargate \
    --name appmesh-pod \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshEnvoyAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapDiscoverInstanceAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapFullAccess \
    --override-existing-serviceaccounts \
    --approve
   ```

   ```
   eksctl create fargateprofile --cluster MyCluster \
   --namespace howto-k8s-fargate --name howto-k8s-fargate
   ```

2. 다음과 같이 프런트 애플리케이션 배포를 포트 포워딩합니다.

```
kubectl -n howto-k8s-fargate port-forward deployment/front 8080:8080
```

3. 다음과 같이 프런트 앱을 curl합니다.

```
while true; do  curl -s http://localhost:8080/color; sleep 0.1; echo ; done
```

4. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
5. 클러스터가 실행되고 있는 AWS 리전에서 왼쪽 탐색 창의 [지표(Metrics)]를 선택합니다. 지표는 ContainerInsights/Prometheus 네임스페이스에 있습니다.
6. CloudWatch Logs 이벤트를 보려면 탐색 창에서 [로그 그룹(Log groups)]을 선택합니다. 이벤트는 로그 그룹 `/aws/containerinsights/your_cluster_name/prometheus` 의 로그 스트림 `kubernetes-pod-appmesh-envoy`에 있습니다.

App Mesh 테스트 환경 삭제

App Mesh 및 샘플 애플리케이션의 사용을 마쳤으면 다음 명령을 사용하여 불필요한 리소스를 삭제합니다. 다음 명령을 입력하여 샘플 애플리케이션을 삭제합니다.

```
cd aws-app-mesh-examples/walkthroughs/howto-k8s-fargate/
kubectl delete -f _output/manifest.yaml
```

다음 명령을 입력하여 App Mesh 컨트롤러를 삭제합니다.

```
helm delete appmesh-controller -n appmesh-system
```

Amazon EKS 및 Kubernetes에서 샘플 트래픽이 포함된 NGINX 설정

NGINX는 로드 밸런서 및 역방향 프록시로도 사용할 수 있는 웹 서버입니다. Kubernetes가 진입에 NGINX를 사용하는 방법에 대한 자세한 내용은 kubernetes/ingress-nginx를 참조하세요.

샘플 트래픽 서비스와 함께 Ingress-NGINX를 설치하여 Container Insights Prometheus 지원을 테스트하려면

1. 다음 명령을 입력하여 Helm ingress-nginx 리포지토리를 추가합니다.

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

2. 다음 명령을 입력합니다.

```
kubectl create namespace nginx-ingress-sample

helm install my-nginx ingress-nginx/ingress-nginx \
--namespace nginx-ingress-sample \
--set controller.metrics.enabled=true \
--set-string controller.metrics.service.annotations."prometheus\.io/port"="10254" \
--set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

3. 다음 명령을 입력하여 서비스가 올바르게 시작되었는지 확인합니다.

```
kubectl get service -n nginx-ingress-sample
```

이 명령의 출력에는 EXTERNAL-IP 열을 포함한 여러 열이 표시되어야 합니다.

4. EXTERNAL-IP 변수를 NGINX 수신 컨트롤러의 행에 있는 EXTERNAL-IP 열 값으로 설정합니다.

```
EXTERNAL_IP=your-nginx-controller-external-ip
```

5. 다음 명령을 입력하여 샘플 NGINX 트래픽을 시작합니다.

```
SAMPLE_TRAFFIC_NAMESPACE=nginx-sample-traffic
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_traffic/nginx-traffic/nginx-traffic-sample.yaml |
sed "s/{{external_ip}}/$EXTERNAL_IP/g" |
sed "s/{{namespace}}/$SAMPLE_TRAFFIC_NAMESPACE/g" |
kubectl apply -f -
```

6. 다음 명령을 입력하여 세 개의 포드가 모두 Running 상태에 있는지 확인합니다.

```
kubectl get pod -n $SAMPLE_TRAFFIC_NAMESPACE
```

실행 중인 경우 ContainerInsights/Prometheus 네임스페이스에 지표가 곧 표시됩니다.

NGINX 및 샘플 트래픽 애플리케이션을 제거하려면

1. 다음 명령을 입력하여 샘플 트래픽 서비스를 삭제합니다.

```
kubectl delete namespace $SAMPLE_TRAFFIC_NAMESPACE
```

2. Helm 릴리스 이름으로 NGINX egress를 삭제합니다.

```
helm uninstall my-nginx --namespace nginx-ingress-sample
kubectl delete namespace nginx-ingress-sample
```

Amazon EKS 및 Kubernetes에서 지표 익스포터가 포함된 Memcached 설정

memcached는 오픈 소스 메모리 객체 캐싱 시스템입니다. 자세한 내용은 Memcached란 무엇입니까?를 참조하세요.

Fargate 시작 유형의 클러스터에서 Memcached를 실행할 경우 이 절차의 단계를 수행하기 전에 먼저, Fargate 프로파일을 설정해야 합니다. 프로파일을 설정하려면 다음 명령을 입력합니다. *MyCluster*를 클러스터 이름으로 바꿉니다.

```
eksctl create fargateprofile --cluster MyCluster \
--namespace memcached-sample --name memcached-sample
```

Container Insights Prometheus 지원을 테스트하기 위해 지표 익스포터와 함께 memcached를 설치하려면

1. 다음 명령을 입력하여 리포지토리를 추가합니다.

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. 다음 명령을 입력하여 새 네임스페이스를 생성합니다.

```
kubectl create namespace memcached-sample
```

3. 다음 명령을 입력하여 Memcached를 설치합니다.

```
helm install my-memcached bitnami/memcached --namespace memcached-sample \
--set metrics.enabled=true \
--set-string serviceAnnotations.prometheus\\.io/port="9150" \
--set-string serviceAnnotations.prometheus\\.io/scrape="true"
```

4. 다음 명령을 입력하여 실행 중인 서비스의 주석을 확인합니다.

```
kubectl describe service my-memcached-metrics -n memcached-sample
```

다음 두 주석이 표시됩니다.

```
Annotations:   prometheus.io/port: 9150
               prometheus.io/scrape: true
```

memcached를 제거하려면

- 다음 명령을 입력합니다.

```
helm uninstall my-memcached --namespace memcached-sample
kubectl delete namespace memcached-sample
```

Amazon EKS 및 Kubernetes에서 Java/JMX 샘플 워크로드 설정

JMX Exporter는 JMX mBeans를 스크레이프하여 노출할 수 있는 공식 Prometheus 익스포터입니다. 자세한 내용은 prometheus/jmx_exporter를 참조하세요.

Container Insights는 JMX Exporter를 사용하여 Java Virtual Machine(JVM), Java 및 Tomcat(Catalina)에서 사전 정의된 Prometheus 지표를 수집할 수 있습니다.

기본 Prometheus 스크레이프 구성

기본적으로 Prometheus가 지원되는 CloudWatch 에이전트는 Amazon EKS 또는 Kubernetes 클러스터에 있는 각 포드의 `http://CLUSTER_IP:9404/metrics`에서 Java/JMX Prometheus 지표를 스크레이프합니다. 이는 Prometheus `kubernetes_sd_config`의 `role: pod` 검색에 의해 수행됩니다. 9404는 Prometheus에서 JMX Exporter에 할당한 기본 포트입니다. `role: pod` 검색에 대한 자세한 내용은 포드를 참조하세요. 지표를 다른 포트 또는 metrics_path에 노출하도록 JMX Exporter를 구성할 수 있습니다. 포트 또는 경로를 변경하는 경우 CloudWatch 에이전트 config 맵의 기본 jmx scrape_config를 업데이트하세요. 다음 명령을 실행하여 현재 CloudWatch 에이전트 Prometheus 구성을 가져옵니다.

```
kubectl describe cm prometheus-config -n amazon-cloudwatch
```

변경할 필드는 다음 예제에서 강조 표시된 대로 `/metrics` 및 `regex: '.*:9404$'` 필드입니다.

```
job_name: 'kubernetes-jmx-pod'
sample_limit: 10000
```

```
metrics_path: /metrics
kubernetes_sd_configs:
- role: pod
relabel_configs:
- source_labels: [__address__]
  action: keep
  regex: '.*:9404$'
- action: replace
  regex: (.+)
  source_labels:
```

기타 Prometheus 스크레이프 구성

Kubernetes Service의 Java/JMX Prometheus Exporter가 있는 포드 세트에서 실행 중인 애플리케이션을 노출하는 경우 Prometheus `kubernetes_sd_config`의 `role: service` 검색 또는 `role: endpoint` 검색을 사용하도록 전환할 수도 있습니다. 이러한 검색 방법에 대한 자세한 내용은 서비스, 엔드포인트, <kubernetes_sd_config>를 참조하세요.

이러한 두 가지 서비스 검색 모드는 더 많은 메타 레이블을 제공하므로 CloudWatch 지표 측정기준을 구축하는 데 유용할 수 있습니다. 예를 들어 `__meta_kubernetes_service_name`의 레이블을 `Service`로 다시 지정하고 이를 지표의 측정기준에 포함할 수 있습니다. CloudWatch 지표 및 해당 측정기준의 사용자 지정에 대한 자세한 내용은 Prometheus에 대한 CloudWatch 에이전트 구성 단원을 참조하세요.

JMX Exporter가 포함된 Docker 이미지

다음으로, Docker 이미지를 구축합니다. 다음 단원에서는 두 가지 Dockerfile 예를 제공합니다.

이미지를 구축한 경우 Amazon EKS 또는 Kubernetes에 로드한 후 다음 명령을 실행하여 Prometheus 지표가 포트 9404에서 `JMX_EXPORTER`에 의해 노출되는지 확인합니다. *`$JAR_SAMPLE_TRAFFIC_POD`*를 실행 중인 포드 이름으로 바꾸고 *`$JAR_SAMPLE_TRAFFIC_NAMESPACE`*를 애플리케이션 네임스페이스로 바꿉니다.

Fargate 시작 유형의 클러스터에서 JMX Exporter를 실행할 경우 이 절차의 단계를 수행하기 전에 먼저, Fargate 프로파일도 설정해야 합니다. 프로파일을 설정하려면 다음 명령을 입력합니다. *`MyCluster`*를 클러스터 이름으로 바꿉니다.

```
eksctl create fargateprofile --cluster MyCluster \
--namespace $JAR_SAMPLE_TRAFFIC_NAMESPACE\
 --name $JAR_SAMPLE_TRAFFIC_NAMESPACE
```

```
kubectl exec $JAR_SAMPLE_TRAFFIC_POD -n $JARCAT_SAMPLE_TRAFFIC_NAMESPACE -- curl
 http://localhost:9404
```

예: Prometheus 지표가 있는 Apache Tomcat Docker 이미지

Apache Tomcat 서버는 기본적으로 JMX mBeans를 노출합니다. JMX Exporter를 Tomcat과 통합하여 JMX mBeans를 Prometheus 지표로 노출할 수 있습니다. 다음 Dockerfile 예는 테스트 이미지를 구축하는 단계를 보여 줍니다.

```
# From Tomcat 9.0 JDK8 OpenJDK
FROM tomcat:9.0-jdk8-openjdk

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter
COPY ./setenv.sh /usr/local/tomcat/bin
COPY your web application.war /usr/local/tomcat/webapps/

RUN chmod  o+x /usr/local/tomcat/bin/setenv.sh

ENTRYPOINT ["catalina.sh", "run"]
```

다음 목록은 이 Dockerfile에 있는 4개의 COPY 줄을 설명합니다.

- https://github.com/prometheus/jmx_exporter에서 최신 JMX Exporter jar 파일을 다운로드합니다.
- config.yaml은 JMX Exporter 구성 파일입니다. 자세한 내용은 https://github.com/prometheus/jmx_exporter#Configuration을 참조하세요.

  다음은 Java 및 Tomcat의 샘플 구성 파일입니다.

  ```
  lowercaseOutputName: true
  lowercaseOutputLabelNames: true

  rules:
  - pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
  TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
  ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
    name: java_lang_OperatingSystem_$1
    type: GAUGE
  ```

```
- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

- `setenv.sh`는 Tomcat과 함께 JMX Exporter를 시작하고 localhost의 포트 9404에 Prometheus 지표를 노출하는 Tomcat 시작 스크립트입니다. 또한 JMX Exporter에 `config.yaml` 파일 경로를 제공합니다.

```
$ cat setenv.sh
export JAVA_OPTS="-javaagent:/opt/jmx_exporter/
jmx_prometheus_javaagent-0.12.0.jar=9404:/opt/jmx_exporter/config.yaml $JAVA_OPTS"
```

- web application.war는 Tomcat에 의해 로드될 웹 애플리케이션 `war` 파일입니다.

이 구성으로 Docker 이미지를 구축하고 이미지 리포지토리에 업로드합니다.

예: Prometheus 지표가 있는 Java Jar 애플리케이션 Docker 이미지

다음 Dockerfile 예는 테스트 이미지를 구축하는 단계를 보여 줍니다.

```
# Alpine Linux with OpenJDK JRE
FROM openjdk:8-jre-alpine

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./SampleJavaApplication-1.0-SNAPSHOT.jar /opt/jmx_exporter
COPY ./start_exporter_example.sh /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter

RUN chmod -R o+x /opt/jmx_exporter
RUN apk add curl

ENTRYPOINT exec /opt/jmx_exporter/start_exporter_example.sh
```

다음 목록은 이 Dockerfile에 있는 4개의 `COPY` 줄을 설명합니다.

- https://github.com/prometheus/jmx_exporter에서 최신 JMX Exporter jar 파일을 다운로드합니다.
- `config.yaml`은 JMX Exporter 구성 파일입니다. 자세한 내용은 https://github.com/prometheus/jmx_exporter#Configuration을 참조하세요.

  다음은 Java 및 Tomcat의 샘플 구성 파일입니다.

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true
```

```
rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
```

```
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

- `start_exporter_example.sh`는 Prometheus 지표를 내보낸 JAR 애플리케이션을 시작하는 스크립트입니다. 또한 JMX Exporter에 `config.yaml` 파일 경로를 제공합니다.

```
$ cat start_exporter_example.sh
java -javaagent:/opt/jmx_exporter/jmx_prometheus_javaagent-0.12.0.jar=9404:/
opt/jmx_exporter/config.yaml -cp  /opt/jmx_exporter/SampleJavaApplication-1.0-
SNAPSHOT.jar com.gubupt.sample.app.App
```

- SampleJavaApplication-1.0-SNAPSHOT.jar은 샘플 Java 애플리케이션 jar 파일입니다. 이 파일을 모니터링할 Java 애플리케이션으로 바꿉니다.

이 구성으로 Docker 이미지를 구축하고 이미지 리포지토리에 업로드합니다.

Amazon EKS 및 Kubernetes에서 지표 익스포터가 포함된 HAProxy 설정

HAProxy는 오픈 소스 프록시 애플리케이션입니다. 자세한 내용은 HAProxy를 참조하세요.

Fargate 시작 유형의 클러스터에서 HAProxy를 실행할 경우 이 절차의 단계를 수행하기 전에 먼저, Fargate 프로파일을 설정해야 합니다. 프로파일을 설정하려면 다음 명령을 입력합니다. *MyCluster*를 클러스터 이름으로 바꿉니다.

```
eksctl create fargateprofile --cluster MyCluster \
--namespace haproxy-ingress-sample --name haproxy-ingress-sample
```

지표 익스포터와 함께 HAProxy를 설치하여 Container Insights Prometheus 지원을 테스트하려면

1. 다음 명령을 입력하여 Helm 인큐베이터 리포지토리를 추가합니다.

   ```
   helm repo add haproxy-ingress https://haproxy-ingress.github.io/charts
   ```

2. 다음 명령을 입력하여 새 네임스페이스를 생성합니다.

   ```
   kubectl create namespace haproxy-ingress-sample
   ```

3. 다음 명령을 입력하여 HAProxy를 설치합니다.

```
helm install haproxy haproxy-ingress/haproxy-ingress \
--namespace haproxy-ingress-sample \
--set defaultBackend.enabled=true \
--set controller.stats.enabled=true \
--set controller.metrics.enabled=true \
--set-string controller.metrics.service.annotations."prometheus\.io/port"="9101" \
--set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

4. 다음 명령을 입력하여 서비스의 주석을 확인합니다.

```
kubectl describe service haproxy-haproxy-ingress-metrics -n haproxy-ingress-sample
```

다음과 같은 주석이 표시됩니다.

```
Annotations:   prometheus.io/port: 9101
               prometheus.io/scrape: true
```

HAProxy를 제거하려면

- 다음 명령을 입력합니다.

```
helm uninstall haproxy --namespace haproxy-ingress-sample
kubectl delete namespace haproxy-ingress-sample
```

새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Amazon EKS 및 Kubernetes 클러스터의 Redis

이 튜토리얼에서는 Amazon EKS 및 Kubernetes에서 샘플 Redis 애플리케이션의 Prometheus 지표를 스크레이프하는 실습을 소개합니다. Redis(https://redis.io/)는 데이터베이스, 캐시 및 메시지 브로커로 사용되는 오픈 소스(BSD 라이선스), 인 메모리 구조 데이터 스토어입니다. 자세한 내용은 redis를 참조하세요.

redis_exporter(MIT License 라이선스)는 지정된 포트(기본값: 0.0.0.0:9121)에서 Redis Prometheus 지표를 노출하는 데 사용됩니다. 자세한 내용은 redis_exporter를 참조하세요.

이 튜토리얼에서는 다음과 같은 두 Docker Hub 리포지토리의 Docker 이미지를 사용합니다.

- redis
- redis_exporter

Prometheus 지표를 노출하는 샘플 Redis 워크로드를 설치하려면

1. 샘플 Redis 워크로드의 네임스페이스를 설정합니다.

```
REDIS_NAMESPACE=redis-sample
```

2. Fargate 시작 유형의 클러스터에서 Redis를 실행할 경우 Fargate 프로파일을 설정해야 합니다. 프로파일을 설정하려면 다음 명령을 입력합니다. *MyCluster*를 클러스터 이름으로 바꿉니다.

```
eksctl create fargateprofile --cluster MyCluster \
--namespace $REDIS_NAMESPACE --name $REDIS_NAMESPACE
```

3. 다음 명령을 입력하여 샘플 Redis 워크로드를 설치합니다.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_traffic/redis/redis-traffic-sample.yaml \
| sed "s/{{namespace}}/$REDIS_NAMESPACE/g" \
| kubectl apply -f -
```

4. 설치에는 포트 9121에서 Redis Prometheus 지표를 노출하는 `my-redis-metrics`라는 서비스가 포함되어 있습니다. 다음 명령을 입력하여 서비스 세부 정보를 가져옵니다.

```
kubectl describe service/my-redis-metrics  -n $REDIS_NAMESPACE
```

결과의 `Annotations` 섹션에는 다음과 같이 워크로드를 자동 검색할 수 있도록 CloudWatch 에이전트의 Prometheus 스크레이프 구성과 일치하는 두 개의 주석이 표시됩니다.

```
prometheus.io/port: 9121
prometheus.io/scrape: true
```

관련 Prometheus 스크레이프 구성은 `kubernetes-eks.yaml` 또는 `kubernetes-k8s.yaml`의 `- job_name: kubernetes-service-endpoints` 섹션에서 찾아볼 수 있습니다.

CloudWatch에서 Redis Prometheus 지표 수집을 시작하려면

1. 다음 명령 중 하나를 입력하여 최신 버전의 kubernetes-eks.yaml 또는 kubernetes-k8s.yaml 파일을 다운로드합니다. EC2 시작 유형의 Amazon EKS 클러스터의 경우 다음 명령을 입력합니다.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
   ```

   Fargate 시작 유형의 Amazon EKS 클러스터의 경우 다음 명령을 입력합니다.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml
   ```

   Amazon EC2 인스턴스에서 실행되는 Kubernetes 클러스터의 경우 다음 명령을 입력합니다.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
   ```

2. 텍스트 편집기를 사용하여 파일을 열고 cwagentconfig.json 섹션을 찾습니다. 다음 하위 섹션을 추가하고 변경 사항을 저장합니다. 이때 들여쓰기가 기존 패턴을 따라야 합니다.

   ```
   {
     "source_labels": ["pod_name"],
     "label_matcher": "^redis-instance$",
     "dimensions": [["Namespace","ClusterName"]],
     "metric_selectors": [
       "^redis_net_(in|out)put_bytes_total$",
       "^redis_(expired|evicted)_keys_total$",
       "^redis_keyspace_(hits|misses)_total$",
       "^redis_memory_used_bytes$",
       "^redis_connected_clients$"
     ]
   },
   {
     "source_labels": ["pod_name"],
     "label_matcher": "^redis-instance$",
     "dimensions": [["Namespace","ClusterName","cmd"]],
   ```

```
  "metric_selectors": [
    "^redis_commands_total$"
  ]
},
{
  "source_labels": ["pod_name"],
  "label_matcher": "^redis-instance$",
  "dimensions": [["Namespace","ClusterName","db"]],
  "metric_selectors": [
    "^redis_db_keys$"
  ]
},
```

추가한 섹션에서는 Redis 지표를 CloudWatch 에이전트 허용 목록에 넣습니다. 해당 지표 목록은 다음 단원을 참조하세요.

3. 이 클러스터에 Prometheus가 지원되는 CloudWatch 에이전트를 이미 배포한 경우 다음 명령을 입력하여 해당 에이전트를 삭제해야 합니다.

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

4. 다음 명령 중 하나를 입력하여 업데이트된 구성으로 CloudWatch 에이전트를 배포합니다. *MyCluster* 및 *region*을 설정과 일치하도록 바꿉니다.

   EC2 시작 유형의 Amazon EKS 클러스터의 경우 다음 명령을 입력합니다.

```
kubectl apply -f prometheus-eks.yaml
```

   Fargate 시작 유형의 Amazon EKS 클러스터의 경우 다음 명령을 입력합니다.

```
cat prometheus-eks-fargate.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

   Kubernetes 클러스터의 경우 다음 명령을 입력합니다.

```
cat prometheus-k8s.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

Redis Prometheus 지표 보기

이 튜토리얼에서는 CloudWatch의 ContainerInsights/Prometheus 네임스페이스에 다음 지표를 전송합니다. CloudWatch 콘솔을 사용하여 해당 네임스페이스의 지표를 볼 수 있습니다.

지표 이름	측정기준	
`redis_net_input_bytes_total`	ClusterName, `Namespace`	
`redis_net_output_bytes_total`	ClusterName, `Namespace`	
`redis_expired_keys_total`	ClusterName, `Namespace`	
`redis_evicted_keys_total`	ClusterName, `Namespace`	
`redis_keyspace_hits_total`	ClusterName, `Namespace`	
`redis_keyspace_misses_total`	ClusterName, `Namespace`	
`redis_memory_used_bytes`	ClusterName, `Namespace`	
`redis_connected_clients`	ClusterName, `Namespace`	
`redis_commands_total`	ClusterName, `Namespace` , cmd	

지표 이름	측정기준	
redis_db_keys	ClusterName, Namespace , db	

> **Note**
>
> [cmd] 측정기준의 값은 append, client, command, config, dbsize, flushall, get, incr, info, latency 또는 slowlog일 수 있습니다.
> [db] 측정기준의 값은 db0~db15일 수 있습니다.

또한 Redis Prometheus 지표에 대한 CloudWatch 대시보드를 생성할 수도 있습니다.

Redis Prometheus 지표에 대한 대시보드를 생성하려면

1. 환경 변수를 만들어서 아래의 값을 배포와 일치하도록 바꿉니다.

```
DASHBOARD_NAME=your_cw_dashboard_name
REGION_NAME=your_metric_region_such_as_us-east-1
CLUSTER_NAME=your_k8s_cluster_name_here
NAMESPACE=your_redis_service_namespace_here
```

2. 다음 명령을 입력하여 대시보드를 생성합니다.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/redis/cw_dashboard_redis.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| sed "s/{{YOUR_NAMESPACE}}/${NAMESPACE}/g" \
```

CloudWatch 에이전트의 Prometheus 지표 유형 변환

Prometheus 클라이언트 라이브러리는 다음과 같은 네 가지 핵심 지표 유형을 제공합니다.

- Counter
- 게이지
- 요약

- 히스토그램

CloudWatch 에이전트는 카운터, 게이지 및 요약 지표 유형을 지원합니다. 히스토그램 지표에 대한 지원은 향후 릴리스에서 계획되어 있습니다.

지원되지 않는 히스토그램 지표 유형이 포함된 Prometheus 지표는 CloudWatch 에이전트에서 삭제합니다. 자세한 내용은 삭제된 Prometheus 지표 로깅 단원을 참조하십시오.

게이지 지표

Prometheus 게이지 지표는 임의로 올라가고 내려갈 수 있는 단일 숫자 값을 나타내는 지표입니다. CloudWatch 에이전트는 게이지 지표를 스크레이프하고 해당 값을 곧바로 내보냅니다.

카운터 지표

Prometheus 카운터 지표는 값이 증가하거나 0으로 재설정될 수만 있는 단일 단조 증가 카운터를 나타내는 누적 지표입니다. CloudWatch 에이전트는 이전 스크레이프에서 델타를 계산하고 델타 값을 로그 이벤트의 지표 값으로 보냅니다. 따라서 CloudWatch 에이전트는 두 번째 스크레이프에서 하나의 로그 이벤트를 생성하고 후속 스크레이프를 계속합니다(있는 경우).

요약 지표

Prometheus 요약 지표는 여러 데이터 요소로 표현되는 복잡한 지표 유형입니다. 이 지표는 총 관측 수 및 모든 관측값의 합계를 제공합니다. 슬라이딩 기간 동안 구성 가능한 분위수를 계산합니다.

요약 지표의 합계 및 개수는 누적되지만 분위수는 누적되지 않습니다. 다음 예에서는 분위수의 분산을 보여 줍니다.

```
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 7.123e-06
go_gc_duration_seconds{quantile="0.25"} 9.204e-06
go_gc_duration_seconds{quantile="0.5"} 1.1065e-05
go_gc_duration_seconds{quantile="0.75"} 2.8731e-05
go_gc_duration_seconds{quantile="1"} 0.003841496
go_gc_duration_seconds_sum 0.37630427
go_gc_duration_seconds_count 9774
```

CloudWatch 에이전트는 이전 단원에서 설명한 대로 카운터 지표를 처리하는 것과 동일한 방식으로 요약 지표의 합계 및 개수를 처리합니다. CloudWatch 에이전트는 원래 보고된 대로 분위수 값을 유지합니다.

CloudWatch 에이전트가 수집하는 Prometheus 지표

Prometheus가 지원되는 CloudWatch 에이전트는 여러 서비스 및 워크로드에서 지표를 자동으로 수집합니다. 기본적으로 수집하는 지표 목록은 다음 단원에 나와 있습니다. 또한 이러한 서비스에서 더 많은 지표를 수집하고 다른 애플리케이션 및 서비스에서 Prometheus 지표를 수집하도록 에이전트를 구성할 수도 있습니다. 추가 지표 수집에 대한 자세한 내용은 Prometheus에 대한 CloudWatch 에이전트 구성 단원을 참조하세요.

Amazon EKS 및 Kubernetes 클러스터에서 수집된 Prometheus 지표는 ContainerInsights/Prometheus 네임스페이스에 있습니다. Amazon ECS 클러스터에서 수집된 Prometheus 지표는 ECS/ContainerInsights/Prometheus 네임스페이스에 있습니다.

주제

- App Mesh의 Prometheus 지표
- NGINX의 Prometheus 지표
- Memcached의 Prometheus 지표
- Java/JMX의 Prometheus 지표
- HAProxy의 Prometheus 지표

App Mesh의 Prometheus 지표

다음 지표는 App Mesh에서 자동으로 수집됩니다.

CloudWatch Container Insights는 App Mesh Envoy 액세스 로그도 수집할 수 있습니다. 자세한 내용은 (선택 사항) App Mesh Envoy 액세스 로그 사용 설정 단원을 참조하십시오.

Amazon EKS 및 Kubernetes 클러스터의 App Mesh에 대한 Prometheus 지표

지표 이름	측정기준	
envoy_http_downstream_rq_total	ClusterName, Namespace	
envoy_http_downstream_rq_xx	ClusterName, Namespace	

지표 이름	측정기준	
	ClusterName, Namespace , envoy_http_conn_manager_prefix, envoy_response_code_class	
envoy_cluster_upstream_cx_rx_bytes_total	ClusterName, Namespace	
envoy_cluster_upstream_cx_tx_bytes_total	ClusterName, Namespace	
envoy_cluster_membership_healthy	ClusterName, Namespace	
envoy_cluster_membership_total	ClusterName, Namespace	
envoy_server_memory_heap_size	ClusterName, Namespace	
envoy_server_memory_allocated	ClusterName, Namespace	
envoy_cluster_upstream_cx_connect_timeout	ClusterName, Namespace	

지표 이름	측정기준	
envoy_cluster_upstream_rq_pending_failure_eject	ClusterName, Namespace	
envoy_cluster_upstream_rq_pending_overflow	ClusterName, Namespace	
envoy_cluster_upstream_rq_timeout	ClusterName, Namespace	
envoy_cluster_upstream_rq_try_per_timeout	ClusterName, Namespace	
envoy_cluster_upstream_rq_rx_reset	ClusterName, Namespace	
envoy_cluster_upstream_cx_destroy_local_with_active_rq	ClusterName, Namespace	

지표 이름	측정기준	
envoy_cluster_upstream_cx_destroy_remote_active_rq	ClusterName, Namespace	
envoy_cluster_upstream_rq_maintenance_mode	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_paused_reading_total	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_resumed_reading_total	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_backed_up_total	ClusterName, Namespace	

지표 이름	측정기준	
envoy_cluster_upstream_flow_control_drained_total	ClusterName, Namespace	
envoy_cluster_upstream_rq_retry	ClusterName, Namespace	
envoy_cluster_upstream_rq_retry_success	ClusterName, Namespace	
envoy_cluster_upstream_rq_retry_overflow	ClusterName, Namespace	
envoy_server_live	ClusterName, Namespace	
envoy_server_uptime	ClusterName, Namespace	

Amazon ECS 클러스터의 App Mesh에 대한 Prometheus 지표

지표 이름	측정기준	
envoy_http_downstream_rq_total	ClusterName, TaskDefinitionFamily	

지표 이름	측정기준	
envoy_http_downstream_rq_xx	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_rx_bytes_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_tx_bytes_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_membership_healthy	ClusterName, TaskDefinitionFamily	
envoy_cluster_membership_total	ClusterName, TaskDefinitionFamily	
envoy_server_memory_heap_size	ClusterName, TaskDefinitionFamily	
envoy_server_memory_allocated	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_connect_timeout	ClusterName, TaskDefinitionFamily	

지표 이름	측정기준	
envoy_cluster_upstream_rq_pending_failure_eject	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_pending_overflow	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_timeout	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_try_per_timeout	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_rx_reset	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_destroy_local_with_active_rq	ClusterName, TaskDefinitionFamily	

지표 이름	측정기준	
envoy_cluster_upstream_cx_destroy_remote_active_rq	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_maintenance_mode	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_paused_reading_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_resumed_reading_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_backed_up_total	ClusterName, TaskDefinitionFamily	

지표 이름	측정기준
envoy_cluster_upstream_flow_control_drained_total	ClusterName, TaskDefinitionFamily
envoy_cluster_upstream_rq_retry	ClusterName, TaskDefinitionFamily
envoy_cluster_upstream_rq_retry_success	ClusterName, TaskDefinitionFamily
envoy_cluster_upstream_rq_retry_overflow	ClusterName, TaskDefinitionFamily
envoy_server_live	ClusterName, TaskDefinitionFamily
envoy_server_uptime	ClusterName, TaskDefinitionFamily
envoy_http_downstream_rq_xx	ClusterName, TaskDefinitionFamily, envoy_http_conn_manager_prefix, envoy_response_code_class ClusterName, TaskDefinitionFamily, envoy_response_code_class

> **Note**
>
> TaskDefinitionFamily는 Mesh의 Kubernetes 네임스페이스입니다.

envoy_http_conn_manager_prefix의 값은 ingress, egress 또는 admin일 수 있습니다.

envoy_response_code_class의 값은 1(1xx를 나타냄), 2(2xx를 나타냄), 3(3xx를 나타냄), 4(4xx를 나타냄) 또는 5(5xx를 나타냄)일 수 있습니다.

NGINX의 Prometheus 지표

다음 지표는 Amazon EKS 및 Kubernetes 클러스터의 NGINX에서 자동으로 수집됩니다.

지표 이름	측정기준	
nginx_ingress_controller_nginx_process_cpu_seconds_total	ClusterName, Namespace , 서비스	
nginx_ingress_controller_success	ClusterName, Namespace , 서비스	
nginx_ingress_controller_requests	ClusterName, Namespace , 서비스	
nginx_ingress_controller_nginx_process_connections	ClusterName, Namespace , 서비스	
nginx_ingress_controller_nginx_proce	ClusterName, Namespace , 서비스	

지표 이름	측정기준	
ss_connections_total		
nginx_ingress_controller_nginx_process_resident_memory_bytes	ClusterName, Namespace , 서비스	
nginx_ingress_controller_config_last_reload_successful	ClusterName, Namespace , 서비스	
nginx_ingress_controller_requests	ClusterName, Namespace , 서비스, 상태	

Memcached의 Prometheus 지표

다음 지표는 Amazon EKS 및 Kubernetes 클러스터의 Memcached에서 자동으로 수집됩니다.

지표 이름	측정기준	
memcached_current_items	ClusterName, Namespace , 서비스	
memcached_current_connections	ClusterName, Namespace , 서비스	

지표 이름	측정기준
memcached_limit_bytes	ClusterName, Namespace , 서비스
memcached_current_bytes	ClusterName, Namespace , 서비스
memcached_written_bytes_total	ClusterName, Namespace , 서비스
memcached_read_bytes_total	ClusterName, Namespace , 서비스
memcached_items_evicted_total	ClusterName, Namespace , 서비스
memcached_items_reclaimed_total	ClusterName, Namespace , 서비스
memcached_commands_total	ClusterName, Namespace , 서비스 ClusterName, Namespace , 서비스, 명령 ClusterName, Namespace , 서비스, 상태, 명령

Java/JMX의 Prometheus 지표

Amazon EKS 및 Kubernetes 클러스터에서 수집된 지표

Amazon EKS 및 Kubernetes 클러스터에서 Container Insights는 JMX Exporter를 사용하여 Java 가상 머신(JVM), Java 및 Tomcat(Catalina)에서 다음과 같은 미리 정의된 Prometheus 지표를 수집할 수 있습니다. 자세한 내용은 Github의 prometheus/jmx_exporter를 참조하세요.

Amazon EKS 및 Kubernetes 클러스터의 Java/JMX

지표 이름	측정기준	
jvm_classes_loaded	ClusterName , Namespace	
jvm_threads_current	ClusterName , Namespace	
jvm_threads_daemon	ClusterName , Namespace	
java_lang_operatingsystem_totalswapspacesize	ClusterName , Namespace	
java_lang_operatingsystem_systemcpuload	ClusterName , Namespace	
java_lang_operatingsystem_processcpuload	ClusterName , Namespace	
java_lang_operatingsystem_freeswapspacesize	ClusterName , Namespace	
java_lang_operatingsystem_totalphysicalmemorysize	ClusterName , Namespace	

지표 이름	측정기준	
java_lang_operatingsystem_freephysicalmemorysize	ClusterName , Namespace	
java_lang_operatingsystem_openfiledescriptorcount	ClusterName , Namespace	
java_lang_operatingsystem_availableprocessors	ClusterName , Namespace	
jvm_memory_bytes_used	ClusterName , Namespace , 영역	
jvm_memory_pool_bytes_used	ClusterName , Namespace , 풀	

> **Note**
>
> area 측정기준 값은 heap 또는 nonheap일 수 있습니다.
> pool 측정기준 값은 Tenured Gen, Compress Class Space, Survivor Space, Eden Space, Code Cache 또는 Metaspace일 수 있습니다.

Amazon EKS 및 Kubernetes 클러스터의 Tomcat/JMX

이전 테이블의 Java/JMX 지표 외에도 Tomcat 워크로드에 대해 다음 지표가 수집됩니다.

지표 이름	측정기준	
catalina_manager_activesessions	ClusterName ,Namespace	
catalina_manager_rejectedsessions	ClusterName ,Namespace	
catalina_globalrequestprocessor_bytesreceived	ClusterName ,Namespace	
catalina_globalrequestprocessor_bytessent	ClusterName ,Namespace	
catalina_globalrequestprocessor_requestcount	ClusterName ,Namespace	
catalina_globalrequestprocessor_errorcount	ClusterName ,Namespace	
catalina_globalrequestproce	ClusterName ,Namespace	

지표 이름	측정기준	
ssor_proc essingtime		

Amazon ECS 클러스터의 Java/JMX

지표 이름	측정기준	
jvm_class es_loaded	ClusterName , TaskDefinitionFamily	
jvm_threa ds_current	ClusterName , TaskDefinitionFamily	
jvm_threa ds_daemon	ClusterName , TaskDefinitionFamily	
java_lang _operatin gsystem_t otalswaps pacesize	ClusterName , TaskDefinitionFamily	
java_lang _operatin gsystem_s ystemcpuload	ClusterName , TaskDefinitionFamily	
java_lang _operatin gsystem_p rocesscpuload	ClusterName , TaskDefinitionFamily	
java_lang _operatin gsystem_f	ClusterName , TaskDefinitionFamily	

지표 이름	측정기준	
reeswapsp acesize		
java_lang _operatin gsystem_t otalphysi calmemorysize	ClusterName , TaskDefinitionFamily	
java_lang _operatin gsystem_f reephysic almemorysize	ClusterName , TaskDefinitionFamily	
java_lang _operatin gsystem_o penfilede scriptorcount	ClusterName , TaskDefinitionFamily	
java_lang _operatin gsystem_a vailablep rocessors	ClusterName , TaskDefinitionFamily	
jvm_memor y_bytes_used	ClusterName , TaskDefinitionFamily, 구역	
jvm_memor y_pool_by tes_used	ClusterName , TaskDefinitionFamily, 풀	

> **Note**
>
> `area` 측정기준 값은 `heap` 또는 `nonheap`일 수 있습니다.
> `pool` 측정기준 값은 `Tenured Gen`, `Compress Class Space`, `Survivor Space`, `Eden Space`, `Code Cache` 또는 `Metaspace`일 수 있습니다.

Amazon ECS 클러스터의 Tomcat/JMX

이전 표의 Java/JMX 지표 외에도 Amazon ECS 클러스터의 Tomcat 워크로드에 대해 다음 지표도 수집됩니다.

지표 이름	측정기준	
`catalina_manager_activesessions`	`ClusterName` , `TaskDefinitionFamily`	
`catalina_manager_rejectedsessions`	`ClusterName` , `TaskDefinitionFamily`	
`catalina_globalrequestprocessor_bytesreceived`	`ClusterName` , `TaskDefinitionFamily`	
`catalina_globalrequestprocessor_bytessent`	`ClusterName` , `TaskDefinitionFamily`	
`catalina_globalrequestproce`	`ClusterName` , `TaskDefinitionFamily`	

지표 이름	측정기준	
ssor_requestcount		
catalina_globalrequestprocessor_errorcount	ClusterName , TaskDefinitionFamily	
catalina_globalrequestprocessor_processingtime	ClusterName , TaskDefinitionFamily	

HAProxy의 Prometheus 지표

다음 지표는 Amazon EKS 및 Kubernetes 클러스터의 HAProxy에서 자동으로 수집됩니다.

수집된 지표는 사용 중인 HAProxy Ingress 버전에 따라 다릅니다. HAProxy Ingress 및 해당 버전에 대한 자세한 내용은 haproxy-ingress를 참조하세요.

지표 이름	측정기준	가용성
haproxy_backend_bytes_in_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_backend_bytes_out_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_backend_connection_errors_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전

지표 이름	측정기준	가용성
haproxy_backend_connections_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_backend_current_sessions	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_backend_http_responses_total	ClusterName , Namespace , 서비스, 코드, 백엔드	모든 HAProxy Ingress 버전
haproxy_backend_status	ClusterName , Namespace , 서비스	HAProxy Ingress 버전 0.10 이상에서만
haproxy_backend_up	ClusterName , Namespace , 서비스	HAProxy Ingress 버전 0.10 미만에서만
haproxy_frontend_bytes_in_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_frontend_bytes_out_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_frontend_connections_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전

지표 이름	측정기준	가용성
haproxy_frontend_current_sessions	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_frontend_http_requests_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_frontend_http_responses_total	ClusterName , Namespace , 서비스, 코드, 프런트엔드	모든 HAProxy Ingress 버전
haproxy_frontend_request_errors_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전
haproxy_frontend_requests_denied_total	ClusterName , Namespace , 서비스	모든 HAProxy Ingress 버전

Note

code 측정기준 값은 1xx, 2xx, 3xx, 4xx, 5xx 또는 other일 수 있습니다.
backend 측정기준의 값은 다음일 수 있습니다.

- HAProxy Ingress 버전 0.0.27 이하의 경우 http-default-backend, http-shared-backend 또는 httpsback-shared-backend
- 0.0.27보다 뒤에 출시된 HAProxy Ingress 버전의 경우 _default_backend

frontend 측정기준의 값은 다음일 수 있습니다.

- HAProxy Ingress 버전 0.0.27 이하의 경우 `httpfront-default-backend`, `httpfront-shared-frontend` 또는 `httpfronts`
- 0.0.27보다 뒤에 출시된 HAProxy Ingress 버전의 경우 `_front_http` 또는 `_front_https`

Prometheus 지표 보기

App Mesh, NGINX, Java/JMX, Memcached, HAProxy의 큐레이팅되고 사전 집계된 지표 및 사용자가 추가했을 수 있는 수동으로 구성된 기타 Prometheus Exporter를 포함하여 모든 Prometheus 지표를 모니터링하고 경보를 보낼 수 있습니다. 다른 Prometheus 익스포터에서 지표 수집에 대한 자세한 내용은 새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Prometheus API 서버 지표 단원을 참조하세요.

CloudWatch 콘솔에서 Container Insights는 다음과 같은 사전 구축된 보고서를 제공합니다.

- Amazon EKS 및 Kubernetes 클러스터의 경우 App Mesh, NGINX, HAPROXY, Memcached, Java/JMX에 대해 사전 구축된 보고서가 있습니다.
- Amazon ECS 클러스터의 경우 App Mesh 및 Java/JMX에 대해 사전 구축된 보고서가 있습니다.

또한 Container Insights는 Container Insights가 큐레이팅된 지표를 수집하는 각 워크로드에 대한 사용자 지정 대시보드를 제공합니다. GitHub에서 이러한 대시보드를 다운로드할 수 있습니다.

모든 Prometheus 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 네임스페이스 목록에서 [ContainerInsights/Prometheus] 또는 [ECS/ContainerInsights/Prometheus]를 선택합니다.
4. 다음 목록에서 측정기준 세트 중 하나를 선택합니다. 그런 다음, 보려는 지표 옆의 확인란을 선택합니다.

Prometheus 지표에 대한 사전 작성된 보고서를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 성능 모니터링을 선택합니다.
3. 페이지 상단 근처의 드롭다운 상자에서 Prometheus 옵션을 선택합니다.

다른 드롭다운 상자에서 보려는 클러스터를 선택합니다.

NGINX, App Mesh, Memcached, HAProxy, Java/JMX에 대한 사용자 지정 대시보드도 제공합니다.

Amazon에서 제공한 사용자 지정 대시보드를 사용하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 대시보드(Dashboards)를 선택합니다.
3. 대시보드 생성을 선택합니다. 새 대시보드의 이름을 입력하고 대시보드 생성을 선택합니다.
4. 이 대시보드에 추가에서 취소를 선택합니다.
5. 작업, 소스 보기/편집을 선택합니다.
6. 다음 JSON 파일 중 하나를 다운로드합니다.
   - Github의 NGINX 사용자 지정 대시보드 소스.
   - Github의 App Mesh 사용자 지정 대시보드 소스
   - Github의 Memcached 사용자 지정 대시보드 소스
   - Github의 HAProxy-Ingress 사용자 지정 대시보드 소스
   - Github의 Java/JMX 사용자 지정 대시보드 소스.
7. 텍스트 편집기로 다운로드한 JSON 파일을 열고, 다음과 같이 변경합니다.
   - 모든 `{{YOUR_CLUSTER_NAME}}` 문자열을 정확한 클러스터 이름으로 바꿉니다. 텍스트 앞이나 뒤에 공백을 추가하지 않도록 하세요.
   - 모든 `{{YOUR_REGION}}` 문자열을 클러스터가 실행되고 있는 AWS 리전으로 바꿉니다. 예를 들어 **`us-west-1`**으로 바꿉니다. 텍스트 앞뒤에 공백을 추가해서는 안 됩니다.
   - 모든 `{{YOUR_NAMESPACE}}` 문자열을 워크로드의 정확한 네임스페이스로 바꿉니다.
   - 모든 `{{YOUR_SERVICE_NAME}}` 문자열을 워크로드의 정확한 서비스 이름으로 바꿉니다. 예제: **`haproxy-haproxy-ingress-controller-metrics`**
8. JSON blob 전체를 복사하여 CloudWatch 콘솔의 텍스트 상자에 붙여넣어 이미 상자에 있는 내용을 바꿉니다.
9. 업데이트, 대시보드 저장을 선택합니다.

Prometheus 지표 문제 해결

이 단원에서는 Prometheus 지표 설정 문제를 해결하기 위한 도움말을 제공합니다.

주제

- Amazon ECS의 Prometheus 지표 문제 해결
- Amazon EKS 및 Kubernetes 클러스터의 Prometheus 지표 문제 해결

Amazon ECS의 Prometheus 지표 문제 해결

이 단원에서는 Amazon ECS 클러스터에서 Prometheus 지표 설정 문제를 해결하기 위한 도움말을 제공합니다.

CloudWatch Logs에 전송된 Prometheus 지표가 표시되지 않음

Prometheus 지표는 로그 그룹 /aws/ecs/containerinsights/cluster-name/Prometheus의 로그 이벤트로 수집됩니다. 로그 그룹이 생성되지 않았거나 Prometheus 지표가 로그 그룹에 전송되지 않은 경우 먼저, CloudWatch 에이전트가 Prometheus 대상을 성공적으로 검색했는지 여부를 확인해야 합니다. 그런 다음, CloudWatch 에이전트의 보안 그룹 및 권한 설정을 확인합니다. 다음 단계에서는 디버깅 수행을 안내합니다.

1단계: CloudWatch 에이전트 디버깅 모드 사용 설정

먼저 AWS CloudFormation 템플릿 파일 `cwagent-ecs-prometheus-metric-for-bridge-host.yaml` 또는 `cwagent-ecs-prometheus-metric-for-awsvpc.yaml`에 다음과 같이 굵게 표시된 줄을 추가하여 CloudWatch 에이전트를 디버그 모드로 변경합니다. 그런 다음 파일을 저장합니다.

```
cwagentconfig.json: |
    {
      "agent": {
        "debug": true
      },
      "logs": {
        "metrics_collected": {
```

기존 스택에 대해 새 AWS CloudFormation 변경 세트를 생성합니다. 변경 세트의 다른 파라미터를 기존 AWS CloudFormation 스택과 동일한 값으로 설정합니다. 다음은 EC2 시작 유형 및 브리지 네트워크 모드를 사용하는 Amazon ECS 클러스터에 설치된 CloudWatch 에이전트를 위한 예입니다.

```
ECS_NETWORK_MODE=bridge
 CREATE_IAM_ROLES=True
```

```
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name
NEW_CHANGESET_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME
 \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name $NEW_CHANGESET_NAME
```

AWS CloudFormation 콘솔로 이동하여 새 변경 세트인 $NEW_CHANGESET_NAME을 검토합니다. CWAgentConfigSSMParameter 리소스에 적용된 변경 사항 하나가 있어야 합니다. 변경 세트를 실행하고 다음 명령을 입력하여 CloudWatch 에이전트 태스크를 다시 시작합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service your_service_name_here \
--region $AWS_REGION
```

10초 정도 기다린 후 다음 명령을 입력합니다.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service your_service_name_here \
--region $AWS_REGION
```

2단계: ECS 서비스 검색 로그 확인

CloudWatch 에이전트의 ECS 태스크 정의는 아래 섹션에서 기본적으로 로그를 사용 설정합니다. 로그는 CloudWatch Logs의 로그 그룹 /ecs/ecs-cwagent-prometheus에 전송됩니다.

```
LogConfiguration:
  LogDriver: awslogs
    Options:
```

```
      awslogs-create-group: 'True'
      awslogs-group: "/ecs/ecs-cwagent-prometheus"
      awslogs-region: !Ref AWS::Region
      awslogs-stream-prefix: !Sub 'ecs-${ECSLaunchType}-awsvpc'
```

다음 예와 같이 ECS_SD_Stats 문자열로 로그를 필터링하여 ECS 서비스 검색과 관련된 지표를 가져옵니다.

```
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeContainerInstances: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeInstancesRequest: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeTaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeTasks: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_ListTasks: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: Exporter_DiscoveredTargetCount: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Get_EC2MetaData: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Get_TaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Size_ContainerInstance: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Size_TaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: Latency: 43.399783ms
```

특정 ECS 서비스 검색 주기에 대한 각 지표의 의미는 다음과 같습니다.

- AWSCLI_DescribeContainerInstances – 수행된 ECS::DescribeContainerInstances API 호출의 수입니다.
- AWSCLI_DescribeInstancesRequest – 수행된 ECS::DescribeInstancesRequest API 호출의 수입니다.
- AWSCLI_DescribeTaskDefinition – 수행된 ECS::DescribeTaskDefinition API 호출의 수입니다.
- AWSCLI_DescribeTasks – 수행된 ECS::DescribeTasks API 호출의 수입니다.
- AWSCLI_ListTasks - 수행된 ECS::ListTasks API 호출의 수입니다.
- ExporterDiscoveredTargetCount – 검색되어 컨테이너 내의 대상 결과 파일로 내보낸 Prometheus 대상의 수입니다.
- LRUCache_Get_EC2MetaData – 컨테이너 인스턴스 메타데이터가 캐시에서 검색된 횟수입니다.
- LRUCache_Get_TaskDefinition – ECS 태스크 정의 메타데이터가 캐시에서 검색된 횟수입니다.
- LRUCache_Size_ContainerInstance – 메모리에 캐시된 고유 컨테이너 인스턴스의 메타데이터 수입니다.
- LRUCache_Size_TaskDefinition – 메모리에 캐시된 고유 ECS 태스크 정의의 수입니다.

- Latency – 서비스 검색 주기에 걸리는 시간입니다.

`ExporterDiscoveredTargetCount` 값을 확인하여 검색된 Prometheus 대상이 예상과 일치하는지 확인하세요. 그렇지 않은 경우 가능한 이유는 다음과 같습니다.

- ECS 서비스 검색의 구성이 애플리케이션의 설정과 일치하지 않을 수 있습니다. Docker 레이블 기반 서비스 검색의 경우 대상 컨테이너에서 자동 검색에 필요한 Docker 레이블이 CloudWatch 에이전트에 구성되어 있지 않을 수 있습니다. ECS 태스크 정의 ARN 정규 표현식 기반 서비스 검색의 경우 CloudWatch 에이전트의 regex 설정이 애플리케이션의 태스크 정의와 일치하지 않을 수 있습니다.
- CloudWatch 에이전트의 ECS 태스크 역할에 ECS 태스크의 메타데이터를 검색할 권한이 없을 수 있습니다. CloudWatch 에이전트에 다음과 같은 읽기 전용 권한이 부여되어 있는지 확인합니다.
  - `ec2:DescribeInstances`
  - `ecs:ListTasks`
  - `ecs:DescribeContainerInstances`
  - `ecs:DescribeTasks`
  - `ecs:DescribeTaskDefinition`

3단계: 네트워크 연결 및 ECS 태스크 역할 정책 확인

`Exporter_DiscoveredTargetCount` 값이 검색된 Prometheus 대상이 있음을 나타내는데도 여전히 대상 CloudWatch Logs 로그 그룹에 전송된 로그 이벤트가 없다면 이는 다음 중 하나로 인해 발생했을 수 있습니다.

- CloudWatch 에이전트가 Prometheus 대상 포트에 연결하지 못할 수 있습니다. CloudWatch 에이전트 뒤에 있는 보안 그룹 설정을 확인합니다. 프라이빗 IP는 CloudWatch 에이전트가 Prometheus Exporter 포트에 연결할 수 있도록 허용해야 합니다.
- CloudWatch 에이전트의 ECS 태스크 역할에 CloudWatchAgentServerPolicy 관리형 정책이 없을 수 있습니다. Prometheus 지표를 로그 이벤트로 전송할 수 있으려면 CloudWatch 에이전트의 ECS 태스크 역할에 이 정책이 있어야 합니다. 샘플 AWS CloudFormation 템플릿을 사용하여 IAM 역할을 자동으로 생성했다면 ECS 태스크 역할과 ECS 실행 역할에 모두 Prometheus 모니터링을 수행할 수 있는 최소 권한이 부여됩니다.

Amazon EKS 및 Kubernetes 클러스터의 Prometheus 지표 문제 해결

이 단원에서는 Amazon EKS 및 Kubernetes 클러스터에서 Prometheus 지표 설정 문제를 해결하기 위한 도움말을 제공합니다.

Amazon EKS의 일반적인 문제 해결 단계

CloudWatch 에이전트가 실행 중인지 확인하려면 다음 명령을 입력합니다.

```
kubectl get pod -n amazon-cloudwatch
```

출력에는 NAME 열에 cwagent-prometheus-*id*가 있고 STATUS column.에 Running이 있는 행이 포함되어 있습니다.

실행 중인 포드에 대한 세부 정보를 표시하려면 다음 명령을 입력합니다. *pod-name*을 cw-agent-prometheus로 시작하는 이름이 있는 포드의 전체 이름으로 바꿉니다.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

CloudWatch Container Insights를 설치한 경우 CloudWatch Logs Insights를 사용하여 Prometheus 지표를 수집하는 CloudWatch 에이전트에서 로그를 쿼리할 수 있습니다.

애플리케이션 로그를 쿼리하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [CloudWatch Logs Insights]를 선택합니다.
3. 애플리케이션 로그의 로그 그룹 /aws/containerinsights/***cluster-name***/application을 선택합니다.
4. 검색 쿼리 표현식을 다음 쿼리로 바꾸고 쿼리 실행을 선택합니다.

```
fields ispresent(kubernetes.pod_name) as haskubernetes_pod_name, stream,
 kubernetes.pod_name, log |
filter haskubernetes_pod_name and kubernetes.pod_name like /cwagent-prometheus
```

Prometheus 지표 및 메타데이터가 CloudWatch Logs 이벤트로 수집되고 있는지 확인할 수도 있습니다.

Prometheus 데이터가 수집되는지 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 [CloudWatch Logs Insights]를 선택합니다.
3. /aws/containerinsights/***cluster-name***/prometheus를 선택합니다.
4. 검색 쿼리 표현식을 다음 쿼리로 바꾸고 쿼리 실행을 선택합니다.

```
fields @timestamp, @message | sort @timestamp desc | limit 20
```

삭제된 Prometheus 지표 로깅

이 릴리스는 히스토그램 유형의 Prometheus 지표를 수집하지 않습니다. CloudWatch 에이전트를 사용하면 Prometheus 지표가 히스토그램 지표라서 삭제되고 있는지 여부를 확인할 수 있습니다. Prometheus 지표가 히스토그램 지표이므로 삭제되고 CloudWatch에 전송되지 않은 처음 500개의 Prometheus 지표 목록을 로그할 수도 있습니다.

지표가 삭제되는지 확인하려면 다음 명령을 입력합니다.

```
kubectl logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

지표가 삭제되는 경우 `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` 파일에 다음 줄이 표시됩니다.

```
I! Drop Prometheus metrics with unsupported types. Only Gauge, Counter and Summary are
 supported.
I! Please enable CWAgent debug mode to view the first 500 dropped metrics
```

이러한 줄이 표시되는 경우 삭제되는 지표가 무엇인지 알고 싶다면 다음 단계를 사용하세요.

삭제된 Prometheus 지표 목록을 로깅하려면

1. `prometheus-eks.yaml` 또는 `prometheus-k8s.yaml` 파일에 다음과 같이 굵게 표시된 줄을 추가하여 CloudWatch 에이전트를 디버그 모드로 변경하고 파일을 저장합니다.

   ```
   {
         "agent": {
           "debug": true
         },
   ```

   파일의 이 섹션은 다음과 같아야 합니다.

   ```
   cwagentconfig.json: |
       {
         "agent": {
           "debug": true
         },
   ```

```
      "logs": {
        "metrics_collected": {
```

2. 다음 명령을 입력함으로써 CloudWatch 에이전트를 다시 설치하여 디버그 모드를 사용합니다.

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
kubectl apply -f prometheus.yaml
```

   삭제된 지표는 CloudWatch 에이전트 포드에 로그됩니다.

3. CloudWatch 에이전트 포드에서 로그를 검색하려면 다음 명령을 입력합니다.

```
kubectl logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

   또는 Container Insights Fluentd 로깅을 설치한 경우 로그는 CloudWatch Logs 로그 그룹 /aws/containerinsights/***cluster_name***/application에도 저장됩니다.

   이러한 로그를 쿼리하려면 Amazon EKS의 일반적인 문제 해결 단계에서 애플리케이션 로그를 쿼리하는 단계를 수행 할 수 있습니다.

CloudWatch Logs 로그 이벤트로 수집된 Prometheus 지표는 어디에 있습니까?

CloudWatch 에이전트는 각 Prometheus 스크레이프 작업 구성에 대한 로그 스트림을 생성합니다. 예를 들어 `prometheus-eks.yaml` 및 `prometheus-k8s.yaml` 파일에서 `job_name: 'kubernetes-pod-appmesh-envoy'` 줄은 App Mesh 지표를 스크레이프합니다. Prometheus 대상은 `kubernetes-pod-appmesh-envoy`로 정의됩니다. 따라서 모든 App Mesh Prometheus 지표는 /aws/containerinsights/cluster-name/Prometheus라는 로그 그룹 아래의 로그 스트림 kubernetes-pod-appmesh-envoy에 CloudWatch Logs 이벤트로 수집됩니다.

CloudWatch 지표에 Amazon EKS 또는 Kubernetes Prometheus 지표가 표시되지 않음

먼저 Prometheus 지표가 로그 그룹 /aws/containerinsights/cluster-name/Prometheus의 로그 이벤트로 수집되는지 확인합니다. CloudWatch Logs 로그 이벤트로 수집된 Prometheus 지표는 어디에 있습니까?의 정보를 사용하여 대상 로그 스트림을 확인할 수 있습니다. 로그 스트림이 생성되지 않았거나 로그 스트림에 새 로그 이벤트가 없는 경우 다음을 확인하세요.

- Prometheus 지표 익스포터 엔드포인트가 올바르게 설정되었는지 확인합니다.
- CloudWatch 에이전트 YAML 파일의 `config map: cwagent-prometheus` 섹션에서 Prometheus 스크레이핑 구성이 올바른지 확인합니다. 구성은 Prometheus 구성 파일에 있는 구성과 동일해야 합니다. 자세한 내용은 Prometheus 설명서의 <scrape_config>를 참조하세요.

Prometheus 지표가 로그 이벤트로 올바르게 수집되는 경우, 포함된 지표 형식 설정이 로그 이벤트에 추가되어 CloudWatch 지표를 생성하는지 확인합니다.

```
"CloudWatchMetrics":[
   {
      "Metrics":[
         {
            "Name":"envoy_http_downstream_cx_destroy_remote_active_rq"
         }
      ],
      "Dimensions":[
         [
            "ClusterName",
            "Namespace"
         ]
      ],
      "Namespace":"ContainerInsights/Prometheus"
   }
],
```

포함된 지표 형식에 대한 자세한 내용은 사양: 임베디드 지표 형식 단원을 참조하세요.

로그 이벤트에 임베디드 지표 형식이 없는 경우 CloudWatch 에이전트 설치 YAML 파일의 `config map: prometheus-cwagentconfig` 섹션에서 `metric_declaration` 섹션이 올바르게 구성되었는지 확인합니다. 자세한 내용은 새로운 Prometheus 스크레이프 대상을 추가하기 위한 튜토리얼: Prometheus API 서버 지표 단원을 참조하십시오.

Application Insights와 통합

Amazon CloudWatch Application Insights는 애플리케이션 리소스와 기술 스택 전반에 걸쳐 애플리케이션을 모니터링하고 주요 지표, 로그, 경보를 식별 및 설정하는 데 도움이 됩니다. 자세한 내용은 Amazon CloudWatch Application Insights 단원을 참조하십시오.

Application Insights를 활성화하여 컨테이너화 애플리케이션 및 마이크로서비스에서 추가 데이터를 수집할 수 있습니다. 아직 Application Insights를 활성화하지 않은 경우 Container Insights 대시보드의 성능 보기 아래에 있는 Application Insights 자동 구성(Auto-configure Application Insights)을 선택해 활성화할 수 있습니다.

컨테이너화된 애플리케이션을 모니터링하도록 CloudWatch Application Insights를 이미 설정한 경우 Application Insights 대시보드가 Container Insights 대시보드 아래에 표시됩니다.

Application Insights 및 컨테이너화 애플리케이션에 대한 자세한 내용은 Amazon ECS 및 Amazon EKS 리소스 모니터링을 위한 Application Insights 활성화 섹션을 참조하세요.

Container Insights 내에서 Amazon ECS 수명 주기 이벤트 보기

Container Insights 콘솔 내에서 Amazon ECS 수명 주기 이벤트를 볼 수 있습니다. 이렇게 하면 단일 보기에서 컨테이너 지표, 로그 및 이벤트의 상관 관계를 지정하여 보다 완전한 운영 가시성을 확보할 수 있습니다.

이벤트에는 컨테이너 인스턴스 상태 변경 이벤트, 태스크 상태 변경 이벤트 및 서비스 작업 이벤트가 포함됩니다. 이러한 이벤트는 Amazon ECS에서 Amazon EventBridge로 자동 전송되며 이벤트 로그 형식으로 CloudWatch에서도 수집됩니다. 이러한 이벤트에 대한 자세한 내용을 알아보려면 Amazon ECS 이벤트를 참조하세요.

표준 Container Insights 요금은 Amazon ECS 수명 주기 이벤트에 적용됩니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

수명 주기 이벤트의 테이블을 구성하고 클러스터에 대한 규칙을 생성하려면 `events:PutRule`, `events:PutTargets` 및 `logs:CreateLogGroup` 권한이 있어야 합니다. 또한 EventBridge가 로그 스트림을 생성하고 로그를 CloudWatch Logs로 전송하도록 지원하는 리소스 정책이 있는지도 확인해야 합니다. 이 리소스 정책이 없는 경우 다음 명령을 입력하여 생성할 수 있습니다.

```
aws --region region logs put-resource-policy --policy-name 'EventBridgeCloudWatchLogs'
 --policy-document '{
    "Statement": [
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": ["events.amazonaws.com", "delivery.logs.amazonaws.com"]
            },
            "Resource": "arn:aws:logs:region:account-id:log-group:/aws/events/ecs/
containerinsights/*:*",
            "Sid": "TrustEventBridgeToStoreECSLifecycleLogEvents"
        }
    ],
    "Version": "2012-10-17"
}'
```

다음 명령을 사용하여 이 정책이 이미 있는지 확인하고 정책이 제대로 연결되었는지 확인할 수 있습니다.

```
aws logs describe-resource-policies --region region --output json
```

수명 주기 이벤트 테이블을 보려면 `events:DescribeRule`, `events:ListTargetsByRule` 및 `logs:DescribeLogGroups` 권한이 있어야 합니다.

CloudWatch Container Insights 콘솔에서 Amazon ECS 수명 주기 이벤트 보기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. Insights, Container Insights를 선택합니다.
3. 성능 대시보드 보기를 선택합니다.
4. 다음 드롭다운에서 ECS Clusters(ECS 클러스터), ECS Services(ECS 서비스) 또는 ECS Tasks(ECS 태스크)를 선택합니다.
5. 이전 단계에서 ECS Services(ECS 서비스) 또는 ECS Tasks(ECS 태스크)를 선택한 경우 Lifecycle events(수명 주기 이벤트) 탭을 선택합니다.
6. 페이지 하단에 Configure lifecycle events(수명 주기 이벤트 구성)가 표시되면 이를 선택하여 클러스터에 대한 EventBridge 규칙을 생성합니다.

   이벤트는 Container Insights 창 아래와 Application Insights 섹션 위에 표시됩니다. 추가 분석을 실행하고 이러한 이벤트에 대한 추가 시각화를 생성하려면 Lifecycle Events(수명 주기 이벤트) 테이블에서 View in Logs Insights(Logs Insights에서 보기)를 선택합니다.

Container Insights 문제 해결

Container Insights에 문제가 있는 경우 다음 단원을 참조하면 도움이 됩니다.

Amazon EKS 또는 Kubernetes에서 배포 실패

에이전트가 Kubernetes 클러스터에 올바르게 배포하지 않는 경우 다음을 수행해 보세요.

- 다음 명령을 실행하여 Pod 목록을 가져옵니다.

  ```
  kubectl get pods -n amazon-cloudwatch
  ```

- 다음 명령을 실행하고 출력 하단에서 이벤트를 확인합니다.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- 다음 명령을 실행하여 로그를 확인합니다.

```
kubectl logs pod-name -n amazon-cloudwatch
```

권한이 없는 패닉: Kubelet에서 cadvisor 데이터를 검색할 수 없음

`Unauthorized panic: Cannot retrieve cadvisor data from kubelet` 오류가 표시되면서 배포에 실패한 경우 kubelet 에 Webhook 인증 모드가 활성화되어 있지 않을 수 있습니다. 이는 Container Insights에 필요한 모드입니다. 자세한 내용은 사전 조건 확인 단원을 참조하십시오.

삭제하고 다시 생성한 Amazon ECS의 클러스터에 Container Insights 배포

Container Insights가 사용되지 않은 기존 Amazon ECS 클러스터를 삭제하고 같은 이름으로 다시 생성하는 경우 다시 생성하는 시점에는 이 새 클러스터에서 Container Insights를 사용할 수 없습니다. 다시 생성한 후 다음 명령을 입력해야만 활성화할 수 있습니다.

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=enabled
```

잘못된 엔드포인트 오류

다음과 유사한 오류 메시지가 표시되는 경우 사용하는 명령의 *cluster-name* 및 *region-name* 같은 자리 표시자를 모두 배포에 해당하는 올바른 정보로 바꿨는지 확인합니다.

```
"log": "2020-04-02T08:36:16Z E! cloudwatchlogs: code: InvalidEndpointURL, message:
 invalid endpoint uri, original error: &url.Error{Op:\"parse\", URL:\"https://
logs.{{region_name}}.amazonaws.com/\", Err:\"{\"}, &awserr.baseError{code:
\"InvalidEndpointURL\", message:\"invalid endpoint uri\", errs:[]error{(*url.Error)
(0xc0008723c0)}}\n",
```

콘솔에 지표가 나타나지 않음

AWS Management Console에 Container Insights 지표가 보이지 않는 경우 Container Insights 설정을 완료했는지 확인합니다. Container Insights가 완전히 설정되지 않으면 지표가 나타나지 않습니다. 자세한 내용은 Container Insights 설정 단원을 참조하십시오.

클러스터를 업그레이드한 후 Amazon EKS 또는 Kubernetes에서 포드 지표가 누락됨

이 섹션은 CloudWatch 에이전트를 새 클러스터 또는 업그레이드된 클러스터에 DaemonSet로 배포한 후 전체 또는 일부 포드 지표가 누락되거나 `W! No pod metric collected` 메시지와 함께 오류 로그가 표시되는 경우에 유용할 수 있습니다.

이러한 오류는 containerd 또는 docker systemd cgroup 드라이버와 같은 컨테이너 런타임의 변경으로 인해 발생할 수 있습니다. 일반적으로 호스트의 containerd 소켓이 컨테이너에 탑재되도록 배포 매니페스트를 업데이트함으로써 이 문제를 해결할 수 있습니다. 다음 예를 참조하세요.

```
# For full example see https://github.com/aws-samples/amazon-cloudwatch-container-
insights/blob/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
container-insights-monitoring/cwagent/cwagent-daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: cloudwatch-agent
  namespace: amazon-cloudwatch
spec:
  template:
    spec:
      containers:
        - name: cloudwatch-agent
# ...
          # Don't change the mountPath
          volumeMounts:
# ...
            - name: dockersock
              mountPath: /var/run/docker.sock
              readOnly: true
            - name: varlibdocker
              mountPath: /var/lib/docker
              readOnly: true
            - name: containerdsock # NEW mount
              mountPath: /run/containerd/containerd.sock
              readOnly: true
# ...
      volumes:
# ...
        - name: dockersock
          hostPath:
            path: /var/run/docker.sock
        - name: varlibdocker
```

```
          hostPath:
            path: /var/lib/docker
        - name: containerdsock # NEW volume
          hostPath:
            path: /run/containerd/containerd.sock
```

Amazon EKS에 Bottlerocket을 사용할 때 포드 지표가 없음

Bottlerocket은 AWS에서 컨테이너를 실행하기 위해 특별히 구축한 Linux 기반 오픈 소스 운영 체제입니다.

Bottlerocket은 호스트에서 다른 `containerd` 경로를 사용하므로 볼륨을 해당 위치로 변경해야 합니다. 그러지 않으면 `W! No pod metric collected`가 포함된 오류가 로그에 표시됩니다. 다음 예를 참조하세요.

```
volumes:
  # ...
    - name: containerdsock
      hostPath:
        # path: /run/containerd/containerd.sock
        # bottlerocket does not mount containerd sock at normal place
        # https://github.com/bottlerocket-os/bottlerocket/
commit/91810c85b83ff4c3660b496e243ef8b55df0973b
        path: /run/dockershim.sock
```

Amazon EKS 또는 Kubernetes에 containerd 런타임을 사용할 때 컨테이너 파일 시스템 지표가 없음

이는 알려진 문제이며, 커뮤니티 기여자가 이 문제를 해결하기 위해 노력하고 있습니다. 자세한 내용은 GitHub에서 containerd의 디스크 사용량 지표 및 containerd용 cadvisor에서 컨테이너 파일 시스템 지표를 지원하지 않음을 참조하세요.

Prometheus 지표를 수집할 때 CloudWatch 에이전트에서 예상치 못한 로그 볼륨 증가

이는 CloudWatch 에이전트 버전 1.247347.6b250880에 도입된 회귀였습니다. 이 회귀는 최신 버전의 에이전트에서 이미 수정되었습니다. 따라서 이 문제의 영향은 고객이 CloudWatch 에이전트 자체의 로그를 수집하고 Prometheus도 사용하는 시나리오로 제한되었습니다. 자세한 내용은 GitHub에서 [prometheus] 에이전트가 로그에 스크레이프한 모든 지표를 인쇄하고 있음을 참조하세요.

릴리스 정보에 언급된 최신 Docker 이미지를 DockerHub에서 찾을 수 없음

내부적으로 실제 릴리스를 시작하기 전에 Github에서 릴리스 정보와 태그를 업데이트합니다. Github에 버전 번호를 올린 후 레지스트리에서 최신 Docker 이미지를 확인하는 데 일반적으로 1~2주가 걸립니다. CloudWatch 에이전트 컨테이너 이미지에 대한 야간 릴리스는 없습니다. 다음 위치의 소스에서 직접 이미지를 구축할 수 있습니다. https://github.com/aws/amazon-cloudwatch-agent/tree/main/amazon-cloudwatch-container-insights/cloudwatch-agent-dockerfile

CloudWatch 에이전트의 CrashLoopBackoff 오류

CloudWatch 에이전트에 대해 `CrashLoopBackOff` 오류가 표시되면 IAM 권한이 정확하게 설정되어 있는지 확인합니다. 자세한 내용은 사전 조건 확인 단원을 참조하십시오.

CloudWatch 에이전트 또는 Fluentd 포드가 보류 상태에서 멈춤

`Pending` 상태에서 멈춰 있거나 `FailedScheduling` 오류가 있는 CloudWatch 에이전트 또는 Fluentd 포드가 있는 경우 에이전트에 필요한 코어 수와 RAM 양을 기반으로 노드에 충분한 컴퓨팅 리소스가 있는지 확인합니다. 다음 명령을 입력하여 Pod를 설명합니다.

```
kubectl describe pod cloudwatch-agent-85ppg -n amazon-cloudwatch
```

자체 CloudWatch 에이전트 Docker 이미지 구축

https://github.com/aws-samples/amazon-cloudwatch-container-insights/blob/latest/cloudwatch-agent-dockerfile/Dockerfile에 있는 Dockerfile을 참조하여 자체 CloudWatch 에이전트 Docker 이미지를 구축할 수 있습니다.

Dockerfile은 `docker buildx`를 사용한 직접 다중 아키텍처 이미지 구축을 지원합니다.

컨테이너에 다른 CloudWatch 에이전트 기능 배포

CloudWatch 에이전트를 사용하여 컨테이너에 추가 모니터링 기능을 배포할 수 있습니다. 이러한 기능은 다음과 같습니다.

- 임베디드 지표 형식 - 자세한 내용은 로그 내에 지표 포함 단원을 참조하세요.
- StatsD - 자세한 내용은 StatsD를 사용하여 사용자 지정 지표 검색 단원을 참조하세요.

지침 및 필요한 파일은 GitHub의 다음 위치에 있습니다.

- Amazon ECS 컨테이너의 경우 배포 모드를 기반으로 한 Amazon ECS 태스크 정의 예를 참조하세요.
- Amazon EKS 및 Kubernetes 컨테이너의 경우 배포 모드를 기반으로 한 Kubernetes YAML 파일 예를 참조하세요.

Lambda Insights

CloudWatch Lambda Insights는 AWS Lambda에서 실행되는 서버리스 애플리케이션에 대한 모니터링 및 문제 해결 솔루션입니다. 이 솔루션은 CPU 시간, 메모리, 디스크, 네트워크를 비롯한 시스템 수준 지표를 수집하고 집계하며 요약합니다. 또한 콜드 스타트 및 Lambda 작업자 종료와 같은 진단 정보를 수집, 집계 및 요약하여 Lambda 함수 관련 문제를 격리하고 신속하게 해결할 수 있도록 지원합니다.

Lambda Insights는 Lambda 계층으로 제공되는 새로운 CloudWatch Lambda 익스텐션을 사용합니다. Lambda 함수에 이 익스텐션을 설치하면 Lambda 익스텐션은 시스템 수준 지표를 수집하며 해당 Lambda 함수가 호출될 때마다 단일 성능 로그 이벤트를 내보냅니다. CloudWatch는 포함된 지표 서식을 사용하여 로그 이벤트에서 지표를 추출합니다.

Lambda 익스텐션에 대한 자세한 내용은 AWS Lambda 익스텐션 사용 단원을 참조하세요. 포함된 지표 형식에 대한 자세한 내용은 로그 내에 지표 포함 단원을 참조하세요.

Lambda 익스텐션을 지원하는 Lambda 런타임을 사용하는 Lambda 함수와 함께 Lambda Insights를 사용할 수 있습니다. 이러한 런타임 목록은 Lambda 익스텐션 API 단원을 참조하세요.

요금

Lambda Insights를 활성화한 각 Lambda 함수에 대해 사용자는 지표 및 로그에 사용한 만큼만 비용을 지불합니다. 요금 예는 Amazon CloudWatch 요금을 참조하세요.

Lambda 확장 기능에서 사용한 실행 시간에 대해 1밀리초 단위로 요금이 청구됩니다. Lambda 요금에 대한 자세한 내용은 AWS Lambda 요금을 참조하세요.

Lambda Insights 시작하기

Lambda 함수에서 Lambda Insights를 사용하려면 Lambda 콘솔에서 원클릭 토글을 사용하면 됩니다. 또는 AWS CLI, AWS CloudFormation, AWS Serverless Application Model CLI 또는 AWS Cloud Development Kit (AWS CDK)를 사용해도 됩니다.

다음 단원에서는 이러한 단계를 완료하기 위한 자세한 지침을 제공합니다.

주제

사용 가능한 Lambda Insights 익스텐션 버전

이 단원에서는 Lambda Insights 익스텐션 버전과 각 AWS 리전에서 이러한 익스텐션에 사용할 ARN을 설명합니다.

주제

x86-64 플랫폼

이 섹션에서는 x86-64 플랫폼용 Lambda Insights 익스텐션 버전과 각 AWS 리전에서 이러한 익스텐션에 사용할 ARN을 나열합니다.

> **Important**
>
> Lambda Insights 확장 1.0.317.0 이상은 Amazon Linux 1을 지원하지 않습니다.

1.0.317.0

버전 1.0.317.0에는 Amazon Linux 1 플랫폼에 대한 지원 제거 및 버그 수정이 포함되어 있습니다. 또한 AWS GovCloud (US) 리전에 대한 지원도 포함됩니다.

버전 1.0.317.0용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:52
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:52
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:52
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:52
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:43
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:43
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:25
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:29
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:20
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:50
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:33
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:51

지역	ARN
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:52
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:52
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:79
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:51
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:12
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:42
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:42
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:52
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:52
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:52
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:43
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:51
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:27

지역	ARN
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:49
유럽(취리히)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:26
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:20
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:43
중동(UAE)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:26
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:51
AWS GovCloud(미국 동부)	arn:aws-us-gov:lambda:us-gov-east-1:122132214140:layer:LambdaInsightsExtension:19
AWS GovCloud(미국 서부)	arn:aws-us-gov:lambda:us-gov-west-1:751350123760:layer:LambdaInsightsExtension:19

1.0.295.0

버전 1.0.295.0에는 호환되는 모든 런타임에 대한 종속성 업데이트가 포함되어 있습니다.

버전 1.0.295.0용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:51

지역	ARN
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:51
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:51
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:51
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:42
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:42
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:24
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:28
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:19
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:49
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:32
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:50
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:51
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:51

지역	ARN
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:78
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:50
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:11
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:41
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:41
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:51
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:51
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:51
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:42
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:50
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:26
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:48
유럽(취리히)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:25

지역	ARN
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:19
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:42
중동(UAE)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:25
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:50

1.0.275.0

버전 1.0.275.0에는 호환되는 모든 런타임에 대한 중요한 종속성 업데이트가 포함되어 있습니다.

버전 1.0.275.0용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:49
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:49
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:49
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:49
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:40

지역	ARN
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:40
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:22
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:26
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:17
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:47
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:30
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:48
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:49
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:49
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:76
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:48
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:9
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:39

지역	ARN
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:39
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:49
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:49
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:49
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:40
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:48
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:24
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:46
유럽(취리히)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:23
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:17
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:40
중동(UAE)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:23
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:48

1.0.273.0

버전 1.0.273.0에는 호환되는 모든 런타임에 대한 중요한 버그 수정이 포함되어 있으며 캐나다 서부(캘거리)에 대한 지원이 추가되었습니다.

버전 1.0.273.0용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:45
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:45
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:45
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:45
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:35
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:35
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:17
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:21
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:12

지역	ARN
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:43
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:26
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:44
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:45
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:45
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:72
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:44
캐나다 서부(캘거리)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:4
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:36
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:36
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:45
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:45
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:45

지역	ARN
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:35
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:44
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:19
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:42
유럽(취리히)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:17
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:12
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:35
중동(UAE)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:18
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:44

1.0.229.0

버전 1.0.229.0에는 호환되는 모든 런타임에 대한 중요한 버그 수정이 포함되어 있으며 이스라엘(텔아비브) 리전에 대한 지원이 추가되었습니다.

버전 1.0.229.0용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:38
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:38
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:38
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:38
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:28
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:28
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:10
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:14
아시아 태평양(멜버른)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:5
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:36
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:19
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:37

지역	ARN
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:38
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:38
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:60
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:37
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:29
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:29
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:38
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:38
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:38
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:28
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:37
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:12
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:35

지역	ARN
유럽(취리히)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:11
이스라엘(텔아비브)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:5
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:28
중동(UAE)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:11
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:37

1.0.178.0

버전 1.0.178.0은 다음 AWS 리전에 대한 지원을 추가합니다.

- 아시아 태평양(하이데라바드)
- 아시아 태평양(자카르타)
- 유럽(스페인)
- 유럽(취리히)
- 중동(UAE)

버전 1.0.178.0용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:35

지역	ARN
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:33
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:33
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:33
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:25
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:25
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:8
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:11
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:31
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:2
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:32
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:33
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:33
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:50

지역	ARN
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:32
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:26
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:26
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:35
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:33
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:33
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:25
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:32
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:10
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:30
유럽(취리히)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:7
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:25
중동(UAE)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:9

지역	ARN
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:32

1.0.143.0

버전 1.0.143.0에는 Python 3.7 및 Go 1.x와의 호환성의 버그 수정이 포함되어 있습니다. Python 3.6 Lambda 런타임은 더 이상 사용되지 않습니다. 자세한 내용은 Lambda 런타임을 참조하세요.

1.0.143.0 버전용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:21
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:21
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:20
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:21
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:13
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:13
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:21

지역	ARN
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:2
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:20
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:21
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:21
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:32
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:20
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:14
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:14
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:21
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:21
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:21
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:13
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:20

지역	ARN
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:20
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:13
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:20

1.0.135.0

버전 1.0.135.0에는 Lambda Insights가 디스크 및 파일 설명자 사용량을 수집하고 보고하는 방법에 대한 버그 수정이 포함되어 있습니다. 이전 익스텐션 버전에서는 함수가 실행되는 동안 tmp_free 지표가 /tmp 디렉터리에서 사용 가능한 최대 공간을 보고했습니다. 이 버전은 지표를 변경하여 최솟값을 보고하므로 디스크 사용량을 평가할 때 더 유용합니다. tmp 디렉터리 스토리지 할당량에 대한 자세한 내용은 Lambda 할당량 섹션을 참조하세요.

이제 버전 1.0.135.0에서도 운영 체제 수준을 보고하지 않고 프로세스 전반의 최댓값으로 파일 설명자 사용량을 보고합니다(fd_use과 fd_max).

1.0.135.0 버전용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:18
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:18
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:18

지역	ARN
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:18
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:11
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:11
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:18
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:1
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:18
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:18
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:18
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:25
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:18
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:11
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:11
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:18

지역	ARN
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:18
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:18
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:11
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:18
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:18
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:11
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:18

1.0.119.0

1.0.119.0 버전용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:16
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:16

지역	ARN
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:16
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:16
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:9
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:9
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:16
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:16
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:16
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:16
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:23
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:16
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:9
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:9
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:16

지역	ARN
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:16
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:16
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:9
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:16
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:16
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:9
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:16

1.0.98.0

이 버전은 불필요한 로깅을 제거하고 AWS Serverless Application Model CLI 로컬 호출 관련 문제도 해결합니다. 이 문제에 대한 자세한 내용은 LambdaInsightsExtension을 추가하면 'sam local invoke'로 시간 초과가 발생함을 참조하세요.

1.0.98.0 버전용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:14

지역	ARN
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:14
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:14
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:14
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:8
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:8
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:14
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:14
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:14
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:14
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:14
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:14
중국(베이징)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:8
중국(닝샤)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:8

지역	ARN
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:14
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:14
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:14
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:8
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:14
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:14
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:8
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:14

1.0.89.0

이 버전은 함수 호출의 시작을 항상 나타내도록 성능 이벤트 타임스탬프를 수정합니다.

1.0.89.0 버전용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:12

지역	ARN
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:12
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:12
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:12
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:12
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:12
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:12
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:12
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:12
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:12
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:12
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:12
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:12
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:12

지역	ARN
유럽(스톡홀름)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:12`
남아메리카(상파울루)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:12`

1.0.86.0

익스텐션 버전 1.0.54.0에서는 메모리 지표가 때로 잘못 보고되고 때로는 100%보다 높았습니다. 1.0.86.0 버전은 Lambda 플랫폼 지표와 동일한 이벤트 데이터를 사용하여 메모리 측정 문제를 수정합니다. 즉, 기록된 메모리 지표 값이 크게 변경될 수도 있습니다. 이는 새로운 Lambda Logs API 사용을 통해 가능합니다. 이렇게 하면 Lambda 샌드박스 메모리 사용량을 더 정확하게 측정할 수 있습니다. 그러나 함수 샌드박스가 시간이 초과되어 결과적으로 스핀다운되는 경우 Lambda Logs API가 플랫폼 보고서 이벤트를 전달할 수 없다는 점에 유의해야 합니다. 이 경우 Lambda Insights가 호출 지표를 기록할 수 없습니다. Lambda Logs API에 대한 자세한 내용은 AWS Lambda Logs API 단원을 참조하세요.

1.0.86.0 버전의 새로운 기능

- Lambda Logs API를 사용하여 메모리 지표를 수정합니다. 이는 메모리 통계가 100%보다 컸던 이전 버전의 문제를 해결합니다.
- `Init Duration`을 새 CloudWatch 지표로 도입합니다.
- 호출 ARN을 사용하여 별칭 및 호출된 버전의 [버전(version)] 측정기준을 추가합니다. Lambda 별칭 또는 버전을 사용하여 증분 배포(예: 블루-그린 배포)를 달성할 경우 호출된 별칭을 기반으로 지표를 볼 수 있습니다. 함수가 별칭 또는 버전을 사용하지 않는 경우 [버전(version)] 측정기준이 적용되지 않습니다. 자세한 내용은 Lambda 함수 별칭 단원을 참조하세요.
- `billed_mb_ms field`를 성능 이벤트에 추가하여 호출당 비용을 표시합니다. 이때 프로비저닝된 동시성과 연결된 비용은 고려하지 않습니다.
- `billed_duration` 및 `duration` 필드를 성능 이벤트에 추가합니다.

1.0.86.0 버전용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:11
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:11
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:11
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:11
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:11
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:11
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:11
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:11
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:11
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:11
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:11
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:11

지역	ARN
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:11
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:11
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:11
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:11

1.0.54.0

1.0.54.0 버전은 Lambda Insights 익스텐션의 초기 릴리스였습니다.

1.0.54.0 버전용 ARN

다음 표에는 익스텐션을 사용할 수 있는 각 AWS 리전에서 이 익스텐션 버전에 사용할 ARN이 나와 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:2
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:2
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:2
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:2
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:2

지역	ARN
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:2
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:2
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:2
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:2
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:2
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:2
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:2
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:2
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:2
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:2
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:2

ARM64 플랫폼

이 섹션에서는 ARM64 플랫폼용 Lambda Insights 익스텐션 버전과 각 AWS 리전에서 이러한 익스텐션에 사용할 ARN을 나열합니다.

> **Important**
>
> Lambda Insights 확장 1.0.317.0 이상은 Amazon Linux 1을 지원하지 않습니다.

1.0.317.0

버전 1.0.317.0에는 Amazon Linux 1 플랫폼에 대한 지원 제거 및 버그 수정이 포함되어 있습니다. 또한 AWS GovCloud (US) 리전에 대한 지원도 포함됩니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:19
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:21
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:17
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:19
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:17
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:17
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension-Arm64:5
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:17
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:21

지역	ARN
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:16
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:19
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:19
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:30
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:17
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:19
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:19
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:19
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:17
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:17
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension-Arm64:5
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:17

지역	ARN
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:17
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:17
AWS GovCloud(미국 동부)	arn:aws-us-gov:lambda:us-gov-east-1:122132214140:layer:LambdaInsightsExtension-Arm64:1
AWS GovCloud(미국 서부)	arn:aws-us-gov:lambda:us-gov-west-1:751350123760:layer:LambdaInsightsExtension-Arm64:1

1.0.295.0

버전 1.0.295.0에는 호환되는 모든 런타임에 대한 종속성 업데이트가 포함되어 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:20
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:16
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:16

지역	ARN
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension-Arm64:4
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:16
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:20
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:15
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:17
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:29
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:16

지역	ARN
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:16
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension-Arm64:4
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:16
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:16

1.0.275.0

버전 1.0.275.0에는 호환되는 모든 런타임에 대한 버그 수정과 유럽(스페인) 및 아시아 태평양(하이데라바드) 리전에 대한 지원이 포함되어 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:16
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:14

지역	ARN
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:14
아시아 태평양(하이데라바드)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:14
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:13
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:15
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:16
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:27
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:16

지역	ARN
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:14
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:14
유럽(스페인)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension-Arm64:2
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:14
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:14

1.0.273.0

버전 1.0.273.0에는 호환되는 모든 런타임에 대한 버그 수정이 포함되어 있습니다.

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:14
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:9
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:12

지역	ARN
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:9
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:9
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:9
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:9
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:11
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:12
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:23
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:10
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:12

지역	ARN
유럽(밀라노)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:9
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:10
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:10
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:9
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:10

1.0.229.0

버전 1.0.229.0에는 호환되는 모든 런타임에 대한 버그 수정이 포함되어 있습니다. 또한 다음 리전에 대한 지원도 추가합니다.

- 미국 서부(캘리포니아 북부)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(자카르타)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 캐나다(중부)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스톡홀름)
- 중동(바레인)
- 남아메리카(상파울루)

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:7
미국 서부(캘리포니아 북부)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:3
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:5
아프리카(케이프타운)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(홍콩)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(자카르타)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:7
아시아 태평양(오사카)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(서울)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:4
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:5

지역	ARN
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:11
캐나다(중부)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:3
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:5
유럽(스페인)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:2
유럽(파리)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:3
유럽(스톡홀름)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:3
중동(바레인)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:2
남아메리카(상파울루)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:3

1.0.135.0

버전 1.0.135.0에는 Lambda Insights가 디스크 및 파일 설명자 사용량을 수집하고 보고하는 방법에 대한 버그 수정이 포함되어 있습니다. 이전 익스텐션 버전에서는 함수가 실행되는 동안 `tmp_free` 지표가 `/tmp` 디렉터리에서 사용 가능한 최대 공간을 보고했습니다. 이 버전은 지표를 변경하여 최솟값을 보고하므로 디스크 사용량을 평가할 때 더 유용합니다. `tmp` 디렉터리 스토리지 할당량에 대한 자세한 내용은 Lambda 할당량 섹션을 참조하세요.

이제 버전 1.0.135.0에서도 운영 체제 수준을 보고하지 않고 프로세스 전반의 최댓값으로 파일 설명자 사용량을 보고합니다(fd_use과 fd_max).

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:2
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:2
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:2

1.0.119.0

지역	ARN
미국 동부(버지니아 북부)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
미국 동부(오하이오)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:1
미국 서부(오레곤)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:1
아시아 태평양(뭄바이)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
아시아 태평양(싱가포르)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
아시아 태평양(시드니)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:1
아시아 태평양(도쿄)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
유럽(프랑크푸르트)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
유럽(아일랜드)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
유럽(런던)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:1

콘솔을 사용하여 기존 Lambda 함수에서 Lambda Insights 사용 설정

Lambda 콘솔의 다음 단계를 사용하여 기존 Lambda 함수에서 Lambda Insights를 사용 설정할 수 있습니다.

Lambda 함수에서 Lambda Insights를 사용 설정하려면

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 함수 이름을 선택하고 다음 화면에서 구성(Configuration)을 선택합니다.
3. 구성 탭의 왼쪽 탐색 메뉴에서 모니터링 및 운영 도구, 편집을 차례로 선택합니다.

   모니터링 도구를 편집할 수 있는 화면으로 이동합니다.
4. Lambda Insights 향상된 모니터링에서 편집을 선택합니다.
5. CloudWatch Lambda Insights에서 향상된 모니터링을 선택한 다음 저장을 선택합니다.

AWS CLI를 사용하여 기존 Lambda 함수에서 Lambda Insights 사용 설정

다음 단계에 따라 AWS CLI를 사용하여 기존 Lambda 함수에서 Lambda Insights를 사용 설정할 수 있습니다.

1단계: 함수 권한 업데이트

함수의 권한을 업데이트하려면

- 다음 명령을 입력하여 [CloudWatchLambdaInsightsExecutionRolePolicy] 관리형 IAM 정책을 함수의 실행 역할에 연결합니다.

```
aws iam attach-role-policy \
--role-name function-execution-role \
--policy-arn "arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy"
```

2단계: Lambda 익스텐션 설치

다음 명령을 입력하여 Lambda 익스텐션을 설치합니다. `layers` 파라미터의 ARN 값을 사용하려는 리전 및 익스텐션 버전과 일치하는 ARN으로 바꿉니다. 자세한 내용은 사용 가능한 Lambda Insights 익스텐션 버전 단원을 참조하세요.

```
aws lambda update-function-configuration \
   --function-name function-name \
   --layers "arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:14"
```

3단계: CloudWatch Logs VPC 엔드포인트 사용 설정

이 단계는 CloudWatch Logs Virtual Private Cloud(VPC) 엔드포인트를 아직 구성하지 않은 경우에 그리고 인터넷에 액세스할 수 없는 프라이빗 서브넷에서 실행되는 함수에만 필요합니다.

이 단계를 수행해야 하는 경우 다음 명령을 입력합니다. 이때 자리 표시자를 VPC에 대한 정보로 바꿉니다.

자세한 내용은 인터페이스 VPC 엔드포인트와 함께 CloudWatch Logs 사용 단원을 참조하세요.

```
aws ec2 create-vpc-endpoint \
--vpc-id vpcId \
--vpc-endpoint-type Interface \
--service-name com.amazonaws.region.logs \
--subnet-id subnetId
--security-group-id securitygroupId
```

AWS SAM CLI를 사용하여 기존 Lambda 함수에서 Lambda Insights 사용 설정

다음 단계에 따라 AWS SAM AWS CLI를 사용하여 기존 Lambda 함수에서 Lambda Insights를 사용 설정할 수 있습니다.

최신 버전의 AWS SAM CLI를 아직 설치하지 않은 경우 먼저, 설치하거나 업그레이드해야 합니다. 자세한 내용은 AWS SAM CLI 설치 단원을 참조하세요.

1단계: 계층 설치

Lambda Insights 익스텐션을 모든 Lambda 함수에서 사용할 수 있도록 하려면 Lambda Insights 계층의 ARN을 사용하여 SAM 템플릿의 `Globals` 섹션에 `Layers` 속성을 추가합니다. 아래 예에서는 초기 Lambda Insights 릴리스의 계층을 사용합니다. 최신 릴리스 버전의 Lambda Insights 익스텐션 계층은 사용 가능한 Lambda Insights 익스텐션 버전 단원을 참조하세요.

```
Globals:
  Function:
    Layers:
      - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

단일 함수에 대해서만 이 계층을 사용하려면 이 예와 같이 함수에 `Layers` 속성을 추가합니다.

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
```

```
      Layers:
        - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

2단계: 관리형 정책 추가

각 함수에 대해 [CloudWatchLambdaInsightsExecutionRolePolicy] IAM 정책을 추가합니다.

AWS SAM은 글로벌 정책을 지원하지 않으므로 이 예와 같이 각 함수에서 개별적으로 정책을 사용 설정해야 합니다. 글로벌에 대한 자세한 내용은 Globals 섹션을 참조하세요.

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Policies:
        - CloudWatchLambdaInsightsExecutionRolePolicy
```

로컬에서 호출

AWS SAM CLI는 Lambda 익스텐션을 지원합니다. 그러나 로컬에서 실행되는 모든 호출은 런타임 환경을 재설정합니다. 런타임이 종료 이벤트 없이 다시 시작되기 때문에 로컬 호출에서 Lambda Insights 데이터를 사용할 수 없습니다. 자세한 내용은 릴리스 1.6.0 - AWS Lambda 익스텐션의 로컬 테스트를 위한 지원 추가를 참조하세요.

문제 해결

Lambda Insights 설치 문제를 해결하려면 Lambda 함수에 다음 환경 변수를 추가하여 디버그 로깅을 사용 설정합니다.

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Environment:
        Variables:
          LAMBDA_INSIGHTS_LOG_LEVEL: info
```

AWS CloudFormation을 사용하여 기존 Lambda 함수에서 Lambda Insights 사용 설정

다음 단계에 따라 AWS CloudFormation을 사용하여 기존 Lambda 함수에서 Lambda Insights를 사용 설정할 수 있습니다.

1단계: 계층 설치

Lambda Insights 계층 ARN 내의 `Layers` 속성에 Lambda Insights 계층을 추가합니다. 아래 예에서는 초기 Lambda Insights 릴리스의 계층을 사용합니다. 최신 릴리스 버전의 Lambda Insights 익스텐션 계층은 사용 가능한 Lambda Insights 익스텐션 버전 단원을 참조하세요.

```
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Properties:
      Layers:
        - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

2단계: 관리형 정책 추가

[CloudWatchLambdaInsightsExecutionRolePolicy] IAM 정책을 함수 실행 역할에 추가합니다.

```
Resources:
  MyFunctionExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy'
```

3단계: (선택 사항) VPC 엔드포인트 추가

이 단계는 CloudWatch Logs Virtual Private Cloud(VPC) 엔드포인트를 아직 구성하지 않은 경우에 그리고 인터넷에 액세스할 수 없는 프라이빗 서브넷에서 실행되는 함수에만 필요합니다. 자세한 내용은 인터페이스 VPC 엔드포인트와 함께 CloudWatch Logs 사용 단원을 참조하세요.

```
Resources:
  CloudWatchLogsVpcPrivateEndpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PrivateDnsEnabled: 'true'
      VpcEndpointType: Interface
      VpcId: !Ref: VPC
      ServiceName: !Sub com.amazonaws.${AWS::Region}.logs
      SecurityGroupIds:
        - !Ref InterfaceVpcEndpointSecurityGroup
      SubnetIds:
```

```
        - !Ref PublicSubnet01
        - !Ref PublicSubnet02
        - !Ref PublicSubnet03
```

AWS CDK를 사용하여 기존 Lambda 함수에서 Lambda Insights 사용 설정

다음 단계에 따라 AWS CDK를 사용하여 기존 Lambda 함수에서 Lambda Insights를 사용 설정할 수 있습니다. 다음 단계를 수행하려면 이미 AWS CDK를 사용하여 리소스를 관리하고 있어야 합니다.

이 단원의 명령은 TypeScript에 있습니다.

먼저, 함수 권한을 업데이트합니다.

```
executionRole.addManagedPolicy(
 ManagedPolicy.fromAwsManagedPolicyName('CloudWatchLambdaInsightsExecutionRolePolicy')
);
```

다음으로, Lambda 함수에 익스텐션을 설치합니다. `layerArn` 파라미터의 ARN 값을 사용하려는 리전 및 익스텐션 버전과 일치하는 ARN으로 바꿉니다. 자세한 내용은 사용 가능한 Lambda Insights 익스텐션 버전 단원을 참조하세요.

```
import lambda = require('@aws-cdk/aws-lambda');
const layerArn = 'arn:aws:lambda:us-
west-1:580247275435:layer:LambdaInsightsExtension:14';
const layer = lambda.LayerVersion.fromLayerVersionArn(this, 'LayerFromArn', layerArn);
```

필요한 경우 CloudWatch Logs의 Virtual Private Cloud(VPC) 엔드포인트를 사용 설정합니다. 이 단계는 CloudWatch Logs VPC 엔드포인트를 아직 구성하지 않은 경우에 그리고 인터넷에 액세스할 수 없는 프라이빗 서브넷에서 실행되는 함수에만 필요합니다.

```
const cloudWatchLogsEndpoint = vpc.addInterfaceEndpoint('cwl-gateway', {
  service: InterfaceVpcEndpointAwsService.CLOUDWATCH_LOGS,
});

cloudWatchLogsEndpoint.connections.allowDefaultPortFromAnyIpv4();
```

서버리스 프레임워크를 사용하여 기존 Lambda 함수에서 Lambda Insights 사용 설정

다음 단계에 따라 서버리스 프레임워크를 사용하여 기존 Lambda 함수에서 Lambda Insights를 사용 설정할 수 있습니다. 서버리스 프레임워크에 대한 자세한 내용은 serverless.com을 참조하세요.

이 작업은 서버리스용 Lambda Insights 플러그 인을 통해 수행됩니다. 자세한 내용은 serverless-plugin-lambda-insights를 참조하세요.

최신 버전의 서버리스 명령줄 인터페이스를 아직 설치하지 않은 경우 먼저, 설치하거나 업그레이드해야 합니다. 자세한 내용은 서버리스 프레임워크 오픈 소스 및 AWS 시작하기를 참조하세요.

서버리스 프레임워크를 사용하여 Lambda 함수에서 Lambda Insights를 사용 설정하려면

1. Serverless 디렉터리에서 다음 명령을 실행하여 Lambda Insights용 서버리스 플러그 인을 설치합니다.

```
npm install --save-dev serverless-plugin-lambda-insights
```

2. `serverless.yml` 파일에서 다음과 같이 `plugins` 섹션에 플러그 인을 추가합니다.

```
provider:
  name: aws
plugins:
  - serverless-plugin-lambda-insights
```

3. Lambda Insights를 사용 설정합니다.
   - serverless.yml 파일에 다음 속성을 추가하여 함수마다 개별적으로 Lambda Insights를 사용 설정할 수 있습니다.

```
functions:
  myLambdaFunction:
    handler: src/app/index.handler
    lambdaInsights: true #enables Lambda Insights for this function
```

   - `serverless.yml` 파일 내에서 다음과 같은 custom 섹션을 추가하여 모든 함수에 대해 Lambda Insights를 사용 설정할 수 있습니다.

```
custom:
  lambdaInsights:
    defaultLambdaInsights: true #enables Lambda Insights for all functions
```

4. 다음 명령을 입력하여 서버리스 서비스를 다시 배포합니다.

```
serverless deploy
```

이렇게 하면 모든 함수가 다시 배포되고 지정한 해당 함수에 대해 Lambda Insights가 사용 설정됩니다. 즉, Lambda Insights 계층을 추가하고 `arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy` IAM 정책을 사용하는 필수 권한을 연결하여 Lambda Insights를 사용 설정합니다.

Lambda 컨테이너 이미지 배포에서 Lambda Insights 사용 설정

컨테이너 이미지로 배포된 Lambda 함수에서 Lambda Insights를 사용하도록 설정하려면 Dockerfile에 줄을 추가하세요. 이 줄은 Lambda Insights 에이전트를 컨테이너 이미지의 확장 프로그램으로 설치합니다. 추가할 줄은 x86-64 컨테이너와 ARM64 컨테이너에 따라 다릅니다.

> **Note**
>
> Lambda Insights 에이전트는 Amazon Linux 2를 사용하는 Lambda 런타임에서만 지원됩니다.

주제

- x86-64 컨테이너 이미지 배포
- ARM64 컨테이너 이미지 배포

x86-64 컨테이너 이미지 배포

x86-64 컨테이너에서 실행되는 컨테이너 이미지로 배포된 Lambda 함수에서 Lambda Insights를 사용하도록 설정하려면 Dockerfile에 다음 줄을 추가하세요. 이 줄은 Lambda Insights 에이전트를 컨테이너 이미지의 확장 프로그램으로 설치합니다.

```
RUN curl -O https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension.rpm && \
    rpm -U lambda-insights-extension.rpm && \
    rm -f lambda-insights-extension.rpm
```

Lambda 함수를 생성한 후 [CloudWatchLambdaInsightsExecutionRolePolicy] IAM 정책을 함수의 실행 역할에 할당하면 Lambda Insights가 컨테이너 이미지 기반 Lambda 함수에서 사용하도록 설정됩니다.

> **Note**
>
> 이전 버전의 Lambda Insights 익스텐션을 사용하려면 위 명령의 URL을 `https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension.1.0.111.0.rpm` URL로 바꿉니다. 현재 Lambda Insights 버전 1.0.111.0 이상만 사용할 수 있습니다. 자세한 내용은 사용 가능한 Lambda Insights 익스텐션 버전 단원을 참조하세요.

Linux 서버에서 Lambda Insights 에이전트 패키지의 서명을 확인하려면

1. 다음 명령을 입력하여 퍼블릭 키를 다운로드합니다.

   ```
   shell$ wget https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
   lambda-insights-extension.gpg
   ```

2. 다음 명령을 입력하여 퍼블릭 키를 키링으로 가져옵니다.

   ```
   shell$ gpg --import lambda-insights-extension.gpg
   ```

   출력은 다음과 비슷합니다. 다음 단계에서 필요하므로 key 값을 기록해 둡니다. 이 출력 예에서 키 값은 848ABDC8입니다.

   ```
   gpg: key 848ABDC8: public key "Amazon Lambda Insights Extension" imported
   gpg: Total number processed: 1
   gpg: imported: 1  (RSA: 1)
   ```

3. 다음 명령을 입력하여 지문을 확인합니다. `key-value`를 이전 단계의 키 값으로 바꿉니다.

   ```
   shell$  gpg --fingerprint key-value
   ```

   이 명령의 출력에서 지문 문자열은 `E0AF FA11 FFF3 5BD7 349E E222 479C 97A1 848A BDC8`이어야 합니다. 문자열이 일치하지 않으면 에이전트를 설치하지 말고 AWS에 문의하세요.

4. 지문을 확인한 후 이를 사용하여 Lambda Insights 에이전트 패키지를 확인할 수 있습니다. 다음 명령을 입력하여 패키지 서명 파일을 다운로드합니다.

   ```
   shell$  wget https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
   amazon_linux/lambda-insights-extension.rpm.sig
   ```

5. 다음 명령을 입력하여 서명을 확인합니다.

```
shell$ gpg --verify lambda-insights-extension.rpm.sig lambda-insights-extension.rpm
```

출력은 다음과 같아야 합니다.

```
gpg: Signature made Thu 08 Apr 2021 06:41:00 PM UTC using RSA key ID 848ABDC8
gpg: Good signature from "Amazon Lambda Insights Extension"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: E0AF FA11 FFF3 5BD7 349E  E222 479C 97A1 848A BDC8
```

예상 출력에 신뢰할 수 있는 서명에 대한 경고가 있을 수 있습니다. 사용자 또는 사용자가 신뢰하는 사람이 서명한 키만 신뢰됩니다. 이는 서명이 잘못되었음을 의미하지 않으며, 단지 해당 사용자가 퍼블릭 키를 확인하지 않은 것입니다.

출력에 `BAD signature`가 포함된 경우 단계를 올바르게 수행했는지 확인합니다. `BAD signature` 응답이 계속되는 경우 AWS에 문의하고, 다운로드한 파일을 사용하지 마세요.

x86-64 예

이 단원에는 컨테이너 이미지 기반 Python Lambda 함수에서 Lambda Insights를 사용 설정하는 예가 포함되어 있습니다.

Lambda 컨테이너 이미지에서 Lambda Insights를 사용 설정하는 예

1. 다음과 유사한 Dockerfile을 생성합니다.

```
FROM public.ecr.aws/lambda/python:3.8

// extra lines to install the agent here
RUN curl -O https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension.rpm && \
    rpm -U lambda-insights-extension.rpm && \
    rm -f lambda-insights-extension.rpm

COPY index.py ${LAMBDA_TASK_ROOT}
CMD [ "index.handler" ]
```

2. 다음과 유사한 `index.py`라는 Python 파일을 생성합니다.

```
def handler(event, context):
  return {
    'message': 'Hello World!'
  }
```

3. Dockerfile 및 `index.py`를 동일한 디렉터리에 넣습니다. 그런 다음, 해당 디렉터리에서 다음 단계를 실행함으로써 Docker 이미지를 구축하여 Amazon ECR에 업로드합니다.

```
// create an ECR repository
aws ecr create-repository --repository-name test-repository
// build the docker image
docker build -t test-image .
// sign in to AWS
aws ecr get-login-password | docker login --username AWS --password-stdin
 "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com
// tag the image
docker tag test-image:latest "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/
test-repository:latest
// push the image to ECR
docker push "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/test-
repository:latest
```

4. 방금 생성한 Amazon ECR 이미지를 사용하여 Lambda 함수를 생성합니다.
5. [CloudWatchLambdaInsightsExecutionRolePolicy] IAM 정책을 함수의 실행 역할에 할당합니다.

ARM64 컨테이너 이미지 배포

AL2_aarch64 컨테이너(ARM64 아키텍처 사용)에서 실행되는 컨테이너 이미지로 배포된 Lambda 함수에서 Lambda Insights를 사용하도록 설정하려면 Dockerfile에 다음 줄을 추가하세요. 이 줄은 Lambda Insights 에이전트를 컨테이너 이미지의 확장 프로그램으로 설치합니다.

```
RUN curl -O https://lambda-insights-extension-arm64.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension-arm64.rpm && \
    rpm -U lambda-insights-extension-arm64.rpm && \
    rm -f lambda-insights-extension-arm64.rpm
```

Lambda 함수를 생성한 후 [CloudWatchLambdaInsightsExecutionRolePolicy] IAM 정책을 함수의 실행 역할에 할당하면 Lambda Insights가 컨테이너 이미지 기반 Lambda 함수에서 사용하도록 설정됩니다.

> **Note**
>
> 이전 버전의 Lambda Insights 익스텐션을 사용하려면 위 명령의 URL을 `https://lambda-insights-extension-arm64.s3-ap-northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.1.0.229.0.rpm` URL로 바꿉니다. 현재 Lambda Insights 버전 1.0.229.0 이상만 사용할 수 있습니다. 자세한 내용은 사용 가능한 Lambda Insights 익스텐션 버전 단원을 참조하십시오.

Linux 서버에서 Lambda Insights 에이전트 패키지의 서명을 확인하려면

1. 다음 명령을 입력하여 퍼블릭 키를 다운로드합니다.

```
shell$ wget https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/lambda-insights-extension.gpg
```

2. 다음 명령을 입력하여 퍼블릭 키를 키링으로 가져옵니다.

```
shell$ gpg --import lambda-insights-extension.gpg
```

   출력은 다음과 비슷합니다. 다음 단계에서 필요하므로 key 값을 기록해 둡니다. 이 출력 예에서 키 값은 848ABDC8입니다.

```
gpg: key 848ABDC8: public key "Amazon Lambda Insights Extension" imported
gpg: Total number processed: 1
gpg: imported: 1  (RSA: 1)
```

3. 다음 명령을 입력하여 지문을 확인합니다. `key-value`를 이전 단계의 키 값으로 바꿉니다.

```
shell$  gpg --fingerprint key-value
```

   이 명령의 출력에서 지문 문자열은 E0AF FA11 FFF3 5BD7 349E E222 479C 97A1 848A BDC8이어야 합니다. 문자열이 일치하지 않으면 에이전트를 설치하지 말고 AWS에 문의하세요.

4. 지문을 확인한 후 이를 사용하여 Lambda Insights 에이전트 패키지를 확인할 수 있습니다. 다음 명령을 입력하여 패키지 서명 파일을 다운로드합니다.

```
shell$  wget https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.rpm.sig
```

5. 다음 명령을 입력하여 서명을 확인합니다.

```
shell$ gpg --verify lambda-insights-extension-arm64.rpm.sig lambda-insights-
extension-arm64.rpm
```

출력은 다음과 같아야 합니다.

```
gpg: Signature made Thu 08 Apr 2021 06:41:00 PM UTC using RSA key ID 848ABDC8
gpg: Good signature from "Amazon Lambda Insights Extension"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: E0AF FA11 FFF3 5BD7 349E  E222 479C 97A1 848A BDC8
```

예상 출력에 신뢰할 수 있는 서명에 대한 경고가 있을 수 있습니다. 사용자 또는 사용자가 신뢰하는 사람이 서명한 키만 신뢰됩니다. 이는 서명이 잘못되었음을 의미하지 않으며, 단지 해당 사용자가 퍼블릭 키를 확인하지 않은 것입니다.

출력에 `BAD signature`가 포함된 경우 단계를 올바르게 수행했는지 확인합니다. `BAD signature` 응답이 계속되는 경우 AWS에 문의하고, 다운로드한 파일을 사용하지 마세요.

ARM64 예

이 단원에는 컨테이너 이미지 기반 Python Lambda 함수에서 Lambda Insights를 사용 설정하는 예가 포함되어 있습니다.

Lambda 컨테이너 이미지에서 Lambda Insights를 사용 설정하는 예

1. 다음과 유사한 Dockerfile을 생성합니다.

```
FROM public.ecr.aws/lambda/python:3.8
// extra lines to install the agent here
RUN curl -O https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.rpm && \
    rpm -U lambda-insights-extension-arm64.rpm && \
    rm -f lambda-insights-extension-arm64.rpm

COPY index.py ${LAMBDA_TASK_ROOT}
CMD [ "index.handler" ]
```

2. 다음과 유사한 `index.py`라는 Python 파일을 생성합니다.

```
def handler(event, context):
  return {
    'message': 'Hello World!'
  }
```

3. Dockerfile 및 `index.py`를 동일한 디렉터리에 넣습니다. 그런 다음, 해당 디렉터리에서 다음 단계를 실행함으로써 Docker 이미지를 구축하여 Amazon ECR에 업로드합니다.

```
// create an ECR repository
aws ecr create-repository --repository-name test-repository
// build the docker image
docker build -t test-image .
// sign in to AWS
aws ecr get-login-password | docker login --username AWS --password-stdin
 "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com
// tag the image
docker tag test-image:latest "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/
test-repository:latest
// push the image to ECR
docker push "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/test-
repository:latest
```

4. 방금 생성한 Amazon ECR 이미지를 사용하여 Lambda 함수를 생성합니다.
5. [CloudWatchLambdaInsightsExecutionRolePolicy] IAM 정책을 함수의 실행 역할에 할당합니다.

Lambda Insights 지표 보기

호출된 Lambda 함수에 Lambda Insights 익스텐션을 설치한 후 CloudWatch 콘솔을 사용하여 지표를 확인할 수 있습니다. 다중 함수 개요를 보거나 단일 함수에 집중할 수 있습니다.

Lambda Insights 지표 목록은 Lambda Insights가 수집하는 지표 단원을 참조하세요.

Lambda Insights 지표의 다중 함수 개요를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 [Lambda Insights]에서 [다중 함수(Multi-function)]를 선택합니다.

   페이지의 위쪽 부분에는 Lambda Insights가 사용 설정된 리전의 모든 Lambda 함수에 대한 집계된 지표가 있는 그래프가 표시됩니다. 페이지의 아래쪽에는 함수를 나열하는 테이블이 있습니다.

3. 표시되는 함수의 수를 줄이기 위해 함수 이름을 기준으로 필터링하려면 페이지 상단 근처의 상자에 함수 이름의 일부를 입력합니다.
4. 이 보기를 대시보드에 위젯으로 추가하려면 [대시보드에 추가(Add to dashboard)]를 선택합니다.

단일 함수의 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 [Lambda Insights]에서 [단일 함수(Single-function)]를 선택합니다.

   페이지의 위쪽 부분에는 선택한 함수에 대한 지표가 있는 그래프가 표시됩니다.
3. X-Ray를 사용 설정한 경우 단일 추적 ID를 선택할 수 있습니다. 그러면 해당 간접 호출에 대한 X-Ray 트레이스 맵 페이지가 열리며, 여기에서 축소하여 해당 특정 트랜잭션 처리와 관련된 분산 트레이스 및 기타 서비스를 확인할 수 있습니다. X-Ray 트레이스 맵에 대한 자세한 내용은 X-Ray 트레이스 맵 사용을 참조하세요.
4. CloudWatch Logs Insights를 열고 특정 오류를 확대하려면 페이지 하단의 테이블별 [로그 보기(View logs)]를 선택합니다.
5. 이 보기를 대시보드에 위젯으로 추가하려면 [대시보드에 추가(Add to dashboard)]를 선택합니다.

Application Insights와 통합

Amazon CloudWatch Application Insights는 애플리케이션 리소스와 기술 스택 전반에 걸쳐 애플리케이션을 모니터링하고 주요 지표, 로그, 경보를 식별 및 설정하는 데 도움이 됩니다. 자세한 내용은 Amazon CloudWatch Application Insights 단원을 참조하십시오.

Application Insights를 활성화하여 Lambda 함수에서 추가 데이터를 수집할 수 있습니다. 아직 Application Insights를 활성화하지 않은 경우 Container Insights 대시보드의 성능 보기 아래에 있는 Application Insights 자동 구성(Auto-configure Application Insights)을 선택해 활성화할 수 있습니다.

Lambda 기능을 모니터링하도록 CloudWatch Application Insights를 이미 설정한 경우 Application Insights 대시보드가 Application Insights 탭 아래에 표시됩니다.

Lambda Insights가 수집하는 지표

Lambda Insights는 해당 서비스가 설치된 Lambda 함수에서 여러 지표를 수집합니다. 이러한 지표 중 일부는 CloudWatch 지표에서 시계열 집계 데이터로 사용할 수 있습니다. 다른 지표는 시계열 데이터로 집계되지는 않지만 CloudWatch Logs Insights를 사용하여 임베디드 지표 형식 로그 항목에서 찾아볼 수 있습니다.

다음 지표는 LambdaInsights 네임스페이스의 CloudWatch 지표에서 시계열 집계 데이터로 사용할 수 있습니다.

지표 이름	측정기준	설명
cpu_total_time	function_name function_name, version	cpu_system_time 및 cpu_user_time 의 합계입니다. 단위: 밀리초
init_duration	function_name function_name, version	Lambda 실행 환경 수명 주기의 init 단계에서 소요된 시간입니다. 단위: 밀리초
memory_utilization	function_name function_name, version	함수에 할당된 메모리의 백분율로 측정된 최대 메모리입니다. 단위: 백분율
rx_bytes	function_name function_name, version	함수가 수신한 바이트 수입니다. 단위: 바이트
tmp_used		/tmp 디렉터리에 사용된 공간량입니다. 단위: 바이트
tx_bytes	function_name function_name, version	함수가 전송한 바이트 수입니다. 단위: 바이트

지표 이름	측정기준	설명
total_memory	function_name function_name, version	Lambda 함수에 할당된 메모리 양입니다. 이는 함수의 메모리 크기와 같습니다. 단위: 메가바이트
total_network	function_name function_name, version	rx_bytes 및 tx_bytes의 합계입니다. I/O 태스크를 수행하지 않는 함수의 경우에도 이 값은 일반적으로 Lambda 런타임에서 수행된 네트워크 호출로 인해 0보다 큽니다. 단위: 바이트
used_memory_max	function_name function_name, version	측정된 함수 샌드박스 메모리입니다. 단위: 메가바이트

다음 지표는 CloudWatch Logs Insights를 사용하여 임베디드 지표 형식 로그 항목에서 찾아볼 수 있습니다. CloudWatch Logs Insights에 대한 자세한 내용은 CloudWatch Logs Insights를 사용한 로그 데이터 분석 단원을 참조하세요.

포함된 지표 형식에 대한 자세한 내용은 로그 내에 지표 포함 단원을 참조하세요.

지표 이름	설명	
cpu_system_time	CPU가 커널 코드를 실행하는 데 사용한 시간입니다. 단위: 밀리초	

지표 이름	설명
cpu_total_time	cpu_system_time 및 cpu_user_time 의 합계입니다. 단위: 밀리초
cpu_user_time	CPU가 사용자 코드를 실행하는 데 사용한 시간입니다. 단위: 밀리초
fd_max	사용 가능한 최대 파일 설명자 수입니다. 단위: 수
fd_use	사용 중인 최대 파일 설명자 수입니다. 단위: 수
memory_utilization	함수에 할당된 메모리의 백분율로 측정된 최대 메모리입니다. 단위: 백분율
rx_bytes	함수가 수신한 바이트 수입니다. 단위: 바이트
tx_bytes	함수가 전송한 바이트 수입니다. 단위: 바이트
threads_max	함수 프로세스에서 사용 중인 스레드 수입니다. 함수 작성자가 런타임에 의해 생성된 초기 스레드 수를 제어하지 않습니다. 단위: 수
tmp_max	/tmp 디렉터리에 사용 가능한 공간량입니다. 단위: 바이트

지표 이름	설명
total_memory	Lambda 함수에 할당된 메모리 양입니다. 이는 함수의 메모리 크기와 같습니다. 단위: 메가바이트
total_network	rx_bytes 및 tx_bytes의 합계입니다. I/O 태스크를 수행하지 않는 함수의 경우에도 이 값은 일반적으로 Lambda 런타임에서 수행된 네트워크 호출로 인해 0보다 큽니다. 단위: 바이트
used_memory_max	측정된 함수 샌드박스 메모리입니다. 단위: 바이트

문제 해결 및 알려진 문제

문제 해결을 위한 첫 번째 단계는 Lambda Insights 익스텐션에 디버그 로깅을 사용 설정하는 것입니다. 이렇게 하려면 Lambda 함수에서 환경 변수 LAMBDA_INSIGHTS_LOG_LEVEL=info를 설정합니다. 자세한 내용은 AWS Lambda 환경 변수 사용 단원을 참조하세요.

익스텐션은 함수와 동일한 로그 그룹에 로그를 내보냅니다(/aws/lambda/*function-name*). 해당 로그를 검토하여 오류가 설정 문제와 관련이 있을 수 있는지 확인합니다.

Lambda Insights의 지표가 표시되지 않음

표시될 것으로 예상했던 Lambda Insights 지표가 표시되지 않으면 다음 가능성을 확인하세요.

- 지표가 지연될 뿐일 수 있음 - 함수가 아직 호출되지 않았거나 데이터가 아직 플러시되지 않은 경우 CloudWatch에 지표가 표시되지 않습니다. 자세한 내용은 이 단원의 뒷부분에 있는 알려진 문제를 참조하세요.
- Lambda 함수에 올바른 권한이 있는지 확인 - [CloudWatchLambdaInsightsExecutionRolePolicy] IAM 정책이 함수의 실행 역할에 할당되었는지 확인합니다.
- Lambda 런타임 확인 - Lambda Insights는 특정 Lambda 런타임만 지원합니다. 지원되는 런타임 목록은 Lambda Insights 단원을 참조하세요.

예를 들어 Java 8에서 Lambda Insights를 사용하려면 `java8` 런타임이 아니라 `java8.al2` 런타임을 사용해야 합니다.

- 네트워크 액세스 확인 - Lambda 함수가 인터넷에 액세스할 수 없는 VPC 프라이빗 서브넷에 있을 수 있으며, CloudWatch Logs에 대해 구성된 VPC 엔드포인트가 없습니다. 이 문제를 디버그하기 위해 환경 변수 `LAMBDA_INSIGHTS_LOG_LEVEL=info`를 설정할 수 있습니다.

알려진 문제

데이터 지연은 20분까지 걸릴 수 있습니다. 함수 핸들러가 완료되면 Lambda는 샌드박스를 고정하며 이에 따라 Lambda Insights 익스텐션도 고정됩니다. 함수가 실행되는 동안에는 함수 TPS를 기반으로 적응형 배치 처리 전략을 사용하여 데이터를 출력할 수 있습니다. 그러나 함수 호출이 장기간 중지되는데 버퍼에 여전히 이벤트 데이터가 있는 경우 이 데이터는 Lambda가 유휴 샌드박스를 종료할 때까지 지연될 수 있습니다. Lambda가 샌드박스를 종료하면 버퍼링된 데이터를 플러시할 수 있습니다.

원격 측정 이벤트 예

Lambda Insights가 사용 설정된 Lambda 함수를 호출할 때마다 단일 로그 이벤트가 `/aws/lambda-insights` 로그 그룹에 기록됩니다. 각 로그 이벤트에는 임베디드 지표 형식의 지표가 포함됩니다. 포함된 지표 형식에 대한 자세한 내용은 로그 내에 지표 포함 단원을 참조하세요.

이러한 로그 이벤트를 분석하기 위해 다음 방법을 사용할 수 있습니다.

- Lambda Insights 지표 보기 단원에 설명된 것처럼 CloudWatch 콘솔의 Lambda Insights 섹션을 활용합니다.
- CloudWatch Logs Insights를 사용하여 로그 이벤트를 쿼리합니다. 자세한 내용은 CloudWatch Logs Insights를 사용한 로그 데이터 분석을 참조하세요.
- `LambdaInsights` 네임스페이스에 수집된 지표를 검토하고 CloudWatch 지표를 사용하여 이를 그래프로 표시합니다.

다음은 임베디드 지표 형식이 있는 Lambda Insights 로그 이벤트의 예입니다.

```
{
    "_aws": {
        "Timestamp": 1605034324256,
        "CloudWatchMetrics": [
            {
                "Namespace": "LambdaInsights",
```

```
                "Dimensions": [
                    [ "function_name" ],
                    [ "function_name", "version" ]
                ],
                "Metrics": [
                    { "Name": "memory_utilization", "Unit": "Percent" },
                    { "Name": "total_memory", "Unit": "Megabytes" },
                    { "Name": "used_memory_max", "Unit": "Megabytes" },
                    { "Name": "cpu_total_time", "Unit": "Milliseconds" },
                    { "Name": "tx_bytes", "Unit": "Bytes" },
                    { "Name": "rx_bytes", "Unit": "Bytes" },
                    { "Name": "total_network", "Unit": "Bytes" },
                    { "Name": "init_duration", "Unit": "Milliseconds" }
                ]
            }
        ],
        "LambdaInsights": {
            "ShareTelemetry": true
        }
    },
    "event_type": "performance",
    "function_name": "cpu-intensive",
    "version": "Blue",
    "request_id": "12345678-8bcc-42f7-b1de-123456789012",
    "trace_id": "1-5faae118-12345678901234567890",
    "duration": 45191,
    "billed_duration": 45200,
    "billed_mb_ms": 11571200,
    "cold_start": true,
    "init_duration": 130,
    "tmp_free": 538329088,
    "tmp_max": 551346176,
    "threads_max": 11,
    "used_memory_max": 63,
    "total_memory": 256,
    "memory_utilization": 24,
    "cpu_user_time": 6640,
    "cpu_system_time": 50,
    "cpu_total_time": 6690,
    "fd_use": 416,
    "fd_max": 32642,
    "tx_bytes": 4434,
    "rx_bytes": 6911,
    "timeout": true,
```

```
    "shutdown_reason": "Timeout",
    "total_network": 11345,
    "agent_version": "1.0.72.0",
    "agent_memory_avg": 10,
    "agent_memory_max": 10
}
```


Contributor Insights를 사용하여 카디널리티가 높은 데이터 분석하기

Contributor Insights를 사용하면 로그 데이터를 분석하고 기고자 데이터를 표시하는 시계열을 생성할 수 있습니다. 상위 N개의 기고자, 총 고유 기고자 수 및 사용량에 대한 지표를 볼 수 있습니다. 이를 통해 상위 대화자를 찾고 시스템 성능에 영향을 미치는 사람 또는 대상을 파악할 수 있습니다. 예를 들어 잘못된 호스트를 찾거나 사용량이 가장 많은 네트워크 사용자를 식별하거나 가장 많은 오류를 생성하는 URL을 찾을 수 있습니다.

Scratch에서 규칙을 작성할 수 있으며, AWS Management Console를 사용하는 경우 AWS에서 만든 샘플 규칙을 사용할 수도 있습니다. 규칙은 `IpAddress`와 같이 기고자 정의에 사용할 로그 필드를 정의합니다. 또한 로그 데이터를 필터링하여 개별 기고자의 동작을 찾고 분석할 수 있습니다.

또한 CloudWatch는 다른 AWS 서비스의 지표를 분석하는 데 사용할 수 있는 기본 제공 규칙도 제공합니다.

모든 규칙은 수신 데이터를 실시간으로 분석합니다.

CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정에 로그인한 경우 해당 모니터링 계정에서 소스 계정과 모니터링 계정의 로그 그룹을 분석하는 Contributor Insights 규칙을 생성할 수 있습니다. 여러 계정의 로그 그룹을 분석하는 단일 규칙을 생성할 수도 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

> **Note**
>
> Contributor Insights를 사용하는 경우 규칙과 일치하는 각 로그 이벤트에 대해 요금이 발생합니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

주제

- Contributor Insights 규칙 생성

- Contributor Insights 규칙 구문
- Contributor Insights 규칙 예시
- Contributor Insights 보고서 보기
- 규칙에 따라 생성된 지표 그래프 작성
- Contributor Insights 기본 제공 규칙 사용하기

Contributor Insights 규칙 생성

규칙을 생성하여 로그 데이터를 분석할 수 있습니다. JSON 또는 CLF(일반적 로그 형식)의 모든 로그를 평가할 수 있습니다. 여기에는 이러한 형식 중 하나를 따르는 사용자 지정 로그와 Amazon VPC 흐름 로그, Amazon Route 53 DNS 쿼리 로그, Amazon ECS 컨테이너 로그 그리고 AWS CloudTrail, Amazon SageMaker, Amazon RDS, AWS AppSync 및 API Gateway의 로그와 같은 AWS 서비스의 로그가 포함됩니다.

규칙에서 필드 이름이나 값을 지정할 때 일치하는 모든 항목은 대/소문자를 구분합니다.

규칙을 생성할 때 기본 제공 샘플 규칙을 사용하거나 Scratch에서 고유한 규칙을 생성할 수 있습니다. Contributor Insights에는 다음 유형의 로그에 대한 샘플 규칙이 포함되어 있습니다.

- Amazon API Gateway 로그
- Amazon Route 53 퍼블릭 DNS 쿼리 로그
- Amazon Route 53 Resolver 쿼리 로그
- CloudWatch Container Insights 로그
- VPC 흐름 로그

CloudWatch 크로스 계정 관측성에서 모니터링 계정으로 설정된 계정에 로그인한 경우 모니터링 계정에서 로그 그룹에 대한 규칙을 생성하는 것 외에도 이 모니터링 계정에 연결된 소스 계정의 로그 그룹에 대한 Contributor Insights 규칙을 생성할 수 있습니다. 여러 계정의 로그 그룹을 모니터링하는 단일 규칙을 설정할 수도 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

> **Important**
>
> 사용자에게 `cloudwatch:PutInsightRule` 권한을 부여하면 기본적으로 해당 사용자는 CloudWatch Logs의 로그 그룹을 평가하는 규칙을 생성할 수 있습니다. 이러한 권한을 제한하는 IAM 정책 조건을 추가하여 사용자가 특정 로그 그룹을 포함하고 제외하도록 할 수 있습니

다. 자세한 내용은 조건 키를 사용하여 Contributor Insights 사용자의 로그 그룹 액세스 제한 단원을 참조하십시오.

기본 제공 샘플 규칙을 사용하여 규칙을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 인사이트(Insights)를 선택한 다음, Contributor Insights를 선택합니다.
3. Create rule을 선택합니다.
4. Select log group(s)(로그 그룹 선택)에서 규칙을 모니터링할 로그 그룹을 선택합니다. 로그 그룹을 20개까지 선택할 수 있습니다. CloudWatch 크로스 계정 관측성을 위해 설정된 모니터링 계정에 로그인한 경우 소스 계정에서 로그 그룹을 선택할 수 있으며 여러 계정의 로그 그룹을 분석하는 단일 규칙을 생성할 수도 있습니다.
   - (선택 사항) 이름이 특정 문자열로 시작하는 모든 로그 그룹을 선택하려면 접두사 일치로 선택 드롭다운을 누른 다음 접두사를 입력합니다. 모니터링 계정인 경우 선택적으로 검색할 계정을 선택할 수 있습니다. 그렇지 않으면 모든 계정이 선택됩니다.

   > **Note**
   >
   > 규칙과 일치하는 각 로그 이벤트에 대해 요금이 발생합니다. 접두사 일치로 선택 드롭다운을 선택하는 경우 접두사가 일치할 수 있는 로그 그룹 수를 알고 있어야 합니다. 실수로 의도한 것보다 많은 로그 그룹을 검색하면 예기치 않은 요금이 발생할 수 있습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.
5. Rule type(규칙 유형)에서 Sample rule(샘플 규칙)을 선택합니다. 그런 다음 Select sample rule(샘플 규칙 선택)을 선택하고 규칙을 선택합니다.
6. 샘플 규칙에는 Log format(로그 형식), Contribution(기여), Filters(필터) 및 Aggregate on(집계) 필드가 작성되어 있습니다. 원하는 경우 이러한 값을 조정할 수 있습니다.
7. 다음을 선택합니다.
8. Rule name(규칙 이름)에 이름을 입력합니다. 유효한 문자는 A~Z, a~z, 0~9, -(하이픈), _(밑줄) 및 .(마침표)입니다.
9. 규칙을 활성화된 또는 비활성화된 상태로 생성할지 여부를 선택합니다. 규칙을 활성화하도록 선택하면 즉시 규칙을 사용하여 데이터 분석이 시작됩니다. 활성화된 규칙을 실행하면 비용이 발생합니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

Contributor Insights는 규칙이 생성된 후 새 로그 이벤트만 분석합니다. 규칙은 이전에 CloudWatch Logs에서 처리한 로그 이벤트를 처리할 수 없습니다.

10. (선택 사항) Tags(태그)에서 이 규칙에 대한 태그로 하나 이상의 키-값 페어를 추가합니다. 태그를 사용하면 AWS 리소스를 식별 및 구성하고 AWS 비용을 추적할 수 있습니다. 자세한 내용은 Amazon CloudWatch 리소스 태그 지정 단원을 참조하십시오.
11. 생성(Create)을 선택합니다.

Scratch에서 규칙을 새로 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Contributor Insights를 선택합니다.
3. Create rule을 선택합니다.
4. Select log group(s)(로그 그룹 선택)에서 규칙을 모니터링할 로그 그룹을 선택합니다. 로그 그룹을 20개까지 선택할 수 있습니다. CloudWatch 크로스 계정 관측성을 위해 설정된 모니터링 계정에 로그인한 경우 소스 계정에서 로그 그룹을 선택할 수 있으며 여러 계정의 로그 그룹을 분석하는 단일 규칙을 생성할 수도 있습니다.

    - (선택 사항) 이름이 특정 문자열로 시작하는 모든 로그 그룹을 선택하려면 접두사 일치로 선택 드롭다운을 누른 다음 접두사를 입력합니다.

    > **Note**
    >
    > 규칙과 일치하는 각 로그 이벤트에 대해 요금이 발생합니다. 접두사 일치로 선택 드롭다운을 선택하는 경우 접두사가 일치할 수 있는 로그 그룹 수를 알고 있어야 합니다. 실수로 의도한 것보다 많은 로그 그룹을 검색하면 예기치 않은 요금이 발생할 수 있습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

5. Rule type(규칙 유형)에서 Custom rule(사용자 지정 규칙)을 선택합니다.
6. 로그 형식에서 JSON 또는 CLF를 선택합니다.
7. 마법사를 사용하거나 Syntax(구문) 탭을 선택하고 규칙 구문을 수동으로 지정하여 규칙 생성을 완료할 수 있습니다.

    마법사를 계속 사용하려면 다음을 수행합니다.

a. Contribution(기고), Key(키)에 보고할 기고자 유형을 입력합니다. 보고서에는 이 기고자 유형에 대한 상위 N개의 값이 표시됩니다.

   유효한 항목은 값이 있는 모든 로그 필드입니다. 예를 들면 **`requestId`**, **`sourceIPaddress`** 및 **`containerID`**입니다.

   특정 로그 그룹에 있는 로그의 로그 필드 이름을 찾는 방법에 대한 자세한 내용은 로그 필드 찾기를 참조하세요.

   1KB보다 큰 키는 1KB 단위로 잘립니다.

b. (선택 사항) Add new key(새 키 추가)를 선택하여 키를 더 추가합니다. 규칙에 최대 4개의 키를 포함할 수 있습니다. 두 개 이상의 키를 입력하면 보고서의 기고자는 키의 고유한 값 조합으로 정의됩니다. 예를 들어 세 개의 키를 지정하면 세 개의 키에 대한 각각의 고유한 값 조합이 고유한 기고자로 계산됩니다.

c. (선택 사항) 결과 범위를 좁히는 필터를 추가하려는 경우 필터 추가(Add filter)를 선택합니다. 일치(Match)에 필터링하려는 로그 필드 이름을 입력합니다. 그런 다음 조건(Condition)에서 비교 연산자를 선택하고 필터링하려는 값을 입력합니다.

   규칙에 필터를 4개까지 추가할 수 있습니다. 여러 필터는 AND 논리로 결합되므로 모든 필터와 일치하는 로그 이벤트만 평가됩니다.

   > **Note**
   >
   > 비교 연산자 다음에 나오는 배열(예:`In`, `NotIn` 또는 `StartsWith`)은 최대 10개의 문자열 값을 포함할 수 있습니다. Contributor Insights 규칙 구문에 대한 자세한 내용은 Contributor Insights 규칙 구문 섹션을 참조하세요.

d. [집계(Aggregate on)]에서 [수(Count)] 또는 [합계(Sum)]를 선택합니다. [수(Count)]를 선택하면 기여 요소 순위가 발생 횟수를 기반으로 결정됩니다. [합계(Sum)]를 선택하면 [기여(Contribution)], [값(Value)]에서 지정한 필드 값의 집계된 합계를 기반으로 순위가 결정됩니다.

8. 마법사를 사용하는 대신 규칙을 JSON 객체로 입력하려면 다음을 수행합니다.

   a. Syntax(구문) 탭을 선택합니다.

   b. Rule body(규칙 본문)에 규칙의 JSON 객체를 입력합니다. 규칙 구문에 대한 자세한 내용은 Contributor Insights 규칙 구문 단원을 참조하십시오.

9. 다음을 선택합니다.

10. Rule name(규칙 이름)에 이름을 입력합니다. 유효한 문자는 A-Z, a-z, 0-9, "-", "_" 및 "."입니다.
11. 규칙을 활성화된 또는 비활성화된 상태로 생성할지 여부를 선택합니다. 규칙을 활성화하도록 선택하면 즉시 규칙을 사용하여 데이터 분석이 시작됩니다. 활성화된 규칙을 실행하면 비용이 발생합니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

    Contributor Insights는 규칙이 생성된 후 새 로그 이벤트만 분석합니다. 규칙은 이전에 CloudWatch Logs에서 처리한 로그 이벤트를 처리할 수 없습니다.
12. (선택 사항) Tags(태그)에서 이 규칙에 대한 태그로 하나 이상의 키-값 페어를 추가합니다. 태그를 사용하면 AWS 리소스를 식별 및 구성하고 AWS 비용을 추적할 수 있습니다. 자세한 내용은 Amazon CloudWatch 리소스 태그 지정 단원을 참조하십시오.
13. Next(다음)를 선택합니다.
14. 입력한 설정을 확인하고 Create rule(규칙 생성)을 선택합니다.

생성한 규칙을 비활성화, 활성화 또는 삭제할 수 있습니다.

Contributor Insights에서 규칙을 활성화, 비활성화 또는 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Contributor Insights를 선택합니다.
3. 규칙 목록에서 단일 규칙 옆에 있는 확인란을 선택합니다.

    기본 제공 규칙은 AWS 서비스에서 생성되며 편집, 비활성화 또는 삭제할 수 없습니다.
4. 작업을 선택한 다음 원하는 옵션을 선택합니다.

로그 필드 찾기

규칙을 생성할 때 로그 그룹의 로그 항목에 있는 필드의 이름을 알아야 합니다.

로그 그룹에서 로그 필드를 찾으려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창의 로그에서 Insights를 선택합니다.
3. 쿼리 편집기 위에서 쿼리할 로그 그룹을 하나 이상 선택합니다.

    로그 그룹을 선택하면 CloudWatch Logs Insights가 로그 그룹의 데이터에서 필드를 자동으로 감지하고 오른쪽 창의 [검색된 필드(Discovered fields)]에 해당 필드를 표시합니다.

Contributor Insights 규칙 구문

이 단원에서는 Contributor Insights 규칙의 구문에 대해 설명합니다. JSON 블록을 입력하여 규칙을 생성하는 경우에만 이 구문을 사용합니다. 마법사를 사용하여 규칙을 생성하는 경우 구문을 알 필요가 없습니다. 마법사를 사용하여 규칙을 생성하는 방법에 대한 자세한 내용은 Contributor Insights 규칙 생성 단원을 참조하십시오.

이벤트 필드 이름 및 값을 기록하기 위한 규칙과 일치하는 모든 항목은 대/소문자를 구분합니다.

다음 예제에서는 JSON 로그의 구문을 보여줍니다.

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "API-Gateway-Access-Logs*",
        "Log-group-name2"
    ],
    "LogFormat": "JSON",
    "Contribution": {
        "Keys": [
            "$.ip"
        ],
        "ValueOf": "$.requestBytes",
        "Filters": [
            {
                "Match": "$.httpMethod",
                "In": [
                    "PUT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

Contributor Insights 규칙의 필드

스키마

CloudWatch Logs 데이터를 분석하는 규칙의 `Schema` 값은 항상 `{"Name": "CloudWatchLogRule", "Version": 1}`이어야 합니다.

LogGroupNames

문자열 배열입니다. 배열의 각 요소에 대해 선택적으로 문자열 끝에 `*`를 사용하여 해당 접두사로 시작하는 이름을 가진 모든 로그 그룹을 포함할 수 있습니다.

로그 그룹 이름과 함께 와일드카드를 사용할 때는 주의해야 합니다. 규칙과 일치하는 각 로그 이벤트에 대해 요금이 발생합니다. 의도한 것보다 많은 로그 그룹을 실수로 검색하면 예기치 않은 요금이 발생할 수 있습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

LogGroupARNs

CloudWatch 크로스 계정 관측성 모니터링 계정에서 이 규칙을 생성하는 경우 `LogGroupARNs`을 사용하여 모니터링 계정에 연결된 소스 계정에 로그 그룹을 지정하고 모니터링 계정 자체에 로그 그룹을 지정할 수 있습니다. 규칙에 `LogGroupNames` 또는 `LogGroupARNs`를 지정해야 하지만 둘 다 지정할 수는 없습니다.

`LogGroupARNs`는 문자열 배열입니다. 배열의 각 요소에 대해 특정 상황에서 선택적으로 `*`를 와일드카드로 사용할 수 있습니다. 예를 들어 `arn:aws:logs:us-west-1:*:log-group/MyLogGroupName2`를 지정하여 미국 서부(캘리포니아 북부) 리전의 모든 소스 계정과 모니터링 계정에 `MyLogGroupName2`라는 로그 그룹을 지정할 수 있습니다. `arn:aws:logs:us-west-1:111122223333:log-group/GroupNamePrefix*`를 지정하여 111122223333에 이름이 `GroupNamePrefix`로 시작하는 미국 서부(캘리포니아 북부)의 모든 로그 그룹을 지정할 수도 있습니다.

부분 AWS 계정 ID를 와일드 카드가 있는 접두사로 지정할 수 없습니다.

로그 그룹 ARN과 함께 와일드카드를 사용할 때는 주의해야 합니다. 규칙과 일치하는 각 로그 이벤트에 대해 요금이 발생합니다. 의도한 것보다 많은 로그 그룹을 실수로 검색하면 예기치 않은 요금이 발생할 수 있습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

LogFormat

유효 값은 `JSON`및 `CLF`입니다.

Contribution

이 객체에는 최대 4개의 멤버가 있는 Keys 배열, 선택적으로 단일 ValueOf 및 선택적으로 최대 4개의 Filters가 있는 배열이 포함됩니다.

키

기고자를 분류하는 측정기준으로 사용되는 최대 4개의 로그 필드의 배열입니다. 두 개 이상의 키를 입력하면 키에 대한 각각의 고유한 값 조합이 고유한 기고자로 계산됩니다. 필드는 JSON 속성 형식 표기법을 사용하여 지정해야 합니다.

ValueOf

(선택 사항) Sum을 AggregateOn의 값으로 지정하는 경우에만 이 옵션을 지정합니다. ValueOf는 숫자 값을 갖는 로그 필드를 지정합니다. 이 유형의 규칙에서 기고자 순위는 로그 항목에서의 발생 횟수 대신 이 필드 값의 합계로 결정됩니다. 예를 들어 기고자를 일정 기간 동안 총 BytesSent로 정렬하려는 경우 ValueOf를 BytesSent로 설정하고 AggregateOn에 Sum을 지정합니다.

필터

(선택 사항) 보고서에 포함된 로그 이벤트의 범위를 좁히기 위해 최대 4개의 필터가 있는 배열을 지정합니다. 여러 필터를 지정하면 Contributor Insights는 논리 AND 연산자를 사용하여 필터를 평가합니다. 이 옵션을 사용하여 검색에서 관련 없는 로그 이벤트를 필터링하거나 단일 기고자를 선택하여 해당 동작을 분석할 수 있습니다.

배열의 각 멤버는 Match 필드와 사용할 일치하는 연산자 유형을 나타내는 필드를 포함해야 합니다.

Match 필드는 필터에서 평가할 로그 필드를 지정합니다. 로그 필드는 JSON 속성 형식 표기법을 사용하여 지정됩니다.

일치하는 연산자 필드는 In, NotIn, StartsWith, GreaterThan, LessThan, EqualTo, NotEqualTo 또는 IsPresent 중 하나여야 합니다. 연산자 필드가 In, NotIn 또는 StartsWith인 경우 확인할 문자열 값의 배열이 뒤에 옵니다. Contributor Insights는 OR 연산자를 사용하여 문자열 값의 배열을 평가합니다. 배열은 최대 10개의 문자열 값을 포함할 수 있습니다.

연산자 필드가 GreaterThan, LessThan, EqualTo 또는 NotEqualTo인 경우 비교할 단일 숫자 값이 뒤에 옵니다.

연산자 필드가 IsPresent인 경우 뒤에 true 또는 false가 옵니다. 이 연산자는 로그 이벤트에 지정된 로그 필드가 있는지 여부에 따라 로그 이벤트를 일치시킵니다. isPresent는 JSON 속성의 리프 노드에 있는 값에서만 작동합니다. 예를 들어 c-count와 일치하는 항목을 찾는 필터는 details.c-count.c1 값이 있는 로그 이벤트를 평가하지 않습니다.

필터의 예는 다음을 참조하세요.

```
{"Match": "$.httpMethod", "In": [ "PUT", ] }
{"Match": "$.StatusCode", "EqualTo": 200 }
{"Match": "$.BytesReceived", "GreaterThan": 10000}
{"Match": "$.eventSource", "StartsWith": [ "ec2", "ecs" ] }
```

AggregateOn

유효 값은 `Count`및 `Sum`입니다. 발생 횟수 또는 `ValueOf` 필드에 지정된 필드 값의 합계를 기준으로 보고서를 집계할지 여부를 지정합니다.

JSON 속성 형식 표기법

`Keys`, `ValueOf` 및 `Match` 필드는 점 표기법이 있는 JSON 속성 형식을 따르며 여기서 $는 JSON 객체의 루트를 나타냅니다. 그 뒤에는 마침표와 하위 속성의 이름을 가진 영숫자 문자열이 옵니다. 여러 속성 레벨이 지원됩니다.

문자열의 첫 번째 문자는 A~Z 또는 a~z만 가능합니다. 문자열의 다음 문자는 A~Z, a~z 또는 0~9 중 하나일 수 있습니다.

다음 목록에서는 JSON 속성 형식의 유효한 예를 보여 줍니다.

```
$.userAgent
$.endpoints[0]
$.users[1].name
$.requestParameters.instanceId
```

CLF 로그에 대한 규칙의 추가 필드

CLF(일반적 로그 형식) 로그 이벤트에는 JSON과 같은 필드에 대한 이름이 없습니다. Contributor Insights 규칙에 사용할 필드를 제공하기 위해 CLF 로그 이벤트를 인덱스가 1부터 시작하는 배열로 취급할 수 있습니다. 예를 들어 첫 번째 필드를 **"1"**로, 두 번째 필드를 **"2"**로 지정할 수 있습니다.

CLF 로그에 대한 규칙을 읽기 쉽게 만들려면 `Fields`를 사용합니다. 이렇게 하면 CLF 필드 위치에 대한 이름 지정 별칭을 제공할 수 있습니다. 예를 들어 위치 “4"가 IP 주소가 되도록 지정할 수 있습니다. 지정한 후에는 `IpAddress`를 규칙에서 `Keys`, `ValueOf` 및 `Filters`의 속성으로 사용할 수 있습니다.

다음은 `Fields` 필드를 사용하는 CLF 로그에 대한 규칙의 예입니다.

```
{
```

```
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "API-Gateway-Access-Logs*"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "4": "IpAddress",
        "7": "StatusCode"
    },
    "Contribution": {
        "Keys": [
            "IpAddress"
        ],
        "Filters": [
            {
                "Match": "StatusCode",
                "EqualTo": 200
            }
        ]
    },
    "AggregateOn": "Count"
}
```

Contributor Insights 규칙 예시

이 단원에는 Contributor Insights 규칙의 사용 사례를 보여주는 예제가 포함되어 있습니다.

VPC 흐름 로그: 소스 및 대상 IP 주소별 바이트 전송

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "4": "srcaddr",
```

```
        "5": "dstaddr",
        "10": "bytes"
    },
    "Contribution": {
        "Keys": [
            "srcaddr",
            "dstaddr"
        ],
        "ValueOf": "bytes",
        "Filters": []
    },
    "AggregateOn": "Sum"
}
```

VPC 흐름 로그: HTTPS 요청 수가 가장 많음

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "5": "destination address",
        "7": "destination port",
        "9": "packet count"
    },
    "Contribution": {
        "Keys": [
            "destination address"
        ],
        "ValueOf": "packet count",
        "Filters": [
            {
                "Match": "destination port",
                "EqualTo": 443
            }
        ]
    },
    "AggregateOn": "Sum"
```

```
}
```

VPC 흐름 로그: 거부된 TCP 연결

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
            "sourceAddress"
        ],
        "Filters": [
            {
                "Match": "protocol",
                "EqualTo": 6
            },
            {
                "Match": "action",
                "In": [
                    "REJECT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

소스 주소별 Route 53 NXDomain 응답

```
{
```

```
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.rcode",
                "StartsWith": [
                    "NXDOMAIN"
                ]
            }
        ],
        "Keys": [
            "$.srcaddr"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
}
```

도메인 이름별 Route 53 Resolver 쿼리

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [],
        "Keys": [
            "$.query_name"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
}
```

쿼리 유형 및 소스 주소별 Route 53 Resolver 쿼리

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [],
        "Keys": [
            "$.query_type",
            "$.srcaddr"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
}
```

Contributor Insights 보고서 보기

보고서 데이터의 그래프와 규칙에 의해 발견된 기고자의 순위가 매겨진 목록을 보려면 다음 단계를 따르십시오.

규칙 보고서를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 Contributor Insights를 선택합니다.
3. 규칙 목록에서 규칙 이름을 선택합니다.

   그래프는 지난 3시간 동안의 규칙 결과를 표시합니다. 그래프 아래의 표에는 상위 10개의 기고자가 나와 있습니다.
4. 표에 표시된 기고자 수를 변경하려면 그래프 상단에 있는 Top 10 contributors(상위 10개의 기고자)를 선택합니다.
5. 단일 기고자의 결과만 표시하도록 그래프를 필터링하려면 표 범례에서 해당 기고자를 선택합니다. 모든 기고자를 다시 표시하려면 범례에서 동일한 기고자를 다시 선택합니다.

6. 보고서에 표시된 시간 범위를 변경하려면 그래프 맨 위에서 15m, 30m, 1h, 2h, 3h 또는 사용자 정의(Custom)를 선택합니다.

   보고서의 최대 시간 범위는 24시간이지만 최대 15일 전에 발생한 24시간 기간을 선택할 수 있습니다. 과거의 기간을 선택하려면 custom(사용자 지정), absolute(절대값)를 선택한 다음 기간을 지정합니다.
7. 기고자의 집계 및 순위에 사용되는 기간의 길이를 변경하려면 그래프 상단에서 period(기간)를 선택합니다. 더 긴 기간을 보면 일반적으로 스파이크가 거의 없는 더 부드러운 보고서가 표시됩니다. 더 짧은 기간을 선택하면 스파이크가 나타날 가능성이 더 큽니다.
8. 이 그래프를 CloudWatch 대시보드에 추가하려면 [대시보드에 추가(Add to dashboard)]를 선택합니다.
9. 이 보고서의 로그 그룹이 쿼리 상자에 이미 로드된 상태로 CloudWatch Logs Insights 쿼리 창을 열려면 [로그 보기(View logs)]를 선택합니다.
10. 보고서 데이터를 클립보드나 CSV 파일로 내보내려면 내보내기를 선택합니다.

규칙에 따라 생성된 지표 그래프 작성

Contributor Insights는 지표 수학 함수 `INSIGHT_RULE_METRIC`을 제공합니다. 이 함수를 사용하여 Contributor Insights 보고서의 데이터를 CloudWatch 콘솔 [지표(Metrics)] 탭의 그래프에 추가할 수 있습니다. 이 수학 함수를 기반으로 경보를 설정할 수도 있습니다. 지표 수학 함수에 대한 자세한 내용은 지표 수학 사용 단원을 참조하세요.

이 지표 수학 함수를 사용하려면 `cloudwatch:GetMetricData` 및 `cloudwatch:GetInsightRuleReport` 권한이 모두 있는 계정에 로그인해야 합니다.

구문은 `INSIGHT_RULE_METRIC(`*`ruleName`*`, `*`metricName`*`)`입니다. *`ruleName`*은 Contributor Insights 규칙의 이름이고 *`metricName`*은 다음 목록의 값 중 하나입니다. *`metricName`*의 값은 수학 함수가 반환하는 데이터의 유형을 결정합니다.

- `UniqueContributors` - 각 데이터 요소에 대한 고유한 기여 요소 수입니다.
- `MaxContributorValue` - 각 데이터 요소에 대한 최상위 기여 요소의 값입니다. 그래프의 각 데이터 포인트에 대해 기고자의 ID가 변경될 수 있습니다.

  이 규칙이 `Count`를 기준으로 집계되는 경우 각 데이터 요소의 최상위 기여 요소는 해당 기간에 가장 많이 발생한 기여 요소입니다. 규칙이 `Sum`을 기준으로 집계되는 경우 최상위 기여 요소는 해당 기간 동안 규칙의 `Value`로 지정된 로그 필드에서 합계가 가장 큰 기여 요소입니다.

- SampleCount - 규칙과 일치하는 데이터 요소의 수입니다.
- Sum - 해당 데이터 요소가 나타내는 기간 동안 모든 기여 요소의 값 합계입니다.
- Minimum - 해당 데이터 요소가 나타내는 기간 동안 단일 관측치의 최솟값입니다.
- Maximum - 해당 데이터 요소가 나타내는 기간 동안 단일 관측치의 최댓값입니다.
- Average - 해당 데이터 요소가 나타내는 기간 동안 모든 기여 요소의 평균 값입니다.

Contributor Insights 지표 데이터에 대한 경보 설정하기

INSIGHT_RULE_METRIC 함수를 사용하여 Contributor Insights가 생성하는 지표에 대한 경보를 설정할 수 있습니다. 예를 들어, 거부된 전송 제어 프로토콜(TCP) 연결의 비율을 기준으로 경보를 생성할 수 있습니다. 이 유형의 경보를 시작하려면 다음 두 예제에 표시된 것과 같은 규칙을 생성할 수 있습니다.

예제 규칙: “RejectedConnectionsRule”

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
            "sourceAddress"
        ],
        "Filters": [
            {
                "Match": "protocol",
                "EqualTo": 6
            },
```

```
            {
                "Match": "action",
                "In": [
                    "REJECT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

예제 규칙: “TotalConnectionsRule”

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
            "sourceAddress"
        ],
        "Filters": [
            {
                "Match": "protocol",
                "EqualTo": 6
            }
    "AggregateOn": "Sum"
}
```

규칙을 생성하면 다음의 지표 수학 표현식 예를 사용하여 Contributor Insights가 보고하는 데이터를 그래프로 표시할 수 있는 CloudWatch 콘솔에서 지표(Metrics) 탭을 선택합니다.

예: 지표 수학 표현식

```
e1 INSIGHT_RULE_METRIC("RejectedConnectionsRule", "Sum")
e2 INSIGHT_RULE_METRIC("TotalConnectionsRule", "Sum")
e3 (e1/e2)*100
```

이 예에서 e3 지표 수학 표현식은 거부된 모든 TCP 연결을 반환합니다. TCP 연결의 20%가 거부되는 경우 알림을 받으려면, 임계값을 100에서 20으로 변경하여 표현식을 수정할 수 있습니다.

> **Note**
>
> 지표(Metrics) 섹션에서 모니터링하는 지표에 대해 경보를 설정할 수 있습니다. 그래프로 표시된 지표(Graphed metrics) 탭에서 작업(Actions) 열 아래에 있는 경보 생성(Create alarm) 아이콘을 선택할 수 있습니다. 경보 생성(Create alarm) 아이콘은 종 모양으로 되어 있습니다.

지표 그래프 작성 및 지표 수학 함수 사용에 대한 자세한 내용은 CloudWatch 그래프에 수학 표현식 추가 섹션을 참조하세요.

Contributor Insights 기본 제공 규칙 사용하기

Contributor Insights 기본 제공 규칙을 사용하여 다른 AWS 서비스의 지표를 분석할 수 있습니다. 기본 제공 규칙을 지원하는 서비스는 다음과 같습니다.

- Amazon DynamoDB 개발자 가이드의 Amazon DynamoDB용 Contributor Insights
- AWS PrivateLink 가이드의 기본 제공되는 Contributor Insights 규칙 사용

Amazon CloudWatch Application Insights

Amazon CloudWatch Application Insights에서 애플리케이션 및 기본 AWS 리소스의 상태를 확인할 수 있습니다. 애플리케이션 리소스에 대한 최상의 모니터를 설정하고 데이터를 분석하여 애플리케이션 문제의 징후가 있는지 지속적으로 확인할 수 있습니다. SageMaker 및 기타 AWS 기술을 기반으로 하는 Application Insights는 모니터링되는 애플리케이션의 잠재적인 문제를 보여 주는 자동화된 대시보드를 제공하므로 애플리케이션 및 인프라와 관련된 지속적인 문제를 신속하게 격리할 수 있습니다. Application Insights가 제공하는 애플리케이션 상태에 대한 향상된 가시성을 통해 MTTR(Mean Time to Repair)을 줄일 수 있으므로 애플리케이션 문제를 해결할 수 있습니다.

Amazon CloudWatch Application Insights에 애플리케이션을 추가하면 Application Insights가 애플리케이션의 리소스를 검색하고 CloudWatch에서 애플리케이션 구성 요소에 대한 지표 및 로그를 권장하고 구성합니다. 애플리케이션 구성 요소의 예로는 SQL Server 백엔드 데이터베이스 및 Microsoft IIS/웹 티어가 있습니다. Application Insights는 기록 데이터를 사용한 지표 패턴 분석을 통해 이상을 감지하고 애플리케이션, 운영 체제 및 인프라 로그에서 오류 및 예외를 지속적으로 감지합니다. 이 알고리즘은 분류 알고리즘과 기본 규칙을 조합하여 이러한 관찰 결과를 연결합니다. 그런 다음, 작업 우선 순위를 지정하는 데 도움이 되는 관련 관찰 및 문제 심각도 정보를 보여 주는 대시보드를 자동으로 생성합니다. 애플리케이션 대기 시간, SQL Server 백업 실패, 메모리 누수, 대용량 HTTP 요청, 취소된 I/O 작업과 같은 일반적인 .NET 및 SQL 애플리케이션 스택 문제에 대해 가능한 근본 원인 및 해결 단계를 나타내는 추가 인사이트를 제공합니다. AWS SSM OpsCenter와의 기본 통합 기능을 통해 관련 Systems Manager Automation 문서를 실행하여 문제를 해결할 수 있습니다.

섹션

Amazon CloudWatch Application Insights란?

CloudWatch Application Insights를 사용하면 다른 애플리케이션 리소스와 함께 Amazon EC2 인스턴스를 사용하는 애플리케이션을 모니터링할 수 있습니다. 이 기능은 애플리케이션 리소스 및 기술 스택(예: Microsoft SQL Server 데이터베이스, 웹(IIS) 및 애플리케이션 서버, OS, 로드 밸런서, 대기열 등) 전반에서 주요 지표 로그 및 경보를 파악하고 설정합니다. 지표 및 로그를 지속적으로 모니터링하여 이상 및 오류를 감지하고 연결합니다. 오류 및 이상이 감지되면 Application Insights에서 알림을 설정하고 작업을 수행할 수 있는 CloudWatch Events를 생성합니다. Application Insights는 문제 해결을 지원하기 위해 잠재적 근본 원인을 알려 주는 추가 인사이트와 함께 상관관계가 있는 지표 이상 및 로그 오

류를 비롯하여 감지한 문제에 대한 자동화된 대시보드를 생성합니다. 자동화된 대시보드를 사용하면 애플리케이션의 상태를 정상으로 유지하고 애플리케이션의 최종 사용자에게 미치는 영향을 방지하기 위한 수정 조치를 취할 수 있습니다. 또한 AWS SSM OpsCenter를 사용하여 문제를 해결할 수 있도록 OpsItem을 생성합니다.

CloudWatch에서 Windows 이벤트 로그뿐만 아니라 미러링된 쓰기 트랜잭션/초, 복구 대기열 길이, 트랜잭션 지연과 같은 중요한 카운터를 구성할 수 있습니다. 대상 데이터베이스 쿼리에 대한 제한된 액세스와 같이 SQL HA 워크로드에 장애 조치 이벤트 또는 문제가 발생하면 CloudWatch Application Insights는 자동화된 인사이트를 제공합니다.

CloudWatch Application Insights는 AWS Launch Wizard와 통합되어 AWS에 SQL Server HA 워크로드를 배포하기 위한 원클릭 모니터링 설정 환경을 제공합니다. Launch Wizard 콘솔에서 Application Insights를 사용하여 모니터링 및 인사이트를 설정하는 옵션을 선택하면 CloudWatch Application Insights는 CloudWatch에서 관련 지표, 로그, 경보를 자동으로 설정하고 새로 배포된 워크로드의 모니터링을 시작합니다. CloudWatch 콘솔에서 SQL Server HA 워크로드의 상태와 함께 자동화된 인사이트 및 감지된 문제를 확인할 수 있습니다.

내용

특성

Application Insights는 다음 기능을 제공합니다.

애플리케이션 리소스 모니터 자동 설정

CloudWatch Application Insights는 애플리케이션에 대한 모니터링을 설정하는 데 걸리는 시간을 줄여 줍니다. 그 방법은 애플리케이션 리소스를 검색하고 권장 지표 및 로그의 사용자 지정 목록을 제공하여 이를 CloudWatch에서 설정함으로써 Amazon EC2 및 Elastic Load Balancer(ELB)와 같은 애플리케이션 리소스에 필요한 가시성을 제공하는 것입니다. 또한 모니터링된 지표에 대해 동적 경보를 설정합니다. 경보는 지난 2주 동안 감지된 이상을 기반으로 자동 업데이트됩니다.

문제 감지 및 알림

CloudWatch Application Insights는 지표 이상 및 로그 오류와 같은 애플리케이션 관련 잠재적 문제의 징후를 감지합니다. 이 관찰 결과를 통해 애플리케이션의 잠재적 문제를 표면화할 수 있습니다. 그런 다음 알림을 받거나 작업을 수행하도록 구성할 수 있는 CloudWatch Events를 생성합니다. 따라서 지표나 로그 오류에 대한 개별 경보를 생성할 필요가 없습니다.

문제 해결

CloudWatch Application Insights는 감지된 문제에 대한 CloudWatch 자동 대시보드를 생성합니다. 대시보드에는 문제 해결에 도움이 되는 관련 지표 이상 및 로그 오류를 포함하여 문제에 대한 세부 정보가 표시됩니다. 또한 이상 및 오류의 잠재적 근본 원인을 파악하는 추가 통찰력을 제공합니다.

개념

다음은 Application Insights가 애플리케이션을 모니터링하는 방법을 이해하는 데 있어 중요한 개념입니다.

구성 요소

애플리케이션을 구성하는 유사한 리소스의 자동 그룹형, 독립 실행형 또는 사용자 지정 그룹. 모니터링 기능을 향상시키기 위해서는 유사한 리소스를 사용자 지정 구성 요소로 그룹화하는 것이 좋습니다.

관측치

애플리케이션 또는 애플리케이션 리소스에서 감지된 개별 이벤트(지표 이상, 로그 오류 또는 예외)

문제

문제는 관련 관찰을 연결, 분류 및 그룹화하여 감지됩니다.

CloudWatch Application Insights의 다른 주요 개념에 대한 정의는 Amazon CloudWatch 개념 단원을 참조하세요.

요금

CloudWatch Application Insights는 탐지된 문제에 대한 알림을 위해 CloudWatch 지표, 로그 및 이벤트를 사용하여 선택한 애플리케이션 리소스에 대해 권장되는 지표 및 로그를 설정합니다. 이러한 기능은 CloudWatch 요금에 따라 AWS 계정으로 청구됩니다. 탐지된 문제의 경우 Application Insights에서 SSM OpsItems가 생성되어 문제에 대해 알려줍니다. 또한 Application Insights는 SSM Parameter Store 파라미터를 생성하여 인스턴스에서 CloudWatch 에이전트를 구성합니다. Amazon

EC2 Systems Manager 기능은 SSM 요금제에 따라 요금이 부과됩니다. 설정 지원, 데이터 분석 모니터링 또는 문제 탐지에 대해서는 비용이 청구되지 않습니다.

CloudWatch Application Insights 비용

Amazon EC2 비용에는 다음 기능 사용이 포함됩니다.

- CloudWatch 에이전트
  - CloudWatch 에이전트 로그 그룹
  - CloudWatch 에이전트 지표
  - Prometheus 로그 그룹(JMX 워크로드용)

모든 리소스 비용에는 다음 기능 사용량이 포함됩니다.

- CloudWatch 경보(비용의 대부분)
- SSM OpsItems(최소 비용)

비용 계산 예

이 예제의 비용은 다음 시나리오에 따라 고려됩니다.

다음을 포함하는 리소스 그룹을 생성했습니다.

- SQL Server가 설치된 Amazon EC2 인스턴스.
- 연결된 Amazon EBS 볼륨.

CloudWatch Application Insights로 이 리소스 그룹을 온보딩하면 Amazon EC2 인스턴스에 설치된 SQL Server 워크로드가 탐지됩니다. CloudWatch Application Insights는 다음 지표를 모니터링하기 시작합니다.

SQL Server 인스턴스에 대해 다음 지표가 모니터링됩니다.

- CPUUtilization
- StatusCheckFailed
- Memory % Committed Bytes in Use
- 사용 가능한 메모리(MB)
- 네트워크 인터페이스 바이트 합계/초

- 페이징된 파일 % 사용량
- Physical Disk % Disk Time
- 프로세서 % 프로세서 시간
- SQLServer:Buffer Manager cache hit ratio
- SQLServer:Buffer Manager life expectancy
- SQLServer:차단된 일반 통계 프로세스
- SQLServer:일반 통계 사용자 연결
- SQLServer:잠금 교착 상태 수/초
- SQLServer:SQL 통계 배치 요청/초
- 시스템 프로세서 대기열 길이

SQL Server 인스턴스에 연결된 볼륨에 대해 다음 지표가 모니터링됩니다.

- VolumeReadBytes
- VolumeWriteBytes
- VolumeReadOps
- VolumeWriteOps
- VolumeTotalReadTime
- VolumeTotalWriteTime
- VolumeIdleTime
- VolumeQueueLength
- VolumeThroughputPercentage
- VolumeConsumedReadWriteOps
- BurstBalance

이 시나리오의 경우 CloudWatch 요금 페이지와 SSM 요금 페이지에 따라 비용이 계산됩니다.

- 사용자 지정 지표

  이 시나리오의 경우 위 지표 중 13개가 CloudWatch 에이전트를 사용하여 CloudWatch로 내보내집니다. 이러한 지표는 사용자 지정 지표로 처리됩니다. 각 사용자 지정 지표의 비용은 월 0.3 USD입니다. 이러한 사용자 지정 지표의 총 비용은 13 * 0.3 USD = 3.90 USD/월입니다.
- 경보

이 시나리오의 경우 CloudWatch Application Insights는 총 26개의 지표를 모니터링하여 26개의 경보를 생성합니다. 각 경보의 비용은 0.1 USD/월입니다. 경보의 총 비용은 26 * $.1 = 2.60 USD/월입니다.

- 데이터 모으기 및 오류 로그

  데이터 모으기 비용은 0.05 USD/GB이고 SQL Server 오류 로그의 스토리지는 0.03 USD/GB입니다. 데이터 모으기 및 오류 로그에 드는 총 비용은 0.05 USD/GB + 0.03 USD/GB= 0.08 USD/GB입니다.

- Amazon EC2 Systems Manager OpsItems

  CloudWatch Application Insights에서 탐지된 각 문제에 대해 SSM OpsItem이 생성됩니다. 애플리케이션에 문제가 n개 있는 경우 총 비용은 월 0.00267 USD * n입니다.

관련 서비스

다음 서비스가 CloudWatch Application Insights와 함께 사용됩니다.

관련 AWS 서비스

- Amazon CloudWatch를 사용하여 시스템 전체의 리소스 사용률, 애플리케이션 성능 및 운영 상태를 파악할 수 있습니다. 지표를 수집 및 추적하고 경보 알림을 보내고 정의한 규칙에 따라 모니터링 중인 리소스를 자동으로 업데이트하고 사용자 지정 지표를 모니터링할 수 있습니다. CloudWatch Application Insights는 CloudWatch를 통해, 특히 CloudWatch 기본 운영 대시보드 내에서 시작됩니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.
- CloudWatch Container Insights는 컨테이너 애플리케이션 및 마이크로서비스의 지표 및 로그를 수집하고, 종합하며, 요약합니다. Container Insights를 사용하여 Amazon EC2에서 Amazon ECS, Amazon Elastic Kubernetes Service 및 Kubernetes 플랫폼을 모니터링할 수 있습니다. Container Insights 또는 Application Insights 콘솔에서 Application Insights를 활성화하면 Application Insights는 Container Insights 대시보드에 감지된 문제를 표시합니다. 자세한 정보는 Container Insights을 참조하세요.
- Amazon DynamoDB는 분산 데이터베이스를 운영하고 크기 조정하는 데 따른 관리 부담을 없애주는 완전관리형 NoSQL 데이터베이스 서비스로, 하드웨어 프로비저닝, 설정 및 구성, 복제, 소프트웨어 패치 적용 또는 클러스터 크기 조정에 대해 걱정할 필요가 없게 합니다. 또한 DynamoDB는 유휴 시 암호화를 제공하여 중요한 데이터 보호와 관련된 운영 부담 및 복잡성을 제거합니다.
- Amazon EC2는 AWS 클라우드에서 확장 가능한 컴퓨팅 용량을 제공합니다. Amazon EC2를 사용하면 필요한 수만큼 적절하게 가상 서버를 시작하고 보안 및 네트워킹을 구성하며 스토리지를 관리할 수 있습니다. 요건이나 갑작스러운 인기 증대 등 변동 사항에 따라 확장하거나 축소할 수 있어 트

래픽 예측 필요성이 줄어듭니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서 또는 Windows 인스턴스용 Amazon EC2 설명서를 참조하세요.

- Amazon Elastic Block Store(Amazon EBS)는 Amazon EC2 인스턴스에 사용할 수 있는 블록 수준 스토리지 볼륨을 제공합니다. Amazon EBS 볼륨은 형식이 지정되지 않은 원시 블록 디바이스처럼 동작합니다. 이러한 볼륨을 인스턴스에 디바이스로 마운트할 수 있습니다. 인스턴스에 연결된 Amazon EBS 볼륨은 스토리지 볼륨으로 표시되며, 인스턴스 수명과 관계없이 지속됩니다. 이러한 볼륨 위에 파일 시스템을 생성하거나 하드 드라이브와 같은 블록 디바이스를 사용하는 것처럼 볼륨을 사용할 수 있습니다. 인스턴스에 연결된 볼륨의 구성을 동적으로 변경할 수 있습니다. 자세한 내용은 Amazon EBS 사용 설명서를 참조하세요.
- Amazon EC2 Auto Scaling을 사용하면 애플리케이션의 로드를 처리할 수 있는 정확한 수의 EC2 인스턴스를 유지할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서를 참조하세요.
- Elastic Load Balancing은 여러 가용 영역에 있는 EC2 인스턴스, 컨테이너, IP 주소와 같은 여러 대상에 걸쳐 수신되는 애플리케이션 또는 네트워크 트래픽을 분산합니다. 자세한 내용은 Elastic Load Balancing 사용 설명서를 참조하세요.
- IAM은 사용자의 AWS 리소스에 대한 액세스를 안전하게 제어하도록 지원하는 웹 서비스입니다. IAM을 사용하여 AWS 리소스를 사용할 수 있는 사람을 제어(인증)하고 해당 사용자가 사용할 수 있는 리소스 및 그 사용 방법을 제어(권한 부여)할 수 있습니다. 자세한 내용은 Amazon CloudWatch에 대한 인증 및 액세스 제어를 참조하세요.
- AWS Lambda를 사용하면 이벤트에 의해 트리거되는 함수로 구성된 서버리스 애플리케이션을 구축하고 CodePipeline 및 AWS CodeBuild를 사용해 이를 자동으로 배포할 수 있습니다. 자세한 내용은 AWS Lambda 애플리케이션 단원을 참조하세요.
- AWS Launch Wizard for SQL Server는 SQL Server 고가용성 솔루션을 클라우드에 배포하는 데 걸리는 시간을 줄여줍니다. 서비스 콘솔에서 성능, 노드 수, 연결을 비롯한 애플리케이션 요구 사항을 입력하면 AWS Launch Wizard가 SQL Server Always On 애플리케이션을 배포하고 실행하는 데 적합한 AWS 리소스를 식별합니다.
- AWS Resource Groups를 사용하면 애플리케이션을 구성하는 리소스를 정리할 수 있습니다. Resource Groups를 사용하여 많은 리소스에 대한 작업을 한 번에 관리하고 자동화할 수 있습니다. 단일 애플리케이션에 대해 하나의 리소스 그룹만 등록할 수 있습니다. 자세한 내용은 AWS Resource Groups 사용 설명서를 참조하세요.
- Amazon SQS는 내구력 있고 가용성이 뛰어난 보안 호스팅 대기열을 제공하며 이를 통해 분산 소프트웨어 시스템과 구성 요소를 통합 및 분리할 수 있습니다. 자세한 내용은 Amazon SQS 사용 설명서를 참조하세요.
- AWS Step Functions는 AWS Lambda 함수를 비롯한 다양한 AWS 서비스 및 리소스를 구조화된 시각적 워크플로로 시퀀싱할 수 있게 하는 서버리스 함수 컴포저입니다. 자세한 내용은 AWS Step Functions 사용 설명서를 참조하십시오.

- AWS SSM OpsCenter는 서비스 전반에 걸쳐 OpsItem을 집계하고 표준화하는 동시에 각 OpsItem, 관련 OpsItem, 관련 리소스에 관한 상황별 조사 데이터를 제공합니다. OpsCenter는 문제를 신속하게 해결하는 데 사용할 수 있는 Systems Manager Automation 문서(실행서)도 제공합니다. 각 OpsItem에 대해 검색 가능한 사용자 지정 데이터를 지정할 수 있습니다. 상태 및 소스별로 OpsItem에 대한 자동 생성 요약 보고서를 볼 수도 있습니다. 자세한 내용은 AWS Systems Manager 사용 설명서를 참조하세요.
- Amazon API Gateway는 규모와 관계없이 REST, HTTP 및 WebSocket API를 생성하고 게시하며 유지 관리하고 모니터링하며 보호하기 위한 AWS 서비스입니다. API 개발자는 AWS 또는 다른 웹 서비스를 비롯해 AWS 클라우드에 저장된 데이터에 액세스하는 API를 생성할 수 있습니다. 자세한 내용은 Amazon API Gateway 사용 설명서를 참조하세요.

  > **Note**
  >
  > Application Insights는 REST API 프로토콜(API Gateway 서비스의 v1)만 지원합니다.

- Amazon Elastic Container Service(Amazon ECS)는 완전관리형 컨테이너 오케스트레이션 서비스입니다. Amazon ECS를 사용하여 가장 민감하고 미션 크리티컬한 애플리케이션을 실행할 수 있습니다. 자세한 내용은 Amazon Elastic Container Service 개발자 안내서를 참조하세요.
- Amazon Elastic Kubernetes Service(Amazon EKS)는 자체 Kubernetes 제어 영역 또는 노드를 설치, 운영, 유지 관리할 필요 없이 AWS에서 Kubernetes를 실행하는 데 사용할 수 있는 관리형 서비스입니다. Kubernetes는 컨테이너화된 애플리케이션의 배포, 조정 및 관리 자동화를 위한 오픈 소스 시스템입니다. 자세한 내용은 Amazon EKS 사용 설명서를 참조하세요.
- Amazon EC2의 Kubernetes. Kubernetes는 컨테이너화된 애플리케이션을 대규모로 배포하고 관리하도록 지원하는 오픈 소스 소프트웨어입니다. Kubernetes는 Amazon EC2 컴퓨팅 인스턴스 클러스터를 관리하며 배포, 유지 관리 및 크기 조정 프로세스를 통해 해당 인스턴스에서 컨테이너를 실행합니다. Kubernetes를 사용하면 온프레미스 및 클라우드에서 동일한 도구 세트로 모든 유형의 컨테이너화된 애플리케이션을 실행할 수 있습니다. 자세한 내용은 Kubernetes 설명서: 시작하기를 참조하세요.
- Amazon FSx를 통해 널리 사용되는 AWS의 완전관리형 파일 시스템을 시작하고 실행할 수 있습니다. Amazon FSx를 사용하면 일반적인 오픈 소스 및 상용 라이선스 파일 시스템의 기능 세트와 성능을 활용하여 시간이 많이 소요되는 관리 태스크를 피할 수 있습니다. 자세한 내용은 Amazon FSx 설명서를 참조하세요.
- Amazon Simple Notification Service (SNS)는 애플리케이션 간 통신과 애플리케이션 대 개인 통신 모두를 위한 완전관리형 메시징 서비스입니다. Application Insights를 통해 모니터링하도록 Amazon SNS를 구성할 수 있습니다. Amazon SNS가 모니터링을 위한 리소스로 구성된 경우 Application

Insights는 SNS 지표를 추적하여 SNS 메시지에 문제가 발생하거나 실패할 수 있는 이유를 파악할 수 있습니다.

- Amazon Elastic File System(Amazon EFS)는AWS 클라우드 서비스 및 온프레미스 리소스와 함께 사용할 수 있는 완전 관리형 엘라스틱 NFS 파일 시스템입니다. 애플리케이션 중단 없이 필요에 따라 페타바이트까지 확장할 수 있도록 구축되었습니다. 파일을 추가 및 제거하면 용량이 자동으로 확장 및 축소되므로 확장을 수용하기 위해 용량을 프로비저닝하고 관리할 필요가 없습니다. 자세한 내용은 Amazon Elastic File System 문서를 참조하세요.

관련 서드 파티 서비스

- Application Insights에서 모니터링되는 일부 워크로드 및 애플리케이션의 경우 Prometheus JMX Exporter가 AWS Systems Manager Distributor를 사용하여 설치되므로 CloudWatch Application Insights가 Java 관련 지표를 검색할 수 있습니다. Java 애플리케이션을 모니터링하도록 선택하면 Application Insights가 자동으로 Prometheus JMX Exporter를 설치합니다.

지원되는 애플리케이션 구성 요소

CloudWatch Application Insights는 리소스 그룹을 검색하여 애플리케이션 구성 요소를 식별합니다. 구성 요소는 독립 실행형, 자동 그룹형(예: Auto Scaling 그룹의 인스턴스 또는 로드 밸런서 뒤의 인스턴스) 또는 사용자 지정(개별 Amazon EC2 인스턴스 그룹화) 형식일 수 있습니다.

CloudWatch Application Insights는 다음 구성 요소를 지원합니다.

AWS 구성 요소

- Amazon EC2
- Amazon EBS
- Amazon RDS
- Elastic Load Balancing: Application Load Balancer 및 Classic Load Balancer(이러한 로드 밸런서의 모든 대상 인스턴스가 식별되고 구성됨)
- Amazon EC2 Auto Scaling 그룹: AWS Auto Scaling(오토 스케일링 그룹은 모든 대상 인스턴스에 대해 동적으로 구성되며, 애플리케이션이 확장되면 CloudWatch Application Insights가 자동으로 새 인스턴스 구성함) Auto Scaling 그룹은 CloudFormation 스택 기반 리소스 그룹에서 지원되지 않습니다.
- AWS Lambda
- Amazon Simple Queue Service(Amazon SQS)

- Amazon DynamoDB 테이블
- Amazon S3 버킷 지표
- AWS Step Functions
- Amazon API Gateway REST API 스테이지
- Amazon Elastic Container Service(Amazon ECS): 클러스터, 서비스, 태스크
- Amazon Elastic Kubernetes Service(Amazon EKS): 클러스터
- Amazon EC2의 Kubernetes: EC2에서 실행되는 Kubernetes 클러스터
- Amazon SNS 주제

CloudWatch Application Insights는 현재 다른 구성 요소 유형 리소스를 추적하지 않습니다. 지원되는 구성 요소 유형이 Application Insights 애플리케이션에 나타나지 않으면 해당 구성 요소가 이미 Application Insights에서 모니터링 중인 다른 애플리케이션에서 등록되고 관리되는 것일 수 있습니다.

지원되는 기술 스택

CloudWatch Application Insights를 사용하면 다음 기술 중 하나에 대한 애플리케이션 티어 드롭다운 메뉴 옵션을 선택하여 Windows Server 및 Linux 운영 체제에서 실행되는 애플리케이션을 모니터링할 수 있습니다.

- 프런트 엔드: Microsoft 인터넷 정보 서비스(IIS) 웹 서버
- 작업자 티어:
  - .NET Framework
  - .NET Core
- 애플리케이션:
  - Java
  - SAP NetWeaver 표준, 분산, 고가용성 배포
- Active Directory
- SharePoint
- 데이터베이스
  - Amazon RDS 또는 Amazon EC2에서 실행되는 Microsoft SQL Server(SQL Server 고가용성 구성 포함, 구성 요소 구성 예제 단원 참조)
  - Amazon RDS, Amazon Aurora 또는 Amazon EC2에서 실행되는 MySQL
  - Amazon RDS 또는 Amazon EC2에서 실행되는 PostgreSQL

- Amazon DynamoDB 테이블
- Amazon RDS 또는 Amazon EC2에서 실행되는 Oracle
- 단일 Amazon EC2 인스턴스와 여러 EC2 인스턴스의 SAP HANA 데이터베이스
- 교차 AZ SAP HANA 데이터베이스 고가용성 설정
- 단일 Amazon EC2 인스턴스의 SAP Sybase ASE 데이터베이스
- 교차 AZ SAP Sybase ASE 데이터베이스 고가용성 설정

위에 나열된 기술 스택이 애플리케이션 리소스에 적용되지 않는 경우 [모니터링 관리(Manage monitoring)] 페이지의 애플리케이션 티어 드롭다운 메뉴에서 [사용자 지정(Custom)]을 선택하여 애플리케이션 스택을 모니터링할 수 있습니다.

Amazon CloudWatch Application Insights 작동 방식

이 단원에서는 다음을 포함하여 CloudWatch Application Insights 작동 방식에 관해 설명합니다.

- Application Insights가 애플리케이션을 모니터링하는 방식
- 데이터 보존
- 할당량
- CloudWatch Application Insights에서 사용하는 AWS Systems Manager(SSM) 패키지
- CloudWatch Application Insights에서 사용하는 AWS Systems Manager(SSM) 문서

Application Insights가 애플리케이션을 모니터링하는 방식

Application Insights는 다음과 같이 애플리케이션을 모니터링합니다.

애플리케이션 검색 및 구성

애플리케이션이 CloudWatch Application Insights에 처음 추가되면 Application Insights는 애플리케이션 구성 요소를 검색하여 애플리케이션에 대해 모니터링할 주요 지표, 로그, 기타 데이터 원본을 권장합니다. 그러면 이러한 권장 사항을 기반으로 애플리케이션을 구성할 수 있습니다.

데이터 사전 처리

CloudWatch Application Insights는 애플리케이션 리소스 전반에서 모니터링되는 데이터 원본을 지속적으로 분석하여 지표 이상 및 로그 오류를 발견합니다(관찰).

지능형 문제 감지

CloudWatch Application Insights 엔진은 분류 알고리즘과 기본 제공 규칙을 사용하여 관찰을 상호 연결함으로써 애플리케이션의 문제를 감지합니다. 문제 해결을 돕기 위해 문제에 대한 상황별 정보가 포함된 자동 CloudWatch 대시보드를 만듭니다.

알림 및 작업

CloudWatch Application Insights는 애플리케이션 관련 문제를 감지하면 CloudWatch Events를 생성하여 문제를 알려 줍니다. 이러한 이벤트를 설정하는 방법에 대한 자세한 내용은 Application Insights CloudWatch Events 및 감지된 문제에 대한 알림 단원을 참조하세요.

예제 시나리오

SQL Server 데이터베이스가 지원하는 ASP .NET 애플리케이션이 있습니다. 갑자기 메모리 부족 문제가 발생하여 데이터베이스가 오작동하기 시작합니다. 이로 인해 애플리케이션 성능이 저하되고 웹 서버 및 로드 밸런서에서 HTTP 500 오류가 발생할 수 있습니다.

CloudWatch Application Insights 및 지능형 분석을 사용하면 동적으로 생성된 대시보드에서 관련 지표 및 로그 파일 조각을 확인하여 문제를 일으키는 애플리케이션 계층을 식별할 수 있습니다. 이 경우 SQL 데이터베이스 계층에 문제가 있을 수 있습니다.

데이터 보존

CloudWatch Application Insights는 55일 동안 문제를 유지하고 60일 동안 관찰을 유지합니다.

할당량

CloudWatch Application Insights의 기본 할당량은 Amazon CloudWatch Application Insights 엔드포인트 및 할당량 단원을 참조하세요. 별도로 명시되지 않는 한 각 할당량은 AWS 리전별로 적용됩니다. 서비스 할당량 증가를 요청하려면 AWS Support에 문의하세요. 많은 서비스에는 변경할 수 없는 할당량이 포함되어 있습니다. 특정 서비스의 할당량에 대한 자세한 내용은 해당 서비스 설명서를 참조하세요.

CloudWatch Application Insights에서 사용하는 AWS Systems Manager(SSM) 패키지

이 섹션에 나열된 패키지는 Application Insights에서 사용하며 AWS Systems Manager Distributor를 통해 개별적으로 관리하고 배포할 수 있습니다. SSM Distributor에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Distributor 단원을 참조하세요.

패키지:

- AWSObservabilityExporter-JMXExporterInstallAndConfigure
- AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure

- AWSObservabilityExporter-HAClusterExporterInstallAndConfigure
- AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure
- AWSObservabilityExporter-SQLExporterInstallAndConfigure

`AWSObservabilityExporter-JMXExporterInstallAndConfigure`

Application Insights의 Prometheus JMX Exporter에서 워크로드별 Java 지표를 검색하여 경보를 구성하고 모니터링할 수 있습니다. Application Insights 콘솔의 [모니터링 관리(Manage monitoring)] 페이지에 있는 [애플리케이션 티어(Application tier)] 드롭다운에서 [JAVA 애플리케이션(JAVA application)]을 선택합니다. 그런 다음, [JAVA Prometheus Exporter 구성(JAVA Prometheus exporter configuration)]에서 [수집 방법(Collection method)] 및 [JMX 포트 번호(JMX port number)]를 선택합니다.

AWS Systems Manager Distributor를 사용하여 Application Insights와 별도로 AWS 제공 Prometheus JMX Exporter 패키지를 패키징하고 설치하며 구성하려면 다음 단계를 완료합니다.

Prometheus JMX Exporter SSM 패키지를 사용하기 위한 사전 조건

- SSM Agent 버전 2.3.1550.0 이상이 설치되어 있어야 합니다.
- JAVA_HOME 환경 변수가 설정되어 있어야 합니다.

`AWSObservabilityExporter-JMXExporterInstallAndConfigure` 패키지 설치 및 구성

`AWSObservabilityExporter-JMXExporterInstallAndConfigure` 패키지는 Prometheus JMX Exporter를 설치하고 구성하는 데 사용할 수 있는 SSM Distributor 패키지입니다. Prometheus JMX Exporter가 Java 지표를 전송하면 CloudWatch 서비스의 지표를 검색하도록 CloudWatch 에이전트를 구성할 수 있습니다.

1. 기본 설정을 기반으로 Prometheus GitHub 리포지토리에 위치한 Prometheus JMX Exporter YAML 구성 파일을 준비합니다. 예제 구성 및 옵션 설명을 사용하여 안내합니다.
2. Base64로 인코딩된 Prometheus JMX Exporter YAML 구성 파일을 SSM 파라미터 스토어의 새 SSM 파라미터에 복사합니다.
3. SSM Distributor 콘솔로 이동하여 Amazon 소유(Owned by Amazon) 탭을 엽니다. [AWSObservabilityExporter-JMXExporterInstallAndConfigure]를 선택하고 [일회 설치(Install one time)]를 선택합니다.
4. "추가 인수"를 다음으로 대체하여 첫 번째 단계에서 생성한 SSM 파라미터를 업데이트합니다.

```
{
```

```
  "SSM_EXPORTER_CONFIGURATION": "{{ssm:<SSM_PARAMETER_STORE_NAME>}}",
  "SSM_EXPOSITION_PORT": "9404"
}
```

> **Note**
>
> 포트 9404는 Prometheus JMX 지표를 전송하는 데 사용되는 기본 포트입니다. 이 포트를 업데이트할 수 있습니다.

예: Java 지표를 검색하도록 CloudWatch 에이전트 구성

1. 이전 절차에서 설명한 대로 Prometheus JMX Exporter를 설치합니다. 그런 다음, 포트 상태를 확인하여 인스턴스에 올바르게 설치되었는지 확인합니다.

   Windows 인스턴스에서의 성공적인 설치 예

   ```
   PS C:\> curl http://localhost:9404 (http://localhost:9404/)
   StatusCode : 200
   StatusDescription : OK
   Content : # HELP jvm_info JVM version info
   ```

   Linux 인스턴스에서의 성공적인 설치 예

   ```
   $ curl localhost:9404
   # HELP jmx_config_reload_failure_total Number of times configuration have failed to
    be reloaded.
   # TYPE jmx_config_reload_failure_total counter
   jmx_config_reload_failure_total 0.0
   ```

2. Prometheus 서비스 검색 YAML 파일을 생성합니다. 아래에 나와 있는 서비스 검색 파일 예에서는 다음을 수행합니다.

   - Prometheus JMX Exporter 호스트 포트를 `localhost: 9404`로 지정합니다.
   - CloudWatch 지표 측정기준으로 설정할 수 있는 레이블(`Application`, `ComponentName`, `InstanceId`)을 지표에 연결합니다.

   ```
   $ cat prometheus_sd_jmx.yaml
   - targets:
   ```

```
  - 127.0.0.1:9404
  labels:
    Application: myApp
    ComponentName: arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/sampl-Appli-MMZW8E3GH4H2/aac36d7fea2a6e5b
    InstanceId: i-12345678901234567
```

3. Prometheus JMX Exporter 구성 YAML 파일을 생성합니다. 아래에 나와 있는 구성 파일 예에서는 다음을 지정합니다.

   - 지표 검색 작업 간격 및 시간 제한 기간
   - 작업 이름, 한 번에 반환되는 최대 시계열, 서비스 검색 파일 경로를 포함하는 지표 검색 작업(jmx 및 sap)으로 스크레이핑이라고도 함

```
$ cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: jmx
    sample_limit: 10000
    file_sd_configs:
      - files: ["/tmp/prometheus_sd_jmx.yaml"]
  - job_name: sap
    sample_limit: 10000
    file_sd_configs:
      - files: ["/tmp/prometheus_sd_sap.yaml"]
```

4. CloudWatch 에이전트가 Amazon EC2 인스턴스에 설치되어 있고 버전이 1.247346.1b249759 이상인지 확인합니다. EC2 인스턴스에 CloudWatch 에이전트를 설치하려면 CloudWatch 에이전트 설치 단원을 참조하세요. 버전을 확인하려면 CloudWatch 에이전트 버전에 관한 정보 찾기 단원을 참조하세요.
5. CloudWatch 에이전트를 구성합니다. CloudWatch 에이전트 구성 파일을 구성하는 방법에 대한 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하세요. 아래에 나와 있는 CloudWatch 에이전트 구성 파일 예에서는 다음을 수행합니다.

   - Prometheus JMX Exporter 구성 파일 경로를 지정합니다.
   - EMF 지표 로그를 게시할 대상 로그 그룹을 지정합니다.
   - 각 지표 이름에 대해 두 세트의 측정기준을 지정합니다.

- CloudWatch 지표 8개(지표 이름 4개 * 지표 이름당 측정기준 세트 2개)를 전송합니다.

```
{
   "logs":{
      "logs_collected":{
         ....
      },
      "metrics_collected":{
         "prometheus":{
            "cluster_name":"prometheus-test-cluster",
            "log_group_name":"prometheus-test",
            "prometheus_config_path":"/tmp/prometheus.yaml",
            "emf_processor":{
               "metric_declaration_dedup":true,
               "metric_namespace":"CWAgent",
               "metric_unit":{
                  "jvm_threads_current":"Count",
                  "jvm_gc_collection_seconds_sum":"Second",
                  "jvm_memory_bytes_used":"Bytes"
               },
               "metric_declaration":[
                  {
                     "source_labels":[
                        "job"
                     ],
                     "label_matcher":"^jmx$",
                     "dimensions":[
                        [
                           "InstanceId",
                           "ComponentName"
                        ],
                        [
                           "ComponentName"
                        ]
                     ],
                     "metric_selectors":[
                        "^java_lang_threading_threadcount$",
                        "^java_lang_memory_heapmemoryusage_used$",
                        "^java_lang_memory_heapmemoryusage_committed$"
                     ]
                  }
               ]
```

```
            }
         }
      }
   },
   "metrics":{
      ....
   }
}
```

AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure

Application Insights의 Prometheus HANA 데이터베이스 Exporter에서 워크로드별 SAP HANA 지표를 검색하여 경보를 구성하고 모니터링할 수 있습니다. 자세한 내용은 이 안내서의 모니터링을 위해 SAP HANA 데이터베이스 설정 섹션을 참조하세요.

AWS Systems Manager Distributor를 사용하여 Application Insights와 별도로 AWS 제공 Prometheus HANA 데이터베이스 Exporter 패키지를 패키징하고, 설치하고, 구성하려면 다음 단계를 완료합니다.

Prometheus HANA 데이터베이스 Exporter SSM 패키지를 사용하기 위한 사전 조건

- SSM Agent 버전 2.3.1550.0 이상이 설치되어 있어야 합니다.
- SAP HANA 데이터베이스
- Linux 운영 체제(SUSE Linux, RedHat Linux)
- AWS Secrets Manager를 통해 SAP HANA 데이터베이스 모니터링 자격 증명을 사용한 보안 암호. 키/값 페어 형식을 사용해 보안 암호를 생성하고, 키 사용자 이름을 지정한 다음, 값에 대한 데이터베이스 사용자를 입력합니다. 두 번째 키 암호를 추가한 후 값에 암호를 입력합니다. 보안 암호 생성에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 보안 암호 생성을 참조하세요. 보안 암호는 다음과 같은 형식이어야 합니다.

```
{
  "username": "<database_user>",
  "password": "<database_password>"
}
```

AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure 패키지 설치 및 구성

AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure 패키지는 Prometheus HANA 데이터베이스 Exporter를 설치하고 구성하는 데 사용할 수 있는 SSM Distributor 패키지입니다. Prometheus HANA 데이터베이스 Exporter가 HANA 데이터베이스 지표를 전송하면 CloudWatch 서비스의 지표를 검색하도록 CloudWatch 에이전트를 구성할 수 있습니다.

1. SSM 파라미터 스토어에서 SSM 파라미터를 생성하여 Exporter 구성을 저장합니다. 다음은 파라미터 값의 예제입니다.

```
{\"exposition_port\":9668,\"multi_tenant\":true,\"timeout\":600,\"hana\":{\"host\":
\"localhost\",\"port\":30013,\"aws_secret_name\":\"HANA_DB_CREDS\",\"scale_out_mode
\":true}}
```

> **Note**
>
> 이 예제에서 내보내기는 SYSTEM 데이터베이스가 활성 상태인 Amazon EC2 인스턴스에서만 실행되며 중복 지표를 피하기 위해 다른 EC2 인스턴스에서는 유휴 상태로 유지됩니다. Exporter는 SYSTEM 데이터베이스에서 모든 데이터베이스 테넌트 정보를 검색할 수 있습니다.

2. SSM 파라미터 스토어에서 SSM 파라미터를 생성하여 Exporter 지표 쿼리를 저장합니다. 패키지는 둘 이상의 지표 파라미터를 적용할 수 있습니다. 각 파라미터에는 유효한 JSON 객체 형식이 있어야 합니다. 다음은 파라미터 값의 예제입니다.

```
{\"SELECT MAX(TIMESTAMP) TIMESTAMP, HOST, MEASURED_ELEMENT_NAME CORE,
 SUM(MAP(CAPTION, 'User Time', TO_NUMBER(VALUE), 0)) USER_PCT, SUM(MAP(CAPTION,
 'System Time', TO_NUMBER(VALUE), 0)) SYSTEM_PCT, SUM(MAP(CAPTION, 'Wait
 Time', TO_NUMBER(VALUE), 0)) WAITIO_PCT, SUM(MAP(CAPTION, 'Idle Time', 0,
 TO_NUMBER(VALUE))) BUSY_PCT, SUM(MAP(CAPTION, 'Idle Time', TO_NUMBER(VALUE), 0))
 IDLE_PCT FROM sys.M_HOST_AGENT_METRICS WHERE MEASURED_ELEMENT_TYPE = 'Processor'
 GROUP BY HOST, MEASURED_ELEMENT_NAME;\":{\"enabled\":true,\"metrics\":[{\"name\":
\"hanadb_cpu_user\",\"description\":\"Percentage of CPU time spent by HANA DB in user
 space, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":
\"USER_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_system
\",\"description\":\"Percentage of CPU time spent by HANA DB in Kernel space,
 over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":
\"SYSTEM_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_waitio
\",\"description\":\"Percentage of CPU time spent by HANA DB in IO mode, over the
 last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":\"WAITIO_PCT\",
\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_busy\",\"description
\":\"Percentage of CPU time spent by HANA DB, over the last minute (in seconds)\",
```

```
\"labels\":[\"HOST\",\"CORE\"],\"value\":\"BUSY_PCT\",\"unit\":\"percent\",\"type\":
\"gauge\"},{\"name\":\"hanadb_cpu_idle\",\"description\":\"Percentage of CPU time not
 spent by HANA DB, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE
\"],\"value\":\"IDLE_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"}]}}
```

지표 쿼리에 대한 자세한 내용은 GitHub의 SUSE / hanadb_exporter 리포지토리를 참조하세요.

3. SSM Distributor 콘솔로 이동하여 Amazon 소유(Owned by Amazon) 탭을 엽니다. AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure*를 선택하고 한 번만 설치(Install one time)를 선택합니다.
4. "추가 인수"를 다음으로 대체하여 첫 번째 단계에서 생성한 SSM 파라미터를 업데이트합니다.

```
{
  "SSM_EXPORTER_CONFIG": "{{ssm:<*SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>*}}",
  "SSM_SID": "<SAP_DATABASE_SID>",
  "SSM_EXPORTER_METRICS_1": "{{ssm:<SSM_FIRST_METRICS_PARAMETER_STORE_NAME>}}",
  "SSM_EXPORTER_METRICS_2": "{{ssm:<SSM_SECOND_METRICS_PARAMETER_STORE_NAME>}}"
}
```

5. SAP HANA 데이터베이스가 있는 Amazon EC2 인스턴스를 선택한 다음 실행(Run)을 선택합니다.

`AWSObservabilityExporter-HAClusterExporterInstallAndConfigure`

Application Insights의 Prometheus HANA 클러스터 Exporter에서 워크로드별 고가용성(HA) 클러스터 지표를 검색하여 SAP HANA 데이터베이스 고가용성 설정에 대한 경보를 구성하고 모니터링할 수 있습니다. 자세한 내용은 이 안내서의 모니터링을 위해 SAP HANA 데이터베이스 설정 섹션을 참조하세요.

AWS Systems Manager Distributor를 사용하여 Application Insights와 별도로 AWS 제공 Prometheus HA Cluster Exporter 패키지를 패키징하고, 설치하고, 구성하려면 다음 단계를 완료합니다.

Prometheus HA Cluster Exporter SSM 패키지를 사용하기 위한 사전 조건

- SSM Agent 버전 2.3.1550.0 이상이 설치되어 있어야 합니다.
- Pacemaker, Corosync, SBD 및 DRBD를 위한 HA 클러스터
- Linux 운영 체제(SUSE Linux, RedHat Linux)

`AWSObservabilityExporter-HAClusterExporterInstallAndConfigure` 패키지 설치 및 구성

AWSObservabilityExporter-HAClusterExporterInstallAndConfigure 패키지는 Prometheus HA Cluster Exporter를 설치하고 구성하는 데 사용할 수 있는 SSM Distributor 패키지입니다. Prometheus HANA 데이터베이스 Exporter가 클러스터 지표를 전송하면 CloudWatch 서비스의 지표를 검색하도록 CloudWatch 에이전트를 구성할 수 있습니다.

1. SSM 파라미터 스토어에서 SSM 파라미터를 생성하여 JSON 형식으로 Exporter 구성을 저장합니다. 다음은 파라미터 값의 예제입니다.

   ```
   {\"port\":\"9664\",\"address\":\"0.0.0.0\",\"log-level\":\"info\",\"crm-mon-path
   \":\"/usr/sbin/crm_mon\",\"cibadmin-path\":\"/usr/sbin/cibadmin\",\"corosync-
   cfgtoolpath-path\":\"/usr/sbin/corosync-cfgtool\",\"corosync-quorumtool-path\":\"/
   usr/sbin/corosync-quorumtool\",\"sbd-path\":\"/usr/sbin/sbd\",\"sbd-config-path\":
   \"/etc/sysconfig/sbd\",\"drbdsetup-path\":\"/sbin/drbdsetup\",\"enable-timestamps
   \":false}
   ```

   Exporter 구성에 대한 자세한 내용은 GitHub의 ClusterLabs / ha_cluster_exporter 리포지토리를 참조하세요.

2. SSM Distributor 콘솔로 이동하여 Amazon 소유(Owned by Amazon) 탭을 엽니다. AWSObservabilityExporter-HAClusterExporterInstallAndConfigure*를 선택한 다음 한 번만 설치(Install one time)를 선택합니다.
3. "추가 인수"를 다음으로 대체하여 첫 번째 단계에서 생성한 SSM 파라미터를 업데이트합니다.

   ```
   {
     "SSM_EXPORTER_CONFIG": "{{ssm:<*SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>*}}"
   }
   ```

4. SAP HANA 데이터베이스가 있는 Amazon EC2 인스턴스를 선택한 다음 실행(Run)을 선택합니다.

AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure

Application Insights용 Prometheus SAP 호스트 내보내기 도구에서 워크로드별 SAP NetWeaver 지표를 검색하여 SAP NetWeaver 분산 및 High Availability 배포에 대한 경보를 구성하고 모니터링할 수 있습니다. 자세한 내용은 Amazon CloudWatch Application Insights 시작하기 단원을 참조하십시오.

AWS Systems Manager Distributor를 사용하여 Application Insights와 별도로 SAP 호스트 내보내기 패키지를 패키징, 설치 및 구성하려면 다음 단계를 수행하세요.

Prometheus SAP 호스트 내보내기 SSM 패키지를 사용하기 위한 사전 조건

- SSM Agent 버전 2.3.1550.0 이상이 설치되어 있어야 합니다.
- SAP NetWeaver 애플리케이션 서버
- Linux 운영 체제(SUSE Linux, RedHat Linux)

AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure 패키지 설치 및 구성

AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure 패키지는 SAP NetWeaver Prometheus 지표 내보내기를 설치하고 구성하는 데 사용할 수 있는 SSM Distributor 패키지입니다. Prometheus 내보내기가 SAP NetWeaver 지표를 전송하면 CloudWatch 서비스의 지표를 검색하도록 CloudWatch 에이전트를 구성할 수 있습니다.

1. SSM 파라미터 스토어에서 SSM 파라미터를 생성하여 JSON 형식으로 Exporter 구성을 저장합니다. 다음은 파라미터 값의 예제입니다.

   ```
   {\"address\":\"0.0.0.0\",\"port\":\"9680\",\"log-level\":\"info\",\"is-HA\":false}
   ```

   - address

     Prometheus 지표를 전송할 대상 주소입니다. 기본 값은 localhost입니다.

   - port

     Prometheus 지표를 전송할 대상 포트입니다. 기본 값은 9680입니다.

   - is-HA

     SAP NetWeaver High Availability 배포용 true. 다른 모든 배포의 경우 값은 false입니다.

2. SSM Distributor 콘솔로 이동하여 Amazon 소유(Owned by Amazon) 탭을 엽니다. AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure를 선택하고 Install one time(한 번 설치)을 선택합니다.
3. "추가 인수"를 다음으로 대체하여 첫 번째 단계에서 생성한 SSM 파라미터를 업데이트합니다.

   ```
   {
     "SSM_EXPORTER_CONFIG": "{{ssm:<SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>}}",
     "SSM_SID": "<SAP_DATABASE_SID>",
     "SSM_INSTANCES_NUM": "<instances_number seperated by comma>"
   }
   ```

예

```
{
  "SSM_EXPORTER_CONFIG": "{{ssm:exporter_config_paramter}}",
  "SSM_INSTANCES_NUM": "11,12,10",
  "SSM_SID": "PR1"
}
```

4. SAP NetWeaver 애플리케이션이 있는 Amazon EC2 인스턴스를 선택하고 Run(실행)을 선택합니다.

> **Note**
>
> Prometheus Exporter는 로컬 엔드포인트에서 SAP NetWeaver 지표를 서비스합니다. Amazon EC2 인스턴스의 운영 체제 사용자만 로컬 엔드포인트에 액세스할 수 있습니다. 따라서 내보내기 도구 패키지가 설치된 후 모든 운영 체제 사용자가 지표를 사용할 수 있습니다. 기본 로컬 엔드포인트는 `localhost:9680/metrics`입니다.

`AWSObservabilityExporter-SQLExporterInstallAndConfigure`

Prometheus SQL 내보내기에서 Application Insights에 대한 워크로드별 SQL Server 지표를 검색하여 주요 지표를 모니터링할 수 있습니다.

AWS Systems Manager Distributor를 사용하여 Application Insights와 별도로 SQL 내보내기 패키지를 패키징, 설치, 구성하려면 다음 단계를 수행하세요.

Prometheus SQL 내보내기 SSM 패키지 사용을 위한 사전 조건

- SSM Agent 버전 2.3.1550.0 이상이 설치되어 있어야 합니다.
- SQL Server 사용자 인증을 사용하도록 설정된 Windows에서 SQL Server를 실행하는 Amazon EC2 인스턴스.
- 다음 권한이 있는 SQL Server 사용자:

```
GRANT VIEW ANY DEFINITION TO
```

```
GRANT VIEW SERVER STATE TO
```

- AWS Secrets Manager를 사용하는 데이터베이스 연결 문자열이 포함된 암호. 보안 암호 생성에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 보안 암호 생성을 참조하세요. 보안 암호는 다음과 같은 형식이어야 합니다.

```
{
  "data_source_name":"sqlserver://<username>:<password>@localhost:1433"
}
```

> **Note**
>
> 암호 또는 사용자 이름에 특수 문자가 포함된 경우 데이터베이스에 성공적으로 연결하려면 특수 문자를 퍼센트 인코딩해야 합니다.

`AWSObservabilityExporter-SQLExporterInstallAndConfigure` 패키지 설치 및 구성

`AWSObservabilityExporter-SQLExporterInstallAndConfigure` 패키지는 SQL Prometheus 지표 내보내기를 설치하고 구성하는 데 사용할 수 있는 SSM Distributor 패키지입니다. Prometheus 내보내기가 지표를 전송하면 CloudWatch 에이전트가 CloudWatch 서비스에 대한 지표를 검색하도록 구성할 수 있습니다.

1. 기본 설정에 따라 SQL Exporter YAML 구성을 준비합니다. 다음 샘플 구성에는 단일 지표가 구성되어 있습니다. 예 구성을 사용하여 추가 지표로 구성을 업데이트하거나 고유한 구성을 생성할 수 있습니다.

```
---
global:
  scrape_timeout_offset: 500ms
  min_interval: 0s
  max_connections: 3
  max_idle_connections: 3
target:
  aws_secret_name: <SECRET_NAME>
  collectors:
    - mssql_standard
collectors:
  - collector_name: mssql_standard
    metrics:
    - metric_name: mssql_batch_requests
      type: counter
```

```
      help: 'Number of command batches received.'
      values: [cntr_value]
      query: |
        SELECT cntr_value
        FROM sys.dm_os_performance_counters WITH (NOLOCK)
        WHERE counter_name = 'Batch Requests/sec'
```

2. Base64로 인코딩된 Prometheus SQL Exporter YAML 구성 파일을 SSM 파라미터 스토어의 새 SSM 파라미터에 복사합니다.
3. SSM Distributor 콘솔로 이동하여 Amazon 소유(Owned by Amazon) 탭을 엽니다. AWSObservabilityExporter-SQLExporterInstallAndConfigure를 선택하고 한 번 설치를 선택합니다.
4. '추가 인수'를 다음 정보로 변경하세요. SSM_PARAMETER_NAME은 2단계에서 생성한 파라미터의 이름입니다.

```
{
  "SSM_EXPORTER_CONFIGURATION":
    "{{ssm:<SSM_PARAMETER_STORE_NAME>}}",
    "SSM_PROMETHEUS_PORT": "9399",
    "SSM_WORKLOAD_NAME": "SQL"
}
```

5. SQL Server 데이터베이스가 있는 Amazon EC2 인스턴스를 선택한 다음 실행을 선택합니다.

CloudWatch Application Insights에서 사용하는 AWS Systems Manager(SSM) 문서

Application Insights는 이 섹션에 나열된 SSM 문서를 사용하여 AWS Systems Manager가 관리형 인스턴스에서 수행하는 작업을 정의합니다. 이러한 문서에서는 Systems Manager의 Run Command 기능을 사용하여 Application Insights 모니터링 기능을 수행하는 데 필요한 작업을 자동화합니다. 이러한 문서의 실행 일정은 Application Insights에서 관리하며 변경할 수 없습니다.

SSM 문서에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Documents를 참조하세요.

CloudWatch Application Insights에서 관리하는 문서

다음 표에는 Application Insights에서 관리하는 SSM 문서가 나와 있습니다.

문서 이름	설명	실행 일정
AWSEC2-DetectWorkload	Application Insights에서 모니터링하도록 설정할 수 있는 애플리케이션 환경에서 실행 중인 애플리케이션을 자동으로 검색합니다.	이 문서는 애플리케이션 환경에서 매시간 실행되어 최신 애플리케이션 세부 정보를 제공합니다.
AWSEC2-CheckPerformanceCounterSets	Amazon EC2 Windows 인스턴스에서 성능 카운터 네임스페이스가 활성화되어 있는지 확인합니다.	이 문서는 애플리케이션 환경에서 매시간 실행되며 해당 네임스페이스가 활성화된 경우에만 성능 카운터 지표를 모니터링합니다.
AWSEC2-ApplicationInsightsCloudwatchAgentInstallAndConfigure	애플리케이션 구성 요소의 모니터링 구성을 기반으로 CloudWatch 에이전트를 설치하고 구성합니다.	이 문서는 CloudWatch 에이전트 구성이 항상 정확하고 최신 상태인지 확인하기 위해 30분마다 실행됩니다. 또한 이 문서는 지표 추가 또는 제거, 로그 구성 업데이트 등 애플리케이션 모니터링 설정을 변경한 직후에도 실행됩니다.

AWS Systems Manager에서 관리하는 문서

다음 문서는 CloudWatch Application Insights에서 사용하고 Systems Manager에서 관리합니다.

AWS-ConfigureAWSPackage

Application Insights는 이 문서를 사용하여 Prometheus 내보내기 배포자 패키지를 설치 및 제거하고, 워크로드별 지표를 수집하고, 고객 Amazon EC2 인스턴스의 워크로드를 포괄적으로 모니터링할 수 있습니다. CloudWatch Application Insights는 상관관계가 있는 대상 워크로드가 인스턴스에서 실행 중인 경우에만 Prometheus 내보내기 배포자 패키지를 설치합니다.

다음 표에는 Prometheus 내보내기 배포자 패키지와 상관관계가 있는 대상 워크로드가 나열되어 있습니다.

Prometheus 내보내기 배포자 패키지 이름	워크로드 배포
AWSObservabilityExporter-HAClusterExporterInstallAndConfigure	SAP HANA HA
AWSObservabilityExporter-JMXExporterInstallAndConfigure	Java/JMX
AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure	SAP HANA
AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure	NetWeaver
AWSObservabilityExporter-SQLExporterInstallAndConfigure	SQL Server(Windows) 및 SAP ASE(Linux)

`AmazonCloudWatch-ManageAgent`

Application Insights는 이 문서를 사용하여 인스턴스의 CloudWatch Agent의 상태와 구성을 관리하고, 운영 체제 전반의 Amazon EC2 인스턴스로부터 내부 시스템 수준 지표와 로그를 수집합니다.

Amazon CloudWatch Application Insights 시작하기

CloudWatch Application Insights를 시작하려면 다음 사전 조건을 충족하고 IAM 정책을 생성했는지 확인합니다. 그런 다음, 콘솔 링크를 사용하여 CloudWatch Application Insights를 사용 설정할 수 있습니다. 애플리케이션 리소스를 구성하려면 모니터링을 위해 애플리케이션 설정, 구성, 관리 단원의 단계를 따르세요.

내용

- CloudWatch Application Insights 액세스
- 필수 조건
- IAM 정책
- 계정 기반 애플리케이션 온보딩을 위한 IAM 역할 권한

- 모니터링을 위해 애플리케이션 설정, 구성, 관리

CloudWatch Application Insights 액세스

다음 인터페이스 중 하나를 통해 CloudWatch Application Insights에 액세스하여 관리할 수 있습니다.

- CloudWatch 콘솔. 애플리케이션에 대한 모니터를 추가하려면 CloudWatch 콘솔의 왼쪽 탐색 창에 있는 인사이트(Insights)에서 Application Insights를 선택합니다. 애플리케이션을 구성한 후 CloudWatch 콘솔을 사용하여 감지된 문제점을 보고 분석할 수 있습니다.
- AWS Command Line Interface(AWS CLI). AWS CLI를 사용하여 AWS API 작업에 액세스할 수 있습니다. 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 AWS 명령줄 인터페이스 설치 단원을 참조하세요. Application Insights API 정보에 대한 자세한 내용은 Amazon CloudWatch Application Insights API 참조를 참조하세요.

필수 조건

CloudWatch Application Insights로 애플리케이션을 구성하려면 다음 사전 조건을 충족해야 합니다.

- AWS Systems Manager 활성화 - Amazon EC2 인스턴스에 SSM Agent(Systems Manager Agent)를 설치하고 SSM에 대해 인스턴스를 활성화합니다. SSM Agent를 설치하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager 설정을 참조하세요.
- EC2 인스턴스 역할 – Systems Manager를 활성화하려면 다음 Amazon EC2 인스턴스 역할을 연결해야 합니다.
  - Systems Manager를 사용 설정하려면 `AmazonSSMManagedInstanceCore` 역할을 연결해야 합니다. 자세한 내용은 AWS Systems Manager 자격 증명 기반 정책 예제를 참조하세요.
  - CloudWatch를 통해 인스턴스 지표 및 로그를 내보낼 수 있게 하려면 `CloudWatchAgentServerPolicy` 정책을 연결해야 합니다. 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성을 참조하세요.
- AWS 리소스 그룹 - CloudWatch Application Insights에 애플리케이션을 온보딩하려면 애플리케이션 스택에서 사용하는 연결된 모든 AWS 리소스를 포함하는 리소스 그룹을 생성합니다. 이 그룹에는 애플리케이션 로드 밸런서, IIS 및 웹 프런트 엔드에서 실행 중인 Amazon EC2 인스턴스, .NET 작업자 티어, SQL Server 데이터베이스가 포함됩니다. Application Insights에서 지원하는 애플리케이션 구성 요소 및 기술 스택에 대한 자세한 내용은 지원되는 애플리케이션 구성 요소의 내용을 참조하세요. Auto Scaling 그룹은 CloudFormation 리소스 그룹에서 지원되지 않기 때문에 CloudWatch Application Insights는 리소스 그룹과 동일한 태그 또는 CloudFormation 스택을 사용하여 Auto

Scaling 그룹을 자동으로 포함합니다. 자세한 내용은 AWS Resource Groups 시작하기 단원을 참조하세요.

- IAM 권한 - 관리 액세스 권한이 없는 사용자의 경우 서비스 연결 역할을 생성하고 이를 사용자의 아이덴티티에 연결할 수 있도록 Application Insights를 허용하는 AWS Identity and Access Management(IAM) 정책을 생성해야 합니다. IAM 정책을 생성하는 방법에 대한 자세한 내용은 IAM 정책의 내용을 참조하세요.
- 서비스 연결 역할 - Application Insights는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Application Insights 관리 콘솔에서 첫 번째 Application Insights 애플리케이션을 생성할 때 생성됩니다. 자세한 내용은 CloudWatch Application Insights에 서비스 연결 역할 사용 단원을 참조하십시오.
- EC2 Windows 인스턴스용 성능 카운터 지표 지원: Amazon EC2 Windows 인스턴스에서 성능 카운터 지표를 모니터링하려면 인스턴스에 성능 카운터를 설치해야 합니다. 성능 카운터 지표와, 이에 해당하는 성능 카운터 이름은 성능 카운터 지표를 참조하세요. 성능 카운터에 대한 자세한 내용은 Performance Counters(성능 카운터)를 참조하세요.
- Amazon CloudWatch 에이전트 – Application Insights가 CloudWatch 에이전트를 설치하고 구성합니다. CloudWatch 에이전트를 설치하면 Application Insights가 구성을 유지합니다. 병합 충돌을 방지하려면 Application Insights에서 사용하려는 리소스 구성을 기존 CloudWatch 에이전트 구성 파일에서 제거하세요. 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하십시오.

IAM 정책

CloudWatch Application Insights를 사용하려면 AWS Identity and Access Management(IAM) 정책을 생성하여 사용자, 그룹 또는 역할에 연결해야 합니다. 사용자, 그룹 및 역할에 대한 자세한 내용은 IAM 자격 증명(사용자, 사용자 그룹 및 역할)을 참조하세요. IAM 정책은 사용자 권한을 정의합니다.

콘솔을 사용해 IAM 정책을 생성하려면

IAM 콘솔을 사용하여 IAM 정책을 생성하려면 다음 단계를 수행하세요.

1. IAM 콘솔로 이동합니다. 왼쪽 탐색 창에서 정책을 선택합니다.
2. 페이지 상단에서 정책 생성을 선택합니다.
3. JSON 탭을 선택합니다.
4. 다음 JSON 문서를 복사하여 JSON 탭에 붙여 넣습니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "applicationinsights:*",
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
                "resource-groups:ListGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

5. 정책 검토를 선택합니다.
6. 정책의 이름을 입력합니다(예: “AppInsightsPolicy”). 필요한 경우 설명을 입력합니다.
7. 정책 생성을 선택합니다.
8. 왼쪽 탐색 창에서 사용자 그룹, 사용자 또는 역할을 선택합니다.
9. 정책을 첨부할 사용자 그룹, 사용자 또는 역할의 이름을 선택합니다.
10. Add permissions(권한 추가)를 선택합니다.
11. 기존 정책 직접 연결을 선택합니다.
12. 방금 만든 정책을 검색하고 정책 이름 왼쪽에 있는 확인란을 선택합니다.
13. Next: Review(다음: 검토)를 선택합니다.
14. 올바른 정책이 나열되어 있는지 확인하고 Add permissions(권한 추가)를 선택합니다.
15. CloudWatch Application Insights를 사용할 때 방금 생성한 정책과 연결된 사용자로 로그인해야 합니다.

AWS CLI를 사용하여 IAM 정책을 생성하려면

AWS CLI를 사용하여 IAM 정책을 생성하려면 위 JSON 문서를 현재 폴더의 파일로 사용해 명령줄에서 create-policy 작업을 실행합니다.

AWS Tools for Windows PowerShell을 사용하여 IAM 정책을 생성하려면

AWS Tools for Windows PowerShell을 사용하여 IAM 정책을 생성하려면 위 JSON 문서를 현재 폴더의 파일로 사용해 New-IAMPolicy cmdlt를 실행합니다.

계정 기반 애플리케이션 온보딩을 위한 IAM 역할 권한

계정에 있는 모든 리소스를 온보딩하려는 경우 애플리케이션 인사이트 기능에 전체 액세스하기 위한 Application Insights 관리 정책을 사용하지 않음을 선택하고 Application Insights가 계정의 모든 리소스를 검색할 수 있도록 IAM 역할에 다음 권한을 연결해야 합니다.

```
"ec2:DescribeInstances"
"ec2:DescribeNatGateways"
"ec2:DescribeVolumes"
"ec2:DescribeVPCs"
"rds:DescribeDBInstances"
"rds:DescribeDBClusters"
"sqs:ListQueues"
"elasticloadbalancing:DescribeLoadBalancers"
"autoscaling:DescribeAutoScalingGroups"
"lambda:ListFunctions"
"dynamodb:ListTables"
"s3:ListAllMyBuckets"
"sns:ListTopics"
"states:ListStateMachines"
"apigateway:GET"
"ecs:ListClusters"
"ecs:DescribeTaskDefinition"
"ecs:ListServices"
"ecs:ListTasks"
"eks:ListClusters"
"eks:ListNodegroups"
"fsx:DescribeFileSystems"
"route53:ListHealthChecks"
"route53:ListHostedZones"
"route53:ListQueryLoggingConfigs"
"route53resolver:ListFirewallRuleGroups"
"route53resolver:ListFirewallRuleGroupAssociations"
"route53resolver:ListResolverEndpoints"
"route53resolver:ListResolverQueryLogConfigs"
"route53resolver:ListResolverQueryLogConfigAssociations"
"logs:DescribeLogGroups"
"resource-explorer:ListResources"
```

모니터링을 위해 애플리케이션 설정, 구성, 관리

이 섹션에서는 콘솔, AWS CLI, AWS Tools for Windows PowerShell을 사용하여 CloudWatch Application Insights 애플리케이션을 설정하고, 구성하고, 관리하는 방법을 단계별로 안내합니다.

주제

- CloudWatch 콘솔에서 모니터링할 애플리케이션 설정, 구성, 관리
- 명령줄을 사용하여 모니터링할 애플리케이션 설정, 구성, 관리
- Application Insights CloudWatch Events 및 감지된 문제에 대한 알림

CloudWatch 콘솔에서 모니터링할 애플리케이션 설정, 구성, 관리

이 섹션에서는 CloudWatch 콘솔에서 모니터링할 애플리케이션을 설정하고, 구성하고, 관리하는 방법을 단계별로 안내합니다.

콘솔 절차

- 애플리케이션 추가 및 구성
- Amazon ECS 및 Amazon EKS 리소스 모니터링을 위한 Application Insights 활성화
- 애플리케이션 구성 요소에 대한 모니터링 사용 중지
- 애플리케이션을 삭제합니다

애플리케이션 추가 및 구성

CloudWatch 콘솔에서 애플리케이션 추가 및 구성

CloudWatch 콘솔에서 CloudWatch Application Insights를 시작하려면 다음 단계를 수행합니다.

1. 시작. CloudWatch 콘솔 랜딩 페이지를 엽니다. 왼쪽 탐색 창의 인사이트(Insights)에서 Application Insights를 선택합니다. 열린 페이지에는 CloudWatch Application Insights로 모니터링되는 애플리케이션 목록과 해당 모니터링 상태가 표시됩니다.
2. 애플리케이션 추가. 애플리케이션에 대한 모니터링을 설정하려면 애플리케이션 추가(Add an application)를 선택합니다. 애플리케이션 추가(Add an application)를 선택한 경우 애플리케이션 유형 선택(Choose Application Type) 메시지가 표시됩니다.
   - 리소스 그룹 기반 애플리케이션. 이 옵션을 선택하면 해당 계정에서 모니터링할 리소스 그룹을 선택할 수 있습니다. 구성 요소에서 여러 애플리케이션을 사용하려면 리소스 그룹 기반 모니터링을 사용해야 합니다.
   - 계정 기반 애플리케이션. 이 옵션을 선택하면 해당 계정의 모든 리소스를 모니터링할 수 있습니다. 계정의 모든 리소스를 모니터링하려면 애플리케이션 온보딩 프로세스가 리소스 그룹 기반 옵션보다 빠른 이 옵션을 사용하는 것이 좋습니다.

> **Note**
>
> Application Insights를 사용하여 리소스 그룹 기반 모니터링과 계정 기반 모니터링을 결합할 수 없습니다. 애플리케이션 유형을 변경하려면 모니터링 중인 모든 애플리케이션을 삭제하고 애플리케이션 유형을 선택합니다.

모니터링을 위해 첫 번째 애플리케이션을 추가하면 CloudWatch Application Insights는 계정에 서비스 연결 역할을 생성하여 사용자 대신 Application Insights 권한을 다른 AWS 사용자에게 호출할 수 있는 권한을 부여합니다. Application Insights가 계정에 생성한 서비스 연결 역할에 대한 자세한 내용은 CloudWatch Application Insights에 서비스 연결 역할 사용 섹션을 참조하세요.

3. Resource-based application monitoring

    1. 리소스 그룹 선택. 애플리케이션 세부 정보 지정(Specify application details) 페이지에서 드롭다운 목록의 애플리케이션 리소스가 포함된 AWS 리소스 그룹을 선택합니다. 이러한 리소스로는 프런트 엔드 서버, 로드 밸런서, auto scaling 그룹, 데이터베이스 서버 등을 들 수 있습니다.

        애플리케이션에 대한 리소스 그룹을 생성하지 않은 경우 새 리소스 그룹 생성(Create new resource group)을 선택해 생성할 수 있습니다. 리소스 그룹 생성에 대한 자세한 내용은 AWS Resource Groups 사용 설명서를 참조하세요.

    2. CloudWatch Events 모니터링. Amazon EBS, Amazon EC2, AWS CodeDeploy, Amazon ECS, AWS Health API 및 알림, Amazon RDS, Amazon S3, AWS Step Functions로부터 인사이트를 얻기 위해 Application Insights 모니터링을 CloudWatch Events와 통합하려면 확인란을 선택합니다.

    3. AWS Systems Manager OpsCenter와 통합. 선택한 애플리케이션에 문제가 감지될 때 이를 확인하고 알림을 받으려면 수정 조치를 위해 Systems Manager OpsCenter OpsItems 생성(Generate Systems Manager OpsCenter OpsItems for remedial actions) 확인란을 선택합니다. AWS 리소스와 관련된 운영 작업 항목(OpsItem)을 해결하기 위해 수행된 작업을 추적하려면 SNS 주제 ARN을 제공합니다.

    4. 태그(선택 사항). CloudWatch Application Insights는 태그 기반 및 CloudFormation 기반 리소스 그룹을 모두 지원합니다(Auto Scaling 그룹 제외). 자세한 내용은 Tag Editor 작업을 참조하세요.

    5. 다음을 선택합니다.

애플리케이션에 대한 ARN은 다음 형식으로 생성됩니다.

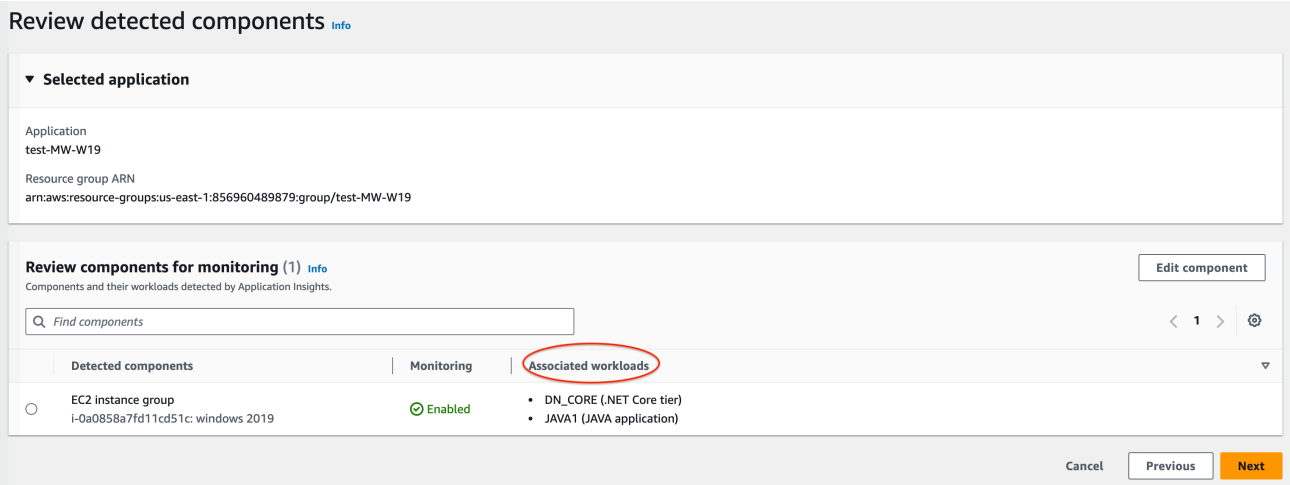
```
arn:partition:applicationinsights:region:account-id:application/resource-group/resource-group-name
```

예

```
arn:aws:applicationinsights:us-east-1:123456789012:application/resource-group/my-resource-group
```

6. 탐지된 구성 요소 검토 페이지의 모니터링할 구성 요소 검토 아래에 있는 테이블에는 탐지된 구성 요소 및 탐지된 관련 워크로드가 나열됩니다.

> **Note**
>
> 여러 개의 사용자 정의 워크로드를 지원하는 구성 요소의 경우 각 구성 요소에 대해 최대 5개의 워크로드를 모니터링할 수 있습니다. 이러한 워크로드는 구성 요소와 별도로 모니터링됩니다.

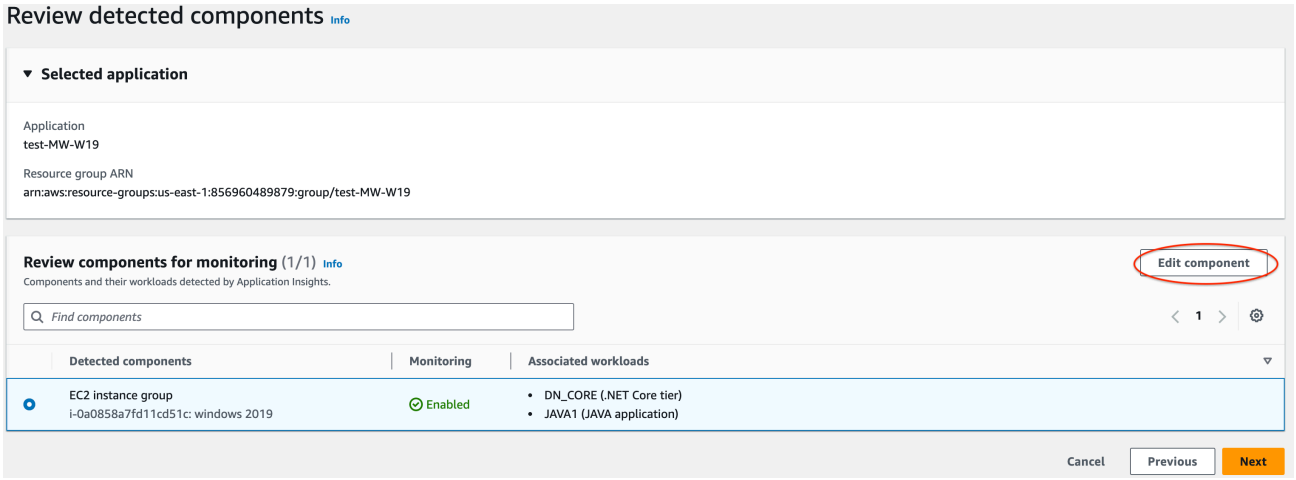


관련 워크로드 아래에는 워크로드가 목록에 없는 경우 표시될 수 있는 몇 가지 메시지가 있습니다.

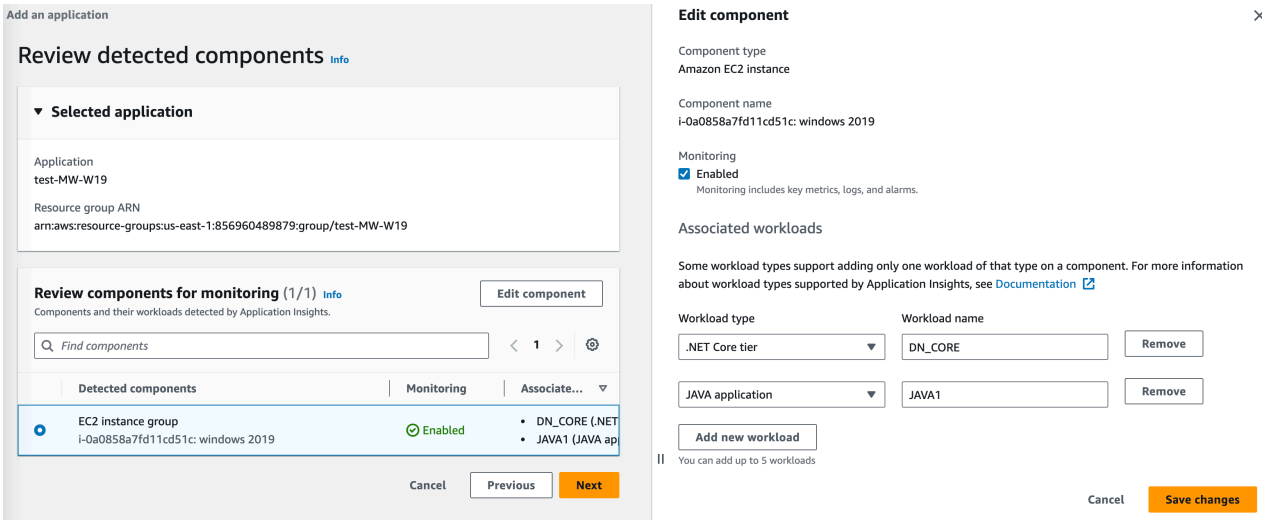
- 워크로드를 탐지할 수 없음 - 워크로드를 탐지하려고 할 때 문제가 발생했습니다. 필수 조건을 완료했는지 확인합니다. 워크로드를 추가해야 하는 경우 구성 요소 편집을 선택합니다.

- 워크로드가 탐지되지 않음 - 워크로드가 탐지되지 않았습니다. 워크로드를 연결해야 할 수도 있습니다. 이렇게 하려면 구성 요소 편집을 선택합니다.
- 해당 없음 - 구성 요소는 사용자 지정된 워크로드를 지원하지 않으며 기본 지표, 경보, 로그를 사용하여 모니터링됩니다. 이러한 구성 요소에는 워크로드를 추가할 수 없습니다.

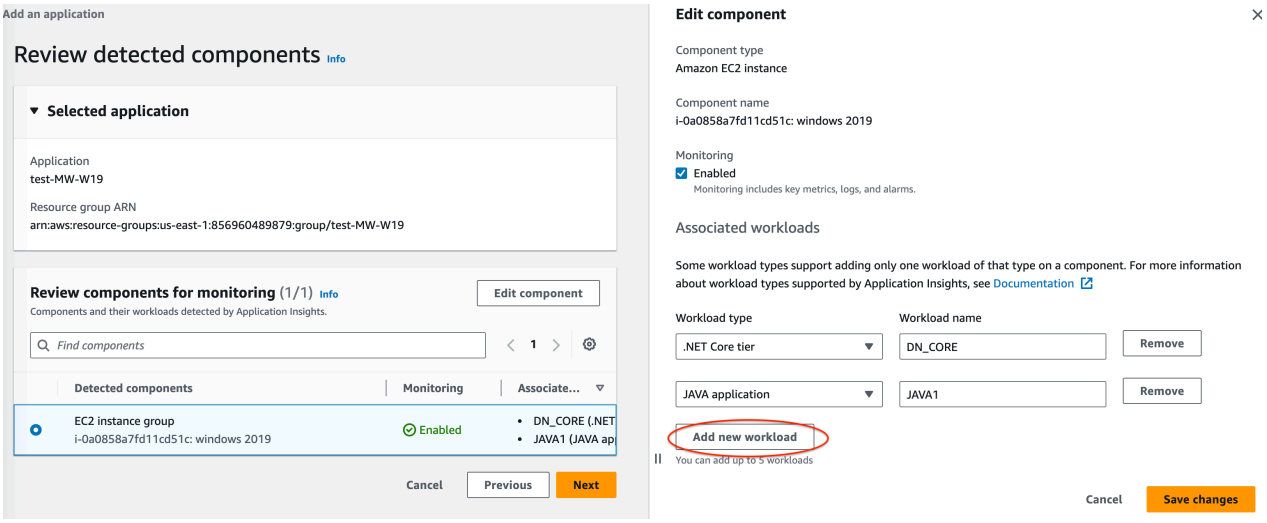
7. 구성 요소를 편집하려면 구성 요소를 선택한 다음 구성 요소 편집을 선택합니다. 구성 요소에서 탐지된 워크로드가 포함된 측면 패널이 열립니다. 이 패널에서는 구성 요소 세부 정보를 편집하고 새 워크로드를 추가할 수 있습니다.



- 워크로드 유형이나 이름을 편집하려면 드롭다운 목록을 사용하세요.

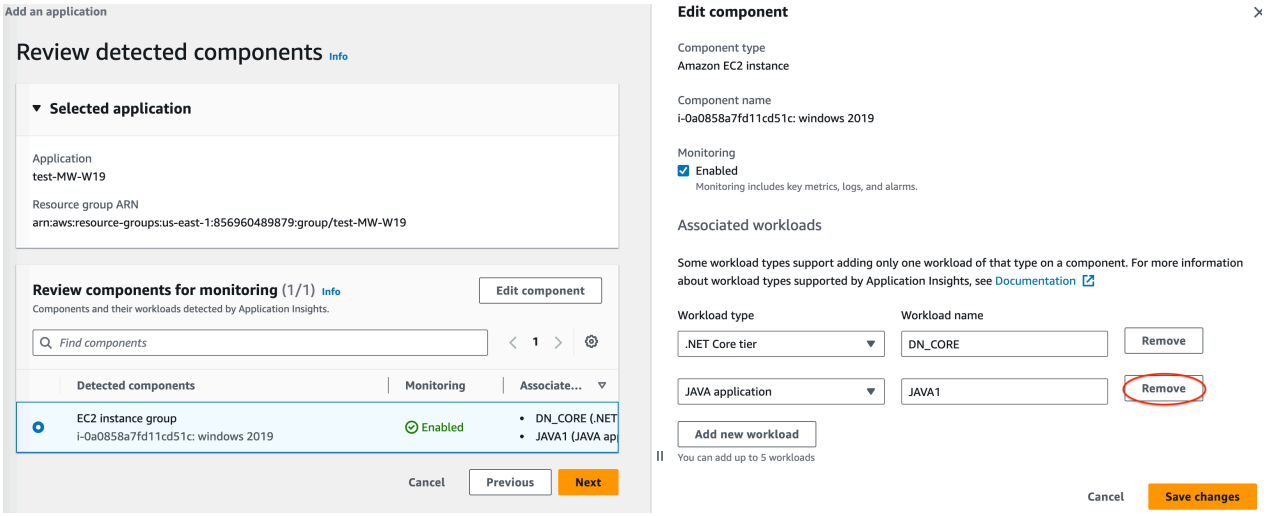


- 구성 요소에 워크로드를 추가하려면 새 워크로드 추가를 선택합니다.

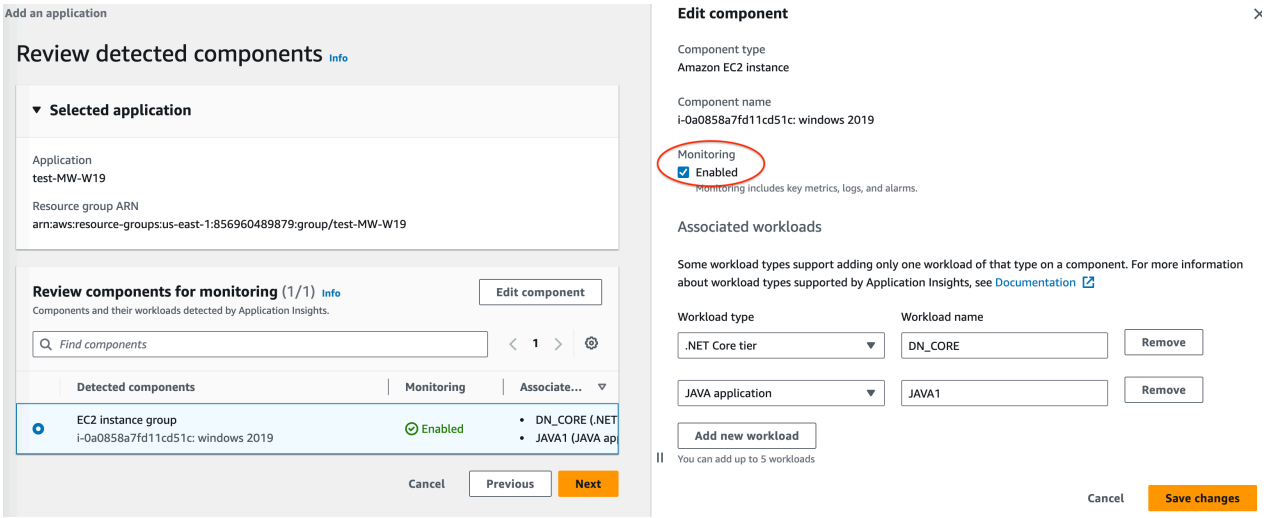


- 새 워크로드 추가가 표시되지 않는 경우 여러 워크로드를 지원하지 않는 구성 요소입니다.
- 관련 워크로드 제목이 표시되지 않으면 사용자 지정된 워크로드를 지원하지 않는 구성 요소입니다.

- 워크로드를 제거하려면 모니터링에서 제거하려는 워크로드 옆에 있는 제거를 선택합니다.



- 전체 구성 요소에 대한 모니터링을 사용하지 않도록 설정하려면 모니터링 확인란의 선택을 취소합니다.



- 구성 요소 편집을 완료하면 오른쪽 하단에서 변경 사항 저장을 선택합니다. 구성 요소의 워크로드 변경 사항은 관련 워크로드 아래의 모니터링을 위한 구성 요소 검토 테이블에서 확인할 수 있습니다.

8. 탐지된 구성 요소 검토 페이지에서 다음을 선택합니다.
9. 구성 요소 세부 정보 지정 페이지에는 이전 단계에서 관련 워크로드를 사용자 지정할 수 있는 모든 구성 요소가 포함되어 있습니다.

> **Note**
>
> 구성 요소 헤더에 선택적 태그가 있는 경우 해당 구성 요소의 워크로드에 대한 추가 세부 정보는 선택 사항입니다.

   이 페이지에 구성 요소가 표시되지 않으면 이 단계에서 지정할 수 있는 추가 세부 정보가 없는 구성 요소입니다.

10. 다음을 선택합니다.
11. 검토 및 제출 페이지에서 모니터링되는 모든 구성 요소 및 워크로드 세부 정보를 검토하세요.
12. 제출을 선택합니다.

Account-based application monitoring

1. 애플리케이션 이름. 계정 기반 애플리케이션의 이름을 입력합니다.

2. 새로운 리소스에 대한 자동 모니터링. 기본적으로 Application Insights는 권장 설정을 사용하여 애플리케이션을 온보딩한 후 계정에 추가되는 리소스 구성 요소에 대한 모니터링을 구성합니다. 확인란을 선택 취소하여 애플리케이션을 온보딩한 후 추가된 리소스에 대한 모니터링을 제외할 수 있습니다.
3. CloudWatch Events 모니터링. Amazon EBS, Amazon EC2, AWS CodeDeploy, Amazon ECS, AWS Health API 및 알림, Amazon RDS, Amazon S3, AWS Step Functions로부터 인사이트를 얻기 위해 Application Insights 모니터링을 CloudWatch Events와 통합하려면 확인란을 선택합니다.
4. AWS Systems Manager OpsCenter와 통합. 선택한 애플리케이션에 문제가 감지될 때 이를 확인하고 알림을 받으려면 수정 조치를 위해 Systems Manager OpsCenter OpsItems 생성(Generate Systems Manager OpsCenter OpsItems for remedial actions) 확인란을 선택합니다. AWS 리소스와 관련된 운영 작업 항목(OpsItem)을 해결하기 위해 수행된 작업을 추적하려면 SNS 주제 ARN을 제공합니다.
5. 태그(선택 사항). CloudWatch Application Insights는 태그 기반 및 CloudFormation 기반 리소스 그룹을 모두 지원합니다(Auto Scaling 그룹 제외). 자세한 내용은 Tag Editor 작업을 참조하세요.
6. 리소스 검색. 계정에서 검색된 모든 리소스가 이 목록에 추가됩니다. Application Insights에서 계정의 모든 리소스를 검색할 수 없는 경우 페이지 상단에 오류 메시지가 표시됩니다. 이 메시지에는 필수 권한을 추가하는 방법에 대한 설명서 링크가 있습니다.
7. 다음을 선택합니다.

   애플리케이션에 대한 ARN은 다음 형식으로 생성됩니다.

   ```
   arn:partition:applicationinsights:region:account-id:application/
   TBD/application-name
   ```

   예

   ```
   arn:aws:applicationinsights:us-east-1:123456789012:application/TBD/my-
   application
   ```

4. 애플리케이션 모니터링 구성을 제출하면 Application summary(애플리케이션 요약), Monitored components(모니터링되는 구성 요소)와 Unmonitored components(모니터링되지 않는 구성 요소) 목록을 볼 수 있으며, Components(구성 요소) 옆에 있는 탭을 선택하여 Configuration history(구성 기록), Log patterns(로그 패턴), 적용한 Tags(태그)를 볼 수 있는 애플리케이션 세부 정보 페이지로 이동합니다.

애플리케이션에 대한 통찰력을 보려면 인사이트 보기(View Insights)를 선택합니다.

편집(Edit)을 선택해 CloudWatch Events 모니터링 및 AWS Systems Manager OpsCenter 통합에 대한 선택 사항을 업데이트할 수 있습니다.

구성 요소(Components)에서 작업(Actions) 메뉴를 선택해 인스턴스 그룹을 생성하거나, 수정하거나, 그룹 해제할 수 있습니다.

구성 요소 옆에 있는 글머리 기호를 선택하고 모니터링 관리(Manage monitoring)를 선택해 애플리케이션 티어, 로그 그룹, 이벤트 로그, 지표, 사용자 지정 경보 등의 구성 요소에 대한 모니터링을 관리할 수 있습니다.

Amazon ECS 및 Amazon EKS 리소스 모니터링을 위한 Application Insights 활성화

Application Insights를 사용 설정해 Container Insights 콘솔의 컨테이너화된 애플리케이션 및 마이크로서비스를 모니터링할 수 있습니다. Application Insights는 다음 리소스에 대한 모니터링을 지원합니다.

- Amazon ECS 클러스터
- Amazon ECS 서비스
- Amazon ECS 작업
- Amazon EKS 클러스터

Application Insights가 사용 설정되면 컨테이너화된 애플리케이션 및 마이크로서비스에 대한 권장 지표 및 로그를 제공하고, 잠재적인 문제를 감지하고, CloudWatch Events를 생성하고, 자동 대시보드를 생성합니다.

Container Insights나 Application Insights 콘솔에서 컨테이너화된 리소스에 대해 Application Insights를 사용 설정할 수 있습니다.

Container Insights 콘솔에서 Application Insights 활성화

Container Insights 콘솔의 Container Insights 성능 모니터링(Performance monitoring) 대시보드에서 애플리케이션 인사이트 자동 구성(Auto-configure Application Insights)을 선택합니다. 애플리케이션 인사이트를 활성화하면 감지된 문제에 대한 세부 정보가 표시됩니다.

Application Insights 콘솔에서 Application Insights 활성화

구성 요소 목록에 ECS 클러스터가 나타나면 Application Insights에서 Container Insights를 사용하여 추가 컨테이너 모니터링을 자동으로 활성화합니다.

EKS 클러스터의 경우 Container Insights로 추가 모니터링을 활성화하여 컨테이너 재시작 실패와 같은 진단 정보를 제공하고 문제를 격리 및 해결할 수 있도록 돕습니다. EKS용 Container Insights를 설정하려면 추가 단계가 필요합니다. EKS에서 Container Insights를 설정하는 단계에 대한 자세한 정보는 Amazon EKS 및 Kubernetes에서 Container Insights 설정 섹션을 참조하세요.

Container Insights를 사용한 EKS에 대한 추가 모니터링은 EKS를 사용하는 Linux 인스턴스에서 지원됩니다

ECS 및 EKS 클러스터용 Container Insights 지원에 대한 자세한 내용은 Container Insights 섹션을 참조하세요.

애플리케이션 구성 요소에 대한 모니터링 사용 중지

애플리케이션 구성 요소에 대한 모니터링을 비활성화하려면 애플리케이션 세부 정보 페이지에서 모니터링을 비활성화하고 싶은 구성 요소를 선택합니다. 작업(Actions)을 선택한 다음 모니터링에서 제거(Remove from monitoring)를 선택합니다.

애플리케이션을 삭제합니다

애플리케이션을 삭제하려면 CloudWatch 대시보드에 있는 왼쪽 탐색 창의 [인사이트(Insights)]에서 [Application Insights]를 선택합니다. 삭제하려는 애플리케이션을 선택합니다. 작업(Actions)에서 애플리케이션 삭제(Delete application)를 선택합니다. 그러면 모니터링이 삭제되고 애플리케이션 구성 요소에 대해 저장된 모니터가 모두 삭제됩니다. 애플리케이션 리소스는 삭제되지 않습니다.

명령줄을 사용하여 모니터링할 애플리케이션 설정, 구성, 관리

이 섹션에서는 AWS CLI 및 AWS Tools for Windows PowerShell을 사용하여 모니터링할 애플리케이션을 설정하고, 구성하고, 관리하는 방법을 단계별로 안내합니다.

명령줄 절차

- 애플리케이션 추가 및 관리
- 모니터링 관리 및 업데이트
- SQL Always On 가용성 그룹에 대한 모니터링 구성
- MySQL RDS에 대한 모니터링 구성
- MySQL EC2에 대한 모니터링 구성
- PostgreSQL RDS에 대한 모니터링 구성
- PostgreSQL EC2에 대한 모니터링 구성
- Oracle RDS에 대한 모니터링 구성

- Oracle EC2에 대한 모니터링 구성

애플리케이션 추가 및 관리

명령줄을 사용해 Application Insights 애플리케이션에 대한 정보를 얻고 이 애플리케이션을 추가, 관리 및 구성할 수 있습니다.

주제

- 애플리케이션 추가
- 애플리케이션 설명
- 애플리케이션의 구성 요소 나열
- 구성 요소 설명
- 유사한 리소스를 사용자 지정 구성 요소로 그룹화
- 사용자 지정 구성 요소 그룹화 해제
- 애플리케이션 업데이트
- 사용자 지정 구성 요소 업데이트

애플리케이션 추가

AWS CLI를 사용하여 애플리케이션 추가

AWS CLI를 사용하여 `my-resource-group`이라는 리소스 그룹(생성된 OpsItem을 SNS 주제 ARN인 `arn:aws:sns:us-east-1:123456789012:MyTopic`에 전달할 수 있도록 OpsCenter가 사용 설정된)의 애플리케이션을 추가하려면 다음 명령을 사용합니다.

```
aws application-insights create-application --resource-group-name my-resource-group --
ops-center-enabled --ops-item-sns-topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic
```

AWS Tools for Windows PowerShell을 사용하여 애플리케이션 추가

AWS Tools for Windows PowerShell을 사용하여 `my-resource-group`이라는 리소스 그룹(생성된 OpsItem을 SNS 주제 ARN인 `arn:aws:sns:us-east-1:123456789012:MyTopic`에 전달할 수 있도록 OpsCenter가 사용 설정된)의 애플리케이션을 추가하려면 다음 명령을 사용합니다.

```
New-CWAIApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -
OpsItemSNSTopicArn arn:aws:sns:us-east-1:123456789012:MyTopic
```

애플리케이션 설명

AWS CLI를 사용하여 애플리케이션 설명

AWS CLI에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션을 설명할 수 있습니다.

```
aws application-insights describe-application --resource-group-name my-resource-group
```

AWS Tools for Windows PowerShell을 사용하여 애플리케이션 설명

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션을 설명할 수 있습니다.

```
Get-CWAIApplication -ResourceGroupName my-resource-group
```

애플리케이션의 구성 요소 나열

AWS CLI를 사용하여 애플리케이션의 구성 요소 나열

AWS CLI에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 구성 요소를 나열할 수 있습니다.

```
aws application-insights list-components --resource-group-name my-resource-group
```

AWS Tools for Windows PowerShell을 사용하여 애플리케이션의 구성 요소 나열

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 구성 요소를 나열할 수 있습니다.

```
Get-CWAIComponentList -ResourceGroupName my-resource-group
```

구성 요소 설명

AWS CLI를 사용하여 구성 요소 설명

다음 AWS CLI 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션에 속해 있는 my-component라는 구성 요소를 설명할 수 있습니다.

```
aws application-insights describe-component --resource-group-name my-resource-group --
component-name my-component
```

AWS Tools for Windows PowerShell을 사용하여 구성 요소 설명

다음 AWS Tools for Windows PowerShell 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션에 속해 있는 my-component라는 구성 요소를 설명할 수 있습니다.

```
Get-CWAIComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

유사한 리소스를 사용자 지정 구성 요소로 그룹화

.NET 웹 서버 인스턴스 등 유사한 리소스를 사용자 지정 구성 요소로 그룹화하여 보다 쉽게 온보딩을 수행하고 모니터링 및 통찰력을 향상시킬 수 있습니다. 현재 CloudWatch Application Insights는 EC2 인스턴스의 사용자 지정 그룹을 지원합니다.

AWS CLI를 사용하여 리소스를 사용자 지정 구성 요소로 그룹화하려면

AWS CLI를 사용하여 세 개의 인스턴스(arn:aws:ec2:us-east-1:123456789012:instance/i-11111, arn:aws:ec2:us-east-1:123456789012:instance/i-22222, arn:aws:ec2:us-east-1:123456789012:instance/i-33333)를 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션의 my-component라는 사용자 지정 구성 요소로 함께 그룹화하려면 다음 명령을 사용합니다.

```
aws application-insights create-component --resource-group-name my-
resource-group --component-name my-component --resource-list arn:aws:ec2:us-
east-1:123456789012:instance/i-11111 arn:aws:ec2:us-east-1:123456789012:instance/
i-22222 arn:aws:ec2:us-east-1:123456789012:instance/i-33333
```

AWS Tools for Windows PowerShell을 사용하여 리소스를 사용자 지정 구성 요소로 그룹화하려면

AWS Tools for Windows PowerShell을 사용하여 세 개의 인스턴스(arn:aws:ec2:us-east-1:123456789012:instance/i-11111, arn:aws:ec2:us-east-1:123456789012:instance/i-22222, arn:aws:ec2:us-east-1:123456789012:instance/i-33333)를 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션의 my-component라는 사용자 지정 구성 요소로 함께 그룹화하려면 다음 명령을 사용합니다.

```
New-CWAIComponent -ResourceGroupName my-resource-group -ComponentName my-component
 -ResourceList arn:aws:ec2:us-east-1:123456789012:instance/i-11111,arn:aws:ec2:us-
```

```
east-1:123456789012:instance/i-22222,arn:aws:ec2:us-east-1:123456789012:instance/
i-33333
```

사용자 지정 구성 요소 그룹화 해제

AWS CLI를 사용하여 사용자 지정 구성 요소의 그룹화를 해제하려면

AWS CLI에서 다음 명령을 사용하면 리소스 그룹 my-resource-group에 생성된 애플리케이션의 my-component라는 사용자 지정 구성 요소의 그룹화를 해제할 수 있습니다.

```
aws application-insights delete-component --resource-group-name my-resource-group --
component-name my-new-component
```

AWS Tools for Windows PowerShell을 사용하여 사용자 지정 구성 요소의 그룹화를 해제하려면

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 리소스 그룹 my-resource-group에 생성된 애플리케이션의 my-component라는 사용자 지정 구성 요소의 그룹화를 해제할 수 있습니다.

```
Remove-CWAIComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

애플리케이션 업데이트

AWS CLI를 사용하여 애플리케이션 업데이트

AWS CLI에서 다음 명령을 사용하면 애플리케이션을 업데이트하여 애플리케이션과 관련하여 감지된 문제의 AWS Systems Manager OpsCenter OpsItem을 생성하고 생성된 OpsItem을 SNS 주제인 arn:aws:sns:us-east-1:123456789012:MyTopic에 연결할 수 있습니다.

```
aws application-insights update-application --resource-group-name my-resource-group --
ops-center-enabled --ops-item-sns-topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic
```

AWS Tools for Windows PowerShell을 사용하여 애플리케이션 업데이트

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 애플리케이션을 업데이트하여 애플리케이션과 관련하여 감지된 문제의 AWS SSM OpsCenter OpsItem을 생성하고 생성된 OpsItem을 SNS 주제인 arn:aws:sns:us-east-1:123456789012:MyTopic 에 연결할 수 있습니다.

```
Update-CWAIApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -
OpsItemSNSTopicArn arn:aws:sns:us-east-1:123456789012:MyTopic
```

사용자 지정 구성 요소 업데이트

AWS CLI를 사용하여 사용자 지정 구성 요소 업데이트

AWS CLI에서 다음 명령을 사용하면 my-component라는 사용자 지정 구성 요소를 새로운 구성 요소 이름인 my-new-component 및 업데이트된 인스턴스 그룹으로 업데이트할 수 있습니다.

```
aws application-insights update-component --resource-group-name my-resource-
group --component-name my-component --new-component-name my-new-component --
resource-list arn:aws:ec2:us-east-1:123456789012:instance/i-44444 arn:aws:ec2:us-
east-1:123456789012:instance/i-55555
```

AWS Tools for Windows PowerShell을 사용하여 사용자 지정 구성 요소 업데이트

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 my-component라는 사용자 지정 구성 요소를 새로운 구성 요소 이름인 my-new-component 및 업데이트된 인스턴스 그룹으로 업데이트할 수 있습니다.

```
Update-CWAIComponent -ComponentName my-component -NewComponentName my-new-
component -ResourceGroupName my-resource-group -ResourceList arn:aws:ec2:us-
east-1:123456789012:instance/i-44444,arn:aws:ec2:us-east-1:123456789012:instance/
i-55555
```

모니터링 관리 및 업데이트

명령줄을 사용하여 Application Insights 애플리케이션에 대한 모니터링을 관리하고 업데이트할 수 있습니다.

주제

- 애플리케이션에서 발생하는 문제 나열
- 애플리케이션 문제 설명
- 문제와 관련된 이상 또는 오류 설명
- 애플리케이션의 이상 또는 오류 설명
- 구성 요소의 모니터링 구성 설명
- 구성 요소의 권장 모니터링 구성 설명
- 구성 요소의 모니터링 구성 업데이트
- Application Insights 모니터링에서 지정된 리소스 그룹 제거

애플리케이션에서 발생하는 문제 나열

AWS CLI를 사용하여 애플리케이션 관련 문제 나열

AWS CLI에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션의 Unix Epoch 이후 1,000~10,000밀리초 사이에 감지된 애플리케이션 관련 문제를 나열할 수 있습니다.

```
aws application-insights list-problems --resource-group-name my-resource-group --start-time 1000 --end-time 10000
```

AWS Tools for Windows PowerShell을 사용하여 애플리케이션 관련 문제 나열

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션의 Unix Epoch 이후 1,000~10,000밀리초 사이에 감지된 애플리케이션 관련 문제를 나열할 수 있습니다.

```
$startDate = "8/6/2019 3:33:00"
$endDate = "8/6/2019 3:34:00"
Get-CWAIProblemList -ResourceGroupName my-resource-group -StartTime $startDate -EndTime $endDate
```

애플리케이션 문제 설명

AWS CLI를 사용하여 애플리케이션 문제 설명

AWS CLI에서 다음 명령을 사용하면 문제 ID가 p-1234567890인 문제를 설명할 수 있습니다.

```
aws application-insights describe-problem –problem-id p-1234567890
```

AWS Tools for Windows PowerShell을 사용하여 애플리케이션 문제 설명

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 문제 ID가 p-1234567890인 문제를 설명할 수 있습니다.

```
Get-CWAIProblem -ProblemId p-1234567890
```

문제와 관련된 이상 또는 오류 설명

AWS CLI를 사용하여 문제와 연결된 이상 또는 오류 설명

AWS CLI에서 다음 명령을 사용하면 문제 ID가 p-1234567890인 문제와 연결된 이상 또는 오류를 설명할 수 있습니다.

```
aws application-insights describe-problem-observations --problem-id -1234567890
```

AWS Tools for Windows PowerShell을 사용하여 문제와 연결된 이상 또는 오류 설명

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 문제 ID가 p-1234567890인 문제와 연결된 이상 또는 오류를 설명할 수 있습니다.

```
Get-CWAIProblemObservation -ProblemId p-1234567890
```

애플리케이션의 이상 또는 오류 설명

AWS CLI를 사용하여 애플리케이션의 이상 또는 오류 설명

AWS CLI에서 다음 명령을 사용하면 관찰 ID가 o-1234567890인 애플리케이션의 이상 또는 오류를 설명할 수 있습니다.

```
aws application-insights describe-observation –observation-id o-1234567890
```

AWS Tools for Windows PowerShell을 사용하여 애플리케이션의 이상 또는 오류 설명

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 관찰 ID가 o-1234567890인 애플리케이션의 이상 또는 오류를 설명할 수 있습니다.

```
Get-CWAIObservation -ObservationId o-1234567890
```

구성 요소의 모니터링 구성 설명

AWS CLI를 사용하여 구성 요소의 모니터링 구성 설명

AWS CLI에서 다음 명령을 사용하면 리소스 그룹 my-resource-group에 생성된 애플리케이션에 있는 my-component라는 구성 요소의 모니터링 구성을 설명할 수 있습니다.

```
aws application-insights describe-component-configuration –resource-group-name my-
resource-group –component-name my-component
```

AWS Tools for Windows PowerShell을 사용하여 구성 요소의 모니터링 구성 설명

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 리소스 그룹 my-resource-group에 생성된 애플리케이션에 있는 my-component라는 구성 요소의 모니터링 구성을 설명할 수 있습니다.

```
Get-CWAIComponentConfiguration -ComponentName my-component -ResourceGroupName my-
resource-group
```

구성 요소 구성에 관한 자세한 내용과 예시 JSON 파일은 구성 요소 구성 작업 단원을 참조하세요.

구성 요소의 권장 모니터링 구성 설명

AWS CLI를 사용하여 구성 요소의 권장 모니터링 구성 설명

구성 요소가 .NET Worker 애플리케이션의 일부인 경우 AWS CLI에서 다음 명령을 사용하면 리소스 그룹 my-resource-group에 생성된 애플리케이션에 있는 my-component라는 구성 요소의 권장 모니터링 구성을 설명할 수 있습니다.

```
aws application-insights describe-component-configuration-recommendation --resource-
group-name my-resource-group --component-name my-component --tier DOT_NET_WORKER
```

AWS Tools for Windows PowerShell을 사용하여 구성 요소의 권장 모니터링 구성 설명

구성 요소가 .NET Worker 애플리케이션의 일부인 경우 AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 리소스 그룹 my-resource-group에 생성된 애플리케이션에 있는 my-component라는 구성 요소의 권장 모니터링 구성을 설명할 수 있습니다.

```
Get-CWAIComponentConfigurationRecommendation -ComponentName my-component -
ResourceGroupName my-resource-group -Tier DOT_NET_WORKER
```

구성 요소 구성에 관한 자세한 내용과 예시 JSON 파일은 구성 요소 구성 작업 단원을 참조하세요.

구성 요소의 모니터링 구성 업데이트

AWS CLI를 사용하여 구성 요소의 모니터링 구성 업데이트

AWS CLI에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션의 my-component라는 구성 요소를 업데이트할 수 있습니다. 이 명령은 다음과 같은 작업을 포함합니다.

1. 구성 요소에 대한 모니터링을 활성화합니다.

2. 구성 요소의 티어를 .NET Worker로 설정합니다.
3. configuration.txt라는 로컬 파일을 읽도록 구성 요소의 JSON 구성을 업데이트합니다.

```
aws application-insights update-component-configuration --resource-group-name my-
resource-group --component-name my-component --tier DOT_NET_WORKER --monitor --
component-configuration "file://configuration.txt"
```

AWS Tools for Windows PowerShell을 사용하여 구성 요소의 모니터링 구성 업데이트

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션의 my-component라는 구성 요소를 업데이트할 수 있습니다. 이 명령은 다음과 같은 작업을 포함합니다.

1. 구성 요소에 대한 모니터링을 활성화합니다.
2. 구성 요소의 티어를 .NET Worker로 설정합니다.
3. configuration.txt라는 로컬 파일을 읽도록 구성 요소의 JSON 구성을 업데이트합니다.

```
[string]$config = Get-Content -Path configuration.txt
Update-CWAIComponentConfiguration -ComponentName my-component -ResourceGroupName my-
resource-group -Tier DOT_NET_WORKER -Monitor 1 -ComponentConfiguration $config
```

구성 요소 구성에 관한 자세한 내용과 예시 JSON 파일은 구성 요소 구성 작업 단원을 참조하세요.

Application Insights 모니터링에서 지정된 리소스 그룹 제거

AWS CLI를 사용하여 Application Insights 모니터링에서 지정된 리소스 그룹 제거

AWS CLI에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션을 모니터링에서 제거할 수 있습니다.

```
aws application-insights delete-application --resource-group-name my-resource-group
```

AWS Tools for Windows PowerShell을 사용하여 Application Insights 모니터링에서 지정된 리소스 그룹 제거

AWS Tools for Windows PowerShell에서 다음 명령을 사용하면 my-resource-group이라는 리소스 그룹에 생성된 애플리케이션을 모니터링에서 제거할 수 있습니다.

```
Remove-CWAIApplication -ResourceGroupName my-resource-group
```

SQL Always On 가용성 그룹에 대한 모니터링 구성

1. SQL HA EC2 인스턴스를 사용하여 리소스 그룹에 대한 애플리케이션을 생성합니다.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 새 애플리케이션 구성 요소를 생성하여 SQL HA 클러스터를 나타내는 EC2 인스턴스를 정의합니다.

```
aws application-insights create-component --resource-group-name
 "<RESOURCE_GROUP_NAME>" --component-name SQL_HA_CLUSTER --resource-list
  "arn:aws:ec2:<REGION>:<ACCOUNT_ID>:instance/<CLUSTER_INSTANCE_1_ID>"
 "arn:aws:ec2:<REGION>:<ACCOUNT_ID>:instance/<CLUSTER_INSTANCE_2_ID>
```

3. SQL HA 구성 요소를 구성합니다.

```
aws application-insights  update-component-configuration --resource-group-name
 "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "SQL_HA_CLUSTER" --
monitor --tier SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP --monitor  --component-
configuration '{
  "subComponents" : [ {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [ {
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
    }, {
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
```

```
    }, {
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Buffer cache hit ratio",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Page life expectancy",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics Processes blocked",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics User Connections",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Locks Number of Deadlocks/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:SQL Statistics Batch Requests/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica File Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log remaining for undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Send Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Mirrored Write Transaction/
sec",
      "monitor" : true
    }, {
```

```
      "alarmMetricName" : "SQLServer:Database Replica Recovery Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redo Bytes Remaining",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redone Bytes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Total Log requiring undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Transaction Delay",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP-
<RESOURCE_GROUP_NAME>",
      "logPath" : "C:\\Program Files\\Microsoft SQL Server\\MSSQL**.MSSQLSERVER\
\MSSQL\\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
```

```
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
}'
```

> **Note**
>
> Application Insights는 장애 조치와 같은 클러스터 활동을 감지하기 위해 애플리케이션 이벤트 로그(정보 수준)를 수집해야 합니다.

MySQL RDS에 대한 모니터링 구성

1. RDS MySQL 데이터베이스 인스턴스를 사용하여 리소스 그룹에 대한 애플리케이션을 생성합니다.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 오류 로그는 기본적으로 활성화됩니다. 느린 쿼리 로그는 데이터 파라미터 그룹을 사용하여 활성화할 수 있습니다. 자세한 내용은 MySQL 느린 쿼리 및 일반 로그 액세스를 참조하세요.
   - `set slow_query_log = 1`
   - `set log_output = FILE`

3. 모니터링할 로그를 CloudWatch Logs로 내보냅니다. 자세한 내용은 CloudWatch Logs에 MySQL Logs 게시를 참조하세요.
4. MySQL RDS 구성 요소를 구성합니다.

```
aws application-insights  update-component-configuration --resource-group-name
 "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
 --monitor --tier DEFAULT --monitor  --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\":\"CPUUtilization\",\"monitor\":true}],\"logs\":[{\"logType\":
\"MYSQL\",\"monitor\":true},{\"logType\": \"MYSQL_SLOW_QUERY\",\"monitor\":false}]}"
```

MySQL EC2에 대한 모니터링 구성

1. SQL HA EC2 인스턴스를 사용하여 리소스 그룹에 대한 애플리케이션을 생성합니다.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 오류 로그는 기본적으로 활성화됩니다. 느린 쿼리 로그는 데이터 파라미터 그룹을 사용하여 활성화할 수 있습니다. 자세한 내용은 MySQL 느린 쿼리 및 일반 로그 액세스를 참조하세요.
   - `set slow_query_log = 1`
   - `set log_output = FILE`
3. MySQL EC2 구성 요소를 구성합니다.

```
aws application-insights  update-component-configuration --resource-group-name
 "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
 --monitor --tier MYSQL --monitor  --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\":\"CPUUtilization\",\"monitor\":true}],\"logs\":[{\"logGroupName
\":\"<UNIQUE_LOG_GROUP_NAME>\",\"logPath\":\"C:\\\\ProgramData\\\\MySQL\\\\MySQL
 Server **\\\\Data\\\\<FILE_NAME>.err\",\"logType\":\"MYSQL\",\"monitor\":true,
\"encoding\":\"utf-8\"}]}"
```

PostgreSQL RDS에 대한 모니터링 구성

1. PostgreSQL RDS 데이터베이스 인스턴스를 사용하여 리소스 그룹의 애플리케이션을 생성합니다.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. CloudWatch에 PostgreSQL 로그를 게시하는 기능은 기본적으로 사용 설정되어 있지 않습니다. 모니터링을 사용 설정하려면 RDS 콘솔을 열고 모니터링할 데이터베이스를 선택합니다. 오른쪽 상단 모서리에서 수정(Modify)을 선택하고 PostgreSQL 로그로 레이블이 지정된 확인란을 선택합니다. [계속(Continue)]을 선택하여 이 설정을 저장합니다.
3. PostgreSQL 로그를 CloudWatch로 내보냅니다.
4. PostgreSQL RDS 구성 요소를 구성합니다.

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier DEFAULT --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\": \"CPUUtilization\",
         \"monitor\": true
      }
   ],
   \"logs\":[
      {
         \"logType\": \"POSTGRESQL\",
         \"monitor\": true
      }
   ]
}"
```

PostgreSQL EC2에 대한 모니터링 구성

1. PostgreSQL EC2 인스턴스를 사용하여 리소스 그룹의 애플리케이션을 생성합니다.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. PostgreSQL EC2 구성 요소를 구성합니다.

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier POSTGRESQL --component-configuration
"{
   \"alarmMetrics\":[
      {
```

```
         \"alarmMetricName\":\"CPUUtilization\",
         \"monitor\":true
      }
   ],
   \"logs\":[
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/var/lib/pgsql/data/log/\",
         \"logType\":\"POSTGRESQL\",
         \"monitor\":true,
         \"encoding\":\"utf-8\"
      }
   ]
}"
```

Oracle RDS에 대한 모니터링 구성

1. Oracle RDS 데이터베이스 인스턴스를 사용하여 리소스 그룹의 애플리케이션을 생성합니다.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. CloudWatch에 Oracle 로그를 게시하는 기능은 기본적으로 사용 설정되어 있지 않습니다. 모니터링을 사용 설정하려면 RDS 콘솔을 열고 모니터링할 데이터베이스를 선택합니다. 오른쪽 상단 모서리에서 수정(Modify)을 선택하고 알림(Alert) 로그 및 리스너(Listener) 로그로 레이블이 지정된 확인란을 선택합니다. [계속(Continue)]을 선택하여 이 설정을 저장합니다.
3. Oracle 로그를 CloudWatch로 내보냅니다.
4. Oracle RDS 구성 요소를 구성합니다.

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier DEFAULT --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\": \"CPUUtilization\",
         \"monitor\": true
      }
   ],
   \"logs\":[
      {
```

```
         \"logType\": \"ORACLE_ALERT\",
         \"monitor\": true
      },
      {
         \"logType\": \"ORACLE_LISTENER\",
         \"monitor\": true
      }
   ]
}"
```

Oracle EC2에 대한 모니터링 구성

1. Oracle EC2 인스턴스를 사용하여 리소스 그룹의 애플리케이션을 생성합니다.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. Oracle EC2 구성 요소를 구성합니다.

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier ORACLE --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\":\"CPUUtilization\",
         \"monitor\":true
      }
   ],
   \"logs\":[
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/opt/oracle/diag/rdbms/*/*/trace\",
         \"logType\":\"ORACLE_ALERT\",
         \"monitor\":true,
      },
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/opt/oracle/diag/tnslsnr/$HOSTNAME/listener/trace/\",
         \"logType\":\"ORACLE_ALERT\",
         \"monitor\":true,
      }
```

```
    ]
}"
```

Application Insights CloudWatch Events 및 감지된 문제에 대한 알림

CloudWatch Application Insights에 추가된 각 애플리케이션의 경우 다음 이벤트에 대해 CloudWatch 이벤트가 최대한 게시됩니다.

- 문제 생성. CloudWatch Application Insights가 새로운 문제를 감지할 때 발생합니다.
  - 세부 정보 유형: "Application Insights 문제 감지"
  - 세부 정보:
    - `problemId`: 감지된 문제 ID
    - `region`: 문제가 생성된 AWS 리전
    - `resourceGroupName`: 문제가 감지된 등록된 애플리케이션의 리소스 그룹
    - `status`: 문제의 상태 가능한 상태 및 정의는 다음과 같습니다.
      - `In progress`: 새로운 문제가 확인되었습니다. 이 문제는 여전히 관찰을 받고 있습니다.
      - `Recovering`: 문제가 안정화되고 있습니다. 문제가 이 상태일 때 수동으로 문제를 해결할 수 있습니다.
      - `Resolved`: 문제가 해결되었습니다. 이 문제에 대한 새로운 관찰은 없습니다.
      - `Recurring`: 문제가 지난 24시간 내에 해결되었습니다. 추가 관찰 결과 다시 열렸습니다.
    - `severity`: 문제의 심각도
    - `problemUrl`: 문제의 콘솔 URL

- 문제 업데이트. 새로운 관찰로 문제가 업데이트되거나 기존의 관찰이 업데이트되어 문제가 업데이트될 때 발생합니다. 업데이트에는 문제의 해결 또는 종료가 포함됩니다.
  - 세부 정보 유형: "Application Insights 문제 업데이트"
  - 세부 정보:
    - `problemId`: 생성된 문제 ID
    - `region`: 문제가 생성된 AWS 리전
    - `resourceGroupName`: 문제가 감지된 등록된 애플리케이션의 리소스 그룹
    - `status`: 문제의 상태
    - `severity`: 문제의 심각도

- `problemUrl`: 문제의 콘솔 URL

애플리케이션에서 생성된 문제 이벤트에 대한 알림 수신 방법

CloudWatch 콘솔에서 왼쪽 탐색 창의 [이벤트(Events)] 아래에서 [규칙(Rules)]을 선택합니다. 규칙 페이지에서 규칙 생성을 선택합니다. [서비스 이름(Service Name)] 드롭다운 목록에서 [Amazon CloudWatch Application Insights]를 선택하고 [이벤트 유형(Event Type)]을 선택합니다. 그런 다음 Add target(대상 추가)을 선택하고 대상 및 파라미터(예: SNS 주제 또는 Lambda 함수)를 선택합니다.

AWS Systems Manager를 통한 작업. CloudWatch Application Insights는 Systems Manager OpsCenter와의 기본 통합 기능을 제공합니다. 애플리케이션에 이 통합을 사용하도록 선택하면 애플리케이션에서 감지된 모든 문제에 대한 OpsItem이 OpsCenter 콘솔에 생성됩니다. OpsCenter 콘솔에서, CloudWatch Application Insights에서 감지한 문제에 대한 요약된 정보를 보고 Systems Manager Automation 실행서를 선택하여 수정 작업을 수행하거나 애플리케이션에서 리소스 문제를 일으키는 Windows 프로세스를 추가로 식별할 수 있습니다.

Application Insights의 교차 계정 관찰성

CloudWatch Application Insights 교차 계정 관찰성을 사용하면 단일 지역 내의 여러 AWS 계정에 걸쳐 있는 애플리케이션을 모니터하고 문제를 해결할 수 있습니다.

Amazon CloudWatch Observability Access Manager를 사용하여 하나 이상의 AWS 계정을 모니터링 계정으로 설정할 수 있습니다. 모니터링 계정에 싱크를 생성하여 소스 계정의 데이터를 볼 수 있는 기능을 모니터링 계정에 제공합니다. 싱크를 사용하여 소스 계정에서 모니터링 계정으로의 링크를 만듭니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

필수 리소스

CloudWatch Application Insights 교차 계정 관찰성의 적절한 작동을 위해 CloudWatch Observability Access Manager를 통해 다음 원격 측정 유형이 공유되는지 확인하세요.

- CloudWatch Application Insights의 애플리케이션
- Amazon CloudWatch의 지표
- Amazon CloudWatch Logs의 로그 그룹
- AWS X-Ray의 추적

구성 요소 구성 작업

구성 요소 구성은 구성 요소의 구성 설정을 설명하는 JSON 형식의 텍스트 파일입니다. 이 단원에서는 템플릿 조각 예 및 구성 요소 구성 예를 제공하고 구성 요소 구성 섹션을 설명합니다.

주제

구성 요소 구성 템플릿 조각

다음 예는 JSON 형식의 템플릿 조각을 보여줍니다.

```
{
  "alarmMetrics" : [
    list of alarm metrics
  ],
  "logs" : [
    list of logs
  ],
  "processes" : [
   list of processes
  ],
  "windowsEvents" : [
    list of windows events channels configurations
  ],
  "alarms" : [
    list of CloudWatch alarms
  ],
  "jmxPrometheusExporter": {
    JMX Prometheus Exporter configuration
  },
  "hanaPrometheusExporter": {
      SAP HANA Prometheus Exporter configuration
  },
  "haClusterPrometheusExporter": {
      HA Cluster Prometheus Exporter configuration
  },
  "netWeaverPrometheusExporter": {
      SAP NetWeaver Prometheus Exporter configuration
```

```
  },
  "subComponents" : [
    {
      "subComponentType" : "AWS::EC2::Instance" ...
      component nested instances configuration
    },
    {
      "subComponentType" : "AWS::EC2::Volume" ...
      component nested volumes configuration
    }
  ]
}
```

구성 요소 구성 섹션

구성 요소 구성에는 여러 주요 섹션이 포함되어 있습니다. 구성 요소 구성의 섹션은 임의의 순서로 나열될 수 있습니다.

- alarmMetrics(선택 사항)

  구성 요소에 대해 모니터링할 지표 목록. alarmMetrics 섹션은 모든 구성 요소 유형에 포함될 수 있습니다.

- 로그(선택 사항)

  구성 요소에 대해 모니터링할 로그 목록. 로그 섹션은 EC2 인스턴스에만 있습니다.

- 프로세스(선택 사항)

  구성 요소에 대해 모니터링하기 위한 프로세스 목록입니다. 프로세스 섹션은 EC2 인스턴스에만 있습니다.

- subComponents(선택 사항)

  구성 요소의 중첩된 인스턴스 및 볼륨 subComponent 구성입니다. ELB, ASG, EC2 인스턴스, 사용자 지정 그룹화 EC2 인스턴스와 같은 유형의 구성 요소에는 중첩된 인스턴스 및 subComponents 섹션이 있을 수 있습니다.

- 경보(선택 사항)

  구성 요소에 대해 모니터링할 경보 목록입니다. 경보 섹션은 모든 구성 요소 유형에 포함될 수 있습니다.

- windowsEvents(선택 사항)

구성 요소에 대해 모니터링할 Windows 이벤트 목록입니다. EC2 인스턴스의 Windows에만 `windowsEvents` 섹션이 있습니다.

- JMXPrometheusExporter(선택 사항)

  JMXPrometheus Exporter 구성입니다.

- hanaPrometheusExporter(선택 사항)

  SAP HANA Prometheus Exporter 구성.

- haClusterPrometheusExporter(선택 사항)

  HA Cluster Prometheus Exporter 구성.

- netWeaverPrometheusExporter(선택 사항)

  SAP NetWeaver Prometheus Exporter 구성.

- sapAsePrometheusExporter(선택 사항)

  SAP ASE Prometheus Exporter 구성입니다.

다음 예는 JSON 형식의 subComponents 섹션 조각에 대한 구문을 보여 줍니다.

```
[
  {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [
      list of alarm metrics
    ],
    "logs" : [
      list of logs
    ],
    "processes": [
      list of processes
    ],
    "windowsEvents" : [
      list of windows events channels configurations
    ]
  },
  {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [
      list of alarm metrics
```

```
    ]
  }
]
```

구성 요소 구성 섹션 속성

이 단원에서는 각 구성 요소 구성 섹션의 속성을 설명합니다.

Sections

- 지표
- Log
- 프로세스
- JMX Prometheus Exporter
- HANA Prometheus Exporter
- HA Cluster Prometheus Exporter
- NetWeaver Prometheus Exporter
- SAP ASE Prometheus Exporter
- Windows 이벤트
- 경보

지표

구성 요소에 대해 모니터링할 지표를 정의합니다.

JSON

```
{
  "alarmMetricName" : "monitoredMetricName",
  "monitor" : true/false
}
```

속성

- alarmMetricName(필수 사항)

  구성 요소에 대해 모니터링할 지표의 이름. Application Insights에서 지원하는 지표에 대해서는 Amazon CloudWatch Application Insights에서 지원하는 로그 및 지표 단원을 참조하세요.
- monitor(선택 사항)

지표를 모니터링할지 여부를 나타내는 부울. 기본 값은 `true`입니다.

Log

구성 요소에 대해 모니터링할 로그를 정의합니다.

JSON

```
{
  "logGroupName" : "logGroupName",
  "logPath" : "logPath",
  "logType" : "logType",
  "encoding" : "encodingType",
  "monitor" : true/false
}
```

속성

- logGroupName(필수 사항)

  모니터링된 로그에 연결할 CloudWatch 로그 그룹 이름. 로그 그룹 이름 제약 조건에 대해서는 CreateLogGroup을 참조하세요.
- logPath(EC2 인스턴스 구성 요소에 필요, AWS Lambda 같은 CloudWatch 에이전트를 사용하지 않는 구성 요소에 불필요)

  모니터링할 로그의 경로. 로그 경로는 절대 Windows 시스템 파일 경로여야 합니다. 자세한 내용은 CloudWatch 에이전트 구성 파일: 로그 섹션을 참조하세요.
- logType(필수 사항)

  로그 유형은 Application Insights가 로그를 분석하는 로그 패턴을 결정합니다. 로그 유형은 다음 중에서 선택됩니다.
  - `SQL_SERVER`
  - `MYSQL`
  - `MYSQL_SLOW_QUERY`
  - `POSTGRESQL`
  - `ORACLE_ALERT`
  - `ORACLE_LISTENER`

  - IIS
  - APPLICATION
  - WINDOWS_EVENTS
  - WINDOWS_EVENTS_ACTIVE_DIRECTORY
  - WINDOWS_EVENTS_DNS
  - WINDOWS_EVENTS_IIS
  - WINDOWS_EVENTS_SHAREPOINT
  - SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP
  - SQL_SERVER_FAILOVER_CLUSTER_INSTANCE
  - DEFAULT
  - CUSTOM
  - STEP_FUNCTION
  - API_GATEWAY_ACCESS
  - API_GATEWAY_EXECUTION
  - SAP_HANA_LOGS
  - SAP_HANA_TRACE
  - SAP_HANA_HIGH_AVAILABILITY
  - SAP_NETWEAVER_DEV_TRACE_LOGS
  - PACEMAKER_HIGH_AVAILABILITY
- encoding(선택 사항)

  모니터링할 로그의 인코딩 유형. 지정된 인코딩은 CloudWatch 에이전트 지원 인코딩 목록에 포함되어야 합니다. 제공되지 않은 경우 CloudWatch Application Insights에서는 다음을 제외한 유형 utf-8의 기본 인코딩을 사용합니다.

  - SQL_SERVER: utf-16 인코딩
  - IIS: ascii 인코딩
- monitor(선택 사항)

  로그를 모니터링할지 여부를 나타내는 부울. 기본 값은 true입니다.

프로세스

구성 요소에 대해 모니터링하기 위한 프로세스를 정의합니다.

JSON

```
{
  "processName" : "monitoredProcessName",
  "alarmMetrics" : [
      list of alarm metrics
  ]
}
```

속성

- processName(필수)

  구성 요소에 대해 모니터링하기 위한 프로세스의 이름. 프로세스 이름에는 다음과 같은 프로세스 스템(예: `sqlservr` 또는 `sqlservr.exe`)이 포함되어서는 안 됩니다.
- alarmMetrics(필수)

  이 프로세스를 모니터링하기 위한 지표 목록입니다. CloudWatch Application Insights에서 지원하는 프로세스 지표를 보려면 Amazon Elastic Compute Cloud(EC2) 의 내용을 참조하세요.

JMX Prometheus Exporter

JMX Prometheus Exporter 설정을 정의합니다.

JSON

```
"JMXPrometheusExporter": {
  "jmxURL" : "JMX URL",
  "hostPort" : "The host and port",
  "prometheusPort" : "Target port to emit Prometheus metrics"
}
```

속성

- jmxURL(선택 사항)

  연결할 전체 JMX URL입니다.

- hostPort(선택 사항)

  원격 JMX를 통해 연결할 호스트 및 포트입니다. `jmxURL` 및 `hostPort` 중 하나만 지정할 수 있습니다.

- prometheusPort(선택 사항)

  Prometheus 지표를 보낼 대상 포트입니다. 지정하지 않으면 기본 포트인 9404가 사용됩니다.

HANA Prometheus Exporter

HANA Prometheus Exporter 설정을 정의합니다.

JSON

```
"hanaPrometheusExporter": {
    "hanaSid": "SAP HANA  SID",
    "hanaPort": "HANA database port",
    "hanaSecretName": "HANA secret name",
    "prometheusPort": "Target port to emit Prometheus metrics"
}
```

속성

- hanaSid

  SAP HANA 시스템의 3글자 SAP 시스템 ID(SID).

- hanaPort

  Exporter가 HANA 지표를 쿼리할 HANA 데이터베이스 포트.

- hanaSecretName

  HANA 모니터링 사용자 자격 증명을 저장하는 AWS Secrets Manager 보안 암호. HANA Prometheus Exporter는 이러한 자격 증명을 사용하여 데이터베이스에 연결하고 HANA 지표를 쿼리합니다.

- prometheusPort(선택 사항)

  Prometheus에서 지표를 보낼 대상 포트. 지정하지 않으면 기본 포트인 9668이 사용됩니다.

HA Cluster Prometheus Exporter

HA Prometheus Exporter 설정을 정의합니다.

JSON

```
"haClusterPrometheusExporter": {
    "prometheusPort": "Target port to emit Prometheus metrics"
}
```

속성

- prometheusPort(선택 사항)

  Prometheus에서 지표를 보낼 대상 포트. 지정하지 않으면 기본 포트인 9664가 사용됩니다.

NetWeaver Prometheus Exporter

NetWeaver Prometheus Exporter 설정을 정의합니다.

JSON

```
"netWeaverPrometheusExporter": {
    "sapSid": "SAP NetWeaver  SID",
    "instanceNumbers": [ "Array of instance Numbers of SAP NetWeaver system "],
"prometheusPort": "Target port to emit Prometheus metrics"
}
```

속성

- sapSid

  SAP NetWeaver 시스템의 3글자 SAP 시스템 ID(SID).
- instanceNumbers

  SAP NetWeaver 시스템의 인스턴스 번호 배열.

  예: `"instanceNumbers": [ "00", "01"]`
- prometheusPort(선택 사항)

  Prometheus 지표를 전송할 대상 포트입니다. 지정하지 않으면 기본 포트인 9680가 사용됩니다.

SAP ASE Prometheus Exporter

SAP ASE Prometheus Exporter 설정을 정의합니다.

JSON

```
"sapASEPrometheusExporter": {
    "sapAseSid": "SAP ASE SID",
    "sapAsePort": "SAP ASE database port",
    "sapAseSecretName": "SAP ASE secret name",
    "prometheusPort": "Target port to emit Prometheus metrics",
    "agreeToEnableASEMonitoring": true
}
```

속성

- sapAseSid

  SAP ASE 시스템의 3글자 SAP 시스템 ID(SID).
- sapAsePort

  Exporter가 ASE 지표를 쿼리할 SAP ASE 데이터베이스 포트.
- sapAseSecretName

  ASE 모니터링 사용자 자격 증명을 저장하는 AWS Secrets Manager 보안 암호. SAP ASE Prometheus Exporter는 이러한 자격 증명을 사용하여 데이터베이스에 연결하고 ASE 지표를 쿼리합니다.
- prometheusPort(선택 사항)

  Prometheus에서 지표를 보낼 대상 포트. 지정하지 않으면 기본 포트인 9399가 사용됩니다. 기본 포트를 사용하는 다른 ASE DB가 있는 경우 9499를 사용합니다.

Windows 이벤트

기록할 Windows 이벤트를 정의합니다.

JSON

```
{
  "logGroupName" : "logGroupName",
  "eventName" : "eventName",
```

```
  "eventLevels" : ["ERROR","WARNING","CRITICAL","INFORMATION","VERBOSE"],
  "monitor" : true/false
}
```

속성

- logGroupName(필수 사항)

  모니터링된 로그에 연결할 CloudWatch 로그 그룹 이름. 로그 그룹 이름 제약 조건에 대해서는 CreateLogGroup을 참조하세요.
- eventName(필수)

  기록할 Windows 이벤트 유형입니다. 이것은 Windows 이벤트 로그 채널 이름과 동일합니다. 예를 들면 System, Security, CustomEventName 등이 있습니다. 이 필드는 기록할 Windows 이벤트 유형마다 있어야 합니다.
- eventLevels(필수)

  기록할 이벤트 수준을 지정합니다. 기록할 각 수준을 지정해야 합니다. 가능한 값은 `INFORMATION`, `WARNING`, `ERROR`, `CRITICAL` 및 `VERBOSE`입니다. 이 필드는 기록할 Windows 이벤트 유형마다 있어야 합니다.
- monitor(선택 사항)

  로그를 모니터링할지 여부를 나타내는 부울. 기본 값은 `true`입니다.

경보

구성 요소에 대해 모니터링할 CloudWatch 경보를 정의합니다.

JSON

```
{
  "alarmName" : "monitoredAlarmName",
  "severity" : HIGH/MEDIUM/LOW
}
```

속성

- alarmName(필수)

  구성 요소에 대해 모니터링할 CloudWatch 경보의 이름입니다.

- severity(선택 사항)

  경보가 울릴 때의 중단 정도를 나타냅니다.

구성 요소 구성 예제

다음 예는 관련 서비스에 대한 구성 요소 구성을 JSON 형식으로 보여줍니다.

구성 요소 구성 예

- Amazon DynamoDB 테이블
- Amazon EC2 Auto Scaling(ASG)
- Amazon EKS 클러스터
- Amazon Elastic Compute Cloud(EC2) 인스턴스
- Amazon Elastic Container Service(Amazon ECS)
- Amazon ECS 서비스
- Amazon ECS 작업
- Amazon Elastic File System(Amazon EFS)
- Amazon FSx
- Amazon Relational Database Service(RDS) Aurora MySQL
- Amazon Relational Database Service(RDS) 인스턴스
- Amazon Route 53 상태 확인
- Amazon Route 53 호스팅 영역
- Amazon Route 53 Resolver 엔드포인트
- Amazon Route 53 Resolver 쿼리 로깅 구성
- Amazon S3 버킷
- Amazon Simple Queue Service(SQS)
- Amazon SNS 주제
- Amazon Virtual Private Cloud(VPC)
- Amazon VPC Network Address Translation(NAT) 게이트웨이
- API Gateway REST API 스테이지
- Application Elastic Load Balancing
- AWS Lambda 함수

- AWS Network Firewall 규칙 그룹
- AWS Network Firewall 규칙 그룹 연결
- AWS Step Functions
- 고객이 그룹화한 Amazon EC2 인스턴스
- Elastic Load Balancing
- Java
- Amazon EC2의 Kubernetes
- RDS MariaDB 및 RDS MySQL
- RDS Oracle
- RDS PostgreSQL
- Amazon EC2에 대한 SAP ASE
- Amazon EC2에 대한 SAP ASE 고가용성
- Amazon EC2에 대한 SAP HANA
- Amazon EC2에 대한 SAP HANA 고가용성
- Amazon EC2의 SAP NetWeaver
- Amazon EC2에 대한 SAP NetWeaver High Availability
- SQL Always On 가용성 그룹
- SQL 장애 조치 클러스터 인스턴스

Amazon DynamoDB 테이블

다음 예는 Amazon DynamoDB 테이블을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "SystemErrors",
      "monitor": false
    },
    {
      "alarmMetricName": "UserErrors",
      "monitor": false
    },
    {
      "alarmMetricName": "ConsumedReadCapacityUnits",
```

```
      "monitor": false
    },
    {
      "alarmMetricName": "ConsumedWriteCapacityUnits",
      "monitor": false
    },
    {
      "alarmMetricName": "ReadThrottleEvents",
      "monitor": false
    },
    {
      "alarmMetricName": "WriteThrottleEvents",
      "monitor": false
    },
    {
      "alarmMetricName": "ConditionalCheckFailedRequests",
      "monitor": false
    },
    {
      "alarmMetricName": "TransactionConflict",
      "monitor": false
    }
  ],
  "logs": []
}
```

Amazon EC2 Auto Scaling(ASG)

다음 예는 Amazon EC2 Auto Scaling(ASG)을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "alarmMetrics" : [
      {
        "alarmMetricName" : "CPUCreditBalance"
      }, {
        "alarmMetricName" : "EBSIOBalance%"
      }
    ],
    "subComponents" : [
      {
        "subComponentType" : "AWS::EC2::Instance",
        "alarmMetrics" : [
          {
            "alarmMetricName" : "CPUUtilization"
```

```
          }, {
            "alarmMetricName" : "StatusCheckFailed"
          }
        ],
        "logs" : [
          {
            "logGroupName" : "my_log_group",
            "logPath" : "C:\\LogFolder\\*",
            "logType" : "APPLICATION"
          }
        ],
        "processes" : [
          {
            "processName" : "my_process",
            "alarmMetrics" : [
              {
                  "alarmMetricName" : "procstat cpu_usage",
                  "monitor" : true
              }, {
                  "alarmMetricName" : "procstat memory_rss",
                  "monitor" : true
              }
            ]
      }
  ],
        "windowsEvents" : [
          {
            "logGroupName" : "my_log_group_2",
            "eventName" : "Application",
            "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ]
          }
        ]
      }, {
        "subComponentType" : "AWS::EC2::Volume",
        "alarmMetrics" : [
          {
            "alarmMetricName" : "VolumeQueueLength"
          }, {
            "alarmMetricName" : "BurstBalance"
          }
        ]
      }
    ],
    "alarms" : [
```

```
      {
        "alarmName" : "my_asg_alarm",
        "severity" : "LOW"
      }
    ]
  }
```

Amazon EKS 클러스터

다음 예는 Amazon EKS 클러스터를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "alarmMetrics":[
       {
          "alarmMetricName": "cluster_failed_node_count",
          "monitor":true
       },
       {
          "alarmMetricName": "node_cpu_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName": "node_cpu_utilization",
          "monitor":true
       },
       {
          "alarmMetricName": "node_filesystem_utilization",
          "monitor":true
       },
       {
          "alarmMetricName": "node_memory_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName": "node_memory_utilization",
          "monitor":true
       },
       {
          "alarmMetricName": "node_network_total_bytes",
          "monitor":true
       },
       {
          "alarmMetricName": "pod_cpu_reserved_capacity",
          "monitor":true
```

```
        },
        {
           "alarmMetricName": "pod_cpu_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_cpu_utilization_over_pod_limit",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_memory_reserved_capacity",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_memory_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_memory_utilization_over_pod_limit",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_network_rx_bytes",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_network_tx_bytes",
           "monitor":true
        }
     ],
     "logs":[
        {
           "logGroupName": "/aws/containerinsights/kubernetes/application",
           "logType":"APPLICATION",
           "monitor":true,
           "encoding":"utf-8"
        }
     ],
     "subComponents":[
        {
           "subComponentType":"AWS::EC2::Instance",
           "alarmMetrics":[
              {
                 "alarmMetricName":"CPUUtilization",
```

```
                "monitor":true
            },
            {
                "alarmMetricName":"StatusCheckFailed",
                "monitor":true
            },
            {
                "alarmMetricName":"disk_used_percent",
                "monitor":true
            },
            {
                "alarmMetricName":"mem_used_percent",
                "monitor":true
            }
        ],
        "logs":[
            {
                "logGroupName":"APPLICATION-KubernetesClusterOnEC2-IAD",
                "logPath":"",
                "logType":"APPLICATION",
                "monitor":true,
                "encoding":"utf-8"
            }
        ],
        "processes" : [
          {
              "processName" : "my_process",
              "alarmMetrics" : [
                  {
                      "alarmMetricName" : "procstat cpu_usage",
                      "monitor" : true
                  }, {
                      "alarmMetricName" : "procstat memory_rss",
                      "monitor" : true
                  }
              ]
          }
      ],
        "windowsEvents":[
            {
                "logGroupName":"my_log_group_2",
                "eventName":"Application",
                "eventLevels":[
                   "ERROR",
```

```
                "WARNING",
                "CRITICAL"
            ],
            "monitor":true
         }
      ]
   },
   {
      "subComponentType":"AWS::AutoScaling::AutoScalingGroup",
      "alarmMetrics":[
         {
            "alarmMetricName":"CPUCreditBalance",
            "monitor":true
         },
         {
            "alarmMetricName":"EBSIOBalance%",
            "monitor":true
         }
      ]
   },
   {
      "subComponentType":"AWS::EC2::Volume",
      "alarmMetrics":[
         {
            "alarmMetricName":"VolumeReadBytes",
            "monitor":true
         },
         {
            "alarmMetricName":"VolumeWriteBytes",
            "monitor":true
         },
         {
            "alarmMetricName":"VolumeReadOps",
            "monitor":true
         },
         {
            "alarmMetricName":"VolumeWriteOps",
            "monitor":true
         },
         {
            "alarmMetricName":"VolumeQueueLength",
            "monitor":true
         },
         {
```

```
                "alarmMetricName":"BurstBalance",
                "monitor":true
            }
        ]
    }
  ]
}
```

> **Note**
>
> - `AWS::EC2::Instance`, `AWS::EC2::Volume`, `AWS::AutoScaling::AutoScalingGroup`의 `subComponents` 섹션은 EC2 시작 유형에서 실행되는 Amazon EKS 클러스터에만 적용됩니다.
> - `subComponents`에 있는 `AWS::EC2::Instance`의 `windowsEvents` 섹션은 Amazon EC2 인스턴스에서 실행되는 Windows에만 적용됩니다.

Amazon Elastic Compute Cloud(EC2) 인스턴스

다음 예는 Amazon EC2 인스턴스를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

> **Important**
>
> Amazon EC2 인스턴스가 `stopped` 상태가 되면 모니터링에서 제거됩니다. `running` 상태로 돌아갈 때 CloudWatch 애플리케이션 인사이트 콘솔의 애플리케이션 세부 정보 페이지의 모니터링되지 않는 구성 요소 목록에 추가됩니다. 애플리케이션에 대해 새 리소스의 자동 모니터링이 활성화된 경우 인스턴스가 다음 모니터링되는 구성 요소 목록에 추가됩니다 그러나 로그와 지표는 작업 로드에 대한 기본값으로 설정됩니다. 이전 로그 및 지표 구성은 저장되지 않습니다.

```
{
    "alarmMetrics" : [
      {
        "alarmMetricName" : "CPUUtilization",
        "monitor" : true
      }, {
        "alarmMetricName" : "StatusCheckFailed"
      }
```

```
    ],
    "logs" : [
      {
        "logGroupName" : "my_log_group",
        "logPath" : "C:\\LogFolder\\*",
        "logType" : "APPLICATION",
        "monitor" : true
      },
      {
        "logGroupName" : "my_log_group_2",
        "logPath" : "C:\\LogFolder2\\*",
        "logType" : "IIS",
        "encoding" : "utf-8"
      }
    ],
    "processes" : [
        {
            "processName" : "my_process",
            "alarmMetrics" : [
                {
                    "alarmMetricName" : "procstat cpu_usage",
                    "monitor" : true
                }, {
                    "alarmMetricName" : "procstat memory_rss",
                    "monitor" : true
                }
            ]
        }
    ],
    "windowsEvents" : [
      {
        "logGroupName" : "my_log_group_3",
        "eventName" : "Application",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
        "monitor" : true
      }, {
        "logGroupName" : "my_log_group_4",
        "eventName" : "System",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
        "monitor" : true
    }],
     "alarms" : [
      {
        "alarmName" : "my_instance_alarm_1",
```

```
        "severity" : "HIGH"
      },
      {
        "alarmName" : "my_instance_alarm_2",
        "severity" : "LOW"
      }
    ],
     "subComponents" : [
     {
       "subComponentType" : "AWS::EC2::Volume",
       "alarmMetrics" : [
       {
         "alarmMetricName" : "VolumeQueueLength",
         "monitor" : "true"
       },
       {
         "alarmMetricName" : "VolumeThroughputPercentage",
         "monitor" : "true"
       },
       {
         "alarmMetricName" : "BurstBalance",
         "monitor" : "true"
       }
    }]
  }
```

Amazon Elastic Container Service(Amazon ECS)

다음 예는 Amazon Elastic Container Service(Amazon ECS)를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "alarmMetrics":[
       {
          "alarmMetricName":"CpuUtilized",
          "monitor":true
       },
       {
          "alarmMetricName":"MemoryUtilized",
          "monitor":true
       },
       {
          "alarmMetricName":"NetworkRxBytes",
```

```
            "monitor":true
         },
         {
            "alarmMetricName":"NetworkTxBytes",
            "monitor":true
         },
         {
            "alarmMetricName":"RunningTaskCount",
            "monitor":true
         },
         {
            "alarmMetricName":"PendingTaskCount",
            "monitor":true
         },
         {
            "alarmMetricName":"StorageReadBytes",
            "monitor":true
         },
         {
            "alarmMetricName":"StorageWriteBytes",
            "monitor":true
         }
      ],
      "logs":[
         {
            "logGroupName":"/ecs/my-task-definition",
            "logType":"APPLICATION",
            "monitor":true
         }
      ],
      "subComponents":[
         {
            "subComponentType":"AWS::ElasticLoadBalancing::LoadBalancer",
            "alarmMetrics":[
               {
                  "alarmMetricName":"HTTPCode_Backend_4XX",
                  "monitor":true
               },
               {
                  "alarmMetricName":"HTTPCode_Backend_5XX",
                  "monitor":true
               },
               {
                  "alarmMetricName":"Latency",
```

```
                "monitor":true
            },
            {
                "alarmMetricName":"SurgeQueueLength",
                "monitor":true
            },
            {
                "alarmMetricName":"UnHealthyHostCount",
                "monitor":true
            }
        ]
    },
    {
        "subComponentType":"AWS::ElasticLoadBalancingV2::LoadBalancer",
        "alarmMetrics":[
            {
                "alarmMetricName":"HTTPCode_Target_4XX_Count",
                "monitor":true
            },
            {
                "alarmMetricName":"HTTPCode_Target_5XX_Count",
                "monitor":true
            },
            {
                "alarmMetricName":"TargetResponseTime",
                "monitor":true
            },
            {
                "alarmMetricName":"UnHealthyHostCount",
                "monitor":true
            }
        ]
    },
    {
        "subComponentType":"AWS::EC2::Instance",
        "alarmMetrics":[
            {
                "alarmMetricName":"CPUUtilization",
                "monitor":true
            },
            {
                "alarmMetricName":"StatusCheckFailed",
                "monitor":true
            },
```

```
            {
               "alarmMetricName":"disk_used_percent",
               "monitor":true
            },
            {
               "alarmMetricName":"mem_used_percent",
               "monitor":true
            }
         ],
         "logs":[
            {
               "logGroupName":"my_log_group",
               "logPath":"/mylog/path",
               "logType":"APPLICATION",
               "monitor":true
            }
         ],
         "processes" : [
            {
               "processName" : "my_process",
               "alarmMetrics" : [
                   {
                       "alarmMetricName" : "procstat cpu_usage",
                       "monitor" : true
                   }, {
                       "alarmMetricName" : "procstat memory_rss",
                       "monitor" : true
                   }
                ]
            }
         ],
         "windowsEvents":[
            {
               "logGroupName":"my_log_group_2",
               "eventName":"Application",
               "eventLevels":[
                  "ERROR",
                  "WARNING",
                  "CRITICAL"
               ],
               "monitor":true
            }
         ]
      },
```

```
        {
           "subComponentType":"AWS::EC2::Volume",
           "alarmMetrics":[
              {
                 "alarmMetricName":"VolumeQueueLength",
                 "monitor":"true"
              },
              {
                 "alarmMetricName":"VolumeThroughputPercentage",
                 "monitor":"true"
              },
              {
                 "alarmMetricName":"BurstBalance",
                 "monitor":"true"
              }
           ]
        }
     ]
  }
```

> **Note**
>
> - `AWS::EC2::Instance` 및 `AWS::EC2::Volume`의 `subComponents` 섹션은 ECS 서비스 또는 ECS 태스크가 EC2 시작 유형에서 실행되는 Amazon ECS 클러스터에만 적용됩니다.
> - `subComponents`에 있는 `AWS::EC2::Instance`의 `windowsEvents` 섹션은 Amazon EC2 인스턴스에서 실행되는 Windows에만 적용됩니다.

Amazon ECS 서비스

다음 예는 Amazon ECS 서비스를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "alarmMetrics":[
       {
          "alarmMetricName":"CPUUtilization",
          "monitor":true
       },
       {
          "alarmMetricName":"MemoryUtilization",
          "monitor":true
```

```
        },
        {
           "alarmMetricName":"CpuUtilized",
           "monitor":true
        },
        {
           "alarmMetricName":"MemoryUtilized",
           "monitor":true
        },
        {
           "alarmMetricName":"NetworkRxBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"NetworkTxBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"RunningTaskCount",
           "monitor":true
        },
        {
           "alarmMetricName":"PendingTaskCount",
           "monitor":true
        },
        {
           "alarmMetricName":"StorageReadBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"StorageWriteBytes",
           "monitor":true
        }
     ],
     "logs":[
        {
           "logGroupName":"/ecs/my-task-definition",
           "logType":"APPLICATION",
           "monitor":true
        }
     ],
     "subComponents":[
        {
           "subComponentType":"AWS::ElasticLoadBalancing::LoadBalancer",
```

```
          "alarmMetrics":[
             {
                "alarmMetricName":"HTTPCode_Backend_4XX",
                "monitor":true
             },
             {
                "alarmMetricName":"HTTPCode_Backend_5XX",
                "monitor":true
             },
             {
                "alarmMetricName":"Latency",
                "monitor":true
             },
             {
                "alarmMetricName":"SurgeQueueLength",
                "monitor":true
             },
             {
                "alarmMetricName":"UnHealthyHostCount",
                "monitor":true
             }
          ]
       },
       {
          "subComponentType":"AWS::ElasticLoadBalancingV2::LoadBalancer",
          "alarmMetrics":[
             {
                "alarmMetricName":"HTTPCode_Target_4XX_Count",
                "monitor":true
             },
             {
                "alarmMetricName":"HTTPCode_Target_5XX_Count",
                "monitor":true
             },
             {
                "alarmMetricName":"TargetResponseTime",
                "monitor":true
             },
             {
                "alarmMetricName":"UnHealthyHostCount",
                "monitor":true
             }
          ]
       },
```

```
        {
           "subComponentType":"AWS::EC2::Instance",
           "alarmMetrics":[
              {
                 "alarmMetricName":"CPUUtilization",
                 "monitor":true
              },
              {
                 "alarmMetricName":"StatusCheckFailed",
                 "monitor":true
              },
              {
                 "alarmMetricName":"disk_used_percent",
                 "monitor":true
              },
              {
                 "alarmMetricName":"mem_used_percent",
                 "monitor":true
              }
           ],
           "logs":[
              {
                 "logGroupName":"my_log_group",
                 "logPath":"/mylog/path",
                 "logType":"APPLICATION",
                 "monitor":true
              }
           ],
           "processes" : [
              {
                 "processName" : "my_process",
                 "alarmMetrics" : [
                 {
                     "alarmMetricName" : "procstat cpu_usage",
                     "monitor" : true
                 }, {
                     "alarmMetricName" : "procstat memory_rss",
                     "monitor" : true
                 }
             ]
         }
    ],
           "windowsEvents":[
              {
```

```
                "logGroupName":"my_log_group_2",
                "eventName":"Application",
                "eventLevels":[
                   "ERROR",
                   "WARNING",
                   "CRITICAL"
                ],
                "monitor":true
             }
          ]
       },
       {
          "subComponentType":"AWS::EC2::Volume",
          "alarmMetrics":[
             {
                "alarmMetricName":"VolumeQueueLength",
                "monitor":"true"
             },
             {
                "alarmMetricName":"VolumeThroughputPercentage",
                "monitor":"true"
             },
             {
                "alarmMetricName":"BurstBalance",
                "monitor":"true"
             }
          ]
       }
    ]
}
```

> **Note**
>
> - `AWS::EC2::Instance` 및 `AWS::EC2::Volume`의 `subComponents` 섹션은 EC2 시작 유형에서 실행되는 Amazon ECS에만 적용됩니다.
> - `subComponents`에 있는 `AWS::EC2::Instance`의 `windowsEvents` 섹션은 Amazon EC2 인스턴스에서 실행되는 Windows에만 적용됩니다.

Amazon ECS 작업

다음 예는 Amazon ECS 태스크를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "logs":[
       {
          "logGroupName":"/ecs/my-task-definition",
          "logType":"APPLICATION",
          "monitor":true
       }
    ],
    "processes" : [
        {
            "processName" : "my_process",
            "alarmMetrics" : [
                {
                    "alarmMetricName" : "procstat cpu_usage",
                    "monitor" : true
                }, {
                    "alarmMetricName" : "procstat memory_rss",
                    "monitor" : true
                }
            ]
        }
    ]
 }
```

Amazon Elastic File System(Amazon EFS)

다음 예는 Amazon FSx를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
   "alarmMetrics": [
     {
       "alarmMetricName": "BurstCreditBalance",
       "monitor": true
     },
     {
       "alarmMetricName": "PercentIOLimit",
       "monitor": true
     },
     {
```

```
        "alarmMetricName": "PermittedThroughput",
        "monitor": true
      },
      {
        "alarmMetricName": "MeteredIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "TotalIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "DataWriteIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "DataReadIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "MetadataIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "ClientConnections",
        "monitor": true
      },
      {
        "alarmMetricName": "TimeSinceLastSync",
        "monitor": true
      },
      {
        "alarmMetricName": "Throughput",
        "monitor": true
      },
      {
        "alarmMetricName": "PercentageOfPermittedThroughputUtilization",
        "monitor": true
      },
      {
        "alarmMetricName": "ThroughputIOPS",
        "monitor": true
      },
      {
```

```
        "alarmMetricName": "PercentThroughputDataReadIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "PercentThroughputDataWriteIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "PercentageOfIOPSDataReadIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "PercentageOfIOPSDataWriteIOBytes",
        "monitor": true
      },
      {
        "alarmMetricName": "AverageDataReadIOBytesSize",
        "monitor": true
      },
      {
        "alarmMetricName": "AverageDataWriteIOBytesSize",
        "monitor": true
      }
    ],
    "logs": [
     {
     "logGroupName": "/aws/efs/utils",
     "logType": "EFS_MOUNT_STATUS",
     "monitor": true,
     }
    ]
 }
```

Amazon FSx

다음 예는 Amazon FSx를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "DataReadBytes",
      "monitor": true
    },
    {
```

```
      "alarmMetricName": "DataWriteBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "DataReadOperations",
      "monitor": true
    },
    {
      "alarmMetricName": "DataWriteOperations",
      "monitor": true
    },
    {
      "alarmMetricName": "MetadataOperations",
      "monitor": true
    },
    {
      "alarmMetricName": "FreeStorageCapacity",
      "monitor": true
    }
  ]
}
```

Amazon Relational Database Service(RDS) Aurora MySQL

다음 예는 Amazon RDS Aurora MySQL을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    },
    {
      "alarmMetricName": "CommitLatency",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "MYSQL",
      "monitor": true,
    },
    {
      "logType": "MYSQL_SLOW_QUERY",
```

```
      "monitor": false
    }
  ]
}
```

Amazon Relational Database Service(RDS) 인스턴스

다음 예는 Amazon RDS 인스턴스를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "BurstBalance",
      "monitor" : true
    }, {
      "alarmMetricName" : "WriteThroughput",
      "monitor" : false
    }
  ],

  "alarms" : [
    {
      "alarmName" : "my_rds_instance_alarm",
      "severity" : "MEDIUM"
    }
  ]
}
```

Amazon Route 53 상태 확인

다음 예에서는 Amazon Route 53 상태 확인을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ChildHealthCheckHealthyCount",
      "monitor": true
    },
    {
      "alarmMetricName": "ConnectionTime",
      "monitor": true
    },
```

```
    {
      "alarmMetricName": "HealthCheckPercentageHealthy",
      "monitor": true
    },
    {
      "alarmMetricName": "HealthCheckStatus",
      "monitor": true
    },
    {
      "alarmMetricName": "SSLHandshakeTime",
      "monitor": true
    },
    {
      "alarmMetricName": "TimeToFirstByte",
      "monitor": true
    }
  ]
}
```

Amazon Route 53 호스팅 영역

다음 예에서는 Amazon Route 53 호스팅 영역에 대한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "DNSQueries",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECInternalFailure",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECKeySigningKeysNeedingAction",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECKeySigningKeyMaxNeedingActionAge",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECKeySigningKeyAge",
      "monitor": true
```

```
    }
  ],
  "logs": [
    {
      "logGroupName":"/hosted-zone/logs",
      "logType": "ROUTE53_DNS_PUBLIC_QUERY_LOGS",
      "monitor": true
    }
  ]
}
```

Amazon Route 53 Resolver 엔드포인트

다음 예에서는 Amazon Route 53 Resolver 엔드포인트에 대한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "EndpointHealthyENICount",
      "monitor": true
    },
    {
      "alarmMetricName": "EndpointUnHealthyENICount",
      "monitor": true
    },
    {
      "alarmMetricName": "InboundQueryVolume",
      "monitor": true
    },
    {
      "alarmMetricName": "OutboundQueryVolume",
      "monitor": true
    },
    {
      "alarmMetricName": "OutboundQueryAggregateVolume",
      "monitor": true
    }
  ]
}
```

Amazon Route 53 Resolver 쿼리 로깅 구성

다음 예에서는 Amazon Route 53 Resolver 쿼리 로깅 구성을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "logs": [
    {
      "logGroupName": "/resolver-query-log-config/logs",
      "logType": "ROUTE53_RESOLVER_QUERY_LOGS",
      "monitor": true
    }
  ]
}
```

Amazon S3 버킷

다음 예는 Amazon S3 버킷을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "alarmMetrics" : [
        {
            "alarmMetricName" : "ReplicationLatency",
            "monitor" : true
        }, {
            "alarmMetricName" : "5xxErrors",
            "monitor" : true
        }, {
            "alarmMetricName" : "BytesDownloaded"
            "monitor" : true
        }
    ]
}
```

Amazon Simple Queue Service(SQS)

다음 예는 Amazon Simple Queue Service를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "ApproximateAgeOfOldestMessage"
```

```
    }, {
      "alarmMetricName" : "NumberOfEmptyReceives"
    }
  ],
  "alarms" : [
    {
      "alarmName" : "my_sqs_alarm",
      "severity" : "MEDIUM"
    }
  ]
}
```

Amazon SNS 주제

다음 예는 Amazon SNS 주제를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "NumberOfNotificationsFailed",
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFilteredOut-InvalidAttributes",
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFilteredOut-NoMessageAttributes",
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFailedToRedriveToDlq",
      "monitor": true
    }
  ]
}
```

Amazon Virtual Private Cloud(VPC)

다음 예에서는 Amazon VPC에 대한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
```

```
  "alarmMetrics": [
    {
      "alarmMetricName": "NetworkAddressUsage",
      "monitor": true
    },
    {
      "alarmMetricName": "NetworkAddressUsagePeered",
      "monitor": true
    },
    {
      "alarmMetricName": "VPCFirewallQueryVolume",
      "monitor": true
    }
  ]
}
```

Amazon VPC Network Address Translation(NAT) 게이트웨이

다음 예에서는 NAT 게이트웨이에 대한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ErrorPortAllocation",
      "monitor": true
    },
    {
      "alarmMetricName": "IdleTimeoutCount",
      "monitor": true
    }
  ]
}
```

API Gateway REST API 스테이지

다음 예는 API Gateway REST API 스테이지를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
     "alarmMetrics" : [
         {
             "alarmMetricName" : "4XXError",
             "monitor" : true
         },
```

```
         {
             "alarmMetricName" : "5XXError",
             "monitor" : true
         }
     ],
    "logs" : [
        {
            "logType" : "API_GATEWAY_EXECUTION",
            "monitor" : true
        },
        {
            "logType" : "API_GATEWAY_ACCESS",
            "monitor" : true
        }
    ]
}
```

Application Elastic Load Balancing

다음 예는 Application Elastic Load Balancing을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ActiveConnectionCount",
    }, {
      "alarmMetricName": "TargetResponseTime"
    }
  ],
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "CPUUtilization",
        }, {
          "alarmMetricName": "StatusCheckFailed"
        }
      ],
      "logs": [
        {
          "logGroupName": "my_log_group",
          "logPath": "C:\\LogFolder\\*",
          "logType": "APPLICATION",
```

```
        }
      ],
      "windowsEvents": [
        {
          "logGroupName": "my_log_group_2",
          "eventName": "Application",
          "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ]
        }
      ]
    }, {
      "subComponentType": "AWS::EC2::Volume",
      "alarmMetrics": [
        {
          "alarmMetricName": "VolumeQueueLength",
        }, {
          "alarmMetricName": "BurstBalance"
        }
      ]
    }
  ],

  "alarms": [
    {
      "alarmName": "my_alb_alarm",
      "severity": "LOW"
    }
  ]
}
```

AWS Lambda 함수

다음 예는 AWS Lambda 함수를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "Errors",
      "monitor": true
    },
    {
      "alarmMetricName": "Throttles",
      "monitor": true
    },
    {
```

```
      "alarmMetricName": "IteratorAge",
      "monitor": true
    },
    {
      "alarmMetricName": "Duration",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "DEFAULT",
      "monitor": true
    }
  ]
}
```

AWS Network Firewall 규칙 그룹

다음 예에서는 AWS Network Firewall 규칙 그룹에 대한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "FirewallRuleGroupQueryVolume",
      "monitor": true
    }
  ]
}
```

AWS Network Firewall 규칙 그룹 연결

다음 예에서는 AWS Network Firewall 규칙 그룹 연결에 대한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "FirewallRuleGroupQueryVolume",
      "monitor": true
    }
  ]
}
```

AWS Step Functions

다음 예는 AWS Step Functions를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ExecutionsFailed",
      "monitor": true
    },
    {
      "alarmMetricName": "LambdaFunctionsFailed",
      "monitor": true
    },
    {
      "alarmMetricName": "ProvisionedRefillRate",
      "monitor": true
    }
  ],
  "logs": [
    {
     "logGroupName": "/aws/states/HelloWorld-Logs",
      "logType": "STEP_FUNCTION",
      "monitor": true,
    }
  ]
}
```

고객이 그룹화한 Amazon EC2 인스턴스

다음 예는 고객이 그룹화한 Amazon EC2 인스턴스를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "subComponents": [
      {
        "subComponentType": "AWS::EC2::Instance",
        "alarmMetrics": [
          {
            "alarmMetricName": "CPUUtilization",
          },
          {
            "alarmMetricName": "StatusCheckFailed"
```

```
          }
        ],
        "logs": [
          {
            "logGroupName": "my_log_group",
            "logPath": "C:\\LogFolder\\*",
            "logType": "APPLICATION",
          }
        ],
        "processes": [
            {
                "processName": "my_process",
                "alarmMetrics": [
                    {
                        "alarmMetricName": "procstat cpu_usage",
                        "monitor": true
                    }, {
                        "alarmMetricName": "procstat memory_rss",
                        "monitor": true
                    }
                ]
            }
        ],
        "windowsEvents": [
          {
            "logGroupName": "my_log_group_2",
            "eventName": "Application",
            "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ]
          }
        ]
      }, {
        "subComponentType": "AWS::EC2::Volume",
        "alarmMetrics": [
          {
            "alarmMetricName": "VolumeQueueLength",
          }, {
            "alarmMetricName": "BurstBalance"
          }
        ]
      }
    ],
    "alarms": [
      {
        "alarmName": "my_alarm",
```

```
        "severity": "MEDIUM"
      }
    ]
  }
```

Elastic Load Balancing

다음 예는 Elastic Load Balancing을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "alarmMetrics": [
      {
        "alarmMetricName": "EstimatedALBActiveConnectionCount"
      }, {
        "alarmMetricName": "HTTPCode_Backend_5XX"
      }
    ],
    "subComponents": [
      {
        "subComponentType": "AWS::EC2::Instance",
        "alarmMetrics": [
          {
            "alarmMetricName": "CPUUtilization"
          }, {
            "alarmMetricName": "StatusCheckFailed"
          }
        ],
        "logs": [
          {
            "logGroupName": "my_log_group",
            "logPath": "C:\\LogFolder\\*",
            "logType": "APPLICATION"
          }
        ],
        "processes": [
          {
            "processName": "my_process",
            "alarmMetrics": [
                {
                  "alarmMetricName": "procstat cpu_usage",
                  "monitor": true
                }, {
                  "alarmMetricName": "procstat memory_rss",
                  "monitor": true
```

```
                }
              ]
          }
        ],
        "windowsEvents": [
          {
            "logGroupName": "my_log_group_2",
            "eventName": "Application",
            "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ],
            "monitor": true
          }
        ]
      }, {
        "subComponentType": "AWS::EC2::Volume",
        "alarmMetrics": [
          {
            "alarmMetricName": "VolumeQueueLength"
          }, {
            "alarmMetricName": "BurstBalance"
          }
        ]
      }
    ],

    "alarms": [
      {
        "alarmName": "my_elb_alarm",
        "severity": "HIGH"
      }
    ]
  }
```

Java

다음 예는 Java를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [ {
    "alarmMetricName": "java_lang_threading_threadcount",
    "monitor": true
  },
  {
    "alarmMetricName": "java_lang_memory_heapmemoryusage_used",
    "monitor": true
```

```
  },
  {
    "alarmMetricName": "java_lang_memory_heapmemoryusage_committed",
    "monitor": true
  }],
  "logs": [ ],
  "JMXPrometheusExporter": {
      "hostPort": "8686",
      "prometheusPort": "9404"
  }
}
```

> **Note**
>
> Application Insights는 Prometheus JMX Exporter에 대한 인증 구성을 지원하지 않습니다. 인증 설정 방법에 대한 자세한 내용은 Prometheus JMX Exporter 구성 예를 참조하세요.

Amazon EC2의 Kubernetes

다음 예는 Amazon EC2의 Kubernetes를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
    "alarmMetrics":[
       {
          "alarmMetricName":"cluster_failed_node_count",
          "monitor":true
       },
       {
          "alarmMetricName":"node_cpu_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName":"node_cpu_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"node_filesystem_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"node_memory_reserved_capacity",
          "monitor":true
```

```
        },
        {
           "alarmMetricName":"node_memory_utilization",
           "monitor":true
        },
        {
           "alarmMetricName":"node_network_total_bytes",
           "monitor":true
        },
        {
           "alarmMetricName":"pod_cpu_reserved_capacity",
           "monitor":true
        },
        {
           "alarmMetricName":"pod_cpu_utilization",
           "monitor":true
        },
        {
           "alarmMetricName":"pod_cpu_utilization_over_pod_limit",
           "monitor":true
        },
        {
           "alarmMetricName":"pod_memory_reserved_capacity",
           "monitor":true
        },
        {
           "alarmMetricName":"pod_memory_utilization",
           "monitor":true
        },
        {
           "alarmMetricName":"pod_memory_utilization_over_pod_limit",
           "monitor":true
        },
        {
           "alarmMetricName":"pod_network_rx_bytes",
           "monitor":true
        },
        {
           "alarmMetricName":"pod_network_tx_bytes",
           "monitor":true
        }
     ],
     "logs":[
        {
```

```
            "logGroupName":"/aws/containerinsights/kubernetes/application",
            "logType":"APPLICATION",
            "monitor":true,
            "encoding":"utf-8"
         }
      ],
      "subComponents":[
         {
            "subComponentType":"AWS::EC2::Instance",
            "alarmMetrics":[
               {
                  "alarmMetricName":"CPUUtilization",
                  "monitor":true
               },
               {
                  "alarmMetricName":"StatusCheckFailed",
                  "monitor":true
               },
               {
                  "alarmMetricName":"disk_used_percent",
                  "monitor":true
               },
               {
                  "alarmMetricName":"mem_used_percent",
                  "monitor":true
               }
            ],
            "logs":[
               {
                  "logGroupName":"APPLICATION-KubernetesClusterOnEC2-IAD",
                  "logPath":"",
                  "logType":"APPLICATION",
                  "monitor":true,
                  "encoding":"utf-8"
               }
            ],
            "processes" : [
              {
                  "processName" : "my_process",
                  "alarmMetrics" : [
                      {
                          "alarmMetricName" : "procstat cpu_usage",
                          "monitor" : true
                      }, {
```

```
                        "alarmMetricName" : "procstat memory_rss",
                        "monitor" : true
                    }
                ]
            }
        ]
       },
       {
          "subComponentType":"AWS::EC2::Volume",
          "alarmMetrics":[
             {
                "alarmMetricName":"VolumeReadBytes",
                "monitor":true
             },
             {
                "alarmMetricName":"VolumeWriteBytes",
                "monitor":true
             },
             {
                "alarmMetricName":"VolumeReadOps",
                "monitor":true
             },
             {
                "alarmMetricName":"VolumeWriteOps",
                "monitor":true
             },
             {
                "alarmMetricName":"VolumeQueueLength",
                "monitor":true
             },
             {
                "alarmMetricName":"BurstBalance",
                "monitor":true
             }
          ]
       }
    ]
 }
```

RDS MariaDB 및 RDS MySQL

다음 예에서는 RDS MariaDB 및 RDS MySQL을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
```

```
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "MYSQL",
      "monitor": true,
    },
    {
      "logType": "MYSQL_SLOW_QUERY",
      "monitor": false
    }
  ]
}
```

RDS Oracle

다음 예는 RDS Oracle을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "ORACLE_ALERT",
      "monitor": true,
    },
    {
      "logType": "ORACLE_LISTENER",
      "monitor": false
    }
  ]
}
```

RDS PostgreSQL

다음 예는 RDS PostgreSQL을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "POSTGRESQL",
      "monitor": true
    }
  ]
}
```

Amazon EC2에 대한 SAP ASE

다음 예는 Amazon EC2에 대한 SAP ASE를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "asedb_database_availability",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_trunc_log_on_chkpt_enabled",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_db_backup_age_in_days",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_transaction_log_backup_age_in_hours",
          "monitor": true
```

```
        },
        {
          "alarmMetricName": "asedb_suspected_database",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_log_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_locked_login",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_data_cache_hit_ratio",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_ASE_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2.log",
          "logType": "SAP_ASE_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_BACKUP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2_BS.log",
          "logType": "SAP_ASE_BACKUP_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
    ],
    "sapAsePrometheusExporter": {
      "sapAseSid": "ASE",
      "sapAsePort": "4901",
      "sapAseSecretName": "ASE_DB_CREDS",
      "prometheusPort": "9399",
      "agreeToEnableASEMonitoring": true
```

```
}
```

Amazon EC2에 대한 SAP ASE 고가용성

다음 예는 Amazon EC2에 대한 SAP ASE 고가용성을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "asedb_database_availability",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_trunc_log_on_chkpt_enabled",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_db_backup_age_in_days",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_transaction_log_backup_age_in_hours",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_suspected_database",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_ha_replication_state",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_ha_replication_mode",
          "monitor": true
```

```
        },
        {
          "alarmMetricName": "asedb_ha_replication_latency_in_minutes",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_ASE_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2.log",
          "logType": "SAP_ASE_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_BACKUP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2_BS.log",
          "logType": "SAP_ASE_BACKUP_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_REP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/DM/repservername/repservername.log",
          "logType": "SAP_ASE_REP_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_RMA_AGENT_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/DM/RMA-*/instances/AgentContainer/logs/",
          "logType": "SAP_ASE_RMA_AGENT_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_FAULT_MANAGER_LOGS-my-resource-group",
          "logPath": "/opt/sap/FaultManager/dev_sybdbfm",
          "logType": "SAP_ASE_FAULT_MANAGER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
  ],
  "sapAsePrometheusExporter": {
```

```
    "sapAseSid": "ASE",
    "sapAsePort": "4901",
    "sapAseSecretName": "ASE_DB_CREDS",
    "prometheusPort": "9399",
    "agreeToEnableASEMonitoring": true
}
```

Amazon EC2에 대한 SAP HANA

다음 예는 Amazon EC2에 대한 SAP HANA를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "hanadb_server_startup_time_variations_seconds",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_5_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_4_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_out_of_memory_events_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_max_trigger_read_ratio_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_table_allocation_limit_used_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_cpu_usage_percent",
          "monitor": true
        },
```

```
        {
          "alarmMetricName": "hanadb_plan_cache_hit_ratio_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_last_data_backup_age_days",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_HANA_TRACE-my-resourge-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.trc",
          "logType": "SAP_HANA_TRACE",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_HANA_LOGS-my-resource-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.log",
          "logType": "SAP_HANA_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "hanaPrometheusExporter": {
    "hanaSid": "HDB",
    "hanaPort": "30013",
    "hanaSecretName": "HANA_DB_CREDS",
    "prometheusPort": "9668"
  }
}
```

Amazon EC2에 대한 SAP HANA 고가용성

다음 예는 Amazon EC2에 대한 SAP HANA 고가용성을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "subComponents": [
    {
```

```
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "hanadb_server_startup_time_variations_seconds",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_5_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_4_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_out_of_memory_events_count",
          "monitor": true
        },
        {
          "alarmMetricName": "ha_cluster_pacemaker_stonith_enabled",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_HANA_TRACE-my-resourge-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.trc",
          "logType": "SAP_HANA_TRACE",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_HANA_HIGH_AVAILABILITY-my-resource-group",
          "logPath": "/var/log/pacemaker/pacemaker.log",
          "logType": "SAP_HANA_HIGH_AVAILABILITY",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "hanaPrometheusExporter": {
    "hanaSid": "HDB",
    "hanaPort": "30013",
```

```
    "hanaSecretName": "HANA_DB_CREDS",
    "prometheusPort": "9668"
  },
  "haClusterPrometheusExporter": {
    "prometheusPort": "9664"
  }
}
```

Amazon EC2의 SAP NetWeaver

다음 예제에서는 Amazon EC2에 대한 SAP NetWeaver를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "CPUUtilization",
          "monitor": true
        },
        {
          "alarmMetricName": "StatusCheckFailed",
          "monitor": true
        },
        {
          "alarmMetricName": "disk_used_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "mem_used_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialog",
          "monitor": true
        },
        {
```

```
          "alarmMetricName": "sap_alerts_ResponseTimeDialogRFC",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_DBRequestTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_LongRunners",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_AbortedJobs",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_BasisSystem",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Database",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Security",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_System",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_QueueTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Availability",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_start_service_processes",
          "monitor": true
        },
        {
```

```
          "alarmMetricName": "sap_dispatcher_queue_now",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_dispatcher_queue_max",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_max",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_now",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_state",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_replication_state",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_NETWEAVER_DEV_TRACE_LOGS-NetWeaver-ML4",
          "logPath": "/usr/sap/ML4/*/work/dev_w*",
          "logType": "SAP_NETWEAVER_DEV_TRACE_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "netWeaverPrometheusExporter": {
    "sapSid": "ML4",
    "instanceNumbers": [
      "00",
      "11"
    ],
    "prometheusPort": "9680"
  }
```

```
}
```

Amazon EC2에 대한 SAP NetWeaver High Availability

다음 예제에서는 Amazon EC2에 대한 SAP NetWeaver High Availability을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "ha_cluster_corosync_ring_errors",
          "monitor": true
        },
        {
          "alarmMetricName": "ha_cluster_pacemaker_fail_count",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_HA_check_failover_config_state",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_HA_get_failover_config_HAActive",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_AbortedJobs",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Availability",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_BasisSystem",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_DBRequestTime",
          "monitor": true
```

```
        },
        {
          "alarmMetricName": "sap_alerts_Database",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_FrontendResponseTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_LongRunners",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_QueueTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialog",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialogRFC",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Security",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Shortdumps",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_SqlError",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_System",
          "monitor": true
```

```
        },
        {
          "alarmMetricName": "sap_enqueue_server_replication_state",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_start_service_processes",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_NETWEAVER_DEV_TRACE_LOGS-NetWeaver-PR1",
          "logPath": "/usr/sap/<SID>/D*/work/dev_w*",
          "logType": "SAP_NETWEAVER_DEV_TRACE_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "haClusterPrometheusExporter": {
    "prometheusPort": "9664"
  },
  "netWeaverPrometheusExporter": {
    "sapSid": "PR1",
    "instanceNumbers": [
      "11",
      "12"
    ],
    "prometheusPort": "9680"
  }
}
```

SQL Always On 가용성 그룹

다음 예는 SQL Always On 가용성 그룹을 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "subComponents" : [ {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [ {
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
```

```
    }, {
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
    }, {
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Buffer cache hit ratio",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Page life expectancy",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics Processes blocked",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics User Connections",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Locks Number of Deadlocks/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:SQL Statistics Batch Requests/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica File Bytes Received/sec",
```

```
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log remaining for undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Send Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Mirrored Write Transaction/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Recovery Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redo Bytes Remaining",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redone Bytes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Total Log requiring undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Transaction Delay",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
```

```
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP-<RESOURCE_GROUP_NAME>",
      "logPath" : "C:\\Program Files\\Microsoft SQL Server\\MSSQL**.MSSQLSERVER\\MSSQL\
\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
}
```

SQL 장애 조치 클러스터 인스턴스

다음 예는 SQL 장애 조치 클러스터 인스턴스를 위한 JSON 형식의 구성 요소 구성을 보여줍니다.

```
{
  "subComponents" : [ {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [ {
```

```
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
    }, {
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
    }, {
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Normal Messages Queue Length/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Urgent Message Queue Length/se",
      "monitor" : true
    }, {
      "alarmMetricName" : "Reconnect Count",
      "monitor" : true
    }, {
      "alarmMetricName" : "Unacknowledged Message Queue Length/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Messages Outstanding",
      "monitor" : true
```

```
    }, {
      "alarmMetricName" : "Messages Sent/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Database Update Messages/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Update Messages/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Flushes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Crypto Checkpoints Saved/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Crypto Checkpoints Restored/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Registry Checkpoints Restored/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Registry Checkpoints Saved/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Cluster API Calls/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Resource API Calls/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Cluster Handles/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Resource Handles/sec",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL"],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
```

```
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_FAILOVER_CLUSTER_INSTANCE-<RESOURCE_GROUP_NAME>",
      "logPath" : "\\\\amznfsxjmzbykwn.mydomain.aws\\SQLDB\\MSSQL**.MSSQLSERVER\\MSSQL\
\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
}
```

CloudFormation 템플릿을 사용하여 CloudWatch Application Insights 모니터링 생성 및 구성

AWS CloudFormation 템플릿에서 직접 애플리케이션, 데이터베이스 및 웹 서버로 주요 지표 및 원격 측정을 포함한 Application Insights 모니터링을 추가할 수 있습니다.

이 단원에서는 Application Insights 모니터링을 생성하고 구성하는 데 도움이 되는 JSON 형식 및 YAML 형식의 샘플 AWS CloudFormation 템플릿을 제공합니다.

Application Insights 리소스 및 속성 참조에 대해 살펴보려면 AWS CloudFormation 사용 설명서의 ApplicationInsights 리소스 유형 참조 단원을 참조하세요.

샘플 템플릿

- AWS CloudFormation 스택 전체에 대한 Application Insights 애플리케이션 생성
- 세부 설정을 사용하여 Application Insights 애플리케이션 생성
- CUSTOM 모드 구성 요소 구성으로 Application Insights 애플리케이션 생성
- DEFAULT 모드 구성 요소 구성으로 Application Insights 애플리케이션 생성
- DEFAULT_WITH_OVERWRITE 모드 구성 요소 구성으로 Application Insights 애플리케이션 생성

AWS CloudFormation 스택 전체에 대한 Application Insights 애플리케이션 생성

다음 템플릿을 적용하려면 AWS 리소스를 생성하고 이러한 리소스를 모니터링할 Application Insights 애플리케이션을 생성할 리소스 그룹을 하나 이상 생성해야 합니다. 자세한 내용은 AWS Resource Groups 시작하기 단원을 참조하세요.

다음 템플릿의 처음 두 부분에서는 리소스 및 리소스 그룹을 지정합니다. 템플릿의 마지막 부분에서는 리소스 그룹의 Application Insights 애플리케이션을 생성하지만 애플리케이션을 구성하거나 모니터링을 적용하지는 않습니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 CreateApplication 명령 세부 정보를 참조하세요.

JSON 형식의 템플릿

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Test Resource Group stack",
    "Resources": {
        "EC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
```

```
                "ImageId" : "ami-abcd1234efgh5678i",
                "SecurityGroupIds" : ["sg-abcd1234"]
            }
        },
        ...
        "ResourceGroup": {
            "Type": "AWS::ResourceGroups::Group",
            "Properties": {
                "Name": "my_resource_group"
            }
        },
        "AppInsightsApp": {
            "Type": "AWS::ApplicationInsights::Application",
            "Properties": {
                "ResourceGroupName": "my_resource_group"
            },
            "DependsOn" : "ResourceGroup"
        }
    }
}
```

YAML 형식의 템플릿

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Test Resource Group stack
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-abcd1234efgh5678i
      SecurityGroupIds:
      - sg-abcd1234
  ...
  ResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name: my_resource_group
  AppInsightsApp:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName: my_resource_group
    DependsOn: ResourceGroup
```

다음 템플릿 섹션에서는 Application Insights 애플리케이션에 기본 모니터링 구성을 적용합니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 CreateApplication 명령 세부 정보를 참조하세요.

AutoConfigurationEnabled를 true로 설정하면 애플리케이션의 모든 구성 요소가 DEFAULT 애플리케이션 티어에 대한 권장 모니터링 설정으로 구성됩니다. 이러한 설정 및 티어에 대한 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 DescribeComponentConfigurationRecommendation 및 UpdateComponentConfiguration을 참조하세요.

JSON 형식의 템플릿

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Test Application Insights Application stack",
    "Resources": {
        "AppInsightsApp": {
            "Type": "AWS::ApplicationInsights::Application",
            "Properties": {
                "ResourceGroupName": "my_resource_group",
                "AutoConfigurationEnabled": true
            }
        }
    }
}
```

YAML 형식의 템플릿

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Test Application Insights Application stack
Resources:
  AppInsightsApp:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName: my_resource_group
      AutoConfigurationEnabled: true
```

세부 설정을 사용하여 Application Insights 애플리케이션 생성

아래에 나와 있는 템플릿은 다음 작업을 수행합니다.

- CloudWatch Events 알림 및 OpsCenter가 활성화된 상태로 Application Insights 애플리케이션을 생성합니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 CreateApplication 명령 세부 정보를 참조하세요.
- 두 개의 태그를 사용하여 애플리케이션에 태그를 지정합니다. 그 중 하나는 태그 값이 없습니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 TagResource를 참조하세요.
- 두 개의 사용자 지정 인스턴스 그룹 구성 요소를 생성합니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 CreateComponent를 참조하세요.
- 두 개의 로그 패턴 세트를 생성합니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 CreateLogPattern을 참조하세요.
- `AutoConfigurationEnabled`를 `true`로 설정합니다. 이렇게 하면 `DEFAULT` 계층에 대한 권장 모니터링 설정을 사용하여 애플리케이션의 모든 구성 요소가 구성됩니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 DescribeComponentConfigurationRecommendation을 참조하세요.

JSON 형식의 템플릿

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "CWEMonitorEnabled": true,
        "OpsCenterEnabled": true,
        "OpsItemSNSTopicArn": "arn:aws:sns:us-east-1:123456789012:my_topic",
        "AutoConfigurationEnabled": true,
        "Tags": [
            {
                "Key": "key1",
                "Value": "value1"
            },
            {
                "Key": "key2",
                "Value": ""
            }
        ],
        "CustomComponents": [
            {
                "ComponentName": "test_component_1",
                "ResourceList": [
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i"
```

```
                ]
            },
            {
                "ComponentName": "test_component_2",
                "ResourceList": [
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i",
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i"
                ]
            }
        ],
        "LogPatternSets": [
            {
                "PatternSetName": "pattern_set_1",
                "LogPatterns": [
                    {
                        "PatternName": "deadlock_pattern",
                        "Pattern": ".*\\sDeadlocked\\sSchedulers(([^\\w].*)|($))",
                        "Rank": 1
                    }
                ]
            },
            {
                "PatternSetName": "pattern_set_2",
                "LogPatterns": [
                    {
                        "PatternName": "error_pattern",
                        "Pattern": ".*[\\s\\[]ERROR[\\s\\]].*",
                        "Rank": 1
                    },
                    {
                        "PatternName": "warning_pattern",
                        "Pattern": ".*[\\s\\[]WARN(ING)?[\\s\\]].*",
                        "Rank": 10
                    }
                ]
            }
        ]
    }
}
```

YAML 형식의 템플릿

```
---
```

```
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  CWEMonitorEnabled: true
  OpsCenterEnabled: true
  OpsItemSNSTopicArn: arn:aws:sns:us-east-1:123456789012:my_topic
  AutoConfigurationEnabled: true
  Tags:
  - Key: key1
    Value: value1
  - Key: key2
    Value: ''
  CustomComponents:
  - ComponentName: test_component_1
    ResourceList:
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
  - ComponentName: test_component_2
    ResourceList:
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
  LogPatternSets:
  - PatternSetName: pattern_set_1
    LogPatterns:
    - PatternName: deadlock_pattern
      Pattern: ".*\\sDeadlocked\\sSchedulers(([^\\w].*)|($))"
      Rank: 1
  - PatternSetName: pattern_set_2
    LogPatterns:
    - PatternName: error_pattern
      Pattern: ".*[\\s\\[]ERROR[\\s\\]].*"
      Rank: 1
    - PatternName: warning_pattern
      Pattern: ".*[\\s\\[]WARN(ING)?[\\s\\]].*"
      Rank: 10
```

CUSTOM 모드 구성 요소 구성으로 Application Insights 애플리케이션 생성

아래에 나와 있는 템플릿은 다음 작업을 수행합니다.

- Application Insights 애플리케이션을 생성합니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 CreateApplication을 참조하세요.
- 구성 요소 `my_component`에 대해 `ComponentConfigurationMode`를 `CUSTOM`으로 설정합니다. 이렇게 하면 이 구성 요소가 `CustomComponentConfiguration`에 지정된 구

성으로 구성됩니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 UpdateComponentConfiguration을 참조하세요.

JSON 형식의 템플릿

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group,
        "ComponentMonitoringSettings": [
            {
                "ComponentARN": "my_component",
                "Tier": "SQL_SERVER",
                "ComponentConfigurationMode": "CUSTOM",
                "CustomComponentConfiguration": {
                    "ConfigurationDetails": {
                        "AlarmMetrics": [
                            {
                                "AlarmMetricName": "StatusCheckFailed"
                            },
                            ...
                        ],
                        "Logs": [
                            {
                                "LogGroupName": "my_log_group_1",
                                "LogPath": "C:\\LogFolder_1\\*",
                                "LogType": "DOT_NET_CORE",
                                "Encoding": "utf-8",
                                "PatternSet": "my_pattern_set_1"
                            },
                            ...
                        ],
                        "WindowsEvents": [
                            {
                                "LogGroupName": "my_windows_event_log_group_1",
                                "EventName": "Application",
                                "EventLevels": [
                                    "ERROR",
                                    "WARNING",
                                    ...
                                ],
                                "Encoding": "utf-8",
                                "PatternSet": "my_pattern_set_2"
```

```
                        },
                        ...
                    ],
                    "Alarms": [
                        {
                            "AlarmName": "my_alarm_name",
                            "Severity": "HIGH"
                        },
                        ...
                    ]
                },
                "SubComponentTypeConfigurations": [
                    {
                        "SubComponentType": "EC2_INSTANCE",
                        "SubComponentConfigurationDetails": {
                            "AlarmMetrics": [
                                {
                                    "AlarmMetricName": "DiskReadOps"
                                },
                                ...
                            ],
                            "Logs": [
                                {
                                    "LogGroupName": "my_log_group_2",
                                    "LogPath": "C:\\LogFolder_2\\*",
                                    "LogType": "IIS",
                                    "Encoding": "utf-8",
                                    "PatternSet": "my_pattern_set_3"
                                },
                                ...
                            ],
                            "processes" : [
                                {
                                    "processName" : "my_process",
                                    "alarmMetrics" : [
                                {
                                    "alarmMetricName" : "procstat cpu_usage",
                                    "monitor" : true
                                }, {
                                    "alarmMetricName" : "procstat memory_rss",
                                    "monitor" : true
                                }
                            ]
                        }
```

```
                        ],
                                "WindowsEvents": [
                                    {
                                        "LogGroupName": "my_windows_event_log_group_2",
                                        "EventName": "Application",
                                        "EventLevels": [
                                            "ERROR",
                                            "WARNING",
                                            ...
                                        ],
                                        "Encoding": "utf-8",
                                        "PatternSet": "my_pattern_set_4"
                                    },
                                    ...
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```

YAML 형식의 템플릿

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
  - ComponentARN: my_component
    Tier: SQL_SERVER
    ComponentConfigurationMode: CUSTOM
    CustomComponentConfiguration:
      ConfigurationDetails:
        AlarmMetrics:
        - AlarmMetricName: StatusCheckFailed
        ...
        Logs:
        - LogGroupName: my_log_group_1
          LogPath: C:\LogFolder_1\*
          LogType: DOT_NET_CORE
```

```
          Encoding: utf-8
          PatternSet: my_pattern_set_1
        ...
        WindowsEvents:
        - LogGroupName: my_windows_event_log_group_1
          EventName: Application
          EventLevels:
          - ERROR
          - WARNING
          ...
          Encoding: utf-8
          PatternSet: my_pattern_set_2
        ...
        Alarms:
        - AlarmName: my_alarm_name
          Severity: HIGH
        ...
      SubComponentTypeConfigurations:
      - SubComponentType: EC2_INSTANCE
        SubComponentConfigurationDetails:
          AlarmMetrics:
          - AlarmMetricName: DiskReadOps
          ...
          Logs:
          - LogGroupName: my_log_group_2
            LogPath: C:\LogFolder_2\*
            LogType: IIS
            Encoding: utf-8
            PatternSet: my_pattern_set_3
          ...
          Processes:
          - ProcessName: my_process
            AlarmMetrics:
            - AlarmMetricName: procstat cpu_usage
              ...
            ...
          WindowsEvents:
          - LogGroupName: my_windows_event_log_group_2
            EventName: Application
            EventLevels:
            - ERROR
            - WARNING
            ...
            Encoding: utf-8
```

```
            PatternSet: my_pattern_set_4
          ...
```

DEFAULT 모드 구성 요소 구성으로 Application Insights 애플리케이션 생성

아래에 나와 있는 템플릿은 다음 작업을 수행합니다.

- Application Insights 애플리케이션을 생성합니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 CreateApplication을 참조하세요.
- 구성 요소 `my_component`에 대해 `ComponentConfigurationMode`를 `DEFAULT`로 설정하고 `Tier`를 `SQL_SERVER`로 설정합니다. 이렇게 하면 `SQL_Server` 계층에 대해 Application Insights에서 권장하는 구성 설정으로 이 구성 요소가 구성됩니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 DescribeComponentConfiguration 및 UpdateComponentConfiguration을 참조하세요.

JSON 형식의 템플릿

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "ComponentMonitoringSettings": [
            {
                "ComponentARN": "my_component",
                "Tier": "SQL_SERVER",
                "ComponentConfigurationMode": "DEFAULT"
            }
        ]
    }
}
```

YAML 형식의 템플릿

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
  - ComponentARN: my_component
    Tier: SQL_SERVER
```

```
    ComponentConfigurationMode: DEFAULT
```

DEFAULT_WITH_OVERWRITE 모드 구성 요소 구성으로 Application Insights 애플리케이션 생성

아래에 나와 있는 템플릿은 다음 작업을 수행합니다.

- Application Insights 애플리케이션을 생성합니다. 자세한 내용은 Amazon CloudWatch Application Insights API 참조의 CreateApplication을 참조하세요.
- 구성 요소 `my_component`에 대해 `ComponentConfigurationMode`를 `DEFAULT_WITH_OVERWRITE`로 설정하고 `tier`를 `DOT_NET_CORE`로 설정합니다. 이렇게 하면 `DOT_NET_CORE` 계층에 대해 Application Insights에서 권장하는 구성 설정으로 이 구성 요소가 구성됩니다. 덮어쓴 구성 설정은 `DefaultOverwriteComponentConfiguration`에서 지정됩니다.
  - 구성 요소 수준에서 `AlarmMetrics` 설정을 덮어씁니다.
  - 하위 구성 요소 수준에서 `EC2_Instance` 유형 하위 구성 요소에 대해 `Logs` 설정을 덮어씁니다.

  자세한 내용은Amazon CloudWatch Application Insights API 참조의 UpdateComponentConfiguration을 참조하세요.

JSON 형식의 템플릿

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "ComponentMonitoringSettings": [
            {
                "ComponentName": "my_component",
                "Tier": "DOT_NET_CORE",
                "ComponentConfigurationMode": "DEFAULT_WITH_OVERWRITE",
                "DefaultOverwriteComponentConfiguration": {
                    "ConfigurationDetails": {
                        "AlarmMetrics": [
                            {
                                "AlarmMetricName": "StatusCheckFailed"
                            }
                        ]
                    },
                    "SubComponentTypeConfigurations": [
                        {
```

```
                            "SubComponentType": "EC2_INSTANCE",
                            "SubComponentConfigurationDetails": {
                                "Logs": [
                                    {
                                        "LogGroupName": "my_log_group",
                                        "LogPath": "C:\\LogFolder\\*",
                                        "LogType": "IIS",
                                        "Encoding": "utf-8",
                                        "PatternSet": "my_pattern_set"
                                    }
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```

YAML 형식의 템플릿

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
  - ComponentName: my_component
    Tier: DOT_NET_CORE
    ComponentConfigurationMode: DEFAULT_WITH_OVERWRITE
    DefaultOverwriteComponentConfiguration:
      ConfigurationDetails:
        AlarmMetrics:
        - AlarmMetricName: StatusCheckFailed
      SubComponentTypeConfigurations:
      - SubComponentType: EC2_INSTANCE
        SubComponentConfigurationDetails:
          Logs:
          - LogGroupName: my_log_group
            LogPath: C:\LogFolder\*
            LogType: IIS
            Encoding: utf-8
            PatternSet: my_pattern_set
```

자습서: SAP ASE의 모니터링 설정

이 자습서에서는 SAP ASE 데이터베이스에 대한 모니터링을 설정하도록 CloudWatch Application Insights를 구성하는 방법을 보여줍니다. CloudWatch Application Insights 자동 대시보드를 사용하여 문제 세부 정보를 시각화하고, 문제 해결을 가속화하고, SAP ASE 데이터베이스의 평균 해결 시간(MTTR)을 단축할 수 있습니다.

SAP ASE 주제를 위한 Application Insights

- 지원 환경
- 지원되는 운영 체제
- 특성
- 필수 조건
- SAP ASE 데이터베이스에 대한 모니터링 설정
- SAP ASE 데이터베이스 모니터링 관리
- 경보 임곗값 구성
- Application Insights에서 감지한 SAP ASE 문제 확인 및 해결
- SAP ASE에 대한 Application Insights 문제 해결

지원 환경

CloudWatch Application Insights는 다음 시스템 및 패턴에 대한 AWS 리소스 배포를 지원합니다. SAP ASE 데이터베이스 소프트웨어 및 지원되는 SAP 애플리케이션 소프트웨어를 제공 및 설치합니다.

- 단일 Amazon EC2 인스턴스의 SAP ASE 데이터베이스 1개 이상 - 단일 노드 확장 아키텍처의 SAP ASE입니다.
- 크로스 AZ SAP ASE 데이터베이스 고가용성 설정 - SUSE/RHEL 클러스터링을 사용한 2개의 가용 영역에 구성된 고가용성 SAP ASE입니다.

> **Note**
>
> CloudWatch Application Insights는 단일 SAP 시스템 ID(SID) ASE HA 환경만 지원합니다. 여러 ASE HA SID가 연결된 경우, 첫 번째로 감지된 SID에 대해서만 모니터링이 설정됩니다.

지원되는 운영 체제

SAP ASE용 CloudWatch Application Insights는 다음 운영 체제에 대한 x86-64 아키텍처를 지원합니다.

- SuSE Linux 12 SP4
- SuSE Linux 12 SP5
- SuSE Linux 15
- SuSE Linux 15 SP1
- SuSE Linux 15 SP2
- SuSE Linux 15 SP3
- SuSE Linux 15 SP4
- SAP용 SuSE Linux 15 SP1
- SAP용 SuSE Linux 15 SP2
- SAP용 SuSE Linux 15 SP3
- SAP용 SuSE Linux 15 SP4
- SAP용 SuSE Linux 12 SP4
- SAP용 SuSE Linux 12 SP5
- RedHat Linux 7.6
- RedHat Linux 7.7
- RedHat Linux 7.9
- RedHat Linux 8.1
- RedHat Linux 8.4
- RedHat Linux 8.6

특성

SAP ASE용 CloudWatch Application Insights는 다음과 같은 기능을 제공합니다.

- SAP ASE 워크로드 자동 감지
- 정적 임곗값을 기반으로 한 SAP ASE 경보 자동 생성
- 이상 탐지를 기반으로 한 SAP ASE 경보 자동 생성

- SAP ASE 로그 패턴 자동 인식
- SAP ASE 상태 대시보드
- SAP ASE 문제 대시보드

필수 조건

CloudWatch Application Insights로 SAP ASE 데이터베이스를 구성하려면 다음 사전 조건을 수행해야 합니다.

- SAP ASE 구성 매개변수 - ASE DB에서 구성 매개변수("`enable monitoring`", "`sql text pipe max messages`", "`sql text pipe active`")를 활성화해야 합니다. 이를 통해 CloudWatch Application Insights는 DB에 대한 전체 모니터링 기능을 제공할 수 있습니다. ASE 데이터베이스에서 이러한 설정이 활성화되지 않은 경우 Application Insights는 모니터링을 허용하는 데 필요한 지표를 자동으로 수집할 수 있도록 합니다.
- SAP ASE 데이터베이스 사용자 - Application Insights 온보딩 중에 제공된 데이터베이스 사용자는 다음에 대한 액세스 권한이 있어야 합니다.
  - 마스터 데이터베이스 및 사용자(테넌트) 데이터베이스의 시스템 테이블
  - 모니터링 테이블
- SAPHostCtrl – Amazon EC2 인스턴스에서 SAPHostCtrl을 설치하고 설정합니다.
- Amazon CloudWatch 에이전트 - Amazon EC2 인스턴스에서 기존 CloudWatch 에이전트를 실행하고 있지 않은지 확인합니다. CloudWatch 에이전트트를 설치한 경우 병합 충돌을 방지하려면 Application Insights에서 사용 중인 리소스 구성을 기존 CloudWatch 에이전트 구성 파일에서 제거하세요. 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하십시오.
- AWS Systems Manager 활성화 - 인스턴스에 SSM Agent를 설치하고 SSM에 대해 인스턴스를 활성화합니다. SSM Agent를 설치하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서에서 SSM Agent 작업을 참조하세요.
- Amazon EC2 인스턴스 역할 - 데이터베이스를 구성하려면 다음 Amazon EC2 인스턴스 역할을 연결해야 합니다.
  - Systems Manager를 사용 설정하려면 `AmazonSSMManagedInstanceCore` 역할을 연결해야 합니다. 자세한 내용은 AWS Systems Manager 자격 증명 기반 정책 예제를 참조하세요.
  - CloudWatch를 통해 내보낼 인스턴스 지표 및 로그를 사용 설정하려면 `CloudWatchAgentServerPolicy`를 연결해야 합니다. 자세한 내용은 Amazon CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성을 참조하세요.

- AWS Secrets Manager에 저장된 암호를 읽으려면 다음 IAM 인라인 정책을 Amazon EC2 인스턴스 역할에 연결해야 합니다. 인라인 정책에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서에서 인라인 정책 을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
        }
    ]
}
```

- AWS Resource Groups – CloudWatch Application Insights에 애플리케이션을 온보딩하려면 애플리케이션 스택이 사용하는 연결된 모든 AWS 리소스를 포함하는 리소스 그룹을 생성해야 합니다. 여기에는 SAP ASE 데이터베이스를 실행하는 Amazon EC2 인스턴스와 Amazon EBS 볼륨이 포함됩니다. 계정당 데이터베이스가 여러 개인 경우, 각 SAP ASE 데이터베이스 시스템에 대한 AWS 리소스를 포함하는 하나의 리소스 그룹을 생성하는 것이 좋습니다.
- IAM 권한 - 관리자가 아닌 사용자의 경우:
  - 서비스 연결 역할을 생성하고 이를 사용자 자격 증명에 연결할 수 있도록 Application Insights를 허용하는 AWS Identity and Access Management (IAM) 정책을 생성해야 합니다. 정책 연결 단계에 대해서는 IAM 정책 섹션을 참조하세요.
  - 사용자가 데이터베이스 사용자 자격 증명을 저장하려면 AWS Secrets Manager에 보안 암호를 생성할 수 있는 권한이 있어야 합니다. 자세한 내용은 예: 보안 암호 생성 권한을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
    }
```

```
  ]
}
```

- 서비스 연결 역할 - Application Insights는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Application Insights 관리 콘솔에서 첫 번째 Application Insights 애플리케이션을 생성할 때 생성됩니다. 자세한 내용은 CloudWatch Application Insights에 서비스 연결 역할 사용 단원을 참조하십시오.

SAP ASE 데이터베이스에 대한 모니터링 설정

SAP ASE 데이터베이스에 대한 모니터링을 설정하려면 다음 단계를 수행하세요

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 인사이트(Insights)에서 Application Insights를 선택합니다.
3. Application Insights 페이지에는 Application Insights를 사용하여 모니터링되는 애플리케이션 목록과 각 애플리케이션의 모니터링 상태가 표시됩니다. 오른쪽 상단 모서리에서 애플리케이션 추가(Add an application)를 선택합니다.
4. 애플리케이션 세부 정보 지정 페이지에서 리소스 그룹의 드롭다운 목록에 있는 SAP ASE 데이터베이스 리소스를 포함하는 AWS 리소스 그룹을 선택합니다. 애플리케이션에 대한 리소스 그룹을 생성하지 않은 경우 리소스 그룹(Resource group) 드롭다운에서 새 리소스 그룹 생성(Create new resource group)을 선택해 생성할 수 있습니다. 리소스 그룹 생성에 대한 자세한 내용은 AWS Resource Groups 사용 설명서를 참조하세요.
5. Amazon EBS, Amazon EC2, AWS CodeDeploy, Amazon ECS, AWS Health API 및 알림, Amazon RDS, Amazon S3, AWS Step Functions로부터 인사이트를 얻기 위해 Application Insights 모니터링을 CloudWatch Events와 통합하려면 CloudWatch Events 모니터링(Monitor CloudWatch Events)에서 확인란을 선택합니다.
6. 선택한 애플리케이션에 문제가 감지될 때 이를 확인하고 알림을 받으려면 Integrate with AWS Systems Manager OpsCenter(AWS Systems Manager OpsCenter와 통합)에서 Generate OpsCenter OpsItems for remedial actions(수정 조치를 위해 OpsCenter OpsItems 생성) 옆의 확인란을 선택합니다. AWS 리소스와 관련된 운영 작업 항목(OpsItem)을 해결하기 위해 수행된 작업을 추적하려면 SNS 주제 ARN을 제공합니다.
7. 태그를 입력하면(선택 사항) 리소스를 식별하고 구성하는 데 도움이 됩니다. CloudWatch Application Insights는 태그 기반 및 AWS CloudFormation 스택 기반 리소스 그룹을 모두 지원합니다(Application Auto Scaling 그룹 제외). 자세한 내용은 AWS Resource Groups 및 태그 사용 설명서에서 Tag Editor를 참조하세요.
8. 다음(Next)을 선택해 모니터링을 계속 설정합니다.

9. 탐지된 구성 요소 검토 페이지에는 CloudWatch 애플리케이션 인사이트에서 자동으로 탐지된 모니터링되는 구성 요소와 해당 워크로드가 나열됩니다.

> **Note**
>
> 탐지된 SAP ASE 고가용성 워크로드가 포함된 구성 요소는 구성 요소에서 하나의 워크로드만 지원합니다. 탐지된 SAP ASE 단일 노드 워크로드가 포함된 구성 요소는 여러 워크로드를 지원하지만 워크로드를 추가하거나 제거할 수는 없습니다. 자동으로 탐지된 모든 워크로드가 모니터링됩니다.

10. 다음을 선택합니다.
11. 구성 요소 세부 정보 지정 페이지에서 SAP ASE 데이터베이스의 사용자 이름과 암호를 입력합니다.
12. 애플리케이션 모니터링 구성을 검토하고 제출을 선택합니다.
13. 애플리케이션 요약, 모니터링되는 구성 요소 및 워크로드와 모니터링되지 않는 구성 요소 및 워크로드 목록을 확인할 수 있는 애플리케이션 세부 정보 페이지가 열립니다. 구성 요소 또는 워크로드 옆에 있는 라디오 버튼을 선택하면 구성 기록, 로그 패턴, 사용자가 생성한 모든 태그를 확인할 수 있습니다. 구성을 제출하면 계정에서 SAP ASE 시스템에 대한 모든 지표 및 경보를 배포하며 최대 2시간이 걸릴 수 있습니다.

SAP ASE 데이터베이스 모니터링 관리

다음 단계를 수행해 SAP ASE 데이터베이스에 대한 사용자 자격 증명, 지표 및 로그 경로를 관리할 수 있습니다.

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 인사이트(Insights)에서 Application Insights를 선택합니다.
3. Application Insights 페이지에는 Application Insights를 사용하여 모니터링되는 애플리케이션 목록과 각 애플리케이션의 모니터링 상태가 표시됩니다.
4. 모니터링되는 구성 요소(Monitored components)에서 구성 요소 이름 옆에 있는 라디오 버튼을 선택합니다. 다음으로 모니터링 관리(Manage monitoring)를 선택합니다.
5. EC2 인스턴스 그룹 로그(EC2 instance group logs)에서 기존 로그 경로, 로그 패턴 세트 및 로그 그룹 이름을 업데이트할 수 있습니다. 또한 애플리케이션 로그(Application logs)를 3개까지 추가할 수 있습니다.
6. 지표에서 요구 사항에 따라 SAP ASE 지표를 선택할 수 있습니다. SAP ASE 지표 이름에는 `asedb` 접두사가 붙습니다. 구성 요소당 60개까지 지표를 추가할 수 있습니다.

7. ASE 구성에서 SAP ASE 데이터베이스의 사용자 이름과 암호를 입력합니다. Amazon CloudWatch 에이전트가 SAP ASE 데이터베이스에 연결하는 데 사용하는 사용자 이름과 암호입니다.
8. 사용자 지정 경보(Custom alarms)에서 CloudWatch Application Insights를 통해 모니터링할 경보를 추가할 수 있습니다.
9. 애플리케이션 모니터링 구성을 검토하고 제출(Submit)을 선택합니다. 구성을 제출하면 계정에서 SAP HANA 시스템에 대한 모든 지표 및 경보를 업데이트하며 최대 2시간이 걸릴 수 있습니다.

경보 임곗값 구성

CloudWatch Application Insights는 감시하는 경보에 대한 Amazon CloudWatch 지표를 해당 지표에 대한 임곗값과 함께 자동으로 생성합니다. 지표가 평가 기간에 지정된 수에 대한 임곗값을 초과한 경우 경보는 `ALARM` 상태로 변경됩니다. 이러한 설정은 Application Insights에 의해 유지되지 않습니다.

단일 지표에 대한 경보를 편집하려면 다음 단계를 수행하세요.

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 경보(Alarms) > 모든 경보(All alarms)를 선택합니다.
3. CloudWatch Application Insights에서 자동으로 생성된 경보 옆에 있는 라디오 버튼을 선택합니다. 그런 다음, 작업(Actions)을 선택하고 드롭다운 메뉴에서 편집(Edit)을 선택합니다.
4. 지표(Metric)에서 다음 파라미터를 편집합니다.
   a. 통계(Statistic)에서 통계 또는 사전 정의된 백분위 수 중 하나를 선택하거나 사용자 지정 백분위 수를 지정합니다. 예를 들면 `p95.45`입니다.
   b. 기간(Period)에서 경보에 대한 평가 기간을 선택합니다. 경보를 평가할 때 각 기간이 하나의 데이터 포인트로 집계됩니다.
5. 조건(Conditions)에서 다음 파라미터를 편집합니다.
   a. 지표가 임곗값보다 크거나, 작거나, 같아야 하는지 여부를 선택합니다.
   b. 임곗값을 지정합니다.
6. 추가 구성(Additional configuration)에서 다음 파라미터를 편집합니다.
   a. 경보에 대한 데이터 포인트(Datapoints to alarm)에서 데이터 포인트 수 또는 평가 기간을 지정합니다. 경보를 시작하려면 `ALARM` 상태여야 합니다. 두 값이 일치하면 지정된 연속 기간 수를 초과한 경우 `ALARM` 상태가 되는 경보가 생성됩니다. n 중 `m` 경보를 생성하려면 두 번

째 값에 지정한 값보다 낮은 값을 첫 번째 값에 지정합니다. 경보 평가에 관한 자세한 내용은 Evaluating an alarm을 참조하세요.

b. 누락 데이터 처리에서 일부 데이터 포인트가 누락된 경우 경보 동작을 선택합니다. 누락 데이터 처리에 대한 자세한 내용은 CloudWatch 경보가 누락 데이터를 처리하는 방법 구성을 참조하세요.

c. 경보가 모니터링된 통계 값으로 백분위수를 사용하는 경우에는 샘플이 부족한 백분위수 상자가 표시됩니다. 샘플 비율이 낮은 사례를 평가 또는 무시할지 여부를 선택합니다. ignore (maintain alarm state)(무시(경보 상태 유지))를 선택하면 샘플 크기가 너무 작을 때 현재 경보 상태가 항상 유지됩니다. 작은 샘플 크기의 백분위 수에 대한 자세한 내용은 백분위수 기반 CloudWatch 경보 및 데이터 샘플 부족 섹션을 참조하세요.

7. 다음을 선택합니다.
8. 알림(Notification)에서 경보가 `ALARM` 상태, `OK` 상태 또는 `INSUFFICIENT_DATA` 상태일 때 알릴 SNS 주제를 선택합니다.
9. 경보 업데이트(Update alarm)을 선택합니다.

Application Insights에서 감지한 SAP ASE 문제 확인 및 해결

이 섹션에서는 Application Insights에서 SAP ASE에 대한 모니터링을 구성할 때 발생하는 일반적인 문제를 해결하는 데 도움이 됩니다.

SAP ASE 백업 서버 오류

동적으로 생성된 대시보드를 확인하여 오류 메시지를 식별할 수 있습니다. 대시보드에는 SAP ASE 백업 서버에 보고된 오류 메시지가 표시됩니다. SAP ASE 백업 서버 로그에 대한 자세한 내용은 SAP 설명서 백업 서버 오류 로깅을 참조하세요.

SAP ASE 장기 실행 트랜잭션

장기 실행 트랜잭션을 식별하고 중지할 수 있는지 또는 실행 시간이 의도적으로 발생했는지 확인합니다. 자세한 내용은 2180410 - 장기 실행 트랜잭션의 트랜잭션 로그 기록을 표시하는 방법을 참조하세요 — SAP ASE

SAP ASE 사용자 연결

데이터베이스에서 실행하려는 워크로드에 맞게 SAP ASE 데이터베이스 크기가 조정되었는지 검토합니다. 자세한 내용은 SAP 설명서의 사용자 연결 구성을 참조하세요.

SAP ASE 디스크 공간

동적으로 생성된 대시보드를 확인하여 문제를 일으키는 데이터베이스 계층을 식별할 수 있습니다. 대시보드에는 관련 지표와 로그 파일 스니펫이 표시됩니다. 디스크 증가의 원인을 파악하고 해당하는 경우 물리적 디스크 크기, 할당된 디스크 공간 또는 둘 다를 늘리는 것이 중요합니다. 자세한 내용은 SAP 설명서의 SAP 설명서 디스크 크기 조정을 참조하세요.

SAP ASE에 대한 Application Insights 문제 해결

이 섹션에서는 Application Insights 대시보드에서 반환하는 일반적인 오류를 해결하는 데 도움이 되는 단계를 제공합니다.

Error	반환된 오류	근본 원인	해결 방법
60개 이상의 모니터 지표를 추가할 수 없습니다.	`Component cannot have more than 60 monitored metric`	현재 지표 제한은 구성 요소당 모니터링 지표 60개입니다.	한도를 준수하려면 불필요한 지표를 제거합니다.
온보딩 프로세스 후에는 SAP 지표 또는 경보가 표시되지 않습니다	AWS Systems Manager에서 `AWS-ConfigureAWSPackage` 에 대한 `run` 명령이 실패했습니다. 출력값은 다음 오류를 보여줍니다. `CT-LIBRARY error:ct_connect(): protocol specific layer: external error: The attempt to connect to the server failed`	사용자 이름과 암호가 올바르지 않을 수 있습니다.	사용자 이름과 암호가 유효한지 확인하고 온보딩 프로세스를 다시 실행합니다.

자습서: SAP HANA에 대한 모니터링 설정

이 자습서에서는 SAP HANA 데이터베이스에 대한 모니터링을 설정하도록 CloudWatch Application Insights를 구성하는 방법을 보여줍니다. CloudWatch Application Insights 자동 대시보드를 사용하여

문제 세부 정보를 시각화하고, 문제 해결을 가속화하고, SAP HANA 데이터베이스의 평균 해결 시간(MTTR)을 단축할 수 있습니다.

SAP HANA 주제를 위한 Application Insights

지원 환경

CloudWatch Application Insights는 다음 시스템 및 패턴에 대한 AWS 리소스 배포를 지원합니다. SAP HANA 데이터베이스 소프트웨어 및 지원되는 SAP 애플리케이션 소프트웨어를 제공 및 설치합니다.

- 단일 Amazon EC2 인스턴스의 SAP HANA 데이터베이스: 최대 24TB 메모리를 갖춘 단일 노드 확장 아키텍처의 SAP HANA.
- 여러 Amazon EC2 인스턴스의 SAP HANA 데이터베이스: 다중 노드 확장 아키텍처의 SAP HANA
- 교차 AZ SAP HANA 데이터베이스 고가용성 설정: SUSE/RHEL 클러스터링을 사용한 2개의 가용 영역에 구성된 고가용성 SAP HANA

> **Note**
>
> CloudWatch Application Insights는 단일 SID HANA 환경만 지원합니다. 여러 HANA SID가 연결된 경우, 첫 번째로 감지된 SID에 대해서만 모니터링이 설정됩니다.

지원되는 운영 체제

SAP HANA용 CloudWatch Application Insights는 다음 운영 체제에 대한 x86-64 아키텍처를 지원합니다.

- SAP용 SuSE Linux 12 SP4
- SAP용 SuSE Linux 12 SP5
- SuSE Linux 15
- SuSE Linux 15 SP1
- SuSE Linux 15 SP2
- SAP용 SuSE Linux 15
- SAP용 SuSE Linux 15 SP1
- SAP용 SuSE Linux 15 SP2
- SAP용 SuSE Linux 15 SP3
- SAP용 SuSE Linux 15 SP4
- SAP용 SuSE Linux 15 SP5
- 고가용성 및 업데이트 서비스를 제공하는 SAP용 RedHat 리눅스 8.6
- 고가용성 및 업데이트 서비스를 제공하는 SAP용 RedHat 리눅스 8.5
- 고가용성 및 업데이트 서비스를 제공하는 SAP용 RedHat 리눅스 8.4
- 고가용성 및 업데이트 서비스를 제공하는 SAP용 RedHat 리눅스 8.3
- 고가용성 및 서비스 업데이트를 제공하는 SAP용 RedHat 리눅스 8.2
- 고가용성 및 서비스 업데이트를 제공하는 SAP용 RedHat 리눅스 8.1
- 고가용성 및 서비스 업데이트를 제공하는 SAP용 RedHat 리눅스 7.9

특성

SAP HANA용 CloudWatch Application Insights는 다음과 같은 기능을 제공합니다.

- SAP HANA 워크로드 자동 감지
- 정적 임곗값을 기반으로 한 SAP HANA 경보 자동 생성
- 이상 탐지를 기반으로 한 SAP HANA 경보 자동 생성
- SAP HANA 로그 패턴 자동 인식
- SAP HANA 상태 대시보드
- SAP HANA 문제 대시보드

필수 조건

CloudWatch Application Insights로 SAP HANA 데이터베이스를 구성하려면 다음 사전 조건을 수행해야 합니다.

- SAP HANA - 실행 및 연결 가능한 SAP HANA 데이터베이스 2.0 SPS05를 Amazon EC2 인스턴스에 설치합니다.
- SAP HANA 데이터베이스 사용자 - SYSTEM 데이터베이스 및 모든 테넌트에서 모니터링 역할이 있는 데이터베이스 사용자를 만들어야 합니다.

  예

  다음 SQL 명령을 실행하여 모니터링 역할이 있는 사용자를 생성합니다.

```
su - <sid>adm
hdbsql -u SYSTEM -p <SYSTEMDB password> -d SYSTEMDB
CREATE USER CW_HANADB_EXPORTER_USER PASSWORD <Monitoring user password> NO
 FORCE_FIRST_PASSWORD_CHANGE;
CREATE ROLE CW_HANADB_EXPORTER_ROLE;
GRANT MONITORING TO CW_HANADB_EXPORTER_ROLE;
GRANT CW_HANADB_EXPORTER_ROLE TO CW_HANADB_EXPORTER_USER;
```

- Python 3.8 - 운영 체제에 Python 3.8 이상 버전을 설치합니다. 최신 릴리스의 Python을 사용하세요. 운영 체제에 Python3이 감지되지 않으면 Python 3.6이 설치됩니다.

  자세한 내용은 installation example 단원을 참조하십시오.

  > **Note**
  >
  > SuSE Linux 15 SP4, RedHat Linux 8.6 이상 운영 체제에서는 Python 3.8 이상을 수동으로 설치해야 합니다.

- Pip3 - 운영 체제에 설치 프로그램 pip3를 설치합니다. 운영 체제에 pip3가 감지되지 않으면 pip3가 설치됩니다.
- hdbclient - CloudWatch Application Insights는 Python 드라이버를 사용하여 SAP HANA 데이터베이스에 연결합니다. 클라이언트가 python3에 설치되어 있지 않은 경우 `/hana/shared/SID/hdbclient/`에서 hdbclient tar 파일 버전이 `2.10 or later`인지 확인합니다.
- Amazon CloudWatch 에이전트 - Amazon EC2 인스턴스에서 기존 CloudWatch 에이전트를 실행하고 있지 않은지 확인합니다. CloudWatch 에이전트트를 설치한 경우 병합 충돌을 방지하려면

Application Insights에서 사용 중인 리소스 구성을 기존 CloudWatch 에이전트 구성 파일에서 제거하세요. 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하십시오.

- AWS Systems Manager 사용 설정 - 인스턴스에 SSM Agent를 설치하고 인스턴스에서 SSM을 활성화해야 합니다. SSM Agent를 설치하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서에서 SSM Agent 작업을 참조하세요.
- Amazon EC2 인스턴스 역할 - 데이터베이스를 구성하려면 다음 Amazon EC2 인스턴스 역할을 연결해야 합니다.
  - Systems Manager를 사용 설정하려면 `AmazonSSMManagedInstanceCore` 역할을 연결해야 합니다. 자세한 내용은 AWS Systems Manager 자격 증명 기반 정책 예제를 참조하세요.
  - CloudWatch를 통해 내보낼 인스턴스 지표 및 로그를 사용 설정하려면 `CloudWatchAgentServerPolicy`를 연결해야 합니다. 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성을 참조하세요.
  - AWS Secrets Manager에 저장된 암호를 읽으려면 다음 IAM 인라인 정책을 Amazon EC2 인스턴스 역할에 연결해야 합니다. 인라인 정책에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서에서 인라인 정책 을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
        }
    ]
}
```

- AWS Resource Groups - CloudWatch Application Insights에 애플리케이션을 온보딩하려면 애플리케이션 스택에 사용되는 연결된 모든 AWS 리소스를 포함하는 리소스 그룹을 생성해야 합니다. 여기에는 SAP HANA 데이터베이스를 실행하는 Amazon EC2 인스턴스와 Amazon EBS 볼륨이 포함됩니다. 계정당 데이터베이스가 여러 개인 경우, 각 SAP HANA 데이터베이스 시스템에 대한 AWS 리소스를 포함하는 하나의 리소스 그룹을 생성하는 것이 좋습니다.
- IAM 권한 - 관리자가 아닌 사용자의 경우:

- 서비스 연결 역할을 생성하고 이를 사용자 자격 증명에 연결할 수 있도록 Application Insights를 허용하는 AWS Identity and Access Management (IAM) 정책을 생성해야 합니다. 정책 연결 단계에 대해서는 IAM 정책 섹션을 참조하세요.
- 사용자가 데이터베이스 사용자 자격 증명을 저장하려면 AWS Secrets Manager에 보안 암호를 생성할 수 있는 권한이 있어야 합니다. 자세한 내용은 예: 보안 암호 생성 권한을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
    }
  ]
}
```

- 서비스 연결 역할 - Application Insights는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Application Insights 관리 콘솔에서 첫 번째 Application Insights 애플리케이션을 생성할 때 생성됩니다. 자세한 내용은 CloudWatch Application Insights에 서비스 연결 역할 사용 단원을 참조하십시오.

모니터링을 위해 SAP HANA 데이터베이스 설정

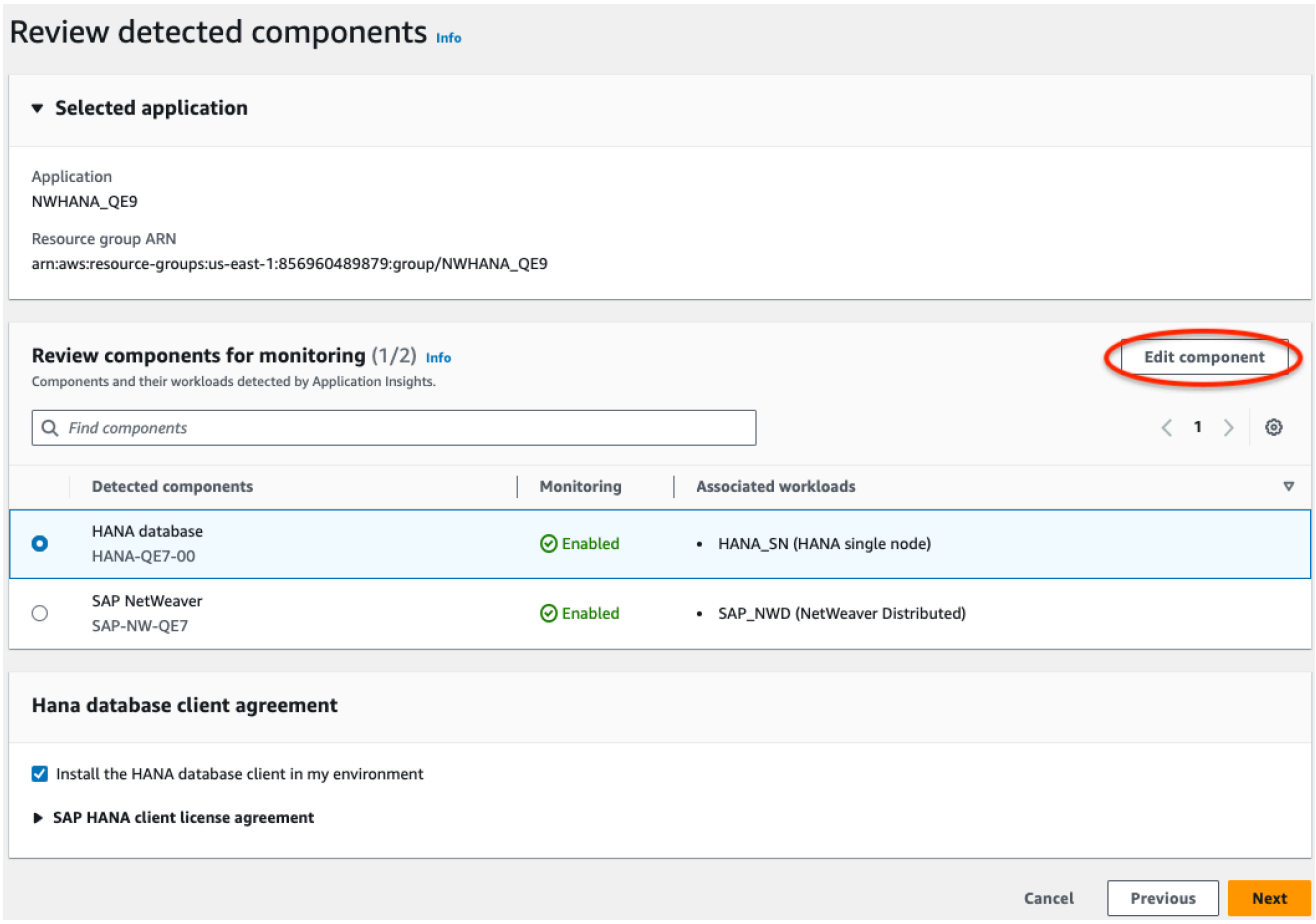
SAP HANA 데이터베이스에 대한 모니터링을 설정하려면 다음 단계를 수행하세요.

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 인사이트(Insights)에서 Application Insights를 선택합니다.
3. Application Insights 페이지에는 Application Insights를 사용하여 모니터링되는 애플리케이션 목록과 각 애플리케이션의 모니터링 상태가 표시됩니다. 오른쪽 상단 모서리에서 애플리케이션 추가(Add an application)를 선택합니다.
4. 애플리케이션 세부 정보 지정(Specify application details) 페이지에서 리소스 그룹(Resource group)의 드롭다운 목록에 있는 SAP HANA 데이터베이스 리소스를 포함하는 AWS 리소스 그룹을 선택합니다. 애플리케이션에 대한 리소스 그룹을 생성하지 않은 경우 리소스 그룹(Resource group) 드롭다운에서 새 리소스 그룹 생성(Create new resource group)을 선택해 생성할 수 있습니다. 리소스 그룹 생성에 대한 자세한 내용은 AWS Resource Groups 사용 설명서를 참조하세요.

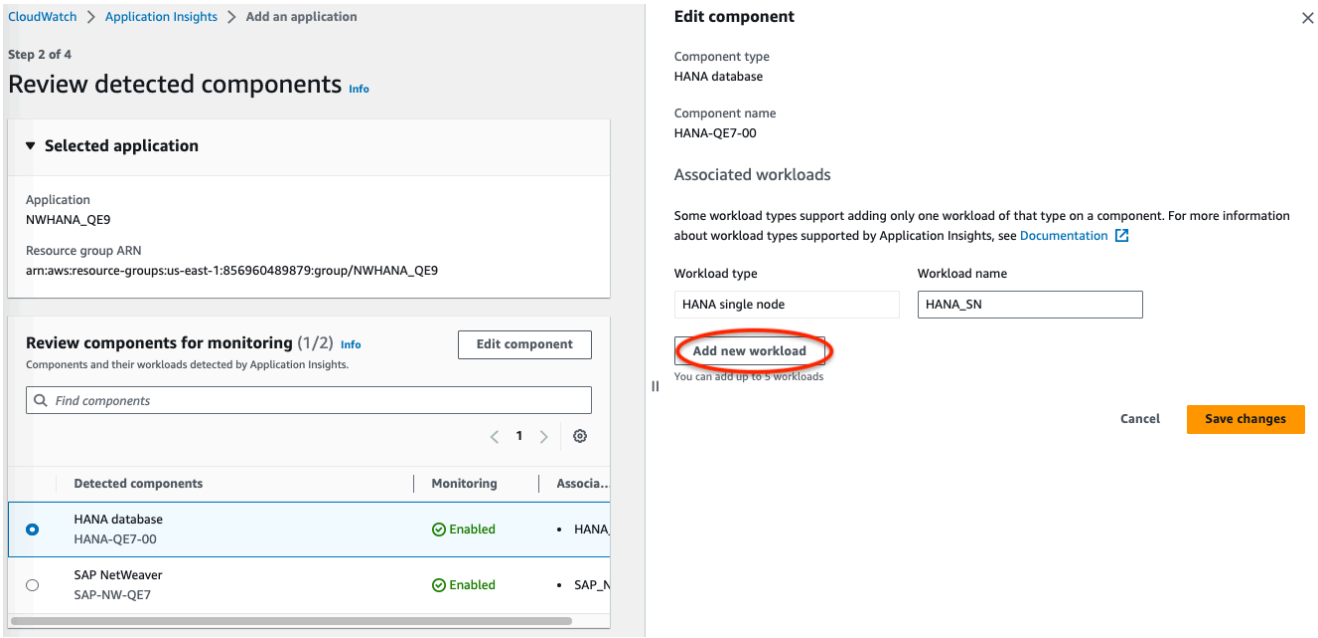
5. Amazon EBS, Amazon EC2, AWS CodeDeploy, Amazon ECS, AWS Health API 및 알림, Amazon RDS, Amazon S3, AWS Step Functions로부터 인사이트를 얻기 위해 Application Insights 모니터링을 CloudWatch Events와 통합하려면 CloudWatch Events 모니터링(Monitor CloudWatch Events)에서 확인란을 선택합니다.
6. 선택한 애플리케이션에 문제가 감지될 때 이를 확인하고 알림을 받으려면 Integrate with AWS Systems Manager OpsCenter(AWS Systems Manager OpsCenter와 통합)에서 Generate OpsCenter OpsItems for remedial actions(수정 조치를 위해 OpsCenter OpsItems 생성) 옆의 확인란을 선택합니다. AWS 리소스와 관련된 운영 작업 항목(OpsItem)을 해결하기 위해 수행된 작업을 추적하려면 SNS 주제 ARN을 제공합니다.
7. 태그를 입력하면(선택 사항) 리소스를 식별하고 구성하는 데 도움이 됩니다. CloudWatch Application Insights는 태그 기반 및 AWS CloudFormation 스택 기반 리소스 그룹을 모두 지원합니다(Application Auto Scaling 그룹 제외). 자세한 내용은 AWS Resource Groups 및 태그 사용 설명서에서 Tag Editor를 참조하세요.
8. 다음(Next)을 선택해 모니터링을 계속 설정합니다.
9. 탐지된 구성 요소 검토 페이지에는 CloudWatch 애플리케이션 인사이트에서 자동으로 탐지된 모니터링되는 구성 요소와 해당 워크로드가 나열됩니다.

    a. 탐지된 SAP HANA 단일 노드 워크로드가 포함된 구성 요소에 워크로드를 추가하려면 구성 요소를 선택한 다음 구성 요소 편집을 선택합니다.

> **Note**
>
> 탐지된 SAP HANA 다중 노드 또는 HANA 고가용성 워크로드가 포함된 구성 요소는 구성 요소에서 하나의 워크로드만 지원합니다.



b. 새 워크로드를 추가하려면 새 워크로드 추가를 선택합니다.



c. 워크로드 편집을 마쳤으면 변경 사항 저장을 선택합니다.

10. 다음을 선택합니다.
11. 구성 요소 세부 정보 지정 페이지에서 사용자 이름과 암호를 입력합니다.
12. 애플리케이션 모니터링 구성을 검토하고 제출을 선택합니다.
13. 애플리케이션 요약, 모니터링되는 구성 요소 및 워크로드와 모니터링되지 않는 구성 요소 및 워크로드 목록을 확인할 수 있는 애플리케이션 세부 정보 페이지가 열립니다. 구성 요소 또는 워크로드 옆에 있는 라디오 버튼을 선택하면 구성 기록, 로그 패턴, 사용자가 생성한 모든 태그를 확인할 수 있습니다. 구성을 제출하면 계정에서 SAP HANA 시스템에 대한 모든 지표 및 경보를 배포하며 최대 2시간이 걸릴 수 있습니다.

SAP HANA 데이터베이스 모니터링 관리

다음 단계를 수행해 SAP HANA 데이터베이스에 대한 사용자 자격 증명, 지표 및 로그 경로를 관리할 수 있습니다.

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 인사이트(Insights)에서 Application Insights를 선택합니다.
3. Application Insights 페이지에는 Application Insights를 사용하여 모니터링되는 애플리케이션 목록과 각 애플리케이션의 모니터링 상태가 표시됩니다.
4. 모니터링되는 구성 요소(Monitored components)에서 구성 요소 이름 옆에 있는 라디오 버튼을 선택합니다. 다음으로 모니터링 관리(Manage monitoring)를 선택합니다.
5. EC2 인스턴스 그룹 로그(EC2 instance group logs)에서 기존 로그 경로, 로그 패턴 세트 및 로그 그룹 이름을 업데이트할 수 있습니다. 또한 애플리케이션 로그(Application logs)를 3개까지 추가할 수 있습니다.
6. 지표(Metrics)에서 요구 사항에 따라 SAP HANA 지표를 선택할 수 있습니다. SAP HANA 지표 이름에는 hanadb 접두사가 붙습니다. 구성 요소당 40개까지 지표를 추가할 수 있습니다.
7. HANA 구성(HANA configuration)에서 SAP HANA 데이터베이스의 암호와 사용자 이름을 입력합니다. Amazon CloudWatch 에이전트가 SAP HANA 데이터베이스에 연결하는 데 사용하는 사용자 이름 및 암호입니다.
8. 사용자 지정 경보(Custom alarms)에서 CloudWatch Application Insights를 통해 모니터링할 경보를 추가할 수 있습니다.
9. 애플리케이션 모니터링 구성을 검토하고 제출(Submit)을 선택합니다. 구성을 제출하면 계정에서 SAP HANA 시스템에 대한 모든 지표 및 경보를 업데이트하며 최대 2시간이 걸릴 수 있습니다.

CloudWatch Application Insights에서 감지한 SAP HANA 문제 보기 및 해결

다음 섹션에서는 Application Insights에서 SAP HANA에 대한 모니터링을 구성할 때 발생하는 일반적인 문제 해결 시나리오를 해결하는 데 도움이 되는 단계를 제공합니다.

주제 문제 해결

- SAP HANA 데이터베이스가 메모리 할당 한도에 도달
- 디스크 가득 참 이벤트
- SAP HANA 백업 실행 중지

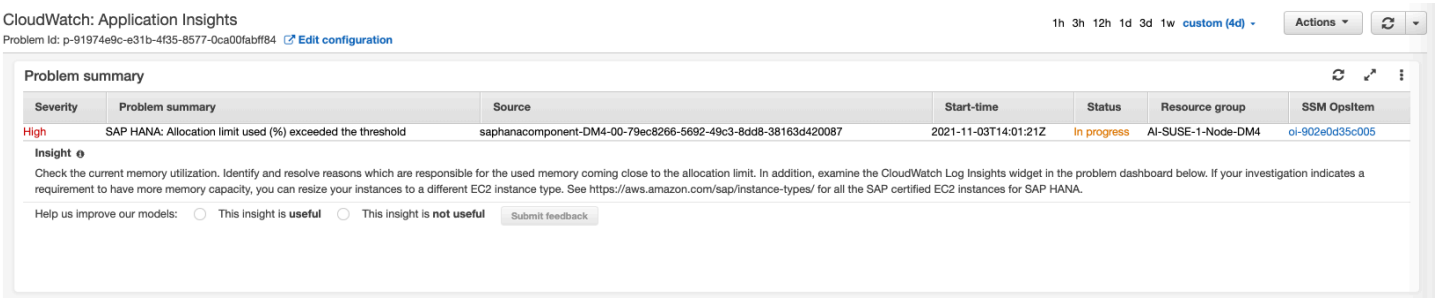
SAP HANA 데이터베이스가 메모리 할당 한도에 도달

설명

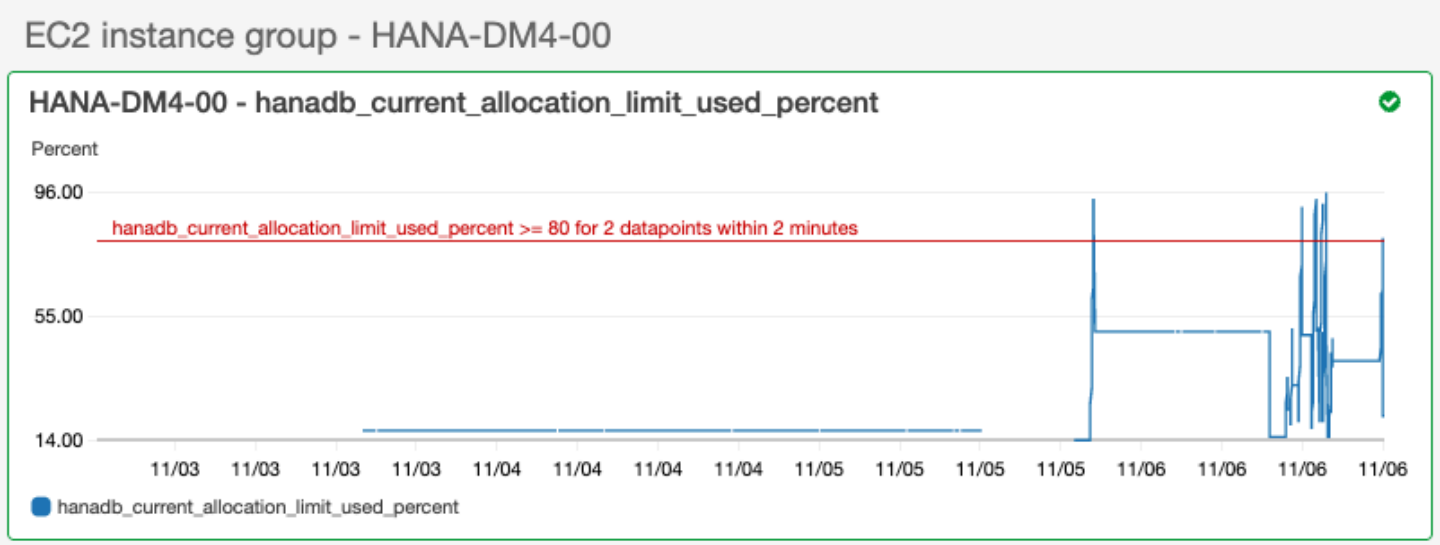
메모리 압력이 높으면 SAP HANA 데이터베이스가 지원하는 SAP 애플리케이션이 제대로 작동하지 않고 이는 애플리케이션 성능 저하로 이어집니다.

해결 방법

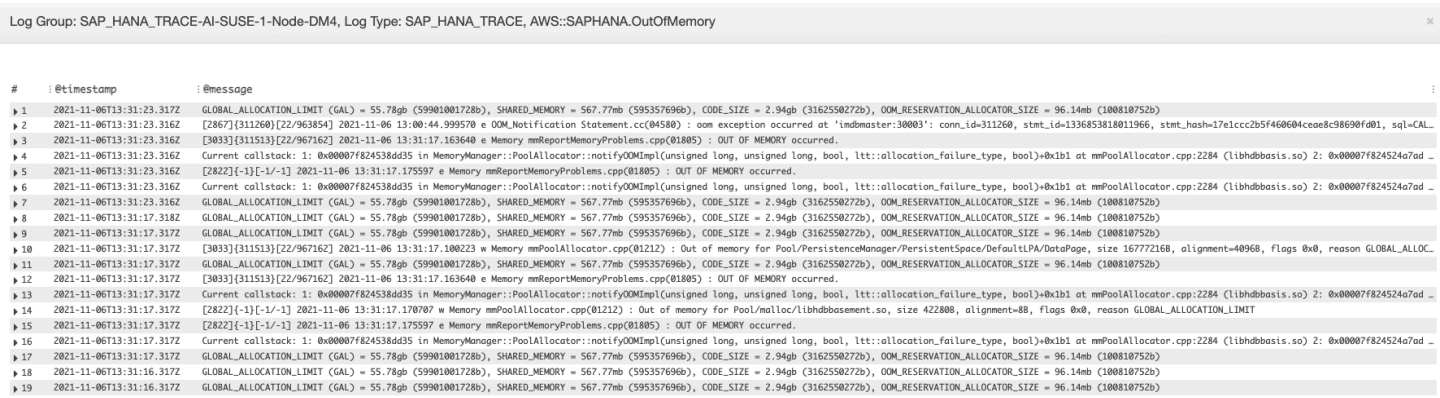
동적으로 생성된 대시보드에서 관련 지표 및 로그 파일 조각을 확인하여 문제를 일으키는 애플리케이션 계층을 식별할 수 있습니다. 다음 예제를 보면 문제는 SAP HANA 시스템의 큰 데이터 로드 때문일 수 있습니다.



사용된 메모리 할당이 총 메모리 할당 제한의 80% 임곗값을 초과합니다.



로그 그룹에 메모리 부족이 발생한 BNR-DATA 체계 및 IMDBMASTER_30003 테이블이 표시됩니다. 또한 로그 그룹에는 문제가 발생한 정확한 시간, 현재 글로벌 위치 제한, 공유 메모리, 코드 크기 및 OOM 예약 할당 크기가 표시됩니다.



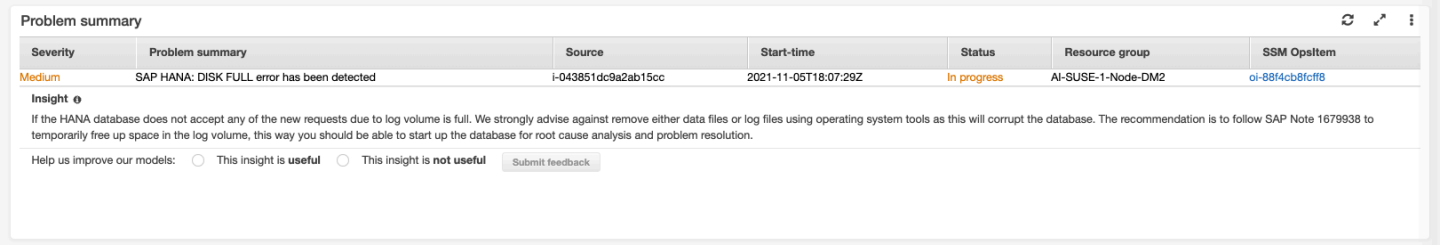
디스크 가득 참 이벤트

설명

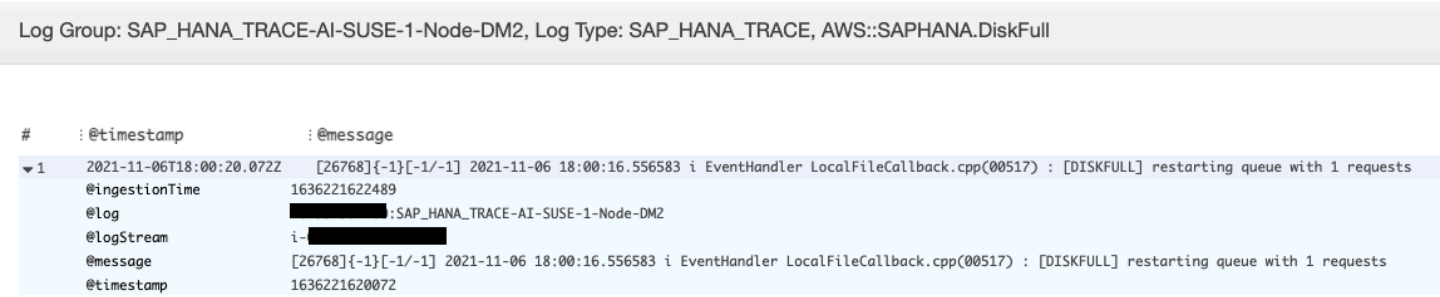
SAP HANA 데이터베이스가 지원하는 SAP 애플리케이션이 응답하지 않아 데이터베이스에 액세스할 수 없게 됩니다.

해결 방법

동적으로 생성된 대시보드에서 관련 지표 및 로그 파일 조각을 확인하여 문제를 일으키는 데이터베이스 계층을 식별할 수 있습니다. 다음 예를 보면 관리자가 자동 로그 백업을 활성화하지 못해서 sap/hana/log 디렉터리가 채워지는 문제가 발생할 수 있습니다.



문제 대시보드의 로그 그룹 위젯에 DISKFULL 이벤트를 표시합니다.



SAP HANA 백업 실행 중지

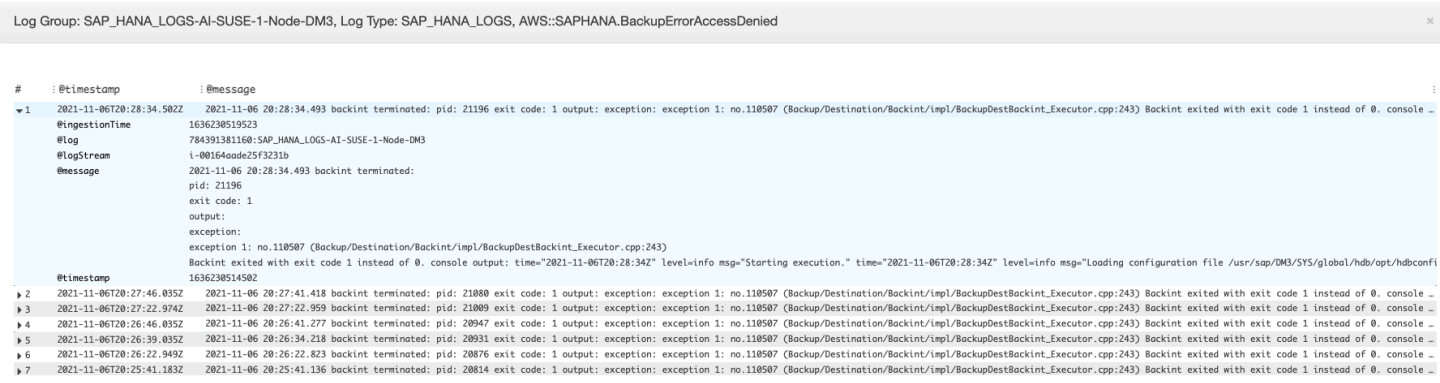
설명

SAP HANA 데이터베이스가 지원하는 SAP 애플리케이션의 작동이 중지되었습니다.

해결 방법

동적으로 생성된 대시보드에서 관련 지표 및 로그 파일 조각을 확인하여 문제를 일으키는 데이터베이스 계층을 식별할 수 있습니다.

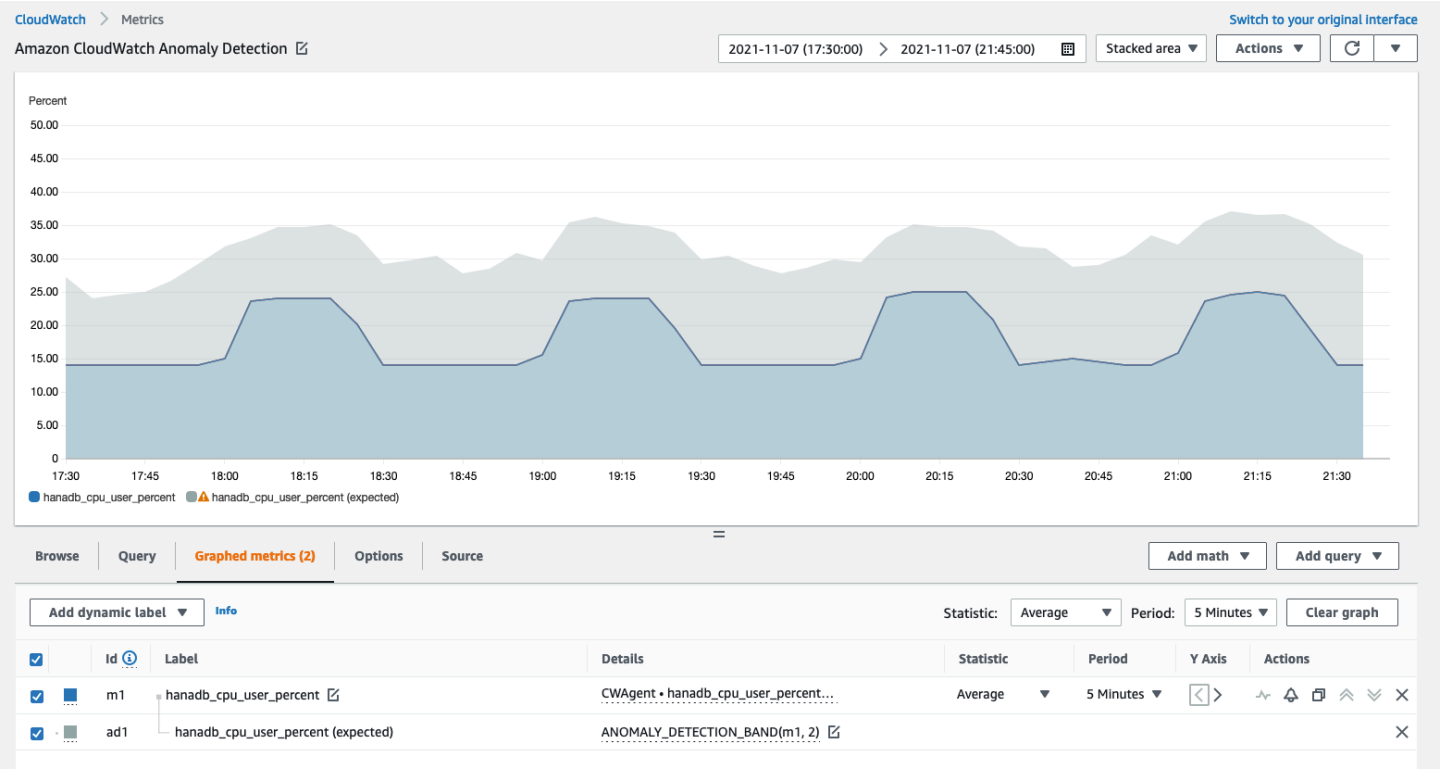
문제 대시보드의 로그 그룹 위젯에 ACCESS DENIED 이벤트를 표시합니다. 여기에는 S3 버킷, S3 버킷 폴더 및 S3 버킷 리전 등의 추가 정보가 포함됩니다.



SAP HANA를 위한 이상 탐지

스레드 수와 같은 특정 SAP HANA 지표의 경우 CloudWatch는 통계 및 기계 학습 알고리즘을 적용해 임곗값을 정의합니다. 이러한 알고리즘은 SAP HANA 데이터베이스 지표를 지속적으로 분석하고, 정상 기준을 결정하며, 최소한의 사용자 개입으로 이상을 나타냅니다. 알고리즘은 이상 탐지 모델을 생성해 정상 지표 동작을 나타내는 예상 값 범위를 생성합니다.

이상 탐지 알고리즘은 지표의 계절성 및 추세 변화를 설명합니다. 계절성 변화는 다음 SAP HANA CPU 사용량 예제에서 볼 수 있듯이 시간별, 일별 또는 주별일 수 있습니다.



모델을 생성한 후 CloudWatch 이상 탐지는 지속적으로 모델을 평가하며 모델이 가능한 한 정확성을 유지하도록 조정합니다. 여기에는 시간 경과에 따라 지표 값이 변화하거나 급격한 변화가 발생할 경우 조정하도록 모델을 재교육하는 작업이 포함됩니다. 또한 계절성이거나, 급증하거나, 희소한 지표에 대한 모형을 개선하기 위한 예측 변수도 포함됩니다.

SAP HANA에 대한 Application Insights 문제 해결

이 섹션에서는 Application Insights 대시보드에서 반환하는 일반적인 오류를 해결하는 데 도움이 되는 단계를 제공합니다.

60개를 초과하는 모니터링 지표를 추가할 수 없음

출력은 다음 오류를 보여줍니다.

```
Component cannot have more than 60 monitored metrics
```

근본 원인 - 현재 지표 제한은 구성 요소당 모니터링 지표 60개입니다.

해결 방법 - 한도를 지키기 위해 필요하지 않은 지표를 제거합니다.

온보딩 프로세스 이후 표시되는 SAP 지표 없음

다음 정보를 사용하여 온보딩 프로세스 후 SAP 지표가 대시보드에 표시되지 않는 이유를 확인합니다. 첫 번째 단계는 Amazon EC2 인스턴스의 AWS Management Console 또는 익스포터 로그를 사용하여 SAP 지표가 표시되지 않는 이유를 해결하는 것입니다. 다음으로 오류 출력을 검토하여 해결 방법을 찾습니다.

온보딩 이후 SAP 지표가 표시되지 않는 이유 해결

Amazon EC2 인스턴스의 AWS Management Console 또는 익스포터 로그를 사용하여 문제를 해결할 수 있습니다.

AWS Management Console

콘솔을 사용하여 온보딩 이후 표시되는 SAP 지표가 없는 문제 해결

1. AWS Systems Manager 콘솔(https://console.aws.amazon.com/systems-manager/)을 엽니다.
2. 왼쪽 탐색 창에서 State Manager를 선택합니다.
3. 연결에서 문서 `AWSEC2-ApplicationInsightsCloudwatchAgentInstallAndConfigure`의 상태를 확인합니다. 상태가 `Failed`인 경우 실행 ID에서 실패한 ID를 선택하고 출력을 확인합니다.
4. 연결에서 문서 `AWS-ConfigureAWSPackage`의 상태를 확인합니다. 상태가 `Failed`인 경우 실행 ID에서 실패한 ID를 선택하고 출력을 확인합니다.

Exporter logs from Amazon EC2 instance

익스포터 로그를 사용하여 온보딩 이후 표시되는 SAP 지표가 없는 문제 해결

1. SAP HANA 데이터베이스가 실행 중인 Amazon EC2 인스턴스에 연결합니다.
2. 다음 명령을 사용하여 `WORKLOAD_SHORT_NAME`에 올바른 명명 규칙을 찾습니다. 다음 두 단계에서는 이 짧은 이름을 사용합니다.

   ```
   sudo systemctl | grep exporter
   ```

> **Note**
>
> Application Insights는 실행 중인 워크로드에 따라 서비스 이름에 접미사 WORKLOAD_SHORT_NAME을 추가합니다. SAP HANA 단일 노드, 다중 노드 및 고가용성 배포에 대한 짧은 이름은 HANA_SN, HANA_MN, HANA_HA입니다.

3. 익스포터 관리자 서비스 로그에서 오류를 확인하려면 다음 명령을 실행하고 WORKLOAD_SHORT_NAME을 Step 2에서 확인한 짧은 이름으로 바꿉니다.

```
sudo journalctl -e --unit=prometheus-
hanadb_exporter_manager_WORKLOAD_SHORT_NAME.service
```

4. 익스포터 관리자 서비스 로그에 오류가 표시되지 않는 경우 다음 명령을 실행하여 익스포터 서비스 로그에서 오류를 확인합니다.

```
sudo journalctl -e --unit=prometheus-hanadb_exporter_WORKLOAD_SHORT_NAME.service
```

온보딩 후 SAP 지표가 표시되지 않는 문제의 일반적인 근본 원인 해결

다음 예시는 온보딩 후 SAP 지표가 표시되지 않는 문제의 일반적인 근본 원인을 해결하는 방법을 설명합니다.

- 출력은 다음 오류를 보여줍니다.

```
Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.d/default ...
Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/
amazon-cloudwatch-agent.d/ssm_AmazonCloudWatch-ApplicationInsights-
SSMParameterForTESTCWEC2INSTANCEi0d88867f1f3e36285.tmp ...
2023/11/30 22:25:17 Failed to merge multiple json config files.
2023/11/30 22:25:17 Failed to merge multiple json config files.
2023/11/30 22:25:17 Under path : /metrics/append_dimensions | Error : Different
 values are specified for append_dimensions
2023/11/30 22:25:17 Under path : /metrics/metrics_collected/disk | Error : Different
 values are specified for disk
2023/11/30 22:25:17 Under path : /metrics/metrics_collected/mem | Error : Different
 values are specified for mem
2023/11/30 22:25:17 Configuration validation first phase failed. Agent version: 1.0.
 Verify the JSON input is only using features supported by this version.
```

해결 방법 - Application Insights는 기존 CloudWatch 에이전트 구성 파일의 일부로 사전 구성된 것과 동일한 지표를 구성하려고 합니다. `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/`에서 기존 파일을 제거하거나 기존 CloudWatch 에이전트 구성 파일에서 충돌을 일으키는 지표를 제거합니다.

- 출력은 다음 오류를 보여줍니다.

```
Unable to find a host with system database, for more info rerun using -v
```

해결 방법 - 사용자 이름, 암호 또는 데이터베이스 포트가 잘못되었을 수 있습니다. 사용자 이름, 암호 및 포트가 유효한지 확인하고 온보딩 프로세스를 다시 실행합니다.

- 출력은 다음 오류를 보여줍니다.

```
This hdbcli installer is not compatible with your Python interpreter
```

해결 방법 - Python 3.6의 경우 다음 예시와 같이 pip3와 wheel을 업그레이드합니다.

```
python3.6 -m pip install --upgrade pip setuptools wheel
```

- 출력은 다음 오류를 보여줍니다.

```
Unable to install hdbcli using pip3. Please try to install it
```

해결 방법 - `hdbclient` 사전 조건을 따랐는지 확인하거나 pip3에서 `hdbclient`를 수동으로 설치합니다.

- 출력은 다음 오류를 보여줍니다.

```
Package 'boto3' requires a different Python: 3.6.15 not in '>= 3.7'
```

해결 방법 - 이 운영 체제 버전에는 Python 3.8 이상이 필요합니다. Python 3.8 사전 조건을 확인하고 설치합니다.

- 출력에서 다음 설치 오류 중 하나를 보여줍니다.

```
Can not execute `setup.py` since setuptools is not available in the build environment
```

또는

```
[SSL: CERTIFICATE_VERIFY_FAILED]
```

해결 방법 - 다음 예시와 같이 SUSE Linux 명령을 사용하여 Python을 설치합니다. 다음 예시는 최신 버전의 Python 3.8을 설치합니다.

```
wget https://www.python.org/ftp/python/3.8.<LATEST_RELEASE>/
Python-3.8.<LATEST_RELEASE>.tgz
tar xf Python-3.*
cd Python-3.*/
sudo zypper install make gcc-c++ gcc automake autoconf libtool
sudo zypper install zlib-devel
sudo zypper install libopenssl-devel libffi-devel
./configure --with-ensurepip=install
sudo make
sudo make install
sudo su
python3.8 -m pip install --upgrade pip setuptools wheel
```

자습서: SAP NetWeaver에 대한 모니터링 설정

이 자습서에서는 SAP NetWeaver에 대한 모니터링을 설정하도록 Amazon CloudWatch Application Insights를 구성하는 방법을 보여줍니다. CloudWatch Application Insights 자동 대시보드를 사용하여 문제 세부 정보를 시각화하고, 문제 해결을 가속화하고, SAP NetWeaver 애플리케이션 서버의 평균 해결 시간(MTTR)을 줄일 수 있습니다.

CloudWatch Application Insights for SAP NetWeaver 주제

- 지원 환경
- 지원되는 운영 체제
- 특성
- 필수 조건
- 모니터링을 위해 SAP NetWeaver 애플리케이션 서버 설정
- SAP NetWeaver 애플리케이션 서버의 모니터링 관리
- CloudWatch Application Insights에서 탐지한 SAP NetWeaver 문제 보기 및 해결
- SAP NetWeaver에 대한 Application Insights 문제 해결

지원 환경

CloudWatch Application Insights는 다음 시스템 및 패턴에 대한 AWS 리소스 배포를 지원합니다.

- SAP NetWeaver 표준 시스템 배포.
- 여러 Amazon EC2 인스턴스에 SAP NetWeaver 분산 배포.
- 크로스 AZ SAP NetWeaver 데이터베이스 고가용성 설정: SUSE/RHEL 클러스터링을 사용한 2개의 가용 영역에 구성된 고가용성 SAP NetWeaver.

지원되는 운영 체제

CloudWatch Application Insights for SAP NetWeaver는 다음 운영 체제에서 지원됩니다.

- Oracle Linux 8
- Red Hat Enterprise Linux 7.6
- Red Hat Enterprise Linux 7.7
- Red Hat Enterprise Linux 7.9
- Red Hat Enterprise Linux 8.1
- Red Hat Enterprise Linux 8.2
- Red Hat Enterprise Linux 8.4
- Red Hat Enterprise Linux 8.6
- SUSE Linux Enterprise Server 15 for SAP
- SUSE Linux Enterprise Server 15 SP1 for SAP
- SUSE Linux Enterprise Server 15 SP2 for SAP
- SUSE Linux Enterprise Server 15 SP3 for SAP
- SUSE Linux Enterprise Server 15 SP4 for SAP
- SUSE Linux Enterprise Server 12 SP4 for SAP
- SUSE Linux Enterprise Server 12 SP5 for SAP
- SUSE Linux Enterprise Server 15(High Availability 패턴 제외)
- SUSE Linux Enterprise Server 15 SP1(High Availability 패턴 제외)
- SUSE Linux Enterprise Server 15 SP2(High Availability 패턴 제외)
- SUSE Linux Enterprise Server 15 SP3(High Availability 패턴 제외)

- SUSE Linux Enterprise Server 15 SP4(High Availability 패턴 제외)
- SUSE Linux Enterprise Server 12 SP4(High Availability 패턴 제외)
- SUSE Linux Enterprise Server 12 SP5(High Availability 패턴 제외)

특성

CloudWatch Application Insights for SAP NetWeaver 7.0x~7.5x(ABAP Platform 포함)는 다음과 같은 기능을 제공합니다.

- 자동 SAP NetWeaver 워크로드 탐지
- 정적 임계값을 기반으로 한 SAP NetWeaver 경보 자동 생성
- SAP NetWeaver 로그 패턴 자동 인식
- SAP NetWeaver 상태 대시보드
- SAP NetWeaver 문제 대시보드

필수 조건

CloudWatch Application Insights로 SAP NetWeaver를 구성하려면 다음 사전 조건을 수행해야 합니다.

- AWS Systems Manager 활성화 - Amazon EC2 인스턴스에 SSM Agent를 설치하고 SSM에 대해 인스턴스를 활성화합니다. SSM Agent를 설치하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager 설정을 참조하세요.
- Amazon EC2 인스턴스 역할 - SAP NetWeaver 모니터링을 구성하려면 다음 Amazon EC2 인스턴스 역할을 연결해야 합니다.
  - Systems Manager를 사용 설정하려면 `AmazonSSMManagedInstanceCore` 역할을 연결해야 합니다. 자세한 내용은 AWS Systems Manager 자격 증명 기반 정책 예제를 참조하세요.
  - CloudWatch를 통해 인스턴스 지표 및 로그를 내보낼 수 있게 하려면 `CloudWatchAgentServerPolicy` 정책을 연결해야 합니다. 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성을 참조하세요.
- AWS Resource Groups - CloudWatch Application Insights에 애플리케이션을 온보딩하려면 애플리케이션 스택에 사용되는 연결된 모든 AWS 리소스를 포함하는 리소스 그룹을 생성해야 합니다. 여기에는 Amazon EC2 인스턴스, Amazon EFS 및 SAP NetWeaver 애플리케이션 서버를 실행하는 Amazon EBS 볼륨이 포함됩니다. 계정당 SAP NetWeaver 시스템이 여러 개인 경우, 각 SAP NetWeaver 시스템에 대한 AWS 리소스를 포함하는 하나의 리소스 그룹을 생성하는 것이 좋습니다. 리소스 그룹 생성에 대한 자세한 내용은 AWS Resource Groups 사용 설명서를 참조하세요.

- IAM 권한 - 관리 액세스 권한이 없는 사용자의 경우 서비스 연결 역할을 생성하고 이를 사용자의 아이덴티티에 연결할 수 있도록 Application Insights를 허용하는 AWS Identity and Access Management(IAM) 정책을 생성해야 합니다. IAM 정책을 생성하는 방법에 대한 자세한 내용은 IAM 정책을 참조하세요.
- 서비스 연결 역할 - Application Insights는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Application Insights 관리 콘솔에서 첫 번째 Application Insights 애플리케이션을 생성할 때 생성됩니다. 자세한 내용은 CloudWatch Application Insights에 서비스 연결 역할 사용 단원을 참조하십시오.
- Amazon CloudWatch 에이전트 – Application Insights가 CloudWatch 에이전트를 설치하고 구성합니다. CloudWatch 에이전트를 설치하면 Application Insights가 구성을 유지합니다. 병합 충돌을 방지하려면 Application Insights에서 사용하려는 리소스 구성을 기존 CloudWatch 에이전트 구성 파일에서 제거하세요. 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하십시오.

모니터링을 위해 SAP NetWeaver 애플리케이션 서버 설정

다음 단계에 따라 SAP NetWeaver 애플리케이션 서버에 대한 모니터링을 설정하세요.

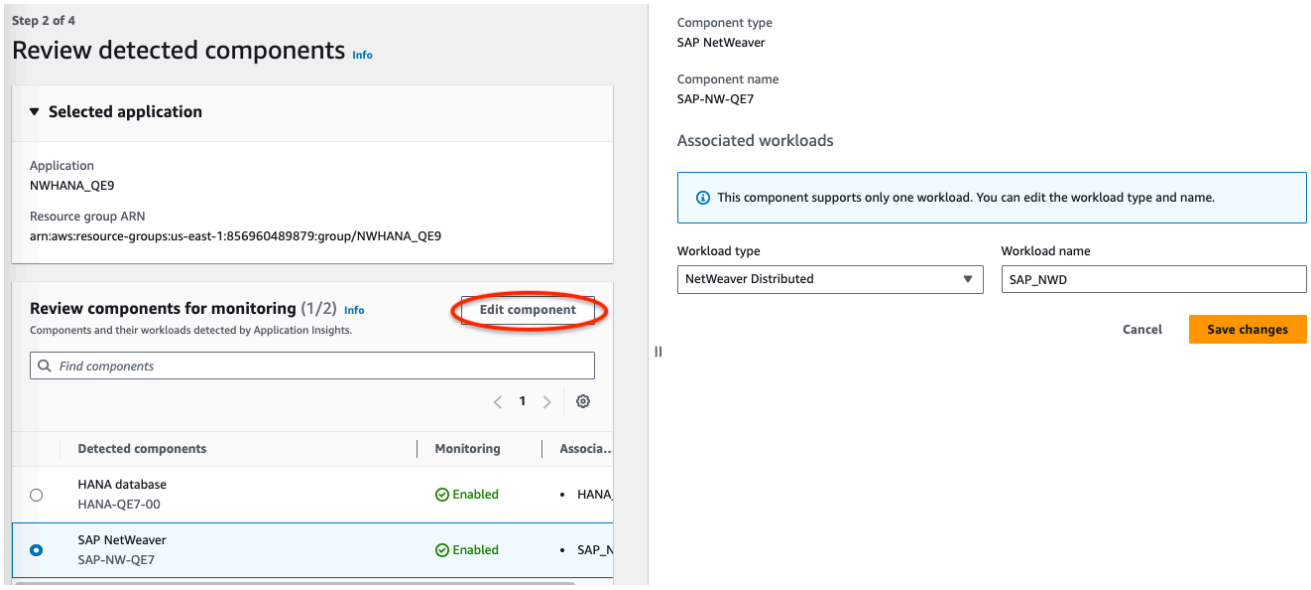
모니터링을 설정하려면 다음을 수행하세요.

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 Insights(인사이트)에서 Application Insights를 선택합니다.
3. Application Insights 페이지에는 Application Insights를 사용하여 모니터링되는 애플리케이션 목록과 각 애플리케이션의 모니터링 상태가 표시됩니다. 오른쪽 상단 모서리에서 Add an application(애플리케이션 추가)을 선택합니다.
4. Specify application details(애플리케이션 세부 정보 지정) 페이지에서 Resource group(리소스 그룹)의 드롭다운 목록에 있는 SAP NetWeaver 리소스를 포함하는 생성한 AWS 리소스 그룹을 선택합니다. 애플리케이션에 대한 리소스 그룹을 생성하지 않은 경우 Resource group(리소스 그룹) 드롭다운 목록에서 Create new resource group(새 리소스 그룹 생성)을 선택해 생성할 수 있습니다.
5. Automatic monitoring of new resources(새로운 리소스 자동 모니터링) 확인란을 선택하여 Application Insight가 온보딩 후 애플리케이션의 리소스 그룹에 추가된 리소스를 자동으로 모니터링할 수 있게 합니다.
6. Monitor EventBridge events(EventBridge 이벤트 모니터링)에서 Application Insights 모니터링을 CloudWatch Events와 통합하려면 확인란을 선택하여 Amazon EBS, Amazon EC2, AWS CodeDeploy, Amazon ECS, AWS Health API 및 알림, Amazon RDS, Amazon S3, AWS Step Functions로부터 인사이트를 얻습니다.

7. 선택한 애플리케이션에 문제가 감지될 때 이를 확인하고 알림을 받으려면 Integrate with AWS Systems Manager OpsCenter(AWS Systems Manager OpsCenter와 통합)에서 Generate OpsCenter OpsItems for remedial actions(수정 조치를 위해 OpsCenter OpsItems 생성) 옆의 확인란을 선택합니다. AWS 리소스와 관련된 운영 작업 항목(OpsItem)을 해결하기 위해 수행된 작업을 추적하려면 SNS 주제 ARN을 제공합니다.
8. 태그를 입력하면(선택 사항) 리소스를 식별하고 구성하는 데 도움이 됩니다. CloudWatch Application Insights는 태그 기반 및 AWS CloudFormation 스택 기반 리소스 그룹을 모두 지원합니다(Application Auto Scaling 그룹 제외). 자세한 내용은 AWS Resource Groups 및 태그 사용 설명서에서 Tag Editor를 참조하세요.
9. 탐지된 구성 요소를 검토하려면 다음을 선택합니다.
10. 탐지된 구성 요소 검토 페이지에는 CloudWatch 애플리케이션 인사이트에서 자동으로 탐지된 모니터링되는 구성 요소와 해당 워크로드가 나열됩니다.

    - 워크로드 유형과 이름을 편집하려면 구성 요소 편집을 선택합니다.

    > **Note**
    >
    > 탐지된 NetWeaver 분산 또는 NetWeaver 고가용성 워크로드가 포함된 구성 요소는 구성 요소에서 하나의 워크로드만 지원합니다.



11. 다음을 선택합니다.
12. Specify component details(구성 요소 세부 정보 지정) 페이지에서 Next(다음)를 선택합니다.

13. 애플리케이션 모니터링 구성을 검토한 다음 제출을 선택합니다.
14. 애플리케이션 요약, 대시보드, 구성 요소, 워크로드를 볼 수 있는 애플리케이션 세부 정보 페이지가 열립니다. Configuration history(구성 기록), Log patterns(로그 패턴), 사용자가 생성한 모든 Tags(태그)를 확인할 수 있습니다. 애플리케이션을 제출하면 CloudWatch Application Insights가 SAP NetWeaver 시스템에 대한 모든 지표와 경보를 배포합니다. 이 작업에는 최대 1시간이 소요될 수 있습니다.

SAP NetWeaver 애플리케이션 서버의 모니터링 관리

다음 단계에 따라 SAP NetWeaver 애플리케이션 서버의 모니터링을 관리하세요.

모니터링을 관리하려면 다음을 수행하세요.

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 Insights(인사이트)에서 Application Insights를 선택합니다.
3. List view(목록 보기) 탭을 선택합니다.
4. Application Insights 페이지에는 Application Insights를 사용하여 모니터링되는 애플리케이션 목록과 각 애플리케이션의 모니터링 상태가 표시됩니다.
5. 애플리케이션을 선택합니다.
6. Components(구성 요소) 탭을 선택합니다.
7. 모니터링되는 구성 요소(Monitored components)에서 구성 요소 이름 옆에 있는 라디오 버튼을 선택합니다. 그런 다음 Manage monitoring(모니터링 관리)을 선택합니다.
8. Instance logs(인스턴스 로그)에서 기존 로그 경로, 로그 패턴 세트 및 로그 그룹 이름을 업데이트할 수 있습니다. 또한 애플리케이션 로그(Application logs)를 3개까지 추가할 수 있습니다.
9. Metrics(지표)에서 요구 사항에 따라 SAP NetWeaver 지표를 선택할 수 있습니다. SAP NetWeaver 지표 이름에는 `sap` 접두사가 붙습니다. 구성 요소당 40개까지 지표를 추가할 수 있습니다.
10. 사용자 지정 경보(Custom alarms)에서 CloudWatch Application Insights를 통해 모니터링할 경보를 추가할 수 있습니다.
11. 애플리케이션 모니터링 구성을 검토하고 Save(저장)를 선택합니다. 구성을 제출하면 계정에서 SAP NetWeaver 시스템에 대한 모든 지표 및 경보를 업데이트합니다.

CloudWatch Application Insights에서 탐지한 SAP NetWeaver 문제 보기 및 해결

다음 섹션에서는 Application Insights에서 SAP NetWeaver에 대한 모니터링을 구성할 때 발생하는 일반적인 문제 해결 시나리오를 해결하는 데 도움이 되는 단계를 제공합니다.

주제 문제 해결

- SAP NetWeaver 데이터베이스 연결 문제
- SAP NetWeaver 애플리케이션 가용성 문제

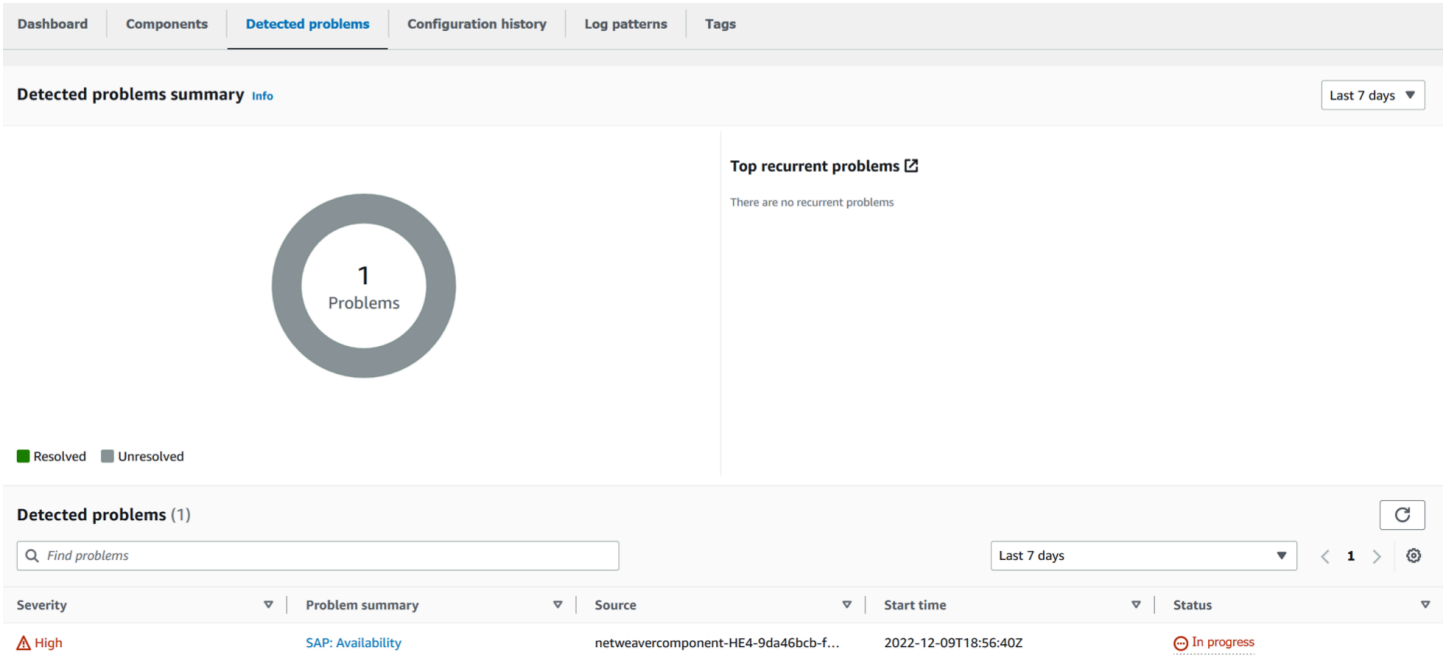
SAP NetWeaver 데이터베이스 연결 문제

설명

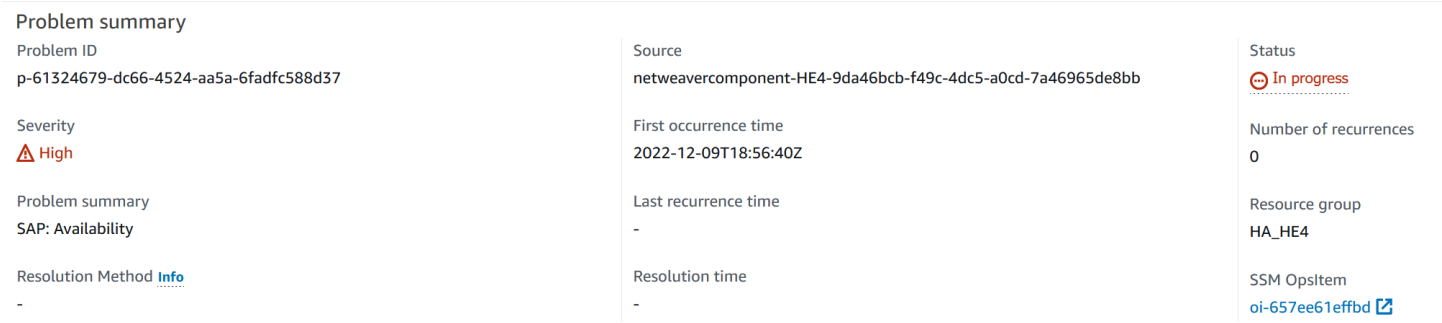
SAP NetWeaver 애플리케이션에서 데이터베이스 연결 문제가 발생했습니다.

원인

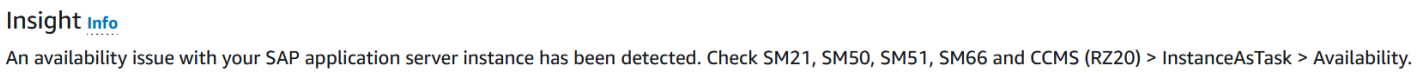
CloudWatch Application Insights 콘솔로 이동하고 SAP NetWeaver Application Insights 문제 대시보드를 확인하여 연결 문제를 식별할 수 있습니다. Problem summary(문제 요약)에서 링크를 선택하여 특정 문제를 확인합니다.



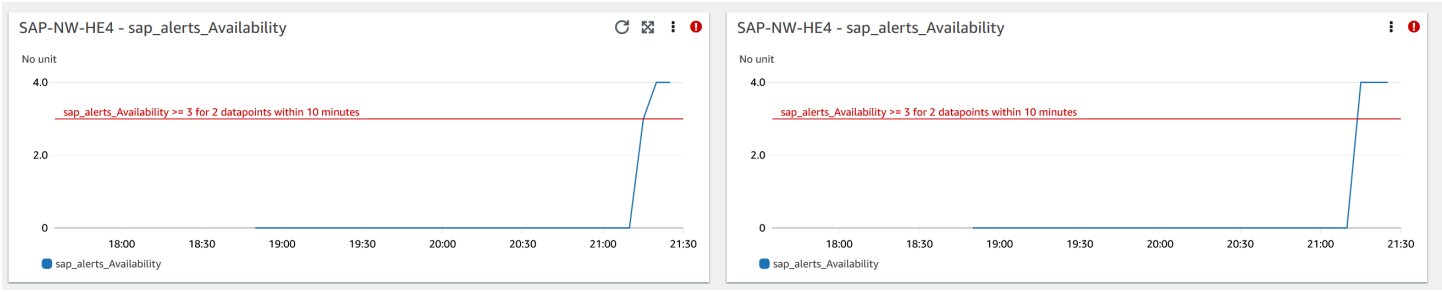
다음 예제에서는 Problem summary(문제 요약) 아래의 SAP: Availability가 문제입니다.



Problem summary(문제 요약) 바로 다음의 Insight(인사이트) 섹션은 오류에 대한 추가 컨텍스트와 문제의 원인에 대한 추가 정보를 얻을 수 있는 위치를 제공합니다.



동일한 문제 대시보드에서 문제 탐지가 그룹화한 관련 로그와 지표를 확인하여 오류의 원인을 격리할 수 있습니다. `sap_alerts_Availability` 지표는 시간 경과에 따른 SAP NetWeaver 시스템의 가용성을 추적합니다. 기록 추적을 사용하여 지표가 오류 상태를 시작하거나 경보 임계값을 위반한 시기를 연관시킬 수 있습니다. 다음 예제에서는 SAP NetWeaver 시스템에 가용성 문제가 있습니다. 이 예제에서는 두 개의 SAP 애플리케이션 서버 인스턴스가 있고 각 인스턴스에 대해 경보가 생성되었으므로 두 개의 경보를 보여줍니다.

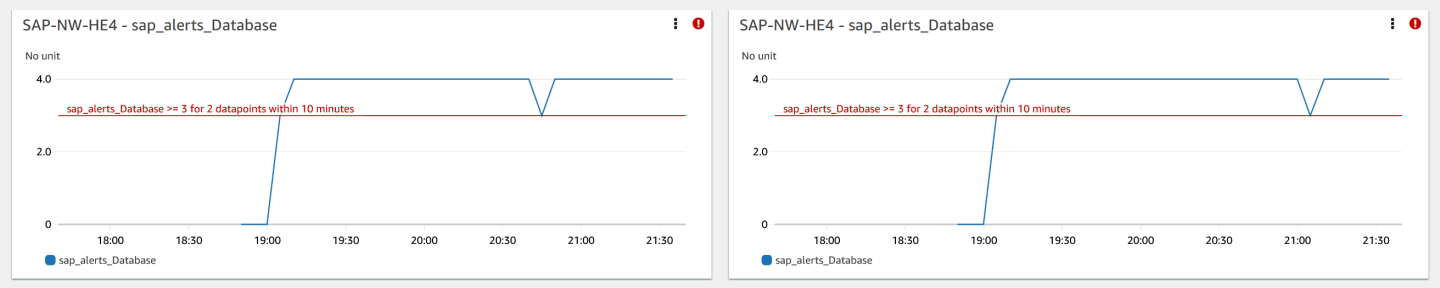


각 경보에 대한 자세한 내용을 보려면 `sap_alerts_Availability` 지표 이름 위로 마우스를 가져갑니다.

CWAgent sap_alerts_Availability

Application:	HA_HE4
ComponentName:	SAP-NW-HE4
instance_hostname:	sapapp
instance_number:	0
object:	InstanceAsTask
SID:	HE4
Region:	us-east-1
Threshold:	sap_alerts_Availability >= 3 for 2 datapoints within 10 minutes
Period:	5 minutes
Statistic:	Maximum
Unit:	None
Min:	0
Max:	4
Average:	0.657143
Sum:	23
Last value:	4
Last time:	2022-12-09 21:40:00 UTC

다음 예제에서 `sap_alerts_Database` 지표는 데이터베이스 계층에 문제 또는 오류가 있음을 보여줍니다. 이 경보는 SAP NetWeaver가 해당 데이터베이스에 연결하거나 이와 통신하는 데 문제가 있음을 나타냅니다.

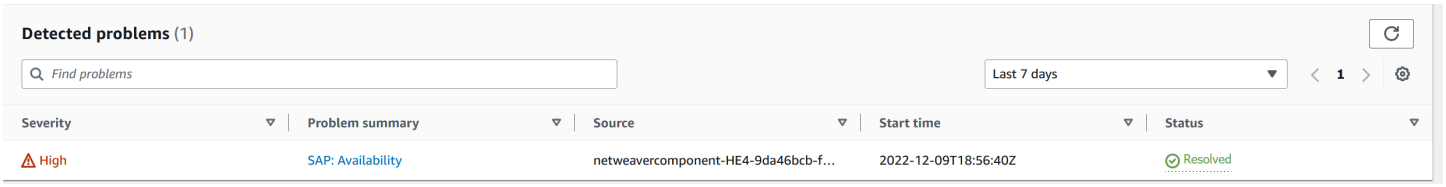


데이터베이스는 SAP NetWeaver의 핵심 리소스이므로 데이터베이스에 문제가 있거나 오류가 발생하면 관련 경보가 많이 발생할 수 있습니다. 다음 예제에서는 데이터베이스를 사용할 수 없어 `sap_alerts_FrontendResponseTime` 및 `sap_alerts_LongRunners` 지표가 시작됩니다.

해결 방법

Application Insights는 탐지된 문제를 매시간 모니터링합니다. SAP NetWeaver 로그 파일에 새로운 관련 로그 항목이 없으면 이전 로그 항목이 해결된 것으로 처리됩니다. CloudWatch 경보와 관련된 모든 오류 조건을 수정해야 합니다. 오류 조건이 수정된 후 경보와 로그가 복구되면 경보가 해결됩니다. CloudWatch 로그 오류 및 경보가 모두 해결되면 Application Insight는 오류 탐지를 중지하고 1시간 내에 문제가 자동으로 해결됩니다. 문제 대시보드에 최신 문제가 표시되도록 모든 로그 오류 조건 및 경보를 해결하는 것이 좋습니다.

다음 예제에서 SAP Availability 문제가 해결되었습니다.



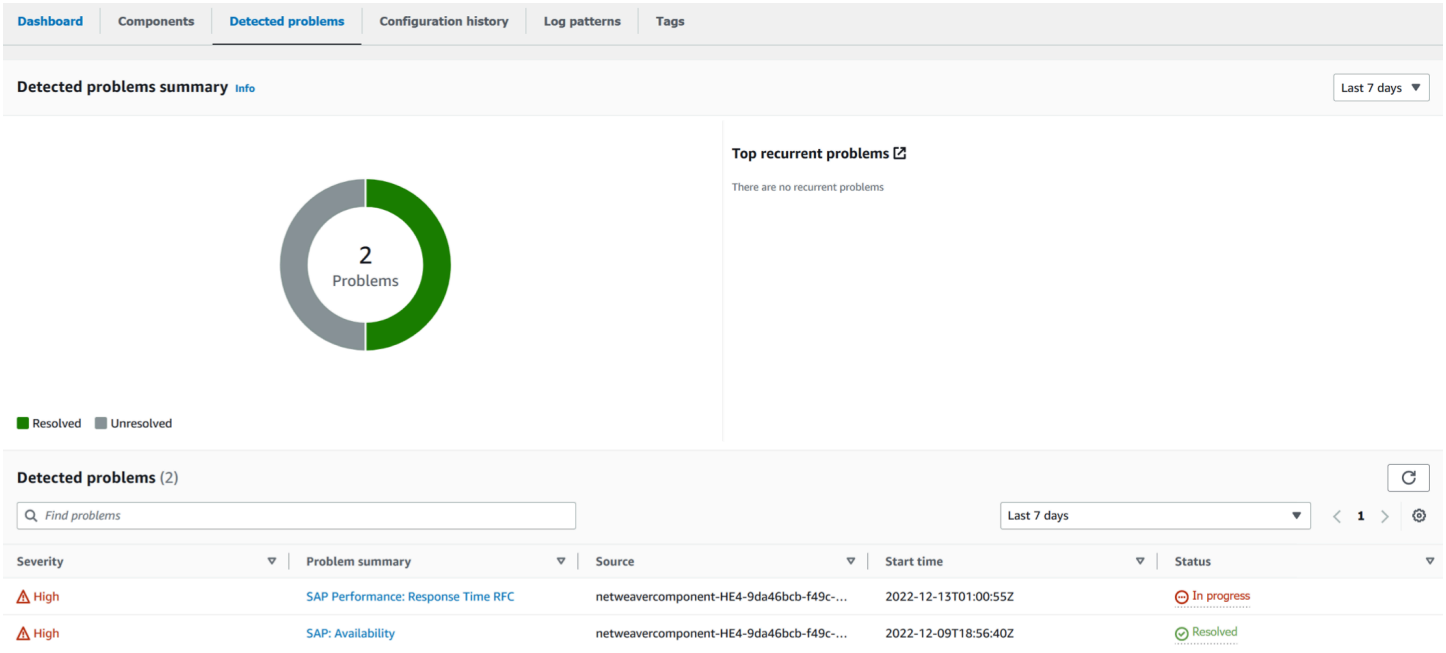
SAP NetWeaver 애플리케이션 가용성 문제

설명

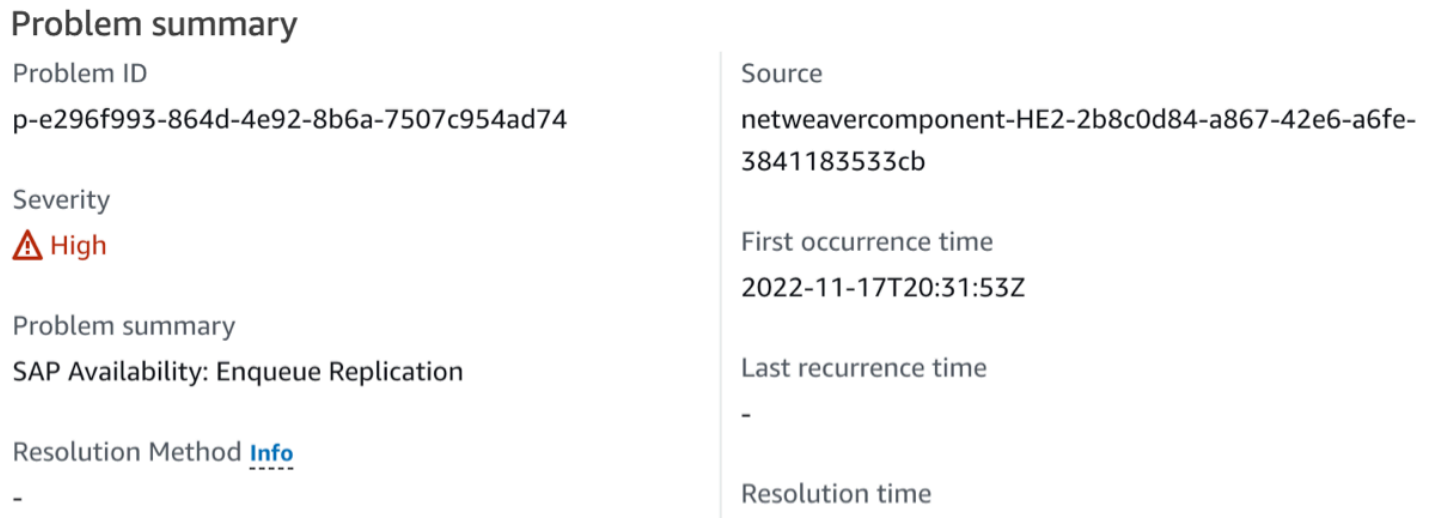
SAP NetWeaver High Availability Enqueue Replication이 작동을 중지했습니다.

원인

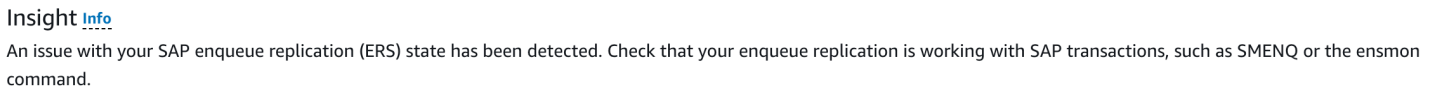
CloudWatch Application Insights 콘솔로 이동하고 SAP NetWeaver Application Insights 문제 대시보드를 확인하여 연결 문제를 식별할 수 있습니다. Problem summary(문제 요약)에서 링크를 선택하여 특정 문제를 확인합니다.



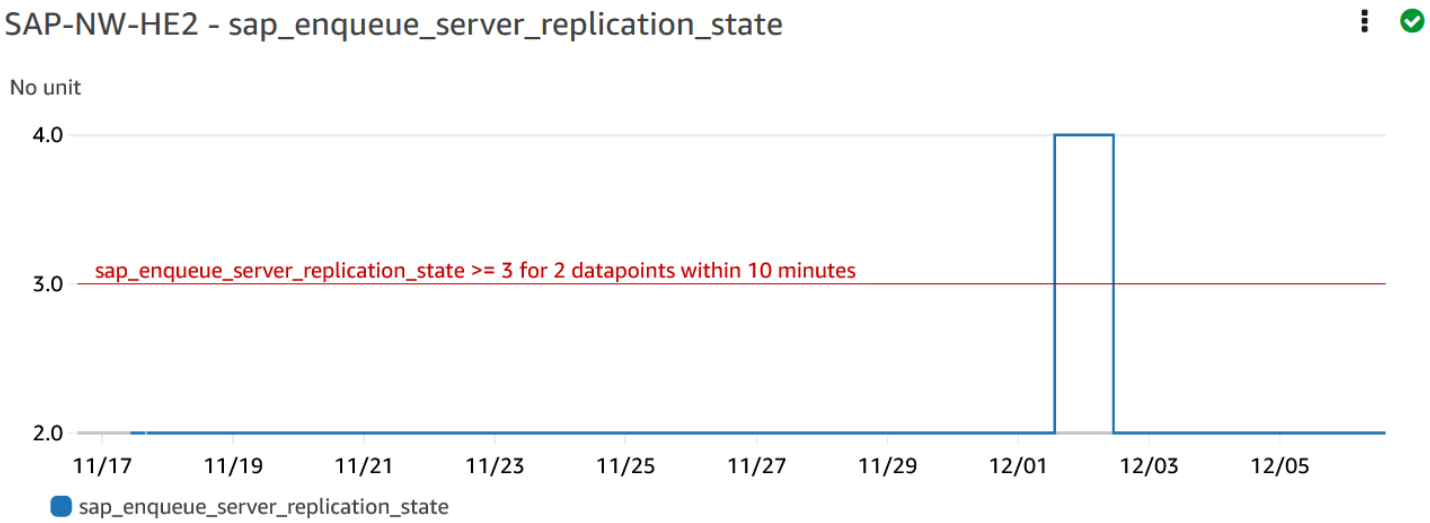
다음 예제에서는 Problem summary(문제 요약) 아래의 High Availability Enqueue Replication이 문제입니다.



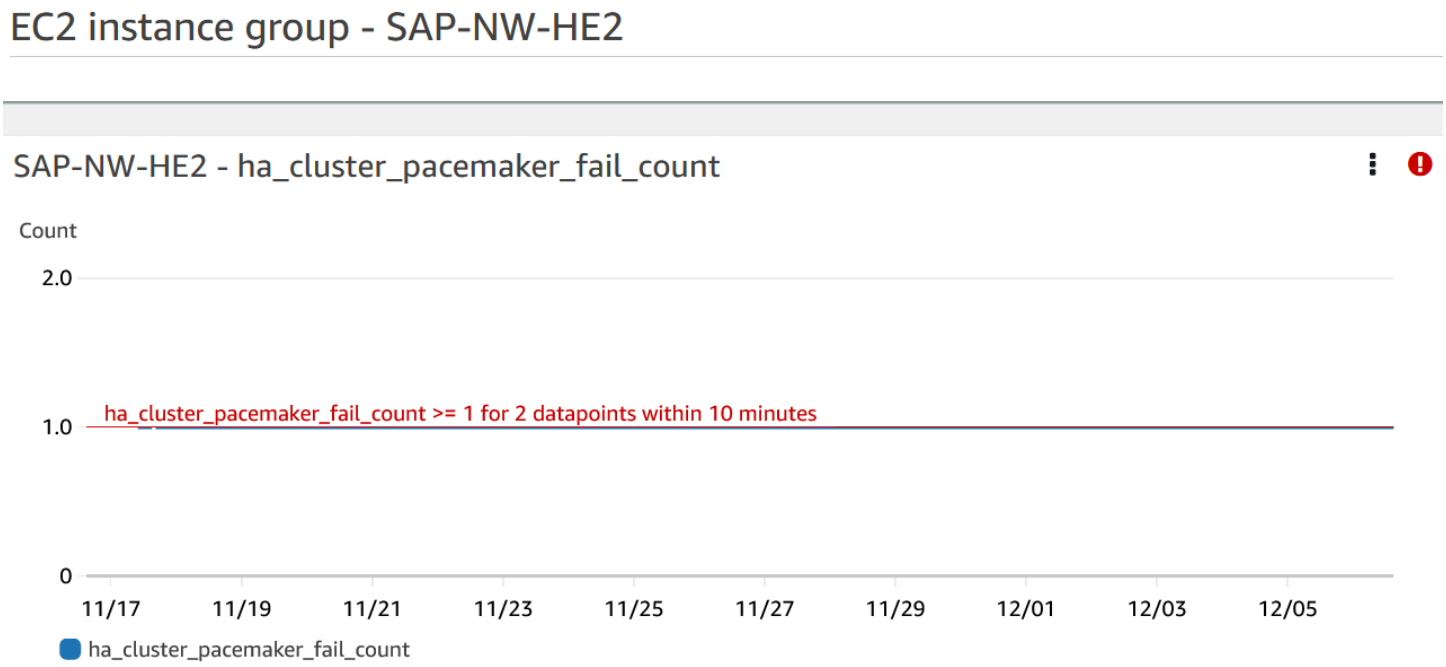
Problem summary(문제 요약) 바로 다음의 Insight(인사이트) 섹션은 오류에 대한 추가 컨텍스트와 문제의 원인에 대한 추가 정보를 얻을 수 있는 위치를 제공합니다.



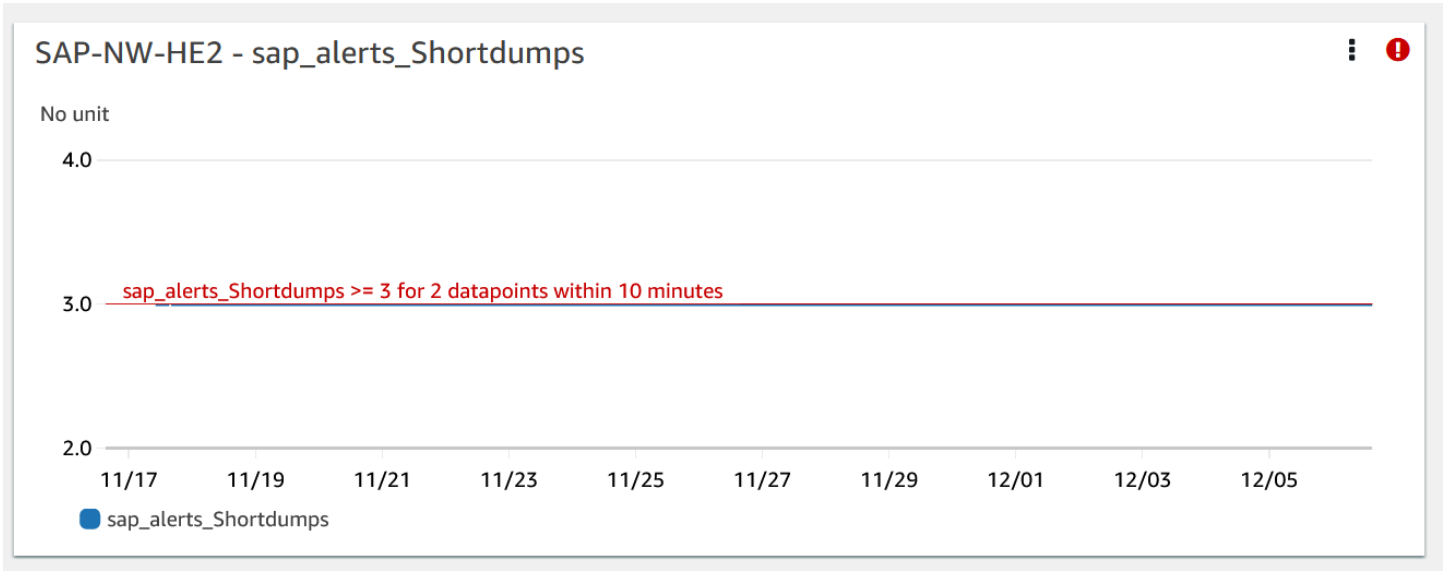
다음 예제에서는 오류의 원인을 격리하는 데 도움이 되도록 그룹화된 로그 및 지표를 보는 문제 대시보드를 보여줍니다. `sap_enqueue_server_replication_state` 지표는 시간 경과에 따라 값을 추적합니다. 기록 추적을 사용하여 지표가 오류 상태를 시작하거나 경보 임계값을 위반한 시기를 연관시킬 수 있습니다.



다음 예제에서 `ha_cluster_pacemaker_fail_count` 지표는 고가용성 페이스메이커 클러스터에서 리소스 오류가 발생했음을 보여줍니다. 실패 횟수가 1보다 크거나 같은 특정 페이스메이커 리소스는 구성 요소 대시보드에서 식별됩니다.

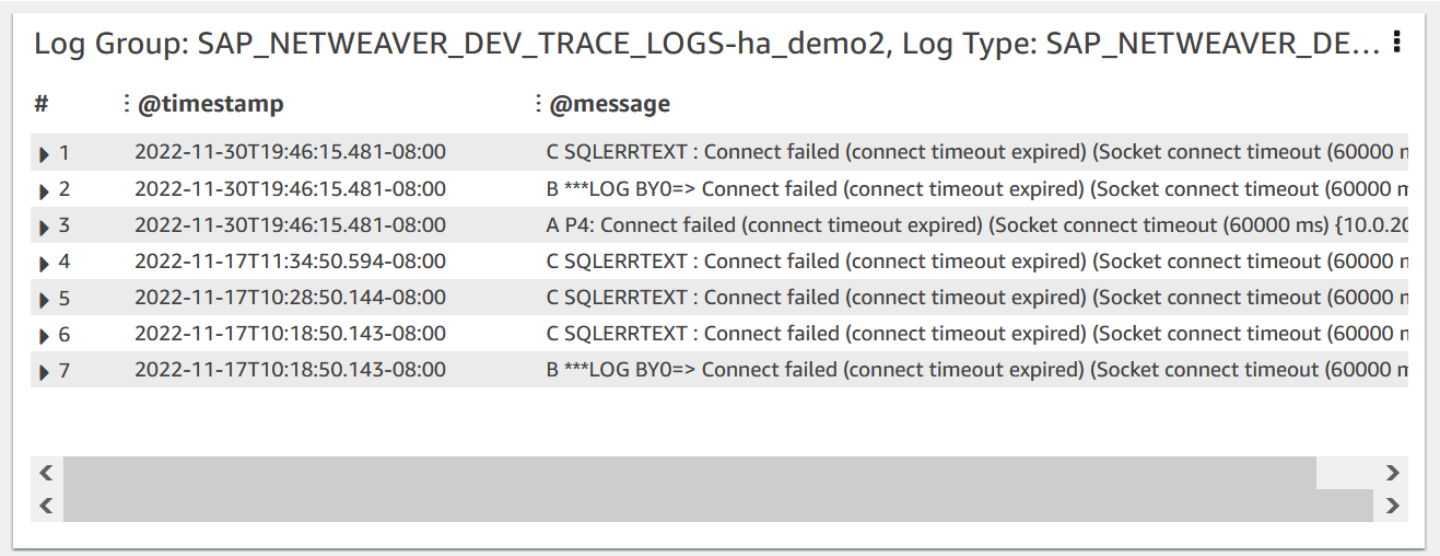


다음 예제에서는 문제 탐지 시 SAP 애플리케이션 성능이 감소했음을 나타내는 `sap_alerts_Shortdumps` 지표를 보여줍니다.

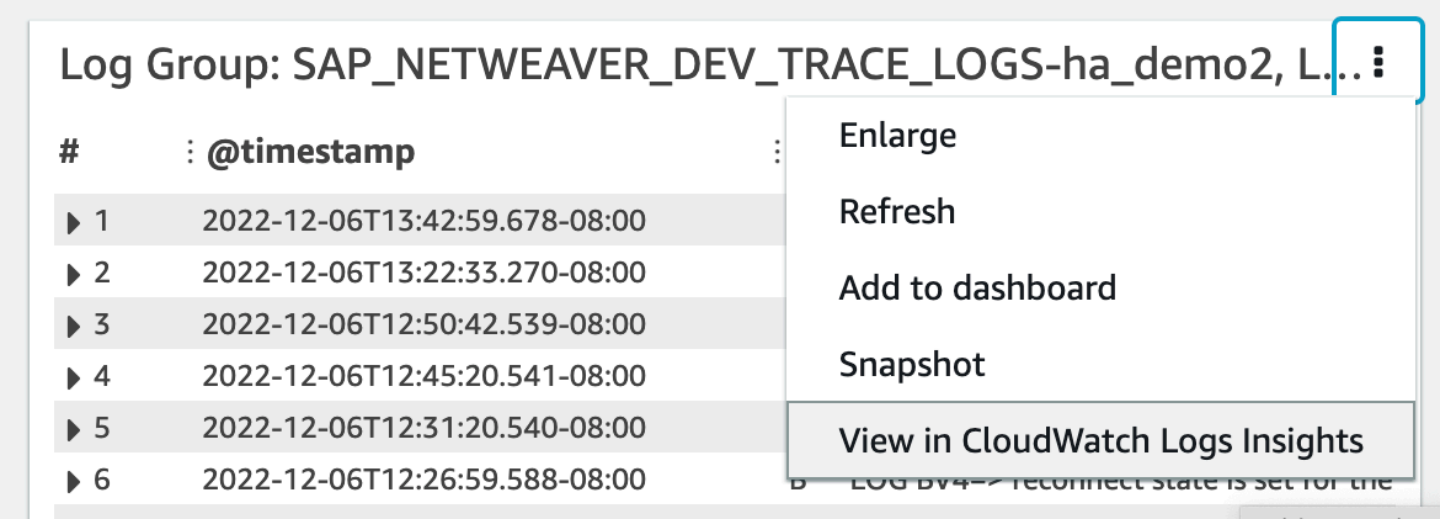


로그

로그 항목은 문제 탐지 시 SAP NetWeaver 계층에서 발생한 문제를 더 잘 이해하는 데 도움이 됩니다. 문제 대시보드의 로그 그룹 위젯은 문제의 특정 시간을 보여줍니다.



로그에 대한 자세한 내용을 보려면 오른쪽 상단 모서리에 있는 세 개의 세로 점을 선택하고 View in CloudWatch Logs Insights(CloudWatch Logs Insights에서 보기)를 선택합니다.

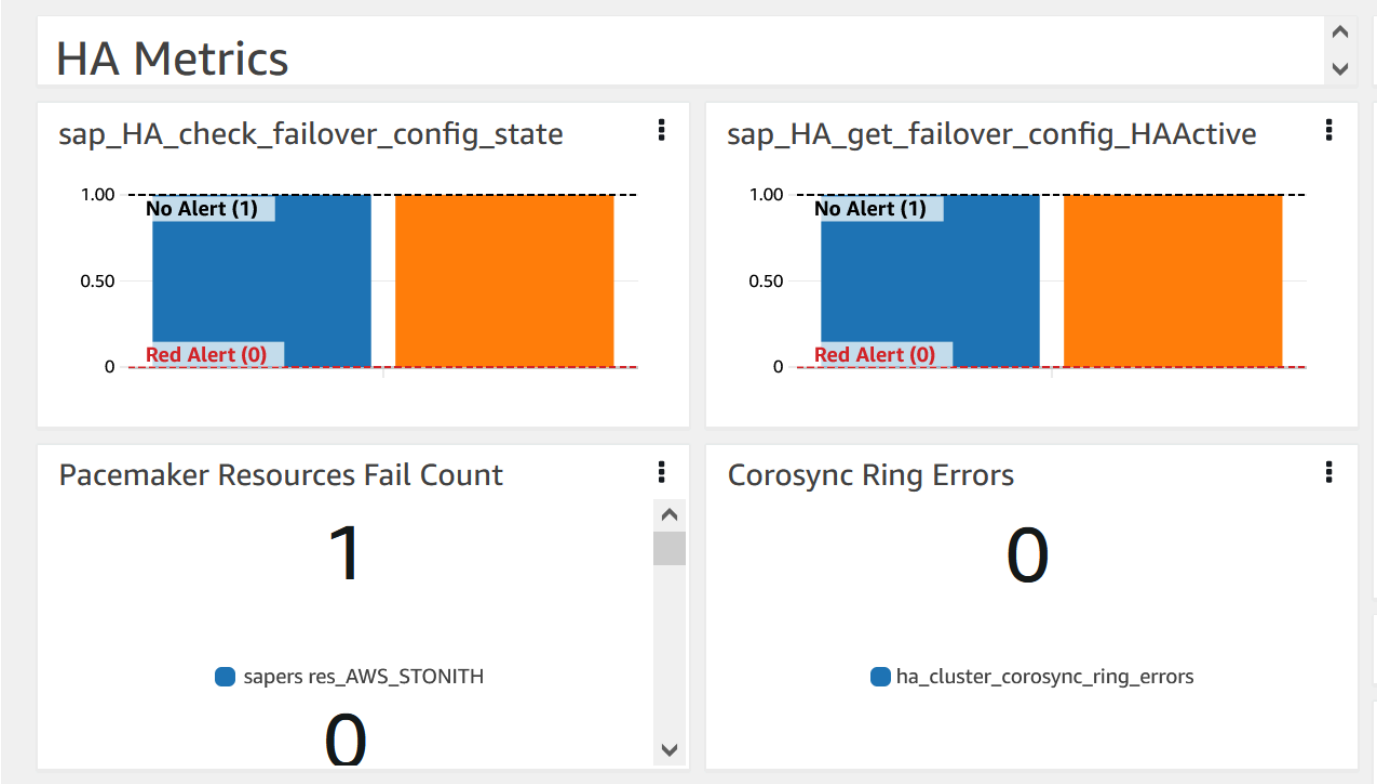


다음 단계에 따라 문제 대시보드에 표시되는 지표와 경보에 대한 자세한 정보를 얻으세요.

지표와 경보에 대한 자세한 정보를 얻으려면 다음을 수행하세요.

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 Insights(인사이트)에서 Application Insights를 선택합니다. 그런 다음 List view(목록 보기) 탭을 선택하고 애플리케이션을 선택합니다.
3. Components(구성 요소) 탭을 선택합니다. 그런 다음 자세한 정보를 얻으려는 SAP NetWeaver 구성 요소를 선택합니다.

다음 예제에서는 문제 대시보드에 표시된 `ha_cluster_pacemaker_fail_count` 지표가 있는 HA Metrics(HA 지표) 섹션을 보여줍니다.

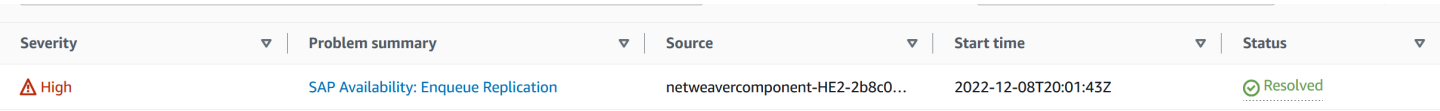


해결 방법

Application Insights는 탐지된 문제를 매시간 모니터링합니다. SAP NetWeaver 로그 파일에 새로운 관련 로그 항목이 없으면 이전 로그 항목이 해결된 것으로 처리됩니다. 이 문제와 관련된 모든 오류 조건을 수정해야 합니다.

`sap_alerts_Shortdumps` 경보의 경우 트랜잭션 코드 `RZ20 # R3Abap # Shortdumps`를 사용하여 CCMS 경보로 이동하여 SAP NetWeaver 시스템에서 경보를 해결해야 합니다. CCMS 알림에 대한 자세한 내용은 SAP 웹 사이트를 참조하세요. Shortdump 트리의 모든 CCMS 경고를 해결합니다. SAP NetWeaver 시스템에서 모든 경보가 해결된 후 CloudWatch는 더 이상 경보 상태의 지표를 보고하지 않습니다.

CloudWatch 로그 오류 및 경보가 모두 해결되면 Application Insight는 오류 탐지를 중지하고 1시간 내에 문제가 자동으로 해결됩니다. 문제 대시보드에 최신 문제가 표시되도록 모든 로그 오류 조건 및 경보를 해결하는 것이 좋습니다. 다음 예제에서는 SAP Netweaver High Availability Enqueue Replication 문제가 해결되었습니다.



Severity	Problem summary	Source	Start time	Status
High	SAP Availability: Enqueue Replication	netweavercomponent-HE2-2b8c0...	2022-12-08T20:01:43Z	Resolved

SAP NetWeaver에 대한 Application Insights 문제 해결

이 섹션에서는 Application Insights 대시보드에서 반환하는 일반적인 오류를 해결하는 데 도움이 되는 단계를 제공합니다.

60개 이상의 모니터 지표를 추가할 수 없음

오류 반환: `Component cannot have more than 60 monitored metrics.`

근본 원인: `The current metric limit is 60 monitor metrics per component.`

해결 방법: 한도를 준수하는 데 필요하지 않은 지표를 제거합니다.

온보딩 프로세스 후 대시보드에 SAP 지표가 표시되지 않음

근본 원인: 구성 요소 대시보드는 5분 지표 기간을 사용하여 데이터 포인트를 집계합니다.

해결 방법: 5분 후 모든 지표가 대시보드에 표시되어야 합니다.

SAP 지표 및 경보가 대시보드에 표시되지 않음

다음 단계에 따라 온보딩 프로세스 후 SAP 지표 및 경보가 대시보드에 표시되지 않는 이유를 확인하세요.

지표 및 경보 관련 문제를 식별하려면 다음을 수행하세요.

1. CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 창의 Insights(인사이트)에서 Application Insights를 선택합니다. 그런 다음 List view(목록 보기) 탭을 선택하고 애플리케이션을 선택합니다.
3. Configuration history(구성 기록) 탭을 선택합니다.
4. 누락된 지표 데이터 포인트가 있는 경우 `prometheus-sap_host_exporter`와 관련된 오류를 확인합니다.
5. 이전 단계에서 오류를 찾을 수 없는 경우 Linux 인스턴스에 연결합니다. High Availability 배포의 경우 기본 클러스터 Amazon EC2 인스턴스에 연결합니다.
6. 인스턴스에서 다음 명령을 사용하여 내보내기 도구가 실행 중인지 확인합니다. 기본 포트는 `9680`입니다. 다른 포트를 사용하는 경우 `9680`을 사용 중인 포트로 바꿉니다.

```
curl localhost:9680/metrics
```

데이터가 반환되지 않으면 내보내기 도구가 시작되지 않은 것입니다.

7. 다음 두 단계에서 WORKLOAD_SHORT_NAME에 사용할 올바른 명명 규칙을 찾으려면 다음 명령을 실행합니다.

> **Note**
>
> Application Insights는 실행 중인 워크로드에 따라 서비스 이름에 접미사 WORKLOAD_SHORT_NAME을 추가합니다. NetWeaver 분산, 표준 및 고가용성 배포의 짧은 이름은 SAP_NWD, SAP_NWS, SAP_NWH입니다.

```
sudo systemctl | grep exporter
```

8. 내보내기 도구 서비스 로그에서 오류를 확인하려면 다음 명령을 실행합니다.

```
sudo journalctl -e --unit=prometheus-sap_host_exporter_WORKLOAD_SHORT_NAME.service
```

9. 내보내기 도구 관리자 서비스 로그에서 오류를 확인하려면 다음 명령을 실행합니다.

```
sudo journalctl -e --unit=prometheus-
sap_host_exporter_manager_WORKLOAD_SHORT_NAME.service
```

> **Note**
>
> 이 서비스는 항상 가동되고 실행되어야 합니다.

이 명령에서 오류를 반환하지 않으면 다음 단계로 이동합니다.

10. 내보내기 도구를 수동으로 시작하려면 다음 명령을 실행합니다. 그런 다음 내보내기 도구 출력을 확인합니다.

```
sudo /opt/aws/sap_host_exporter/sap_host_exporter
```

오류를 확인한 후 내보내기 프로세스를 종료할 수 있습니다.

근본 원인: 이 문제에는 여러 가지 원인이 있을 수 있습니다. 일반적인 원인은 내보내기 도구가 애플리케이션 서버 인스턴스 중 하나에 연결할 수 없기 때문입니다.

해결 방법

다음 단계에 따라 애플리케이션 서버 인스턴스에 내보내기 도구를 연결하세요. SAP 애플리케이션 인스턴스가 실행 중인지 확인하고 SAPControl을 사용하여 인스턴스에 연결합니다.

애플리케이션 서버 인스턴스에 내보내기 도구를 연결하려면 다음을 수행하세요.

1. Amazon EC2 인스턴스에서 다음 명령을 실행하여 SAP 애플리케이션이 실행 중인지 확인합니다.

   ```
   sapcontrol -nr <App_InstNo> -function GetProcessList
   ```

2. 작동하는 SAP Control 연결을 설정해야 합니다. SAPControl 연결이 작동하지 않으면 관련 SAP 애플리케이션 인스턴스에서 문제의 근본 원인을 찾습니다.
3. SAP Control 연결 문제를 해결한 후 내보내기 도구를 수동으로 시작하려면 다음 명령을 실행하세요.

   ```
   sudo systemctl start prometheus-sap_host_exporter.service
   ```

4. SAPControl 연결 문제를 해결할 수 없으면 다음 절차를 임시 해결책으로 사용하세요.

   a. AWS Systems Manager 콘솔을 엽니다.
   b. 왼쪽 탐색 창에서 State Manager를 선택합니다.
   c. Associations(연결)에서 SAP NetWeaver 시스템의 연결을 검색합니다.

      ```
      Association Name: Equal: AWS-ApplicationInsights-
      SSMSAPHostExporterAssociationForCUSTOMSAPNW<SID>-1
      ```

   d. Association id(연결 ID)를 선택합니다.
   e. Parameters(파라미터) 탭을 선택하고 additionalArguments에서 애플리케이션 서버 번호를 제거합니다.
   f. Apply Association Now(지금 연결 적용)를 선택합니다.

      > **Note**
      >
      > 이는 임시 해결책입니다. 구성 요소의 모니터링 구성이 업데이트되면 인스턴스가 다시 추가됩니다.

Amazon CloudWatch Application Insights에서 감지한 문제 보기 및 해결

이 섹션의 주제는 Application Insights가 표시하는 감지된 문제 및 인사이트에 대한 자세한 정보를 제공합니다. 또한 계정이나 구성에서 발견된 문제에 대한 권장되는 해결 방법을 제공합니다.

주제 문제 해결

- CloudWatch 콘솔 개요
- 애플리케이션 인사이트 문제 요약 페이지
- CloudWatch 에이전트 병합 충돌 실패
- 경보가 생성되지 않음
- 피드백
- 구성 오류

CloudWatch 콘솔 개요

모니터링되는 애플리케이션에 영향을 주는 문제에 대한 개요는 CloudWatch 콘솔의 개요 페이지에 있는 CloudWatch Application Insights 창에서 확인할 수 있습니다. 자세한 내용은 Amazon CloudWatch Application Insights 시작하기 단원을 참조하십시오.

CloudWatch Application Insights 개요 창에는 다음 내용이 표시됩니다.

- 감지된 문제의 심각도: 높음/중간/낮음
- 문제의 간략한 요약
- 문제 소스
- 문제 시작 시간
- 문제의 해결 상태
- 영향 받는 리소스 그룹

특정 문제의 세부 정보를 확인하려면 문제 요약(Problem Summary)에서 문제에 대한 설명을 선택합니다. 세부 대시보드에는 문제에 대한 통찰력과 관련 지표 이상 및 로그 오류의 조각이 표시됩니다. 유용한지 여부를 선택하여 인사이트의 관련성에 대한 피드백을 제공할 수 있습니다.

구성되지 않은 새 리소스가 감지된 경우 문제점 요약 설명은 구성 편집(Edit configuration) 마법사로 이동하여 새 리소스를 구성합니다. 세부 대시보드의 오른쪽 상단에 있는 구성 보기/편집(View/edit configuration)을 선택하여 리소스 그룹 구성을 보거나 편집할 수 있습니다.

개요로 돌아가려면 CloudWatch Application Insights 세부 대시보드 헤더 옆에 있는 [개요로 돌아가기(Back to overview)]를 선택합니다.

애플리케이션 인사이트 문제 요약 페이지

애플리케이션 인사이트 문제 요약 페이지

CloudWatch Application Insights는 문제 요약 페이지에 감지된 문제에 관한 다음 정보를 제공합니다.

- 문제의 간략한 요약
- 문제의 시작 시간 및 날짜
- 문제의 심각도: 높음/중간/낮음
- 감지된 문제의 상태: 진행 중/해결됨
- 통찰력: 감지된 문제 및 가능한 근본 원인에 대해 자동으로 생성된 통찰력
- 인사이트에 대한 피드백: CloudWatch Application Insights에서 생성한 인사이트의 유용성에 관해 제공한 피드백
- 관련 관찰: 다양한 애플리케이션 구성 요소에서 발생한 문제와 관련된 로그의 오류 조각 및 지표 이상의 상세 보기

CloudWatch 에이전트 병합 충돌 실패

CloudWatch Application Insights는 고객 인스턴스에 CloudWatch 에이전트를 설치하고 구성합니다. 여기에는 지표 또는 로그에 대한 구성이 포함된 CloudWatch 에이전트 구성 파일 생성이 포함됩니다. 고객의 인스턴스에 동일한 지표 또는 로그에 대해 서로 다른 구성이 정의된 CloudWatch 에이전트 구성 파일이 이미 있는 경우 병합 충돌이 발생할 수 있습니다. 병합 충돌을 해결하려면 다음 단계를 수행합니다.

1. 시스템에서 CloudWatch 에이전트 구성 파일을 식별합니다. 파일 위치에 대한 자세한 내용은 CloudWatch 에이전트 파일 및 위치 섹션을 참조하세요.
2. 기존 CloudWatch 에이전트 구성 파일에서 Application Insights에서 사용하려는 리소스 구성을 제거합니다. Application Insights 구성만 사용하려면 기존 CloudWatch 에이전트 구성 파일을 삭제하세요.

경보가 생성되지 않음

일부 지표의 경우 Application Insights는 지표의 이전 데이터 포인트를 기반으로 경보 임곗값을 예측합니다. 이 예측을 사용하도록 설정하려면 다음 기준을 충족해야 합니다.

- 최근 데이터 포인트 - 지난 24시간 동안의 데이터 포인트가 최소 100개 이상 있어야 합니다. 해당 데이터 포인트는 연속적일 필요가 없으며 24시간의 기간 안에 분산된 값이어도 됩니다.
- 기록 데이터 - 현재 날짜 15일 전부터 현재 날짜 1일 전까지의 기간에 걸쳐 최소 100개의 데이터 포인트가 있어야 합니다. 해당 데이터 포인트는 연속적일 필요가 없으며 15일의 기간 안에 분산된 값이어도 됩니다.

> **Note**
>
> 일부 지표의 경우 Application Insights는 이전 조건이 충족될 때까지 경보 생성을 지연합니다. 이 경우 지표에 경보 임곗값을 설정하기에 충분한 데이터 포인트가 부족하다는 구성 기록 이벤트가 표시됩니다.

피드백

피드백

감지된 문제에 대해 자동으로 생성된 통찰력에 대한 유용성을 평가하여 피드백을 제공할 수 있습니다. 통찰력에 대한 피드백은 애플리케이션 진단(지표 이상 및 로그 예외)과 함께 향후 유사한 문제의 탐지를 개선하는 데 사용됩니다.

구성 오류

CloudWatch Application Insights는 구성을 사용하여 구성 요소에 대한 모니터링 원격 분석을 생성합니다. Application Insights가 계정 또는 구성과 관련된 문제를 감지하면 애플리케이션의 구성 문제를 해결하는 방법에 대한 정보를 애플리케이션 요약의 설명(Remarks) 필드에 제공합니다.

다음 표는 특정 설명에 대해 제안된 해결 방법을 보여줍니다.

설명	제안된 해결 방법	추가 참고 사항
CloudFormation 할당량에 이미 도달했습니다.	Application Insights는 모든 애플리케이션 구성 요소에 대한 CloudWatch 에이전트 설치 및 구성을 관리하기 위해 각 애플리케이션에 대해 하나의 CloudFormation 스택을 만듭니다. 기본적으로 각 AWS 계정에	해당 사항 없음

설명	제안된 해결 방법	추가 참고 사항
	2,000개의 스택이 있을 수 있습니다. AWS CloudFormation 한도를 참조합니다. 이를 해결하려면 CloudFormation 스택의 제한을 높이세요.	
다음 인스턴스에 SSM 인스턴스 역할이 없습니다.	Application Insights가 애플리케이션 인스턴스에 CloudWatch 에이전트를 설치 및 구성할 수 있으려면 AmazonSSMManagedInstanceCore 및 CloudWatchAgentServerPolicy 정책을 인스턴스 역할에 연결해야 합니다.	ApplicationInsights는 SSM DescribeInstanceInformation API를 호출하여 SSM 권한이 있는 인스턴스 목록을 가져옵니다. 인스턴스에 역할을 연결한 후 SSM이 인스턴스를 DescribeInstanceInformation 결과에 포함시키는 데는 약간의 시간이 필요합니다. SSM이 결과에 인스턴스를 포함할 때까지는 NO_SSM_INSTANCE_ROLE 오류가 애플리케이션에 남아 있습니다.
새 구성 요소에 구성이 필요할 수 있습니다.	Application Insights는 애플리케이션 리소스 그룹에 새 구성 요소가 있음을 감지합니다. 이 문제를 해결하려면 새 구성 요소를 적절히 구성하세요.	해당 사항 없음

Amazon CloudWatch Application Insights에서 지원하는 로그 및 지표

다음 목록은 Amazon CloudWatch Application Insights에서 지원되는 로그 및 지표를 보여 줍니다.

CloudWatch Application Insights는 다음 로그를 지원합니다.

- Microsoft 인터넷 정보 서비스(IIS) 로그
- EC2상의 SQL Server에 대한 오류 로그
- 사용자 지정 .NET 애플리케이션 로그(예: Log4Net)

- Windows 로그(시스템, 애플리케이션 및 보안)와 애플리케이션 및 서비스 로그가 포함된 Windows 이벤트 로그
- AWS Lambda의 Amazon CloudWatch Logs
- EC2의 RDS MySQL, Aurora MySQL 및 MySQL에 대한 오류 로그 및 느린 로그
- EC2의 PostgreSQL RDS 및 PostgreSQL에 대한 Postgresql 로그
- AWS Step Functions의 Amazon CloudWatch Logs
- API Gateway REST API 스테이지에 대한 실행 로그 및 액세스 로그(JSON, CSV, XML 단, CLF 제외)
- Prometheus JMX Exporter 로그(EMF)
- Amazon RDS의 Oracle 및 Amazon EC2의 Oracle에 대한 알림 로그 및 리스너 로그
- `awslogs` 로그 드라이버를 사용하여 Amazon ECS 컨테이너에서 CloudWatch로의 컨테이너 로그 라우팅
- FireLens 컨테이너 로그 라우터를 사용하여 Amazon ECS 컨테이너에서 CloudWatch로의 컨테이너 로그 라우팅
- Container Insights와 함께 Fluent Bit 또는 Fluentd 로그 프로세서를 사용하여 Amazon EC2에서 실행되는 Amazon EKS 또는 Kubernetes에서 CloudWatch로의 컨테이너 로그 라우팅
- SAP HANA 추적 및 오류 로그
- HA Pacemaker 로그
- SAP ASE 서버 로그
- SAP ASE 백업 서버 로그
- SAP ASE 복제 서버 로그
- SAP ASE RMA 에이전트 로그
- SAP ASE 장애 관리자 로그
- SAP NetWeaver 개발 추적 로그
- CloudWatch 에이전트용 proctstat 플러그인을 사용하는 Windows 프로세스에 대한 프로세스 지표
- 호스팅 영역의 퍼블릭 DNS 쿼리 로그
- Amazon Route 53 Resolver DNS 쿼리 로그

CloudWatch Application Insights는 다음 로그 클래스를 지원합니다.

- 표준 - Amazon CloudWatch Application Insights에서는 모니터링을 활성화하려면 CloudWatch Logs 표준 로그 클래스로 로그 그룹을 구성해야 합니다.

CloudWatch Application Insights는 다음 애플리케이션 구성 요소의 지표를 지원합니다.

- Amazon Elastic Compute Cloud(EC2)
  - CloudWatch 기본 지표
  - CloudWatch 에이전트 지표(Windows Server)
  - CloudWatch 에이전트 프로세스 지표(Windows Server)
  - CloudWatch 에이전트 지표(Linux 서버)
- Elastic Block Store(EBS)
- Amazon Elastic File System(Amazon EFS)
- Elastic Load Balancer(ELB)
- 애플리케이션 ELB
- Amazon EC2 Auto Scaling 그룹
- Amazon Simple Queue Server(SQS)
- Amazon Relational Database Service(RDS)
  - RDS 데이터베이스 인스턴스
  - RDS 데이터베이스 클러스터
- AWS Lambda 함수
- Amazon DynamoDB 테이블
- Amazon S3 버킷
- AWS Step Functions
  - 실행 수준
  - 활동
  - Lambda 함수
  - 서비스 통합
  - Step Functions API
- API Gateway REST API 스테이지
- SAP HANA
- SAP ASE
- Amazon EC2에 대한 SAP ASE 고가용성
- SAP NetWeaver
- HA 클러스터

- Java
- Amazon Elastic Container Service(Amazon ECS)
  - CloudWatch 기본 지표
  - Container Insights 지표
  - Container Insights Prometheus 지표
- AWS의 Kubernetes
  - Container Insights 지표
  - Container Insights Prometheus 지표
- Amazon FSx
- Amazon VPC
- Amazon VPC NAT 게이트웨이
- Amazon Route 53 상태 확인
- Amazon Route 53 호스팅 영역
- Amazon Route 53 Resolver 엔드포인트
- AWS Network Firewall 규칙 그룹
- AWS Network Firewall 규칙 그룹 연결
- 데이터 포인트 요구 사항이 있는 지표
  - AWS/ApplicationELB
  - AWS/AutoScaling
  - AWS/EC2
  - Elastic Block Store(EBS)
  - AWS/ELB
  - AWS/RDS
  - AWS/Lambda
  - AWS/SQS
  - AWS/CWAgent
  - AWS/DynamoDB
  - AWS/S3
  - AWS/States
  - AWS/ApiGateway

  - AWS/SNS
- 권장 지표
- 성능 카운터 지표

Amazon Elastic Compute Cloud(EC2)

CloudWatch Application Insights는 다음 지표를 지원합니다.

지표

- CloudWatch 기본 지표
- CloudWatch 에이전트 지표(Windows Server)
- CloudWatch 에이전트 프로세스 지표(Windows Server)
- CloudWatch 에이전트 지표(Linux 서버)

CloudWatch 기본 지표

CPUCreditBalance

CPUCreditUsage

CPUSurplusCreditBalance

CPUSurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBSByteBalance%

EBSIOBalance%

EBSReadBytes

EBSReadOps

EBSWriteBytes

EBSWriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed_Instance

StatusCheckFailed_System

CloudWatch 에이전트 지표(Windows Server)

.NET CLR 예외 발생된 예외 수

.NET CLR 예외 발생된 예외 수/초

.NET CLR 예외 필터 수/초

.NET CLR 예외 최종 수/초

.NET CLR 예외 발생 깊이 파악/초

.NET CLR Interop CCW 수

.NET CLR Interop 스텁 수

.NET CLR Interop TLB 내보내기 횟수/초

.NET CLR Interop TLB 가져오기 횟수/초

.NET CLR Interop 마샬링 수

.NET CLR Jit Jit 시간(%)

.NET CLR Jit 표준 Jit 실패

.NET CLR 로딩 로딩 시간(%)

.NET CLR 로딩 로드 실패 비율

.NET CLR LocksAndThreads 경합 속도/초

.NET CLR LocksAndThreads 대기열 길이/초

.NET CLR 메모리 수 총 커밋된 바이트 수

.NET CLR 메모리 GC의 시간(%)

.NET CLR 네트워킹 4.0.0.0 HttpWebRequest 평균 대기열 시간

.NET CLR 네트워킹 4.0.0.0 중단된 HttpWebRequests/초

.NET CLR 네트워킹 4.0.0.0 실패한 HttpWebRequests/초

.NET CLR 네트워킹 4.0.0.0 대기 중인 HttpWebRequests/초

APP_POOL_WAS Total Worker Process Ping Failures

ASP.NET 응용 프로그램 재시작

ASP.NET Applications % Managed Processor Time(estimated)

ASP.NET 응용 프로그램 오류 합계/초

ASP.NET 응용 프로그램 오류 실행 중 처리되지 않은 오류/초

ASP.NET 응용 프로그램 응용 프로그램 대기열의 요청

ASP.NET 응용 프로그램 요청/초

ASP.NET 요청 대기 시간

ASP.NET 대기 중인 요청

HTTP 서비스 요청 대기열 CurrentQueueSize

LogicalDisk % 사용 가능한 공간

사용 중인 메모리 % 커밋된 바이트

사용 가능한 메모리(MB)

메모리 페이지/초

네트워크 인터페이스 바이트 합계/초

페이징된 파일 % 사용량

PhysicalDisk % 디스크 시간

PhysicalDisk 평균 디스크 대기열 길이

PhysicalDisk 평균 디스크 초/읽기

PhysicalDisk 평균 디스크 초/쓰기

PhysicalDisk 디스크 읽기 바이트/초

PhysicalDisk 디스크 읽기/초

PhysicalDisk 디스크 쓰기 바이트/초

PhysicalDisk 디스크 쓰기/초

프로세서 % 유휴 시간

프로세서 % 중단 시간

프로세서 % 프로세서 시간

프로세서 % 사용자 시간

SQLServer:액세스 방법 전달된 레코드/초

SQL Server:액세스 방법 전체 스캔/초

SQLServer:액세스 방법 페이지 분할/초

SQLServer:버퍼 관리자 버퍼 캐시 적중률

SQLServer:버퍼 관리자 페이지 예상 수명

SQLServer:차단된 일반 통계 프로세스

SQLServer:일반 통계 사용자 연결

SQLServer:래치 평균 래치 대기 시간(ms)

SQLServer:잠금 평균 대기 시간(ms)

SQLServer:잠금 잠금 제한 시간/초

SQLServer:잠금 잠금 대기/초

SQLServer:잠금 교착 상태 수/초

SQLServer:메모리 관리자 보류 중인 메모리 부여

SQLServer:SQL 통계 배치 요청/초

SQLServer:SQL 통계 SQL 컴파일/초

SQLServer:SQL 통계 SQL 재컴파일/초

시스템 프로세서 대기열 길이

TCPv4 연결 설정

TCPv6 연결 설정

W3SVC_W3WP 파일 캐시 플러시

W3SVC_W3WP 파일 캐시 누락

W3SVC_W3WP 요청/초

W3SVC_W3WP URI 캐시 플러시

W3SVC_W3WP URI 캐시 누락

웹 서비스 수신된 바이트/초

웹 서비스 전송된 바이트/초

웹 서비스 연결 시도/초

웹 서비스 현재 연결

웹 서비스 요청 가져오기/초

웹 서비스 POST 요청/초

Bytes Received/sec

Normal Messages Queue Length/sec

Urgent Message Queue Length/sec

Reconnect Count

Unacknowledged Message Queue Length/sec

Messages Outstanding

Messages Sent/sec

Database Update Messages/sec

Update Messages/sec

Flushes/sec

Crypto Checkpoints Saved/sec

Crypto Checkpoints Restored/sec

Registry Checkpoints Restored/sec

Registry Checkpoints Saved/sec

Cluster API Calls/sec

Resource API Calls/sec

Cluster Handles/sec

Resource Handles/sec

CloudWatch 에이전트 프로세스 지표(Windows Server)

프로세스 지표는 CloudWatch 에이전트 procstat 플러그인을 사용하여 수집됩니다. Windows 워크로드를 실행하는 Amazon EC2 인스턴스만 프로세스 지표를 지원합니다.

procstat cpu_time_system

procstat cpu_time_user

procstat cpu_usage

procstat memory_rss

procstat memory_vms

procstat read_bytes

procstat write_bytes

.procstat read_count

procstat write_count

CloudWatch 에이전트 지표(Linux 서버)

cpu_time_active

cpu_time_guest

cpu_time_guest_nice

cpu_time_idle

cpu_time_iowait

cpu_time_irq

cpu_time_nice

cpu_time_softirq

cpu_time_steal

cpu_time_system

cpu_time_user

cpu_usage_active

cpu_usage_guest

cpu_usage_guest_nice

cpu_usage_idle

cpu_usage_iowait

cpu_usage_irq

cpu_usage_nice

cpu_usage_softirq

cpu_usage_steal

cpu_usage_system

cpu_usage_user

disk_free

disk_inodes_free

disk_inodes_used

disk_used

disk_used_percent

diskio_io_time

diskio_iops_in_progress

diskio_read_bytes

diskio_read_time

diskio_reads

diskio_write_bytes

diskio_write_time

diskio_writes

mem_active

mem_available

mem_available_percent

mem_buffered

mem_cached

mem_free

mem_inactive

mem_used

mem_used_percent

net_bytes_recv

net_bytes_sent

net_drop_in

net_drop_out

net_err_in

net_err_out

net_packets_recv

net_packets_sent

netstat_tcp_close

netstat_tcp_close_wait

netstat_tcp_closing

netstat_tcp_established

netstat_tcp_fin_wait1

netstat_tcp_fin_wait2

netstat_tcp_last_ack

netstat_tcp_listen

netstat_tcp_none

netstat_tcp_syn_recv

netstat_tcp_syn_sent

netstat_tcp_time_wait

netstat_udp_socket

processes_blocked

processes_dead

processes_idle

processes_paging

processes_running

processes_sleeping

processes_stopped

processes_total

processes_total_threads

processes_wait

processes_zombies

swap_free

swap_used

swap_used_percent

Elastic Block Store(EBS)

CloudWatch Application Insights는 다음 지표를 지원합니다.

VolumeReadBytes

VolumeWriteBytes

VolumeReadOps

VolumeWriteOps

VolumeTotalReadTime

VolumeTotalWriteTime

VolumeIdleTime

VolumeQueueLength

VolumeThroughputPercentage

VolumeConsumedReadWriteOps

BurstBalance

Amazon Elastic File System(Amazon EFS)

CloudWatch Application Insights는 다음 지표를 지원합니다.

버스트 크레딧 밸런스

입출력 제한 수치

허용된 처리량

허용 처리량 사용률

총 입출력 바이트 수

데이터 쓰기 입출력 바이트

데이터 읽기 입출력 바이트

메타데이터 입출력 바이트

클라이언트 연결

마지막 동기화 이후 흐른 시간

StorageBytes

처리량

PercentageOfPermittedThroughputUtilization

ThroughputIOPS

PercentThroughputDataReadIOByte

PercentThroughputDataWriteIOBytes

PercentageOfIOPSDataReadIOBytes

PercentageOfIOPSDataWriteIOBytes

AverageDataReadIOBytesSize

AverageDataWriteIOBytesSize

Elastic Load Balancer(ELB)

CloudWatch Application Insights는 다음 지표를 지원합니다.

EstimatedALBActiveConnectionCount

EstimatedALBConsumedLCUs

EstimatedALBNewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

RequestCount

UnHealthyHostCount

애플리케이션 ELB

CloudWatch Application Insights는 다음 지표를 지원합니다.

EstimatedALBActiveConnectionCount

EstimatedALBConsumedLCUs

EstimatedALBNewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

지연 시간

RequestCount

SurgeQueueLength

UnHealthyHostCount

Amazon EC2 Auto Scaling 그룹

CloudWatch Application Insights는 다음 지표를 지원합니다.

CPUCreditBalance

CPUCreditUsage

CPUSurplusCreditBalance

CPUSurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBSByteBalance%

EBSIOBalance%

EBSReadBytes

EBSReadOps

EBSWriteBytes

EBSWriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed_Instance

StatusCheckFailed_System

Amazon Simple Queue Server(SQS)

CloudWatch Application Insights는 다음 지표를 지원합니다.

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

Amazon Relational Database Service(RDS)

CloudWatch Application Insights는 다음 지표를 지원합니다.

지표

- RDS 데이터베이스 인스턴스
- RDS 데이터베이스 클러스터

RDS 데이터베이스 인스턴스

BurstBalance

CPUCreditBalance

CPUUtilization

DatabaseConnections

DiskQueueDepth

FailedSQLServerAgentJobsCount

FreeStorageSpace

FreeableMemory

NetworkReceiveThroughput

NetworkTransmitThroughput

ReadIOPS

ReadLatency

ReadThroughput

WriteIOPS

WriteLatency

WriteThroughput

RDS 데이터베이스 클러스터

ActiveTransactions

AuroraBinlogReplicaLag

AuroraReplicaLag

BackupRetentionPeriodStorageUsed

BinLogDiskUsage

BlockedTransactions

BufferCacheHitRatio

CPUUtilization

CommitLatency

CommitThroughput

DDLLatency

DDLThroughput

DMLLatency

DMLThroughput

DatabaseConnections

교착

DeleteLatency

DeleteThroughput

EngineUptime

FreeLocalStorage

FreeableMemory

InsertLatency

InsertThroughput

LoginFailures

NetworkReceiveThroughput

NetworkThroughput

NetworkTransmitThroughput

쿼리

ResultSetCacheHitRatio

SelectLatency

SelectThroughput

SnapshotStorageUsed

TotalBackupStorageBilled

UpdateLatency

UpdateThroughput

VolumeBytesUsed

VolumeReadIOPs

VolumeWriteIOPs

AWS Lambda 함수

CloudWatch Application Insights는 다음 지표를 지원합니다.

Errors

DeadLetterErrors

지속 시간

제한

IteratorAge

ProvisionedConcurrencySpilloverInvocations

Amazon DynamoDB 테이블

CloudWatch Application Insights는 다음 지표를 지원합니다.

SystemErrors

UserErrors

ConsumedReadCapacityUnits

ConsumedWriteCapacityUnits

ReadThrottleEvents

WriteThrottleEvents

TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount

ReplicationLatency

Amazon S3 버킷

CloudWatch Application Insights는 다음 지표를 지원합니다.

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxErrors

5xxErrors

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

AWS Step Functions

CloudWatch Application Insights는 다음 지표를 지원합니다.

지표

- 실행 수준
- 활동
- Lambda 함수
- 서비스 통합
- Step Functions API

실행 수준

ExecutionTime

ExecutionThrottled

ExecutionsFailed

ExecutionsTimedOut

ExecutionsAborted

ExecutionsSucceeded

ExecutionsStarted

활동

ActivityRunTime

ActivityScheduleTime

ActivityTime

ActivitiesFailed

ActivitiesHeartbeatTimedOut

ActivitiesTimedOut

ActivitiesScheduled

ActivitiesSucceeded

ActivitiesStarted

Lambda 함수

LambdaFunctionRunTime

LambdaFunctionScheduleTime

LambdaFunctionTime

LambdaFunctionsFailed

LambdaFunctionsTimedOut

LambdaFunctionsScheduled

LambdaFunctionsSucceeded

LambdaFunctionsStarted

서비스 통합

ServiceIntegrationRunTime

ServiceIntegrationScheduleTime

ServiceIntegrationTime

ServiceIntegrationsFailed

ServiceIntegrationsTimedOut

ServiceIntegrationsScheduled

ServiceIntegrationsSucceeded

ServiceIntegrationsStarted

Step Functions API

ThrottledEvents

ProvisionedBucketSize

ProvisionedRefillRate

ConsumedCapacity

API Gateway REST API 스테이지

CloudWatch Application Insights는 다음 지표를 지원합니다.

4XXError

5XXError

IntegrationLatency

지연 시간

CacheHitCount

CacheMissCount

SAP HANA

> **Note**
>
> CloudWatch Application Insights는 단일 SID HANA 환경만 지원합니다. 여러 HANA SID가 연결된 경우, 첫 번째로 감지된 SID에 대해서만 모니터링이 설정됩니다.

CloudWatch Application Insights는 다음 지표를 지원합니다.

hanadb_every_service_started_status

hanadb_daemon_service_started_status

hanadb_preprocessor_service_started_status

hanadb_webdispatcher_service_started_status

hanadb_compileserver_service_started_status

hanadb_nameserver_service_started_status

hanadb_server_startup_time_variations_seconds

hanadb_level_5_alerts_count

hanadb_level_4_alerts_count

hanadb_out_of_memory_events_count

hanadb_max_trigger_read_ratio_percent

hanadb_max_trigger_write_ratio_percent

hanadb_log_switch_wait_ratio_percent

hanadb_log_switch_race_ratio_percent

hanadb_time_since_last_savepoint_seconds

hanadb_disk_usage_highlevel_percent

hanadb_max_converter_page_number_count

hanadb_long_running_savepoints_count

hanadb_failed_io_reads_count

hanadb_failed_io_writes_count

hanadb_disk_data_unused_percent

hanadb_current_allocation_limit_used_percent

hanadb_table_allocation_limit_used_percent

hanadb_host_total_physical_memory_mb

hanadb_host_physical_memory_used_mb

hanadb_host_physical_memory_free_mb

hanadb_swap_memory_free_mb

hanadb_swap_memory_used_mb

hanadb_host_allocation_limit_mb

hanadb_host_total_memory_used_mb

hanadb_host_total_peak_memory_used_mb

hanadb_host_total_allocation_limit_mb

hanadb_host_code_size_mb

hanadb_host_shared_memory_allocation_mb

hanadb_cpu_usage_percent

hanadb_cpu_user_percent

hanadb_cpu_system_percent

hanadb_cpu_waitio_percent

hanadb_cpu_busy_percent

hanadb_cpu_idle_percent

hanadb_long_delta_merge_count

hanadb_unsuccessful_delta_merge_count

hanadb_successful_delta_merge_count

hanadb_row_store_allocated_size_mb

hanadb_row_store_free_size_mb

hanadb_row_store_used_size_mb

hanadb_temporary_tables_count

hanadb_large_non_compressed_tables_count

hanadb_total_non_compressed_tables_count

hanadb_longest_running_job_seconds

hanadb_average_commit_time_milliseconds

hanadb_suspended_sql_statements_count

hanadb_plan_cache_hit_ratio_percent

hanadb_plan_cache_lookup_count

hanadb_plan_cache_hit_count

hanadb_plan_cache_total_execution_microseconds

hanadb_plan_cache_cursor_duration_microseconds

hanadb_plan_cache_preparation_microseconds

hanadb_plan_cache_evicted_count

hanadb_plan_cache_evicted_microseconds

hanadb_plan_cache_evicted_preparation_count

hanadb_plan_cache_evicted_execution_count

hanadb_plan_cache_evicted_preparation_microseconds

hanadb_plan_cache_evicted_cursor_duration_microseconds

hanadb_plan_cache_evicted_total_execution_microseconds

hanadb_plan_cache_evicted_plan_size_mb

hanadb_plan_cache_count

hanadb_plan_cache_preparation_count

hanadb_plan_cache_execution_count

hanadb_network_collision_rate

hanadb_network_receive_rate

hanadb_network_transmit_rate

hanadb_network_packet_receive_rate

hanadb_network_packet_transmit_rate

hanadb_network_transmit_error_rate

hanadb_network_receive_error_rate

hanadb_time_until_license_expires_days

hanadb_is_license_valid_status

hanadb_local_running_connections_count

hanadb_local_idle_connections_count

hanadb_remote_running_connections_count

hanadb_remote_idle_connections_count

hanadb_last_full_data_backup_age_days

hanadb_last_data_backup_age_days

hanadb_last_log_backup_age_hours

hanadb_failed_data_backup_past_7_days_count

hanadb_failed_log_backup_past_7_days_count

hanadb_oldest_backup_in_catalog_age_days

hanadb_backup_catalog_size_mb

hanadb_hsr_replication_status

hanadb_hsr_log_shipping_delay_seconds

hanadb_hsr_secondary_failover_count

hanadb_hsr_secondary_reconnect_count

hanadb_hsr_async_buffer_used_mb

hanadb_hsr_secondary_active_status

hanadb_handle_count

hanadb_ping_time_milliseconds

hanadb_connection_count

hanadb_internal_connection_count

hanadb_external_connection_count

hanadb_idle_connection_count

hanadb_transaction_count

hanadb_internal_transaction_count

hanadb_external_transaction_count

hanadb_user_transaction_count

hanadb_blocked_transaction_count

hanadb_statement_count

hanadb_active_commit_id_range_count

hanadb_mvcc_version_count

hanadb_pending_session_count

hanadb_record_lock_count

hanadb_read_count

hanadb_write_count

hanadb_merge_count

hanadb_unload_count

hanadb_active_thread_count

hanadb_waiting_thread_count

hanadb_total_thread_count

hanadb_active_sql_executor_count

hanadb_waiting_sql_executor_count

hanadb_total_sql_executor_count

hanadb_data_write_size_mb

hanadb_data_write_time_milliseconds

hanadb_log_write_size_mb

hanadb_log_write_time_milliseconds

hanadb_data_read_size_mb

hanadb_data_read_time_milliseconds

hanadb_log_read_size_mb

hanadb_log_read_time_milliseconds

hanadb_data_backup_write_size_mb

hanadb_data_backup_write_time_milliseconds

hanadb_log_backup_write_size_mb

hanadb_log_backup_write_time_milliseconds

hanadb_mutex_collision_count

hanadb_read_write_lock_collision_count

hanadb_admission_control_admit_count

hanadb_admission_control_reject_count

hanadb_admission_control_queue_size_mb

hanadb_admission_control_wait_time_milliseconds

SAP ASE

CloudWatch Application Insights는 다음 지표를 지원합니다.

asedb_database_availability

asedb_trunc_log_on_chkpt_enabled

asedb_last_db_backup_age_in_days

asedb_last_transaction_log_backup_age_in_hours

asedb_suspected_database

asedb_db_space_usage_percent

asedb_db_log_space_usage_percent

asedb_locked_login

asedb_has_mixed_log_and_data

asedb_runtime_for_open_transactions

asedb_data_cache_hit_ratio

asedb_data_cache_usage

asedb_sql_cache_hit_ratio

asedb_cache_usage

asedb_run_queue_length

asedb_number_of_rollbacks

asedb_number_of_commits

asedb_number_of_transactions

asedb_outstanding_disk_io

asedb_percent_io_busy

asedb_percent_system_busy

asedb_percent_locks_active

asedb_scheduled_jobs_failed_percent

asedb_user_connections_percent

asedb_query_logical_reads

asedb_query_physical_reads

asedb_query_cpu_time

asedb_query_memory_usage

Amazon EC2에 대한 SAP ASE 고가용성

CloudWatch Application Insights는 다음 지표를 지원합니다.

asedb_ha_replication_state

asedb_ha_replication_mode

asedb_ha_replication_latency_in_minutes

SAP NetWeaver

CloudWatch Application Insights는 다음 지표를 지원합니다.

지표	설명
sap_alerts_ResponseTime	CCMS(RZ20)>R3Services>Dialog>ResponseTime의 SAP 응답 시간 알림.
sap_alerts_ResponseTimeDialog	CCMS(RZ20)>R3Services>Dialog> ResponseTimeDialog의 SAP 응답 시간 대화 상자 경고.
sap_alerts_ResponseTimeDialogRFC	CCMS(RZ20)>R3Services> Dialog>ResponseTimeDialogRFC의 SAP 응답 시간 알림.
sap_alerts_DBRequestTime	CCMS(RZ20)>R3Services>Dialog>DBRequestTime의 SAP 응답 시간 알림.
sap_alerts_FrontendResponseTime	CCMS(RZ20)>R3Services > Dialog>FrontEndResponseTime의 SAP 응답 시간 알림.
sap_alerts_Database	SAP 시스템에서 데이터베이스 관련 오류를 기록했습니다. SM21 또는 CCMS(RZ20)>R3Syslog>Database의 알림.
sap_alerts_QueueTime	CCMS(RZ20)>R3Services>Dialog>QueueTime의 SAP 대기열 시간 알림.
sap_alerts_AbortedJobs	SAP 시스템에서 실패한 백그라운드 작업. (RZ20)>R3Services > Background>Aborted Jobs의 알림.
sap_alerts_BasisSystem	SAP 시스템에서 시스템 수준 오류를 기록했습니다. SM21 또는 CCMS(RZ20)>R3Syslog>BasisSystem의 알림.

지표	설명
sap_alerts_Security	SAP 시스템에서 보안 관련 메시지를 기록했습니다. SM21 또는 CCMS(RZ20)>R3Syslog>Security의 알림.
sap_alerts_System	SAP 시스템에서 보안 또는 감사 관련 메시지를 기록했습니다. SM21 또는 CCMS(RZ20)>Security>System의 알림.
sap_alerts_LongRunners	SAP 시스템에 장기 실행 프로그램이 있습니다. CCMS(RZ20)>R3Services > Dialog>LongRunners의 알림.
sap_alerts_SqlError	SAP 데이터베이스 클라이언트 계층 오류 로그가 있습니다. CCMS(RZ20)>DatabaseClient>AbapSql>SqlError의 알림.
sap_alerts_State	CCMS (RZ20)>OS Collector>State의 상태 알림.
sap_alerts_Shortdumps	ST22 및 CCMS(RZ20)>R3Abap>Shortdumps의 Shortdumps 알림.
sap_alerts_Availability	SM21, SM50, SM51, SM66 및 CCMS(RZ20)>InstanceAsTask>Availability의 SAP 애플리케이션 서버 인스턴스에 대한 가용성 알림.
sap_dispatcher_queue_high	SAPControl 웹 서비스 함수 `GetQueueStatistic` 은 디스패처 대기열 높음 개수를 제공합니다.
sap_dispatcher_queue_max	SAPControl 웹 서비스 함수 `GetQueueStatistic` 은 디스패처 대기열 최대 개수를 제공합니다.
sap_dispatcher_queue_now	SAPControl 웹 서비스 함수 `GetQueueStatistic` 은 디스패처 대기열 현재 개수를 제공합니다.

지표	설명
sap_dispatcher_queue_reads	SAPControl 웹 서비스 함수 `GetQueueStatistic` 은 디스패처 대기열 읽기 개수를 제공합니다.
sap_dispatcher_queue_writes	SAPControl 웹 서비스 함수 `GetQueueStatistic` 은 디스패처 대기열 쓰기 개수를 제공합니다.
sap_enqueue_server_arguments_high	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 인수 높음을 제공합니다.
sap_enqueue_server_arguments_max	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 인수 최대를 제공합니다.
sap_enqueue_server_arguments_now	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 인수 지금을 제공합니다.
sap_enqueue_server_arguments_state	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 인수 상태를 제공합니다.
sap_enqueue_server_backup_requests	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 백업 요청을 제공합니다.
sap_enqueue_server_cleanup_requests	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 정리 요청을 제공합니다.
sap_enqueue_server_dequeue_all_requests	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에서 제거 모든 요청을 제공합니다.

지표	설명
sap_enqueue_server_dequeue_errors	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에서 제거 오류를 제공합니다.
sap_enqueue_server_dequeue_requests	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에서 제거 요청을 제공합니다.
sap_enqueue_server_enqueue_errors	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 오류를 제공합니다.
sap_enqueue_server_enqueue_rejects	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 거부를 제공합니다.
sap_enqueue_server_enqueue_requests	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 요청을 제공합니다.
sap_enqueue_server_lock_time	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 잠금 시간을 제공합니다.
sap_enqueue_server_lock_wait_time	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 잠금 대기 시간을 제공합니다.
sap_enqueue_server_locks_high	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 잠금 높음을 제공합니다.
sap_enqueue_server_locks_max	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 잠금 최대를 제공합니다.
sap_enqueue_server_locks_now	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 잠금 지금을 제공합니다.

지표	설명
sap_enqueue_server_locks_state	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 잠금 상태를 제공합니다.
sap_enqueue_server_owner_high	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 소유자 높음을 제공합니다.
sap_enqueue_server_owner_max	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 소유자 최대를 제공합니다.
sap_enqueue_server_owner_now	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 소유자 지금을 제공합니다.
sap_enqueue_server_owner_state	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 소유자 상태를 제공합니다.
sap_enqueue_server_replication_state	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 대기열에 추가 복제 상태 상태를 제공합니다.
sap_enqueue_server_reporting_requests	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 보고 요청 상태를 제공합니다.
sap_enqueue_server_server_time	SAPControl 웹 서비스 함수 `EnqGetStatistic` 은 Enqueue Server 시간을 제공합니다.
sap_HA_check_failover_config_state	SAPControl 웹 서비스 함수 `HACheckFailoverConfig` 는 SAP High Availability 상태를 제공합니다.

지표	설명
sap_HA_get_failover_config_HAActive	SAPControl 웹 서비스 함수 `HAGetFailoverConfig` 는 SAP High Availability 클러스터 구성 및 상태를 제공합니다.
sap_start_service_processes	SAPControl 웹 서비스 함수 `GetProcessList` 는 disp+work, IGS, gwrd, icman, 메시지 서버, Enqueue Server 프로세스 상태를 제공합니다.

HA 클러스터

CloudWatch Application Insights는 다음 지표를 지원합니다.

ha_cluster_pacemaker_stonith_enabled

ha_cluster_corosync_quorate

hanadb_webdispatcher_service_started_status

ha_cluster_pacemaker_nodes

ha_cluster_corosync_ring_errors

ha_cluster_pacemaker_fail_count

Java

CloudWatch Application Insights는 다음 지표를 지원합니다.

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_threading_daemonthreadcount

java_lang_classloading_loadedclasscount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

Amazon Elastic Container Service(Amazon ECS)

CloudWatch Application Insights는 다음 지표를 지원합니다.

지표

- CloudWatch 기본 지표
- Container Insights 지표
- Container Insights Prometheus 지표

CloudWatch 기본 지표

CPUReservation

CPUUtilization

MemoryReservation

MemoryUtilization

GPUReservation

Container Insights 지표

ContainerInstanceCount

CpuUtilized

CpuReserved

DeploymentCount

DesiredTaskCount

MemoryUtilized

MemoryReserved

NetworkRxBytes

NetworkTxBytes

PendingTaskCount

RunningTaskCount

ServiceCount

StorageReadBytes

StorageWriteBytes

TaskCount

TaskSetCount

instance_cpu_limit

instance_cpu_reserved_capacity

instance_cpu_usage_total

instance_cpu_utilization

instance_filesystem_utilization

instance_memory_limit

instance_memory_reserved_capacity

instance_memory_utilization

instance_memory_working_set

instance_network_total_bytes

instance_number_of_running_tasks

Container Insights Prometheus 지표

Java JMX 지표

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_classloading_loadedclasscount

java_lang_threading_daemonthreadcount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

AWS의 Kubernetes

CloudWatch Application Insights는 다음 지표를 지원합니다.

지표

- Container Insights 지표
- Container Insights Prometheus 지표

Container Insights 지표

cluster_failed_node_count

cluster_node_count

namespace_number_of_running_pods

node_cpu_limit

node_cpu_reserved_capacity

node_cpu_usage_total

node_cpu_utilization

node_filesystem_utilization

node_memory_limit

node_memory_reserved_capacity

node_memory_utilization

node_memory_working_set

node_network_total_bytes

node_number_of_running_containers

node_number_of_running_pods

pod_cpu_reserved_capacity

pod_cpu_utilization

pod_cpu_utilization_over_pod_limit

pod_memory_reserved_capacity

pod_memory_utilization

pod_memory_utilization_over_pod_limit

pod_network_rx_bytes

pod_network_tx_bytes

service_number_of_running_pods

Container Insights Prometheus 지표

Java JMX 지표

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_classloading_loadedclasscount

java_lang_threading_daemonthreadcount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

Amazon FSx

CloudWatch Application Insights는 다음 지표를 지원합니다.

DataReadBytes

DataWriteBytes

DataReadOperations

DataWriteOperations

MetadataOperations

FreeStorageCapacity

FreeDataStorageCapacity

LogicalDiskUsage

PhysicalDiskUsage

Amazon VPC

CloudWatch Application Insights는 다음 지표를 지원합니다.

NetworkAddressUsage

NetworkAddressUsagePeered

VPCFirewallQueryVolume

Amazon VPC NAT 게이트웨이

CloudWatch Application Insights는 다음 지표를 지원합니다.

ErrorPortAllocation

IdleTimeoutCount

Amazon Route 53 상태 확인

CloudWatch Application Insights는 다음 지표를 지원합니다.

ChildHealthCheckHealthyCount

ConnectionTime

HealthCheckPercentageHealthy

HealthCheckStatus

SSLHandshakeTime

TimeToFirstByte

Amazon Route 53 호스팅 영역

CloudWatch Application Insights는 다음 지표를 지원합니다.

DNSQueries

DNSSECInternalFailure

DNSSECKeySigningKeysNeedingAction

DNSSECKeySigningKeyMaxNeedingActionAge

DNSSECKeySigningKeyAge

Amazon Route 53 Resolver 엔드포인트

CloudWatch Application Insights는 다음 지표를 지원합니다.

EndpointHealthyENICount

EndpointUnHealthyENICount

InboundQueryVolume

OutboundQueryVolume

OutboundQueryAggregateVolume

AWS Network Firewall 규칙 그룹

CloudWatch Application Insights는 다음 지표를 지원합니다.

FirewallRuleGroupQueryVolume

AWS Network Firewall 규칙 그룹 연결

CloudWatch Application Insights는 다음 지표를 지원합니다.

FirewallRuleGroupVpcQueryVolume

데이터 포인트 요구 사항이 있는 지표

경보가 발생하는 명확한 기본 임곗값이 없는 지표의 경우, Application Insights는 경보가 발생하는 데 필요한 임곗값을 예측할 수 있을 만큼의 충분한 데이터 포인트가 지표에 확보될 때까지 기다립니다. 경보가 생성되기 전에 CloudWatch Application Insights가 확인하는 지표 데이터 포인트 요구 사항은 다음과 같습니다.

- 지표에 지난 2~15일 동안의 데이터 포인트가 100개 이상 있는 경우

- 지표에 마지막 날 이후 데이터 포인트가 100개 이상 있는 경우

다음 지표는 이러한 데이터 포인트 요구 사항을 따릅니다. CloudWatch 에이전트 지표는 경보를 생성하는 데 최대 1시간이 걸립니다.

지표

- AWS/ApplicationELB
- AWS/AutoScaling
- AWS/EC2
- Elastic Block Store(EBS)
- AWS/ELB
- AWS/RDS
- AWS/Lambda
- AWS/SQS
- AWS/CWAgent
- AWS/DynamoDB
- AWS/S3
- AWS/States
- AWS/ApiGateway
- AWS/SNS

AWS/ApplicationELB

ActiveConnectionCount

ConsumedLCUs

HTTPCode_ELB_4XX_Count

HTTPCode_Target_2XX_Count

HTTPCode_Target_3XX_Count

HTTPCode_Target_4XX_Count

HTTPCode_Target_5XX_Count

NewConnectionCount

ProcessedBytes

TargetResponseTime

UnHealthyHostCount

AWS/AutoScaling

GroupDesiredCapacity

GroupInServiceInstances

GroupMaxSize

GroupMinSize

GroupPendingInstances

GroupStandbyInstances

GroupTerminatingInstances

GroupTotalInstances

AWS/EC2

CPUCreditBalance

CPUCreditUsage

CPUSurplusCreditBalance

CPUSurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBSByteBalance%

EBSIOBalance%

EBSReadBytes

EBSReadOps

EBSWriteBytes

EBSWriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

Elastic Block Store(EBS)

VolumeReadBytes

VolumeWriteBytes

VolumeReadOps

VolumeWriteOps

VolumeTotalReadTime

VolumeTotalWriteTime

VolumeIdleTime

VolumeQueueLength

VolumeThroughputPercentage

VolumeConsumedReadWriteOps

BurstBalance

AWS/ELB

EstimatedALBActiveConnectionCount

EstimatedALBConsumedLCUs

EstimatedALBNewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

지연 시간

RequestCount

SurgeQueueLength

UnHealthyHostCount

AWS/RDS

ActiveTransactions

AuroraBinlogReplicaLag

AuroraReplicaLag

BackupRetentionPeriodStorageUsed

BinLogDiskUsage

BlockedTransactions

CPUCreditBalance

CommitLatency

CommitThroughput

DDLLatency

DDLThroughput

DMLLatency

DMLThroughput

DatabaseConnections

교착

DeleteLatency

DeleteThroughput

DiskQueueDepth

EngineUptime

FreeLocalStorage

FreeStorageSpace

FreeableMemory

InsertLatency

InsertThroughput

LoginFailures

NetworkReceiveThroughput

NetworkThroughput

NetworkTransmitThroughput

쿼리

ReadIOPS

ReadThroughput

SelectLatency

SelectThroughput

SnapshotStorageUsed

TotalBackupStorageBilled

UpdateLatency

UpdateThroughput

VolumeBytesUsed

VolumeReadIOPs

VolumeWriteIOPs

WriteIOPS

WriteThroughput

AWS/Lambda

Errors

DeadLetterErrors

지속 시간

제한

IteratorAge

ProvisionedConcurrencySpilloverInvocations

AWS/SQS

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

AWS/CWAgent

LogicalDisk % 사용 가능한 공간

사용 중인 메모리 % 커밋된 바이트

사용 가능한 메모리(MB)

네트워크 인터페이스 바이트 합계/초

페이징된 파일 % 사용량

PhysicalDisk % 디스크 시간

PhysicalDisk 평균 디스크 초/읽기

PhysicalDisk 평균 디스크 초/쓰기

PhysicalDisk 디스크 읽기 바이트/초

PhysicalDisk 디스크 읽기/초

PhysicalDisk 디스크 쓰기 바이트/초

PhysicalDisk 디스크 쓰기/초

프로세서 % 유휴 시간

프로세서 % 중단 시간

프로세서 % 프로세서 시간

프로세서 % 사용자 시간

SQLServer:액세스 방법 전달된 레코드/초

SQLServer:액세스 방법 페이지 분할/초

SQLServer:버퍼 관리자 버퍼 캐시 적중률

SQLServer:버퍼 관리자 페이지 예상 수명

SQLServer:수신된 데이터베이스 복제본 파일 바이트/초

SQLServer:수신된 데이터베이스 복제본 로그 바이트/초

SQLServer:실행 취소를 위한 나머지 데이터베이스 복제본 로그

SQLServer:데이터베이스 복제본 로그 전송 대기열

SQLServer:데이터베이스 복제본 미러링된 쓰기 트랜잭션/초

SQLServer:데이터베이스 복제본 복구 대기열

SQLServer:나머지 데이터베이스 복제본 다시 실행 바이트

SQLServer:데이터베이스 복제본 다시 실행 바이트/초

SQLServer:실행 취소가 필요한 데이터베이스 복제본 총 로그

SQLServer:데이터베이스 복제본 트랜잭션 지연

SQLServer:차단된 일반 통계 프로세스

SQLServer:SQL 통계 배치 요청/초

SQLServer:SQL 통계 SQL 컴파일/초

SQLServer:SQL 통계 SQL 재컴파일/초

시스템 프로세서 대기열 길이

TCPv4 연결 설정

TCPv6 연결 설정

AWS/DynamoDB

ConsumedReadCapacityUnits

ConsumedWriteCapacityUnits

ReadThrottleEvents

WriteThrottleEvents

TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount

ReplicationLatency

AWS/S3

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxErrors

5xxErrors

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

AWS/States

ActivitiesScheduled

ActivitiesStarted

ActivitiesSucceeded

ActivityScheduleTime

ActivityRuntime

ActivityTime

LambdaFunctionsScheduled

LambdaFunctionsStarted

LambdaFunctionsSucceeded

LambdaFunctionScheduleTime

LambdaFunctionRuntime

LambdaFunctionTime

ServiceIntegrationsScheduled

ServiceIntegrationsStarted

ServiceIntegrationsSucceeded

ServiceIntegrationScheduleTime

ServiceIntegrationRuntime

ServiceIntegrationTime

ProvisionedRefillRate

ProvisionedBucketSize

ConsumedCapacity

ThrottledEvents

AWS/ApiGateway

4XXError

IntegrationLatency

지연 시간

DataProcessed

CacheHitCount

CacheMissCount

AWS/SNS

NumberOfNotificationsDelivered

NumberOfMessagesPublished

NumberOfNotificationsFailed

NumberOfNotificationsFilteredOut

NumberOfNotificationsFilteredOut-InvalidAttributes

NumberOfNotificationsFilteredOut-NoMessageAttributes

NumberOfNotificationsRedrivenToDlq

NumberOfNotificationsFailedToRedriveToDlq

SMSSuccessRate

권장 지표

다음 표에는 각 구성 요소 유형에 대해 권장되는 지표가 나와 있습니다.

구성 요소 유형	워크로드 유형	권장 지표
EC2 인스턴스(Windows Server)	기본값/사용자 지정	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 LogicalDisk % 사용 가능한 공간 사용 가능한 메모리(MB)
	Active Directory	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 사용 가능한 메모리(MB) 데이터베이스 ==> 인스턴스 데이터베이스 캐시 적중률 DirectoryServices DRA에서 보류 중인 복제 운영 DirectoryServices DRA에서 보류 중인 복제 동기화 DNS 재귀 쿼리 실패/초 LogicalDisk 평균 디스크 대기열 길이
	Java 애플리케이션	CPUUtilization

구성 요소 유형	워크로드 유형	권장 지표
		StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 사용 가능한 메모리(MB) java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

구성 요소 유형	워크로드 유형	권장 지표
	Microsoft IIS/.NET 웹 프런트엔드	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 사용 가능한 메모리(MB) .NET CLR 예외 발생된 예외 수/초 .NET CLR 메모리 수 총 커밋된 바이트 수 .NET CLR 메모리 GC의 시간(%) ASP.NET 응용 프로그램 응용 프로그램 대기열의 요청 ASP.NET 대기 중인 요청 ASP.NET 응용 프로그램 재시작

구성 요소 유형	워크로드 유형	권장 지표
	Microsoft SQL Server 데이터베이스 티어	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 사용 가능한 메모리(MB) 페이징된 파일 % 사용량 시스템 프로세서 대기열 길이 네트워크 인터페이스 바이트 합계/초 PhysicalDisk % 디스크 시간 SQLServer:버퍼 관리자 버퍼 캐시 적중률 SQLServer:버퍼 관리자 페이지 예상 수명 SQLServer:차단된 일반 통계 프로세스 SQLServer:일반 통계 사용자 연결 SQLServer:잠금 교착 상태 수/초 SQLServer:SQL 통계 배치 요청/초

구성 요소 유형	워크로드 유형	권장 지표
	MySQL	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 LogicalDisk % 사용 가능한 공간 사용 가능한 메모리(MB)
	.NET 작업자 풀/중간 티어	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 사용 가능한 메모리(MB) .NET CLR 예외 발생된 예외 수/초 .NET CLR 메모리 수 총 커밋된 바이트 수 .NET CLR 메모리 GC의 시간(%)

구성 요소 유형	워크로드 유형	권장 지표
	.NET Core 티어	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 사용 가능한 메모리(MB)
	Oracle	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 LogicalDisk % 사용 가능한 공간 사용 가능한 메모리(MB)
	Postgres	CPUUtilization StatusCheckFailed 프로세서 % 프로세서 시간 사용 중인 메모리 % 커밋된 바이트 LogicalDisk % 사용 가능한 공간 사용 가능한 메모리(MB)

구성 요소 유형	워크로드 유형	권장 지표
	SharePoint	CPUUtilization
		StatusCheckFailed
		프로세서 % 프로세서 시간
		사용 중인 메모리 % 커밋된 바이트
		사용 가능한 메모리(MB)
		ASP.NET 애플리케이션 캐시 API 공백 제거
		ASP.NET 요청 거부됨
		ASP.NET 작업자 프로세스 다시 시작
		메모리 페이지/초
		SharePoint Publishing Cache 게시 캐시 플러시/초
		SharePoint Foundation 실행 시간/페이지 요청
		SharePoint Disk-Based Cache 총 캐시 압축 횟수
		SharePoint Disk-Based Cache Blob 캐시 적중률
		SharePoint Disk-Based Cache Blob 캐시 채우기 비율
		SharePoint Disk-Based Cache Blob 캐시 플러시/초

구성 요소 유형	워크로드 유형	권장 지표
		ASP.NET 대기 중인 요청 ASP.NET 응용 프로그램 응용 프로그램 대기열의 요청 ASP.NET 응용 프로그램 재시작 LogicalDisk 평균 디스크 초/쓰기 LogicalDisk 평균 디스크 초/읽기 프로세서 % 중단 시간
EC2 인스턴스(Linux 서버)	기본값/사용자 지정	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

구성 요소 유형	워크로드 유형	권장 지표
	Java 애플리케이션	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize
	.NET Core 티어 또는 SQL Server 데이터베이스 티어	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

구성 요소 유형	워크로드 유형	권장 지표
	Oracle	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent
	Postgres	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

구성 요소 유형	워크로드 유형	권장 지표
EC2 인스턴스 그룹	SAP HANA 다중 노드 또는 단일 노드	• hanadb_server_startup_time_variations_seconds • hanadb_level_5_alerts_count • hanadb_level_4_alerts_count • hanadb_out_of_memory_events_count • hanadb_max_trigger_read_ratio_percent • hanadb_max_trigger_write_ratio_percent • hanadb_log_switch_race_ratio_percent • hanadb_time_since_last_savepoint_seconds • hanadb_disk_usage_highlevel_percent • hanadb_current_allocation_limit_used_percent • hanadb_table_allocation_limit_used_percent • hanadb_cpu_usage_percent • hanadb_plan_cache_hit_ratio_percent • hanadb_last_data_backup_age_days

구성 요소 유형	워크로드 유형	권장 지표
EBS 볼륨	모두	VolumeReadBytes VolumeWriteBytes VolumeReadOps VolumeWriteOps VolumeQueueLength VolumeThroughputPercentage VolumeConsumedReadWriteOps BurstBalance
Classic ELB	모두	HTTPCode_Backend_4XX HTTPCode_Backend_5XX 지연 시간 SurgeQueueLength UnHealthyHostCount
애플리케이션 ELB	모두	HTTPCode_Target_4XX_Count HTTPCode_Target_5XX_Count TargetResponseTime UnHealthyHostCount

구성 요소 유형	워크로드 유형	권장 지표
RDS 데이터베이스 인스턴스	모두	CPUUtilization ReadLatency WriteLatency BurstBalance FailedSQLServerAgentJobsCount
RDS 데이터베이스 클러스터	모두	CPUUtilization CommitLatency DatabaseConnections 교착 FreeableMemory NetworkThroughput VolumeBytesUsed
Lambda 함수	모두	지속 시간 Errors IteratorAge ProvisionedConcurrencySpilloverInvocations 제한

구성 요소 유형	워크로드 유형	권장 지표
SQS 대기열	모두	ApproximateAgeOfOldestMessage ApproximateNumberOfMessagesVisible NumberOfMessagesSent
Amazon DynamoDB 테이블	모두	SystemErrors UserErrors ConsumedReadCapacityUnits ConsumedWriteCapacityUnits ReadThrottleEvents WriteThrottleEvents ConditionalCheckFailedRequests TransactionConflict

구성 요소 유형	워크로드 유형	권장 지표
Amazon S3 버킷	모두	RTC(복제 시간 제어)를 사용하는 복제 구이 활성화되어 있는 경우: ReplicationLatency BytesPendingReplication OperationsPendingReplication 요청 지표가 켜져 있는 경우: 5xxErrors 4xxErrors BytesDownloaded BytesUploaded

구성 요소 유형	워크로드 유형	권장 지표
AWS Step Functions	모두	일반 • ExecutionThrottled • ExecutionsAborted • ProvisionedBucketSize • ProvisionedRefillRate • ConsumedCapacity 상태 머신 유형이 **EXPRESS**이거나 로그 그룹 수준이 **OFF**인 경우 • ExecutionsFailed • ExecutionsTimedOut 상태 머신에 Lambda 함수가 있는 경우 • LambdaFunctionsFailed • LambdaFunctionsTimedOut 상태 머신에 활동이 있는 경우 • ActivitiesFailed • ActivitiesTimedOut • ActivitiesHeartbeatTimedOut 상태 머신에 서비스 통합이 있는 경우 • ServiceIntegrationsFailed

구성 요소 유형	워크로드 유형	권장 지표
		• ServiceIntegrationsTimedOut
API Gateway REST API 스테이지	모두	• 4XXErrors • 5XXErrors • 지연 시간

구성 요소 유형	워크로드 유형	권장 지표
ECS 클러스터	모두	CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes CPUReservation(EC2 시작 유형만 해당) CPUUtilization(EC2 시작 유형만 해당) MemoryReservation(EC2 시작 유형만 해당) MemoryUtilization(EC2 시작 유형만 해당) GPUReservation(EC2 시작 유형만 해당) instance_cpu_utilization(EC2 시작 유형만 해당) instance_filesystem_utilization(EC2 시작 유형만 해당) instance_memory_utilization(EC2 시작 유형만 해당)

구성 요소 유형	워크로드 유형	권장 지표
		instance_network_total_bytes(EC2 시작 유형만 해당)

구성 요소 유형	워크로드 유형	권장 지표
	Java 애플리케이션	CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes CPUReservation(EC2 시작 유형만 해당) CPUUtilization(EC2 시작 유형만 해당) MemoryReservation(EC2 시작 유형만 해당) MemoryUtilization(EC2 시작 유형만 해당) GPUReservation(EC2 시작 유형만 해당) instance_cpu_utilization(EC2 시작 유형만 해당) instance_filesystem_utilization(EC2 시작 유형만 해당) instance_memory_utilization(EC2 시작 유형만 해당)

구성 요소 유형	워크로드 유형	권장 지표
		instance_network_total_bytes(EC2 시작 유형만 해당) java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize
ECS 서비스	모두	CPUUtilization MemoryUtilization CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes

구성 요소 유형	워크로드 유형	권장 지표
	Java 애플리케이션	CPUUtilization MemoryUtilization CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

구성 요소 유형	워크로드 유형	권장 지표
EKS 클러스터	모두	cluster_failed_node_count
		node_cpu_reserved_capacity
		node_cpu_utilization
		node_filesystem_utilization
		node_memory_reserved_capacity
		node_memory_utilization
		node_network_total_bytes
		pod_cpu_reserved_capacity
		pod_cpu_utilization
		pod_cpu_utilization_over_pod_limit
		pod_memory_reserved_capacity
		pod_memory_utilization
		pod_memory_utilization_over_pod_limit
		pod_network_rx_bytes
		pod_network_tx_bytes

구성 요소 유형	워크로드 유형	권장 지표
	Java 애플리케이션	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount

구성 요소 유형	워크로드 유형	권장 지표
		java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

구성 요소 유형	워크로드 유형	권장 지표
EC2의 Kubernetes 클러스터	모두	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes

구성 요소 유형	워크로드 유형	권장 지표
	Java 애플리케이션	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount

구성 요소 유형	워크로드 유형	권장 지표
		java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

다음 표에는 각 구성 요소 유형에 대해 권장되는 프로세스 및 프로세스 지표가 나와 있습니다. CloudWatch Application Insights는 인스턴스에서 실행되지 않는 프로세스에 대한 프로세스 모니터링을 권장하지 않습니다.

구성 요소 유형	워크로드 유형	권장 프로세스	권장 지표

EC2 인스턴스(Windows Server)	Microsoft IIS/.NET 웹 프런트 엔드	`w3wp`	`procstat cpu_usage ,`
			
			`procstat memory_rss ,`
			
			`procstat memory_vms ,`
			
			`procstat read_bytes ,`
			
			`procstat write_bytes`

	Microsoft SQL Server 데이터베이스 티어	`SQLAgent`	`procstat cpu_usage ,`

구성 요소 유형	워크로드 유형	권장 프로세스	권장 지표
			procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes
		sqlservr	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes
		sqlwriter	procstat cpu_usage , procstat memory_rss
		Reporting ServicesS ervice	procstat cpu_usage , procstat memory_rss

구성 요소 유형	워크로드 유형	권장 프로세스	권장 지표
		MsDtsServr	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes
		Msmdsrv	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes

구성 요소 유형	워크로드 유형	권장 프로세스	권장 지표
	.NET 작업자 풀/중간 티어	w3wp	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes
	.NET Core 티어	w3wp	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes

성능 카운터 지표

성능 카운터 지표는 해당 성능 카운터 세트가 Windows 인스턴스에 설치된 경우에만 인스턴스에 권장됩니다.

성능 카운터 지표 이름	성능 카운터 집합 이름
.NET CLR 예외 발생된 예외 수	.NET CLR 예외
.NET CLR 예외 발생된 예외 수/초	.NET CLR 예외
.NET CLR 예외 필터 수/초	.NET CLR 예외
.NET CLR 예외 최종 수/초	.NET CLR 예외
.NET CLR 예외 발생 깊이 파악/초	.NET CLR 예외
.NET CLR Interop CCW 수	.NET CLR Interop
.NET CLR Interop 스텁 수	.NET CLR Interop
.NET CLR Interop TLB 내보내기 횟수/초	.NET CLR Interop
.NET CLR Interop TLB 가져오기 횟수/초	.NET CLR Interop
.NET CLR Interop 마샬링 수	.NET CLR Interop
.NET CLR Jit Jit 시간(%)	.NET CLR Jit
.NET CLR Jit 표준 Jit 실패	.NET CLR Jit
.NET CLR 로딩 로딩 시간(%)	.NET CLR 로딩
.NET CLR 로딩 로드 실패 비율	.NET CLR 로딩
.NET CLR LocksAndThreads 경합 속도/초	.NET CLR LocksAndThreads
.NET CLR LocksAndThreads 대기열 길이/초	.NET CLR LocksAndThreads
.NET CLR 메모리 수 총 커밋된 바이트 수	.NET CLR 메모리
.NET CLR 메모리 GC의 시간(%)	.NET CLR 메모리
.NET CLR 네트워킹 4.0.0.0 HttpWebRequest 평균 대기열 시간	.NET CLR 네트워킹 4.0.0.0

성능 카운터 지표 이름	성능 카운터 집합 이름
.NET CLR 네트워킹 4.0.0.0 중단된 HttpWebRequests/초	.NET CLR 네트워킹 4.0.0.0
.NET CLR 네트워킹 4.0.0.0 실패한 HttpWebRequests/초	.NET CLR 네트워킹 4.0.0.0
.NET CLR 네트워킹 4.0.0.0 대기 중인 HttpWebRequests/Sec	.NET CLR 네트워킹 4.0.0.0
APP_POOL_WAS Total Worker Process Ping Failures	APP_POOL_WAS
ASP.NET 응용 프로그램 재시작	ASP.NET
ASP.NET 요청 거부됨	ASP.NET
ASP.NET 작업자 프로세스 다시 시작	ASP.NET
ASP.NET 애플리케이션 캐시 API 공백 제거	ASP.NET 응용 프로그램
ASP.NET Applications % Managed Processor Time(estimated)	ASP.NET 응용 프로그램
ASP.NET 응용 프로그램 오류 합계/초	ASP.NET 응용 프로그램
ASP.NET 응용 프로그램 오류 실행 중 처리되지 않은 오류/초	ASP.NET 응용 프로그램
ASP.NET 응용 프로그램 응용 프로그램 대기열의 요청	ASP.NET 응용 프로그램
ASP.NET 응용 프로그램 요청/초	ASP.NET 응용 프로그램
ASP.NET 요청 대기 시간	ASP.NET
ASP.NET 대기 중인 요청	ASP.NET
데이터베이스 ==> 인스턴스 데이터베이스 캐시 적중률	데이터베이스 ==> 인스턴스

성능 카운터 지표 이름	성능 카운터 집합 이름
데이터베이스 ==> 인스턴스 I/O 데이터베이스 읽기 평균 대기 시간	데이터베이스 ==> 인스턴스
데이터베이스 ==> 인스턴스 I/O 데이터베이스 읽기/초	데이터베이스 ==> 인스턴스
데이터베이스 ==> 인스턴스 I/O 로그 쓰기 평균 대기 시간	데이터베이스 ==> 인스턴스
DirectoryServices DRA에서 보류 중인 복제 운영	DirectoryServices
DirectoryServices DRA에서 보류 중인 복제 동기화	DirectoryServices
DirectoryServices LDAP 바인딩 시간	DirectoryServices
DNS 재귀 쿼리/초	DNS
DNS 재귀 쿼리 실패/초	DNS
DNS TCP 쿼리 수신/초	DNS
DNS 총 쿼리 수신/초	DNS
DNS 총 응답 발송/초	DNS
DNS UDP 쿼리 수신/초	DNS
HTTP 서비스 요청 대기열 CurrentQueueSize	HTTP 서비스 요청 대기열
LogicalDisk % 사용 가능한 공간	LogicalDisk
LogicalDisk 평균 디스크 초/쓰기	LogicalDisk
LogicalDisk 평균 디스크 초/읽기	LogicalDisk
LogicalDisk 평균 디스크 대기열 길이	LogicalDisk

성능 카운터 지표 이름	성능 카운터 집합 이름
사용 중인 메모리 % 커밋된 바이트	메모리
사용 가능한 메모리(MB)	메모리
메모리 페이지/초	메모리
메모리 장기 평균 대기 캐시 수명(초)	메모리
네트워크 인터페이스 바이트 합계/초	네트워크 인터페이스
네트워크 인터페이스 바이트 수신/초	네트워크 인터페이스
네트워크 인터페이스 바이트 발송/초	네트워크 인터페이스
네트워크 인터페이스 현재 대역폭	네트워크 인터페이스
페이징된 파일 % 사용량	페이징된 파일
PhysicalDisk % 디스크 시간	PhysicalDisk
PhysicalDisk 평균 디스크 대기열 길이	PhysicalDisk
PhysicalDisk 평균 디스크 초/읽기	PhysicalDisk
PhysicalDisk 평균 디스크 초/쓰기	PhysicalDisk
PhysicalDisk 디스크 읽기 바이트/초	PhysicalDisk
PhysicalDisk 디스크 읽기/초	PhysicalDisk
PhysicalDisk 디스크 쓰기 바이트/초	PhysicalDisk
PhysicalDisk 디스크 쓰기/초	PhysicalDisk
프로세서 % 유휴 시간	처리자
프로세서 % 중단 시간	처리자
프로세서 % 프로세서 시간	처리자

성능 카운터 지표 이름	성능 카운터 집합 이름
프로세서 % 사용자 시간	처리자
SharePoint Disk-Based Cache Blob 캐시 채우기 비율	SharePoint Disk-Based Cache
SharePoint Disk-Based Cache Blob 캐시 플러시/초	SharePoint Disk-Based Cache
SharePoint Disk-Based Cache Blob 캐시 적중률	SharePoint Disk-Based Cache
SharePoint Disk-Based Cache 총 캐시 압축 횟수	SharePoint Disk-Based Cache
SharePoint Foundation 실행 시간/페이지 요청	SharePoint Foundation
SharePoint Publishing Cache 게시 캐시 플러시/초	SharePoint Publishing Cache
보안 시스템 전체 통계 Kerberos 인증	보안 시스템 전체 통계
보안 시스템 전체 통계 NTLM 인증	보안 시스템 전체 통계
SQLServer:액세스 방법 전달된 레코드/초	SQLServer:액세스 방법
SQLServer:액세스 방법 전체 스캔/초	SQLServer:액세스 방법
SQLServer:액세스 방법 페이지 분할/초	SQLServer:액세스 방법
SQLServer:버퍼 관리자 버퍼 캐시 적중률	SQLServer:버퍼 관리자
SQLServer:버퍼 관리자 페이지 예상 수명	SQLServer:버퍼 관리자
SQLServer:수신된 데이터베이스 복제본 파일 바이트/초	SQLServer:Database Replica
SQLServer:수신된 데이터베이스 복제본 로그 바이트/초	SQLServer:Database Replica

성능 카운터 지표 이름	성능 카운터 집합 이름
SQLServer:실행 취소를 위한 나머지 데이터베이스 복제본 로그	SQLServer:Database Replica
SQLServer:데이터베이스 복제본 로그 전송 대기열	SQLServer:Database Replica
SQLServer:데이터베이스 복제본 미러링된 쓰기 트랜잭션/초	SQLServer:Database Replica
SQLServer:데이터베이스 복제본 복구 대기열	SQLServer:Database Replica
SQLServer:나머지 데이터베이스 복제본 다시 실행 바이트	SQLServer:Database Replica
SQLServer:데이터베이스 복제본 다시 실행 바이트/초	SQLServer:Database Replica
SQLServer:실행 취소가 필요한 데이터베이스 복제본 총 로그	SQLServer:Database Replica
SQLServer:데이터베이스 복제본 트랜잭션 지연	SQLServer:Database Replica
SQLServer:차단된 일반 통계 프로세스	SQLServer:일반 통계
SQLServer:일반 통계 사용자 연결	SQLServer:일반 통계
SQLServer:래치 평균 래치 대기 시간(ms)	SQLServer:래치
SQLServer:잠금 평균 대기 시간(ms)	SQLServer:잠금
SQLServer:잠금 잠금 제한 시간/초	SQLServer:잠금
SQLServer:잠금 잠금 대기/초	SQLServer:잠금
SQLServer:잠금 교착 상태 수/초	SQLServer:잠금
SQLServer:메모리 관리자 보류 중인 메모리 부여	SQLServer:메모리 관리자

성능 카운터 지표 이름	성능 카운터 집합 이름
SQLServer:SQL 통계 배치 요청/초	SQLServer:SQL 통계
SQLServer:SQL 통계 SQL 컴파일/초	SQLServer:SQL 통계
SQLServer:SQL 통계 SQL 재컴파일/초	SQLServer:SQL 통계
시스템 프로세서 대기열 길이	시스템
TCPv4 연결 설정	TCPv4
TCPv6 연결 설정	TCPv6
W3SVC_W3WP 파일 캐시 플러시	W3SVC_W3WP
W3SVC_W3WP 파일 캐시 누락	W3SVC_W3WP
W3SVC_W3WP 요청/초	W3SVC_W3WP
W3SVC_W3WP URI 캐시 플러시	W3SVC_W3WP
W3SVC_W3WP URI 캐시 누락	W3SVC_W3WP
웹 서비스 수신된 바이트/초	웹 서비스
웹 서비스 전송된 바이트/초	웹 서비스
웹 서비스 연결 시도/초	웹 서비스
웹 서비스 현재 연결	웹 서비스
웹 서비스 요청 가져오기/초	웹 서비스
웹 서비스 POST 요청/초	웹 서비스

CloudWatch 콘솔에서 리소스 상태 뷰 사용

리소스 상태 보기를 사용하여 단일 보기에서 애플리케이션 전반의 호스트 상태 및 성능을 자동으로 검색하고 관리하며 시각화할 수 있습니다. CPU 또는 메모리와 같은 성능 차원별로 해당 호스트의 상태를 시각화하고, 필터를 사용하여 단일 보기에서 수백 개의 호스트를 쪼개어 분석할 수 있습니다. 동일

한 Auto Scaling 그룹의 호스트 또는 동일한 로드 밸런서를 사용하는 호스트와 같이 사용 사례 또는 태그별로 필터링할 수 있습니다.

사전 조건

리소스 상태 보기를 최대한 활용하려면 다음 사전 조건이 있는지 확인해야 합니다.

- 호스트의 메모리 사용률을 확인하고 이를 필터로 사용하려면 호스트에 CloudWatch 에이전트를 설치하여 기본 `CWAgent` 네임스페이스의 CloudWatch에 메모리 지표를 전송하도록 설정해야 합니다. Linux 및 macOS 인스턴스에서는 CloudWatch 에이전트가 `mem_used_percent` 지표를 전송해야 합니다. Windows 인스턴스에서는 에이전트가 `Memory % Committed Bytes In Use` 지표를 전송해야 합니다. 이러한 지표는 마법사를 사용하여 CloudWatch 에이전트 구성 파일을 생성하고 미리 정의된 지표 세트를 선택하는 경우에 포함됩니다. CloudWatch 에이전트가 수집한 지표는 사용자 지정 지표로 청구됩니다. 자세한 내용은 CloudWatch 에이전트 설치 단원을 참조하세요.

  CloudWatch 에이전트를 사용하여 리소스 상태 보기와 함께 사용할 이러한 메모리 지표를 수집하는 경우 CloudWatch 에이전트 구성 파일에 다음 섹션을 포함해야 합니다. 이 섹션은 기본적으로 생성되며 기본 차원 설정을 포함하므로 이 섹션의 어느 부분도 다음 예에 나와 있는 것과 다른 내용으로 변경해서는 안 됩니다.

  ```
  "append_dimensions": {
    "ImageId": "${aws:ImageId}",
    "InstanceId": "${aws:InstanceId}",
    "InstanceType": "${aws:InstanceType}",
    "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
  },
  ```

- 리소스 상태 보기에서 사용 가능한 모든 정보를 보려면 다음 권한이 있는 계정에 로그인해야 합니다. 더 적은 권한으로 로그인한 경우 여전히 리소스 상태 보기를 사용할 수는 있지만, 일부 성능 데이터가 표시되지 않습니다.

  ```
  {
      "Version": "2012-10-17",
      "Statement": [
          {
              "Action": [
                  "autoscaling:Describe*",
                  "cloudwatch:Describe*",
                  "cloudwatch:Describe*",
                  "cloudwatch:Get*",
                  "cloudwatch:List*",
  ```

```
                "logs:Get*",
                "logs:Describe*",
                "sns:Get*",
                "sns:List*",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeRegions"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

계정의 리소스 상태를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 인프라 모니터링, 리소스 상태를 선택합니다.

   계정의 각 호스트에 대한 사각형이 표시된 리소스 상태 페이지가 나타납니다. 각 사각형은 해당 호스트의 현재 상태를 기반으로 [색상 기준(Color by)] 설정에 따른 색상이 지정됩니다. 경보 기호가 있는 호스트 사각형에는 현재 ALARM 상태에 있는 경보가 하나 이상 있습니다.

   단일 보기에는 호스트가 최대 500개까지 표시될 수 있습니다. 계정에 더 많은 호스트가 있는 경우 이 절차의 6단계에서 필터 설정을 사용합니다.
3. 각 호스트의 상태를 표시하는 데 사용되는 기준을 변경하려면 [색상 기준(Color by)]의 설정을 선택합니다. [CPU 사용률(CPU Utilization)], [메모리 사용률(Memory Utilization)] 또는 [상태 확인(Status check)]을 선택할 수 있습니다. 메모리 사용률 지표는 CloudWatch 에이전트를 실행 중이며 메모리 지표를 수집하여 기본 `CWAgent` 네임스페이스에 전송하도록 구성된 호스트에서만 사용할 수 있습니다. 자세한 내용은 CloudWatch 에이전트를 사용하여 지표, 로그, 추적 수집 단원을 참조하세요.
4. 그리드의 상태 표시기에 사용되는 임계값 및 색상을 변경하려면 그리드 위의 기어 아이콘을 선택합니다.
5. 호스트 그리드에 경보를 표시할지 여부를 전환하려면 [모든 지표의 경보 표시(Show alarms across all metrics)]를 선택하거나 선택 취소합니다.
6. 맵의 호스트를 그룹으로 분할하려면 [그룹 기준(Group by)]의 그룹화 기준을 선택합니다.

7. 더 적은 수의 호스트로 보기 범위를 좁히려면 [필터링 기준(Filter by)]의 필터 기준을 선택합니다. 태그별로 그리고 Auto Scaling 그룹, 인스턴스 유형, 보안 그룹 등과 같은 리소스 그룹별로 필터링할 수 있습니다.
8. 호스트를 정렬하려면 [정렬 기준(Sort by)]의 정렬 기준을 선택합니다. 상태 확인 결과, 인스턴스 상태, CPU 또는 메모리 사용률, ALARM 상태에 있는 경보 수를 기준으로 정렬할 수 있습니다.
9. 호스트에 관한 추가 정보를 보려면 해당 호스트를 나타내는 사각형을 선택합니다. 팝업 창이 나타납니다. 해당 호스트에 관한 정보를 더 자세히 알아보려면 [대시보드 보기(View dashboard)] 또는 [목록으로 보기(View on list)]를 선택합니다.

CloudWatch 크로스 계정 관측성

Amazon CloudWatch 크로스 계정 관측성을 사용하면 리전 내 여러 계정에 걸쳐 있는 애플리케이션을 모니터링하고 문제를 해결할 수 있습니다. 계정 경계 없이 연결된 모든 계정에서 지표, 로그, 추적, Application Insights 애플리케이션 및 Internet Monitor 모니터를 원활하게 검색, 시각화하고 분석할 수 있습니다.

하나 이상의 AWS 계정을 모니터링 계정으로 설정하고 여러 소스 계정과 연결하세요. 모니터링 계정은 소스 계정에서 생성된 관측성 데이터를 보고 상호 작용할 수 있는 중앙 AWS 계정입니다. 소스 계정은 해당 계정에 있는 리소스에 대한 관측성 데이터를 생성하는 개별 AWS 계정입니다. 소스 계정은 관측성 데이터를 모니터링 계정과 공유합니다. 공유된 관측 가능성 데이터에는 다음과 같은 유형의 텔레메트리가 포함될 수 있습니다.

- Amazon CloudWatch의 지표. 모든 네임스페이스의 지표를 모니터링 계정과 공유하거나 네임스페이스의 하위 집합으로 필터링할 수 있습니다.
- Amazon CloudWatch Logs의 로그 그룹. 모든 로그 그룹을 모니터링 계정과 공유하거나 로그 그룹의 하위 집합으로 필터링할 수 있습니다.
- AWS X-Ray의 추적
- Amazon CloudWatch Application Insights의 애플리케이션
- CloudWatch Internet Monitor에서 모니터

모니터링 계정과 소스 계정 간 링크를 생성하려면 CloudWatch 콘솔을 사용할 수 있습니다. 또는 AWS CLI 및 API에서 Observability Access Manager 명령을 사용합니다. 자세한 내용은 Observability Access Manager API Reference(Observability Access Manager API 참조)에 나와 있습니다.

싱크는 모니터링 계정의 연결 지점을 나타내는 리소스입니다. 소스 계정은 싱크에 연결하여 관측성 데이터를 공유할 수 있습니다. 각 계정에는 리전당 싱크가 하나씩 있을 수 있습니다. 각 싱크는 해당 싱크가 위치한 모니터링 계정에서 관리합니다. 관측성 링크는 소스 계정과 모니터링 계정 간에 설정된 링크를 나타내는 리소스입니다. 링크는 소스 계정에서 관리합니다.

CloudWatch 크로스 계정 관측성 설정에 대한 동영상 데모는 다음 동영상을 참조하세요.

다음 주제에서는 모니터링 계정과 소스 계정 모두에서 CloudWatch 크로스 계정 관측성을 설정하는 방법을 설명합니다. 크로스 계정 크로스 리전 CloudWatch 대시보드에 대한 자세한 내용은 교차 계정 교차 리전 CloudWatch 콘솔 섹션을 참조하세요.

소스 계정에 Organizations 사용

소스 계정을 모니터링 계정에 연결하는 두 가지 옵션이 있습니다. 하나 또는 두 가지 옵션을 모두 사용할 수 있습니다.

- AWS Organizations를 사용하여 조직 또는 조직 단위의 계정을 모니터링 계정에 연결합니다.
- 모니터링 계정에 개별 AWS 계정 연결.

조직에서 나중에 생성된 새 AWS 계정이 크로스 계정 관측성에 소스 계정으로 자동 온보드되도록 Organizations를 사용하는 것이 좋습니다.

모니터링 계정과 소스 계정 연결에 대한 세부 정보

- 각 모니터링 계정을 최대 10만 개의 소스 계정에 연결할 수 있습니다.
- 각 소스 계정은 최대 5개의 모니터링 계정과 데이터를 공유할 수 있습니다.
- 단일 계정을 모니터링 계정과 소스 계정 모두로 설정할 수 있습니다. 이 경우 이 계정은 자체의 관측성 데이터만 연결된 모니터링 계정으로 전송합니다. 소스 계정의 데이터는 릴레이하지 않습니다.
- 모니터링 계정은 해당 계정과 공유할 수 있는 텔레메트리 유형을 지정합니다. 소스 계정은 공유할 텔레메트리 유형을 지정합니다.
  - 모니터링 계정에서 선택한 텔레메트리 유형이 소스 계정보다 많은 경우 계정이 연결됩니다. 두 계정에서 선택한 데이터 유형만 공유됩니다.
  - 소스 계정에서 선택한 텔레메트리 유형이 모니터링 계정보다 많은 경우 연결 생성에 실패하고 아무 것도 공유되지 않습니다.
  - 링크가 생성된 후 해당 지표가 새 데이터 포인트를 내보내기 전까지는 모니터링 계정 콘솔에 지표 이름이 표시되지 않습니다.
- 계정 간 링크를 제거하려면 소스 계정에서 제거합니다.
- 모니터링 계정에서 싱크를 삭제하려면 먼저 모니터링 계정에 대한 모든 링크를 제거해야 합니다.

요금

CloudWatch의 크로스 계정 관측성은 로그 및 지표에 대한 추가 비용 없이 제공되며 첫 번째 추적 사본은 무료입니다. 요금에 대한 자세한 정보는 Amazon CloudWatch 요금을 참조하세요.

목차

소스 계정과 모니터링 계정 연결

이 섹션의 항목에서는 모니터링 계정과 소스 계정 간 링크를 설정하는 방법을 설명합니다.

조직의 모니터링 계정으로 사용할 새 AWS 계정을 생성하는 것이 좋습니다.

목차

필요한 권한

모니터링 계정과 소스 계정 간 링크를 생성하려면 특정 권한으로 로그인해야 합니다.

- 모니터링 계정을 설정하려면 - 모니터링 계정에서 전체 관리자 액세스 권한이 있거나 다음 권한으로 해당 계정에 로그인해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSinkModification",
            "Effect": "Allow",
            "Action": [
                "oam:CreateSink",
                "oam:DeleteSink",
                "oam:PutSinkPolicy",
                "oam:TagResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnly",
            "Effect": "Allow",
            "Action": ["oam:Get*", "oam:List*"],
            "Resource": "*"
        }
    ]
}
```

- 특정 모니터링 계정으로 범위가 지정된 소스 계정 - 지정된 하나의 모니터링 계정에 대한 링크를 생성, 업데이트 및 관리하려면 최소한 다음 권한이 있는 계정에 로그인해야 합니다. 이 예제에서 모니터링 계정은 999999999999입니다.

  링크가 5가지 리소스 유형(지표, 로그, 추적, Application Insights 애플리케이션, Internet Monitor 모니터)을 모두 공유하지 않는 경우 필요에 따라 `cloudwatch:Link`, `logs:Link`, `xray:Link`, `applicationinsights:Link` 또는 `internetmonitor:Link`를 생략할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink",
                "oam:DeleteLink",
```

```
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Effect": "Allow",
            "Resource": "arn:*:oam:*:*:link/*"
        },
        {
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Effect": "Allow",
            "Resource": "arn:*:oam:*:*:sink/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": [
                        "999999999999"
                    ]
                }
            }
        },
        {
            "Action": "oam:ListLinks",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "cloudwatch:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "logs:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "xray:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
             "Action": "applicationinsights:Link",
             "Effect": "Allow",
```

```
                "Resource": "*"
            },
            {
                "Action": "internetmonitor:Link",
                "Effect": "Allow",
                "Resource": "*"
            }
        ]
}
```

- 모든 모니터링 계정에 연결할 수 있는 권한이 있는 소스 계정 – 기존 모니터링 계정 싱크에 대한 링크를 만들고 지표, 로그 그룹, 추적, Application Insights 애플리케이션, Internet Monitor 모니터를 공유하려면, 전체 관리자 권한으로 소스 계정에 로그인하거나 다음 권한으로 로그인해야 합니다.

  링크가 5가지 리소스 유형(지표, 로그, 추적, Application Insights 애플리케이션, Internet Monitor 모니터)을 모두 공유하지 않는 경우 필요에 따라 `cloudwatch:Link`, `logs:Link`, `xray:Link`, `applicationinsights:Link` 또는 `internetmonitor:Link`를 생략할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:List*",
                "oam:Get*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:DeleteLink",
```

```
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
        {
            "Action": "cloudwatch:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "xray:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "logs:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
             "Action": "applicationinsights:Link",
             "Effect": "Allow",
             "Resource": "*"
        },
        {
             "Action": "internetmonitor:Link",
             "Effect": "Allow",
             "Resource": "*"
        }
    ]
}
```

설정 개요

다음의 개괄적인 단계는 CloudWatch 크로스 계정 관측성을 설정하는 방법을 보여줍니다.

> **Note**
>
> 조직의 모니터링 계정으로 사용할 새 AWS 계정을 생성하는 것이 좋습니다.

1. 전용 모니터링 계정을 설정합니다.
2. (선택 사항) AWS CloudFormation 템플릿을 다운로드하거나 URL을 복사하여 소스 계정을 연결합니다.
3. 모니터링 계정에 소스 계정을 연결합니다.

이 단계를 완료한 후 모니터링 계정을 사용하여 소스 계정의 관측성 데이터를 볼 수 있습니다.

1단계: 모니터링 계정 설정

이 섹션의 단계에 따라 AWS 계정을 CloudWatch 크로스 계정 관측성을 위한 모니터링 계정으로 설정하세요.

필수 조건

- AWS Organizations 조직의 계정을 소스 계정으로 설정하는 경우 – 조직 경로 또는 조직 ID를 가져옵니다.
- 소스 계정에 Organizations를 사용하지 않는 경우 - 소스 계정의 계정 ID를 가져옵니다.

계정을 모니터링 계정으로 설정하려면 특정 권한이 있어야 합니다. 자세한 내용은 필요한 권한 단원을 참조하십시오.

모니터링 계정을 설정하려면 다음을 수행하세요.

1. 모니터링 계정으로 사용할 계정에 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 왼쪽 탐색 창에서 설정을 선택합니다.
4. Monitoring account configuration(모니터링 계정 구성)에서 Configure(구성)를 선택합니다.
5. 데이터 선택에서 이 모니터링 계정이 연결된 소스 계정의 로그, 지표, 추적, Application Insights - 애플리케이션, Internet Monitor - 모니터를 볼 수 있을지 여부를 선택합니다.
6. List source accounts(소스 계정 나열)에 이 모니터링 계정이 표시될 소스 계정을 입력합니다. 소스 계정을 식별하려면 개별 계정 ID, 조직 경로 또는 조직 ID를 입력합니다. 조직 경로나 조직 ID를 입력하면 이 모니터링 계정에서 해당 조직의 모든 연결된 계정에서 관측성 데이터를 볼 수 있습니다.

   이 목록의 항목을 쉼표로 구분합니다.

> **Important**
>
> 조직 경로를 입력할 때는 정확한 형식을 따라야 합니다. ou-id는 /(슬래시 문자)로 끝나야 합니다. 예: `o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-def0-awsbbbb/`

7. Define a label to identify your source account(소스 계정을 식별하기 위한 레이블 정의)에서 모니터링 계정을 사용하여 소스 계정을 볼 때 소스 계정을 식별하기 위해 계정 이름을 사용할지 아니면 이메일 주소를 사용할지 지정합니다.
8. 구성을 선택합니다.

> **Important**
>
> 모니터링 계정과 소스 계정 간 링크는 소스 계정을 구성할 때까지 완료되지 않습니다. 자세한 내용은 다음 섹션을 참조하십시오.

2단계: (선택 사항) AWS CloudFormation 템플릿 또는 URL 다운로드

소스 계정을 모니터링 계정에 연결하려면 AWS CloudFormation 템플릿이나 URL을 사용하는 것이 좋습니다.

- 전체 조직을 연결하는 경우 - CloudWatch에서 AWS CloudFormation 템플릿을 제공합니다.
- 개별 계정을 연결하는 경우 - CloudWatch에서 제공하는AWS CloudFormation 템플릿이나 URL을 사용합니다.

AWS CloudFormation 템플릿을 사용하려면 이 단계에서 템플릿을 다운로드해야 합니다. 모니터링 계정을 하나 이상의 소스 계정과 연결한 후에는 AWS CloudFormation 템플릿을 더 이상 다운로드할 수 없습니다.

모니터링 계정에 소스 계정을 연결하기 위해 AWS CloudFormation 템플릿을 다운로드하거나 URL을 복사하려면 다음을 수행하세요.

1. 모니터링 계정으로 사용할 계정에 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 왼쪽 탐색 창에서 설정을 선택합니다.

4. Monitoring account configuration(모니터링 계정 구성)에서 Resources to link accounts(계정을 연결하기 위한 리소스)를 선택합니다.
5. 다음 중 하나를 수행하십시오.
   - 조직의 계정을 이 모니터링 계정에 연결하는 데 사용할 템플릿을 가져오려면 AWS organization을 선택합니다.
   - 개별 계정을 소스 계정으로 설정하기 위해 템플릿 또는 URL을 가져오려면 Any account(모든 계정)를 선택합니다.
6. 다음 중 하나를 수행하십시오.
   - AWS organization을 선택한 경우 Download CloudFormation template(CloudFormation 템플릿 다운로드)을 선택합니다.
   - Any account(모든 계정)를 선택한 경우 Download CloudFormation template(CloudFormation 템플릿 다운로드) 또는 Copy URL(URL 복사)을 선택합니다.
7. (선택 사항) 5~6단계를 반복하여 AWS CloudFormation 템플릿과 URL을 모두 다운로드합니다.

3단계: 소스 계정 연결

이 섹션의 단계에 따라 모니터링 계정에 소스 계정을 연결합니다.

모니터링 계정을 소스 계정과 연결하려면 특정 권한이 있어야 합니다. 자세한 내용은 필요한 권한 단원을 참조하십시오.

AWS CloudFormation 템플릿을 사용하여 조직 또는 조직 단위의 모든 계정을 소스 계정으로 설정합니다.

이 단계에서는 2단계: (선택 사항) AWS CloudFormation 템플릿 또는 URL 다운로드의 단계를 수행하여 필요한 AWS CloudFormation 템플릿을 이미 다운로드했다고 가정합니다.

AWS CloudFormation 템플릿을 사용하여 조직 또는 조직 단위의 계정을 모니터링 계정에 연결합니다.

1. 조직의 관리 계정에 로그인합니다.
2. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
3. 왼쪽 탐색 모음에서 StackSets(스택 세트)를 선택합니다.
4. 원하는 리전에 로그인했는지 확인한 다음 Create StackSet(StackSet 생성)를 선택합니다.
5. 다음을 선택합니다.

6. Template is ready(템플릿이 준비됨)를 선택하고 Upload a template file(템플릿 파일 업로드)를 선택합니다.
7. Choose file(파일 선택)을 선택하고 모니터링 계정에서 다운로드한 템플릿을 선택한 다음 Open(열기)을 선택합니다.
8. 다음을 선택합니다.
9. Specify StackSet details(스택 세트 세부 정보 지정)에 StackSet 이름을 입력하고 Next(다음)를 선택합니다.
10. Add stacks to stack set(스택 세트에 스택 추가)에서 Deploy new stacks(새 스택 배포)를 선택합니다.
11. Deployment targets(배포 대상)에서 전체 조직에 배포할지 아니면 지정된 조직 단위에 배포할지 선택합니다.
12. Specify regions(리전 지정)에서 CloudWatch 크로스 계정 관측성을 배포할 리전을 선택합니다.
13. 다음을 선택합니다.
14. Review(검토) 페이지에서 선택한 옵션을 확인하고 Submit(제출)을 선택합니다.
15. Stack instances(스택 인스턴스) 탭에서 스택 인스턴스의 상태가 CREATE_COMPLETE로 표시될 때까지 화면을 새로 고칩니다.

AWS CloudFormation 템플릿을 사용하여 개별 소스 계정 설정

이 단계에서는 2단계: (선택 사항) AWS CloudFormation 템플릿 또는 URL 다운로드의 단계를 수행하여 필요한 AWS CloudFormation 템플릿을 이미 다운로드했다고 가정합니다.

AWS CloudFormation 템플릿을 사용하여 CloudWatch 크로스 계정 관측성을 위한 개별 소스 계정을 설정하려면 다음을 수행하세요.

1. 소스 계정에 로그인합니다.
2. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
3. 왼쪽 탐색 모음에서 Stacks(스택)를 선택합니다.
4. 원하는 리전에 로그인했는지 확인한 다음 Create stack(StackSet 생성), With new resources (standard)(새 리소스 사용(표준))를 선택합니다.
5. 다음을 선택합니다.
6. 템플릿 파일 업로드를 선택합니다.
7. Choose file(파일 선택)을 선택하고 모니터링 계정에서 다운로드한 템플릿을 선택한 다음 Open(열기)을 선택합니다.

8. 다음을 선택합니다.
9. Specify stack details(스택 세부 정보 지정)에 스택 이름을 입력하고 Next(다음)를 선택합니다.
10. 스택 옵션 구성 페이지에서 다음을 선택합니다.
11. Review(검토) 페이지에서 Submit(제출)을 선택합니다.
12. 스택의 상태 페이지에서 스택의 상태가 CREATE_COMPLETE로 표시될 때까지 화면을 새로 고칩니다.
13. 동일한 템플릿을 사용하여 더 많은 소스 계정을 이 모니터링 계정에 연결하려면 이 계정에서 로그아웃하고 다음 소스 계정에 로그인합니다. 그런 다음 2~12단계를 반복합니다.

URL을 사용하여 개별 소스 계정 설정

이 단계에서는 2단계: (선택 사항) AWS CloudFormation 템플릿 또는 URL 다운로드의 단계를 수행하여 필요한 URL을 이미 복사했다고 가정합니다.

URL을 사용하여 모니터링 계정에 개별 소스 계정을 연결하려면 다음을 수행하세요.

1. 소스 계정으로 사용할 계정에 로그인합니다.
2. 모니터링 계정에서 복사한 URL을 입력합니다.

   일부 정보가 입력된 CloudWatch 설정 페이지가 표시됩니다.
3. 데이터 선택에서 이 소스 계정이 연결된 소스 계정의 로그, 지표, 추적, Application Insights - 애플리케이션, Internet Monitor - 모니터를 공유할지 여부를 선택합니다.

   로그와 지표 모두 모니터링 계정과 모든 리소스 또는 하위 집합을 공유할지 선택할 수 있습니다.

   a. (선택 사항) 이 계정의 로그 그룹 하위 집합을 모니터링 계정과 공유하려면 로그를 선택하고 로그 필터링을 선택합니다. 이후 로그 필터링 상자를 사용하여 공유하려는 로그 그룹을 찾기 위한 쿼리를 구성합니다. 쿼리에서는 `LogGroupName` 텀과 다음 피연산자 중 하나 이상을 사용합니다.

      - `=` 및 `!=`
      - `AND`
      - `OR`
      - `^` 기호는 LIKE, `!^` 기호는 NOT LIKE를 나타냅니다. 접두사 검색으로만 사용할 수 있습니다. 검색 및 포함하려는 문자열의 끝에 `%` 기호를 포함합니다.
      - `IN` 및 `NOT IN`, 괄호(`(` `)`) 사용

전체 쿼리는 2,000자를 넘지 않아야 하고 조건부 피연산자는 5개로 제한됩니다. 조건부 피연산자는 AND 및 OR입니다. 다른 피연산자의 수에는 제한이 없습니다.

> **Tip**
>
> 샘플 쿼리 보기를 선택하면 일반 쿼리 형식의 올바른 구문을 확인할 수 있습니다.

b. (선택 사항) 이 계정의 지표 네임스페이스 하위 집합을 모니터링 계정과 공유하려면 지표를 선택하고 지표 필터링을 선택합니다. 이후 지표 필터링 상자를 사용하여 공유하려는 지표 네임스페이스를 찾기 위한 쿼리를 구성합니다. Namespace 텀과 다음 피연산자 중 하나 이상을 사용합니다.

- = 및 !=
- AND
- OR
- LIKE 및 NOT LIKE. 접두사 검색으로만 사용할 수 있습니다. 검색 및 포함하려는 문자열의 끝에 % 기호를 포함합니다.
- IN 및 NOT IN, 괄호(( )) 사용

전체 쿼리는 2,000자를 넘지 않아야 하고 조건부 피연산자는 5개로 제한됩니다. 조건부 피연산자는 AND 및 OR입니다. 다른 피연산자의 수에는 제한이 없습니다.

> **Tip**
>
> 샘플 쿼리 보기를 선택하면 일반 쿼리 형식의 올바른 구문을 확인할 수 있습니다.

4. Enter monitoring account configuration ARN(모니터링 계정 구성 ARN 입력)에서 ARN을 변경하지 않습니다.
5. Define a label to identify your source account(소스 계정을 식별하기 위한 레이블 정의) 섹션은 모니터링 계정에서 선택한 레이블로 미리 채워져 있습니다. 필요에 따라 Edit(편집)을 선택하여 변경합니다.
6. 연결을 선택합니다.
7. 상자에 **Confirm**을 입력하고 Confirm(확인)을 선택합니다.

8. 동일한 URL을 사용하여 더 많은 소스 계정을 이 모니터링 계정에 연결하려면 이 계정에서 로그아웃하고 다음 소스 계정에 로그인합니다. 그런 다음 2~7단계를 반복합니다.

모니터링 계정과 소스 계정 관리

모니터링 계정과 소스 계정을 설정한 후 이 섹션의 단계를 사용하여 이들을 관리할 수 있습니다.

목차

기존 모니터링 계정에 더 많은 소스 계정 연결

이 섹션의 단계에 따라 추가 소스 계정의 링크를 기존 모니터링 계정에 추가합니다.

각 소스 계정을 최대 5개의 모니터링 계정에 연결할 수 있습니다. 각 모니터링 계정을 최대 10만 개의 소스 계정에 연결할 수 있습니다.

소스 계정을 관리하려면 특정 권한이 있어야 합니다. 자세한 내용은 필요한 권한 단원을 참조하십시오.

모니터링 계정에 더 많은 소스 계정을 연결하려면 다음을 수행하세요.

1. 모니터링 계정에 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 왼쪽 탐색 창에서 설정을 선택합니다.
4. Monitoring account configuration(모니터링 계정 구성)에서 Manage source accounts(소스 계정 관리)를 선택합니다.
5. Configuration policy(구성 정책) 탭을 선택합니다.
6. Configuration policy(구성 정책) 상자의 Principal(보안 주체) 줄에 새 소스 계정 ID를 추가합니다.

   예를 들어, 현재 Principal(보안 주체) 줄이 현재 다음과 같다고 가정합니다.

   ```
   "Principal": {"AWS": ["111111111111", "222222222222"]}
   ```

   999999999999를 세 번째 소스 계정으로 추가하려면 줄을 다음과 같이 편집합니다.

```
"Principal": {"AWS": ["111111111111", "222222222222", "999999999999"]}
```

7. 업데이트를 선택합니다.
8. Configuration details(구성 세부 정보) 탭을 선택합니다.
9. 모니터링 계정의 싱크 ARN 옆에 있는 복사 아이콘을 선택합니다.
10. 새 소스 계정으로 사용할 계정에 로그인합니다.
11. 9단계에서 복사한 모니터링 계정의 싱크 ARN을 붙여넣습니다.

    일부 정보가 입력된 CloudWatch 설정 페이지가 표시됩니다.
12. 데이터 선택에서 이 소스 계정이 연결된 모니터링 계정에 로그, 지표, 추적, Application Insights - 애플리케이션 데이터를 전송할지 여부를 선택합니다.
13. Enter monitoring account configuration ARN(모니터링 계정 구성 ARN 입력)에서 ARN을 변경하지 않습니다.
14. Define a label to identify your source account(소스 계정을 식별하기 위한 레이블 정의) 섹션은 모니터링 계정에서 선택한 레이블로 미리 채워져 있습니다. 필요에 따라 Edit(편집)을 선택하여 변경합니다.
15. 연결을 선택합니다.
16. 상자에 **Confirm**을 입력하고 Confirm(확인)을 선택합니다.

모니터링 계정과 소스 계정 간 링크 제거

이 섹션의 단계에 따라 한 소스 계정에서 모니터링 계정으로 데이터 전송을 중지합니다.

이 작업을 완료하려면 소스 계정을 관리하는 데 필요한 권한이 있어야 합니다. 자세한 내용은 필요한 권한 단원을 참조하십시오.

모니터링 계정과 소스 계정 간 링크를 제거하려면 다음을 수행하세요.

1. 소스 계정에 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 왼쪽 탐색 창에서 설정을 선택합니다.
4. Source account information(소스 계정 정보)에서 View monitoring accounts(모니터링 계정 보기)를 선택합니다.
5. 데이터 공유를 중단할 모니터링 계정 옆의 확인란을 선택합니다.

6. Stop sharing data(데이터 공유 중지), Confirm(확인)을 선택합니다.
7. 모니터링 계정에 로그인합니다.
8. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
9. 설정을 선택합니다.
10. Monitoring account information(모니터링 계정 정보)에서 View configuration(구성 보기)를 선택합니다.
11. Policy(정책) 상자의 Principal(보안 주체) 줄에서 소스 계정 ID를 삭제하고 Update(업데이트)를 선택합니다.

모니터링 계정에 대한 정보 보기

이 섹션의 단계에 따라 모니터링 계정의 크로스 계정 설정을 보세요.

모니터링 계정을 관리하려면 특정 권한이 있어야 합니다. 자세한 내용은 필요한 권한 단원을 참조하십시오.

모니터링 계정을 관리하려면 다음을 수행하세요.

1. 모니터링 계정에 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 왼쪽 탐색 창에서 설정을 선택합니다.
4. Monitoring account configuration(모니터링 계정 구성)에서 Manage source accounts(소스 계정 관리)를 선택합니다.
5. 이 계정이 모니터링 계정이 될 수 있도록 하는 Observability Access Manager 정책을 보려면 Configuration policy(구성 정책) 탭을 선택합니다.
6. 이 모니터링 계정에 연결된 소스 계정을 보려면 Linked source accounts(연결된 소스 계정) 탭을 선택합니다.
7. 모니터링 계정 싱크 ARN과 이 모니터링 계정이 연결된 소스 계정에서 볼 수 있는 데이터 유형을 보려면 Linked source accounts(연결된 소스 계정) 탭을 선택합니다.

다른 데이터 소스의 쿼리 지표

CloudWatch를 사용하여 다른 데이터 소스의 지표를 쿼리하고 시각화하고 경보를 생성할 수 있습니다. 이렇게 하려면 다른 데이터 소스에 CloudWatch를 연결합니다. 이를 통해 CloudWatch 콘솔 내에서 통합된 단일 모니터링 환경을 이용할 수 있습니다. 데이터 저장 위치에 관계없이 인프라 및 애플리케이션 지표를 통합적으로 볼 수 있으므로 문제를 더 빠르게 식별하고 해결할 수 있습니다.

CloudWatch 마법사를 사용하여 데이터 소스에 연결하면 CloudWatch는 AWS Lambda 함수를 배포하고 구성하는 AWS CloudFormation 스택을 생성합니다. 이 Lambda 함수는 데이터 소스를 쿼리할 때마다 온디맨드 방식으로 실행됩니다. CloudWatch 쿼리 작성기는 지표, 테이블, 필드 또는 레이블과 같이 쿼리할 수 있는 요소의 목록을 실시간으로 보여줍니다. 선택을 하면 쿼리 작성기가 선택한 소스의 모국어로 쿼리를 미리 채웁니다.

CloudWatch는 다음 데이터 소스에 연결할 수 있도록 안내 마법사를 제공합니다. 이러한 데이터 소스의 경우 데이터 소스와 보안 인증을 식별하기 위한 기본 정보를 제공합니다. 자체 Lambda 함수를 생성하여 다른 데이터 소스에 대한 커넥터를 수동으로 생성할 수도 있습니다.

- Amazon OpenSearch Service – OpenSearch Service 로그 및 트레이스에서 지표를 파생합니다.
- Amazon Managed Service for Prometheus - PromQL을 사용하여 이러한 지표를 쿼리합니다.
- Amazon RDS for MySQL - SQL을 사용하여 Amazon RDS 테이블에 저장된 데이터를 지표로 변환합니다.
- Amazon RDS for PostgreSQL - SQL을 사용하여 Amazon RDS 테이블에 저장된 데이터를 지표로 변환합니다.
- Amazon S3 CSV 파일 - Amazon S3 버킷에 저장된 CSV 파일의 지표 데이터를 표시합니다.
- Microsoft Azure Monitor - Microsoft Azure Monitor 계정에서 지표를 쿼리합니다.
- Prometheus - PromQL을 사용하여 이러한 지표를 쿼리합니다.

데이터 소스에 대한 커넥터를 생성한 후 데이터 소스의 지표 그래프 작성에 대한 자세한 내용은 다른 데이터 소스의 지표 그래프 생성 섹션을 참조하세요. 데이터 소스의 지표에 대한 경보를 설정하는 방법에 대한 자세한 내용은 연결된 데이터 소스를 기반으로 경보 생성 섹션을 참조하세요.

주제

- 데이터 소스에 대한 액세스 관리
- 마법사로 사전 구축된 데이터 소스에 연결
- 데이터 소스에 대한 사용자 지정 커넥터 생성

- 사용자 지정 데이터 소스 사용
- 데이터 소스에 대한 커넥터 삭제

데이터 소스에 대한 액세스 관리

CloudWatch는 AWS CloudFormation을 사용하여 계정에 필요한 리소스를 생성합니다. IAM 사용자에게 `CreateStack` 권한을 부여할 때 `cloudformation:TemplateUrl` 조건을 사용하여 AWS CloudFormation 템플릿에 대한 액세스를 제어하는 것이 좋습니다.

> **Warning**
>
> 데이터 소스 호출 권한을 부여받은 모든 사용자는 해당 데이터 소스에 대한 직접적인 IAM 권한이 없더라도 해당 데이터 소스에서 지표를 쿼리할 수 있습니다. 예를 들어, Amazon Managed Service for Prometheus 데이터 소스 Lambda 함수에 대한 `lambda:InvokeFunction` 권한을 사용자에게 부여하면 해당 작업 공간에 대한 직접적인 IAM 액세스 권한을 부여하지 않았더라도 사용자는 해당하는 Amazon Managed Service for Prometheus 작업 영역에서 지표를 쿼리할 수 있습니다.

CloudWatch 설정 콘솔의 스택 생성 페이지에서 데이터 소스의 템플릿 URL을 찾을 수 있습니다.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect" : "Allow",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "cloudformation:TemplateUrl" : [ data-source-template-url ]
        }
      }
    }
  ]
}
```

AWS CloudFormation 액세스 제어에 대한 자세한 내용은 AWS Identity and Access Management를 사용한 액세스 제어를 참조하세요.

마법사로 사전 구축된 데이터 소스에 연결

이 항목에서는 마법사를 사용하여 다음 데이터 소스에 CloudWatch를 연결하는 방법에 대한 지침을 제공합니다.

- Amazon OpenSearch Service
- Amazon Managed Service for Prometheus
- Amazon RDS for MySQL
- Amazon RDS for PostgreSQL
- Amazon S3 CSV 파일
- Microsoft Azure Monitor
- Prometheus

이 섹션의 뒷부분에는 이러한 각 데이터 소스를 사용한 관리 및 쿼리에 대한 참고 사항이 포함된 하위 섹션이 있습니다.

데이터 소스에 대한 커넥터 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. 지표 데이터 소스 탭을 선택합니다.
4. 데이터 소스 생성을 선택합니다.
5. 원하는 소스를 선택한 후 다음을 선택합니다.
6. 데이터 소스 이름을 입력합니다.
7. 선택한 데이터 소스에 따라 기타 필수 정보를 입력합니다. 여기에는 Prometheus 작업 공간 이름, 데이터베이스 이름 또는 Amazon S3 버킷 이름과 같은 데이터 소스와 데이터 소스 식별 정보에 액세스하기 위한 보안 인증이 포함될 수 있습니다. AWS 서비스의 경우 마법사가 리소스를 검색하여 선택 드롭다운에 채웁니다.

   사용 중인 데이터 소스에 대한 자세한 내용은 이 절차 이후의 섹션을 참조하세요.
8. CloudWatch가 VPC의 데이터 소스에 연결되도록 하려면 VPC 사용을 선택하고 사용할 VPC를 선택합니다. 그런 다음, 서브넷과 보안 그룹을 선택합니다.
9. AWS CloudFormation이 IAM 리소스를 생성하는 것을 승인함을 선택합니다. 이 리소스는 Lambda 함수 실행 역할입니다.

10. 데이터 소스 생성을 선택합니다.

    방금 추가한 새 소스는 AWS CloudFormation 스택 생성이 완료될 때까지 표시되지 않습니다. 진행 상황을 확인하려면 내 CloudFormation 스택의 상태 보기를 선택합니다. 또는 새로 고침 아이콘을 선택하여 이 목록을 업데이트할 수 있습니다.

    새 데이터 소스가 이 목록에 나타나면 바로 사용할 수 있습니다. CloudWatch 지표에서 쿼리를 선택하여 쿼리를 시작할 수 있습니다. 자세한 내용은 다른 데이터 소스의 지표 그래프 생성 단원을 참조하십시오.

Amazon Managed Service for Prometheus

데이터 소스 구성 업데이트

- 다음을 수행하여 데이터 소스를 수동으로 업데이트할 수 있습니다.
  - Amazon Managed Service for Prometheus 작업 공간 ID를 업데이트하려면 데이터 소스 커넥터 Lambda 함수의 `AMAZON_PROMETHEUS_WORKSPACE_ID` 환경 변수를 업데이트합니다.
  - VPC 구성을 업데이트하려면 자세한 내용은 VPC 액세스 구성(콘솔)을 참조하세요.

데이터 소스 쿼리

- Amazon Managed Service for Prometheus를 쿼리할 때 다중 소스 쿼리 탭에서 데이터 소스를 선택하고 Amazon Managed Service for Prometheus 커넥터를 선택한 후 쿼리 도우미를 사용하여 지표와 레이블을 검색하고 간단한 PromQL 쿼리를 제공할 수 있습니다. PromQL 쿼리 편집기를 사용하여 PromQL 쿼리를 작성할 수도 있습니다.
- CloudWatch 데이터 소스 커넥터는 여러 줄 쿼리를 지원하지 않습니다. 쿼리가 실행되거나 쿼리로 경보 또는 대시보드 위젯을 생성할 때 모든 줄 바꿈이 공백으로 바뀝니다. 어떤 경우에는 이로 인해 쿼리가 유효하지 않게 될 수 있습니다. 예를 들어, 쿼리에 한 줄 주석이 포함되어 있으면 유효하지 않습니다. 명령줄이나 코드형 인프라에서 여러 줄 쿼리를 사용하여 대시보드나 경보를 생성하려고 하면 API가 구문 분석 오류와 함께 작업을 거부합니다.

Amazon OpenSearch Service

데이터 소스 생성

FGAC에 대한 OpenSearch 도메인이 활성화된 경우 커넥터 Lambda 함수의 실행 역할을 OpenSearch Service의 사용자에게 매핑해야 합니다. 자세한 내용은 OpenSearch Service 설명서의 권한 관리에서 역할에 사용자 매핑 섹션을 참조하세요.

Virtual Private Cloud(VPC)에서만 OpenSearch 도메인에 액세스할 수 있는 경우 `AMAZON_OPENSEARCH_ENDPOINT`라는 Lambda 함수에 새 환경 변수를 수동으로 포함해야 합니다. 이 변수 값은 OpenSearch 엔드포인트의 루트 도메인이어야 합니다. OpenSearch Service 콘솔에 나열된 도메인 엔드포인트에서 `https://` 및 `<region>.es.amazonaws.com`을 제거하면 이 루트 도메인을 얻을 수 있습니다. 예를 들어 도메인 엔드포인트가 `https://sample-domain.us-east-1.es.amazonaws.com`인 경우 루트 도메인은 `sample-domain`입니다.

데이터 소스 업데이트

- 다음을 수행하여 데이터 소스를 수동으로 업데이트할 수 있습니다.
  - OpenSearch Service 도메인을 업데이트하려면 데이터 소스 커넥터 Lambda 함수의 `AMAZON_OPENSEARCH_DOMAIN_NAME` 환경 변수를 업데이트합니다.
  - VPC 구성을 업데이트하려면 자세한 내용은 VPC 액세스 구성(콘솔)을 참조하세요.

데이터 소스 쿼리

- OpenSearch Service를 쿼리할 때 다중 소스 쿼리 탭에서 데이터 소스를 선택한 후 다음을 수행합니다.
  - 쿼리할 색인을 선택합니다.
  - 지표 이름(문서의 모든 숫자 필드)과 통계를 선택합니다.
  - 시간 축(문서의 모든 날짜 필드)을 선택합니다.
  - 적용할 필터(문서의 모든 문자열 필드)를 선택합니다.
  - 쿼리 그래프로 표시를 선택합니다.

Amazon RDS for PostgreSQL 및 Amazon RDS for MySQL

데이터 소스 생성

- VPC에서만 데이터 소스에 액세스할 수 있는 경우 마법사로 사전 구축된 데이터 소스에 연결에 설명된 대로 커넥터에 대한 VPC 구성을 포함해야 합니다. 데이터 소스가 보안 인증을 위해 VPC에 연결하려면 엔드포인트가 VPC에 구성되어 있어야 합니다. 자세한 내용을 알아보려면 AWS Secrets Manager VPC 엔드포인트 사용을 참조하세요.

또한 Amazon RDS 서비스에 대한 VPC 엔드포인트도 생성해야 합니다. 자세한 내용은 Amazon RDS API 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조하세요.

데이터 소스 업데이트

- 다음을 수행하여 데이터 소스를 수동으로 업데이트할 수 있습니다.
  - 데이터베이스 인스턴스를 업데이트하려면 데이터 소스 커넥터 Lambda 함수의 `RDS_INSTANCE` 환경 변수를 업데이트합니다.
  - Amazon RDS에 연결하는 데 사용되는 사용자 이름과 암호를 업데이트하려면 AWS Secrets Manager를 사용합니다. 데이터 소스 Lambda 함수의 환경 변수 `RDS_SECRET`에서 데이터 소스에 사용된 보안 암호의 ARN을 찾을 수 있습니다. AWS Secrets Manager에서 보안 암호를 업데이트하는 방법에 대한 자세한 내용은 AWS Secrets Manager 보안 암호 수정을 참조하세요.
  - VPC 구성을 업데이트하려면 자세한 내용은 VPC 액세스 구성(콘솔)을 참조하세요.

데이터 소스 쿼리

- Amazon RDS를 쿼리할 때 다중 소스 쿼리 탭에서 데이터 소스를 선택하고 Amazon RDS 커넥터를 선택한 후 데이터베이스 검색기를 사용하여 사용 가능한 데이터베이스, 테이블 및 열을 볼 수 있습니다. SQL 편집기를 사용하여 SQL 쿼리를 생성할 수도 있습니다.

  쿼리에 다음 변수를 사용할 수 있습니다.

  - `$start.iso` – ISO 데이터 형식의 시작 시간
  - `$end.iso` - ISO 날짜 형식의 종료 시간
  - `$period` - 선택한 기간(초)

  예를 들어, `SELECT value, timestamp FROM table WHERE timestamp BETWEEN $start.iso and $end.iso`를 쿼리할 수 있습니다.
- CloudWatch 데이터 소스 커넥터는 여러 줄 쿼리를 지원하지 않습니다. 쿼리가 실행되거나 쿼리로 경보 또는 대시보드 위젯을 생성할 때 모든 줄 바꿈이 공백으로 바뀝니다. 어떤 경우에는 이로 인해 쿼리가 유효하지 않게 될 수 있습니다. 예를 들어, 쿼리에 한 줄 주석이 포함되어 있으면 유효하지 않습니다. 명령줄이나 코드형 인프라에서 여러 줄 쿼리를 사용하여 대시보드나 경보를 생성하려고 하면 API가 구문 분석 오류와 함께 작업을 거부합니다.

> **Note**
>
> 결과에 날짜 필드가 없는 경우 각 숫자 필드의 값이 단일 값으로 합산되어 제공된 시간 범위에 걸쳐 도표화됩니다. 타임스탬프가 CloudWatch에서 선택한 기간과 일치하지 않는 경우 데이터는 SUM을 사용하여 자동으로 집계되고 CloudWatch의 기간에 맞춰 정렬됩니다.

Amazon S3 CSV 파일

데이터 소스 쿼리

- Amazon S3 CSV 파일을 쿼리할 때는 다중 소스 쿼리 탭에서 데이터 소스를 선택하고 Amazon S3 커넥터를 선택한 후 Amazon S3 버킷과 키를 선택합니다.

  CSV 파일은 다음 방식으로 형식이 지정되어야 합니다.

  - 타임스탬프는 첫 번째 열이어야 합니다.
  - 테이블에는 헤더 행이 있어야 합니다. 헤더는 지표의 이름을 지정하는 데 사용됩니다. 타임스탬프 열의 제목은 무시되고, 지표 열의 제목만 사용됩니다.
  - 타임스탬프는 ISO 날짜 형식이어야 합니다.
  - 지표는 숫자 필드여야 합니다.

```
Timestamp, Metric-1, Metric-2, ...
```

  다음은 그 예제입니다.

타임스탬프	CPU(%)	메모리(%)	스토리지(%)
2023-11-23T17:09:41+00:00	1	2	3
2023-11-23T17:04:41+00:00	4	5	6
2023-11-23T16:59:41+00:00	7	8	9

타임스탬프	CPU(%)	메모리(%)	스토리지(%)
2023-11-23T16:54:41+00:00	10	11	12

> **Note**
>
> 타임스탬프가 제공되지 않은 경우 각 지표의 값이 단일 값으로 합산되어 제공된 시간 범위에 걸쳐 도표화됩니다. 타임스탬프가 CloudWatch에서 선택한 기간과 일치하지 않는 경우 데이터는 SUM을 사용하여 자동으로 집계되고 CloudWatch의 기간에 맞춰 정렬됩니다.

Microsoft Azure Monitor

데이터 소스 생성

- Microsoft Azure Monitor에 연결하려면 테넌트 ID, 클라이언트 ID 및 클라이언트 보안 암호를 제공해야 합니다. 보안 인증 정보는 AWS Secrets Manager에 저장됩니다. 자세한 내용은 Microsoft 설명서의 리소스에 액세스할 수 있는 Microsoft Entra 애플리케이션 및 서비스 주체 만들기를 참조하세요.

데이터 소스 업데이트

- 다음을 수행하여 데이터 소스를 수동으로 업데이트할 수 있습니다.
  - Azure Monitor에 연결하는 데 사용되는 테넌트 ID, 클라이언트 ID 및 클라이언트 보안 암호를 업데이트하기 위해 데이터 소스 Lambda 함수에서 AZURE_CLIENT_SECRET 환경 변수로 데이터 소스에 사용된 암호의 ARN을 찾을 수 있습니다. AWS Secrets Manager에서 보안 암호를 업데이트하는 방법에 대한 자세한 내용은 AWS Secrets Manager 보안 암호 수정을 참조하세요.

데이터 소스 쿼리

- Azure Monitor를 쿼리할 때 다중 소스 쿼리 탭에서 데이터 소스를 선택하고 Azure Monitor 커넥터를 선택한 후 Azure 구독과 리소스 그룹 및 리소스를 지정합니다. 그런 다음, 지표 네임스페이스, 지표 및 집계를 선택하고 측정기준별로 필터링할 수 있습니다.

Prometheus

데이터 소스 생성

- Prometheus 엔드포인트와 Prometheus를 쿼리하는 데 필요한 사용자 및 암호를 제공해야 합니다. 보안 인증 정보는 AWS Secrets Manager에 저장됩니다.
- VPC에서만 데이터 소스에 액세스할 수 있는 경우 마법사로 사전 구축된 데이터 소스에 연결에 설명된 대로 커넥터에 대한 VPC 구성을 포함해야 합니다. 데이터 소스가 보안 인증을 위해 연결하려면 엔드포인트가 VPC에 구성되어 있어야 합니다. 자세한 내용을 알아보려면 AWS Secrets Manager VPC 엔드포인트 사용을 참조하세요.

데이터 소스 구성 업데이트

- 다음을 수행하여 데이터 소스를 수동으로 업데이트할 수 있습니다.
  - Prometheus 엔드포인트를 업데이트하려면 새 엔드포인트를 데이터 소스 Lambda 함수의 `PROMETHEUS_API_ENDPOINT` 환경 변수로 지정합니다.
  - Prometheus에 연결하는 데 사용되는 사용자 이름과 암호를 업데이트하려면 데이터 소스 Lambda 함수에서 `PROMETHEUS_API_SECRET` 환경 변수로 데이터 소스에 사용된 보안 암호의 ARN을 찾을 수 있습니다. AWS Secrets Manager에서 보안 암호를 업데이트하는 방법에 대한 자세한 내용은 AWS Secrets Manager 보안 암호 수정을 참조하세요.
  - VPC 구성을 업데이트하려면 자세한 내용은 VPC 액세스 구성(콘솔)을 참조하세요.

데이터 소스 쿼리

> **Important**
>
> Prometheus 지표 유형은 CloudWatch 지표와 다르며 Prometheus를 통해 사용할 수 있는 많은 지표는 설계상 누적됩니다. Prometheus 지표를 쿼리할 때 CloudWatch는 데이터에 추가 변환을 적용하지 않습니다. 지표 이름 또는 레이블만 지정하는 경우 표시된 값은 누적됩니다. 자세한 내용은 Prometheus 설명서의 Metric types를 참조하세요.
> Prometheus 지표 데이터를 CloudWatch 지표와 같이 불연속형 값으로 보려면 쿼리를 실행하기 전에 쿼리를 편집해야 합니다. 예를 들어, Prometheus 지표 이름에 대해 rate 함수에 대한 호출을 추가해야 할 수도 있습니다. rate 함수와 기타 Prometheus 함수에 대한 설명서는 Prometheus 설명서의 rate()를 참조하세요.

CloudWatch 데이터 소스 커넥터는 여러 줄 쿼리를 지원하지 않습니다. 쿼리가 실행되거나 쿼리로 경보 또는 대시보드 위젯을 생성할 때 모든 줄 바꿈이 공백으로 바뀝니다. 어떤 경우에는 이로 인해 쿼리가 유효하지 않게 될 수 있습니다. 예를 들어, 쿼리에 한 줄 주석이 포함되어 있으면 유효하지 않습니다. 명령줄이나 코드형 인프라에서 여러 줄 쿼리를 사용하여 대시보드나 경보를 생성하려고 하면 API가 구문 분석 오류와 함께 작업을 거부합니다.

사용 가능한 업데이트 알림

때때로 Amazon에서 사용 가능한 최신 버전으로 커넥터를 업데이트할 것을 권장함을 알리고 업데이트 방법에 대한 지침을 제공할 수 있습니다.

데이터 소스에 대한 사용자 지정 커넥터 생성

CloudWatch에 사용자 지정 데이터 소스를 연결하는 데 다음 두 가지 옵션이 있습니다.

- CloudWatch에서 제공하는 샘플 템플릿을 사용하여 시작합니다. 이 템플릿에는 JavaScript 또는 Python을 사용할 수 있습니다. 이러한 템플릿에는 Lambda 함수를 생성할 때 유용하게 사용할 수 있는 샘플 Lambda 코드가 포함되어 있습니다. 그런 다음, 템플릿에서 Lambda 함수를 수정하여 사용자 지정 데이터 소스에 연결할 수 있습니다.
- CloudWatch에서 사용할 데이터 소스 커넥터, 데이터 쿼리, 시계열 준비를 구현하는 AWS Lambda 함수를 처음부터 생성합니다. 이 함수는 필요한 경우 데이터 포인트를 사전 집계하거나 병합해야 하며 CloudWatch와 호환되도록 기간과 타임스탬프도 정렬해야 합니다.

목차

템플릿 사용

템플릿을 사용하면 샘플 Lambda 함수가 생성되므로 사용자 지정 커넥터를 더 빨리 구축할 수 있습니다. 이러한 샘플 함수는 사용자 지정 커넥터 구축과 관련된 여러 일반적인 시나리오에 대한 샘플 코드를 제공합니다. 템플릿을 사용하여 커넥터를 생성한 후 Lambda 코드를 검사한 다음, 이를 수정하여 데이터 소스에 연결하는 데 사용할 수 있습니다.

또한 템플릿을 사용하는 경우 CloudWatch가 Lambda 권한 정책을 생성하고 Lambda 함수에 리소스 태그를 연결합니다.

템플릿을 사용하여 사용자 지정 데이터 소스에 대한 커넥터 생성

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. 지표 데이터 소스 탭을 선택합니다.
4. 데이터 소스 생성을 선택합니다.
5. 사용자 지정 - 시작하기 템플릿의 라디오 버튼을 선택한 후 다음을 선택합니다.
6. 데이터 소스 이름을 입력합니다.
7. 나열된 템플릿 중 하나를 선택합니다.
8. Node.js 또는 Python을 선택합니다.
9. 데이터 소스 생성을 선택합니다.

   방금 추가한 새 사용자 지정 소스는 AWS CloudFormation 스택 생성이 완료될 때까지 표시되지 않습니다. 진행 상황을 확인하려면 내 CloudFormation 스택의 상태 보기를 선택합니다. 또는 새로 고침 아이콘을 선택하여 이 목록을 업데이트할 수 있습니다.

   새 데이터 소스가 이 목록에 나타나면 콘솔에서 테스트하고 수정할 준비가 된 것입니다.
10. (선택 사항) 콘솔에서 이 소스의 테스트 데이터를 쿼리하려면 다른 데이터 소스의 지표 그래프 생성의 지침을 따릅니다.
11. 필요에 맞게 Lambda 함수를 수정합니다.

    a. 탐색 창에서 설정을 선택합니다.

    b. 지표 데이터 소스 탭을 선택합니다.

    c. 수정하려는 소스에 대해 Lambda 콘솔에서 보기를 선택합니다.

이제 데이터 소스에 액세스하도록 함수를 수정할 수 있습니다. 자세한 내용은 1단계: 함수 생성 단원을 참조하십시오.

> **Note**
>
> 템플릿을 사용하면 Lambda 함수를 작성할 때 2단계: Lambda 권한 정책 생성 및 3단계: Lambda 함수에 리소스 태그 연결의 지침을 따를 필요가 없습니다. 템플릿을 사용했기 때문에 CloudWatch에서 이러한 단계를 수행했습니다.

처음부터 사용자 지정 데이터 소스 생성

이 섹션의 단계에 따라 데이터 소스에 CloudWatch를 연결하는 Lambda 함수를 생성합니다.

1단계: 함수 생성

사용자 지정 데이터 소스 커넥터는 CloudWatch의 `GetMetricData` 이벤트를 지원해야 합니다. 필요에 따라 `DescribeGetMetricData` 이벤트를 구현하여 CloudWatch 콘솔에서 커넥터 사용 방법에 대한 설명서를 사용자에게 제공할 수도 있습니다. `DescribeGetMetricData` 응답을 사용하여 CloudWatch 사용자 지정 쿼리 작성기에 사용되는 기본값을 설정할 수도 있습니다.

CloudWatch는 시작하는 데 도움이 되는 코드 조각을 샘플로 제공합니다. 자세한 내용은 https://github.com/aws-samples/cloudwatch-data-source-samples의 샘플 리포지토리를 참조하세요.

제약 조건

- Lambda의 응답은 6Mb보다 작아야 합니다. 응답이 6Mb를 초과하는 경우 `GetMetricData` 응답은 Lambda 함수를 `InternalError`로 표시하고 데이터가 반환되지 않습니다.
- Lambda 함수는 시각화 및 대시보드 작성의 경우 10초 이내에, 경보 사용의 경우 4.5초 이내에 실행을 완료해야 합니다. 실행 시간이 해당 시간을 초과하는 경우 `GetMetricData` 응답은 Lambda 함수를 `InternalError`로 표시하고 데이터가 반환되지 않습니다.
- Lambda 함수는 epoch 타임스탬프를 사용하여 출력을 초 단위로 전송해야 합니다.
- Lambda 함수가 데이터를 리샘플링하지 않고 대신 CloudWatch 사용자가 요청한 시작 시간 및 기간 길이에 해당하지 않는 데이터를 반환하는 경우 CloudWatch에서 해당 데이터를 무시합니다. 추가 데이터는 모든 시각화 또는 경보에서 삭제됩니다. 시작 시간과 종료 시간 사이에 있지 않은 데이터도 모두 삭제됩니다.

예를 들어, 사용자가 5분 간격으로 10:00~11:00에 데이터를 요청하는 경우 '10:00:00~10:04:59' 및 '10:05:00~10:09:59'가 반환될 데이터의 유효한 시간 범위입니다. `10:00 value1`, `10:05 value2` 등을 포함하는 시계열을 반환해야 합니다. 예를 들어, 함수가 `10:03 valueX`를 반환하는 경우 10:03이 요청된 시작 시간 및 기간과 일치하지 않기 때문에 삭제됩니다.

- CloudWatch 데이터 소스 커넥터는 여러 줄 쿼리를 지원하지 않습니다. 쿼리가 실행되거나 쿼리로 경보 또는 대시보드 위젯을 생성할 때 모든 줄 바꿈이 공백으로 바뀝니다. 어떤 경우에는 이로 인해 쿼리가 유효하지 않게 될 수 있습니다.

GetMetricData 이벤트

요청 페이로드

다음은 Lambda 함수에 입력으로 전송된 `GetMetricData` 요청 페이로드의 예제입니다.

```
{
  "EventType": "GetMetricData",
  "GetMetricDataRequest": {
    "StartTime": 1697060700,
    "EndTime": 1697061600,
    "Period": 300,
    "Arguments": ["serviceregistry_external_http_requests{host_cluster!=\"prod\"}"]
  }
}
```

- StartTime - 반환할 가장 빠른 데이터를 지정하는 타임스탬프입니다. 유형은 타임스탬프 epoch 초입니다.
- EndTime - 반환할 최신 데이터를 지정하는 타임스탬프입니다. 유형은 타임스탬프 epoch 초입니다.
- Period - 지표 데이터의 각 집계가 나타내는 시간(초)입니다. 최소 시간은 60초입니다. 유형은 초입니다.
- Arguments - Lambda 지표 수학 표현식에 전달할 인수 배열입니다. 인수 전달에 대한 자세한 내용은 Lambda 함수에 인수를 전달하는 방법 섹션을 참조하세요.

응답 페이로드

다음은 Lambda 함수에서 반환하는 `GetMetricData` 응답 페이로드의 예제입니다.

```
{
```

```
   "MetricDataResults": [
      {
         "StatusCode": "Complete",
         "Label": "CPUUtilization",
         "Timestamps": [ 1697060700, 1697061000, 1697061300 ],
         "Values": [ 15000, 14000, 16000 ]
      }
   ]
}
```

응답 페이로드는 MetricDataResults 필드 또는 Error 필드 중 하나를 포함하되 두 필드 모두를 포함하지는 않습니다.

MetricDataResults 필드는 MetricDataResult 유형의 시계열 필드 목록입니다. 각 시계열 필드는 다음 필드를 포함할 수 있습니다.

- StatusCode – (선택 사항) Complete은 요청된 시간 범위의 모든 데이터 포인트가 반환되었음을 나타냅니다. PartialData는 불완전한 데이터 포인트 세트가 반환되었음을 의미합니다. 생략 시 기본값은 Complete입니다.

  유효한 값: Complete | InternalError | PartialData | Forbidden
- Messages – 반환된 데이터에 대한 추가 정보가 포함된 선택적 메시지 목록입니다.

  유형: Code 및 Value 문자열이 포함된 MessageData 객체의 배열입니다.
- Label - 데이터와 연결된 사람이 읽을 수 있는 레이블입니다.

  타입: 문자열
- Timestamps – epoch 시간으로 형식이 지정된 데이터 포인트의 타임스탬프입니다. 타임스탬프 수는 항상 값 수와 일치하며 Timestamps[x] 값은 Values[x]입니다.

  유형: 타임스탬프의 배열입니다.
- Values – Timestamps에 해당하는 지표의 데이터 포인트 값입니다. 값 수는 항상 타임스탬프 수와 일치하며 Timestamps[x] 값은 Values[x]입니다.

  유형: 실수의 배열입니다.

Error 객체에 대한 자세한 내용은 다음 섹션을 참조하세요.

오류 응답 형식

필요에 따라 오류 응답을 사용하여 오류에 대한 자세한 정보를 제공할 수 있습니다. 파라미터가 누락되었거나 유형이 잘못된 경우와 같이 검증 오류가 발생하는 경우 코드 검증을 통해 오류를 반환하는 것이 좋습니다.

다음은 Lambda 함수가 `GetMetricData` 검증 예외를 발생시키려고 할 때 응답의 예제입니다.

```
{
   "Error": {
      "Code": "Validation",
      "Value": "Invalid Prometheus cluster"
   }
}
```

다음은 Lambda 함수가 액세스 문제로 인해 데이터를 반환할 수 없다고 표시할 때의 응답 예제입니다. 응답은 상태 코드가 `Forbidden`인 단일 시계열로 변환됩니다.

```
{
   "Error": {
      "Code": "Forbidden",
      "Value": "Unable to access ..."
   }
}
```

다음은 Lambda 함수가 전체 `InternalError` 예외를 발생시키는 예로, 상태 코드가 `InternalError`이고 메시지가 있는 단일 시계열로 변환됩니다. CloudWatch는 오류 코드에 `Validation` 또는 `Forbidden` 이외의 값이 있을 때마다 일반적인 내부 오류로 간주합니다.

```
{
   "Error": {
      "Code": "PrometheusClusterUnreachable",
      "Value": "Unable to communicate with the cluster"
   }
}
```

DescribeGetMetricData 이벤트

요청 페이로드

다음은 `DescribeGetMetricData` 요청 페이로드의 예제입니다.

```
{
  "EventType": "DescribeGetMetricData"
```

```
}
```

응답 페이로드

다음은 DescribeGetMetricData 응답 페이로드의 예제입니다.

```
{
    "Description": "Data source connector",
    "ArgumentDefaults": [{
        Value: "default value"
     }]
}
```

- Description – 데이터 소스 커넥터를 사용하는 방법에 대한 설명입니다. 이 설명은 CloudWatch 콘솔에 표시됩니다. 마크다운이 지원됩니다.

  타입: 문자열
- ArgumentDefaults – 사용자 지정 데이터 소스 작성기를 미리 채우는 데 사용되는 인수 기본값의 선택적 배열입니다.

  `[{ Value: "default value 1"}, { Value: 10}]`이 반환되면 CloudWatch 콘솔의 쿼리 작성기는 2개의 입력을 표시합니다. 첫 번째 입력은 "default value 1"이고 두 번째 입력은 10입니다.

  `ArgumentDefaults`가 제공되지 않으면 기본 유형이 `String`으로 설정된 단일 입력이 표시됩니다.

  유형: 값과 유형을 포함하는 객체 배열입니다.
- Error - (선택 사항) 모든 응답에 오류 필드를 포함할 수 있습니다. GetMetricData 이벤트에서 예제를 볼 수 있습니다.

CloudWatch 경보에 대한 중요 고려 사항

데이터 소스를 사용하여 CloudWatch 경보를 설정하려는 경우 1분마다 타임스탬프가 있는 데이터를 CloudWatch에 보고하도록 설정해야 합니다. 연결된 데이터 소스의 지표에 대한 경보를 생성하는 방법에 대한 자세한 내용과 기타 고려 사항은 연결된 데이터 소스를 기반으로 경보 생성 섹션을 참조하세요.

(선택 사항) AWS Secrets Manager를 사용하여 보안 인증 저장

Lambda 함수가 보안 인증을 사용하여 데이터 소스에 액세스해야 하는 경우 Lambda 함수에 보안 인증을 하드코딩하는 대신 AWS Secrets Manager를 사용하여 보안 인증을 저장하는 것이 좋습니다.

Lambda에서 AWS Secrets Manager를 사용하는 방법에 대한 자세한 내용은 AWS Lambda 함수에서 AWS Secrets Manager 보안 암호 사용을 참조하세요.

(선택 사항) VPC의 데이터 소스에 연결

데이터 소스가 Amazon Virtual Private Cloud에서 관리하는 VPC에 있는 경우 이에 액세스하려면 Lambda 함수를 구성해야 합니다. 자세한 내용은 아웃바운드 네트워킹을 VPC의 리소스에 연결을 참조하세요.

AWS Secrets Manager와 같은 서비스에 액세스하려면 VPC 서비스 엔드포인트를 구성해야 할 수도 있습니다. 자세한 내용은 인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스에 액세스를 참조하세요.

2단계: Lambda 권한 정책 생성

생성한 Lambda 함수를 사용할 수 있는 권한을 CloudWatch에 부여하는 정책 설명 생성을 사용해야 합니다. AWS CLI 또는 Lambda 콘솔을 사용하여 정책 설명을 생성할 수 있습니다.

AWS CLI를 사용하여 정책 설명 생성

- 다음 명령을 입력합니다. *123456789012*를 계정 ID로 바꾸고, *my-data-source-function*을 Lambda 함수 이름으로 바꾸고, *MyDataSource-DataSourcePermission1234*를 임의의 고유 값으로 바꿉니다.

```
aws lambda add-permission --function-name my-data-source-function --statement-
id MyDataSource-DataSourcePermission1234 --action lambda:InvokeFunction --principal
 lambda.datasource.cloudwatch.amazonaws.com --source-account 123456789012
```

3단계: Lambda 함수에 리소스 태그 연결

CloudWatch 콘솔은 태그를 사용하여 어떤 Lambda 함수가 데이터 소스 커넥터인지 결정합니다. AWS CloudFormation 마법사 중 하나를 사용하여 데이터 소스를 생성하면 이를 구성하는 스택에 의해 태그가 자동으로 적용됩니다. 데이터 소스를 직접 생성할 때 Lambda 함수에 다음 태그를 사용할 수 있습니다. 이렇게 하면 지표를 쿼리할 때 CloudWatch 콘솔의 데이터 소스 드롭다운에 커넥터가 표시됩니다.

- `cloudwatch:datasource`가 키이고 `custom`이 값인 태그

사용자 지정 데이터 소스 사용

데이터 소스를 생성한 후 이를 사용하여 해당 소스의 데이터를 쿼리하여 시각화하고 경보를 설정할 수 있습니다. 템플릿을 사용하여 사용자 지정 데이터 소스 커넥터를 생성했거나 3단계: Lambda 함수에 리소스 태그 연결에 나열된 태그를 추가한 경우 다른 데이터 소스의 지표 그래프 생성의 단계에 따라 쿼리할 수 있습니다.

다음 섹션에 설명된 대로 지표 수학 함수 LAMBDA를 사용하여 쿼리할 수도 있습니다.

데이터 소스의 지표에 대한 경보를 생성하는 방법에 대한 자세한 내용은 연결된 데이터 소스를 기반으로 경보 생성 섹션을 참조하세요.

Lambda 함수에 인수를 전달하는 방법

사용자 지정 데이터 소스에 인수를 전달하는 권장 방법은 데이터 소스를 쿼리할 때 CloudWatch 콘솔의 쿼리 작성기를 사용하는 것입니다.

CloudWatch 지표 수학의 새로운 LAMBDA 표현식을 사용하여 Lambda 함수로 데이터 소스에서 데이터를 검색할 수도 있습니다.

```
LAMBDA("LambdaFunctionName" [, optional-arg]*)
```

`optional-arg`는 최대 20개의 문자열, 숫자 또는 부울입니다. 예: `param`, `3.14` 또는 `true`.

> **Note**
>
> CloudWatch 데이터 소스 커넥터는 여러 줄 문자열을 지원하지 않습니다. 쿼리가 실행되거나 쿼리로 경보 또는 대시보드 위젯을 생성할 때 모든 줄 바꿈이 공백으로 바뀝니다. 어떤 경우에는 이로 인해 쿼리가 유효하지 않게 될 수 있습니다.

LAMBDA 지표 수학 함수를 사용하는 경우 함수 이름(`"MyFunction"`)을 제공할 수 있습니다. 리소스 정책이 허용하는 경우 특정 버전의 함수(`"MyFunction:22"`) 또는 Lambda 함수 별칭(`"MyFunction:MyAlias"`)을 사용할 수도 있습니다. *를 사용할 수 없습니다.

다음은 LAMBDA 함수를 호출하는 몇 가지 예입니다.

```
LAMBDA("AmazonOpenSearchDataSource", "MyDomain", "some-query")
```

```
LAMBDA("MyCustomDataSource", true, "fuzzy", 99.9)
```

LAMBDA 지표 수학 함수는 요청자에게 반환되거나 다른 지표 수학 함수와 결합될 수 있는 시계열 목록을 반환합니다. 다음은 LAMBDA를 다른 지표 수학 함수와 결합하는 예제입니다.

```
FILL(LAMBDA("AmazonOpenSearchDataSource", "MyDomain", "some-query"), 0)
```

데이터 소스에 대한 커넥터 삭제

데이터 소스에 대한 커넥터를 삭제하려면 이 섹션의 지침을 따르세요.

데이터 소스에 대한 커넥터 삭제

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. 지표 데이터 소스 탭을 선택합니다.
4. 삭제하려는 데이터 소스의 행에서 CloudFormation에서 관리를 선택합니다.

   이제 AWS CloudFormation 콘솔로 이동하게 됩니다.
5. 데이터 소스 이름이 표시된 섹션에서 삭제를 선택합니다.
6. 확인 팝업에서 삭제를 선택합니다.

CloudWatch 에이전트를 사용하여 지표, 로그, 추적 수집

통합 CloudWatch 에이전트를 사용하면 다음을 수행할 수 있습니다.

- 운영 체제 전반에 걸쳐 Amazon EC2 인스턴스에서 내부 시스템 수준 지표를 수집할 수 있습니다. 지표에는 EC2 인스턴스 지표뿐만 아니라 인게스트 지표도 포함될 수 있습니다. 수집할 수 있는 추가 지표는 CloudWatch 에이전트가 수집하는 지표에 나열되어 있습니다.
- 온프레미스 서버로부터 시스템 수준 지표를 수집합니다. 여기에는 AWS가 관리하지 않는 서버뿐만 아니라 하이브리드 환경의 서버도 포함될 수 있습니다.
- `StatsD` 및 `collectd` 프로토콜을 사용하여 애플리케이션 또는 서비스에서 사용자 지정 지표를 검색할 수 있습니다. `StatsD`는 Windows Server가 실행되는 서버와 Linux 서버에서 모두 지원되며, `collectd`는 Linux 서버에서만 지원됩니다.
- Linux 또는 Windows Server를 실행하는 Amazon EC2 인스턴스 및 온프레미스 서버에서 로그를 수집할 수 있습니다.

  > **Note**
  >
  > CloudWatch 에이전트는 FIFO 파이프에서의 로그 수집을 지원하지 않습니다.

- 버전 1.300031.0 이상을 사용하여 CloudWatch Application Signals를 활성화할 수 있습니다. 자세한 내용은 Application Signals 단원을 참조하십시오.
- 버전 1.300025.0 이상은 OpenTelemetry 또는 X-Ray 클라이언트 SDK에서 추적을 수집하여 X-Ray에 전송할 수 있습니다.

  CloudWatch 에이전트를 사용하면 별도의 트레이스 수집 대몬(daemon)을 실행할 필요 없이 트레이스를 수집할 수 있으므로 실행하고 관리하는 에이전트 수를 줄일 수 있습니다.

다른 CloudWatch 지표와 마찬가지로 CloudWatch에서 CloudWatch 에이전트를 사용하여 수집한 지표를 저장하고 볼 수 있습니다. CloudWatch 에이전트가 수집하는 지표의 기본 네임스페이스는 `CWAgent`이지만, 에이전트를 구성할 때 다른 네임스페이스를 지정할 수 있습니다.

통합 CloudWatch 에이전트가 수집한 로그는 이전 CloudWatch Logs 에이전트가 수집한 로그와 마찬가지로 Amazon CloudWatch Logs에서 처리되고 저장됩니다. CloudWatch Logs 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

CloudWatch 에이전트가 수집한 지표는 사용자 지정 지표로 청구됩니다. CloudWatch 지표 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

CloudWatch 에이전트는 MIT 라이선스에 따라 오픈 소스이며 GitHub에 호스트됩니다. CloudWatch 에이전트를 구축 또는 사용자 지정하거나 이에 기여하려는 경우 GitHub 리포지토리에서 최신 지침을 참조하세요. 잠재적인 보안 문제를 찾았다고 생각한다면 GitHub 또는 공개 게시판에 게시하지 마세요. 대신 취약성 보고의 지침을 따르거나 AWS 보안에 직접 이메일을 보내시기 바랍니다.

이 단원의 단계에서는 Amazon EC2 인스턴스 및 온프레미스 서버에 통합 CloudWatch 에이전트를 설치하는 방법을 설명합니다. CloudWatch 에이전트가 수집할 수 있는 지표에 대한 자세한 내용은 CloudWatch 에이전트가 수집하는 지표 단원을 참조하세요.

지원되는 운영 체제

CloudWatch 에이전트는 다음 운영 체제의 x86-64 아키텍처에서 지원됩니다. 여기에 나열된 각 주요 버전의 모든 마이너 버전 업데이트에서도 지원됩니다.

- Amazon Linux 2023
- Amazon Linux 2
- Ubuntu 서버 버전 23.10, 22.04, 20.04, 18.04, 16.04 및 14.04
- CentOS 버전 9, 8, 7
- Red Hat Enterprise Linux(RHEL) 버전 9, 8, 7
- Debian 버전 12, 11 및 10
- SUSE Linux Enterprise Server(SLES) 버전 15 및 버전 12
- Oracle Linux 버전 9, 8 및 7
- AlmaLinux 버전 9 및 8
- Rocky Linux 버전 9 및 8
- 다음 macOS 컴퓨터: EC2 M1 Mac1 인스턴스 및 macOS 14(Sonoma), macOS 13(Ventura) 그리고 macOS 12(Monterey)를 실행하는 컴퓨터
- 64비트 버전의 Windows Server 2022, Windows Server 2019, Windows Server 2016
- 64비트 Windows 10

에이전트는 다음 운영 체제의 ARM64 아키텍처에서 지원됩니다. 여기에 나열된 각 주요 버전의 모든 마이너 버전 업데이트에서도 지원됩니다.

- Amazon Linux 2023
- Amazon Linux 2
- Ubuntu 서버 버전 23.10, 22.04, 20.04, 18.04 및 16.04

- CentOS 버전 9, 8
- Red Hat Enterprise Linux(RHEL) 버전 9, 8, 7
- Debian 버전 12, 11 및 10
- SUSE Linux Enterprise Server 15
- 다음 macOS 컴퓨터: macOS 14(Sonoma), macOS 13(Ventura) 및 macOS 12(Monterey)

설치 프로세스 개요

명령줄을 사용하여 CloudWatch 에이전트를 수동으로 다운로드하여 설치하거나 SSM과 통합할 수 있습니다. 두 방법 중 어떤 방법을 사용하든 CloudWatch 에이전트를 설치하는 일반적인 흐름은 다음과 같습니다.

1. 에이전트가 서버에서 지표를 수집하고 필요한 경우 AWS Systems Manager와 통합할 수 있게 하는 IAM 역할 또는 사용자를 생성합니다.
2. 에이전트 패키지를 다운로드합니다.
3. CloudWatch 에이전트 구성 파일을 수정하고 수집하려는 지표를 지정합니다.
4. 서버에 에이전트를 설치하고 시작합니다. EC2 인스턴스에 에이전트를 설치할 때 1단계에서 생성한 IAM 역할을 연결합니다. 온프레미스 서버에 에이전트를 설치할 때 1단계에서 생성한 IAM 사용자의 자격 증명이 포함되어 있는 명명된 프로파일을 지정합니다.

내용

CloudWatch 에이전트 설치

CloudWatch 에이전트는 Amazon Linux 2023 및 Amazon Linux 2에서 패키지로 사용할 수 있습니다. 이러한 운영 체제 중 하나를 사용하는 경우 다음 명령을 입력하여 패키지를 설치할 수 있습니다. 또한 인스턴스에 연결된 IAM 역할에 CloudWatchAgentServerPolicy가 연결되어 있는지 확인해야 합니다.

자세한 내용은 Amazon EC2 인스턴스에서 CloudWatch 에이전트와 함께 사용할 IAM 역할 생성 단원을 참조하세요.

```
sudo yum install amazon-cloudwatch-agent
```

Linux 및 Windows Server 등 지원되는 모든 운영 체제에서 Amazon S3 다운로드 링크와 함께 명령줄을 사용하거나 Amazon EC2 Systems Manager를 사용하거나 또는 AWS CloudFormation 템플릿을 사용하여 CloudWatch 에이전트를 다운로드하고 설치할 수 있습니다. 세부 정보는 다음 단원을 참조하세요.

내용

명령줄을 사용하여 CloudWatch 에이전트 설치

다음 주제를 참조하여 CloudWatch 에이전트 패키지를 다운로드하고 구성하며 설치할 수 있습니다.

주제

명령줄을 사용하여 CloudWatch 에이전트 다운로드 및 구성

다음 단계를 통해 CloudWatch 에이전트 패키지를 다운로드하고, IAM 역할 또는 사용자를 생성하며, 필요한 경우 일반 구성 파일을 수정합니다.

CloudWatch 에이전트 패키지 다운로드

> **Note**
>
> CloudWatch 에이전트를 다운로드하려면 연결에서 TLS 1.2 이상을 사용해야 합니다.

CloudWatch 에이전트는 Amazon Linux 2023 및 Amazon Linux 2에서 패키지로 사용할 수 있습니다. 이 운영 체제를 사용하는 경우 다음 명령을 입력하여 패키지를 설치할 수 있습니다. 또한 인스턴스에 연결된 IAM 역할에 CloudWatchAgentServerPolicy가 연결되어 있는지 확인해야 합니다. 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성 단원을 참조하세요.

```
sudo yum install amazon-cloudwatch-agent
```

지원되는 모든 운영 체제에서 명령줄을 사용하여 CloudWatch 에이전트를 다운로드하고 설치할 수 있습니다.

각 다운로드 링크에는 전체 링크뿐 아니라 리전별 링크도 있습니다. 예를 들어 Amazon Linux 2023, Amazon Linux 2 및 x86-64 아키텍처의 경우 다음과 같은 유효한 다운로드 링크 세 개가 있습니다.

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

또한 에이전트의 최신 변경 사항에 대한 README 파일과 다운로드할 수 있는 버전 번호를 나타내는 파일을 다운로드할 수 있습니다. 이러한 파일은 다음 위치에 있습니다.

- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/RELEASE_NOTES` 또는 `https://amazoncloudwatch-agent-`*`region`*`.s3.`*`region`*`.amazonaws.com/info/latest/RELEASE_NOTES`
- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/CWAGENT_VERSION` 또는 `https://amazoncloudwatch-agent-`*`region`*`.s3.`*`region`*`.amazonaws.com/info/latest/CWAGENT_VERSION`

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
x86-64	Amazon Linux 2023	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
	및 Amazon Linux 2	amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm	amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
x86-64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg.sig
x86-64	Windows	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
ARM64	Amazon Linux 2023 및 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
ARM64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	MacOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/arm64/latest/amazon-cloudwatch-agent.pkg https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/darwin/arm64/latest/amazon-cloudwatch-agent.pkg	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/arm64/latest/amazon-cloudwatch-agent.pkg.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/darwin/arm64/latest/amazon-cloudwatch-agent.pkg.sig

명령줄을 사용하여 CloudWatch 에이전트 패키지를 다운로드하고 설치하려면

1. CloudWatch 에이전트를 다운로드합니다.

   Linux 서버의 경우 다음을 입력합니다. *download-link*에 이전 테이블의 해당 다운로드 링크를 사용합니다.

   ```
   wget download-link
   ```

   Windows Server가 실행되는 서버의 경우 다음 파일을 다운로드합니다.

   ```
   https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi
   ```

2. 패키지를 다운로드했으면 선택 사항으로 패키지 서명을 확인할 수 있습니다. 자세한 내용은 CloudWatch 에이전트 패키지의 서명 확인 단원을 참조하세요.
3. 패키지를 설치합니다. Linux 서버에서 RPM 패키지를 다운로드한 경우 패키지가 있는 디렉터리로 변경하고 다음을 입력합니다.

   ```
   sudo rpm -U ./amazon-cloudwatch-agent.rpm
   ```

   Linux 서버에서 DEB 패키지를 다운로드한 경우 패키지가 있는 디렉터리로 변경하고 다음을 입력합니다.

   ```
   sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
   ```

   Windows Server가 실행되는 서버에서 MSI 패키지를 다운로드한 경우 패키지가 있는 디렉터리로 변경하고 다음을 입력합니다.

   ```
   msiexec /i amazon-cloudwatch-agent.msi
   ```

   또한 이 명령은 PowerShell 내에서도 작동합니다. MSI 명령 옵션에 대한 자세한 내용은 Microsoft Windows 문서의 명령줄 옵션 단원을 참조하세요.

   macOS 서버에 PKG 패키지를 다운로드한 경우 패키지가 포함된 디렉터리로 변경하고 다음을 입력합니다.

   ```
   sudo installer -pkg ./amazon-cloudwatch-agent.pkg -target /
   ```

에이전트 구성 파일 생성 및 수정

CloudWatch 에이전트를 다운로드했으면 서버에서 에이전트를 시작하기 전에 구성 파일을 생성해야 합니다. 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.

CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성

AWS 리소스에 액세스하려면 권한이 필요합니다. IAM 역할, IAM 사용자 또는 둘 다를 생성하여 CloudWatch 에이전트가 CloudWatch에 지표를 작성하는 데 필요한 권한을 부여합니다. Amazon EC2 인스턴스에서 에이전트를 사용하려면 IAM 역할을 생성해야 합니다. 온프레미스 서버에서 에이전트를 사용하려면 IAM 사용자를 생성해야 합니다.

> **Note**
>
> 최근에 당사는 고객들에게 정책을 직접 생성하도록 요구하는 대신 Amazon에서 만든 새로운 `CloudWatchAgentServerPolicy` 및 `CloudWatchAgentAdminPolicy` 정책을 사용하여 다음과 같은 절차를 수정했습니다. 파라미터 스토어에 파일을 작성하고 파라미터 스토어에서 파일을 다운로드하는 경우 Amazon에서 생성한 정책은 이름이 `AmazonCloudWatch-`로 시작하는 파일만 지원합니다. 파일 이름이 `AmazonCloudWatch-`로 시작하지 않는 CloudWatch 에이전트 구성 파일이 있는 경우 이러한 정책을 사용하여 파라미터 스토어에 파일을 작성하거나 파라미터 스토어에서 파일을 다운로드할 수 없습니다.

Amazon EC2 인스턴스에서 CloudWatch 에이전트를 실행하려는 경우 다음 단계에 따라 필요한 IAM 역할을 생성합니다. 이 역할은 정보를 인스턴스에서 읽고 CloudWatch에 쓸 수 있는 권한을 제공합니다.

EC2 인스턴스에서 CloudWatch 에이전트를 실행하는 데 필요한 IAM 역할을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Roles(역할)를 선택한 후 Create role(역할 생성)을 선택합니다.
3. 신뢰할 수 있는 엔터티 유형(Trusted entity type)에서 AWS 서비스(AWS service)를 선택해야 합니다.
4. 사용 사례(Use case)의 경우 일반 사용 사례(Common use cases)에서 EC2를 선택하고,
5. 다음을 선택합니다.
6. 정책 목록에서 CloudWatchAgentServerPolicy 옆의 확인란을 선택합니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다.
7. (선택 사항) 에이전트가 X-Ray를 추적을 전송하려는 경우 해당 역할에 AWSXRayDaemonWriteAccess 정책도 제공해야 합니다. 이렇게 하려면 목록에서 해당 정책을 찾아 옆에 있는 확인란을 선택하세요.
8. 다음을 선택합니다.
9. Role name(역할 이름)에 *`CloudWatchAgentServerRole`* 등의 역할 이름을 입력합니다. 필요한 경우 설명을 입력합니다. 그런 다음 역할 생성을 선택합니다.

   이제 역할이 생성되었습니다.
10. (선택 사항) 에이전트가 CloudWatch Logs에 로그를 전송하고 에이전트가 이러한 로그 그룹에 대한 보존 정책을 설정할 수 있도록 하려면 역할에 `logs:PutRetentionPolicy` 권한을 추가해야

합니다. 자세한 내용은 CloudWatch 에이전트가 로그 보존 정책을 설정하도록 허용 단원을 참조하십시오.

온프레미스 서버에서 CloudWatch 에이전트를 실행하려는 경우 다음 단계에 따라 필요한 IAM 사용자를 생성합니다.

> **Warning**
>
> 이 시나리오에서는 프로그래밍 방식 액세스 권한과 장기 보안 인증이 있는 IAM 사용자가 필요하며 이는 보안 위험을 내포합니다. 이 위험을 줄이려면 이러한 사용자에게 작업을 수행하는 데 필요한 권한만 제공하고 더 이상 필요하지 않을 경우 이러한 사용자를 제거하는 것이 좋습니다. 필요한 경우 액세스 키를 업데이트할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 액세스 키 업데이트를 참조하세요.

CloudWatch 에이전트가 온프레미스 서버에서 실행되는 데 필요한 IAM 사용자를 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 사용자(Users)를 선택한 후 사용자 추가(Add user)를 선택합니다.
3. 새 사용자의 사용자 이름을 입력합니다.
4. 액세스 키 - 프로그래밍 방식 액세스(Access key - Programmatic access)를 선택하고 다음: 권한(Next: Permissions)을 선택합니다.
5. 기존 정책 직접 첨부를 선택합니다.
6. 정책 목록에서 CloudWatchAgentServerPolicy 옆의 확인란을 선택합니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다.
7. (선택 사항) 에이전트가 X-Ray를 추적하려는 경우 해당 역할에 AWSXRayDaemonWriteAccess 정책도 제공해야 합니다. 이렇게 하려면 목록에서 해당 정책을 찾아 옆에 있는 확인란을 선택하세요.
8. 다음: 태그를 선택합니다.
9. 필요에 따라 새 IAM 사용자에 대한 태그를 생성한 다음 다음: 검토(Next: Review)를 선택합니다.
10. 올바른 정책이 나열되었는지 확인하고 사용자 생성(Create user)을 선택합니다.
11. 새 사용자 이름 옆에서 표시를 선택합니다. 에이전트를 설치할 때 사용할 수 있도록 액세스 키 및 보안 키를 파일에 복사한 다음, 닫기를 선택하세요.

CloudWatch 에이전트가 로그 보존 정책을 설정하도록 허용

CloudWatch 에이전트가 로그 이벤트를 전송하는 로그 그룹에 대한 보존 정책을 설정하도록 구성할 수 있습니다. 이렇게 하면 에이전트가 사용하는 IAM 역할 또는 사용자에게 `logs:PutRetentionPolicy`를 부여해야 합니다. 에이전트는 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행하고 온프레미스 서버에는 IAM 사용자를 사용합니다.

CloudWatch 에이전트의 IAM 역할에 로그 보존 정책을 설정할 수 있는 권한 부여

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 검색 상자에 CloudWatch 에이전트의 IAM 역할 이름의 시작을 입력합니다. 역할을 생성했을 때 이 이름을 선택했습니다. 이름은 `CloudWatchAgentServerRole`이 될 수 있습니다.

   역할이 표시되면 역할의 이름을 선택합니다.
4. 권한(Permissions) 탭에서 권한 추가(Add permissions), 인라인 정책 생성(Create inline policy)을 선택합니다.
5. JSON 탭을 선택하고 다음 정책을 상자에 복사하고 상자의 기본 JSON을 대체합니다.

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

6. 정책 검토를 선택합니다.
7. 이름(Name)의 경우 **`CloudWatchAgentPutLogsRetention`** 또는 이와 비슷한 것을 입력하고 정책 생성(Create policy)을 선택합니다.

CloudWatch 에이전트의 IAM 사용자에게 로그 보존 정책을 설정할 수 있는 권한 부여

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 사용자를 선택합니다.
3. 검색 상자에 CloudWatch 에이전트의 IAM 사용자 이름의 시작을 입력합니다. 사용자를 생성했을 때 이 이름을 선택했습니다.

   사용자가 표시되면 사용자의 이름을 선택합니다.
4. 권한(Permissions) 탭에서 인라인 정책 추가(Add inline policy)를 선택합니다.
5. JSON 탭을 선택하고 다음 정책을 상자에 복사하고 상자의 기본 JSON을 대체합니다.

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

6. 정책 검토를 선택합니다.
7. 이름(Name)의 경우 **`CloudWatchAgentPutLogsRetention`** 또는 이와 비슷한 것을 입력하고 정책 생성(Create policy)을 선택합니다.

서버에 CloudWatch 에이전트 설치 및 실행

원하는 에이전트 구성 파일을 생성하고 IAM 역할 또는 IAM 사용자를 생성했으면 다음 단계에 따라 해당 구성을 사용하여 서버에 에이전트를 설치하고 실행합니다. 먼저, 에이전트를 실행할 서버에 IAM 역할 또는 IAM 사용자를 연결합니다. 그런 다음 해당 서버에 에이전트 패키지를 다운로드한 후 만든 에이전트 구성을 사용하여 시작합니다.

S3 다운로드 링크를 사용하여 CloudWatch 에이전트 패키지 다운로드

> **Note**
>
> CloudWatch 에이전트를 다운로드하려면 연결에서 TLS 1.2 이상을 사용해야 합니다.

에이전트를 실행할 각 서버에 에이전트를 설치해야 합니다.

Amazon Linux AMI

CloudWatch 에이전트는 Amazon Linux 2023 및 Amazon Linux 2에서 패키지로 사용할 수 있습니다. 이 운영 체제를 사용하는 경우 다음 명령을 입력하여 패키지를 설치할 수 있습니다. 또한 인스턴스에 연결된 IAM 역할에 CloudWatchAgentServerPolicy가 연결되어 있는지 확인해야 합니다. 자세한 내용은 Amazon EC2 인스턴스에서 CloudWatch 에이전트와 함께 사용할 IAM 역할 생성 단원을 참조하세요.

```
sudo yum install amazon-cloudwatch-agent
```

모든 운영 체제

지원되는 모든 운영 체제에서 다음 단계에 설명된 대로 Amazon S3 다운로드 링크와 함께 명령줄을 사용하여 CloudWatch 에이전트를 다운로드하고 설치할 수 있습니다.

각 다운로드 링크에는 전체 링크뿐 아니라 리전별 링크도 있습니다. 예를 들어 Amazon Linux 2023, Amazon Linux 2 및 x86-64 아키텍처의 경우 다음과 같은 유효한 다운로드 링크 세 개가 있습니다.

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
x86-64	Amazon Linux 2023 및 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
		https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/amd64/la	https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/amd64/la

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
		test/amazon-cloudwatch-agent.rpm	test/amazon-cloudwatch-agent.rpm.sig
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg.sig
x86-64	Windows	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig
ARM64	Amazon Linux 2023 및 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig
ARM64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig

명령줄을 사용하여 Amazon EC2 인스턴스에 CloudWatch 에이전트를 설치하려면

1. CloudWatch 에이전트를 다운로드합니다. Linux 서버의 경우 다음을 입력합니다. *download-link*에 이전 테이블의 해당 다운로드 링크를 사용합니다.

```
wget download-link
```

Windows 서버를 실행하는 서버의 경우, 다음 파일을 다운로드합니다.

```
https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-
cloudwatch-agent.msi
```

2. 패키지를 다운로드했으면 선택 사항으로 패키지 서명을 확인할 수 있습니다. 자세한 내용은 CloudWatch 에이전트 패키지의 서명 확인 단원을 참조하세요.
3. 패키지를 설치합니다. Linux 서버에서 RPM 패키지를 다운로드한 경우 패키지가 있는 디렉터리로 변경하고 다음을 입력합니다.

```
sudo rpm -U ./amazon-cloudwatch-agent.rpm
```

Linux 서버에서 DEB 패키지를 다운로드한 경우 패키지가 있는 디렉터리로 변경하고 다음을 입력합니다.

```
sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
```

Windows Server가 실행되는 서버에서 MSI 패키지를 다운로드한 경우 패키지가 있는 디렉터리로 변경하고 다음을 입력합니다.

```
msiexec /i amazon-cloudwatch-agent.msi
```

또한 이 명령은 PowerShell 내에서도 작동합니다. MSI 명령 옵션에 대한 자세한 내용은 Microsoft Windows 문서의 명령줄 옵션 단원을 참조하세요.

(EC2 인스턴스에 설치) IAM 역할 연결

CloudWatch 에이전트가 인스턴스의 데이터를 보낼 수 있도록 하려면 IAM 역할을 인스턴스에 연결해야 합니다. 연결할 역할은 CloudWatchAgentServerRole입니다. 이 역할은 이전에 생성했어야 합니다. 자세한 정보는 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성 섹션을 참조하세요.

IAM 역할을 인스턴스에 연결하는 방법에 대한 자세한 내용은 Windows 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에 IAM 역할 연결 단원을 참조하세요.

(온프레미스 서버에 설치) IAM 자격 증명 및 AWS 리전 지정

CloudWatch 에이전트가 온프레미스 서버의 데이터를 보낼 수 있도록 하려면 이전에 생성한 IAM 사용자의 액세스 키 및 보안 키를 지정해야 합니다. 이 사용자 생성에 대한 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성 단원을 참조하세요.

또한 다음 예에 표시된 대로 AWS 구성 파일의 `[AmazonCloudWatchAgent]` 섹션에 있는 `region` 필드를 사용하여 지표를 보낼 AWS 리전을 지정해야 합니다.

```
[profile AmazonCloudWatchAgent]
region = us-west-1
```

다음은 `aws configure` 명령을 사용하여 CloudWatch 에이전트에 대한 명명된 프로파일을 생성하는 예입니다. 이 예에서는 `AmazonCloudWatchAgent`의 기본 프로필 이름을 사용하는 것으로 가정합니다.

CloudWatch 에이전트에 대한 AmazonCloudWatchAgent 프로파일을 생성하려면

1. 아직 설치하지 않았다면 서버에 AWS Command Line Interface를 설치합니다. 자세한 내용은 AWS CLI 설치 단원을 참조하세요.
2. Linux 서버에서 다음 명령을 입력하고 표시되는 메시지를 따릅니다.

   ```
   sudo aws configure --profile AmazonCloudWatchAgent
   ```

   Windows Server에서는 관리자 권한으로 PowerShell을 열고 다음 명령을 입력한 후 표시되는 메시지를 따릅니다.

   ```
   aws configure --profile AmazonCloudWatchAgent
   ```

인터넷 액세스 확인

CloudWatch 또는 CloudWatch Logs에 데이터를 전송하려면 Amazon EC2 인스턴스에 아웃바운드 인터넷 액세스 권한이 있어야 합니다. 인터넷 액세스를 구성하는 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터넷 게이트웨이 단원을 참조하세요.

프록시에서 구성할 엔드포인트와 포트는 다음과 같습니다.

- 에이전트를 사용하여 지표를 수집하는 경우 적절한 리전의 CloudWatch 엔드포인트를 허용 목록에 추가해야 합니다. 이러한 엔드포인트는 Amazon CloudWatch endpoints and quotas에 나열되어 있습니다.
- 에이전트를 사용하여 로그를 수집하는 경우 적절한 리전의 CloudWatch Logs 엔드포인트를 허용 목록에 추가해야 합니다. 이러한 엔드포인트는 Amazon CloudWatch Logs endpoints and quotas에 나열되어 있습니다.
- Systems Manager를 사용하여 에이전트를 설치하거나 파라미터 스토어를 사용하여 구성 파일을 저장하는 경우 적절한 리전의 Systems Manager 엔드포인트를 허용 목록에 추가해야 합니다. 이러한 엔드포인트는 AWS Systems Manager endpoints and quotas에 나열되어 있습니다.

(선택 사항) 프록시 또는 리전 정보에 대한 일반 구성 수정

CloudWatch 에이전트에는 `common-config.toml`이라는 구성 파일이 포함되어 있습니다. 필요한 경우 이 파일을 사용하여 프록시 및 리전 정보를 지정할 수 있습니다.

Linux를 실행하는 서버에서는 이 파일이 `/opt/aws/amazon-cloudwatch-agent/etc` 디렉터리에 있습니다. Windows Server를 실행하는 서버에서는 이 파일이 `C:\ProgramData\Amazon\AmazonCloudWatchAgent` 디렉터리에 있습니다.

> **Note**
>
> CloudWatch 에이전트를 온프레미스 모드에서 실행할 때 `common-config.toml` 파일을 사용하여 공유 구성 및 자격 증명을 제공하는 것이 좋습니다. 또한 Amazon EC2에서 실행 중이고 기존의 공유 자격 증명 프로필 및 파일을 재사용하려는 경우에도 이 방법이 유용할 수 있습니다. `common-config.toml`을 통해 활성화하면 공유 자격 증명 파일이 만료된 후 갱신된 자격 증명으로 교체되는 경우 에이전트가 다시 시작할 필요 없이 새 자격 증명을 자동으로 가져올 수 있다는 추가적인 이점이 있습니다.

기본 `common-config.toml`은 다음과 같습니다.

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for the on-premises case by
 default.
```

```
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

처음에는 모든 줄이 코멘트 아웃 처리되어 있습니다. 자격 증명 프로필이나 프록시 설정을 설정하려면 해당 줄에서 #을 제거하고 값을 지정하세요. 이 파일을 수동으로 편집하거나 다음과 같이 Systems Manager에서 `RunShellScript` Run Command를 사용하여 편집할 수 있습니다.

- `shared_credential_profile` – 온프레미스 서버의 경우 이 줄은 CloudWatch에 데이터를 전송하는 데 사용할 IAM 사용자 자격 증명 프로파일을 지정합니다. 이 줄을 코멘트 아웃 처리된 상태로 유지할 경우 `AmazonCloudWatchAgent`가 사용됩니다. 이 프로필 생성에 대한 자세한 내용은 (온프레미스 서버에 설치) IAM 자격 증명 및 AWS 리전 지정 단원을 참조하세요.

  EC2 인스턴스에서 이 줄을 사용하여 CloudWatch 에이전트가 이 인스턴스의 데이터를 다른 AWS 리전의 CloudWatch에 전송하도록 할 수 있습니다. 이렇게 하려면 보낼 리전의 이름을 지정하는 `region` 필드가 있는 명명된 프로필을 지정합니다.

  `shared_credential_profile`을 지정하는 경우 `[credentials]` 행의 시작 부분에서 #도 제거해야 합니다.

- `shared_credential_file` – 에이전트가 기본 경로 이외의 다른 경로에 있는 파일에서 자격 증명을 찾도록 하려면 여기에 전체 경로 및 파일 이름을 지정합니다. 기본 경로는 Linux의 경우 `/root/.aws`이며 Windows Server의 경우 `C:\\Users\\Administrator\\.aws`입니다.

  아래의 첫 번째 예는 Linux 서버의 유효한 `shared_credential_file` 행 구문이고 두 번째 예는 Windows Server에 유효한 행 구문입니다. Windows Server에서는 \ 문자를 이스케이프 처리해야 합니다.

  ```
  shared_credential_file= "/usr/username/credentials"
  ```

```
shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
```

shared_credential_file을 지정하는 경우 [credentials] 행의 시작 부분에서 #도 제거해야 합니다.

- 프록시 설정 - 서버가 HTTP 또는 HTTPS 프록시를 사용하여 AWS 서비스에 연결하는 경우 http_proxy 및 https_proxy 필드에 해당 프록시를 지정합니다. 프록시 설정에서 제외해야 하는 URL이 있다면 이를 쉼표로 구분하여 no_proxy 필드에 지정하세요.

명령줄을 사용하여 CloudWatch 에이전트 시작

다음 단계에 따라 명령줄을 사용하여 서버에서 CloudWatch 에이전트를 시작합니다.

명령줄을 사용하여 서버에서 CloudWatch 에이전트를 시작하려면

1. 사용하려는 에이전트 구성 파일을 에이전트를 실행할 서버에 복사합니다. 해당 파일을 복사할 경로 이름을 기록해 둡니다.
2. 이 명령에서 -a fetch-config는 에이전트가 최신 버전의 CloudWatch 에이전트 구성 파일을 로드하도록 하며 -s는 에이전트를 시작합니다.

   다음 명령 중 하나를 입력합니다. *configuration-file-path*를 에이전트 구성 파일의 경로로 바꿉니다. 이 파일을 마법사로 생성한 경우 config.json이라고 하며 수동으로 생성한 경우 amazon-cloudwatch-agent.json이라고 할 수 있습니다.

   Linux가 실행되는 EC2 인스턴스의 경우 다음 명령을 입력합니다.

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
   ```

   Linux가 실행되는 온프레미스 서버의 경우 다음을 입력합니다.

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m onPremise -s -c file:configuration-file-path
   ```

   Windows Server가 실행되는 EC2 인스턴스의 경우 PowerShell 콘솔에서 다음을 입력합니다.

   ```
   & "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -s -c file:configuration-file-path
   ```

Windows Server가 실행되는 온프레미스 서버의 경우 PowerShell 콘솔에서 다음을 입력합니다.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
a fetch-config -m onPremise -s -c file:configuration-file-path
```

AWS Systems Manager를 사용하여 CloudWatch 에이전트 설치

다음 주제를 참조하여 AWS Systems Manager를 사용해 CloudWatch 에이전트를 설치하고 실행할 수 있습니다.

주제

CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성

AWS 리소스에 액세스하려면 권한이 필요합니다. CloudWatch 에이전트가 CloudWatch에 지표를 작성하고 CloudWatch 에이전트가 Amazon EC2 및 AWS Systems Manager와 통신하도록 하는 데 필요한 권한이 포함된 IAM 역할 및 사용자를 생성할 수 있습니다. Amazon EC2 인스턴스에서는 IAM 역할을 사용하고 온프레미스 서버에서는 IAM 사용자를 사용합니다.

하나의 역할 또는 사용자를 사용하여 CloudWatch 에이전트를 서버에 설치하고 이 에이전트가 CloudWatch에 지표를 전송할 수 있도록 합니다. Systems Manager 파라미터 스토어에 CloudWatch 에이전트 구성을 저장하려면 다른 역할 또는 사용자가 필요합니다. 파라미터 스토어를 사용하면 여러 서버가 하나의 CloudWatch 에이전트 구성을 사용할 수 있습니다.

파라미터 스토어에 쓸 수 있는 기능은 광범위하고 강력한 권한이며, 필요 시에만 사용해야 하고, 배포 시 여러 인스턴스에 연결해선 안 됩니다. CloudWatch 에이전트 구성을 파라미터 스토어에 저장하는 경우 다음을 권장합니다.

- 이 구성을 수행할 단일 인스턴스를 설정합니다.
- 이 인스턴스에서만 파라미터 스토어에 쓸 수 있는 권한이 있는 IAM 역할을 사용합니다.
- CloudWatch 에이전트 구성 파일로 작업하고 이 파일을 저장하는 동안에만 파라미터 스토어에 쓸 수 있는 권한이 있는 IAM 역할을 사용합니다.

> **Note**
>
> 최근에 당사는 고객들에게 정책을 직접 생성하도록 요구하는 대신 Amazon에서 만든 새로운 `CloudWatchAgentServerPolicy` 및 `CloudWatchAgentAdminPolicy` 정책을 사용하여 다음과 같은 절차를 수정했습니다. 이러한 정책을 사용하여 에이전트 구성 파일을 파라미터 스토어에 작성한 다음, 파라미터 스토어에서 해당 파일을 다운로드하려면 에이전트 구성 파일의 이름이 `AmazonCloudWatch-`로 시작해야 합니다. 파일 이름이 `AmazonCloudWatch-`로 시작하지 않는 CloudWatch 에이전트 구성 파일이 있는 경우 이러한 정책을 사용하여 파라미터 스토어에 파일을 작성하거나 파라미터 스토어에서 파일을 다운로드할 수 없습니다.

Amazon EC2 인스턴스에서 CloudWatch 에이전트와 함께 사용할 IAM 역할 생성

첫 번째 절차에서는 CloudWatch 에이전트를 실행하는 각 Amazon EC2 인스턴스에 연결해야 하는 IAM 역할을 생성합니다. 이 역할은 정보를 인스턴스에서 읽고 CloudWatch에 쓸 수 있는 권한을 제공합니다.

두 번째 절차에서는 CloudWatch 에이전트 구성 파일을 생성하는 데 사용되는 Amazon EC2 인스턴스에 연결해야 하는 IAM 역할을 생성합니다. 다른 서버에서 사용할 수 있도록 이 파일을 Systems Manager 파라미터 스토어에 저장하려는 경우 이 단계가 필요합니다. 이 역할은 정보를 인스턴스에서 읽고 CloudWatch에 쓸 수 있는 권한 외에도 파라미터 스토어에 쓸 수 있는 권한도 제공합니다. 이 역할에는 CloudWatch 에이전트를 실행하는 것은 물론 파라미터 스토어에 쓰는 데 충분한 권한이 포함됩니다.

> **Note**
>
> 파라미터 스토어는 표준 및 고급 티어의 파라미터를 지원합니다. 이러한 파라미터 티어는 CloudWatch 에이전트의 미리 정의된 지표 세트에서 사용할 수 있는 기본, 표준 및 고급 세부 정보 수준과 관련이 없습니다.

각 서버가 CloudWatch 에이전트를 실행하는 데 필요한 IAM 역할을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔티티 유형 선택 아래에서 AWS 서비스를 선택합니다.

4. [일반 사용 사례(Common use cases)] 바로 아래에서 [EC2]를 선택한 후 [다음: 권한(Next: Permissions)]을 선택합니다.
5. 정책 목록에서 CloudWatchAgentServerPolicy 옆의 확인란을 선택합니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다.
6. Systems Manager를 사용하여 CloudWatch 에이전트를 설치하거나 구성하려면 AmazonSSMManagedInstanceCore 옆의 확인란을 선택합니다. 이 AWS 관리형 정책을 사용하면 인스턴스가 Systems Manager 서비스 핵심 기능을 사용할 수 있습니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다. 명령줄을 통해서만 에이전트를 시작하고 구성하는 경우에는 이 정책이 필요하지 않습니다.
7. 다음: 태그를 선택합니다.
8. (선택 사항) 이 역할에 대한 액세스를 구성, 추적 또는 제어할 태그-키 값 페어를 하나 이상 추가한 후 다음: 검토(Next: Review)를 선택합니다.
9. 역할 이름에 새 역할의 이름(예: **`CloudWatchAgentServerRole`** 또는 자신이 선호하는 다른 이름)을 입력합니다.
10. (선택 사항)역할 설명에 설명을 입력합니다.
11. CloudWatchAgentServerPolicy 및 AmazonSSMManagedInstanceCore이 정책 옆에 표시되는지 확인하세요.
12. 역할 생성을 선택합니다.

    이제 역할이 생성되었습니다.

다음 절차에서는 파라미터 스토어에 대한 쓰기 권한도 있는 IAM 역할을 생성합니다. 이 역할을 사용하여 에이전트 구성 파일을 파라미터 스토어에 저장하면 다른 서버가 이를 검색할 수 있습니다.

파라미터 스토어에 대한 쓰기 권한은 광범위한 액세스를 제공하므로 이 역할은 일부 서버에만 연결해야 하며, 관리자만 사용해야 합니다. 에이전트 구성 파일을 생성하여 파라미터 스토어에 복사한 후에는 이 역할을 인스턴스에서 분리하고 대신 `CloudWatchAgentServerRole`을 사용해야 합니다.

관리자가 파라미터 스토어에 쓸 수 있도록 IAM 역할을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔티티 유형 선택 아래에서 AWS 서비스를 선택합니다.
4. 이 역할을 사용할 서비스 선택에서 EC2와 다음: 권한을 차례대로 선택합니다.

5. 정책 목록에서 CloudWatchAgentAdminPolicy 옆의 확인란을 선택합니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다.
6. Systems Manager를 사용하여 CloudWatch 에이전트를 설치하거나 구성하려면 AmazonSSMManagedInstanceCore 옆의 확인란을 선택합니다. 이 AWS 관리형 정책을 사용하면 인스턴스가 Systems Manager 서비스 핵심 기능을 사용할 수 있습니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다. 명령줄을 통해서만 에이전트를 시작하고 구성하는 경우에는 이 정책이 필요하지 않습니다.
7. 다음: 태그를 선택합니다.
8. (선택 사항) 이 역할에 대한 액세스를 구성, 추적 또는 제어할 태그-키 값 페어를 하나 이상 추가한 후 다음: 검토(Next: Review)를 선택합니다.
9. 역할 이름에 새 역할의 이름(예: **`CloudWatchAgentAdminRole`** 또는 자신이 선호하는 다른 이름)을 입력합니다.
10. (선택 사항)역할 설명에 설명을 입력합니다.
11. CloudWatchAgentAdminPolicy 및 AmazonSSMManagedInstanceCore(선택 사항)이 정책 옆에 표시되는지 확인하세요.
12. 역할 생성을 선택합니다.

    이제 역할이 생성되었습니다.

온프레미스 서버에서 CloudWatch 에이전트와 함께 사용할 IAM 사용자 생성

첫 번째 절차에서는 CloudWatch 에이전트를 실행하는 데 필요한 IAM 사용자를 생성합니다. 이 사용자는 CloudWatch에 데이터를 전송할 수 있는 권한을 제공합니다.

두 번째 절차에서는 CloudWatch 에이전트 구성 파일을 생성할 때 사용할 수 있는 IAM 사용자를 생성합니다. 다른 서버에서 사용할 수 있도록 이 파일을 Systems Manager 파라미터 스토어에 저장하려면 이 절차를 사용합니다. 이 사용자는 CloudWatch에 데이터를 쓸 수 있는 권한 외에도 파라미터 스토어에 쓸 수 있는 권한도 제공합니다.

> **Note**
>
> 파라미터 스토어는 표준 및 고급 티어의 파라미터를 지원합니다. 이러한 파라미터 티어는 CloudWatch 에이전트의 미리 정의된 지표 세트에서 사용할 수 있는 기본, 표준 및 고급 세부 정보 수준과 관련이 없습니다.

CloudWatch 에이전트가 CloudWatch에 데이터를 쓰는 데 필요한 IAM 사용자를 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Users와 Add user를 차례대로 선택합니다.
3. 새 사용자의 사용자 이름을 입력합니다.
4. Access type(액세스 유형)에서 Programmatic access(프로그래밍 방식 액세스)를 선택한 다음 Next: Permissions(다음: 권한)를 선택합니다.
5. Set permissions(권한 설정)에서 Attach existing policies directly(기존 정책을 직접 연결)를 선택합니다.
6. 정책 목록에서 CloudWatchAgentServerPolicy 옆의 확인란을 선택합니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다.
7. Systems Manager를 사용하여 CloudWatch 에이전트를 설치하거나 구성하려면 AmazonSSMManagedInstanceCore 옆의 확인란을 선택합니다. 이 AWS 관리형 정책을 사용하면 인스턴스가 Systems Manager 서비스 핵심 기능을 사용할 수 있습니다. (필요한 경우 검색 상자를 사용하여 정책을 찾습니다. 명령줄을 통해서만 에이전트를 시작하고 구성하는 경우에는 이 정책이 필요하지 않습니다.)
8. 다음: 태그를 선택합니다.
9. (선택 사항) 이 역할에 대한 액세스를 구성, 추적 또는 제어할 태그-키 값 페어를 하나 이상 추가한 후 다음: 검토(Next: Review)를 선택합니다.
10. 올바른 정책이 나열되는지 확인하고 사용자 생성을 선택합니다.
11. 새 사용자를 위한 행에서 표시를 선택합니다. 에이전트를 설치할 때 사용할 수 있도록 액세스 키 및 보안 키를 파일에 복사한 다음, 닫기를 선택하세요.

다음 절차에서는 파라미터 스토어에 대한 쓰기 권한도 있는 IAM 사용자를 생성합니다. 다른 서버에서 사용할 수 있도록 에이전트 구성 파일을 파라미터 스토어에 저장하려면 이 IAM 사용자를 사용해야 합니다. 이 IAM 사용자는 파라미터 스토어에 대한 쓰기 권한을 제공합니다. 또한 이 사용자는 정보를 인스턴스에서 읽고 CloudWatch에 쓸 수 있는 권한도 제공합니다. Systems Manager 파라미터 스토어에 대한 쓰기 권한은 광범위한 액세스를 제공하므로 이 IAM 사용자는 일부 서버에만 연결해야 하며, 관리자만 사용해야 합니다. 에이전트 구성 파일을 파라미터 스토어에 저장하는 경우에만 이 IAM 사용자를 사용해야 합니다.

파라미터 스토어에 구성 파일을 저장하고 CloudWatch에 정보를 전송하는 데 필요한 IAM 사용자를 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Users와 Add user를 차례대로 선택합니다.
3. 새 사용자의 사용자 이름을 입력합니다.
4. Access type(액세스 유형)에서 Programmatic access(프로그래밍 방식 액세스)를 선택한 다음 Next: Permissions(다음: 권한)를 선택합니다.
5. Set permissions(권한 설정)에서 Attach existing policies directly(기존 정책을 직접 연결)를 선택합니다.
6. 정책 목록에서 CloudWatchAgentAdminPolicy 옆의 확인란을 선택합니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다.
7. Systems Manager를 사용하여 CloudWatch 에이전트를 설치하거나 구성하려면 AmazonSSMManagedInstanceCore 옆의 확인란을 선택합니다. 이 AWS 관리형 정책을 사용하면 인스턴스가 Systems Manager 서비스 핵심 기능을 사용할 수 있습니다. (필요한 경우 검색 상자를 사용하여 정책을 찾습니다. 명령줄을 통해서만 에이전트를 시작하고 구성하는 경우에는 이 정책이 필요하지 않습니다.)
8. 다음: 태그를 선택합니다.
9. (선택 사항) 이 역할에 대한 액세스를 구성, 추적 또는 제어할 태그-키 값 페어를 하나 이상 추가한 후 다음: 검토(Next: Review)를 선택합니다.
10. 올바른 정책이 나열되는지 확인하고 사용자 생성을 선택합니다.
11. 새 사용자를 위한 행에서 표시를 선택합니다. 에이전트를 설치할 때 사용할 수 있도록 액세스 키 및 보안 키를 파일에 복사한 다음, 닫기를 선택하세요.

CloudWatch 에이전트 다운로드 및 구성

이 단원에서는 Systems Manager를 사용하여 에이전트를 다운로드하는 방법과 에이전트 구성 파일을 생성하는 방법을 설명합니다. Systems Manager를 사용하여 에이전트를 다운로드하려면 먼저, 인스턴스가 Systems Manager에 대해 올바르게 구성되어 있는지 확인해야 합니다.

SSM Agent 설치 또는 업데이트

Amazon EC2 인스턴스에서 CloudWatch 에이전트를 사용하려면 인스턴스가 2.2.93.0 이상 버전을 실행하고 있어야 합니다. CloudWatch 에이전트를 설치하기 전에 인스턴스에서 SSM Agent를 업데이트하거나 설치합니다(아직 하지 않은 경우).

Linux를 실행하는 인스턴스에 SSM Agent를 설치하거나 업데이트하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 Linux 인스턴스에 SSM Agent 설치 및 구성 단원을 참조하세요.

SSM Agent 설치 또는 업데이트에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 SSM Agent 작업 단원을 참조하세요.

(선택 사항) Systems Manager 사전 조건 확인

인터넷 액세스 확인

CloudWatch 또는 CloudWatch Logs에 데이터를 전송하려면 Amazon EC2 인스턴스에 아웃바운드 인터넷 액세스 권한이 있어야 합니다. 인터넷 액세스를 구성하는 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터넷 게이트웨이 단원을 참조하세요.

프록시에서 구성할 엔드포인트와 포트는 다음과 같습니다.

- 에이전트를 사용하여 지표를 수집하는 경우 적절한 리전의 CloudWatch 엔드포인트를 허용 목록에 추가해야 합니다. 이러한 엔드포인트는 Amazon Web Services 일반 참조의 Amazon CloudWatch에 나열되어 있습니다.
- 에이전트를 사용하여 로그를 수집하는 경우 적절한 리전의 CloudWatch Logs 엔드포인트를 허용 목록에 추가해야 합니다. 이러한 엔드포인트는 Amazon Web Services 일반 참조의 Amazon CloudWatch Logs에 나열되어 있습니다.
- Systems Manager를 사용하여 에이전트를 설치하거나 파라미터 스토어를 사용하여 구성 파일을 저장하는 경우 적절한 리전의 Systems Manager 엔드포인트를 허용 목록에 추가해야 합니다. 이러한 엔드포인트는 Amazon Web Services 일반 참조의 AWS Systems Manager에 에 나열되어 있습니다.

다음 단계에 따라 Systems Manager를 사용하여 CloudWatch 에이전트 패키지를 다운로드합니다.

Systems Manager를 사용하여 CloudWatch 에이전트 다운로드

1. https://console.aws.amazon.com/systems-manager/에서 시스템 관리자 콘솔을 엽니다.
2. 탐색 창에서 Run Command를 선택합니다.

   -또는-

   AWS Systems Manager 홈페이지가 열리면 아래로 스크롤하여 [Run Command 탐색(Explore Run Command)]을 선택합니다.
3. Run command(Run 명령)를 선택합니다.

4. 명령 문서 목록에서 AWS-ConfigureAWSPackage를 선택합니다.
5. [대상(Targets)] 영역에서 CloudWatch 에이전트를 설치할 인스턴스를 선택합니다. 특정 인스턴스가 표시되지 않으면 Systems Manager에서 사용할 관리형 인스턴스로 구성되지 않은 것일 수 있습니다. 자세한 내용은 AWS Systems Manager 사용 설명서에서 하이브리드 환경의 AWS Systems Manager 설정 단원을 참조하세요.
6. 작업 목록에서 설치를 선택합니다.
7. 이름 필드에 *AmazonCloudWatchAgent*를 입력합니다.
8. 최신 버전의 에이전트를 설치하도록 버전을 최신 상태로 설정한 채로 유지합니다.
9. Run(실행)을 선택합니다.
10. (선택 사항) [대상 및 출력(Targets and outputs)] 영역에서 인스턴스 이름 옆의 버튼을 선택하고 [출력 보기(View output)]를 선택합니다. Systems Manager에 에이전트가 성공적으로 설치되었다고 표시되어야 합니다.

에이전트 구성 파일 생성 및 수정

CloudWatch 에이전트를 다운로드했으면 서버에서 에이전트를 시작하기 전에 구성 파일을 생성해야 합니다.

에이전트 구성 파일을 Systems Manager 파라미터 스토어에 저장하려는 경우 EC2 인스턴스를 사용하여 파라미터 스토어에 저장해야 합니다. 또한 먼저, 해당 인스턴스에 `CloudWatchAgentAdminRole` IAM 역할을 연결해야 합니다. 역할 연결에 대한 자세한 내용은 Windows 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에 IAM 역할 연결 단원을 참조하세요.

CloudWatch 에이전트 구성 파일 생성에 대한 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.

에이전트 구성을 사용하여 EC2 인스턴스에 CloudWatch 에이전트 설치

CloudWatch 에이전트 구성을 파라미터 스토어에 저장했으면 다른 서버에 에이전트를 설치할 때 이를 사용할 수 있습니다.

주제

- 인스턴스에 IAM 역할 연결
- Amazon EC2 인스턴스에 CloudWatch 에이전트 패키지 다운로드
- (선택 사항) CloudWatch 에이전트에 대한 일반 구성 및 명명된 프로파일 수정
- CloudWatch 에이전트 시작

인스턴스에 IAM 역할 연결

EC2 인스턴스에 CloudWatchAgentServerRole IAM 역할을 연결해야 인스턴스에서 CloudWatch 에이전트를 실행할 수 있습니다. 이 역할을 통해 CloudWatch 에이전트는 인스턴스에서 작업을 수행할 수 있습니다. 이 역할은 이전에 생성했어야 합니다. 자세한 정보는 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성 섹션을 참조하세요.

자세한 내용은 Windows 인스턴스용 Amazon EC2 사용 설명서의 인스턴스에 IAM 역할 연결 단원을 참조하세요.

Amazon EC2 인스턴스에 CloudWatch 에이전트 패키지 다운로드

에이전트를 실행할 각 서버에 에이전트를 설치해야 합니다. CloudWatch 에이전트는 Amazon Linux 2023 및 Amazon Linux 2에서 패키지로 사용할 수 있습니다. 이 운영 체제를 사용하는 경우 다음 명령을 입력하여 패키지를 설치할 수 있습니다. 또한 인스턴스에 연결된 IAM 역할에 CloudWatchAgentServerPolicy가 연결되어 있는지 확인해야 합니다. 자세한 내용은 Amazon EC2 인스턴스에서 CloudWatch 에이전트와 함께 사용할 IAM 역할 생성 단원을 참조하세요.

```
sudo yum install amazon-cloudwatch-agent
```

지원되는 모든 운영 체제에서 Systems Manager Run Command 또는 Amazon S3 다운로드 링크를 사용하여 CloudWatch 에이전트 패키지를 다운로드할 수 있습니다. Amazon S3 다운로드 링크 사용에 대한 자세한 내용은 CloudWatch 에이전트 패키지 다운로드 단원을 참조하세요.

> **Note**
>
> CloudWatch 에이전트를 설치하거나 업데이트하는 경우 제거 및 재설치(Uninstall and reinstall) 옵션만 지원됩니다. 현재 위치 업데이트(In-place update) 옵션을 사용할 수 없습니다.

Systems Manager를 사용하여 Amazon EC2 인스턴스에 CloudWatch 에이전트 다운로드

Systems Manager를 사용하여 CloudWatch 에이전트를 설치하려면 먼저, 인스턴스가 Systems Manager에 대해 올바르게 구성되어 있는지 확인해야 합니다.

SSM Agent 설치 또는 업데이트

Amazon EC2 인스턴스에서 CloudWatch 에이전트를 사용하려면 인스턴스가 2.2.93.0 이상 버전을 실행하고 있어야 합니다. CloudWatch 에이전트를 설치하기 전에 인스턴스에서 SSM Agent를 업데이트하거나 설치합니다(아직 하지 않은 경우).

Linux를 실행하는 인스턴스에 SSM Agent를 설치하거나 업데이트하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 Linux 인스턴스에 SSM Agent 설치 및 구성 단원을 참조하세요.

Windows Server를 실행하는 인스턴스에 SSM Agent를 설치하거나 업데이트하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 Windows 인스턴스에 SSM Agent 설치 및 구성 단원을 참조하세요.

(선택 사항) Systems Manager 사전 조건 확인

Systems Manager Run Command를 사용하여 CloudWatch 에이전트를 설치 및 구성하기 전에 인스턴스가 Systems Manager 최소 요구 사항을 충족하는지 확인하세요. 자세한 내용은 AWS Systems Manager 사용 설명서에서 AWS Systems Manager 설정을 참조하세요.

인터넷 액세스 확인

CloudWatch 또는 CloudWatch Logs에 데이터를 전송하려면 Amazon EC2 인스턴스에 아웃바운드 인터넷 액세스 권한이 있어야 합니다. 인터넷 액세스를 구성하는 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터넷 게이트웨이 단원을 참조하세요.

CloudWatch 에이전트 패키지 다운로드

Systems Manager Run Command를 사용하면 인스턴스의 구성을 관리할 수 있습니다. Systems Manager 문서 및 파라미터를 지정하고 하나 이상의 인스턴스에서 명령을 실행할 수 있습니다. 인스턴스의 SSM Agent는 명령을 처리하고 지정된 대로 인스턴스를 구성합니다.

Run Command를 사용하여 CloudWatch 에이전트를 다운로드하려면

1. https://console.aws.amazon.com/systems-manager/에서 시스템 관리자 콘솔을 엽니다.
2. 탐색 창에서 Run Command를 선택합니다.

   -또는-

   AWS Systems Manager 홈페이지가 열리면 아래로 스크롤하여 [Run Command 탐색(Explore Run Command)]을 선택합니다.
3. Run command(Run 명령)를 선택합니다.
4. 명령 문서 목록에서 AWS-ConfigureAWSPackage를 선택합니다.
5. [대상(Targets)] 영역에서 CloudWatch 에이전트를 설치할 인스턴스를 선택합니다. 특정 인스턴스가 표시되지 않으면 Run Command에 대해 구성되지 않은 것일 수 있습니다. 자세한 내용은 AWS

Systems Manager 사용 설명서에서 하이브리드 환경의 AWS Systems Manager 설정 단원을 참조하세요.

6. 작업 목록에서 설치를 선택합니다.
7. 이름 상자에 *AmazonCloudWatchAgent*를 입력합니다.
8. 최신 버전의 에이전트를 설치하도록 버전을 최신 상태로 설정한 채로 유지합니다.
9. Run(실행)을 선택합니다.
10. (선택 사항) [대상 및 출력(Targets and outputs)] 영역에서 인스턴스 이름 옆의 버튼을 선택하고 [출력 보기(View output)]를 선택합니다. Systems Manager에 에이전트가 성공적으로 설치되었다고 표시되어야 합니다.

(선택 사항) CloudWatch 에이전트에 대한 일반 구성 및 명명된 프로파일 수정

CloudWatch 에이전트에는 `common-config.toml`이라는 구성 파일이 포함되어 있습니다. 필요한 경우 이 파일을 사용하여 프록시 및 리전 정보를 지정할 수 있습니다.

Linux를 실행하는 서버에서는 이 파일이 `/opt/aws/amazon-cloudwatch-agent/etc` 디렉터리에 있습니다. Windows Server를 실행하는 서버에서는 이 파일이 `C:\ProgramData\Amazon\AmazonCloudWatchAgent` 디렉터리에 있습니다.

기본 `common-config.toml`은 다음과 같습니다.

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
```

```
#    no_proxy = "{domain}"
```

처음에는 모든 줄이 코멘트 아웃 처리되어 있습니다. 자격 증명 프로필이나 프록시 설정을 설정하려면 해당 줄에서 #을 제거하고 값을 지정하세요. 이 파일을 수동으로 편집하거나 다음과 같이 Systems Manager에서 RunShellScript Run Command를 사용하여 편집할 수 있습니다.

- shared_credential_profile – 온프레미스 서버의 경우 이 줄은 CloudWatch에 데이터를 전송하는 데 사용할 IAM 사용자 자격 증명 프로파일을 지정합니다. 이 줄을 코멘트 아웃 처리된 상태로 유지할 경우 AmazonCloudWatchAgent가 사용됩니다.

  EC2 인스턴스에서 이 줄을 사용하여 CloudWatch 에이전트가 이 인스턴스의 데이터를 다른 AWS 리전의 CloudWatch에 전송하도록 할 수 있습니다. 이렇게 하려면 보낼 리전의 이름을 지정하는 region 필드가 있는 명명된 프로필을 지정합니다.

  shared_credential_profile을 지정하는 경우 [credentials] 행의 시작 부분에서 #도 제거해야 합니다.

- shared_credential_file – 에이전트가 기본 경로 이외의 다른 경로에 있는 파일에서 자격 증명을 찾도록 하려면 여기에 전체 경로 및 파일 이름을 지정합니다. 기본 경로는 Linux의 경우 /root/.aws이며 Windows Server의 경우 C:\\Users\\Administrator\\.aws입니다.

  아래의 첫 번째 예는 Linux 서버의 유효한 shared_credential_file 행 구문이고 두 번째 예는 Windows Server에 유효한 행 구문입니다. Windows Server에서는 \ 문자를 이스케이프 처리해야 합니다.

  ```
  shared_credential_file= "/usr/username/credentials"
  ```

  ```
  shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
  ```

  shared_credential_file을 지정하는 경우 [credentials] 행의 시작 부분에서 #도 제거해야 합니다.

- 프록시 설정 - 서버가 HTTP 또는 HTTPS 프록시를 사용하여 AWS 서비스에 연결하는 경우 http_proxy 및 https_proxy 필드에 해당 프록시를 지정합니다. 프록시 설정에서 제외해야 하는 URL이 있다면 이를 쉼표로 구분하여 no_proxy 필드에 지정하세요.

CloudWatch 에이전트 시작

Systems Manager Run Command 또는 명령줄을 사용하여 에이전트를 시작할 수 있습니다.

Systems Manager Run Command를 사용하여 CloudWatch 에이전트 시작

다음 단계에 따라 Systems Manager Run Command를 사용하여 에이전트를 시작할 수 있습니다.

Run Command를 사용하여 CloudWatch 에이전트를 시작하려면

1. https://console.aws.amazon.com/systems-manager/에서 시스템 관리자 콘솔을 엽니다.
2. 탐색 창에서 Run Command를 선택합니다.

   -또는-

   AWS Systems Manager 홈페이지가 열리면 아래로 스크롤하여 [Run Command 탐색(Explore Run Command)]을 선택합니다.
3. Run command(Run 명령)를 선택합니다.
4. 명령 문서 목록에서 AmazonCloudWatch-ManageAgent를 선택합니다.
5. [대상(Targets)] 영역에서 CloudWatch 에이전트를 설치한 인스턴스를 선택합니다.
6. 작업 목록에서 구성을 선택합니다.
7. Optional Configuration Source(구성 소스(선택 사항)) 목록에서 ssm을 선택합니다.
8. CloudWatch 에이전트 구성 파일 생성에 설명된 대로 선택적 구성 위치 상자에서 Systems Manager Parameter Store에 생성 및 저장한 에이전트 구성 파일의 Systems Manager 매개변수 이름을 입력합니다.
9. Optional Restart(재시작(선택 사항)) 목록에서 예를 선택하여 해당 단계를 마친 후 에이전트가 시작되도록 합니다.
10. Run(실행)을 선택합니다.
11. (선택 사항) [대상 및 출력(Targets and outputs)] 영역에서 인스턴스 이름 옆의 버튼을 선택하고 [출력 보기(View output)]를 선택합니다. Systems Manager에 에이전트가 성공적으로 시작되었다고 표시되어야 합니다.

명령줄을 사용하여 Amazon EC2 인스턴스에서 CloudWatch 에이전트 시작

다음 단계에 따라 명령줄을 사용하여 Amazon EC2 인스턴스에 CloudWatch 에이전트를 설치할 수 있습니다.

명령줄을 사용하여 Amazon EC2 인스턴스에서 CloudWatch 에이전트를 시작하려면

- 이 명령에서 `-a fetch-config`는 에이전트가 최신 버전의 CloudWatch 에이전트 구성 파일을 로드하도록 하며 `-s`는 에이전트를 시작합니다.

Linux 및 macOS: Systems Manager 파라미터 스토어에 구성 파일을 저장한 경우 다음을 입력합니다.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c ssm:configuration-parameter-store-name
```

Linux 및 macOS: 로컬 컴퓨터에 구성 파일을 저장한 경우 다음 명령을 입력합니다. *configuration-file-path*를 에이전트 구성 파일의 경로로 바꿉니다. 이 파일을 마법사로 생성한 경우 `config.json`이라고 하며 수동으로 생성한 경우 `amazon-cloudwatch-agent.json`이라고 할 수 있습니다.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
```

Windows Server: Systems Manager 파라미터 스토어에 에이전트 구성 파일을 저장한 경우 PowerShell 콘솔에서 다음을 입력합니다.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -s -c ssm:configuration-parameter-store-name
```

Windows Server: 에이전트 구성 파일을 로컬 컴퓨터에 저장한 경우 PowerShell 콘솔에서 다음을 입력합니다.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a fetch-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
\config.json"
```

온프레미스 서버에 CloudWatch 에이전트 설치

한 컴퓨터에 CloudWatch 에이전트를 다운로드하고 원하는 에이전트 구성 파일을 생성한 경우 해당 구성 파일을 사용하여 다른 온프레미스 서버에 에이전트를 설치할 수 있습니다.

온프레미스 서버에 CloudWatch 에이전트 다운로드

Systems Manager Run Command 또는 Amazon S3 다운로드 링크를 사용하여 CloudWatch 에이전트 패키지를 다운로드할 수 있습니다. Amazon S3 다운로드 링크 사용에 대한 자세한 내용은 CloudWatch 에이전트 패키지 다운로드 단원을 참조하세요.

Systems Manager를 사용하여 다운로드

Systems Manager Run Command를 사용하려면 Amazon EC2 Systems Manager에 온프레미스 서버를 등록해야 합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 하이브리드 환경에서 Systems Manager 설정 단원을 참조하세요.

서버를 이미 등록한 경우 SSM Agent를 최신 버전으로 업데이트합니다.

Linux를 실행하는 서버에서 SSM Agent를 업데이트하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서에서 하이브리드 환경(Linux)의 SSM Agent 설치 단원을 참조하세요.

Windows Server를 실행하는 서버에서 SSM Agent를 업데이트하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서에서 하이브리드 환경(Windows)의 SSM Agent 설치 단원을 참조하세요.

SSM Agent를 사용하여 온프레미스 서버에 CloudWatch 에이전트 패키지를 다운로드하려면

1. https://console.aws.amazon.com/systems-manager/에서 시스템 관리자 콘솔을 엽니다.
2. 탐색 창에서 Run Command를 선택합니다.

   -또는-

   AWS Systems Manager 홈페이지가 열리면 아래로 스크롤하여 [Run Command 탐색(Explore Run Command)]을 선택합니다.
3. Run command(Run 명령)를 선택합니다.
4. 명령 문서 목록에서 AWS-ConfigureAWSPackage 옆의 버튼을 선택합니다.
5. [대상(Targets)] 영역에서 CloudWatch 에이전트를 설치할 서버를 선택합니다. 특정 서버가 표시되지 않으면 Run Command에 대해 구성되지 않은 것일 수 있습니다. 자세한 내용은 AWS Systems Manager 사용 설명서에서 하이브리드 환경의 AWS Systems Manager 설정 단원을 참조하세요.
6. 작업 목록에서 설치를 선택합니다.
7. 이름 상자에 *AmazonCloudWatchAgent*를 입력합니다.
8. 최신 버전의 에이전트를 설치하도록 버전을 비워 둡니다.
9. Run(실행)을 선택합니다.

   에이전트 패키지가 다운로드되며 다음 단계는 이를 구성하고 시작하는 것입니다.

(온프레미스 서버에 설치) IAM 자격 증명 및 AWS 리전 지정

CloudWatch 에이전트가 온프레미스 서버의 데이터를 보낼 수 있도록 하려면 이전에 생성한 IAM 사용자의 액세스 키 및 보안 키를 지정해야 합니다. 이 사용자 생성에 대한 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성 단원을 참조하세요.

또한 `region` 필드를 사용하여 지표를 보낼 AWS 리전을 지정해야 합니다.

다음은 이 파일의 예입니다.

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
region = us-west-1
```

my_access_key 및 *my_secret_key*의 경우 Systems Manager 파라미터 스토어에 대한 쓰기 권한이 없는 IAM 사용자의 키를 사용합니다. CloudWatch 에이전트에 필요한 IAM 사용자에 대한 자세한 내용은 온프레미스 서버에서 CloudWatch 에이전트와 함께 사용할 IAM 사용자 생성 단원을 참조하세요.

이 프로필의 이름을 `AmazonCloudWatchAgent`로 지정하면 아무 작업도 수행할 필요가 없습니다. 다음 단원에서 설명된 대로 필요한 경우 다른 이름을 지정하고 해당 이름을 `common-config.toml` 파일에서 `shared_credential_profile`에 대한 값으로 지정할 수 있습니다.

다음은 aws configure 명령을 사용하여 CloudWatch 에이전트에 대한 명명된 프로파일을 생성하는 예입니다. 이 예에서는 `AmazonCloudWatchAgent`의 기본 프로필 이름을 사용하는 것으로 가정합니다.

CloudWatch 에이전트에 대한 AmazonCloudWatchAgent 프로파일을 생성하려면

1. 아직 설치하지 않았다면 서버에 AWS Command Line Interface를 설치합니다. 자세한 내용은 AWS CLI 설치 단원을 참조하세요.
2. Linux 서버에서 다음 명령을 입력하고 표시되는 메시지를 따릅니다.

   ```
   sudo aws configure --profile AmazonCloudWatchAgent
   ```

   Windows Server에서는 관리자 권한으로 PowerShell을 열고 다음 명령을 입력한 후 표시되는 메시지를 따릅니다.

   ```
   aws configure --profile AmazonCloudWatchAgent
   ```

(선택 사항) CloudWatch 에이전트에 대한 일반 구성 및 명명된 프로파일 수정

CloudWatch 에이전트에는 common-config.toml이라는 구성 파일이 포함되어 있습니다. 필요한 경우 이 파일을 사용하여 프록시 및 리전 정보를 지정할 수 있습니다.

Linux를 실행하는 서버에서는 이 파일이 /opt/aws/amazon-cloudwatch-agent/etc 디렉터리에 있습니다. Windows Server를 실행하는 서버에서는 이 파일이 C:\ProgramData\Amazon\AmazonCloudWatchAgent 디렉터리에 있습니다.

기본 common-config.toml은 다음과 같습니다.

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

처음에는 모든 줄이 코멘트 아웃 처리되어 있습니다. 자격 증명 프로필이나 프록시 설정을 설정하려면 해당 줄에서 #을 제거하고 값을 지정하세요. 이 파일을 수동으로 편집하거나 다음과 같이 Systems Manager에서 RunShellScript Run Command를 사용하여 편집할 수 있습니다.

- shared_credential_profile – 온프레미스 서버의 경우 이 줄은 CloudWatch에 데이터를 전송하는 데 사용할 IAM 사용자 자격 증명 프로파일을 지정합니다. 이 줄을 코멘트 아웃 처리된 상태로 유지할 경우 AmazonCloudWatchAgent가 사용됩니다. 이 프로필 생성에 대한 자세한 내용은 (온프레미스 서버에 설치) IAM 자격 증명 및 AWS 리전 지정 단원을 참조하세요.

EC2 인스턴스에서 이 줄을 사용하여 CloudWatch 에이전트가 이 인스턴스의 데이터를 다른 AWS 리전의 CloudWatch에 전송하도록 할 수 있습니다. 이렇게 하려면 보낼 리전의 이름을 지정하는 `region` 필드가 있는 명명된 프로필을 지정합니다.

`shared_credential_profile`을 지정하는 경우 `[credentials]` 행의 시작 부분에서 #도 제거해야 합니다.

- `shared_credential_file` – 에이전트가 기본 경로 이외의 다른 경로에 있는 파일에서 자격 증명을 찾도록 하려면 여기에 전체 경로 및 파일 이름을 지정합니다. 기본 경로는 Linux의 경우 `/root/.aws`이며 Windows Server의 경우 `C:\\Users\\Administrator\\.aws`입니다.

  아래의 첫 번째 예는 Linux 서버의 유효한 `shared_credential_file` 행 구문이고 두 번째 예는 Windows Server에 유효한 행 구문입니다. Windows Server에서는 \ 문자를 이스케이프 처리해야 합니다.

  ```
  shared_credential_file= "/usr/username/credentials"
  ```

  ```
  shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
  ```

  `shared_credential_file`을 지정하는 경우 `[credentials]` 행의 시작 부분에서 #도 제거해야 합니다.
- 프록시 설정 - 서버가 HTTP 또는 HTTPS 프록시를 사용하여 AWS 서비스에 연결하는 경우 `http_proxy` 및 `https_proxy` 필드에 해당 프록시를 지정합니다. 프록시 설정에서 제외해야 하는 URL이 있다면 이를 쉼표로 구분하여 `no_proxy` 필드에 지정하세요.

CloudWatch 에이전트 시작

Systems Manager Run Command 또는 명령줄을 사용하여 CloudWatch 에이전트를 시작할 수 있습니다.

SSM Agent를 사용하여 온프레미스 서버에서 CloudWatch 에이전트를 시작하려면

1. https://console.aws.amazon.com/systems-manager/에서 시스템 관리자 콘솔을 엽니다.
2. 탐색 창에서 Run Command를 선택합니다.

   -또는-

AWS Systems Manager 홈페이지가 열리면 아래로 스크롤하여 [Run Command 탐색(Explore Run Command)]을 선택합니다.

3. Run command(Run 명령)를 선택합니다.
4. 명령 문서 목록에서 AmazonCloudWatch-ManageAgent 옆의 버튼을 선택합니다.
5. 대상 영역에서 에이전트를 설치한 인스턴스를 선택합니다.
6. 작업 목록에서 구성을 선택합니다.
7. 모드 목록에서 onPremise를 선택합니다.
8. [구성 위치(선택 사항)(Optional Configuration Location)] 상자에 마법사로 생성하여 파라미터 스토어에 저장한 에이전트 구성 파일의 이름을 입력합니다.
9. Run(실행)을 선택합니다.

   구성 파일에 지정한 구성을 사용하여 에이전트가 시작됩니다.

명령줄을 사용하여 온프레미스 서버에서 CloudWatch 에이전트를 시작하려면

- 이 명령에서 `-a fetch-config`는 에이전트가 최신 버전의 CloudWatch 에이전트 구성 파일을 로드하도록 하며 `-s`는 에이전트를 시작합니다.

  Linux: Systems Manager 파라미터 스토어에 구성 파일을 저장한 경우 다음을 입력합니다.

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m onPremise -s -c ssm:configuration-parameter-store-name
  ```

  Linux: 구성 파일을 로컬 컴퓨터에 저장한 경우 다음 명령을 입력합니다. *configuration-file-path*를 에이전트 구성 파일의 경로로 바꿉니다. 이 파일을 마법사로 생성한 경우 `config.json`이라고 하며 수동으로 생성한 경우 `amazon-cloudwatch-agent.json`이라고 할 수 있습니다.

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m onPremise -s -c file:configuration-file-path
  ```

  Windows Server: Systems Manager 파라미터 스토어에 에이전트 구성 파일을 저장한 경우 PowerShell 콘솔에서 다음을 입력합니다.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
a fetch-config -m onPremise -s -c ssm:configuration-parameter-store-name
```

Windows Server: 에이전트 구성 파일을 로컬 컴퓨터에 저장한 경우 PowerShell 콘솔에서 다음을 입력합니다. *configuration-file-path*를 에이전트 구성 파일의 경로로 바꿉니다. 이 파일을 마법사로 생성한 경우 `config.json`이라고 하며 수동으로 생성한 경우 `amazon-cloudwatch-agent.json`이라고 할 수 있습니다.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
a fetch-config -m onPremise -s -c file:configuration-file-path
```

AWS CloudFormation을 사용하여 새 인스턴스에 CloudWatch 에이전트 설치

Amazon은 새 Amazon EC2 인스턴스에서의 CloudWatch 에이전트 설치 및 업데이트를 지원하기 위해 GitHub에 여러 AWS CloudFormation 템플릿을 업로드했습니다. AWS CloudFormation 사용에 대한 자세한 내용은 AWS CloudFormation란 무엇입니까?를 참조하세요.

템플릿 위치는 AWS CloudFormation을 사용하여 EC2 인스턴스에 Amazon CloudWatch 에이전트 배포입니다. 이 위치에는 `inline` 디렉터리와 `ssm` 디렉터리가 모두 포함되어 있습니다. 각 디렉터리에는 Linux 및 Windows 인스턴스 둘 다에 대한 템플릿이 포함되어 있습니다.

- `inline` 디렉터리의 템플릿에는 AWS CloudFormation 템플릿에 포함된 CloudWatch 에이전트 구성이 있습니다. 기본적으로 Linux 템플릿은 지표 `mem_used_percent` 및 `swap_used_percent`를 수집하고, Windows 템플릿은 `Memory % Committed Bytes In Use` 및 `Paging File % Usage`를 수집합니다.

  다른 지표를 수집하도록 이러한 템플릿을 수정하려면 템플릿의 다음 섹션을 수정하세요. 다음 예제는 Linux 서버용 템플릿에서 가져옵니다. 에이전트 구성 파일의 형식 및 구문에 따라 다음과 같이 변경합니다. 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하세요.

```
{
   "metrics":{
      "append_dimensions":{
         "AutoScalingGroupName":"${!aws:AutoScalingGroupName}",
```

```
            "ImageId":"${!aws:ImageId}",
            "InstanceId":"${!aws:InstanceId}",
            "InstanceType":"${!aws:InstanceType}"
         },
         "metrics_collected":{
            "mem":{
               "measurement":[
                  "mem_used_percent"
               ]
            },
            "swap":{
               "measurement":[
                  "swap_used_percent"
               ]
            }
         }
      }
}
```

> **Note**
>
> 인라인 템플릿에서 모든 자리 표시자 변수에는 이스케이프 문자로 앞에 느낌표(!)가 있어야 합니다. 이러한 내용은 예제 템플릿에서 확인할 수 있습니다. 다른 자리 표시자 변수를 추가한 경우 해당 변수의 이름 앞에 느낌표를 추가해야 합니다.

- `ssm` 디렉터리의 템플릿은 파라미터 스토어에서 에이전트 구성 파일을 로드합니다. 이러한 템플릿을 사용하려면 먼저, 구성 파일을 생성하여 파라미터 스토어에 업로드해야 합니다. 그런 다음, 템플릿에서 파일의 파라미터 스토어 이름을 지정합니다. 구성 파일은 수동으로 생성하거나 마법사를 사용해 생성할 수 있습니다. 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.

CloudWatch 에이전트를 설치하고 에이전트 구성을 업데이트하는 데 두 가지 템플릿 유형을 모두 사용할 수 있습니다.

튜토리얼: AWS CloudFormation 인라인 템플릿을 사용하여 CloudWatch 에이전트 설치 및 구성

이 튜토리얼에서는 AWS CloudFormation을 사용하여 새 Amazon EC2 인스턴스에 CloudWatch 에이전트를 설치하는 과정을 안내합니다. 이 튜토리얼에서는 JSON 구성 파일 또는 파라미터 스토어를 사용할 필요가 없는 인라인 템플릿을 사용하여 Amazon Linux 2를 실행하는 새 인스턴스에 설치합니다.

인라인 템플릿에는 에이전트 구성이 포함되어 있습니다. 이 자습서에서는 템플릿에 포함된 기본 에이전트 구성을 사용합니다.

에이전트를 설치하는 절차를 수행한 다음 자습서에서는 계속해서 에이전트를 업데이트하는 방법을 설명합니다.

AWS CloudFormation을 사용하여 새 인스턴스에 CloudWatch 에이전트를 설치하려면

1. GitHub에서 템플릿을 다운로드합니다. 이 튜토리얼에서는 다음과 같이 Amazon Linux 2용 인라인 템플릿을 다운로드합니다.

   ```
   curl -O https://raw.githubusercontent.com/awslabs/aws-cloudformation-templates/
   master/aws/solutions/AmazonCloudWatchAgent/inline/amazon_linux.template
   ```

2. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
3. 스택 생성을 선택합니다.
4. 템플릿 선택에서 Amazon S3에 템플릿 업로드를 선택하고, 다운로드한 템플릿을 선택한 후 다음을 선택합니다.
5. 세부 정보 지정 페이지에 다음 파라미터를 입력하고 다음을 선택합니다.

   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다.
   - IAMRole: CloudWatch 지표, 로그, 추적을 작성할 수 있는 권한이 있는 IAM 역할을 선택합니다. 자세한 내용은 Amazon EC2 인스턴스에서 CloudWatch 에이전트와 함께 사용할 IAM 역할 생성 단원을 참조하십시오.
   - InstanceAMI: 스택을 시작하려는 리전에서 유효한 AMI를 선택합니다.
   - InstanceType: 유효한 인스턴스 유형을 선택합니다.
   - KeyName: 새 인스턴스에 대한 SSH 액세스를 사용 설정하려면 기존 Amazon EC2 키 페어를 선택합니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어 단원을 참조하세요.
   - SSHLocation: SSH를 사용하여 인스턴스에 연결하는 데 사용할 수 있는 IP 주소 범위를 지정합니다. 기본값은 모든 IP 주소에서의 액세스를 허용합니다.
6. 옵션 페이지에서 스택 리소스에 태그를 지정할 수 있습니다. 다음을 선택합니다.
7. [검토(Review)] 페이지에서 정보를 검토하고 스택이 IAM 리소스를 생성할 수 있음을 확인한 다음, [생성(Create)]을 선택합니다.

   콘솔을 새로 고치면 새 스택에 `CREATE_IN_PROGRESS` 상태가 있음을 알 수 있습니다.

8. 인스턴스가 생성되면 Amazon EC2 콘솔에서 해당 인스턴스를 볼 수 있습니다. 필요한 경우 호스트에 연결해 진행 상황을 확인할 수 있습니다.

   다음 명령을 사용하여 에이전트가 설치되었는지 확인합니다.

   ```
   rpm -qa amazon-cloudwatch-agent
   ```

   다음 명령을 사용하여 에이전트가 실행 중인지 확인합니다.

   ```
   ps aux | grep amazon-cloudwatch-agent
   ```

다음 절차에서는 AWS CloudFormation에서 인라인 템플릿을 사용하여 CloudWatch 에이전트를 업데이트하는 방법을 보여 줍니다. 기본 인라인 템플릿은 `mem_used_percent` 지표를 수집합니다. 이 자습서에서는 이 지표 수집을 중지하도록 에이전트 구성을 변경합니다.

AWS CloudFormation을 사용하여 CloudWatch 에이전트를 업데이트하려면

1. 이전 절차에서 다운로드한 템플릿에서 다음 줄을 제거하고 템플릿을 저장합니다.

   ```
   "mem": {

        "measurement": [
            "mem_used_percent"
          ]
    },
   ```

2. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
3. AWS CloudFormation 대시보드에서 생성된 스택을 선택하고 스택 업데이트를 선택합니다.
4. 템플릿 선택에서 Amazon S3에 템플릿 업로드를 선택하고, 수정한 템플릿을 선택한 후 다음을 선택합니다.
5. 옵션 페이지에서 다음을 선택한 후 다음을 선택합니다.
6. 검토 페이지에서 정보를 검토하고 업데이트를 선택합니다.

   잠시 후 `UPDATE_COMPLETE`가 표시됩니다.

튜토리얼: AWS CloudFormation 및 파라미터 스토어를 사용하여 CloudWatch 에이전트 설치

이 튜토리얼에서는 AWS CloudFormation을 사용하여 새 Amazon EC2 인스턴스에 CloudWatch 에이전트를 설치하는 과정을 안내합니다. 이 튜토리얼에서는 생성하여 파라미터 스토어에 저장한 에이전트 구성 파일을 사용해 Amazon Linux 2를 실행하는 새 인스턴스에 설치합니다.

에이전트를 설치하는 절차를 수행한 다음 자습서에서는 계속해서 에이전트를 업데이트하는 방법을 설명합니다.

AWS CloudFormation에서 파라미터 스토어의 구성을 사용해 새 인스턴스에 CloudWatch 에이전트를 설치하려면

1. 아직 설치하지 않은 경우 에이전트 구성 파일을 생성할 수 있도록 컴퓨터 중 한 대에 CloudWatch 에이전트 패키지를 다운로드합니다. 파라미터 스토어를 사용하여 에이전트를 다운로드하는 방법에 대한 자세한 내용은 CloudWatch 에이전트 다운로드 및 구성 단원을 참조하세요. 명령줄을 사용하여 패키지를 다운로드하는 방법에 대한 자세한 내용은 명령줄을 사용하여 CloudWatch 에이전트 다운로드 및 구성 단원을 참조하세요.
2. 에이전트 구성 파일을 생성하여 파라미터 스토어에 저장합니다. 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.
3. 다음과 같이 GitHub에서 템플릿을 다운로드합니다.

   ```
   curl -O https://raw.githubusercontent.com/awslabs/aws-cloudformation-templates/
   master/aws/solutions/AmazonCloudWatchAgent/ssm/amazon_linux.template
   ```

4. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
5. 스택 생성을 선택합니다.
6. 템플릿 선택에서 Amazon S3에 템플릿 업로드를 선택하고, 다운로드한 템플릿을 선택한 후 다음을 선택합니다.
7. 세부 정보 지정 페이지에 다음 파라미터를 입력하고 다음을 선택합니다.

   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다.
   - IAMRole: CloudWatch 지표, 로그, 추적을 작성할 수 있는 권한이 있는 IAM 역할을 선택합니다. 자세한 내용은 Amazon EC2 인스턴스에서 CloudWatch 에이전트와 함께 사용할 IAM 역할 생성 단원을 참조하십시오.
   - InstanceAMI: 스택을 시작하려는 리전에서 유효한 AMI를 선택합니다.
   - InstanceType: 유효한 인스턴스 유형을 선택합니다.

- KeyName: 새 인스턴스에 대한 SSH 액세스를 사용 설정하려면 기존 Amazon EC2 키 페어를 선택합니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어 단원을 참조하세요.
- SSHLocation: SSH를 사용하여 인스턴스에 연결하는 데 사용할 수 있는 IP 주소 범위를 지정합니다. 기본값은 모든 IP 주소에서의 액세스를 허용합니다.
- SSMKey: 생성하여 파라미터 스토어에 저장한 에이전트 구성 파일을 지정합니다.

8. 옵션 페이지에서 스택 리소스에 태그를 지정할 수 있습니다. 다음을 선택합니다.
9. [검토(Review)] 페이지에서 정보를 검토하고 스택이 IAM 리소스를 생성할 수 있음을 확인한 다음, [생성(Create)]을 선택합니다.

   콘솔을 새로 고치면 새 스택에 `CREATE_IN_PROGRESS` 상태가 있음을 알 수 있습니다.
10. 인스턴스가 생성되면 Amazon EC2 콘솔에서 해당 인스턴스를 볼 수 있습니다. 필요한 경우 호스트에 연결해 진행 상황을 확인할 수 있습니다.

    다음 명령을 사용하여 에이전트가 설치되었는지 확인합니다.

    ```
    rpm -qa amazon-cloudwatch-agent
    ```

    다음 명령을 사용하여 에이전트가 실행 중인지 확인합니다.

    ```
    ps aux | grep amazon-cloudwatch-agent
    ```

다음 절차에서는 AWS CloudFormation에서 파라미터 스토어에 저장한 에이전트 구성을 사용해 CloudWatch 에이전트를 업데이트하는 방법을 보여 줍니다.

AWS CloudFormation에서 파라미터 스토어의 구성을 사용해 CloudWatch 에이전트를 업데이트하려면

1. 파라미터 스토어에 저장된 에이전트 구성 파일을 원하는 새 구성으로 변경합니다.
2. the section called “튜토리얼: AWS CloudFormation 및 파라미터 스토어를 사용하여 CloudWatch 에이전트 설치” 주제에서 다운로드한 AWS CloudFormation 템플릿에서 버전 번호를 변경합니다. 예를 들어, `VERSION=1.0`을 `VERSION=2.0`으로 변경할 수 있습니다.
3. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
4. AWS CloudFormation 대시보드에서 생성된 스택을 선택하고 스택 업데이트를 선택합니다.

5. 템플릿 선택에서 Amazon S3에 템플릿 업로드를 선택하고, 방금 수정한 템플릿을 선택한 후 다음을 선택합니다.
6. 옵션 페이지에서 다음을 선택한 후 다음을 선택합니다.
7. 검토 페이지에서 정보를 검토하고 업데이트를 선택합니다.

   잠시 후 UPDATE_COMPLETE가 표시됩니다.

AWS CloudFormation을 사용한 CloudWatch 에이전트 설치 문제 해결

이 단원에서는 AWS CloudFormation을 사용한 CloudWatch 에이전트 설치 및 업데이트와 관련된 문제를 해결할 수 있도록 지원합니다.

업데이트 실패 시 감지

AWS CloudFormation을 사용하여 CloudWatch 에이전트 구성을 업데이트하는데 잘못된 구성을 사용하는 경우 에이전트는 CloudWatch로의 지표 전송을 중지합니다. `cfn-init-cmd.log` 파일을 살펴보면 에이전트 구성 업데이트에 성공했는지 여부를 빠르게 확인할 수 있습니다. Linux 서버에서 이 파일은 `/var/log/cfn-init-cmd.log`에 있습니다. Windows 인스턴스에서 이 파일은 `C:\cfn\log\cfn-init-cmd.log`에 있습니다.

지표가 누락됨

에이전트를 설치 또는 업데이트한 후 원하는 지표가 표시되지 않는 경우 에이전트가 해당 지표를 수집하도록 구성되어 있는지 확인하세요. 이렇게 하려면 `amazon-cloudwatch-agent.json` 파일을 점검하여 해당 지표가 나열되어 있는지 확인하고 지금 살펴보고 있는 지표 네임스페이스가 올바른 네임스페이스인지를 확인하세요. 자세한 내용은 CloudWatch 에이전트 파일 및 위치 단원을 참조하십시오.

CloudWatch 에이전트 자격 증명 기본 설정

이 섹션에서는 CloudWatch 에이전트가 다른 AWS 서비스 및 API와 통신할 때 자격 증명을 얻기 위해 사용하는 자격 증명 공급자 체인을 간략하게 설명합니다. 순서는 다음과 같습니다. 다음 목록의 2~5번에 나열된 기본 설정은 AWS SDK에 정의된 기본 설정 순서와 동일합니다. 자세한 내용은 SDK 설명서의 자격 증명 지정을 참조하세요.

1. CloudWatch 에이전트의 `common-config.toml` 파일에 정의된 공유 구성 및 자격 증명 파일. 자세한 내용은 (선택 사항) 프록시 또는 리전 정보에 대한 일반 구성 수정 단원을 참조하십시오.
2. AWS SDK 환경 변수

> **Important**
>
> Linux에서 `amazon-cloudwatch-agent-ctl` 스크립트를 사용하여 CloudWatch 에이전트를 실행하는 경우 스크립트는 에이전트를 `systemd` 서비스로 시작합니다. 이 경우 `HOME`, `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` 등과 같은 환경 변수는 에이전트에서 액세스할 수 없습니다.

3. `$HOME/%USERPROFILE%`에서 찾은 공유 구성 및 자격 증명 파일

> **Note**
>
> CloudWatch 에이전트는 Linux와 macOS의 경우 `$HOME`에서 `.aws/credentials`를 찾고 Windows의 경우 `%USERPROFILE%`을 찾습니다. AWS SDK와 달리 CloudWatch 에이전트에는 환경 변수에 액세스할 수 없는 경우 홈 디렉터리를 확인하는 대체 방법이 없습니다. 이러한 차이는 AWS SDK의 이전 버전과의 호환성을 유지하기 위한 것입니다.
> 또한 `common-config.toml`에 있는 공유 자격 증명과 달리 AWS SDK에서 파생된 공유 자격 증명이 만료되어 교체된 경우 CloudWatch 에이전트가 갱신된 자격 증명을 자동으로 선택하지 않으므로 에이전트를 다시 시작해야 합니다.

4. Amazon Elastic Container Service 태스크 정의 또는 RunTask API 작업을 사용하는 애플리케이션이 있는 경우 태스크에 대한 AWS Identity and Access Management 역할.
5. Amazon EC2 인스턴스에 연결된 인스턴스 프로파일.

모범 사례로 CloudWatch 에이전트를 사용할 때 자격 증명을 다음 순서로 지정하는 것이 가장 좋습니다.

1. 애플리케이션이 Amazon Elastic Container Service 태스크 정의 또는 RunTask API 작업을 사용하는 경우 태스크에 IAM 역할을 사용합니다.
2. 애플리케이션이 Amazon EC2 인스턴스에서 실행되는 경우 IAM 역할을 사용합니다.
3. CloudWatch 에이전트 `common-config.toml` 파일을 사용하여 자격 증명 파일을 지정합니다. 이 자격 증명 파일은 다른 AWS SDK 및 AWS CLI에서 사용하는 것과 동일합니다. 공유 자격 증명 파일을 이미 사용 중인 경우 이 용도로 해당 파일을 사용할 수도 있습니다. CloudWatch 에이전트의 `common-config.toml` 파일을 사용하여 제공하는 경우 에이전트가 만료될 때 교체된 자격 증명을 사용하고 에이전트를 재시작할 필요 없이 교체되도록 합니다.
4. 환경 변수를 사용합니다. 환경 변수 설정은 Amazon EC2 인스턴스 이외의 컴퓨터에서 개발 작업을 수행 중인 경우에 유용합니다.

> **Note**
>
> 다른 계정에 지표, 로그, 추적 전송에서 설명하는 것과 같이 다른 계정으로 원격 분석을 전송하는 경우 CloudWatch 에이전트는 이 섹션에서 설명하는 자격 증명 공급자 체인을 사용하여 초기 자격 증명 세트를 얻습니다. 이후 CloudWatch 에이전트 구성 파일에서 `role_arn`에 의해 지정된 IAM 역할을 수임할 때 해당 자격 증명을 사용합니다.

CloudWatch 에이전트 패키지의 서명 확인

Linux 서버에서는 CloudWatch 에이전트 패키지에 대한 GPG 서명 파일이 포함되어 있습니다. 퍼블릭 키를 사용하여 에이전트 다운로드 파일이 원본이며 수정되지 않았는지 확인할 수 있습니다.

Windows Server의 경우 MSI를 사용하여 서명을 확인할 수 있습니다.

macOS 컴퓨터의 경우 서명은 에이전트 다운로드 패키지에 포함되어 있습니다.

올바른 서명 파일을 찾으려면 다음 표를 참조하세요. 각 아키텍처 및 운영 체제에는 일반 링크와 각 리전별 링크가 있습니다. 예를 들어 Amazon Linux 2023, Amazon Linux 2 및 x86-64 아키텍처의 경우 다음과 같은 유효한 링크 세 개가 있습니다.

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

> **Note**
>
> CloudWatch 에이전트를 다운로드하려면 연결에서 TLS 1.2 이상을 사용해야 합니다.

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
x86-64	Amazon Linux 2023 및 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/	https://amazoncloudwatch-agent.s3.amazonaws.com/

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
		suse/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm	suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/darwin/amd64/latest/amazon-cloudwatch-agent.pkg.sig
x86-64	Windows	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
ARM64	Amazon Linux 2023 및 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig

아키텍처	플랫폼	다운로드 링크	서명 파일 링크
ARM64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig

Linux 서버에서 CloudWatch 에이전트 패키지를 확인하려면

1. 퍼블릭 키를 다운로드합니다.

```
shell$ wget https://amazoncloudwatch-agent.s3.amazonaws.com/assets/amazon-
cloudwatch-agent.gpg
```

2. 퍼블릭 키를 인증 키로 가져옵니다.

```
shell$  gpg --import amazon-cloudwatch-agent.gpg
gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

   다음 단계에서 필요하므로 키 값을 적어 둡니다. 이전 예제에서 키 값은 3B789C72입니다.

3. *#-#*을 이전 단계의 값으로 대체하고 다음 명령을 실행하여 지문을 확인합니다.

```
shell$  gpg --fingerprint key-value
pub   2048R/3B789C72 2017-11-14
      Key fingerprint = 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
uid                  Amazon CloudWatch Agent
```

   지문 문자열이 다음과 동일해야 합니다.

```
9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
```

   지문 문자열이 일치하지 않으면 에이전트를 설치하지 말고 Amazon Web Services에 문의하세요.

지문을 확인한 후 이를 사용하여 CloudWatch 에이전트 패키지의 서명을 확인할 수 있습니다.

4. wget을 사용하여 패키지 서명 파일을 다운로드합니다. 올바른 서명 파일을 확인하려면 이전 표를 참조하세요.

```
wget Signature File Link
```

5. 서명을 확인하려면 gpg --verify를 실행합니다.

```
shell$ gpg --verify signature-filename agent-download-filename
gpg: Signature made Wed 29 Nov 2017 03:00:59 PM PST using RSA key ID 3B789C72
gpg: Good signature from "Amazon CloudWatch Agent"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

출력에 BAD signature 문구가 포함된 경우, 절차를 올바르게 수행했는지 확인합니다. 이 응답이 계속되는 경우 Amazon Web Services에 문의하고, 다운로드한 파일을 사용하지 마세요.

신뢰에 대한 경고를 참조하세요. 사용자 또는 사용자가 신뢰하는 사람이 서명한 키만 신뢰됩니다. 이는 서명이 잘못되었음을 의미하지 않으며, 단지 해당 사용자가 퍼블릭 키를 확인하지 않은 것입니다.

Windows Server를 실행하는 서버에서 CloudWatch 에이전트 패키지를 확인하려면

1. https://gnupg.org/download/에서 Windows용 GnuPG를 다운로드 및 설치합니다. 설치할 때 Shell Extension (GpgEx)(셸 확장(GpgEx)) 옵션을 포함합니다.

나머지 단계는 Windows PowerShell에서 수행할 수 있습니다.

2. 퍼블릭 키를 다운로드합니다.

```
PS> wget https://amazoncloudwatch-agent.s3.amazonaws.com/assets/amazon-cloudwatch-
agent.gpg -OutFile amazon-cloudwatch-agent.gpg
```

3. 퍼블릭 키를 인증 키로 가져옵니다.

```
PS>  gpg --import amazon-cloudwatch-agent.gpg
gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
gpg: Total number processed: 1
```

```
gpg: imported: 1 (RSA: 1)
```

다음 단계에서 필요하므로 키 값을 기록해 둡니다. 이전 예제에서 키 값은 3B789C72입니다.

4. *#-#*을 이전 단계의 값으로 대체하고 다음 명령을 실행하여 지문을 확인합니다.

```
PS>  gpg --fingerprint key-value
pub   rsa2048 2017-11-14 [SC]
      9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
uid           [ unknown] Amazon CloudWatch Agent
```

지문 문자열이 다음과 동일해야 합니다.

```
9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
```

지문 문자열이 일치하지 않으면 에이전트를 설치하지 말고 Amazon Web Services에 문의하세요.

지문을 확인한 후 이를 사용하여 CloudWatch 에이전트 패키지의 서명을 확인할 수 있습니다.

5. wget을 사용하여 패키지 서명 파일을 다운로드합니다. 올바른 서명 파일인지 확인하려면 CloudWatch 에이전트 다운로드 링크 단원을 참조하세요.
6. 서명을 확인하려면 gpg --verify를 실행합니다.

```
PS> gpg --verify sig-filename agent-download-filename
gpg: Signature made 11/29/17 23:00:45 Coordinated Universal Time
gpg:                using RSA key D58167303B789C72
gpg: Good signature from "Amazon CloudWatch Agent" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

출력에 `BAD signature` 문구가 포함된 경우, 절차를 올바르게 수행했는지 확인합니다. 이 응답이 계속되는 경우 Amazon Web Services에 문의하고, 다운로드한 파일을 사용하지 마세요.

신뢰에 대한 경고를 참조하세요. 사용자 또는 사용자가 신뢰하는 사람이 서명한 키만 신뢰됩니다. 이는 서명이 잘못되었음을 의미하지 않으며, 단지 해당 사용자가 퍼블릭 키를 확인하지 않은 것입니다.

macOS 컴퓨터에서 CloudWatch 에이전트 패키지를 확인하려면

- macOS에서 서명을 확인하기 위한 방법은 두 가지가 있습니다.

- 다음 명령을 실행하여 지문을 확인합니다.

```
pkgutil --check-signature amazon-cloudwatch-agent.pkg
```

다음과 유사한 결과가 표시되어야 합니다.

```
Package "amazon-cloudwatch-agent.pkg":
        Status: signed by a developer certificate issued by Apple for
 distribution
        Signed with a trusted timestamp on: 2020-10-02 18:13:24 +0000
        Certificate Chain:
        1. Developer ID Installer: AMZN Mobile LLC (94KV3E626L)
        Expires: 2024-10-18 22:31:30 +0000
        SHA256 Fingerprint:
        81 B4 6F AF 1C CA E1 E8 3C 6F FB 9E 52 5E 84 02 6E 7F 17 21 8E FB
        0C 40 79 13 66 8D 9F 1F 10 1C

 ------------------------------------------------------------------------
        2. Developer ID Certification Authority
        Expires: 2027-02-01 22:12:15 +0000
        SHA256 Fingerprint:
        7A FC 9D 01 A6 2F 03 A2 DE 96 37 93 6D 4A FE 68 09 0D 2D E1 8D 03
        F2 9C 88 CF B0 B1 BA 63 58 7F

 ------------------------------------------------------------------------
        3. Apple Root CA
        Expires: 2035-02-09 21:40:36 +0000
        SHA256 Fingerprint:
        B0 B1 73 0E CB C7 FF 45 05 14 2C 49 F1 29 5E 6E DA 6B CA ED 7E 2C
        68 C5 BE 91 B5 A1 10 01 F0 24
```

- 또는 .sig 파일을 다운로드하여 사용합니다. 이 방법을 사용하려면 다음 단계를 따릅니다.
  - 다음 명령을 입력하여 macOS 호스트에 GPG 애플리케이션을 설치합니다.

```
brew install GnuPG
```

- curl을 사용하여 패키지 서명 파일을 다운로드합니다. 올바른 서명 파일인지 확인하려면 CloudWatch 에이전트 다운로드 링크 단원을 참조하세요.
- 서명을 확인하려면 gpg --verify를 실행합니다.

```
PS> gpg --verify sig-filename agent-download-filename
gpg: Signature made 11/29/17 23:00:45 Coordinated Universal Time
gpg:                using RSA key D58167303B789C72
gpg: Good signature from "Amazon CloudWatch Agent" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

출력에 BAD signature 문구가 포함된 경우, 절차를 올바르게 수행했는지 확인합니다. 이 응답이 계속되는 경우 Amazon Web Services에 문의하고, 다운로드한 파일을 사용하지 마세요.

신뢰에 대한 경고를 참조하세요. 사용자 또는 사용자가 신뢰하는 사람이 서명한 키만 신뢰됩니다. 이는 서명이 잘못되었음을 의미하지 않으며, 단지 해당 사용자가 퍼블릭 키를 확인하지 않은 것입니다.

CloudWatch 에이전트 구성 파일 생성

서버에서 CloudWatch 에이전트를 실행하기 전에 하나 이상의 CloudWatch 에이전트 구성 파일을 생성해야 합니다.

에이전트 구성 파일은 사용자 지정 지표를 포함하여 에이전트가 수집할 지표, 로그, 추적을 지정하는 JSON 파일입니다. 마법사를 사용하거나 Scratch에서 직접 생성하여 이 구성 파일을 생성할 수 있습니다. 또한 마법사를 사용하여 구성 파일을 처음으로 만든 다음, 수동으로 수정할 수 있습니다. 수동으로 파일을 생성하거나 수정하는 경우 프로세스는 더 복잡하지만, 수집된 지표를 더 잘 제어할 수 있으며 마법사를 통해 사용할 수 없는 지표를 지정할 수 있습니다.

에이전트 구성 파일을 변경할 때마다 에이전트를 다시 시작하여 변경 사항이 적용되도록 해야 합니다. 에이전트를 다시 시작하려면 CloudWatch 에이전트 시작 섹션의 지침을 따르세요.

구성 파일을 만든 후 수동으로 JSON 파일로 저장하여 서버에 에이전트를 설치할 때 이 파일을 사용할 수 있습니다. 또는 서버에 에이전트를 설치할 때 Systems Manager를 사용하려는 경우 해당 파일을 Systems Manager 파라미터 스토어에 저장할 수 있습니다.

CloudWatch 에이전트는 여러 개의 구성 파일을 사용하는 것을 지원합니다. 자세한 내용은 여러 CloudWatch 에이전트 구성 파일 단원을 참조하십시오.

CloudWatch 에이전트가 수집한 지표, 로그, 추적 사용 시 요금이 발생합니다. 요금에 대한 자세한 정보는 Amazon CloudWatch 요금을 참조하세요.

내용

마법사로 CloudWatch 에이전트 구성 파일 생성

에이전트 구성 파일 마법사 `amazon-cloudwatch-agent-config-wizard`는 필요에 맞게 CloudWatch 에이전트를 구성하는 데 도움이 되는 일련의 질문을 합니다.

필수 자격 증명

마법사를 시작하기 전에 AWS 자격 증명 및 구성 파일이 제자리에 있는 경우, 마법사가 사용할 자격 증명 및 AWS 리전을 자동 검색할 수 있습니다. 이러한 파일에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 구성 및 자격 증명 파일 단원을 참조하세요.

AWS 자격 증명 파일에서 마법사는 기본 자격 증명을 확인하고 다음과 같은 `AmazonCloudWatchAgent` 섹션도 찾습니다.

```
[AmazonCloudWatchAgent]
aws_access_key_id = my_access_key
aws_secret_access_key = my_secret_key
```

마법사는 기본 자격 증명, `AmazonCloudWatchAgent`의 자격 증명 및 `Others` 옵션을 표시합니다. 사용할 자격 증명을 선택할 수 있습니다. `Others`를 선택할 경우 자격 증명을 입력할 수 있습니다.

my_access_key 및 *my_secret_key*의 경우 Systems Manager 파라미터 스토어에 대한 쓰기 권한이 있는 IAM 사용자의 키를 사용합니다. CloudWatch 에이전트에 필요한 IAM 사용자에 대한 자세한 내용은 온프레미스 서버에서 CloudWatch 에이전트와 함께 사용할 IAM 사용자 생성 단원을 참조하세요.

AWS 구성 파일에서 에이전트가 지표를 보낼 리전을 지정할 수 있습니다(해당 리전이 `[default]` 섹션의 리전과 다른 경우). 기본값은 Amazon EC2 인스턴스가 있는 리전에 지표를 게시하는 것입니다. 지표를 다른 리전에 게시해야 하는 여기에서 리전을 지정합니다. 다음 예에서는 지표가 `us-west-1` 리전에 게시되어 있습니다.

```
[AmazonCloudWatchAgent]
region = us-west-1
```

CloudWatch 에이전트 구성 마법사 실행

CloudWatch 에이전트 구성 파일을 생성하려면

1. 다음을 입력하여 CloudWatch 에이전트 구성 마법사를 시작합니다.

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
   ```

   Windows Server를 실행하는 서버에서 다음 명령을 실행하여 마법사를 시작합니다.

   ```
   cd "C:\Program Files\Amazon\AmazonCloudWatchAgent"
   ```

   ```
   .\amazon-cloudwatch-agent-config-wizard.exe
   ```

2. 질문에 답하여 서버의 구성 파일을 사용자 지정합니다.
3. 구성 파일을 로컬에 저장하는 경우 구성 파일 `config.json`은 Linux 서버에서는 `/opt/aws/amazon-cloudwatch-agent/bin/`에, Windows 서버에서는 `C:\Program Files\Amazon\AmazonCloudWatchAgent`에 저장됩니다. 그러면 이 파일을 에이전트를 설치하려고 하는 다른 서버에 복사할 수 있습니다.

   Systems Manager를 사용하여 에이전트를 설치하고 구성하려는 경우 Systems Manager 파라미터 스토어에 파일을 저장할지 여부를 묻는 메시지가 표시되면 [예(Yes)]라고 대답해야 합니다. 또한 CloudWatch 에이전트를 설치하는 데 SSM Agent를 사용하지 않는 경우에도 파라미터 스토어에 파일을 저장하도록 선택할 수 있습니다. 파일을 파라미터 스토어에 저장할 수 있으려면 충분한 권한이 있는 IAM 역할을 사용해야 합니다. 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성 단원을 참조하십시오.

CloudWatch 에이전트의 미리 정의된 지표 세트

마법사는 상세 레벨이 다양한 사전 정의된 지표 집합을 사용하여 구성되어 있습니다. 이러한 지표 집합은 다음 표에 표시되어 있습니다. 지표에 대한 자세한 내용은 CloudWatch 에이전트가 수집하는 지표 섹션을 참조하세요.

> **Note**
>
> 파라미터 스토어는 표준 및 고급 티어의 파라미터를 지원합니다. 이러한 파라미터 계층은 다음 표에 설명된 기본, 표준 및 고급 지표 상세 수준과 관련이 없습니다.

Linux를 실행하는 Amazon EC2 인스턴스

세부 정보 수준	포함된 지표
기본	Mem: mem_used_percent Disk: disk_used_percent disk와 같은 disk_used_percent 지표에는 Partition 의 측정기준이 있는데, 이는 생성된 사용자 지정 지표의 수가 인스턴스와 연결된 파티션의 수에 따라 달라짐을 뜻합니다. 보유한 디스크 파티션의 수는 사용 중인 AMI 및 서버에 연결하는 Amazon EBS 볼륨의 수에 따라 달라집니다.
표준	CPU: cpu_usage_idle , cpu_usage_iowait , cpu_usage_user , cpu_usage_system Disk: disk_used_percent , disk_inodes_free Diskio: diskio_io_time Mem: mem_used_percent Swap: swap_used_percent
고급	CPU: cpu_usage_idle , cpu_usage_iowait , cpu_usage_user , cpu_usage_system Disk: disk_used_percent , disk_inodes_free Diskio: diskio_io_time , diskio_write_bytes , diskio_read_bytes , diskio_writes , diskio_reads Mem: mem_used_percent Netstat: netstat_tcp_established , netstat_tcp_time_wait Swap: swap_used_percent

Linux를 실행하는 온프레미스 서버

세부 정보 수준	포함된 지표
기본	Disk: disk_used_percent Diskio: diskio_write_bytes , diskio_read_bytes , diskio_writes , diskio_reads Mem: mem_used_percent Net: net_bytes_sent , net_bytes_recv , net_packets_sent , net_packets_recv Swap: swap_used_percent
표준	CPU: cpu_usage_idle , cpu_usage_iowait Disk: disk_used_percent , disk_inodes_free Diskio: diskio_io_time , diskio_write_bytes , diskio_read_bytes , diskio_writes , diskio_reads Mem: mem_used_percent Net: net_bytes_sent , net_bytes_recv , net_packets_sent , net_packets_recv Swap: swap_used_percent
고급	CPU: cpu_usage_guest , cpu_usage_idle , cpu_usage_iowait , cpu_usage_steal , cpu_usage_user , cpu_usage_system Disk: disk_used_percent , disk_inodes_free Diskio: diskio_io_time , diskio_write_bytes , diskio_read_bytes , diskio_writes , diskio_reads Mem: mem_used_percent Net: net_bytes_sent , net_bytes_recv , net_packets_sent , net_packets_recv

세부 정보 수준	포함된 지표
	Netstat: `netstat_tcp_established` , `netstat_tcp_time_wait` Swap: `swap_used_percent`

Windows Server를 실행하는 Amazon EC2 인스턴스

> **Note**
>
> 이 테이블에 나열된 지표 이름은 콘솔에서 볼 때 지표가 어떻게 표시되는지를 보여줍니다. 실제 지표 이름에는 첫 단어가 포함되지 않을 수 있습니다. 예를 들어 `LogicalDisk % Free Space`의 실제 지표 이름은 `% Free Space`입니다.

세부 정보 수준	포함된 지표
기본	Memory: `Memory % Committed Bytes In Use` LogicalDisk: `LogicalDisk % Free Space`
표준	Memory: `Memory % Committed Bytes In Use` Paging: `Paging File % Usage` 프로세서: `Processor % Idle Time`, `Processor % Interrupt Time`, `Processor % User Time` PhysicalDisk: `PhysicalDisk % Disk Time` LogicalDisk: `LogicalDisk % Free Space`
고급	Memory: `Memory % Committed Bytes In Use` Paging: `Paging File % Usage` 프로세서: `Processor % Idle Time`, `Processor % Interrupt Time`, `Processor % User Time` LogicalDisk: `LogicalDisk % Free Space`

세부 정보 수준	포함된 지표
	PhysicalDisk: PhysicalDisk % Disk Time , PhysicalDisk Disk Write Bytes/sec , PhysicalDisk Disk Read Bytes/sec , PhysicalDisk Disk Writes/sec , PhysicalDisk Disk Reads/sec TCP: TCPv4 Connections Established , TCPv6 Connections Established

Windows Server를 실행하는 온프레미스 서버

> **Note**
>
> 이 테이블에 나열된 지표 이름은 콘솔에서 볼 때 지표가 어떻게 표시되는지를 보여줍니다. 실제 지표 이름에는 첫 단어가 포함되지 않을 수 있습니다. 예를 들어 LogicalDisk % Free Space의 실제 지표 이름은 % Free Space입니다.

세부 정보 수준	포함된 지표
기본	페이징: Paging File % Usage 프로세서: Processor % Processor Time LogicalDisk:LogicalDisk % Free Space PhysicalDisk: PhysicalDisk Disk Write Bytes/sec , PhysicalDisk Disk Read Bytes/sec , PhysicalDisk Disk Writes/sec , PhysicalDisk Disk Reads/sec Memory: Memory % Committed Bytes In Use 네트워크 인터페이스: Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec , Network Interface Packets Sent/sec, Network Interface Packets Received/sec
표준	Paging: Paging File % Usage

세부 정보 수준	포함된 지표
	프로세서: Processor % Processor Time, Processor % Idle Time, Processor % Interrupt Time LogicalDisk: LogicalDisk % Free Space PhysicalDisk: PhysicalDisk % Disk Time , PhysicalDisk Disk Write Bytes/sec , PhysicalDisk Disk Read Bytes/sec , PhysicalDisk Disk Writes/sec , PhysicalDisk Disk Reads/sec Memory: Memory % Committed Bytes In Use 네트워크 인터페이스: Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec , Network Interface Packets Sent/sec, Network Interface Packets Received/sec
고급	페이징:Paging File % Usage 프로세서: Processor % Processor Time, Processor % Idle Time, Processor % Interrupt Time, Processor % User Time LogicalDisk: LogicalDisk % Free Space PhysicalDisk: PhysicalDisk % Disk Time , PhysicalDisk Disk Write Bytes/sec , PhysicalDisk Disk Read Bytes/sec , PhysicalDisk Disk Writes/sec , PhysicalDisk Disk Reads/sec Memory: Memory % Committed Bytes In Use 네트워크 인터페이스: Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec , Network Interface Packets Sent/sec, Network Interface Packets Received/sec TCP: TCPv4 Connections Established , TCPv6 Connections Established

수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집

CloudWatch 에이전트 구성 파일은 `agent`, `metrics`, `logs`, `traces`의 네 섹션이 있는 JSON 파일입니다.

- `agent` 섹션에는 에이전트의 전체 구성에 대한 필드가 포함되어 있습니다.
- `metrics` 섹션은 수집하여 CloudWatch에 게시할 사용자 지정 지표를 지정합니다. 에이전트를 사용하여 로그만 수집하는 경우 파일에서 `metrics` 섹션을 생략할 수 있습니다.
- `logs` 섹션은 CloudWatch Logs에 게시되는 로그 파일을 지정합니다. 서버에서 Windows Server가 실행되는 경우 Windows 이벤트 로그에 따른 이벤트가 포함될 수 있습니다.
- 이 `traces` 섹션에서는 수집되어 AWS X-Ray로 전송되는 추적의 소스를 지정합니다.

다음 섹션에서는 이 JSON 파일의 구조 및 필드에 대해 설명합니다. 이 구성 파일에 대한 스키마 정의도 볼 수 있습니다. 스키마 정의는 Linux 서버의 *`installation-directory`*`/doc/amazon-cloudwatch-agent-schema.json`, Windows Server를 실행하는 서버의 *`installation-directory`*`/amazon-cloudwatch-agent-schema.json`에 있습니다.

에이전트 구성 파일을 수동으로 생성하거나 편집할 경우 이름을 지정할 수 있습니다. 문제를 간단하게 해결할 수 있도록 이름을 Linux 서버에서는 `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`로 지정하고, Windows 서버를 실행하는 서버에서는 `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`로 지정하는 것이 좋습니다. 파일을 생성한 후에는 해당 파일을 에이전트를 설치하려고 하는 다른 서버에 복사할 수 있습니다.

> **Note**
>
> CloudWatch 에이전트가 수집한 지표, 로그, 추적 사용 시 요금이 발생합니다. 요금에 대한 자세한 정보는 Amazon CloudWatch 요금을 참조하세요.

CloudWatch 에이전트 구성 파일: 에이전트 섹션

`agent` 섹션은 다음 필드를 포함할 수 있습니다. 마법사는 `agent` 섹션을 생성하지 않습니다. 대신, 이를 생략하고 이 섹션의 모든 필드에 대해 기본값을 사용합니다.

- `metrics_collection_interval` – 선택 사항입니다. 이 구성 파일에 지정되어 있는 모든 지표가 수집될 빈도를 지정합니다. 특정 유형의 지표에 대해 이 값을 재정의할 수 있습니다.

이 값은 초 단위로 지정됩니다. 예를 들어, 10을 지정하면 10초마다 지표가 수집되도록 설정되며, 300으로 설정하면 5분마다 지표가 수집되도록 지정됩니다.

이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 고분해능 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하세요.

기본값은 60입니다.

- `region` – Amazon EC2 인스턴스를 모니터링할 때 CloudWatch 엔드포인트에 사용할 리전을 지정합니다. 수집되는 지표는 이 리전(예: `us-west-1`)에 전송됩니다. 이 필드를 생략하면 에이전트가 지표를 Amazon EC2 인스턴스가 있는 리전에 전송합니다.

  온프레미스 서버를 모니터링하는 경우 이 필드가 사용되지 않으며, 에이전트는 AWS 구성 파일의 `AmazonCloudWatchAgent` 프로필에서 리전을 읽습니다.
- `credentials` – 지표, 로그, 추적을 다른 AWS 계정에 전송할 때 사용할 IAM 역할을 지정합니다. 지정된 경우 이 필드는 1개의 파라미터 `role_arn`를 포함합니다.
  - `role_arn` – 지표, 로그, 추적을 다른 AWS 계정에 전송할 때 인증에 사용할 IAM 역할의 Amazon 리소스 이름(ARN)을 지정합니다. 자세한 내용은 다른 계정에 지표, 로그, 추적 전송 단원을 참조하십시오.
- `debug` – 선택 사항입니다. 디버그 로그 메시지와 함께 CloudWatch 에이전트를 실행하도록 지정합니다. 기본 값은 `false`입니다.
- `aws_sdk_log_level` – 선택 사항입니다. 버전 1.247350.0 이상의 CloudWatch 에이전트에서만 지원됩니다.

  에이전트가 AWSSDK 엔드포인트에 대한 로깅을 수행하도록 이 필드를 지정할 수 있습니다. 이 필드의 값은 다음 옵션 중 하나 이상을 포함할 수 있습니다. `|`문자를 사용하여 여러 옵션을 구분합니다.
  - `LogDebug`
  - `LogDebugWithSigning`
  - `LogDebugWithHTTPBody`
  - `LogDebugRequestRetries`
  - `LogDebugWithEventStreamBody`

  이러한 옵션에 대한 자세한 정보는 LogLevelType 단원을 참조하세요.
- `logfile` – CloudWatch 에이전트가 로그 메시지를 작성할 위치를 지정합니다. 비어 있는 문자열을 지정하면 로그가 stderr에 저장됩니다. 이 옵션을 지정하지 않으면 다음과 같이 기본 위치가 사용됩니다.

- Linux: /opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log
- Windows Server: c:\\ProgramData\\Amazon\\CloudWatchAgent\\Logs\\amazon-cloudwatch-agent.log

CloudWatch 에이전트는 생성한 로그 파일을 자동으로 교체합니다. 로그 파일의 크기가 100MB에 도달하면 교체됩니다. 에이전트는 교체된 로그 파일을 최대 7일 동안 보존하며 교체된 백업 로그 파일을 최대 5개까지 보존합니다. 백업 로그 파일의 파일 이름에는 타임스탬프가 추가됩니다. 이 타임스탬프는 파일이 교체된 날짜와 시간을 표시합니다(예: amazon-cloudwatch-agent-2018-06-08T21-01-50.247.log.gz).

- omit_hostname – 선택 사항입니다. 기본적으로 호스트 이름은 metrics 섹션의 append_dimensions 필드를 사용하는 경우 외에는 에이전트가 수집하는 지표의 측정기준으로 게시됩니다. append_dimensions를 사용하지 않는 경우에도 호스트 이름이 차원으로 게시되지 않도록 하려면 omit_hostname 을 true로 설정합니다. 기본 값은 false입니다.
- run_as_user – 선택 사항입니다. CloudWatch 에이전트를 실행하는 데 사용할 사용자를 지정합니다. 이 파라미터를 지정하지 않으면 루트 사용자가 사용됩니다. 이 옵션은 Linux 서버에서만 유효합니다.

  이 옵션을 지정하면 CloudWatch 에이전트를 시작하기 전에 사용자가 존재해야 합니다. 자세한 내용은 다른 사용자로 CloudWatch 에이전트 실행 단원을 참조하세요.
- user_agent – 선택 사항입니다. CloudWatch 백엔드에 대한 API 호출을 수행할 때 CloudWatch 에이전트가 사용하는 user-agent 문자열을 지정합니다. 기본값은 에이전트 버전, 에이전트를 컴파일하는 데 사용된 Go 프로그래밍 언어의 버전, 런타임 운영 체제 및 아키텍처, 구축 시간, 사용 설정된 플러그 인으로 구성된 문자열입니다.
- usage_data – 선택 사항입니다. 기본적으로 CloudWatch 에이전트는 지표나 로그를 CloudWatch에 게시할 때마다 자신에 대한 상태 및 성능 데이터를 CloudWatch로 보냅니다. 이 데이터에는 비용이 발생하지 않습니다. usage_data에 false를 지정하면 에이전트가 이 데이터를 보내는 것을 방지할 수 있습니다. 이 매개 변수를 생략하면 기본값인 true 값이 사용되고 에이전트는 상태 및 성능 데이터를 보냅니다.

  이 값을 false로 설정한 경우 에이전트를 중지했다가 다시 시작해야 적용됩니다.

다음은 agent 섹션의 예입니다.

```
"agent": {
   "metrics_collection_interval": 60,
   "region": "us-west-1",
   "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log",
```

```
  "debug": false,
  "run_as_user": "cwagent"
 }
```

CloudWatch 에이전트 구성 파일: 지표 섹션

Linux 및 Windows 공통 필드

Linux 또는 Windows Server가 실행되는 서버의 경우 metrics 섹션에 다음 필드가 있습니다.

- namespace – 선택 사항입니다. 에이전트에 의해 수집되는 지표에 대해 사용할 네임스페이스입니다. 기본 값은 CWAgent입니다. 최대 길이는 255자입니다. 다음은 그 예제입니다.

  ```
  {
    "metrics": {
      "namespace": "Development/Product1Metrics",
     ......
     },
  }
  ```

- append_dimensions – 선택 사항입니다. 에이전트가 수집한 모든 지표에 Amazon EC2 지표 측정기준을 추가합니다. 이로 인해 에이전트가 호스트 이름을 측정기준으로 게시하지는 않습니다.

  다음 목록은 append_dimensions에 대해 지원되는 키-값 페어만 보여 줍니다. 다른 키-값 페어는 무시됩니다. 에이전트는 다음 목록에 표시된 대로 키-값 쌍을 지원합니다. 다른 측정기준 이름을 게시하기 위해 키 값을 변경할 수 없습니다.

  - "ImageId":"${aws:ImageId}"는 인스턴스의 AMI ID를 ImageId 측정기준의 값으로 설정합니다.
  - "InstanceId":"${aws:InstanceId}"는 인스턴스의 인스턴스 ID를 InstanceId 측정기준의 값으로 설정합니다.
  - "InstanceType":"${aws:InstanceType}"는 인스턴스의 인스턴스 유형을 InstanceType 측정기준의 값으로 설정합니다.
  - "AutoScalingGroupName":"${aws:AutoScalingGroupName}"는 인스턴스의 Auto Scaling 그룹 이름을 AutoScalingGroupName 측정기준의 값으로 설정합니다.

  임의의 키-값 페어를 사용하여 지표에 측정기준을 추가하려면 특정 지표 유형에 대해 필드의 append_dimensions 파라미터를 사용해야 합니다.

  Amazon EC2 메타데이터에 따라 달라지는 값을 지정하고 프록시를 사용하는 경우 서버가 Amazon EC2의 엔드포인트에 액세스할 수 있는지 확인해야 합니다. 이러한 엔드포인트에 대한 자세한 내용

은 Amazon Web Services 일반 참조의 Amazon Elastic Compute Cloud(Amazon EC2)를 참조하세요.

- `aggregation_dimensions` – 선택 사항입니다. 수집된 지표가 집계될 측정기준을 지정합니다. 예를 들어 `AutoScalingGroupName` 측정기준의 지표를 롤업하는 경우 각 Auto Scaling 그룹의 모든 인스턴스에서 지표가 집계되어 전체적으로 표시될 수 있습니다.

  하나 또는 여러 측정기준에 따라 지표를 롤업할 수 있습니다. 예를 들어, `[["InstanceId"], ["InstanceType"], ["InstanceId","InstanceType"]]`을 지정하면 인스턴스 ID 단일, 인스턴스 유형 단일 및 두 측정기준의 조합에 대해 지표를 집계할 수 있습니다.

  `[]`를 지정하면 모든 측정기준을 무시하고 모든 지표를 하나의 모음에 롤업할 수도 있습니다.

- `endpoint_override` – 에이전트가 지표를 전송하는 엔드포인트로 사용할 FIPS 엔드포인트 또는 프라이빗 링크를 지정합니다. 이 필드를 지정하고 프라이빗 링크를 설정하면 지표를 Amazon VPC 엔드포인트에 전송할 수 있습니다. 자세한 내용은 Amazon VPC란? 단원을 참조하세요.

  `endpoint_override`의 값은 URL인 문자열이어야 합니다.

  예를 들어, 구성 파일의 지표 섹션에서 다음 부분은 지표를 전송할 때 에이전트가 VPC 엔드포인트를 사용하도록 설정합니다.

```
{
  "metrics": {
    "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXXXXX.monitoring.us-
east-1.vpce.amazonaws.com",
   ......
   },
}
```

- `metrics_collected` - 필수입니다. `StatsD` 또는 `collectd`를 통해 수집된 사용자 지정 지표를 포함하여 수집할 지표를 지정합니다. 이 단원에는 여러 하위 섹션이 포함되어 있습니다.

  `metrics_collected` 섹션의 내용은 이 구성 파일이 Linux를 실행하는 서버용인지 아니면 Windows Server를 실행하는 서버용인지에 따라 달라집니다.

- `force_flush_interval` – 지표가 서버로 전송되기 전에 메모리 버퍼에 남아 있는 최대 시간(초)을 지정합니다. 이 설정과 상관없이 버퍼 내 지표의 크기가 1MB에 도달하거나 지표 개수가 1,000개가 되면 지표가 즉시 서버로 전송됩니다.

  기본값은 60입니다.

- credentials – 지표를 다른 계정에 전송할 때 사용할 IAM 역할을 지정합니다. 지정된 경우 이 필드는 1개의 파라미터 role_arn를 포함합니다.
  - role_arn – 지표를 다른 계정에 전송할 때 인증에 사용할 IAM 역할의 ARN을 지정합니다. 자세한 내용은 다른 계정에 지표, 로그, 추적 전송 단원을 참조하세요. 여기에 지정한 경우 이 값은 구성 파일의 agent 섹션에 지정되어 있는 role_arn을 재정의합니다(있는 경우).

Linux 섹션

Linux를 실행하는 서버에서 구성 파일의 metrics_collected 섹션에는 다음 필드도 포함될 수 있습니다.

이러한 필드 중 다수에는 해당 리소스에 대해 수집하고 싶은 지표나 나열된 measurement 섹션이 포함되어 있습니다. 이러한 measurement 섹션은 swap_used 같이 완전한 척도 이름을 지정하거나, 리소스 유형에 추가되는 척도 이름의 일부만 지정할 수 있습니다. 예를 들어 diskio 섹션의 measurement 섹션에 reads를 지정하면 diskio_reads 지표가 수집될 수 있습니다.

- collectd – 선택 사항입니다. collectd 프로토콜을 사용하여 사용자 지정 지표를 검색하려 한다는 것을 지정합니다. collectd 소프트웨어를 사용하여 지표를 CloudWatch 에이전트에 전송합니다. collectd에 사용할 수 있는 구성 옵션에 대한 자세한 내용은 collectd를 사용하여 사용자 지정 지표 검색 단원을 참조하세요.
- cpu – 선택 사항입니다. CPU 지표가 수집되도록 지정합니다. 이 단원은 리눅스 인스턴스에만 유효합니다. 수집할 CPU 지표에 대해 resources 및 totalcpu 필드 중 하나 이상을 포함해야 합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.
  - drop_original_metrics – 선택 사항입니다. metrics 섹션의 aggregation_dimensions 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.
  - resources – 선택 사항입니다. CPU당 지표가 수집되도록 하려면 이 필드의 값을 *로 지정해야 합니다. 허용되는 유일한 값은 *입니다.
  - totalcpu – 선택 사항입니다. 모든 CPU 코어에서 집계되는 CPU 지표를 보고할지 여부를 지정합니다. 기본값은 true입니다.
  - measurement – 수집할 cpu 지표의 배열을 지정합니다. 가능한 값은 time_active, time_guest, time_guest_nice, time_idle, time_iowait, time_irq, time_nice, time_softirq, time_steal, time_system, time_user, usage_active, usage_guest,

usage_guest_nice, usage_idle, usage_iowait, usage_irq, usage_nice, usage_softirq, usage_steal, usage_system, usage_user입니다. cpu를 포함하는 경우 이 필드는 필수입니다.

기본적으로 cpu_usage_* 지표의 단위는 Percent이며, cpu_time_* 지표에는 단위가 없습니다.

각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.

- rename – 이 지표에 대해 다른 이름을 지정합니다.
- unit – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None)를 재정의합니다. 지정하는 단위는 MetricDatum의 Unit 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.

- metrics_collection_interval – 선택 사항입니다. CPU 지표를 수집하여 구성 파일의 agent 섹션에 지정되어 있는 글로벌 metrics_collection_interval 값을 재정의할 빈도를 지정합니다.

  이 값은 초 단위로 지정됩니다. 예를 들어, 10을 지정하면 10초마다 지표가 수집되도록 설정되며, 300으로 설정하면 5분마다 지표가 수집되도록 지정됩니다.

  이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 고분해능 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하세요.
- append_dimensions – 선택 사항입니다. CPU 지표에만 사용할 수 있는 추가 측정기준입니다. 이 필드를 지정하면 에이전트가 수집하는 모든 유형의 지표에 대해 사용되는 글로벌 append_dimensions 필드에 지정되어 있는 측정기준 외에 이 측정기준도 사용됩니다.

- disk – 선택 사항입니다. 디스크 지표가 수집됨을 나타냅니다. 이 단원은 리눅스 인스턴스에만 유효합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.
  - drop_original_metrics – 선택 사항입니다. metrics 섹션의 aggregation_dimensions 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.
  - resources – 선택 사항입니다. 디스크 탑재 지점에 대한 배열을 지정합니다. 이 필드는 CloudWatch가 나열된 탑재 지점에서만 지표를 수집하도록 제한합니다. 값으로 *를 지정하면 모든 탑재 지점에서 지표를 수집할 수 있습니다. 기본값은 모든 탑재 지점에서 지표를 수집하는 것입니다.

- `measurement` – 수집할 디스크 지표의 배열을 지정합니다. 가능한 값은 `free`, `total`, `used`, `used_percent`, `inodes_free`, `inodes_used`, `inodes_total`입니다. `disk`를 포함하는 경우 이 필드는 필수입니다.

  > **Note**
  >
  > `disk` 지표에는 `Partition`의 측정기준이 있는데, 이는 생성된 사용자 지정 지표의 수가 인스턴스와 연결된 파티션의 수에 따라 달라짐을 뜻합니다. 보유한 디스크 파티션의 수는 사용 중인 AMI 및 서버에 연결하는 Amazon EBS 볼륨의 수에 따라 달라집니다.

  각 `disk` 지표에 대한 기본 단위를 보려면 Linux 및 macOS 인스턴스의 CloudWatch 에이전트가 수집하는 지표 섹션을 참조하세요.

  각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.

  - `rename` – 이 지표에 대해 다른 이름을 지정합니다.
  - `unit` – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None의 None)를 재정의합니다. 지정하는 단위는 MetricDatum의 `Unit` 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.

- `ignore_file_system_types` – 디스크 지표를 수집할 때 제외할 파일 시스템 유형을 지정합니다. 유효한 값으로는 `sysfs`, `devtmpfs` 등이 있습니다.
- `drop_device` – 이 값을 `true`로 설정하면 `Device`가 디스크 지표의 측정기준으로 포함되지 않습니다.

  `Device`가 측정기준으로 사용되지 않도록 하면 Nitro 시스템을 사용하는 인스턴스에서 인스턴스가 재부팅될 때 각 디스크 마운트마다 디바이스 이름이 변경되기 때문에 유용합니다. 이 경우 지표의 데이터가 일관되지 않으므로 이러한 지표를 기반으로 하는 경보가 `INSUFFICIENT DATA` 상태가 될 수 있습니다.

  기본값은 `false`입니다.

- `metrics_collection_interval` – 선택 사항입니다. 디스크 지표를 수집하여 구성 파일의 `agent` 섹션에 지정되어 있는 글로벌 `metrics_collection_interval` 값을 재정의할 빈도를 지정합니다.

  이 값은 초 단위로 지정됩니다.

이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 자세한 내용은 고분해능 지표 단원을 참조하세요.

- append_dimensions – 선택 사항입니다. 디스크 지표에만 사용할 수 있는 추가 측정기준입니다. 이 필드를 지정하면 에이전트에 의해 수집되는 모든 유형의 지표에 대해 사용되는 append_dimensions 필드에 지정되어 있는 측정기준 외에 지정한 측정기준도 사용됩니다.

- diskio – 선택 사항입니다. 디스크 I/O 지표가 수집되도록 지정합니다. 이 단원은 리눅스 인스턴스에만 유효합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.
  - drop_original_metrics – 선택 사항입니다. metrics 섹션의 aggregation_dimensions 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.
  - resources – 선택 사항입니다. 디바이스 배열을 지정하면 CloudWatch가 해당 디바이스에서만 지표를 수집합니다. 그렇지 않으면, 모든 디바이스의 지표가 수집됩니다. 값으로 *를 지정하여 모든 디바이스에서 지표를 수집할 수도 있습니다.
  - measurement – 수집할 diskio 지표의 배열을 지정합니다. 가능한 값은 reads, writes, read_bytes, write_bytes, read_time, write_time, io_time, iops_in_progress입니다. diskio를 포함하는 경우 이 필드는 필수입니다.

    각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.
    - rename – 이 지표에 대해 다른 이름을 지정합니다.
    - unit – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None의 None)를 재정의합니다. 지정하는 단위는 MetricDatum의 Unit 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.
  - metrics_collection_interval – 선택 사항입니다. diskio 지표를 수집하여 구성 파일의 agent 섹션에 지정되어 있는 글로벌 metrics_collection_interval 값을 재정의할 빈도를 지정합니다.

    이 값은 초 단위로 지정됩니다.

    이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 고분해능 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하세요.

  - append_dimensions – 선택 사항입니다. diskio 지표에만 사용할 수 있는 추가 측정기준입니다. 이 필드를 지정하면 에이전트에 의해 수집되는 모든 유형의 지표에 대해 사용되는 append_dimensions 필드에 지정되어 있는 측정기준 외에 지정한 측정기준도 사용됩니다.
- swap – 선택 사항입니다. 스왑 메모리 지표가 수집됨을 나타냅니다. 이 단원은 리눅스 인스턴스에만 유효합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.
  - drop_original_metrics – 선택 사항입니다. metrics 섹션의 aggregation_dimensions 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.
  - measurement – 수집할 swap 지표의 배열을 지정합니다. 가능한 값은 free, used 및 used_percent입니다. swap를 포함하는 경우 이 필드는 필수입니다.

    각 swap 지표에 대한 기본 단위를 보려면 Linux 및 macOS 인스턴스의 CloudWatch 에이전트가 수집하는 지표 섹션을 참조하세요.

    각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.

    - rename – 이 지표에 대해 다른 이름을 지정합니다.
    - unit – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None의 None)를 재정의합니다. 지정하는 단위는 MetricDatum의 Unit 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.
  - metrics_collection_interval – 선택 사항입니다. 스왑 지표를 수집하여 구성 파일의 agent 섹션에 지정되어 있는 글로벌 metrics_collection_interval 값을 재정의할 빈도를 지정합니다.

    이 값은 초 단위로 지정됩니다.

    이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 고분해능 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하세요.
  - append_dimensions – 선택 사항입니다. 스왑 지표에만 사용할 수 있는 추가 측정기준입니다. 이 필드를 지정하면 에이전트에 의해 수집되는 모든 유형의 지표에 대해 사용되는 글로벌 append_dimensions 필드에 지정되어 있는 측정기준 외에 지정한 측정기준도 사용됩니다. 고해상도 지표로 수집됩니다.
- mem – 선택 사항입니다. 메모리 지표가 수집됨을 나타냅니다. 이 단원은 리눅스 인스턴스에만 유효합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.

- drop_original_metrics – 선택 사항입니다. metrics 섹션의 aggregation_dimensions 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.
- measurement – 수집할 메모리 지표의 배열을 지정합니다. 가능한 값은 active, available, available_percent, buffered, cached, free, inactive, total, used, used_percent입니다. mem를 포함하는 경우 이 필드는 필수입니다.

  각 mem 지표에 대한 기본 단위를 보려면 Linux 및 macOS 인스턴스의 CloudWatch 에이전트가 수집하는 지표 섹션을 참조하세요.

  각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.

  - rename – 이 지표에 대해 다른 이름을 지정합니다.
  - unit – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None)를 재정의합니다. 지정하는 단위는 MetricDatum의 Unit 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.
- metrics_collection_interval – 선택 사항입니다. mem 지표를 수집하여 구성 파일의 agent 섹션에 지정되어 있는 글로벌 metrics_collection_interval 값을 재정의할 빈도를 지정합니다.

  이 값은 초 단위로 지정됩니다.

  이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 고분해능 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하세요.
- append_dimensions – 선택 사항입니다. mem 지표에만 사용할 수 있는 추가 측정기준입니다. 이 필드를 지정하면 에이전트가 수집하는 모든 유형의 지표에 대해 사용되는 append_dimensions 필드에 지정되어 있는 측정기준 외에 이 측정기준도 사용됩니다.

- net – 선택 사항입니다. 네트워킹 지표가 수집됨을 나타냅니다. 이 단원은 리눅스 인스턴스에만 유효합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.
  - drop_original_metrics – 선택 사항입니다. metrics 섹션의 aggregation_dimensions 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지

표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.

- `resources` – 선택 사항입니다. 네트워크 인터페이스 배열을 지정하면 CloudWatch가 해당 인터페이스에서만 지표를 수집합니다. 그렇지 않으면, 모든 디바이스의 지표가 수집됩니다. 또한 값으로 *를 지정하면 모든 인터페이스에서 지표를 수집할 수 있습니다.
- `measurement` – 수집할 네트워킹 지표의 배열을 지정합니다. 가능한 값은 `bytes_sent`, `bytes_recv`, `drop_in`, `drop_out`, `err_in`, `err_out`, `packets_sent`, `packets_recv`입니다. `net`를 포함하는 경우 이 필드는 필수입니다.

  각 `net` 지표에 대한 기본 단위를 보려면 Linux 및 macOS 인스턴스의 CloudWatch 에이전트가 수집하는 지표 섹션을 참조하세요.

  각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.

  - `rename` – 이 지표에 대해 다른 이름을 지정합니다.
  - `unit` – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None)를 재정의합니다. 지정하는 단위는 MetricDatum의 `Unit` 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.
- `metrics_collection_interval` – 선택 사항입니다. net 지표를 수집하여 구성 파일의 `agent` 섹션에 지정되어 있는 글로벌 `metrics_collection_interval` 값을 재정의할 빈도를 지정합니다.

  이 값은 초 단위로 지정됩니다. 예를 들어, 10을 지정하면 10초마다 지표가 수집되도록 설정되며, 300으로 설정하면 5분마다 지표가 수집되도록 지정됩니다.

  이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 고분해능 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하세요.
- `append_dimensions` – 선택 사항입니다. net 지표에만 사용할 수 있는 추가 측정기준입니다. 이 필드를 지정하면 에이전트에 의해 수집되는 모든 유형의 지표에 대해 사용되는 `append_dimensions` 필드에 지정되어 있는 측정기준 외에 이 측정기준도 사용됩니다.

- `netstat` – 선택 사항입니다. TCP 연결 상태 및 UDP 연결 지표가 수집됨을 나타냅니다. 이 단원은 리눅스 인스턴스에만 유효합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.
  - `drop_original_metrics` – 선택 사항입니다. `metrics` 섹션의 `aggregation_dimensions` 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지

표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.

- `measurement` – 수집할 netstat 지표의 배열을 지정합니다. 가능한 값은 `tcp_close`, `tcp_close_wait`, `tcp_closing`, `tcp_established`, `tcp_fin_wait1`, `tcp_fin_wait2`, `tcp_last_ack`, `tcp_listen`, `tcp_none`, `tcp_syn_sent`, `tcp_syn_recv`, `tcp_time_wait` 및 `udp_socket`입니다. `netstat`를 포함하는 경우 이 필드는 필수입니다.

  각 `netstat` 지표에 대한 기본 단위를 보려면 Linux 및 macOS 인스턴스의 CloudWatch 에이전트가 수집하는 지표 섹션을 참조하세요.

  각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.

  - `rename` – 이 지표에 대해 다른 이름을 지정합니다.
  - `unit` – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None)를 재정의합니다. 지정하는 단위는 MetricDatum의 `Unit` 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.

- `metrics_collection_interval` – 선택 사항입니다. netstat 지표를 수집하여 구성 파일의 `agent` 섹션에 지정되어 있는 글로벌 `metrics_collection_interval` 값을 재정의할 빈도를 지정합니다.

  이 값은 초 단위로 지정됩니다.

  이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 고분해능 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하세요.

- `append_dimensions` – 선택 사항입니다. netstat 지표에만 사용할 수 있는 추가 측정기준입니다. 이 필드를 지정하면 에이전트에 의해 수집되는 모든 유형의 지표에 대해 사용되는 `append_dimensions` 필드에 지정되어 있는 측정기준 외에 이 측정기준도 사용됩니다.

- `processes` – 선택 사항입니다. 프로세스 지표가 수집됨을 나타냅니다. 이 단원은 리눅스 인스턴스에만 유효합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.

  - `drop_original_metrics` – 선택 사항입니다. `metrics` 섹션의 `aggregation_dimensions` 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.

- `measurement` – 수집할 프로세스 지표의 배열을 지정합니다. 가능한 값은 `blocked`, `dead`, `idle`, `paging`, `running`, `sleeping`, `stopped`, `total`, `total_threads`, `wait`, `zombies`입니다. `processes`를 포함하는 경우 이 필드는 필수입니다.

  모든 `processes` 지표에 대해 기본 단위는 `None`입니다.

  각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.

  - `rename` – 이 지표에 대해 다른 이름을 지정합니다.
  - `unit` – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None)를 재정의합니다. 지정하는 단위는 MetricDatum의 `Unit` 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.

- `metrics_collection_interval` – 선택 사항입니다. 프로세스 지표를 수집하여 구성 파일의 `agent` 섹션에 지정되어 있는 글로벌 `metrics_collection_interval` 값을 재정의할 빈도를 지정합니다.

  이 값은 초 단위로 지정됩니다. 예를 들어, 10을 지정하면 10초마다 지표가 수집되도록 설정되며, 300으로 설정하면 5분마다 지표가 수집되도록 지정됩니다.

  이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 자세한 내용은 고분해능 지표 단원을 참조하세요.

- `append_dimensions` – 선택 사항입니다. 프로세스 지표에만 사용할 수 있는 추가 측정기준입니다. 이 필드를 지정하면 에이전트에 의해 수집되는 모든 유형의 지표에 대해 사용되는 `append_dimensions` 필드에 지정되어 있는 측정기준 외에 이 측정기준도 사용됩니다.

- `nvidia_gpu` – 선택 사항입니다. NVIDIA GPU 지표가 수집되도록 지정합니다. 이 섹션은 NVIDIA GPU 가속기로 구성되고 NVIDIA 시스템 관리 인터페이스(nvidia-smi)가 설치된 호스트의 Linux 인스턴스에 대해서만 유효합니다.

  수집된 NVIDIA GPU 지표에는 `nvidia_smi_` 문자열이 앞에 추가되어 다른 액셀러레이터 유형에 대해 수집된 지표와 구분할 수 있습니다. 이 섹션은 다음 필드를 포함할 수 있습니다.

  - `drop_original_metrics` – 선택 사항입니다. `metrics` 섹션의 `aggregation_dimensions` 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.

  - `measurement` – 수집할 NVIDIA GPU 지표의 배열을 지정합니다. 여기에서 사용할 수 있는 값 목록은 NVIDIA GPU 지표 수집의 테이블에서 지표(Metric) 열을 참조하세요.

    각 개별 지표의 항목 내에서 다음 중 하나 또는 둘 다를 선택적으로 지정할 수 있습니다.

    - `rename` – 이 지표에 대해 다른 이름을 지정합니다.
    - `unit` – 이 지표에 사용할 단위를 지정하여 지표에 대한 기본 단위(None)를 재정의합니다. 지정하는 단위는 MetricDatum의 `Unit` 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.
  - `metrics_collection_interval` – 선택 사항입니다. NVIDIA GPU 지표를 수집하여 구성 파일의 `agent` 섹션에 지정되어 있는 글로벌 `metrics_collection_interval` 값을 재정의할 빈도를 지정합니다.
- `procstat` – 선택 사항입니다. 개별 프로세스에서 지표를 검색하려 한다고 지정합니다. procstat에 사용할 수 있는 구성 옵션에 대한 자세한 내용은 procstat 플러그 인을 사용하여 프로세스 지표 수집 단원을 참조하세요.
- `statsd` – 선택 사항입니다. `StatsD` 프로토콜을 사용하여 사용자 지정 지표를 검색하려 한다는 것을 지정합니다. CloudWatch 에이전트는 프로토콜의 데몬 역할을 합니다. 표준 `StatsD` 클라이언트를 사용하여 지표를 CloudWatch 에이전트에 전송합니다. StatsD에 사용할 수 있는 구성 옵션에 대한 자세한 내용은 StatsD를 사용하여 사용자 지정 지표 검색 단원을 참조하세요.
- `ethtool` – 선택 사항입니다. `ethtool` 플러그 인을 사용하여 네트워크 지표를 검색하도록 지정합니다. 이 플러그 인은 표준 ethtool 유틸리티가 수집한 지표와 Amazon EC2 인스턴스의 네트워크 성능 지표를 모두 가져올 수 있습니다. ethtool에 사용할 수 있는 구성 옵션에 대한 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요.

다음은 Linux 서버용 `metrics` 섹션의 예입니다. 이 예에서는 3개의 CPU 지표, 3개의 netstat 지표, 3개의 프로세스 지표 및 1개의 디스크 지표가 수집되며, 에이전트는 `collectd` 클라이언트로부터 추가 지표를 받도록 설정됩니다.

```
"metrics": {
    "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId",
 "InstanceType"],[]],
    "metrics_collected": {
      "collectd": {},
      "cpu": {
        "resources": [
          "*"
        ],
        "measurement": [
```

```
          {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent"},
          {"name": "cpu_usage_nice", "unit": "Percent"},
          "cpu_usage_guest"
        ],
        "totalcpu": false,
        "drop_original_metrics": [ "cpu_usage_guest" ],
        "metrics_collection_interval": 10,
        "append_dimensions": {
          "test": "test1",
          "date": "2017-10-01"
        }
      },
      "netstat": {
        "measurement": [
          "tcp_established",
          "tcp_syn_sent",
          "tcp_close"
        ],
        "metrics_collection_interval": 60
      },
       "disk": {
        "measurement": [
          "used_percent"
        ],
        "resources": [
          "*"
        ],
        "drop_device": true
      },
      "processes": {
        "measurement": [
          "running",
          "sleeping",
          "dead"
        ]
      }
    },
    "append_dimensions": {
      "ImageId": "${aws:ImageId}",
      "InstanceId": "${aws:InstanceId}",
      "InstanceType": "${aws:InstanceType}",
      "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
    }
```

```
  }
```

Windows Server

Windows Server의 `metrics_collected` 섹션에서는 `Memory`, `Processor` 및 `LogicalDisk`와 같은 각 Windows 성능 객체에 대한 하위 섹션을 생성할 수 있습니다. 사용 가능한 개체와 카운터에 대한 자세한 내용은 Microsoft Windows 설명서의 Performance Counters를 참조하세요.

각 객체의 하위 섹션 내에서 수집할 카운터의 `measurement` 어레이를 지정합니다. `measurement` 어레이는 구성 파일에서 지정하는 객체마다 있어야 합니다. 또한 `resources` 필드를 지정하여 지표를 수집할 인스턴스의 이름을 지정할 수 있습니다. 또한 `resources`에 대해 `*`를 지정하면 모든 인스턴스에 대해 별도의 지표를 수집할 수 있습니다. 인스턴스가 있는 카운터에 대해 `resources`를 생략하면 모든 인스턴스의 데이터가 하나의 세트로 집계됩니다. 인스턴스가 없는 카운터에 대해 `resources`를 생략하면 CloudWatch 에이전트가 카운터를 수집하지 않습니다. 카운터에 인스턴스가 있는지 확인하려면 다음 명령 중 하나를 사용할 수 있습니다.

Powershell:

```
Get-Counter -ListSet *
```

명령줄(Powershell 아님):

```
TypePerf.exe -q
```

각 객체 섹션 내에서 다음 선택 필드를 지정할 수도 있습니다.

- `metrics_collection_interval` – 선택 사항입니다. 이 객체에 대해 지표를 수집하여 구성 파일의 `agent` 섹션에 지정되어 있는 글로벌 `metrics_collection_interval` 값을 재정의할 빈도를 지정합니다.

  이 값은 초 단위로 지정됩니다. 예를 들어, 10을 지정하면 10초마다 지표가 수집되도록 설정되며, 300으로 설정하면 5분마다 지표가 수집되도록 지정됩니다.

  이 값을 60초 미만으로 설정하면 각 지표가 고분해능 지표로 수집됩니다. 자세한 내용은 고분해능 지표 단원을 참조하세요.
- `append_dimensions` – 선택 사항입니다. 이 객체의 지표에만 사용할 추가 측정기준을 지정합니다. 이 필드를 지정하면 에이전트에 의해 수집되는 모든 유형의 지표에 대해 사용되는 글로벌 `append_dimensions` 필드에 지정되어 있는 측정기준 외에 이 측정기준도 사용됩니다.

- `drop_original_metrics` – 선택 사항입니다. `metrics` 섹션의 `aggregation_dimensions` 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정 기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.

각 카운터 섹션 내에서 다음 선택 필드를 지정할 수도 있습니다.

- `rename` – CloudWatch에서 이 지표에 사용할 다른 이름을 지정합니다.
- `unit` – 이 지표에 사용할 단위를 지정합니다. 지정하는 단위는 MetricDatum의 `Unit` 설명에 나열된 대로 유효한 CloudWatch 지표 단위여야 합니다.

`metrics_collected`에는 다음과 같이 두 가지 다른 선택적 섹션을 포함할 수 있습니다.

- `statsd` – `StatsD` 프로토콜을 사용하여 사용자 지정 지표를 검색할 수 있도록 합니다. CloudWatch 에이전트는 프로토콜의 데몬 역할을 합니다. 표준 `StatsD` 클라이언트를 사용하여 지표를 CloudWatch 에이전트에 전송합니다. 자세한 내용은 StatsD를 사용하여 사용자 지정 지표 검색 단원을 참조하세요.
- `procstat` – 개별 프로세스에서 지표를 검색할 수 있도록 합니다. 자세한 내용은 procstat 플러그 인을 사용하여 프로세스 지표 수집 단원을 참조하세요.

다음은 Windows Server에서 사용하기 위한 `metrics` 섹션의 예입니다. 이 예에서는 다양한 Windows 지표가 수집되며, 컴퓨터가 `StatsD` 클라이언트로부터 추가 지표를 받도록 설정되기도 합니다.

```
"metrics": {
    "metrics_collected": {
      "statsd": {},
      "Processor": {
        "measurement": [
          {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
          "% Interrupt Time",
          "% User Time",
          "% Processor Time"
        ],
        "resources": [
          "*"
        ],
```

```
        "append_dimensions": {
          "d1": "win_foo",
          "d2": "win_bar"
        }
      },
      "LogicalDisk": {
        "measurement": [
          {"name": "% Idle Time", "unit": "Percent"},
          {"name": "% Disk Read Time", "rename": "DISK_READ"},
          "% Disk Write Time"
        ],
        "resources": [
          "*"
        ]
      },
      "Memory": {
        "metrics_collection_interval": 5,
        "measurement": [
          "Available Bytes",
          "Cache Faults/sec",
          "Page Faults/sec",
          "Pages/sec"
        ],
        "append_dimensions": {
          "d3": "win_bo"
        }
      },
      "Network Interface": {
        "metrics_collection_interval": 5,
        "measurement": [
          "Bytes Received/sec",
          "Bytes Sent/sec",
          "Packets Received/sec",
          "Packets Sent/sec"
        ],
        "resources": [
          "*"
        ],
        "append_dimensions": {
          "d3": "win_bo"
        }
      },
      "System": {
        "measurement": [
```

```
          "Context Switches/sec",
          "System Calls/sec",
          "Processor Queue Length"
        ],
        "append_dimensions": {
          "d1": "win_foo",
          "d2": "win_bar"
        }
      }
    },
    "append_dimensions": {
      "ImageId": "${aws:ImageId}",
      "InstanceId": "${aws:InstanceId}",
      "InstanceType": "${aws:InstanceType}",
      "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
    },
    "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"],[]]
    }
  }
```

CloudWatch 에이전트 구성 파일: 로그 섹션

`logs` 섹션에 포함되는 필드는 다음과 같습니다.

- `logs_collected` – `logs` 섹션이 포함되어 있는 경우 필수입니다. 서버로부터 수집될 로그 파일 및 Windows 서버 로그를 지정합니다. 두 필드인 `files` 및 `windows_events`를 포함할 수 있습니다.
  - `files` – CloudWatch 에이전트가 수집할 일반 로그 파일을 지정합니다. 여기에는 `collect_list`라는 필드 하나가 포함되는데, 이 필드는 이러한 파일을 추가로 정의합니다.
    - `collect_list` – `files`가 포함되어 있는 경우 필수입니다. 항목 어레이를 포함하며, 항목 각각이 수집될 하나의 로그 파일을 지정합니다. 이러한 항목 각각에는 다음 필드를 포함할 수 있습니다.
      - `file_path` – CloudWatch Logs에 업로드할 로그 파일의 경로를 지정합니다. 표준 Unix 글로브 일치 규칙이 허용되며 **를 '슈퍼 별표'로 추가합니다. 예를 들어 `/var/log/**.log`를 지정하면 `/var/log` 디렉터리 트리의 모든 `.log` 파일이 수집됩니다. 보다 자세한 예는 Glob Library를 참조하세요.

        표준 별표를 표준 와일드카드로 사용할 수도 있습니다. 예를 들어, `/var/log/system.log*`는 `/var/log`에서 `system.log_1111`, `system.log_2222` 등의 파일과 일치합니다.

파일 수정 시간에 따라 최신 파일만 CloudWatch Logs에 푸시됩니다. 와일드카드는 여러 종류의 파일(예: access_log_80 및 access_log_443)이 아니라 종류가 같은 일련의 파일(예: access_log.2018-06-01-01 및 access_log.2018-06-01-02)을 지정할 때 사용하는 것이 좋습니다. 여러 종류의 파일을 지정하려면 로그 파일이 종류별로 서로 다른 로그 스트림에 들어가도록 에이전트 구성 파일에 다른 로그 스트림 항목 하나를 추가합니다.

- auto_removal – 선택 사항입니다. true인 경우 CloudWatch 에이전트는 이 로그 파일을 읽은 후 자동으로 삭제하며 이 파일은 순환됩니다. 일반적으로 로그 파일은 전체 내용이 CloudWatch Logs에 업로드된 후 삭제되지만, 에이전트가 EOF(파일 끝)에 도달한 후 동일한 file_path와 일치하는 다른 최신 로그 파일도 탐지하면 이전 파일을 삭제하므로 새 파일을 생성하기 전에 이전 파일 쓰기를 완료해야 합니다. RUST 추적 라이브러리는 잠재적으로 새 로그 파일을 만든 다음에도 계속해서 이전 로그 파일에 쓰기를 시도하기 때문에 호환되지 않는다고 알려져 있습니다.

  에이전트는 각 날짜에 대해 별도의 파일을 만드는 로그와 같이 여러 파일을 생성하는 로그에서만 전체 파일을 제거합니다. 로그가 단일 파일에 연속적으로 기록되는 경우에는 제거되지 않습니다.

  로그 파일 순환 또는 제거 방법이 이미 있는 경우 이 필드를 생략하거나 false로 설정하는 것이 좋습니다.

  이 필드를 생략하면 기본값인 false가 사용됩니다.

- log_group_name – 선택 사항입니다. CloudWatch Logs에서 로그 그룹 이름으로 사용할 이름을 지정합니다.

  혼동을 방지하기 위해 이 필드를 사용하여 로그 그룹 이름을 지정하는 것이 좋습니다. log_group_name을 생략하면 마지막 점까지의 file_path 값이 로그 그룹 이름으로 사용됩니다. 예를 들어 파일 경로가 /tmp/TestLogFile.log.2017-07-11-14이면 로그 그룹 이름은 /tmp/TestLogFile.log입니다.

  로그 그룹 이름을 지정할 경우 {instance_id}, {hostname}, {local_hostname}, {ip_address}를 이름 안의 변수로 사용할 수 있습니다. {hostname}은 EC2 메타데이터에서 호스트 이름을 가져오고, {local_hostname}은 네트워크 구성 파일에 있는 호스트 이름을 사용합니다.

  이러한 변수를 사용하여 많은 로그 그룹을 생성하는 경우 계정별 리전당 1,000,000개 로그 그룹이 한도라는 점에 유의하세요.

  허용되는 문자에는 a–z, A–Z, 0–9, '_'(밑줄), '-'(하이픈), '/'(슬래시), '.'(마침표)가 포함됩니다.

- `log_group_class` – 선택 사항입니다. 새 로그 그룹에 사용할 로그 그룹 클래스를 지정합니다. 로그 그룹 클래스에 대한 자세한 내용은 로그 클래스를 참조하세요.

  유효 값은 `STANDARD`및 `INFREQUENT_ACCESS`입니다. 이 필드를 생략하면 기본값인 `STANDARD`가 사용됩니다.

  > **Important**
  >
  > 로그 그룹이 생성된 후에는 해당 클래스를 변경할 수 없습니다.

- `log_stream_name` – 선택 사항입니다. CloudWatch Logs에서 로그 스트림 이름으로 사용할 이름을 지정합니다. 이름의 일부로 `{instance_id}`, `{hostname}`, `{local_hostname}`, `{ip_address}`를 이름 안의 변수로 사용할 수 있습니다. `{hostname}`은 EC2 메타데이터에서 호스트 이름을 가져오고, `{local_hostname}`은 네트워크 구성 파일에 있는 호스트 이름을 사용합니다.

  이 필드를 생략하면 글로벌 `logs` 섹션의 `log_stream_name` 파라미터 값이 사용됩니다. 이 값도 생략하면 기본값 `{instance_id}`가 사용됩니다.

  로그 스트림이 아직 없는 경우 로그 스트림이 자동으로 생성됩니다.

- `retention_in_days` – 선택 사항입니다. 지정된 로그 그룹에 로그 이벤트를 보관하는 일수를 지정합니다.
  - 에이전트가 지금 이 로그 그룹을 생성하고 있는 경우 이 필드를 생략하면 이 새 로그 그룹의 보존 기간이 절대 만료되지 않도록 설정됩니다.
  - 이 로그 그룹이 이미 존재하는 경우 이 필드를 지정하면 지정한 새 보존 기간이 사용됩니다. 이미 존재하는 로그 그룹에 대해 이 필드를 생략하면 로그 그룹의 보존 기간이 변경되지 않습니다.

    CloudWatch 에이전트 마법사는 에이전트 구성 파일을 생성하는 데 사용되고 로그 보존 값을 지정하지 않은 경우 이 필드의 기본값으로 `-1`을 사용합니다. 마법사가 설정한 `-1` 값은 로그 그룹의 이벤트가 절대로 만료되지 않도록 지정합니다. 그러나 이 값을 `-1`로 수동으로 편집해도 아무런 효과가 없습니다.

  유효한 값은 1, 3, 5, 7, 14, 30, 60, 90, 120, 150, 180, 365, 400, 545, 731, 1827, 2192, 2557, 2922, 3288 및 3653입니다.

  여러 로그 스트림을 동일한 로그 그룹에 쓰도록 에이전트를 구성한 경우 한 곳에서 `retention_in_days` 값을 지정하면 전체 로그 그룹에 대한 로그 보존이 설정됩니다. 여러

위치에 있는 동일한 로그 그룹에 대해 `retention_in_days` 값을 지정하는 경우 이러한 값이 모두 동일하면 보존이 설정됩니다. 그러나 여러 위치에서 동일한 로그 그룹에 대해 다른 `retention_in_days` 값을 지정하는 경우 로그 보존이 설정되지 않고 에이전트가 중지되어 오류가 반환됩니다.

> **Note**
>
> 에이전트의 IAM 역할 또는 IAM 사용자에게는 `logs:PutRetentionPolicy` 값이 있어야 보존 정책을 설정할 수 있습니다. 자세한 내용은 CloudWatch 에이전트가 로그 보존 정책을 설정하도록 허용 단원을 참조하십시오.

> **Warning**
>
> 이미 존재하는 로그 그룹에 대해 `retention_in_days` 값을 설정한 경우 지정한 일수 전에 게시된 해당 로그 그룹의 모든 로그가 삭제됩니다. 예를 들어, 3으로 설정하면 3일 전과 그 이전의 모든 로그가 삭제됩니다.

- `filters` – 선택 사항입니다. 항목의 배열을 포함할 수 있으며, 각 항목은 정규 표현식과 필터 유형을 지정하여 필터와 일치하는 로그 항목을 게시할지 삭제할지 여부를 지정합니다. 이 필드를 생략하면 로그 파일의 모든 로그가 CloudWatch Logs에 게시됩니다. 이 필드를 포함하는 경우 에이전트는 사용자가 지정한 모든 필터를 통해 각 로그 메시지를 처리하고 모든 필터를 전달하는 로그 이벤트만 CloudWatch Logs에 게시됩니다. 모든 필터를 전달하지 않는 로그 항목은 여전히 호스트의 로그 파일에 남아 있지만, CloudWatch Logs로 전송되지는 않습니다.

  필터 배열의 각 항목에는 다음 필드가 포함될 수 있습니다.

  - `type` - 필터 유형을 표시합니다. 유효 값은 `include`및 `exclude`입니다. `include` 값을 선택하면 로그 항목이 CloudWatch Logs에 게시할 표현식과 일치해야 합니다. `exclude` 값을 선택하면 필터와 일치하는 각 로그 항목이 CloudWatch Logs로 전송되지 않습니다.
  - `expression` - RE2 구문 다음에 나오는 정규 표현식 문자열입니다.

    > **Note**
    >
    > CloudWatch 에이전트는 사용자가 제공하는 정규 표현식의 성능을 확인하거나 정규 표현식 평가의 실행 시간을 제한하지 않습니다. 평가할 때 비용이 많이 드는 표

현식을 작성하지 않도록 주의하는 것이 좋습니다. 가능한 문제에 대한 자세한 내용은 정규 표현식 서비스 거부 - ReDOS를 참조하세요.

예를 들어, CloudWatch 에이전트 구성 파일의 다음 발췌문은 PUT 및 POST 요청인 로그를 CloudWatch Logs 로그에 게시하지만, Firefox에서 오는 로그는 제외합니다.

```
"collect_list": [
  {
    "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
    "log_group_name": "test.log",
    "log_stream_name": "test.log",
    "filters": [
      {
        "type": "exclude",
        "expression": "Firefox"
      },
      {
        "type": "include",
        "expression": "P(UT|OST)"
      }
    ]
  },
  .....
]
```

> **Note**
>
> 구성 파일의 필터 순서는 성능에 영향을 미칩니다. 이전 예제에서 에이전트는 두 번째 필터 평가를 시작하기 전에 `Firefox`와 일치하는 모든 로그를 삭제합니다. 둘 이상의 필터에 의해 평가되는 로그 항목 수를 줄이려면 먼저 더 많은 로그를 제외할 것으로 예상되는 필터를 구성 파일에 넣습니다.

- `timezone` – 선택 사항입니다. 로그 이벤트에 타임스탬프를 표시할 때 사용할 시간대를 지정합니다. 유효 값은 `UTC`와 `Local`입니다. 기본 값은 `Local`입니다.

  `timestamp_format`에 값을 지정하지 않으면 이 파라미터는 무시됩니다.

- `timestamp_format` – 선택 사항입니다. 일반 텍스트와 %로 시작하는 특수 기호를 사용하여 타임스탬프 형식을 지정합니다. 이 필드를 생략한 경우 현재 시간이 사용됩니다. 이 필드를 사용하면 다음 목록에 있는 기호를 형식의 일부로 사용할 수 있습니다.

단일 로그 항목에 형식과 일치하는 두 개의 타임스탬프가 포함된 경우 첫 번째 타임스탬프가 사용됩니다.

이 기호 목록은 이전 CloudWatch Logs 에이전트에서 사용되는 목록과 다릅니다. 이러한 차이점에 대한 요약은 통합 CloudWatch 에이전트와 이전 CloudWatch Logs 에이전트 간의 타임스탬프 차이 단원을 참조하세요.

`%y`

제로 패딩된 10진수 형태의 세기를 제외한 연도 예를 들어 19는 2019년을 나타냅니다.

`%Y`

10진수 형태로 세기가 포함된 연도 예를 들면 2019입니다.

`%b`

로캘 약어 형태의 월

`%B`

로캘 전체 이름 형태의 월

`%m`

제로 패딩된 10진수 형태의 월

`%-m`

10진수 형태의 월(제로 패딩되지 않음)

`%d`

제로 패딩된 10진수 형태의 월 날짜

`%-d`

10진수 형태의 월 날짜(제로 패딩되지 않음)

`%A`

평일의 전체 이름(예: Monday)

`%a`

평일의 약어(예: Mon)

`%H`

제로 패딩된 10진수 형태의 시간(24시간 방식)

`%I`

제로 패딩된 10진수 형태의 시간(12시간 방식)

`%-I`

10진수 형태의 시간(12시간 방식, 제로 패딩되지 않음)

`%p`

AM(오전) 또는 PM(오후)

`%M`

제로 패딩된 10진수 형태의 분

`%-M`

10진수 형태의 분(제로 패딩되지 않음)

`%S`

제로 패딩된 10진수 형태의 초

`%-S`

10진수 형태의 초(제로 패딩되지 않음)

`%f`

10진수 형태(1-9의 숫자)의 소수부 초(왼쪽에 제로 패딩됨)

`%Z`

시간대(예: PST)

`%z`

현지 시간대와 UTC 간의 오프셋으로 표시되는 시간대입니다. 예를 들면 `-0700`입니다. 이 형식만 지원됩니다. 예를 들어, `-07:00`은 유효한 형식이 아닙니다.

- `multi_line_start_pattern` – 로그 메시지의 시작을 식별하기 위한 패턴을 지정합니다. 로그 메시지는 패턴과 일치하는 하나의 줄 및 패턴과 일치하지 않는 후속 줄로 이루어져 있습니다.

이 필드를 생략할 경우, 여러 줄 모드가 비활성화되고 기본적으로 공백이 아닌 문자로 시작되는 줄에서 이전의 로그 메시지가 종료되고 새로운 로그 메시지가 시작됩니다.

이 필드를 포함하는 경우, 타임스탬프 형식과 동일한 정규식을 사용하도록 `{timestamp_format}`을 지정할 수 있습니다. 그렇지 않으면 CloudWatch Logs가 여러 줄 항목의 시작 줄을 결정하는 데 사용할 다른 정규 표현식을 지정할 수 있습니다.

- `encoding` – 로그 파일을 정확하게 읽을 수 있도록 로그 파일의 인코딩을 지정합니다. 코딩을 잘못 지정하면 디코딩할 수 없는 문자를 다른 문자가 대체하기 때문에 데이터가 손실될 수 있습니다.

  기본 값은 `utf-8`입니다. 가능한 모든 값은 다음과 같습니다.

  ```
  ascii, big5, euc-jp, euc-kr, gbk, gb18030, ibm866, iso2022-jp,
  iso8859-2, iso8859-3, iso8859-4, iso8859-5, iso8859-6, iso8859-7,
  iso8859-8, iso8859-8-i, iso8859-10, iso8859-13, iso8859-14,
  iso8859-15, iso8859-16, koi8-r, koi8-u, macintosh, shift_jis, utf-8,
  utf-16, utf-16le, UTF-16, UTF-16LE, windows-874, windows-1250,
  windows-1251, windows-1252, windows-1253, windows-1254,
  windows-1255, windows-1256, windows-1257, windows-1258, x-mac-
  cyrillic
  ```

- `windows_events` 섹션은 Windows Server가 실행되는 서버로부터 수집할 Windows 이벤트 유형을 지정합니다. 여기에는 다음 필드가 포함됩니다.
  - `collect_list` – `windows_events`가 포함되어 있는 경우 필수입니다. 수집될 Windows 이벤트의 유형 및 수준을 지정합니다. 수집될 각 로그에는 이 섹션에 있는 항목이 있으며, 여기에는 다음 필드를 포함할 수 있습니다.
    - `event_name` – 로그할 Windows 이벤트 유형을 지정합니다. 이 유형은 Windows 이벤트 로그 채널 이름과 동일합니다(예: `System`, `Security`, `Application` 등). 이 필드는 기록할 Windows 이벤트 유형마다 있어야 합니다.

      > **Note**
      >
      > CloudWatch는 Windows 로그 채널에서 메시지를 검색할 때 `Full Name` 속성을 기반으로 로그 채널을 조회합니다. 한편, Windows 이벤트 뷰어 탐색 창에는 로그 채널의 `Log Name` 속성이 표시됩니다. `Full Name`과 `Log Name`이 항상 일치하는 것은 아닙니다. 채널의 `Full Name`을 확인하려면 Windows 이벤트 뷰어에서 해당 채널을 마우스 오른쪽 버튼으로 클릭하고 [속성(Properties)]을 엽니다.

- event_levels – 로그할 이벤트 수준을 지정합니다. 기록할 각 수준을 지정해야 합니다. 가능한 값은 INFORMATION, WARNING, ERROR, CRITICAL 및 VERBOSE입니다. 이 필드는 기록할 Windows 이벤트 유형마다 있어야 합니다.
- log_group_name - 필수입니다. CloudWatch Logs에서 로그 그룹 이름으로 사용할 이름을 지정합니다.
- log_stream_name – 선택 사항입니다. CloudWatch Logs에서 로그 스트림 이름으로 사용할 이름을 지정합니다. 이름의 일부로 {instance_id}, {hostname}, {local_hostname}, {ip_address}를 이름 안의 변수로 사용할 수 있습니다. {hostname}은 EC2 메타데이터에서 호스트 이름을 가져오고, {local_hostname}은 네트워크 구성 파일에 있는 호스트 이름을 사용합니다.

  이 필드를 생략하면 글로벌 logs 섹션의 log_stream_name 파라미터 값이 사용됩니다. 이 값도 생략하면 기본값 {instance_id}가 사용됩니다.

  로그 스트림이 아직 없는 경우 로그 스트림이 자동으로 생성됩니다.
- event_format – 선택 사항입니다. CloudWatch Logs에 Windows 이벤트를 저장할 때 사용할 형식을 지정합니다. xml은 Windows 이벤트 뷰어에서처럼 XML 형식을 사용합니다. text는 레거시 CloudWatch Logs 에이전트 형식을 사용합니다.
- retention_in_days – 선택 사항입니다. 지정된 로그 그룹에 Windows 이벤트를 보존할 일수를 지정합니다.
  - 에이전트가 지금 이 로그 그룹을 생성하고 있는 경우 이 필드를 생략하면 이 새 로그 그룹의 보존 기간이 절대 만료되지 않도록 설정됩니다.
  - 이 로그 그룹이 이미 존재하는 경우 이 필드를 지정하면 지정한 새 보존 기간이 사용됩니다. 이미 존재하는 로그 그룹에 대해 이 필드를 생략하면 로그 그룹의 보존 기간이 변경되지 않습니다.

    CloudWatch 에이전트 마법사는 에이전트 구성 파일을 생성하는 데 사용되고 로그 보존 값을 지정하지 않은 경우 이 필드의 기본값으로 -1을 사용합니다. 마법사가 설정한 -1 값 지정은 로그 그룹의 이벤트가 만료되지 않도록 지정합니다. 그러나 이 값을 -1로 수동으로 편집해도 아무런 효과가 없습니다.

  유효한 값은 1, 3, 5, 7, 14, 30, 60, 90, 120, 150, 180, 365, 400, 545, 731, 1827, 2192, 2557, 2922, 3288 및 3653입니다.

  여러 로그 스트림을 동일한 로그 그룹에 쓰도록 에이전트를 구성한 경우 한 곳에서 retention_in_days 값을 지정하면 전체 로그 그룹에 대한 로그 보존이 설정됩니다. 여러 위치에 있는 동일한 로그 그룹에 대해 retention_in_days 값을 지정하는 경우 이러한 값

이 모두 동일하면 보존이 설정됩니다. 그러나 여러 위치에서 동일한 로그 그룹에 대해 다른 `retention_in_days` 값을 지정하는 경우 로그 보존이 설정되지 않고 에이전트가 중지되어 오류가 반환됩니다.

> **Note**
>
> 에이전트의 IAM 역할 또는 IAM 사용자에게는 `logs:PutRetentionPolicy` 값이 있어야 보존 정책을 설정할 수 있습니다. 자세한 내용은 CloudWatch 에이전트가 로그 보존 정책을 설정하도록 허용 단원을 참조하십시오.

> **Warning**
>
> 이미 존재하는 로그 그룹에 대해 `retention_in_days` 값을 설정한 경우 지정한 일 수 전에 게시된 해당 로그 그룹의 모든 로그가 삭제됩니다. 예를 들어, 3으로 설정하면 3일 전과 그 이전의 모든 로그가 삭제됩니다.

- `log_stream_name` - 필수입니다. `collect_list`의 항목 내 `log_stream_name` 파라미터에 개별 로그 스트림 이름이 정의되어 있지 않은 Windows 이벤트나 로그에 사용될 기본 로그 스트림 이름을 지정합니다.
- `endpoint_override` – 에이전트가 로그를 전송하는 엔드포인트로 사용할 FIPS 엔드포인트 또는 프라이빗 링크를 지정합니다. 이 필드를 지정하고 프라이빗 링크를 설정하면 로그를 Amazon VPC 엔드포인트에 전송할 수 있습니다. 자세한 내용은 Amazon VPC란? 단원을 참조하세요.

  `endpoint_override`의 값은 URL인 문자열이어야 합니다.

  예를 들어, 구성 파일의 로그 섹션에서 다음 부분은 에이전트가 로그를 전송할 때 VPC 엔드포인트를 사용하도록 설정합니다.

```
{
  "logs": {
    "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXXXXX.logs.us-
east-1.vpce.amazonaws.com",
   ......
   },
}
```

- `force_flush_interval` – 로그가 서버로 전송되기 전에 메모리 버퍼에 남아 있는 최대 시간(초)을 지정합니다. 이 필드에 대한 설정과 상관없이 버퍼 내 로그의 크기가 1MB에 도달하면 로그가 즉시 서버로 전송됩니다. 기본값은 5입니다.

  에이전트를 사용하여 임베디드 지표 형식으로 고분해능 지표를 보고하고 해당 지표에 대한 경보를 설정하는 경우에는 이 파라미터를 기본값인 5로 설정하세요. 그렇지 않으면 지표가 지연되어 보고되므로 부분적이거나 불완전한 데이터에 대해 경보가 발생할 수 있습니다.
- `credentials` – 로그를 다른 AWS 계정에 전송할 때 사용할 IAM 역할을 지정합니다. 지정된 경우 이 필드는 1개의 파라미터 `role_arn`를 포함합니다.
  - `role_arn` – 로그를 다른 AWS 계정에 전송할 때 인증에 사용할 IAM 역할의 ARN을 지정합니다. 자세한 내용은 다른 계정에 지표, 로그, 추적 전송 단원을 참조하세요. 여기에 지정한 경우 이를 통해 구성 파일(해당 시)의 `agent` 섹션에 지정되어 있는 `role_arn`을 재정의합니다.
- `metrics_collected` – 이 필드에는 CloudWatch Application Signals, Container Insights 등의 사용 사례를 활성화하기 위해 Amazon EKS에 대한 향상된 관찰성을 이용하여 에이전트가 로그를 수집하도록 지정하는 섹션이 포함될 수 있습니다.
  - `app_signals` (선택 사항) CloudWatch Application Signals를 활성화하도록 지정합니다. 이 구성에 대한 자세한 내용은 CloudWatch Application Signals 활성화 섹션을 참조하세요.
  - `kubernetes`— 이 필드에는 Amazon EKS의 향상된 관찰 기능을 통해 Container Insights를 활성화하는 데 사용할 수 있는 `enhanced_container_insights` 파라미터가 포함될 수 있습니다.
    - `enhanced_container_insights`— Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 활성화하려면 이 `true`를 설정하세요. 자세한 내용은 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights 단원을 참조하십시오.
    - `accelerated_compute_metrics`-Amazon EKS 클러스터에서 Nvidia GPU 메트릭을 수집하지 않도록 하려면 이 옵션을 `false`로 설정하십시오. 자세한 내용은 NVIDIA GPU 지표 단원을 참조하십시오.
  - `emf` - 로그에 포함된 지표를 수집하기 위해 더 이상 `emf` 필드를 추가할 필요가 없습니다. 에이전트가 임베디드 지표 형식의 로그를 수집하도록 지정하는 레거시 필드입니다. 이러한 로그에서 지표 데이터를 생성할 수 있습니다. 자세한 내용은 로그 내에 지표 포함 단원을 참조하세요.

다음은 `logs` 섹션의 예입니다.

```
"logs":
   {
       "logs_collected": {
           "files": {
               "collect_list": [
```

```
                    {
                        "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\Logs\\amazon-cloudwatch-agent.log",
                        "log_group_name": "amazon-cloudwatch-agent.log",
                        "log_stream_name": "my_log_stream_name_1",
                        "timestamp_format": "%H: %M: %S%y%b%-d"
                    },
                    {
                        "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\Logs\\test.log",
                        "log_group_name": "test.log",
                        "log_stream_name": "my_log_stream_name_2"
                    }
                ]
            },
            "windows_events": {
                "collect_list": [
                    {
                        "event_name": "System",
                        "event_levels": [
                            "INFORMATION",
                            "ERROR"
                        ],
                        "log_group_name": "System",
                        "log_stream_name": "System"
                    },
                    {
                        "event_name": "CustomizedName",
                        "event_levels": [
                            "INFORMATION",
                            "ERROR"
                        ],
                        "log_group_name": "CustomizedLogGroup",
                        "log_stream_name": "CustomizedLogStream"
                    }
                ]
            }
        },
        "log_stream_name": "my_log_stream_name",
        "metrics_collected": {
            "kubernetes": {
                 "enhanced_container_insights": true
      }
    }
```

```
  }
```

CloudWatch 에이전트 구성 파일: 추적 섹션

CloudWatch 에이전트 구성 파일에 `traces` 섹션을 추가하여 CloudWatch Application Insignal을 활성화하거나 X-Ray 및 OpenTelemetry 계측 SDK에서 트레이스를 수집하여 X-Ray로 전송할 수 있습니다.

> **Important**
>
> 에이전트의 IAM 역할 또는 IAM 사용자는 트레이스 데이터를 X-Ray로 전송하기 위해 AWSXrayWriteOnlyAccess 정책을 가지고 있어야 합니다. 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성 단원을 참조하십시오.

추적 수집을 빠르게 시작하려면 CloudWatch 에이전트 구성 파일에 다음 사항만 추가하면 됩니다.

```
"traces_collected": {
        "xray": {
        },
        "otlp": {
        }
      }
```

이전 섹션을 CloudWatch 에이전트 구성 파일에 추가하고 에이전트를 다시 시작하면, 에이전트가 다음 기본 옵션 및 값을 사용하여 추적 수집을 시작합니다. 이러한 파라미터에 대한 자세한 내용은 이 섹션 뒷부분의 파라미터 정의를 참조하세요.

```
"traces_collected": {
        "xray": {
            "bind_address": "127.0.0.1:2000",
            "tcp_proxy": {
              "bind_address": "127.0.0.1:2000"
            }
        },
        "otlp": {
            "grpc_endpoint": "127.0.0.1:4317",
            "http_endpoint": "127.0.0.1:4318"
        }
      }
```

`traces` 섹션에는 다음 필드가 포함될 수 있습니다.

- `traces_collected` – `traces` 섹션이 포함되어 있는 경우 필수입니다. 추적을 수집할 SDK를 지정합니다. 여기에는 다음 필드가 포함될 수 있습니다.
  - `app_signals` – 선택 사항입니다. CloudWatch Application Signals를 활성화하도록 지정합니다. 이 구성에 대한 자세한 내용은 CloudWatch Application Signals 활성화 섹션을 참조하세요.
  - `xray` – 선택 사항입니다. X-Ray SDK에서 추적을 수집하도록 지정합니다. 이 섹션은 다음 필드를 포함할 수 있습니다.
    - `bind_address` – 선택 사항입니다. CloudWatch 에이전트가 X-Ray 추적을 수신하는 데 사용할 UDP 주소를 지정합니다. 형식은 `ip:port`입니다. 이 주소는 X-Ray SDK에 설정된 주소와 일치해야 합니다.

      이 필드를 생략하면 기본값인 `127.0.0.1:2000`가 사용됩니다.
    - `tcp_proxy` – 선택 사항입니다. X-Ray 원격 샘플링을 지원하는 데 사용되는 프록시 주소를 구성합니다. 자세한 내용은 X-Ray 문서에서 Configuring sampling rules를 참조하세요.

      이 섹션에는 다음 필드가 포함될 수 있습니다.
      - `bind_address` – 선택 사항입니다. CloudWatch 에이전트가 프록시를 설정할 TCP 주소를 지정합니다. 형식은 `ip:port`입니다. 이 주소는 X-Ray SDK에 설정된 주소와 일치해야 합니다.

        이 필드를 생략하면 기본값인 `127.0.0.1:2000`가 사용됩니다.
- `otlp` – 선택 사항입니다. OpenTelemetry SDK에서 추적을 수집하도록 지정합니다. AWS Distro for OpenTelemetry에 대한 자세한 내용은 AWS Distro for OpenTelemetry를 참조하세요. AWS OpenTelemetry SDK용 배포판에 대한 자세한 내용은 Introduction을 참조하세요.

  이 섹션은 다음 필드를 포함할 수 있습니다.
  - `grpc_endpoint` – 선택 사항입니다. gRPC 원격 프로시저 호출을 사용하여 전송된 OpenTelemetry 추적을 수신하는 데 사용할 CloudWatch 에이전트의 주소를 지정합니다. 형식은 `ip:port`입니다. 이 주소는 OpenTelemetry SDK에서 gRPC 내보내기에 설정된 주소와 일치해야 합니다.

    이 필드를 생략하면 기본값인 `127.0.0.1:4317`가 사용됩니다.
  - `http_endpoint` – 선택 사항입니다. HTTP를 통해 전송된 OTLP 추적을 수신하는 데 사용할 CloudWatch 에이전트의 주소를 지정합니다. 형식은 `ip:port`입니다. s이 주소는 OpenTelemetry SDK에서 HTTP 내보내기에 설정된 주소와 일치해야 합니다.

    이 필드를 생략하면 기본값인 `127.0.0.1:4318`가 사용됩니다.

- concurrency – 선택 사항입니다. 추적 업로드에 사용할 수 있는 X-Ray의 최대 동시 호출 수를 지정합니다. 기본값은 8입니다.
- local_mode – 선택 사항입니다. true의 경우 에이전트는 Amazon EC2 인스턴스 메타데이터를 수집하지 않습니다. 기본값은 입니다.false
- endpoint_override – 선택 사항입니다. CloudWatch 에이전트가 추적을 전송하는 엔드포인트로 사용할 FIPS 엔드포인트 또는 프라이빗 링크를 지정합니다. 이 필드를 지정하고 프라이빗 링크를 설정하면 Amazon VPC 엔드포인트로 추적을 전송할 수 있습니다. 자세한 내용은 Amazon VPC란 무엇인가?를 참조하세요.

  endpoint_override의 값은 URL인 문자열이어야 합니다.
- region_override – 선택 사항입니다. X-Ray 엔드포인트에 사용할 리전을 지정합니다. CloudWatch 에이전트는 지정된 리전의 X-Ray로 추적을 보냅니다. 이 필드를 생략하면 에이전트는 Amazon EC2 인스턴스가 있는 리전으로 추적을 보냅니다.

  여기에서 리전을 지정하면 구성 파일의 agent 섹션에 있는 region 파라미터 설정보다 우선합니다.
- proxy_override – 선택 사항입니다. X-Ray에 요청을 보낼 때 사용할 CloudWatch 에이전트의 프록시 서버 주소를 지정합니다. 프록시 서버의 프로토콜은 이 주소의 일부로 지정되어 있어야 합니다.
- credentials – 추적을 다른 AWS 계정에 전송할 때 사용할 IAM 역할을 지정합니다. 지정된 경우 이 필드는 1개의 파라미터 role_arn를 포함합니다.
  - role_arn – 추적을 다른 AWS 계정에 전송할 때 인증에 사용할 IAM 역할의 ARN을 지정합니다. 자세한 내용은 다른 계정에 지표, 로그, 추적 전송 단원을 참조하십시오. 여기에 지정한 경우 이를 통해 구성 파일(해당 시)의 agent 섹션에 지정되어 있는 role_arn을 재정의합니다.

CloudWatch 에이전트 구성 파일: 전체 예

다음은 Linux 서버에 대한 전체 CloudWatch 에이전트 구성 파일의 예입니다.

수집하려는 척도에 대해 measurement 섹션에 나열된 항목들은 완전한 척도 이름을 지정하거나, 리소스 유형에 추가되는 척도 이름의 일부만 지정할 수 있습니다. 예를 들어 diskio 섹션의 measurement 섹션에 reads 또는 diskio_reads을 지정하면 diskio_reads 지표가 수집될 수 있습니다.

이 예에는 measurement 섹션에서 지표를 지정하는 두 가지 방법이 모두 포함되어 있습니다.

```
{
  "agent": {
    "metrics_collection_interval": 10,
    "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log"
```

```
      },
      "metrics": {
        "namespace": "MyCustomNamespace",
        "metrics_collected": {
          "cpu": {
            "resources": [
              "*"
            ],
            "measurement": [
              {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit":
 "Percent"},
              {"name": "cpu_usage_nice", "unit": "Percent"},
              "cpu_usage_guest"
            ],
            "totalcpu": false,
            "metrics_collection_interval": 10,
            "append_dimensions": {
              "customized_dimension_key_1": "customized_dimension_value_1",
              "customized_dimension_key_2": "customized_dimension_value_2"
            }
          },
          "disk": {
            "resources": [
              "/",
              "/tmp"
            ],
            "measurement": [
              {"name": "free", "rename": "DISK_FREE", "unit": "Gigabytes"},
              "total",
              "used"
            ],
             "ignore_file_system_types": [
              "sysfs", "devtmpfs"
            ],
            "metrics_collection_interval": 60,
            "append_dimensions": {
              "customized_dimension_key_3": "customized_dimension_value_3",
              "customized_dimension_key_4": "customized_dimension_value_4"
            }
          },
          "diskio": {
            "resources": [
              "*"
            ],
```

```
            "measurement": [
              "reads",
              "writes",
              "read_time",
              "write_time",
              "io_time"
            ],
            "metrics_collection_interval": 60
          },
          "swap": {
            "measurement": [
              "swap_used",
              "swap_free",
              "swap_used_percent"
            ]
          },
          "mem": {
            "measurement": [
              "mem_used",
              "mem_cached",
              "mem_total"
            ],
            "metrics_collection_interval": 1
          },
          "net": {
            "resources": [
              "eth0"
            ],
            "measurement": [
              "bytes_sent",
              "bytes_recv",
              "drop_in",
              "drop_out"
            ]
          },
          "netstat": {
            "measurement": [
              "tcp_established",
              "tcp_syn_sent",
              "tcp_close"
            ],
            "metrics_collection_interval": 60
          },
          "processes": {
```

```
            "measurement": [
              "running",
              "sleeping",
              "dead"
            ]
          }
        },
        "append_dimensions": {
          "ImageId": "${aws:ImageId}",
          "InstanceId": "${aws:InstanceId}",
          "InstanceType": "${aws:InstanceType}",
          "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
        },
        "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"],
 ["d1"],[]],
        "force_flush_interval" : 30
      },
      "logs": {
        "logs_collected": {
          "files": {
            "collect_list": [
              {
                "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
                "log_group_name": "amazon-cloudwatch-agent.log",
                "log_stream_name": "amazon-cloudwatch-agent.log",
                "timezone": "UTC"
              },
              {
                "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
                "log_group_name": "test.log",
                "log_stream_name": "test.log",
                "timezone": "Local"
              }
            ]
          }
        },
        "log_stream_name": "my_log_stream_name",
        "force_flush_interval" : 15,
        "metrics_collected": {
           "kubernetes": {
                "enhanced_container_insights": true
      }
    }
```

```
  }
}
```

다음은 Windows Server가 실행되는 서버에 대한 전체 CloudWatch 에이전트 구성 파일의 예입니다.

```
{
      "agent": {
        "metrics_collection_interval": 60,
        "logfile": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log"
      },
      "metrics": {
        "namespace": "MyCustomNamespace",
        "metrics_collected": {
          "Processor": {
            "measurement": [
              {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
              "% Interrupt Time",
              "% User Time",
              "% Processor Time"
            ],
            "resources": [
              "*"
            ],
            "append_dimensions": {
              "customized_dimension_key_1": "customized_dimension_value_1",
              "customized_dimension_key_2": "customized_dimension_value_2"
            }
          },
          "LogicalDisk": {
            "measurement": [
              {"name": "% Idle Time", "unit": "Percent"},
              {"name": "% Disk Read Time", "rename": "DISK_READ"},
              "% Disk Write Time"
            ],
            "resources": [
              "*"
            ]
          },
          "customizedObjectName": {
            "metrics_collection_interval": 60,
            "customizedCounterName": [
              "metric1",
```

```
            "metric2"
          ],
          "resources": [
            "customizedInstances"
          ]
        },
        "Memory": {
          "metrics_collection_interval": 5,
          "measurement": [
            "Available Bytes",
            "Cache Faults/sec",
            "Page Faults/sec",
            "Pages/sec"
          ]
        },
        "Network Interface": {
          "metrics_collection_interval": 5,
          "measurement": [
            "Bytes Received/sec",
            "Bytes Sent/sec",
            "Packets Received/sec",
            "Packets Sent/sec"
          ],
          "resources": [
            "*"
          ],
          "append_dimensions": {
            "customized_dimension_key_3": "customized_dimension_value_3"
          }
        },
        "System": {
          "measurement": [
            "Context Switches/sec",
            "System Calls/sec",
            "Processor Queue Length"
          ]
        }
      },
      "append_dimensions": {
        "ImageId": "${aws:ImageId}",
        "InstanceId": "${aws:InstanceId}",
        "InstanceType": "${aws:InstanceType}",
        "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
      },
```

```
        "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"],
 ["d1"],[]]
      },
      "logs": {
        "logs_collected": {
          "files": {
            "collect_list": [
              {
                "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\
\amazon-cloudwatch-agent.log",
                "log_group_name": "amazon-cloudwatch-agent.log",
                "timezone": "UTC"
              },
              {
                "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\
\test.log",
                "log_group_name": "test.log",
                "timezone": "Local"
              }
            ]
          },
          "windows_events": {
            "collect_list": [
              {
                "event_name": "System",
                "event_levels": [
                  "INFORMATION",
                  "ERROR"
                ],
                "log_group_name": "System",
                "log_stream_name": "System",
                "event_format": "xml"
              },
              {
                "event_name": "CustomizedName",
                "event_levels": [
                  "WARNING",
                  "ERROR"
                ],
                "log_group_name": "CustomizedLogGroup",
                "log_stream_name": "CustomizedLogStream",
                "event_format": "xml"
              }
            ]
```

```
          }
        },
        "log_stream_name": "example_log_stream_name"
      }
    }
```

수동으로 CloudWatch 에이전트 구성 파일 저장

CloudWatch 에이전트 구성 파일을 수동으로 생성하거나 편집하는 경우 이름을 지정할 수 있습니다. 문제를 간단하게 해결할 수 있도록 이름을 Linux 서버에서는 `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`로 지정하고, Windows 서버를 실행하는 서버에서는 `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`로 지정하는 것이 좋습니다. 파일을 생성한 후에는 해당 파일을 에이전트를 실행하려고 하는 다른 서버에 복사할 수 있습니다.

Systems Manager 파라미터 스토어에 CloudWatch 에이전트 구성 파일 업로드

SSM Agent를 사용하여 서버에 CloudWatch 에이전트를 설치하려는 경우 CloudWatch 에이전트 구성 파일을 수동으로 편집한 후 이를 Systems Manager 파라미터 스토어에 업로드할 수 있습니다. 그렇게 하려면 Systems Manager `put-parameter` 명령을 사용합니다.

파일을 파라미터 스토어에 저장할 수 있으려면 충분한 권한이 있는 IAM 역할을 사용해야 합니다. 자세한 내용은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성 단원을 참조하세요.

다음 명령을 사용합니다. 여기서 *`parameter name`*은 파라미터 스토어에서 이 파일에 사용할 이름이며 *`configuration_file_pathname`*은 편집한 구성 파일의 경로 및 파일 이름입니다.

```
aws ssm put-parameter --name "parameter name" --type "String" --value
 file://configuration_file_pathname
```

CloudWatch Application Signals 활성화

CloudWatch Application Signals를 사용하면 AWS에서 애플리케이션을 자동으로 계측하여 비즈니스 목표에 따라 애플리케이션 성능을 추적할 수 있습니다. Application Signals는 Java 애플리케이션, 해당 종속성 및 엣지에 대한 통합 애플리케이션 중심 보기를 제공합니다. 자세한 내용은 Application Signals 단원을 참조하십시오.

CloudWatch Application Signals는 CloudWatch 에이전트를 활용하여 자동 계측 애플리케이션으로부터 지표와 트레이스를 수신하고, 필요에 따라 높은 카디널리티를 줄이기 위한 규칙을 적용한 다음, 처리된 텔레메트리를 CloudWatch에 게시합니다. 에이전트 구성 파일을 사용하여 Application Signals

용으로 CloudWatch 에이전트에 사용자 지정 구성을 제공할 수 있습니다. 우선, 에이전트 구성 파일의 `logs` 섹션 내 `metrics_collected` 섹션 아래에 `app_signals` 섹션이 있으면 CloudWatch 에이전트가 자동 계측된 애플리케이션에서 지표를 수신하도록 지정됩니다. 마찬가지로, 에이전트 구성 파일의 `traces` 섹션 내 `traces_collected` 섹션 아래에 `app_signals` 섹션이 있으면 CloudWatch 에이전트가 자동 계측된 애플리케이션에서 지표를 수신할 수 있게 활성화되도록 지정됩니다. 또한 이 섹션에 설명된 대로 사용자 지정 구성 규칙을 전달하여 높은 카디널리티 텔레메트리 게시를 줄일 수도 있습니다.

- Amazon EKS 클러스터의 경우 Amazon CloudWatch Observability EKS 추가 기능을 설치하면 기본적으로 CloudWatch 에이전트가 자동 계측된 애플리케이션에서 지표와 트레이스를 모두 수신하도록 활성화됩니다. 필요에 따라 사용자 지정 구성 규칙을 전달하려면 (선택 사항) 추가 구성에 설명된 대로 추가 구성을 사용하여 생성하거나 업데이트할 때 Amazon EKS 추가 기능에 사용자 지정 에이전트 구성을 전달하면 됩니다.
- Amazon EC2를 비롯한 기타 지원되는 플랫폼의 경우 이 섹션 뒷부분에 설명된 대로 `app_signals` 섹션과 사용자 지정 구성 규칙(선택 사항)을 지정하여 Application Signals를 활성화하는 에이전트 구성으로 CloudWatch 에이전트를 시작해야 합니다.

다음은 CloudWatch Application Signals와 관련된 CloudWatch 에이전트 구성 파일의 필드 개요입니다.

- `logs`
  - `metrics_collected` – 이 필드에는 CloudWatch Application Signals, Container Insights 등의 사용 사례를 활성화하기 위해 Amazon EKS에 대한 향상된 관찰성을 이용하여 에이전트가 로그를 수집하도록 지정하는 섹션이 포함될 수 있습니다.

    > **Note**
    >
    > 이전에는 에이전트가 임베디드 지표 형식의 로그를 수집하도록 지정하는 데에도 이 섹션을 사용했습니다. 이러한 설정은 더 이상 필요하지 않습니다.

    - `app_signals`(선택 사항) CloudWatch Application Signals 촉진을 위해 자동 계측 애플리케이션에서 지표를 수신하도록 CloudWatch Application Signals를 활성화할 것을 지정합니다.
      - `rules`(선택 사항) 지표와 트레이스를 조건부로 선택하고 카디널리티가 높은 시나리오를 처리하기 위한 작업을 적용하는 규칙의 배열입니다. 각 규칙은 다음 필드를 포함할 수 있습니다.
        - `rule_name`(선택 사항) 규칙의 이름입니다.

- `selectors`(선택 사항) 지표 및 트레이스 측정기준 매처의 배열입니다. 각 선택기는 다음 필드를 제공해야 합니다.
  - `dimension` `selectors`가 비어 있지 않은 경우 필수입니다. 필터로 사용할 지표와 트레이스의 측정기준을 지정합니다.
  - `match` `selectors`가 비어 있지 않은 경우 필수입니다. 지정된 측정기준의 값을 일치시키는 데 사용되는 와일드카드 패턴입니다.
- `action`(선택 사항) 지정된 선택기와 일치하는 지표 및 트레이스에 적용할 작업입니다. `action`의 값은 다음 키워드 중 하나여야 합니다.
  - `keep` `selectors`와 일치하는 경우 지표 및 트레이스만 CloudWatch로 전송하도록 지정합니다.
  - `drop` `selectors`와 일치하는 지표와 트레이스를 삭제하도록 지정합니다.
  - `replace` `selectors`와 일치하는 지표와 트레이스의 측정기준을 교체하도록 지정합니다. `replacements` 섹션에 따라 교체됩니다.
- `replacements` `action`가 `replace`인 경우 필수입니다. `action`이 `replace`일 때 지정된 `selectors`와 일치하는 지표 및 트레이스에 적용되는 측정기준 및 값 페어의 배열입니다. 각 대체는 다음 필드를 제공해야 합니다.
  - `target_dimension` `replacements`가 비어 있지 않은 경우 필수입니다. 대체해야 하는 측정기준을 지정합니다.
  - `value` `replacements`가 비어 있지 않은 경우 필수입니다. `target_dimension`의 원래 값을 대체할 값입니다.
- `limiter`(선택 사항) 이 섹션을 사용하여 Application Signals가 CloudWatch로 보내는 지표 및 차원 수를 제한하여 비용을 최적화합니다.
  - `disabled`(선택 사항) `true`인 경우 지표 제한 기능이 비활성화됩니다. 기본값은 입니다.`false`
  - `drop_threshold`(선택 사항) CloudWatch 에이전트 하나로 내보낼 수 있는 한 교체 간격의 서비스당 최대 개별 지표 수입니다. 기본값은 500입니다.
  - `rotation_interval`(선택 사항) 제한기가 개별 계산을 위해 지표 레코드를 재설정하는 간격입니다. 이는 일련의 숫자와 단위 접미사가 포함된 문자열로 표현됩니다. 분수는 지원됩니다. 지원되는 단위 접미사는 `s`, `m`, `h`, `ms`, `us`, `ns`입니다.

    `1h`의 기본값은 1시간입니다.
  - `log_dropped_metrics`(선택 사항) Application Signals 지표가 삭제될 때 에이전트가 CloudWatch 에이전트 로그에 로그를 기록해야 하는지 여부를 지정합니다. 기본값은 `false`입니다.

> **Note**
>
> 이 로깅을 활성화하려면 `agent` 섹션의 `debug` 파라미터도 `true`로 설정해야 합니다.

- `traces`
  - `traces_collected`
    - `app_signals` 선택 사항입니다. 이를 지정하면 CloudWatch Application Signals 촉진을 위해 CloudWatch 에이전트가 자동 계측 애플리케이션에서 트레이스를 수신할 수 있습니다.

> **Note**
>
> 사용자 지정 `app_signals` 규칙이 `logs` 섹션에 포함된 `metrics_collected` 섹션 아래에 지정되어 있더라도 암시적으로 `traces_collected` 섹션에도 적용됩니다. 지표와 트레이스 모두에 동일한 규칙 세트가 적용됩니다.

서로 다른 작업을 포함하는 여러 규칙이 있는 경우 `keep`, `drop`, `replace` 순서로 적용됩니다.

다음은 사용자 지정 규칙을 적용하는 전체 CloudWatch 에이전트 구성 파일의 예제입니다.

```
{
  "logs": {
    "metrics_collected": {
      "app_signals": {
        "rules": [
          {
            "rule_name": "keep01",
            "selectors": [
              {
                "dimension": "Service",
                "match": "pet-clinic-frontend"
              },
              {
                "dimension": "RemoteService",
                "match": "customers-service"
              }
            ],
            "action": "keep"
```

```
          },
          {
            "rule_name": "drop01",
            "selectors": [
              {
                "dimension": "Operation",
                "match": "GET /api/customer/owners/*"
              }
            ],
            "action": "drop"
          },
          {
            "rule_name": "replace01",
            "selectors": [
              {
                "dimension": "Operation",
                "match": "PUT /api/customer/owners/*/pets/*"
              },
              {
                "dimension": "RemoteOperation",
                "match": "PUT /owners"
              }
            ],
            "replacements": [
              {
                "target_dimension": "Operation",
                "value": "PUT /api/customer/owners/{ownerId}/pets{petId}"
              }
            ],
            "action": "replace"
          }
        ]
      }
    }
  },
  "traces": {
    "traces_collected": {
      "app_signals": {}
    }
  }
}
```

이전 예제 구성 파일의 경우 `rules`는 다음과 같이 처리됩니다.

1. keep01 규칙은 Service 측정기준이 pet-clinic-frontend이고 RemoteService 측정기준이 customers-service인 모든 지표와 트레이스가 유지되도록 합니다.
2. keep01을 적용한 후 처리된 지표 및 트레이스의 경우 drop01 규칙은 Operation 측정기준이 GET /api/customer/owners/*인 지표 및 트레이스가 삭제되도록 합니다.
3. drop01을 적용한 후 처리된 지표 및 트레이스의 경우 replace01 규칙은 Operation 측정기준이 PUT /api/customer/owners/*/pets/*이고 RemoteOperation 측정기준이 PUT /owners인 지표 및 트레이스를 업데이트하므로 Operation 측정기준이 이제PUT /api/customer/owners/{ownerId}/pets{petId}로 대체됩니다.

다음은 지표 제한을 100으로 변경하고, 삭제된 지표의 로깅을 활성화하고, 교체 간격을 2시간으로 설정하여 Application Signals의 카디널리티를 관리하는 CloudWatch 구성 파일의 전체 예시입니다.

```
{
    "logs": {
        "metrics_collected": {
            "app_signals": {
                "limiter": {
                    "disabled": false,
                    "drop_threshold": 100,
                    "rotation_interval": "2h",
                    "log_dropped_metrics": true
                }
            }
        },
        "traces": {
            "traces_collected": {
                "app_signals": {}
            }
        }
    }
}
```

네트워크 성능 지표 수집

Elastic Network Adapter(ENA)를 사용하는 Linux에서 실행되는 EC2 인스턴스는 네트워크 성능 지표를 게시합니다. 버전 1.246396.0 이상의 CloudWatch 에이전트를 사용하면 이러한 네트워크 성능 지표를 CloudWatch로 가져올 수 있습니다. 이러한 네트워크 성능 지표를 CloudWatch로 가져오면 해당 지표는 CloudWatch 사용자 지정 지표로 요금이 청구됩니다.

ENA 드라이버에 대한 자세한 내용은 Linux 인스턴스에서 Elastic Network Adapter(ENA)를 통한 향상된 네트워킹 사용 및 Windows 인스턴스에서 Elastic Network Adapter(ENA)를 통한 향상된 네트워킹 사용 단원을 참조하세요.

네트워크 성능 지표 수집을 설정하는 방법은 Linux 서버와 Windows 서버에서 서로 다릅니다.

다음 표에는 ENA 어댑터에서 사용 설정하는 네트워크 성능 지표가 나와 있습니다. CloudWatch 에이전트가 Linux 인스턴스에서 CloudWatch로 이러한 지표를 가져올 때 이러한 각 지표 이름의 시작 부분에 `ethtool_`을 추가합니다.

지표	설명
Linux 서버에서의 이름: **`bw_in_allowance_exceeded`** Windows 서버에서의 이름: **`Aggregate inbound BW allowance exceeded`**	인바운드 집계 대역폭이 인스턴스의 최댓값을 초과하여 대기열에 있거나 삭제된 패킷 수입니다. 이 지표는 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 있는 `ethtool` 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음
Linux 서버에서의 이름: **`bw_out_allowance_exceeded`** Windows 서버에서의 이름: **`Aggregate outbound BW allowance exceeded`**	아웃바운드 집계 대역폭이 인스턴스의 최댓값을 초과하여 대기열에 있거나 삭제된 패킷 수입니다. 이 지표는 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 있는 `ethtool` 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음
Linux 서버에서의 이름: **`conntrack_allowance_available`** Windows 서버에서의 이름: **`Available connection tracking allowance`**	해당 인스턴스 유형의 추적된 연결 허용 한도에 도달하기 전에 인스턴스에서 설정할 수 있는 추적된 연결 수를 보고합니다. 이 지표는 버전 2.8.1부터 Elastic Network Adapter(ENA)용 Linux 드라이버를 사용하는 Nitro 기반 EC2 인스턴스와 버전 2.6.0부터 Elastic Network Adapter(ENA)용 Windows 드라이버를 사용하는 컴퓨터에서만 사용할 수 있습니다.

지표	설명
	이 지표는 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 있는 `ethtool` 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음
Linux 서버에서의 이름: **`ena_srd_mode`** Windows 서버에서의 이름: **`ena srd mode`**	어떤 ENA Express 기능이 활성화되어 있는지 설명합니다. ENA Express에 대한 자세한 내용은 Linux 인스턴스에서 ENA Express를 사용하여 네트워크 성능 개선을 참조하세요. 값은 다음과 같습니다. • `0` = ENA Express 꺼짐, UDP 꺼짐 • `1` = ENA Express 켜짐, UDP 꺼짐 • `2` = ENA Express 꺼짐, UDP 켜짐 **Note** ENA Express가 원래 활성화되어 있고, UDP가 ENA Express를 사용하도록 구성된 경우에만 이렇게 나타납니다. UDP 트래픽에 대한 이전 값이 유지됩니다. • `3` = ENA Express 켜짐, UDP 켜짐

지표	설명
Linux 서버에서의 이름: **ena_srd_eligible_tx_pkts** Windows 서버에서의 이름: **ena srd eligible tx pkts**	다음과 같이, 일정 기간 동안 전송된 네트워크 패킷 중 적합성에 대한 AWS SRD(Scalable Reliable Datagram) 요구 사항을 충족하는 네트워크 패킷의 수입니다. • 전송 인스턴스 유형과 수신 인스턴스 유형이 모두 지원되어야 합니다. • 전송 인스턴스와 수신 인스턴스 모두에 ENA Express가 구성되어 있어야 합니다. • 전송 인스턴스와 수신 인스턴스가 동일한 서브넷에 있어야 합니다. • 인스턴스 간 네트워크 경로에 미들웨어 박스가 포함되지 않아야 합니다. ENA Express는 현재 미들웨어 박스를 지원하지 않습니다.
Linux 서버에서의 이름: **ena_srd_tx_pkts** Windows 서버에서의 이름: **ena srd tx pkts**	일정 기간 동안 전송한 SRD 패킷의 수입니다.
Linux 서버에서의 이름: **ena_srd_rx_pkts** Windows 서버에서의 이름: **ena srd rx pkts**	일정 기간 동안 수신한 SRD 패킷의 수입니다.
Linux 서버에서의 이름: **ena_srd_resource_utilization** Windows 서버에서의 이름: **ena srd resource utilization**	동시 SRD 연결에 허용된 최대 메모리 사용률 중 인스턴스에 사용된 비율입니다.

지표	설명
Linux 서버에서의 이름: **linklocal_allowance_exceeded** Windows 서버에서의 이름: **Link local packet rate allowance exceeded**	로컬 프록시 서비스에 대한 트래픽의 PPS가 네트워크 인터페이스의 최대값을 초과하여 손실된 패킷 수입니다. 이는 DNS 서비스, Instance Metadata Service 및 Amazon Time Sync Service에 대한 트래픽에 영향을 미칩니다. 이 지표는 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 있는 `ethtool` 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음
Linux 서버에서의 이름: **linklocal_allowance_exceeded** Windows 서버에서의 이름: **Link local packet rate allowance exceeded**	로컬 프록시 서비스에 대한 트래픽의 PPS가 네트워크 인터페이스의 최대값을 초과하여 손실된 패킷 수입니다. 이는 DNS 서비스, Instance Metadata Service 및 Amazon Time Sync Service에 대한 트래픽에 영향을 미칩니다. 이 지표는 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 있는 `ethtool` 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음
Linux 서버에서의 이름: **pps_allowance_exceeded** Windows 서버에서의 이름: **PPS allowance exceeded**	양방향 PPS가 인스턴스의 최댓값을 초과하여 대기열에 있거나 삭제된 패킷 수입니다. 이 지표는 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 있는 `ethtool` 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음

Linux 설정

Linux 서버에서 ethtool '플러그 인'을 사용하면 네트워크 성능 지표를 CloudWatch로 가져올 수 있습니다.

ethtool은 Linux 서버의 이더넷 디바이스에 관한 통계를 수집할 수 있는 표준 Linux 유틸리티입니다. 수집하는 통계는 네트워크 디바이스 및 드라이버에 따라 다릅니다. 이러한 통계의 예로는 `tx_cnt`, `rx_bytes`, `tx_errors`, `align_errors`가 있습니다. CloudWatch 에이전트와 함께 ethtool 플러그 인을 사용하는 경우 이 단원의 앞부분에서 설명한 EC2 네트워크 성능 지표와 함께 이러한 통계도 CloudWatch로 가져올 수 있습니다.

> **Tip**
>
> 운영 체제 및 네트워크 장치에서 사용 가능한 통계를 찾으려면 `ethtool -S` 명령을 사용하세요.

CloudWatch 에이전트는 지표를 CloudWatch로 가져올 때 가져온 모든 지표의 이름에 `ethtool_` 접두사를 추가합니다. 따라서 표준 ethtool 통계 `rx_bytes`는 CloudWatch에서 `ethtool_rx_bytes`라고 하며, EC2 네트워크 성능 지표 `bw_in_allowance_exceeded`는 CloudWatch에서 `ethtool_bw_in_allowance_exceeded`라고 합니다.

Linux 서버에서, ethtool 지표를 가져오려면 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 `ethtool` 섹션을 추가합니다. `ethtool` 섹션에는 다음과 같은 하위 섹션이 포함될 수 있습니다.

- interface_include - 이 섹션을 포함하면 에이전트가 이 섹션에 나열된 이름이 있는 인터페이스에서만 지표를 수집합니다. 이 섹션을 생략하면 `interface_exclude`에 나열되지 않은 모든 이더넷 인터페이스에서 지표를 수집합니다.

  기본 이더넷 인터페이스는 `eth0`입니다.
- interface_exclude - 이 섹션을 포함하는 경우 지표를 수집하지 않으려는 이더넷 인터페이스를 나열합니다.

  ethtool 플러그 인은 항상 루프백 인터페이스를 무시합니다.
- metrics_include - 이 섹션에서는 CloudWatch로 가져올 지표를 나열합니다. 여기에는 ethtool이 수집한 표준 통계와 Amazon EC2 고분해능 네트워크 지표가 모두 포함될 수 있습니다.

다음 예는 CloudWatch 에이전트 구성 파일의 일부를 보여 줍니다. 이 구성은 eth1 인터페이스에서만 표준 ethtool 지표인 rx_packets 및 tx_packets와 Amazon EC2 네트워크 성능 지표를 수집합니다.

CloudWatch 에이전트 구성 파일에 대한 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하세요.

```
"metrics": {
    "append_dimensions": {
      "InstanceId": "${aws:InstanceId}"
    },
    "metrics_collected": {
      "ethtool": {
        "interface_include": [
          "eth1"
        ],
        "metrics_include": [
          "rx_packets",
          "tx_packets",
          "bw_in_allowance_exceeded",
          "bw_out_allowance_exceeded",
          "conntrack_allowance_exceeded",
          "linklocal_allowance_exceeded",
          "pps_allowance_exceeded"
         ]
      }
   }
}
```

Windows 설정

Windows 서버에서는 CloudWatch 에이전트가 이미 지표를 수집하는 Windows 성능 카운터를 통해 네트워크 성능 지표를 사용할 수 있습니다. 따라서 Windows 서버에서 이러한 지표를 수집하는 데 플러그인이 필요하지 않습니다.

다음은 Windows에서 네트워크 성능 지표를 수집하는 샘플 구성 파일입니다. CloudWatch 에이전트 구성 파일 편집에 대한 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하세요.

```
{
    "metrics": {
        "append_dimensions": {
            "InstanceId": "${aws:InstanceId}"
```

```
        },
        "metrics_collected": {
            "ENA Packets Shaping": {
                "measurement": [
                    "Aggregate inbound BW allowance exceeded",
                    "Aggregate outbound BW allowance exceeded",
                    "Connection tracking allowance exceeded",
                    "Link local packet rate allowance exceeded",
                    "PPS allowance exceeded"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ]
            }
        }
    }
}
```

네트워크 성능 지표 보기

네트워크 성능 지표를 CloudWatch로 가져온 후 이러한 지표를 시계열 그래프로 보고, 해당 지표를 감시하여 지표가 지정된 임곗값을 위반하는 경우 이를 알릴 수 있는 경보를 생성할 수 있습니다. 다음 절차에서는 ethtool 지표를 시계열 그래프로 보는 방법을 보여 줍니다. 경보 설정에 대한 자세한 내용은 Amazon CloudWatch 경보 사용 단원을 참조하세요.

이러한 지표는 모두 집계 카운터이므로 `RATE(METRICS())`와 같은 CloudWatch 지표 수학 함수를 사용하여 그래프에서 해당 지표의 비율을 계산하거나 경보를 설정하는 데 사용할 수 있습니다. 지표 수학 함수에 대한 자세한 내용은 지표 수학 사용 단원을 참조하세요.

CloudWatch 콘솔에서 네트워크 성능 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 에이전트가 수집한 지표의 네임스페이스를 선택합니다. 기본적으로 이 네임스페이스는 CWAgent이지만, CloudWatch 에이전트 구성 파일에서 다른 네임스페이스를 지정했을 수 있습니다.
4. 지표 측정기준(예: 인스턴스별 지표)을 선택합니다.
5. 모든 지표 탭에 네임스페이스의 해당 측정기준에 대한 모든 지표가 표시됩니다. 다음을 수행할 수 있습니다.

a. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.

b. 테이블을 정렬하려면 열 머리글을 사용합니다.

c. 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.

d. 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.

6. (선택 사항) 이 그래프를 CloudWatch 대시보드에 추가하려면 [작업(Actions)]을 선택한 다음, [대시보드에 추가(Add to dashboard)]를 선택합니다.

NVIDIA GPU 지표 수집

CloudWatch 에이전트를 사용하여 Linux 서버에서 NVIDIA GPU 지표를 수집할 수 있습니다. 이를 설정하려면 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 `nvidia_gpu` 섹션을 추가합니다. 자세한 내용은 Linux 섹션 단원을 참조하십시오.

또한 인스턴스에는 NVIDIA 드라이버가 설치되어 있어야 합니다. 일부 Amazon Machine Image(AMI)에는 NVIDIA 드라이버가 사전 설치되어 있습니다. 그렇지 않다면 드라이버를 수동으로 설치하세요. 자세한 내용은 Linux 인스턴스에 NVIDIA 드라이버 설치를 참조하세요.

다음 지표를 수집할 수 있습니다. 이러한 모든 지표는 CloudWatch `Unit` 없이 수집되지만, CloudWatch 에이전트 구성 파일에 파라미터를 추가하여 각 지표에 대한 단위를 지정할 수 있습니다. 자세한 내용은 Linux 섹션 단원을 참조하십시오.

지표	CloudWatch의 지표 이름	설명
`utilization_gpu`	`nvidia_smi_utilization_gpu`	GPU에서 하나 이상의 커널이 실행 중이었던 과거 샘플 기간에 대한 시간 비율입니다.
`temperature_gpu`	`nvidia_smi_temperature_gpu`	코어 GPU 온도(섭씨)입니다.
`power_draw`	`nvidia_smi_power_draw`	전체 보드에 대해 마지막으로 측정된 소비 전력(와트)입니다.
`utilization_memory`	`nvidia_smi_utilization_memory`	글로벌(디바이스) 메모리를 읽거나 쓰는 동안 과거 샘플 기간에 대한 시간 비율입니다.

지표	CloudWatch의 지표 이름	설명
fan_speed	nvidia_smi_fan_speed	디바이스의 팬이 현재 동작하려는 최대 팬 속도의 비율입니다.
memory_total	nvidia_smi_memory_total	보고된 총 메모리(MB)입니다.
memory_used	nvidia_smi_memory_used	사용된 메모리(MB)입니다.
memory_free	nvidia_smi_memory_free	여유 메모리(MB)입니다.
pcie_link_gen_current	nvidia_smi_pcie_link_gen_current	현재 링크 생성입니다.
pcie_link_width_current	nvidia_smi_pcie_link_width_current	현재 링크 폭입니다.
encoder_stats_session_count	nvidia_smi_encoder_stats_session_count	현재 인코더 세션 수입니다.
encoder_stats_average_fps	nvidia_smi_encoder_stats_average_fps	초당 인코딩 프레임의 이동 평균입니다.
encoder_stats_average_latency	nvidia_smi_encoder_stats_average_latency	인코딩 대기 시간(마이크로초)의 이동 평균입니다.
clocks_current_graphics	nvidia_smi_clocks_current_graphics	그래픽(셰이더) 클럭의 현재 주파수입니다.

지표	CloudWatch의 지표 이름	설명
clocks_current_sm	nvidia_smi_clocks_current_sm	스트리밍 멀티프로세서(SM) 클럭의 현재 주파수입니다.
clocks_current_memory	nvidia_smi_clocks_current_memory	메모리 클럭의 현재 주파수입니다.
clocks_current_video	nvidia_smi_clocks_current_video	비디오(인코더 및 디코더) 클럭의 현재 주파수입니다.

이러한 모든 지표는 다음 측정기준으로 수집됩니다.

측정기준	설명
index	이 서버의 GPU의 고유 식별자입니다. 디바이스의 NVIDIA 관리 라이브러리(NVML) 인덱스를 나타냅니다.
name	GPU의 유형입니다. 예제: NVIDIA Tesla A100
host	서버 호스트 이름입니다.

procstat 플러그 인을 사용하여 프로세스 지표 수집

procstat 플러그인을 사용하면 개별 프로세스에서 지표를 수집할 수 있습니다. Linux 서버와 지원되는 Windows Server 버전에서 실행하는 서버에서 사용할 수 있습니다.

주제

procstat를 사용하도록 CloudWatch 에이전트 구성

procstat 플러그 인을 사용하려면 CloudWatch 에이전트 구성 파일의 `metrics_collected` 섹션에 `procstat` 섹션을 추가합니다. 모니터링할 프로세스를 지정하는 방법은 세 가지가 있습니다. 이러한 메서드는 하나만 사용할 수 있지만 해당 메서드를 사용해 모니터링할 프로세스를 여러 개 지정할 수 있습니다.

- `pid_file`: 프로세스를 통해 생성된 프로세스 식별 번호(PID)의 이름으로 프로세스를 선택합니다.
- `exe`: 정규식 일치 규칙을 사용하여 프로세스 이름이 지정한 문자열과 일치하는 프로세스를 선택합니다. 일치는 “포함”일치입니다. 즉, 일치시킬 용어로 `agent`를 지정하면 `cloudwatchagent`와 같은 이름의 프로세스는 해당 용어와 일치합니다. 자세한 내용은 구문을 참조하세요.
- `pattern`: 프로세스를 시작하는 데 사용된 명령줄로 프로세스를 선택합니다. 정규식 일치 규칙을 사용하여 명령줄이 지정한 문자열과 일치하는 프로세스가 모두 선택됩니다. 명령과 함께 사용된 파라미터 및 옵션을 포함하여 전체 명령줄을 확인합니다.

  일치는 “포함”일치입니다. 즉, 일치시킬 용어로 `-c`를 지정하면 `-config`와 같은 파라미터가 있는 프로세스는 해당 용어와 일치합니다.
- `drop_original_metrics` – 선택 사항입니다. `metrics` 섹션의 `aggregation_dimensions` 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정 기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.

위 섹션 중 두 개 이상을 포함했더라도 CloudWatch 에이전트는 이러한 방법 중 하나만 사용합니다. 섹션을 두 개 이상 지정한 경우 CloudWatch 에이전트는 `pid_file` 섹션을 사용합니다(있는 경우). 없는 경우에는 `exe` 섹션을 사용합니다.

Linux 서버의 경우 `exe` 또는 `pattern` 섹션에서 지정한 문자열은 정규식으로 평가됩니다. Windows Server를 실행하는 서버에서 이러한 문자열은 WMI 쿼리로 평가됩니다. 예를 들어 `pattern: "%apache%"`가 됩니다. 자세한 내용은 LIKE 연산자를 참조하세요.

어떤 방법을 사용하든 간에 지표 수집 빈도(초)를 지정하는 선택적 metrics_collection_interval 파라미터를 포함할 수 있습니다. 이 파라미터를 생략하면 기본값인 60초가 사용됩니다.

다음 단원의 예제에서 procstat 섹션은 에이전트 구성 파일의 metrics_collected 섹션에 포함된 유일한 섹션입니다. 실제 구성 파일에서는 metrics_collected에 다른 섹션을 포함할 수도 있습니다. 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하세요.

pid_file을 사용하여 구성

다음 예제 procstat 섹션에서는 PID 파일 example1.pid 및 example2.pid를 생성하는 프로세스를 모니터링합니다. 각 프로세스에서는 여러 지표가 수집됩니다. example2.pid를 생성하는 프로세스에서 수집되는 지표는 10초마다 수집되고, example1.pid 프로세스에서 수집되는 지표는 60초마다 수집됩니다(기본값).

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "pid_file": "/var/run/example1.pid",
                    "measurement": [
                        "cpu_usage",
                        "memory_rss"
                    ]
                },
                {
                    "pid_file": "/var/run/example2.pid",
                    "measurement": [
                        "read_bytes",
                        "read_count",
                        "write_bytes"
                    ],
                    "metrics_collection_interval": 10
                }
            ]
        }
    }
}
```

exe를 사용하여 구성

다음 예제 procstat 섹션은 문자열 agent 또는 plugin과 이름이 일치하는 모든 프로세스를 모니터링합니다. 각 프로세스에서 동일한 지표가 수집됩니다.

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "exe": "agent",
                    "measurement": [
                        "cpu_time",
                        "cpu_time_system",
                        "cpu_time_user"
                    ]
                },
                {
                    "exe": "plugin",
                    "measurement": [
                        "cpu_time",
                        "cpu_time_system",
                        "cpu_time_user"
                    ]
                }
            ]
        }
    }
}
```

pattern을 사용하여 구성

다음 예제 procstat 섹션은 문자열 config 또는 -c와 명령줄이 일치하는 모든 프로세스를 모니터링합니다. 각 프로세스에서 동일한 지표가 수집됩니다.

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "pattern": "config",
                    "measurement": [
                        "rlimit_memory_data_hard",
```

```
                        "rlimit_memory_data_soft",
                        "rlimit_memory_stack_hard",
                        "rlimit_memory_stack_soft"
                    ]
                },
                {
                    "pattern": "-c",
                    "measurement": [
                        "rlimit_memory_data_hard",
                        "rlimit_memory_data_soft",
                        "rlimit_memory_stack_hard",
                        "rlimit_memory_stack_soft"
                    ]
                }
            ]
        }
    }
}
```

procstat가 수집하는 지표

다음 표에는 `procstat` 플러그인을 사용하여 수집할 수 있는 지표가 나열되어 있습니다.

CloudWatch 에이전트는 다음 지표 이름의 시작 부분에 `procstat`를 추가합니다. Linux 서버 또는 Windows Server를 실행하는 서버에서 수집되었는지 여부에 따라 구문이 다릅니다. 예를 들어, `cpu_time` 지표는 Linux에서 수집된 경우 `procstat_cpu_time`으로 표시되고, Windows Server에서 수집된 경우에는 `procstat cpu_time`으로 표시됩니다.

지표 이름	제공 위치	설명
`cpu_time`	Linux	프로세스가 CPU를 사용하는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 수
`cpu_time_guest`	Linux	프로세스가 게스트 모드에 있

지표 이름	제공 위치	설명
		는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음
cpu_time_guest_nice	Linux	Niced 게스트에서 프로세스가 실행되는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음
cpu_time_idle	Linux	프로세스가 유휴 모드에 있는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음

지표 이름	제공 위치	설명
cpu_time_iowait	Linux	프로세스가 I/O 작업이 완료될 때까지 대기하는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음
cpu_time_irq	Linux	프로세스가 서비스 중단되는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음
cpu_time_nice	Linux	프로세스가 양호 모드에 있는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음

지표 이름	제공 위치	설명
cpu_time_soft_irq	Linux	프로세스가 소프트웨어 인터럽트를 제공하는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음
cpu_time_steal	Linux	가상화된 환경에서 실행할 때 다른 운영 체제에서 실행하는 데 소요되는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음

지표 이름	제공 위치	설명
cpu_time_stolen	Linux, Windows Server	프로세스가 도용 시간에 있는 시간입니다. 도용 시간은 가상화 환경에서 다른 운영 체제에서 사용된 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 없음
cpu_time_system	Linux, Windows Server, macOS	프로세스가 시스템 모드에 있는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 유형: Float 단위: 수
cpu_time_user	Linux, Windows Server, macOS	프로세스가 사용자 모드에 있는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 수

지표 이름	제공 위치	설명
cpu_usage	Linux, Windows Server, macOS	어떠한 용량에서든 프로세스가 활성화되어 있는 시간(백분율)입니다. 단위: 백분율
memory_data	Linux, macOS	프로세스가 데이터에 사용하는 메모리의 양입니다. 단위: 바이트
memory_locked	Linux, macOS	프로세스가 잠근 메모리의 양입니다. 단위: 바이트
memory_rss	Linux, Windows Server, macOS	프로세스에서 사용 중인 실제 메모리의 크기(실제 상주 메모리)입니다. 단위: 바이트
memory_stack	Linux, macOS	프로세스에서 사용 중인 스택 메모리의 양입니다. 단위: 바이트

지표 이름	제공 위치	설명
memory_swap	Linux, macOS	프로세스에서 사용 중인 스왑 메모리의 양입니다. 단위: 바이트
memory_vms	Linux, Windows Server, macOS	프로세스에서 사용 중인 가상 메모리의 양입니다. 단위: 바이트
num_fds	Linux	이 프로세스가 열어둔 파일 설명자의 수입니다. 단위: 없음
num_threads	Linux, Windows, macOS	이 프로세스의 스레드 수입니다. 단위: 없음
pid	Linux, Windows Server, macOS	프로세스 식별자(ID)입니다. 단위: 없음

지표 이름	제공 위치	설명
pid_count	Linux, Windows Server, macOS	프로세스와 관련된 프로세스 ID의 수입니다. 이 지표의 전체 이름은 Linux 서버 및 macOS 컴퓨터에서 procstat_lookup_pid_count 이며 Windows Server에서는 procstat_lookup pid_count 입니다. 단위: 없음
read_bytes	Linux, Windows Server	프로세스가 디스크에서 읽은 바이트 수입니다. 단위: 바이트
write_bytes	Linux, Windows Server	프로세스가 디스크에 기록한 바이트 수입니다. 단위: 바이트

지표 이름	제공 위치	설명
read_count	Linux, Windows Server	프로세스에서 실행한 디스크 읽기 작업의 수입니다. 단위: 없음
rlimit_realtime_priority_hard	Linux	이 프로세스에 설정할 수 있는 실시간 우선 순위에 대한 엄격한 제한입니다. 단위: 없음
rlimit_realtime_priority_soft	Linux	이 프로세스에 설정할 수 있는 실시간 우선 순위에 대한 가벼운 제한입니다. 단위: 없음
rlimit_signals_pending_hard	Linux	이 프로세스에서 대기열에 넣을 수 있는 최대 신호 수에 대한 엄격한 제한입니다. 단위: 없음

지표 이름	제공 위치	설명
rlimit_signals_pending_soft	Linux	이 프로세스에서 대기열에 넣을 수 있는 최대 신호 수에 대한 가벼운 제한입니다. 단위: 없음
rlimit_nice_priority_hard	Linux	이 프로세스에서 설정할 수 있는 최대 nice 우선 순위에 대한 엄격한 제한입니다. 단위: 없음
rlimit_nice_priority_soft	Linux	이 프로세스에서 설정할 수 있는 최대 nice 우선 순위에 대한 가벼운 제한입니다. 단위: 없음
rlimit_num_fds_hard	Linux	이 프로세스가 열 수 있는 최대 파일 설명자 수에 대한 엄격한 제한입니다. 단위: 없음

지표 이름	제공 위치	설명
rlimit_num_fds_soft	Linux	이 프로세스가 열 수 있는 최대 파일 설명자 수에 대한 가벼운 제한입니다. 단위: 없음
write_count	Linux, Windows Server	프로세스에서 실행한 디스크 쓰기 작업의 수입니다. 단위: 없음
involuntary_context_switches	Linux	프로세스의 컨텍스트가 강제로 전환된 횟수입니다. 단위: 없음
voluntary_context_switches	Linux	프로세스의 컨텍스트가 자의적으로 전환된 횟수입니다. 단위: 없음
realtime_priority	Linux	프로세스에 대한 실시간 우선 순위의 현재 사용량입니다. 단위: 없음

지표 이름	제공 위치	설명
nice_priority	Linux	프로세스에 대한 nice priority의 현재 사용입니다. 단위: 없음
signals_pending	Linux	프로세스에서 처리하도록 대기 중인 신호 수입니다. 단위: 없음
rlimit_cpu_time_hard	Linux	프로세스에 대한 하드 CPU 시간 리소스 제한입니다. 단위: 없음
rlimit_cpu_time_soft	Linux	프로세스에 대한 소프트 CPU 시간 리소스 제한입니다. 단위: 없음
rlimit_file_locks_hard	Linux	프로세스에 대한 하드 파일 잠금 리소스 제한입니다. 단위: 없음

지표 이름	제공 위치	설명
rlimit_file_locks_soft	Linux	프로세스에 대한 소프트 파일 잠금 리소스 제한입니다. 단위: 없음
rlimit_memory_data_hard	Linux	데이터에 사용되는 메모리와 관련해 프로세스에 대한 하드 리소스 제한입니다. 단위: 바이트
rlimit_memory_data_soft	Linux	데이터에 사용되는 메모리와 관련해 프로세스에 대한 소프트 리소스 제한입니다. 단위: 바이트
rlimit_memory_locked_hard	Linux	잠긴 메모리와 관련해 프로세스에 대한 하드 리소스 제한입니다. 단위: 바이트

지표 이름	제공 위치	설명
rlimit_memory_locked_soft	Linux	잠긴 메모리와 관련해 프로세스에 대한 소프트 리소스 제한입니다. 단위: 바이트
rlimit_memory_rss_hard	Linux	물리적 메모리와 관련해 프로세스에 대한 하드 리소스 제한입니다. 단위: 바이트
rlimit_memory_rss_soft	Linux	물리적 메모리와 관련해 프로세스에 대한 소프트 리소스 제한입니다. 단위: 바이트
rlimit_memory_stack_hard	Linux	프로세스 스택에 대한 하드 리소스 제한입니다. 단위: 바이트
rlimit_memory_stack_soft	Linux	프로세스 스택에 대한 소프트 리소스 제한입니다. 단위: 바이트

지표 이름	제공 위치	설명
rlimit_memory_vms_hard	Linux	가상 메모리와 관련해 프로세스에 대한 하드 리소스 제한입니다. 단위: 바이트
rlimit_memory_vms_soft	Linux	가상 메모리와 관련해 프로세스에 대한 소프트 리소스 제한입니다. 단위: 바이트

CloudWatch 에이전트가 가져온 프로세스 지표 보기

프로세스 지표를 CloudWatch로 가져온 후 이러한 지표를 시계열 그래프로 보고, 해당 지표를 감시하여 지표가 지정된 임곗값을 위반하는 경우 이를 알릴 수 있는 경보를 생성할 수 있습니다. 다음 절차에서는 프로세스 지표를 시계열 그래프로 보는 방법을 보여 줍니다. 경보 설정에 대한 자세한 내용은 Amazon CloudWatch 경보 사용 단원을 참조하세요.

CloudWatch 콘솔에서 프로세스 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 에이전트가 수집한 지표의 네임스페이스를 선택합니다. 기본적으로 이 네임스페이스는 CWAgent이지만, CloudWatch 에이전트 구성 파일에서 다른 네임스페이스를 지정했을 수 있습니다.
4. 지표 측정기준(예: 인스턴스별 지표)을 선택합니다.
5. 모든 지표 탭에 네임스페이스의 해당 측정기준에 대한 모든 지표가 표시됩니다. 다음을 수행할 수 있습니다.

a. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.

b. 테이블을 정렬하려면 열 머리글을 사용합니다.

c. 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.

d. 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.

6. (선택 사항) 이 그래프를 CloudWatch 대시보드에 추가하려면 [작업(Actions)], [대시보드에 추가(Add to dashboard)]를 선택합니다.

StatsD를 사용하여 사용자 지정 지표 검색

StatsD 프로토콜과 함께 CloudWatch 에이전트를 사용하여 애플리케이션 또는 서비스에서 추가 사용자 지정 지표를 검색할 수 있습니다. StatsD는 다양한 애플리케이션에서 지표를 수집할 수 있는 인기 있는 오픈 소스 솔루션입니다. StatsD는 고유한 지표를 계측하는 데 특히 유용합니다. CloudWatch 에이전트와 StatsD를 함께 사용하는 예는 Amazon CloudWatch 에이전트를 사용하여 사용자 지정 애플리케이션 지표를 효과적으로 모니터링하는 방법을 참조하세요.

StatsD는 Windows Server를 실행하는 서버와 Linux 서버에서 모두 지원됩니다. CloudWatch는 다음과 같은 StatsD 형식을 지원합니다.

```
MetricName:value|type|@sample_rate|#tag1:
  value,tag1...
```

- MetricName – 콜론, 막대, # 문자 또는 @ 문자가 없는 문자열입니다.
- value – 정수 또는 부동 소수점일 수 있습니다.
- type – 카운터의 경우 c, 게이지의 경우 g, 타이머의 경우 ms, 히스토그램의 경우 h, 설정의 경우 s를 지정합니다.
- sample_rate – (선택 사항) 0과 1(포함) 사이의 부동 소수점입니다. 카운터, 히스토그램 및 타이머 지표에만 사용하세요. 기본값은 1(시간의 샘플링 100%)입니다.
- tags – (선택 사항) 쉼표로 구분된 태그 목록입니다. StatsD 태그는 CloudWatch의 측정기준과 유사합니다. env:prod와 같이 키/값 태그에 콜론을 사용하세요.

이 형식을 따르는 StatsD 클라이언트를 사용하여 지표를 CloudWatch 에이전트에 전송할 수 있습니다. 사용 가능한 일부 StatsD 클라이언트에 대한 자세한 내용은 GitHub의 StatsD 클라이언트 페이지를 참조하세요.

이러한 사용자 지정 지표를 수집하려면 에이전트 구성 파일의 `metrics_collected` 섹션에 `"statsd": {}` 줄을 추가합니다. 이 줄을 수동으로 추가할 수 있습니다. 마법사를 사용하여 구성 파일을 생성하는 경우 자동으로 수행됩니다. 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.

StatsD 기본 구성은 대부분의 사용자에게 작동합니다. 필요에 따라 에이전트 구성 파일의 statsd 섹션에 추가할 수 있는 선택 사항 필드가 있습니다.

- `service_address` – CloudWatch 에이전트가 수신 대기해야 하는 서비스 주소입니다. 형식은 *`ip`*`:`*`port`*입니다. IP 주소를 생략하면 에이전트가 유효한 인터페이스를 모두 수신합니다. UDP 형식만 지원되므로 UDP 접두사를 지정할 필요가 없습니다.

  기본 값은 `:8125`입니다.
- `metrics_collection_interval` – StatsD 플러그 인이 지표를 실행하고 수집하는 빈도(초)입니다. 기본값은 10초입니다. 범위는 1~172,000입니다.
- `metrics_aggregation_interval` – CloudWatch가 지표를 단일 데이터 요소로 집계하는 빈도(초)입니다. 기본값은 60초입니다.

  예를 들어 `metrics_collection_interval`이 10이고 `metrics_aggregation_interval`이 60이면 CloudWatch는 10초마다 데이터를 수집합니다. 1분 후마다 해당 1분의 6개 데이터 판독값이 단일 데이터 요소에 집계되어 CloudWatch에 전송됩니다.

  범위는 0~172,000입니다. `metrics_aggregation_interval`을 0으로 설정하면 StatsD 지표를 집계할 수 없습니다.
- `allowed_pending_messages` – 대기열에 대기하도록 허용된 UDP 메시지 수입니다. 대기열이 가득 차면 StatsD 서버가 패킷을 삭제하기 시작합니다. 기본값은 10,000입니다.
- `drop_original_metrics` – 선택 사항입니다. `metrics` 섹션의 `aggregation_dimensions` 필드를 사용하여 지표를 집계된 결과로 롤업하는 경우 기본적으로 에이전트는 집계된 지표와 측정 기준의 각 값에 대해 구분된 원래 지표를 모두 전송합니다. 원본 지표를 CloudWatch로 전송하지 않으려면 지표 목록과 함께 이 파라미터를 지정할 수 있습니다. 이 파라미터와 함께 지정된 지표에는 CloudWatch에 보고되는 측정기준별 지표가 없습니다. 대신 집계된 지표만 보고됩니다. 이렇게 하면 에이전트가 수집하는 지표의 수가 줄어들어 비용이 절감됩니다.

다음은 기본값 포트와 사용자 지정 수집 및 집계 간격을 사용하는 에이전트 구성 파일의 statsd 섹션의 예입니다.

```
{
```

```
   "metrics":{
      "metrics_collected":{
         "statsd":{
            "service_address":":8125",
            "metrics_collection_interval":60,
            "metrics_aggregation_interval":300
         }
      }
   }
}
```

CloudWatch 에이전트가 가져온 StatsD 지표 보기

StatsD 지표를 CloudWatch로 가져온 후 이러한 지표를 시계열 그래프로 보고, 해당 지표를 감시하여 지표가 지정된 임곗값을 위반하는 경우 이를 알릴 수 있는 경보를 생성할 수 있습니다. 다음 절차에서는 StatsD 지표를 시계열 그래프로 보는 방법을 보여 줍니다. 경보 설정에 대한 자세한 내용은 Amazon CloudWatch 경보 사용 단원을 참조하세요.

CloudWatch 콘솔에서 StatsD 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 에이전트가 수집한 지표의 네임스페이스를 선택합니다. 기본적으로 이 네임스페이스는 CWAgent이지만, CloudWatch 에이전트 구성 파일에서 다른 네임스페이스를 지정했을 수 있습니다.
4. 지표 측정기준(예: 인스턴스별 지표)을 선택합니다.
5. 모든 지표 탭에 네임스페이스의 해당 측정기준에 대한 모든 지표가 표시됩니다. 다음을 수행할 수 있습니다.

   a. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.

   b. 테이블을 정렬하려면 열 머리글을 사용합니다.

   c. 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.

   d. 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.
6. (선택 사항) 이 그래프를 CloudWatch 대시보드에 추가하려면 [작업(Actions)], [대시보드에 추가(Add to dashboard)]를 선택합니다.

collectd를 사용하여 사용자 지정 지표 검색

Linux 서버에서만 지원되는 collectd 프로토콜과 함께 CloudWatch 에이전트를 사용하여 애플리케이션 또는 서비스에서 추가 지표를 검색할 수 있습니다. collectd 는 매우 다양한 애플리케이션에 대한 시스템 통계를 수집할 수 있는 플러그 인이 포함된 인기 있는 오픈 소스 솔루션입니다. CloudWatch 에이전트가 이미 수집할 수 있는 시스템 지표를 collectd의 추가 지표와 결합하면 시스템 및 애플리케이션을 효과적으로 모니터링하고 분석하며 관련 문제를 해결할 수 있습니다. collectd에 대한 자세한 내용은 collectd - 시스템 통계 수집 데몬을 참조하세요.

collectd 소프트웨어를 사용하여 지표를 CloudWatch 에이전트에 전송합니다. collectd 지표의 경우 CloudWatch 에이전트가 서버 역할을 하며 collectd 플러그 인은 클라이언트 역할을 합니다.

collectd 소프트웨어는 모든 서버에 자동으로 설치되지 않습니다. Amazon Linux 2를 실행하는 서버에서 다음 단계에 따라 collectd를 설치합니다.

```
sudo amazon-linux-extras install collectd
```

다른 시스템에 collectd를 설치하는 방법에 대한 자세한 내용은 collectd 다운로드 페이지를 참조하세요.

이러한 사용자 지정 지표를 수집하려면 에이전트 구성 파일의 metrics_collected 섹션에 "collectd": {} 줄을 추가합니다. 이 줄을 수동으로 추가할 수 있습니다. 마법사를 사용하여 이 구성 파일을 생성하는 경우, 사용자를 위해 이루어집니다. 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.

추가 선택적 파라미터도 사용 가능합니다. collectd를 사용하고 있고 `/etc/collectd/auth_file`를 collectd_auth_file로 사용하지 않는 경우 이러한 선택 사항을 일부 설정해야 합니다.

- service_address: CloudWatch 에이전트가 수신 대기해야 하는 서비스 주소입니다. 형식은 "udp://*ip*:*port*입니다. 기본값은 udp://127.0.0.1:25826입니다.
- name_prefix: 각 collectd 지표의 이름 시작 부분에 부착하는 접두사. 기본값은 `collectd_`입니다. 최대 길이는 255자입니다.
- collectd_security_level: 네트워크 구성의 보안 수준을 설정합니다. 기본값은 encrypt입니다.

  encrypt는 암호화된 데이터만 수락하도록 지정합니다. sign은 서명되고 암호화된 데이터만 수락하도록 지정합니다. none은 모든 데이터를 수락하도록 지정합니다. collectd_auth_file에 값을 지정하는 경우 가능하면 암호화된 데이터가 복호화됩니다.

  자세한 내용은 collectd Wiki의 클라이언트 설정 및 가능한 상호작용을 참조하세요.

- collectd_auth_file 사용자 이름이 비밀번호에 매핑되는 파일을 설정합니다. 이 비밀번호는 서명을 확인하고 암호화된 네트워크 패킷을 복호화하는 데 사용합니다. 제공된 경우 서명된 데이터를 확인하고 암호화된 패킷을 복호화합니다. 그렇지 않은 경우 서명을 확인하지 않아도 서명된 데이터가 수락 완료되어 암호화된 데이터를 복호화할 수 없습니다.

  기본값은 `/etc/collectd/auth_file`입니다.

  collectd_security_level이 none으로 설정된 경우 이것은 선택 사항입니다. collectd_security_level을 `encrypt` 또는 sign으로 설정한 경우 collectd_auth_file을 지정해야 합니다.

  auth 파일 형식은 각 줄에 사용자 이름 다음에 콜론이 나오고 스페이스 다음에 비밀번호가 나옵니다. 예:

  ```
  user1: user1_password

  user2: user2_password
  ```

- collectd_typesdb: 데이터 세트 설명을 포함하는 1개 이상의 파일 목록. 이 목록은 목록에 항목이 1개만 있어도 괄호로 묶어야 합니다. 이 목록의 각 항목은 큰따옴표로 묶어야 합니다. 여러 항목이 있는 경우 각각 쉼표로 구분하세요. Linux 서버에서 기본값은 `["/usr/share/collectd/types.db"]`입니다. macOs 컴퓨터에서 기본값은 collectd의 버전에 따라 다릅니다. 예를 들면 `["/usr/local/Cellar/collectd/5.12.0/share/collectd/types.db"]`입니다.

  자세한 내용은 https://www.collectd.org/documentation/manpages/types.db.html 단원을 참조하십시오.

- metrics_aggregation_interval: CloudWatch가 지표를 단일 데이터 요소로 집계하는 빈도(초)입니다. 기본값은 60초입니다. 범위는 0 ~ 172,000입니다. 이것을 0에 설정하면 collectd 지표를 집계할 수 없습니다.

다음은 에이전트 구성 파일의 collectd 섹션의 예입니다.

```
{
   "metrics":{
      "metrics_collected":{
         "collectd":{
            "name_prefix":"My_collectd_metrics_",
            "metrics_aggregation_interval":120
         }
      }
   }
```

```
}
```

CloudWatch 에이전트가 가져온 collectd 지표 보기

수집된 지표를 CloudWatch로 가져온 후 이러한 지표를 시계열 그래프로 보고, 해당 지표를 감시하여 지표가 지정된 임곗값을 위반하는 경우 이를 알릴 수 있는 경보를 생성할 수 있습니다. 다음 절차에서는 collectd 지표를 시계열 그래프로 보는 방법을 보여 줍니다. 경보 설정에 대한 자세한 내용은 Amazon CloudWatch 경보 사용 단원을 참조하세요.

CloudWatch 콘솔에서 collectd 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 에이전트가 수집한 지표의 네임스페이스를 선택합니다. 기본적으로 이 네임스페이스는 CWAgent이지만, CloudWatch 에이전트 구성 파일에서 다른 네임스페이스를 지정했을 수 있습니다.
4. 지표 측정기준(예: 인스턴스별 지표)을 선택합니다.
5. 모든 지표 탭에 네임스페이스의 해당 측정기준에 대한 모든 지표가 표시됩니다. 다음을 수행할 수 있습니다.
   a. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택합니다.
   b. 테이블을 정렬하려면 열 머리글을 사용합니다.
   c. 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.
   d. 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.
6. (선택 사항) 이 그래프를 CloudWatch 대시보드에 추가하려면 [작업(Actions)], [대시보드에 추가(Add to dashboard)]를 선택합니다.

Amazon EC2 인스턴스에서 Prometheus 지표 수집 설정 및 구성

다음 단원에서는 EC2 인스턴스에 Prometheus 모니터링이 포함된 CloudWatch 에이전트를 설치하는 방법과 추가 대상을 스크레이프하도록 에이전트를 구성하는 방법을 설명합니다. 또한 Prometheus 모니터링을 통해 테스트하는 데 사용할 샘플 워크로드를 설정하기 위한 튜토리얼도 제공합니다.

CloudWatch 에이전트에서 지원하는 운영 체제에 대한 자세한 내용은 CloudWatch 에이전트를 사용하여 지표, 로그, 추적 수집 단원을 참조하세요.

VPC 보안 그룹 요구 사항

VPC를 사용하는 경우 다음 요구 사항이 적용됩니다.

- Prometheus 워크로드의 보안 그룹 수신 규칙은 프라이빗 IP로 Prometheus 지표를 스크레이프하기 위해 CloudWatch 에이전트에 대한 Prometheus 포트를 열어야 합니다.
- CloudWatch 에이전트의 보안 그룹 송신 규칙은 CloudWatch 에이전트가 프라이빗 IP로 Prometheus 워크로드의 포트에 연결할 수 있도록 허용해야 합니다.

주제

1단계: CloudWatch 에이전트 설치

첫 번째 단계는 EC2 인스턴스에 CloudWatch 에이전트를 설치하는 것입니다. 지침은 CloudWatch 에이전트 설치 섹션을 참조하세요.

2단계: Prometheus 소스 스크레이프 및 지표 가져오기

Prometheus 모니터링이 포함된 CloudWatch 에이전트는 Prometheus 지표를 스크레이프하는 데 두 가지 구성이 필요합니다. 하나는 Prometheus 설명서의 <scrape_config>에 설명된 표준 Prometheus 구성을 위한 것입니다. 다른 하나는 CloudWatch 에이전트 구성을 위한 것입니다.

Prometheus 스크레이프 구성

CloudWatch 에이전트는 Prometheus 설명서의 <scrape_config>에 설명된 대로 표준 Prometheus 스크레이프 구성을 지원합니다. 이 섹션을 편집하여 이 파일에 이미 있는 구성을 업데이트하고 Prometheus 스크레이핑 대상을 더 추가할 수 있습니다. 샘플 구성 파일에는 다음과 같은 글로벌 구성 줄이 포함되어 있습니다.

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
- job_name: MY_JOB
  sample_limit: 10000
  file_sd_configs:
```

```
    - files: ["C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\prometheus_sd_1.yaml",
 "C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\prometheus_sd_2.yaml"]
```

global 섹션에서는 모든 구성 컨텍스트에서 유효한 파라미터를 지정합니다. 이러한 파라미터는 다른 구성 섹션의 기본값으로도 사용됩니다. 여기에는 다음 파라미터가 포함됩니다.

- scrape_interval - 대상을 스크레이프하는 빈도를 정의합니다.
- scrape_timeout - 스크레이프 요청 시간이 초과되기 전에 대기할 시간을 정의합니다.

scrape_configs 섹션에서는 대상 세트 및 해당 대상을 스크레이프하는 방법을 정의하는 파라미터를 지정합니다. 여기에는 다음 파라미터가 포함됩니다.

- job_name - 기본적으로 스크레이프한 지표에 할당된 작업 이름입니다.
- sample_limit - 허용될 스크레이프한 샘플 수에 대한 스크레이프당 제한입니다.
- file_sd_configs - 파일 서비스 검색 구성 목록입니다. 0개 이상의 정적 구성 목록이 포함된 파일 세트를 읽습니다. file_sd_configs 섹션에는 대상 그룹이 추출되는 파일의 패턴을 정의하는 files 파라미터가 포함되어 있습니다.

CloudWatch 에이전트는 다음과 같은 서비스 검색 구성 유형을 지원합니다.

static_config 대상 목록 및 해당 대상의 공통 레이블 세트를 지정할 수 있습니다. 스크레이프 구성에서 정적 대상을 지정하는 표준 방법입니다.

다음은 로컬 호스트에서 Prometheus 지표를 스크레이프하는 정적 구성 샘플입니다. 에이전트가 실행되는 서버에 대해 Prometheus 포트가 열려 있는 경우 다른 서버에서도 지표를 스크레이프할 수 있습니다.

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_sd_1.yaml
- targets:
    - 127.0.0.1:9404
  labels:
    key1: value1
    key2: value2
```

이 예에는 다음 파라미터가 포함되어 있습니다.

- targets - 정적 구성에 의해 스크레이프되는 대상입니다.
- labels - 대상에서 스크레이프한 모든 지표에 할당되는 레이블입니다.

ec2_sd_config Amazon EC2 인스턴스에서 스크레이프 대상을 검색할 수 있습니다. 다음은 EC2 인스턴스 목록에서 Prometheus 지표를 스크레이프하는 ec2_sd_config 샘플입니다. 이러한 인스턴스의 Prometheus 포트는 CloudWatch 에이전트가 실행되는 서버에 대해 열려 있어야 합니다. CloudWatch 에이전트가 실행되는 EC2 인스턴스의 IAM 역할에는 ec2:DescribeInstance 권한이 포함되어 있어야 합니다. 예를 들어 관리형 정책 AmazonEC2ReadOnlyAccess를 CloudWatch 에이전트가 실행되는 인스턴스에 연결할 수 있습니다.

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: MY_JOB
    sample_limit: 10000
    ec2_sd_configs:
      - region: us-east-1
        port: 9404
        filters:
          - name: instance-id
            values:
              - i-98765432109876543
              - i-12345678901234567
```

이 예에는 다음 파라미터가 포함되어 있습니다.

- region - 대상 EC2 인스턴스가 있는 AWS 리전입니다. 이 값을 비워 두면 인스턴스 메타데이터의 리전이 사용됩니다.
- port - 지표를 스크레이프할 포트입니다.
- filters - 인스턴스 목록을 필터링하는 데 사용할 필터입니다(선택 사항). 이 예에서는 EC2 인스턴스 ID를 기준으로 필터링합니다. 필터링에 사용할 수 있는 다른 기준에 대해서는 DescribeInstances 단원을 참조하세요.

Prometheus에 대한 CloudWatch 에이전트 구성

CloudWatch 에이전트 구성 파일에는 logs와 metrics_collected 모두에 prometheus 섹션이 포함되어 있습니다. 이 섹션에는 다음 파라미터가 포함됩니다.

- cluster_name - 로그 이벤트에서 레이블로 추가할 클러스터 이름을 지정합니다. 이 필드는 선택 사항입니다.

- log_group_name - 스크레이프한 Prometheus 지표의 로그 그룹 이름을 지정합니다.
- prometheus_config_path - Prometheus 스크레이프 구성 파일 경로를 지정합니다.
- emf_processor - 임베디드 지표 형식 프로세서 구성을 지정합니다. 포함된 지표 형식에 대한 자세한 내용은 로그 내에 지표 포함 단원을 참조하세요.

  `emf_processor` 섹션에는 다음 파라미터가 포함될 수 있습니다.

  - metric_declaration_dedup - true로 설정하면 임베디드 지표 형식 지표에 대한 중복 제거 기능이 사용됩니다.
  - metric_namespace - 내보낸 CloudWatch 지표의 지표 네임스페이스를 지정합니다.
  - metric_unit - 지표 이름:지표 단위 맵을 지정합니다. 지원되는 지표 단위에 대한 자세한 내용은 MetricDatum 단원을 참조하세요.
  - metric_declaration - 생성할 임베디드 지표 형식이 있는 로그 배열을 지정하는 섹션입니다. CloudWatch 에이전트가 기본적으로 가져오는 각 Prometheus 소스에 대한 `metric_declaration` 섹션이 있습니다. 이러한 섹션에는 각각 다음 필드가 포함됩니다.
    - `source_labels`는 `label_matcher` 줄에 의해 확인되는 레이블의 값을 지정합니다.
    - `label_matcher`는 `source_labels`에 나열된 레이블의 값을 확인하는 정규 표현식입니다. 일치하는 지표는 CloudWatch에 전송된 임베디드 지표 형식에 포함할 수 있습니다.
    - `metric_selectors`는 수집하여 CloudWatch에 보낼 지표를 지정하는 정규 표현식입니다.
    - `dimensions`는 선택한 각 지표의 CloudWatch 측정기준으로 사용할 레이블 목록입니다.

다음은 Prometheus에 대한 CloudWatch 에이전트 구성의 예입니다.

```
{
   "logs":{
      "metrics_collected":{
         "prometheus":{
            "cluster_name":"prometheus-cluster",
            "log_group_name":"Prometheus",
            "prometheus_config_path":"C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\prometheus.yaml",
            "emf_processor":{
               "metric_declaration_dedup":true,
               "metric_namespace":"CWAgent-Prometheus",
               "metric_unit":{
                  "jvm_threads_current": "Count",
                  "jvm_gc_collection_seconds_sum": "Milliseconds"
               },
```

```
               "metric_declaration":[
                  {
                     "source_labels":[
                        "job", "key2"
                     ],
                     "label_matcher":"MY_JOB;^value2",
                     "dimensions":[
                        [
                           "key1", "key2"
                        ],
                        [
                           "key2"
                        ]
                     ],
                     "metric_selectors":[
                        "^jvm_threads_current$",
                        "^jvm_gc_collection_seconds_sum$"
                     ]
                  }
               ]
            }
         }
      }
   }
}
```

위 예에서는 다음 조건이 충족될 경우 임베디드 지표 형식 섹션을 로그 이벤트로 전송하도록 구성합니다.

- job 레이블의 값은 MY_JOB입니다.
- key2 레이블의 값은 value2입니다.
- Prometheus 지표 jvm_threads_current 및 jvm_gc_collection_seconds_sum에는 job 레이블과 key2 레이블이 모두 포함되어 있습니다.

전송되는 로그 이벤트에는 다음과 같이 강조 표시된 섹션이 포함됩니다.

```
{
    "CloudWatchMetrics": [
        {
            "Metrics": [
                {
```

```
                    "Unit": "Count",
                    "Name": "jvm_threads_current"
                },
                {
                    "Unit": "Milliseconds",
                    "Name": "jvm_gc_collection_seconds_sum"
                }
            ],
            "Dimensions": [
                [
                    "key1",
                    "key2"
                ],
                [
                    "key2"
                ]
            ],
            "Namespace": "CWAgent-Prometheus"
        }
    ],
    "ClusterName": "prometheus-cluster",
    "InstanceId": "i-0e45bd06f196096c8",
    "Timestamp": "1607966368109",
    "Version": "0",
    "host": "EC2AMAZ-PDDOIUM",
    "instance": "127.0.0.1:9404",
    "jvm_threads_current": 2,
    "jvm_gc_collection_seconds_sum": 0.006000000000000002,
    "prom_metric_type": "gauge",
    ...
}
```

예: Prometheus 지표 테스트를 위한 Java/JMX 샘플 워크로드 설정

JMX Exporter는 JMX mBeans를 스크레이프하여 노출할 수 있는 공식 Prometheus 익스포터입니다. 자세한 내용은 prometheus/jmx_exporter를 참조하세요.

CloudWatch 에이전트는 EC2 인스턴스의 JMX Exporter를 통해 Java 가상 머신(JVM), Hjava, Tomcat(Catalina)에서 미리 정의된 Prometheus 지표를 수집할 수 있습니다.

1단계: CloudWatch 에이전트 설치

첫 번째 단계는 EC2 인스턴스에 CloudWatch 에이전트를 설치하는 것입니다. 지침은 CloudWatch 에이전트 설치 섹션을 참조하세요.

2단계: Java/JMX 워크로드 시작

다음 단계는 Java/JMX 워크로드를 시작하는 것입니다.

먼저, prometheus/jmx_exporter에서 최신 JMX Exporter jar 파일을 다운로드합니다.

샘플 애플리케이션에 jar 사용

다음 단원의 명령 예에서는 `SampleJavaApplication-1.0-SNAPSHOT.jar`를 jar 파일로 사용합니다. 명령의 이 부분을 자체 애플리케이션의 jar로 바꿉니다.

JMX Exporter 구성 준비

`config.yaml` 파일은 JMX Exporter 구성 파일입니다. 자세한 내용은 JMX Exporter 설명서의 구성을 참조하세요.

다음은 Java 및 Tomcat의 구성 샘플입니다.

```
---
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
```

```
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

Prometheus Exporter로 Java 애플리케이션 시작

샘플 애플리케이션을 시작합니다. 그러면 Prometheus 지표가 포트 9404로 내보내집니다. 진입점 `com.gubupt.sample.app.App`을 샘플 Java 애플리케이션의 올바른 정보로 바꿔야 합니다.

Linux에서 다음 명령을 입력합니다.

```
$ nohup java -javaagent:./jmx_prometheus_javaagent-0.14.0.jar=9404:./config.yaml -cp
  ./SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App &
```

Windows에서 다음 명령을 입력합니다.

```
PS C:\> java -javaagent:.\jmx_prometheus_javaagent-0.14.0.jar=9404:.\config.yaml -cp  .
\SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App
```

Prometheus 지표 내보내기 확인

Prometheus 지표가 내보내지고 있는지 확인합니다.

Linux에서 다음 명령을 입력합니다.

```
$ curl localhost:9404
```

Windows에서 다음 명령을 입력합니다.

```
PS C:\> curl  http://localhost:9404
```

Linux에서의 샘플 출력:

```
StatusCode        : 200
StatusDescription : OK
Content           : # HELP jvm_classes_loaded The number of classes that are currently
 loaded in the JVM
                    # TYPE jvm_classes_loaded gauge
                    jvm_classes_loaded 2526.0
                    # HELP jvm_classes_loaded_total The total number of class...
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 71908
                    Content-Type: text/plain; version=0.0.4; charset=utf-8
                    Date: Fri, 18 Dec 2020 16:38:10 GMT

                    # HELP jvm_classes_loaded The number of classes that are
 currentl...
Forms             : {}
Headers           : {[Content-Length, 71908], [Content-Type, text/plain; version=0.0.4;
 charset=utf-8], [Date, Fri, 18
                    Dec 2020 16:38:10 GMT]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : System.__ComObject
RawContentLength  : 71908
```

3단계: Prometheus 지표를 스크레이프하도록 CloudWatch 에이전트 구성

다음으로, CloudWatch 에이전트 구성 파일에서 Prometheus 스크레이프 구성을 설정합니다.

Java/JMX 예의 Prometheus 스크레이프 구성을 설정하려면

1. `file_sd_config` 및 `static_config`의 구성을 설정합니다.

   Linux에서 다음 명령을 입력합니다.

   ```
   $ cat /opt/aws/amazon-cloudwatch-agent/var/prometheus.yaml
   global:
     scrape_interval: 1m
     scrape_timeout: 10s
   scrape_configs:
     - job_name: jmx
       sample_limit: 10000
       file_sd_configs:
         - files: [ "/opt/aws/amazon-cloudwatch-agent/var/prometheus_file_sd.yaml" ]
   ```

   Windows에서 다음 명령을 입력합니다.

   ```
   PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
   global:
     scrape_interval: 1m
     scrape_timeout: 10s
   scrape_configs:
     - job_name: jmx
       sample_limit: 10000
       file_sd_configs:
         - files: [ "C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
   \prometheus_file_sd.yaml" ]
   ```

2. 스크레이프 대상 구성을 설정합니다.

   Linux에서 다음 명령을 입력합니다.

   ```
   $ cat /opt/aws/amazon-cloudwatch-agent/var/prometheus_file_sd.yaml
   - targets:
     - 127.0.0.1:9404
     labels:
       application: sample_java_app
       os: linux
   ```

   Windows에서 다음 명령을 입력합니다.

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_file_sd.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    application: sample_java_app
    os: windows
```

3. ec2_sc_config의 Prometheus 스크레이프 구성을 설정합니다. *your-ec2-instance-id*를 올바른 EC2 인스턴스 ID로 바꿉니다.

   Linux에서 다음 명령을 입력합니다.

```
$ cat .\prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: jmx
    sample_limit: 10000
    ec2_sd_configs:
      - region: us-east-1
        port: 9404
        filters:
          - name: instance-id
            values:
              - your-ec2-instance-id
```

   Windows에서 다음 명령을 입력합니다.

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_file_sd.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    application: sample_java_app
    os: windows
```

4. CloudWatch 에이전트 구성을 설정합니다. 먼저 올바른 디렉터리로 이동합니다. Linux의 경우 `/opt/aws/amazon-cloudwatch-agent/var/cwagent-config.json`입니다. Windows의 경우 `C:\ProgramData\Amazon\AmazonCloudWatchAgent\cwagent-config.json`입니다.

다음은 Java/JHX Prometheus 지표가 정의된 샘플 구성입니다. *path-to-Prometheus-Scrape-Configuration-file*을 올바른 경로로 바꿔야 합니다.

```
{
  "agent": {
    "region": "us-east-1"
  },
  "logs": {
    "metrics_collected": {
      "prometheus": {
        "cluster_name": "my-cluster",
        "log_group_name": "prometheus-test",
        "prometheus_config_path": "path-to-Prometheus-Scrape-Configuration-file",
        "emf_processor": {
          "metric_declaration_dedup": true,
          "metric_namespace": "PrometheusTest",
          "metric_unit":{
            "jvm_threads_current": "Count",
            "jvm_classes_loaded": "Count",
            "java_lang_operatingsystem_freephysicalmemorysize": "Bytes",
            "catalina_manager_activesessions": "Count",
            "jvm_gc_collection_seconds_sum": "Seconds",
            "catalina_globalrequestprocessor_bytesreceived": "Bytes",
            "jvm_memory_bytes_used": "Bytes",
            "jvm_memory_pool_bytes_used": "Bytes"
          },
          "metric_declaration": [
            {
              "source_labels": ["job"],
              "label_matcher": "^jmx$",
              "dimensions": [["instance"]],
              "metric_selectors": [
                "^jvm_threads_current$",
                "^jvm_classes_loaded$",
                "^java_lang_operatingsystem_freephysicalmemorysize$",
                "^catalina_manager_activesessions$",
                "^jvm_gc_collection_seconds_sum$",
                "^catalina_globalrequestprocessor_bytesreceived$"
              ]
            },
            {
              "source_labels": ["job"],
```

```
              "label_matcher": "^jmx$",
              "dimensions": [["area"]],
              "metric_selectors": [
                "^jvm_memory_bytes_used$"
              ]
            },
            {
              "source_labels": ["job"],
              "label_matcher": "^jmx$",
              "dimensions": [["pool"]],
              "metric_selectors": [
                "^jvm_memory_pool_bytes_used$"
              ]
            }
          ]
        }
      }
    },
    "force_flush_interval": 5
  }
}
```

5. 다음 명령 중 하나를 입력하여 CloudWatch 에이전트를 다시 시작합니다.

   Linux에서 다음 명령을 입력합니다.

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
   config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/var/cwagent-config.json
   ```

   Windows에서 다음 명령을 입력합니다.

   ```
   & "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
    -a fetch-config -m ec2 -s -c file:C:\ProgramData\Amazon\AmazonCloudWatchAgent
   \cwagent-config.json
   ```

Prometheus 지표 및 로그 보기

이제 수집 중인 Java/JMX 지표를 볼 수 있습니다.

샘플 Java/JMX 워크로드에 대한 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 클러스터가 실행되고 있는 리전에서 왼쪽 탐색 창의 [지표(Metrics)]를 선택합니다. PrometheusTest 네임스페이스를 찾아 지표를 확인합니다.
3. CloudWatch Logs 이벤트를 보려면 탐색 창에서 [로그 그룹(Log groups)]을 선택합니다. 이벤트는 로그 그룹 prometheus-test에 있습니다.

Amazon CloudWatch Observability EKS 추가 기능을 사용하여 CloudWatch 에이전트 설치

Amazon CloudWatch Observability EKS 추가 기능은 Amazon EKS 클러스터에 CloudWatch 에이전트와 Fluent-Bit 에이전트를 설치하며, Container Insights는 Amazon EKS와 CloudWatch Application Signals에 대한 향상된 관찰성을 기본적으로 활성화합니다. 추가 기능을 사용하면 Amazon EKS 클러스터에서 인프라 지표, 애플리케이션 성능 텔레메트리 및 컨테이너 로그를 수집할 수 있습니다.

Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights를 사용하면 Container Insights 지표는 저장된 지표나 수집된 로그별로 요금이 부과되는 대신 관찰당 요금이 부과됩니다. Application Signals의 경우 결제는 애플리케이션에 대한 인바운드 요청, 애플리케이션의 아웃바운드 요청, 구성된 각 서비스 수준 목표(SLO)를 기준으로 합니다. 수신된 각 인바운드 요청은 하나의 애플리케이션 신호를 생성하고, 각 아웃바운드 요청은 하나의 애플리케이션 신호를 생성합니다. 모든 SLO는 측정 기간당 두 개의 애플리케이션 신호를 생성합니다. CloudWatch 요금에 대한 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

Amazon EKS 추가 기능을 사용하면 Amazon EKS 클러스터의 Linux 및 Windows 워커 노드 모두에서 Container Insights를 사용할 수 있습니다. Windows에서 Container Insights를 활성화하려면 Amazon EKS 추가 기능 버전 1.5.0 이상을 사용해야 합니다. Amazon EKS 클러스터의 Windows에서는 Application Signals가 현재 지원되지 않습니다.

Amazon CloudWatch Observability EKS 추가 기능은 Kubernetes 버전 1.23 이상을 실행하는 Amazon EKS 클러스터에서 지원됩니다.

추가 기능을 설치할 때 CloudWatch 에이전트가 지표, 로그 및 트레이스를 CloudWatch에 전송할 수 있도록 IAM 권한을 부여해야 합니다. 이렇게 하는 방법은 두 가지입니다.

- 작업자 노드의 IAM 역할에 정책을 연결합니다. 이 옵션은 워커 노드에 CloudWatch로 원격 분석을 전송할 수 있는 권한을 부여합니다.
- 에이전트 포드에 대해 서비스 계정의 IAM 역할을 사용하고 이 역할에 정책을 연결합니다. 이는 Amazon EKS 클러스터에만 적용됩니다. 이 옵션을 사용하면 CloudWatch가 해당 에이전트 포드에만 액세스할 수 있습니다.

옵션 1: 워커 노드에 IAM 권한으로 설치

이 방법을 사용하려면 먼저 다음 명령을 입력하여 CloudWatchAgentServerPolicy IAM 정책을 워커 노드에 연결합니다. 이 명령에서는 *my-worker-node-role*을 Kubernetes 워커 노드에서 사용하는 IAM 역할로 대체합니다.

```
aws iam attach-role-policy \
--role-name my-worker-node-role \
--policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
```

그런 다음 Amazon CloudWatch Observability EKS 추가 기능을 사용하여 설치합니다. 추가 기능을 설치하려면 AWS CLI, 콘솔, AWS CloudFormation 또는 Terraform을 사용할 수 있습니다.

AWS CLI

AWS CLI를 사용하여 Amazon CloudWatch Observability EKS 추가 기능 설치

다음 명령을 입력합니다. *my-cluster-name*를 클러스터 이름으로 바꿉니다.

```
aws eks create-addon --addon-name amazon-cloudwatch-observability --cluster-name my-cluster-name
```

Amazon EKS console

Amazon EKS 콘솔을 사용하여 Amazon CloudWatch Observability EKS 추가 기능 설치

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. Amazon CloudWatch Observabiilty EKS 추가 기능을 구성할 클러스터의 이름을 선택합니다.
4. 추가 기능(Add-ons) 탭을 선택합니다.
5. 추가 기능 더 가져오기를 선택합니다.
6. 추가 기능 선택 페이지에서 다음을 수행합니다.
   a. Amazon EKS 추가 기능 섹션에서 Amazon CloudWatch Observability 확인란을 선택합니다.
   b. 다음을 선택합니다.
7. 선택한 추가 기능 설정 구성 페이지에서 다음을 수행합니다.
   a. 사용할 버전(Version)을 선택합니다.

b. IAM 역할 선택의 노드에서 상속을 선택합니다.

c. (선택 사항) 선택적 구성 설정을 확장할 수 있습니다. 충돌 해결 방법에서 재정의를 선택한 경우 기존 추가 기능에 대한 하나 이상의 설정을 Amazon EKS 추가 기능의 설정으로 덮어쓸 수 있습니다. 이 옵션을 사용 설정하지 않고 기존 설정과 충돌이 있는 경우 작업이 실패합니다. 결과 오류 메시지를 사용하여 충돌을 해결할 수 있습니다. 이 옵션을 선택하기 전에 Amazon EKS 추가 기능이 사용자가 자체 관리해야 하는 설정을 관리하지 않는지 확인하세요.

d. 다음을 선택합니다.

8. 검토 및 추가 페이지에서 생성을 선택합니다. 추가 기능 설치가 완료되면 설치한 추가 기능이 표시됩니다.

AWS CloudFormation

AWS CloudFormation을 사용하여 Amazon CloudWatch Observability EKS 추가 기능 설치

*my-cluster-name*를 클러스터 이름으로 바꿉니다. 자세한 내용은 AWS::EKS::Addon을 참조하세요.

```
{
    "Resources": {
        "EKSAddOn": {
            "Type": "AWS::EKS::Addon",
            "Properties": {
                "AddonName": "amazon-cloudwatch-observability",
                "ClusterName": "my-cluster-name"
            }
        }
    }
}
```

Terraform

Terraform을 사용하여 Amazon CloudWatch Observability EKS 추가 기능 설치

*my-cluster-name*를 클러스터 이름으로 바꿉니다. 자세한 내용은 리소스: aws_eks_addon를 참조하세요.

```
resource "aws_eks_addon" "example" {
  addon_name   = "amazon-cloudwatch-observability"
```

```
  cluster_name = "my-cluster-name"
}
```

옵션 2: IAM 서비스 계정 역할을 사용하여 설치

이 방법을 사용하기 전에 다음과 같은 사전 요구 사항을 확인하세요.

- Container Insights를 지원하는 AWS 리전 중 하나에 노드가 연결되어 있는 Amazon EKS 클러스터가 작동 중입니다. 지원되는 리전 목록은 Container Insights 단원을 참조하세요.
- 클러스터에 대해 kubectl가 설치 및 구성되어 있습니다. 자세한 내용은 Amazon EKS 사용 설명서의 kubectl 설치를 참조하세요.
- eksctl이 설치되어 있습니다. 자세한 내용은 Amazon EKS 사용 설명서의 eksctl설치 또는 업데이트 섹션을 참조하세요.

IAM 서비스 계정 역할을 사용하여 Amazon CloudWatch Observability EKS 추가 기능 설치

1. 클러스터에 아직 OpenID Connect(OIDC) 공급자가 없는 경우 다음 명령을 입력하여 해당 공급자를 생성합니다. 자세한 내용은 Amazon EKS 사용자 가이드의 IAM 역할을 가정하는 Kubernetes 서비스 계정 구성을 참조하세요.

   ```
   eksctl utils associate-iam-oidc-provider --cluster my-cluster-name --approve
   ```

2. 다음 명령을 입력하여 CloudWatchAgentServerPolicy 정책이 연결된 IAM 역할을 생성하고 OIDC를 사용하여 해당 역할을 수임하도록 에이전트 서비스 계정을 구성합니다. *my-cluster-name*을 클러스터 이름으로, *my-service-account-role*을 서비스 계정을 연결할 역할의 이름으로 바꿉니다. 역할이 아직 없는 경우 eksctl이 자동으로 생성합니다.

   ```
   eksctl create iamserviceaccount \
     --name cloudwatch-agent \
     --namespace amazon-cloudwatch --cluster my-cluster-name \
     --role-name my-service-account-role \
     --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
     --role-only \
     --approve
   ```

3. 다음 명령을 입력하여 추가 기능을 설치합니다. *my-cluster-name*을 클러스터 이름으로, *111122223333*을 계정 ID로, *my-service-account-role*을 이전 단계에서 생성한 IAM 역할로 바꿉니다.

```
aws eks create-addon --addon-name amazon-cloudwatch-observability --cluster-
name my-cluster-name --service-account-role-arn arn:aws:iam::111122223333:role/my-
service-account-role
```

(선택 사항) 추가 구성

컨테이너 로그 수집 옵트아웃

기본적으로 추가 기능은 Fluent Bit를 사용하여 모든 포드에서 컨테이너 로그를 수집한 다음 로그를 CloudWatch Logs로 보냅니다. 수집되는 로그에 대한 자세한 내용은 Fluent Bit 설정을 참조하세요.

컨테이너 로그 수집을 옵트아웃하려면 추가 기능을 생성하거나 업데이트할 때 다음 옵션을 전달하세요.

```
--configuration-values '{ "containerLogs": { "enabled": false } }'
```

NVIDIA GPU 메트릭 수집에서 옵트아웃

CloudWatch 에이전트 버전 1.300034.0부터 Container Insights는 기본적으로 EKS 워크로드에서 NVIDIA GPU 지표를 수집합니다. 이러한 지표는 NVIDIA GPU 지표의 표에 나열되어 있습니다.

CloudWatch 에이전트 구성 파일의 `accelerated_compute_metrics` 옵션을 `false`로 설정하여 NVIDIA GPU 지표 수집을 옵트아웃할 수 있습니다. 이 옵션은 CloudWatch 구성 파일의 `metrics_collected`섹션의 `kubernetes` 섹션에 있습니다. 다음은 옵트아웃 구성의 예입니다.

```
{
  "agent": {
    "region": "us-east-1"
  },
  "logs": {
    "metrics_collected": {
      "emf": {
      },
      "kubernetes": {
        "enhanced_container_insights": true,
        "accelerated_compute_metrics": false
      }
    },
    "force_flush_interval": 5,
```

```
  }
}
```

사용자 지정 CloudWatch 에이전트 구성 사용

CloudWatch 에이전트를 사용하여 다른 지표, 로그 또는 트레이스를 수집하려면 Container Insights와 CloudWatch Application Signals를 활성화한 상태로 유지하면서 사용자 지정 구성을 지정할 수 있습니다. 이렇게 하려면 EKS 추가 기능을 생성하거나 업데이트할 때 사용할 수 있는 고급 구성의 에이전트 키 아래에 있는 구성 키 내에 CloudWatch 에이전트 구성 파일을 포함합니다. 다음은 추가 구성을 제공하지 않은 경우의 기본 에이전트 구성을 나타냅니다.

> **Important**
>
> 추가 구성 설정을 사용하여 제공하는 모든 사용자 지정 구성은 에이전트가 사용하는 기본 구성보다 우선합니다. 향상된 관찰성을 갖춘 Container Insights 및 CloudWatch Application Signals와 같이 기본적으로 활성화되는 기능을 실수로 비활성화하지 않도록 주의합니다. 사용자 지정 에이전트 구성을 제공해야 하는 시나리오에서는 다음 기본 구성을 기준으로 사용하고 그에 따라 수정하는 것이 좋습니다.

```
--configuration-values '{
  "agent": {
    "config": {
      "logs": {
        "metrics_collected": {
          "app_signals": {},
          "kubernetes": {
            "enhanced_container_insights": true
          }
        }
      },
      "traces": {
        "traces_collected": {
          "app_signals": {}
        }
      }
    }

}'
```

다음 예시는 Windows의 CloudWatch 에이전트에 대한 기본 에이전트 구성을 보여줍니다. Windows의 CloudWatch 에이전트는 사용자 지정 구성을 지원하지 않습니다.

```
{
  "logs": {
    "metrics_collected": {
      "kubernetes": {
        "enhanced_container_insights": true
      },
    }
  }
}
```

승인 웹후크 TLS 인증서 관리

Amazon CloudWatch Observability EKS 추가 기능은 Kubernetes 승인 웹후크를 활용하여 `AmazonCloudWatchAgent` 및 `Instrumentation` 사용자 지정 리소스(CR) 요청과 클러스터에 대한 Kubernetes 포드 요청(CloudWatch Application Signals가 활성화된 경우)을 검증하고 변경합니다. Kubernetes에서 웹후크에는 보안 통신을 보장하기 위해 API 서버가 신뢰할 수 있도록 구성된 TLS 인증서가 필요합니다.

기본적으로 Amazon CloudWatch Observability EKS 추가 기능은 API 서버와 웹후크 서버 간의 통신을 보호하기 위해 자체 서명된 CA와 이 CA에서 서명한 TLS 인증서를 자동으로 생성합니다. 이 자동 생성된 인증서의 기본 만료 기한은 10년이며 만료 시 자동 갱신되지 않습니다. 또한 추가 기능을 업그레이드하거나 다시 설치할 때마다 CA 번들과 인증서가 다시 생성되므로 만료 기한이 재설정됩니다. 자동 생성된 인증서의 기본 만료 기한을 변경하려면 추가 기능을 만들거나 업데이트할 때 다음과 같은 추가 구성을 사용합니다. *`expiry-in-days`*를 원하는 만료 기간(일)으로 바꿉니다.

```
--configuration-values '{ "admissionWebhooks": { "autoGenerateCert":
 { "expiryDays": expiry-in-days } } }'
```

보다 안전하고 기능이 풍부한 인증 기관 솔루션을 위해 이 추가 기능에는 cert-manager에 대한 옵트인 지원이 있습니다. 이 솔루션은 Kubernetes에서 TLS 인증서 관리를 위해 널리 채택되고 있으며, 이를 통해 인증서를 획득, 갱신, 관리 및 사용하는 프로세스가 간소화됩니다. 인증서가 유효하고 최신 상태인지 확인하고 만료되기 전에 구성된 시간에 인증서 갱신을 시도합니다. cert-manager는 또한 AWS Certificate Manager Private Certificate Authority를 포함하여 지원되는 다양한 소스에서 인증서 발급을 용이하게 합니다.

클러스터의 TLS 인증서 관리에 대한 모범 사례를 검토하고 프로덕션 환경에서는 cert-manager로 옵트인하는 것이 좋습니다. 승인 웹후크 TLS 인증서를 관리하기 위해 cert-manager를 활성화하도록 옵트

인 경우 Amazon CloudWatch Observability EKS 추가 기능을 설치하기 전에 Amazon EKS 클러스터에 cert-manager를 미리 설치해야 합니다. 사용 가능한 설치 옵션에 대한 자세한 내용은 cert-manager 설명서를 참조하세요. 설치한 후에는 추가 기능을 만들거나 업데이트할 때 다음 추가 구성을 사용하여 승인 웹후크 TLS 인증서를 관리하는 데 cert-manager를 사용하도록 옵트인할 수 있습니다.

```
--configuration-values '{ "admissionWebhooks": { "certManager": { "enabled":
 true } } }'
```

이 섹션에서 설명하는 고급 구성에서는 기본적으로 SelfSigned 발급자를 사용합니다.

Amazon EBS 볼륨 ID 수집

성능 로그에서 Amazon EBS 볼륨 ID를 수집하려는 경우 워커 노드나 서비스 계정에 연결된 IAM 역할에 다른 정책을 추가해야 합니다. 다음 내용을 인라인 정책으로 추가합니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

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

Amazon CloudWatch Observability EKS 추가 기능 문제 해결

다음은 Amazon CloudWatch Observability EKS 추가 기능 관련 문제를 해결하는 데 도움이 되는 정보입니다.

Amazon CloudWatch Observability EKS 추가 기능 업데이트 및 삭제

Amazon CloudWatch Observability EKS 추가 기능의 업데이트 또는 삭제에 대한 지침은 Amazon EKS 추가 기능 관리를 참조하세요. 추가 기능 이름으로 `amazon-cloudwatch-observability`를 사용합니다.

Amazon CloudWatch Observability EKS 추가 기능에서 사용하는 CloudWatch 에이전트의 버전 확인

Amazon CloudWatch Observability EKS 추가 기능은 사용 중인 CloudWatch 에이전트의 버전을 포함하여 클러스터에서 CloudWatch 에이전트 대몬 세트의 동작을 제어하는 `AmazonCloudWatchAgent` 종류의 사용자 지정 리소스를 설치합니다. 다음 명령을 입력하면 클러스터에 설치된 모든 `AmazonCloudWatchAgent` 사용자 지정 리소스 목록을 가져올 수 있습니다.

```
kubectl get amazoncloudwatchagent -A
```

이 명령의 출력에서 CloudWatch 에이전트의 버전을 확인할 수 있습니다. 또는 클러스터에서 실행 중인 `amazoncloudwatchagent` 리소스 또는 `cloudwatch-agent-*` 포드 중 하나를 설명하여 사용 중인 이미지를 검사할 수도 있습니다.

추가 기능 관리 시 ConfigurationConflict 처리

Amazon CloudWatch Observability EKS 추가 기능을 설치하거나 업데이트할 때 `Conflicts found when trying to apply. Will not continue due to resolve conflicts mode`로 시작하는 설명과 함께 `ConfigurationConflict` 유형의 `Health Issue`로 인해 발생한 오류가 발견된 경우, CloudWatch 에이전트와 ServiceAccount, ClusterRole, ClusterRoleBinding 등의 연결된 구성 요소가 클러스터에 이미 설치되어 있기 때문일 수 있습니다. 추가 기능이 CloudWatch 에이전트 및 연결된 구성 요소를 설치하려고 할 때 콘텐츠의 변경 사항이 탐지되면 기본적으로 클러스터의 리소스 상태를 덮어쓰지 않도록 설치 또는 업데이트가 실패합니다.

Amazon CloudWatch Observability EKS 추가 기능에 온보딩하려고 하는데 이 오류가 표시되는 경우 이전에 클러스터에 설치한 기존 CloudWatch 에이전트 설정을 삭제한 다음, EKS 추가 기능을 설치하는 것이 좋습니다. 사용자 지정 에이전트 구성과 같이 원래 CloudWatch 에이전트 설정에 대한 모든 사용자 지정을 백업하고 다음에 설치하거나 업데이트할 때 Amazon CloudWatch Observability EKS 추가 기능에 제공합니다. 이전에 Container Insights 온보딩을 위해 CloudWatch 에이전트를 설치한 경우 자세한 내용은 Container Insights의 CloudWatch 에이전트 및 Fluent Bit 삭제 섹션을 참조하세요.

또는 추가 기능은 `OVERWRITE`를 지정하는 기능이 있는 충돌 해결 구성 옵션을 지원합니다. 이 옵션을 사용하면 클러스터의 충돌을 덮어써서 추가 기능 설치 또는 업데이트를 진행할 수 있습니다. Amazon EKS 콘솔을 사용하는 경우 추가 기능을 생성하거나 업데이트할 때 선택적 구성 설정을 선택하면 충돌 해결 방법을 찾을 수 있습니다. AWS CLI를 사용하는 경우 명령에 `--resolve-conflicts OVERWRITE`를 제공하여 추가 기능을 생성하거나 업데이트할 수 있습니다.

CloudWatch 에이전트가 수집하는 지표

서버에 CloudWatch 에이전트를 설치하면 서버에서 지표를 수집할 수 있습니다. Amazon EC2 인스턴스와 온프레미스 서버 모두에 그리고 Linux, Windows Server 또는 macOS를 실행하는 컴퓨터에 에이전트를 설치할 수 있습니다. Amazon EC2 인스턴스에 에이전트를 설치하는 경우 에이전트가 수집하는 지표는 Amazon EC2 인스턴스에서 기본적으로 사용 설정된 지표에 추가됩니다.

인스턴스에 CloudWatch 에이전트를 설치하는 방법에 대한 자세한 내용은 CloudWatch 에이전트를 사용하여 지표, 로그, 추적 수집 단원을 참조하세요.

이 단원에서 설명하는 모든 지표는 CloudWatch 에이전트가 직접 수집합니다.

Windows Server 인스턴스의 CloudWatch 에이전트가 수집하는 지표

Windows Server를 실행하는 서버에 CloudWatch 에이전트를 설치하면 Windows 성능 모니터의 카운터와 연결된 지표를 수집할 수 있습니다. 이러한 카운터의 CloudWatch 지표 이름은 객체 이름과 카운터 이름 사이에 공백을 넣어 생성됩니다. 예를 들어 `Processor` 객체의 `% Interrupt Time` 카운터는 CloudWatch에서 `Processor % Interrupt Time`이라는 지표 이름이 부여됩니다. Windows 성능 모니터 카운터에 대한 자세한 내용은 Microsoft Windows 서버 설명서를 참조하세요.

CloudWatch 에이전트가 수집하는 지표의 기본 네임스페이스는 `CWAgent`이지만, 에이전트를 구성할 때 다른 네임스페이스를 지정할 수 있습니다.

Linux 및 macOS 인스턴스의 CloudWatch 에이전트가 수집하는 지표

다음 표에는 Linux 서버 및 macOS 컴퓨터에서 CloudWatch 에이전트를 사용하여 수집할 수 있는 지표가 나와 있습니다.

지표	설명
`cpu_time_active`	어떠한 용량에서든 CPU가 활성화되어 있는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
`cpu_time_guest`	CPU가 게스트 운영 체제에 대해 가상 CPU를 실행하는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음

지표	설명
cpu_time_guest_nice	CPU가 우선 순위가 낮으며, 다른 프로세스에 의해 중단될 수 있는 게스트 운영 체제에 대해 가상 CPU를 실행하는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
cpu_time_idle	CPU가 유휴 상태인 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
cpu_time_iowait	CPU가 I/O 작업이 완료될 때까지 기다리는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
cpu_time_irq	CPU가 인터럽트를 제공하는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
cpu_time_nice	우선 순위가 높은 프로세스에 의해 쉽게 중단될 수 있는 우선 순위가 낮은 프로세스가 있는 사용자 모드에 CPU가 있는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
cpu_time_softirq	CPU가 소프트웨어 인터럽트를 제공하는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음

지표	설명
cpu_time_steal	CPU가 도용 시간에 있는 시간입니다. 도용 시간은 가상화 환경에서 다른 운영 체제에서 사용된 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
cpu_time_system	CPU가 시스템 모드에 있는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
cpu_time_user	CPU가 사용자 모드에 있는 시간입니다. 이 지표는 수백 초 단위로 측정됩니다. 단위: 없음
cpu_usage_active	어떠한 용량에서든 CPU가 활성화되어 있는 시간(백분율)입니다. 단위: 백분율
cpu_usage_guest	CPU가 게스트 운영 체제에 대해 가상 CPU를 실행하는 시간 비율입니다. 단위: 백분율
cpu_usage_guest_nice	CPU가 우선 순위가 낮으며, 다른 프로세스에 의해 중단될 수 있는 게스트 운영 체제에 대해 가상 CPU를 실행하는 시간 비율입니다. 단위: 백분율
cpu_usage_idle	CPU가 유휴 상태인 시간 비율 단위: 백분율

지표	설명
cpu_usage_iowait	CPU가 I/O 작업이 완료될 때까지 기다리는 시간 비율입니다. 단위: 백분율
cpu_usage_irq	CPU가 인터럽트를 제공하는 시간 비율입니다. 단위: 백분율
cpu_usage_nice	우선 순위가 높은 프로세스에 의해 쉽게 중단될 수 있는 우선 순위가 낮은 프로세스가 있는 사용자 모드에 CPU가 있는 시간 비율입니다. 단위: 백분율
cpu_usage_softirq	CPU가 소프트웨어 인터럽트를 제공하는 시간 비율입니다. 단위: 백분율
cpu_usage_steal	CPU가 도용 시간에 있는 시간 또는 가상화 환경에서 다른 운영 체제에서 사용된 시간의 비율입니다. 단위: 백분율
cpu_usage_system	CPU가 시스템 모드에 있는 시간 비율입니다. 단위: 백분율
cpu_usage_user	CPU가 사용자 모드에 있는 시간 비율입니다. 단위: 백분율
disk_free	디스크의 사용 가능한 공간입니다. 단위: 바이트
disk_inodes_free	디스크에서 사용 가능한 인덱스 노드 수입니다. 단위: 수

지표	설명
disk_inodes_total	디스크에서 예약되어 있는 총 인덱스 노드 수입니다. 단위: 수
disk_inodes_used	디스크에서 사용된 인덱스 노드 수입니다. 단위: 수
disk_total	사용된 공간이나 사용 가능한 공간을 포함한 디스크의 총 공간입니다. 단위: 바이트
disk_used	디스크의 사용된 공간입니다. 단위: 바이트
disk_used_percent	사용된 총 디스크 공간 비율입니다. 단위: 백분율
diskio_iops_in_progress	디바이스 드라이버에 발행되었지만 아직 완료되지는 않은 I/O 요청 수입니다. 단위: 수
diskio_io_time	디스크가 I/O 요청이 대기하도록 한 시간입니다. 단위: 밀리초 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
diskio_reads	디스크 읽기 작업 수입니다. 단위: 수 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.

지표	설명
diskio_read_bytes	디스크에서 읽어온 바이트 수입니다. 단위: 바이트 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
diskio_read_time	읽기 요청이 디스크에서 대기하고 있는 시간입니다. 동시에 여러 개의 읽기 요청이 대기하게 되면 숫자가 증가합니다. 예를 들어, 5개의 요청 모두가 평균 100 밀리초 동안 대기하는 경우 500이 보고됩니다. 단위: 밀리초 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
diskio_writes	디스크 쓰기 작업 수입니다. 단위: 수 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
diskio_write_bytes	디스크에 기록한 바이트 수입니다. 단위: 바이트 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.

지표	설명
diskio_write_time	쓰기 요청이 디스크에서 대기하고 있는 시간입니다. 동시에 여러 개의 쓰기 요청이 대기하게 되면 숫자가 증가합니다. 예를 들어, 8개의 요청 모두가 평균 1000밀리초 동안 대기하는 경우 8000이 보고됩니다. 단위: 밀리초 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
ethtool_bw_in_allowance_exceeded	인바운드 집계 대역폭이 인스턴스의 최댓값을 초과하여 대기열에 있거나 삭제된 패킷 수입니다. 이 지표는 CloudWatch 에이전트 구성 파일의 metrics_collected 섹션에 있는 ethtool 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음
ethtool_bw_out_allowance_exceeded	아웃바운드 집계 대역폭이 인스턴스의 최댓값을 초과하여 대기열에 있거나 삭제된 패킷 수입니다. 이 지표는 CloudWatch 에이전트 구성 파일의 metrics_collected 섹션에 있는 ethtool 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음

지표	설명
ethtool_conntrack_allowance_exceeded	연결 추적에서 인스턴스의 최대값이 초과되어 새 연결을 설정하지 못했기 때문에 손실된 패킷 수입니다. 이로 인해 인스턴스의 수신 또는 송신 트래픽에 대한 패킷 손실이 발생할 수 있습니다. 이 지표는 CloudWatch 에이전트 구성 파일의 metrics_collected 섹션에 있는 ethtool 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음
ethtool_linklocal_allowance_exceeded	로컬 프록시 서비스에 대한 트래픽의 PPS가 네트워크 인터페이스의 최대값을 초과하여 손실된 패킷 수입니다. 이는 DNS 서비스, Instance Metadata Service 및 Amazon Time Sync Service에 대한 트래픽에 영향을 미칩니다. 이 지표는 CloudWatch 에이전트 구성 파일의 metrics_collected 섹션에 있는 ethtool 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음
ethtool_pps_allowance_exceeded	양방향 PPS가 인스턴스의 최댓값을 초과하여 대기열에 있거나 삭제된 패킷 수입니다. 이 지표는 CloudWatch 에이전트 구성 파일의 metrics_collected 섹션에 있는 ethtool 하위 섹션에 나열된 경우에만 수집됩니다. 자세한 내용은 네트워크 성능 지표 수집 단원을 참조하세요. 단위: 없음

지표	설명
mem_active	마지막 샘플 기간 동안 몇 가지 방식으로 사용된 메모리 양입니다. 단위: 바이트
mem_available	사용 가능하며 즉시 프로세스에 제공할 수 있는 메모리 양입니다. 단위: 바이트
mem_available_percent	사용 가능하며 즉시 프로세스에 제공할 수 있는 메모리 비율입니다. 단위: 백분율
mem_buffered	버퍼에 사용되고 있는 메모리 양입니다. 단위: 바이트
mem_cached	파일 캐시에 사용되고 있는 메모리 양입니다. 단위: 바이트
mem_free	사용되지 않는 메모리 양입니다. 단위: 바이트
mem_inactive	마지막 샘플 기간 동안 몇 가지 방식으로 사용되지 않은 메모리 양입니다. 단위: 바이트
mem_total	총 메모리 크기 단위: 바이트
mem_used	현재 사용 중인 메모리 양입니다. 단위: 바이트

지표	설명
mem_used_percent	현재 사용 중인 메모리 비율입니다. 단위: 백분율
net_bytes_recv	네트워크 인터페이스에서 받은 바이트 수입니다. 단위: 바이트 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
net_bytes_sent	네트워크 인터페이스가 보낸 바이트 수입니다. 단위: 바이트 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
net_drop_in	이 네트워크 인터페이스에서 받았지만 삭제된 패킷 수입니다. 단위: 수 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
net_drop_out	이 네트워크 인터페이스가 전송했지만 삭제된 패킷 수입니다. 단위: 수 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.

지표	설명
net_err_in	이 네트워크 인터페이스가 탐지한 수신 오류 수입니다. 단위: 수 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
net_err_out	이 네트워크 인터페이스가 탐지한 전송 오류 수입니다. 단위: 수 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
net_packets_sent	이 네트워크 인터페이스가 보낸 패킷 수입니다. 단위: 수 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
net_packets_recv	이 네트워크 인터페이스에서 받은 패킷 수입니다. 단위: 수 이 지표에는 Sum 통계만 사용해야 합니다. Average는 사용하지 마세요.
netstat_tcp_close	상태가 없는 TCP 연결 수입니다. 단위: 수
netstat_tcp_close_wait	클라이언트로부터 종료 요청을 기다리고 있는 TCP 연결 수입니다. 단위: 수

지표	설명
netstat_tcp_closing	클라이언트로부터 승인과 함께 종료 요청을 기다리고 있는 TCP 연결 수입니다. 단위: 수
netstat_tcp_established	설정된 TCP 연결 수입니다. 단위: 수
netstat_tcp_fin_wait1	연결 종료 프로세스 중에 FIN_WAIT1 상태에 있는 TCP 연결 수입니다. 단위: 수
netstat_tcp_fin_wait2	연결 종료 프로세스 중에 FIN_WAIT2 상태에 있는 TCP 연결 수입니다. 단위: 수
netstat_tcp_last_ack	클라이언트가 연결 종료 메시지 승인을 보내길 기다리고 있는 TCP 연결 수입니다. 연결이 종료되기 직전의 마지막 단계입니다. 단위: 수
netstat_tcp_listen	현재 연결 요청을 수신하고 있는 TCP 포트 수입니다. 단위: 수
netstat_tcp_none	비활성 클라이언트가 있는 TCP 연결 수입니다. 단위: 수
netstat_tcp_syn_sent	연결 요청을 보낸 후 일치하는 연결 요청을 기다리고 있는 TCP 연결 수입니다. 단위: 수

지표	설명
netstat_tcp_syn_recv	연결 요청을 보내고 받은 후 연결 요청 승인을 기다리고 있는 TCP 연결 수입니다. 단위: 수
netstat_tcp_time_wait	클라이언트가 연결 종료 요청 승인을 받았는지 확인하기 위해 현재 기다리고 있는 TCP 연결 수입니다. 단위: 수
netstat_udp_socket	현재 UDP 연결 수입니다. 단위: 수
processes_blocked	차단된 프로세스 수입니다. 단위: 수
processes_dead	"데드(Dead)" 상태인 프로세스 수이며, Linux에서 X 상태 코드로 나타납니다. macOS 컴퓨터에서는 이 지표가 수집되지 않습니다. 단위: 수
processes_idle	유휴 상태(20초 이상 절전 모드)인 프로세스 수입니다. FreeBSD 인스턴스에서만 사용할 수 있습니다. 단위: 수
processes_paging	페이징되고 있는 프로세스 수이며, Linux에서 W 상태 코드로 나타납니다. macOS 컴퓨터에서는 이 지표가 수집되지 않습니다. 단위: 수

지표	설명
processes_running	실행 중인 프로세스 수이며, R 상태 코드로 나타납니다. 단위: 수
processes_sleeping	절전 모드의 프로세스 수이며, S 상태 코드로 나타납니다. 단위: 수
processes_stopped	중지된 프로세스 수이며, T 상태 코드로 나타납니다. 단위: 수
processes_total	인스턴스에 있는 총 프로세스 수입니다. 단위: 수
processes_total_threads	프로세스를 구성하는 총 스레드 수입니다. 이 지표는 Linux 인스턴스에서만 사용할 수 있습니다. macOS 컴퓨터에서는 이 지표가 수집되지 않습니다. 단위: 수
processes_wait	페이징되고 있는 프로세스 수이며, FreeBSD 인스턴스에서 W 상태 코드로 나타납니다. 이 지표는 FreeBSD 인스턴스에서만 사용할 수 있으며 Linux, Windows Server 또는 macOS 인스턴스에서는 사용할 수 없습니다. 단위: 수
processes_zombies	좀비 프로세스 수이며, Z 상태 코드로 나타납니다. 단위: 수

지표	설명
swap_free	사용되지 않는 스왑 공간 크기입니다. 단위: 바이트
swap_used	현재 사용 중인 스왑 공간 크기입니다. 단위: 바이트
swap_used_percent	현재 사용 중인 스왑 공간 비율입니다. 단위: 백분율

CloudWatch 에이전트가 수집하는 메모리 지표의 정의

CloudWatch 에이전트가 메모리 지표를 수집할 때 소스는 호스트의 메모리 관리 하위 시스템입니다. 예를 들어 Linux 커널은 /proc에서 OS 유지 관리 데이터를 노출합니다. 메모리의 경우 데이터가 /proc/meminfo에 들어 있습니다.

운영 체제 및 아키텍처마다 프로세스에서 사용하는 리소스의 계산 방식이 다릅니다. 자세한 내용은 다음 섹션을 참조하십시오.

각 수집 간격 동안 각 인스턴스의 CloudWatch 에이전트는 인스턴스 리소스를 수집하고 인스턴스에서 실행 중인 모든 프로세스에서 사용 중인 리소스를 계산합니다. 이 정보는 CloudWatch 지표에 다시 보고됩니다. CloudWatch 에이전트 구성 파일에서 수집 간격의 길이를 구성할 수 있습니다. 자세한 내용은 CloudWatch 에이전트 구성 파일: 에이전트 섹션 단원을 참조하십시오.

다음 목록은 CloudWatch 에이전트가 수집하는 메모리 지표가 어떻게 정의되는지 설명합니다.

- 활성 메모리 - 프로세스에서 사용 중인 메모리입니다. 즉, 현재 실행 중인 앱이 사용하는 메모리입니다.
- 가용 메모리 - 시스템을 교체하지 않고도 프로세스에 즉시 제공할 수 있는 메모리입니다(가상 메모리라고도 함).
- 버퍼 메모리 - 서로 다른 속도와 우선 순위로 작동하는 하드웨어 디바이스 또는 프로그램 프로세스가 공유하는 데이터 영역입니다.
- 캐시된 메모리 - 프로그램 작동에 반복적으로 사용되는 프로그램 명령어와 데이터를 저장하여 CPU가 다음에 필요로 할 가능성이 큽니다.

- 여유 메모리 - 전혀 사용되지 않았으며 사용할 수 있는 메모리입니다. 필요할 때 시스템에서 완전히 무료로 사용할 수 있습니다.
- 비활성 메모리 - '최근' 액세스한 적이 없는 페이지입니다.
- 총 메모리 - 실제 물리적 메모리 RAM의 크기입니다.
- 사용된 메모리 - 프로그램과 프로세스에서 현재 사용 중인 메모리입니다.

주제

Linux: 수집된 지표 및 사용된 계산

수집된 지표 및 단위:

- 활성(바이트)
- 사용 가능(바이트)
- 가용 퍼센트(퍼센트)
- 버퍼링됨(바이트)
- 캐시됨(바이트)
- 여유(바이트)
- 비활성(바이트)
- 총(바이트)
- 사용됨(바이트)
- 사용된 퍼센트(퍼센트)

사용된 메모리 = 총 메모리 - 여유 메모리 - 캐시된 메모리 - 버퍼 메모리

총 메모리 = 사용 메모리 + 여유 메모리 + 캐시 메모리 + 버퍼 메모리

macOS: 수집된 지표 및 사용된 계산

수집된 지표 및 단위:

- 활성(바이트)
- 사용 가능(바이트)
- 가용 퍼센트(퍼센트)
- 여유(바이트)
- 비활성(바이트)
- 총(바이트)
- 사용됨(바이트)
- 사용된 퍼센트(퍼센트)

사용 가능한 메모리 = 여유 메모리 + 비활성 메모리

사용된 메모리 = 총 메모리 - 사용 가능한 메모리

사용된 메모리 = 사용 가능한 메모리 + 사용된 메모리

Windows: 수집된 지표

Windows 호스트에서 수집된 지표는 다음과 같습니다. 이러한 모든 지표는 `Unit`에 대해 `None`입니다.

- 사용 가능(바이트)
- 캐시 장애/초
- 페이지 장애/초
- 페이지/초

CloudWatch 에이전트는 성능 카운터에서 이벤트를 구문 분석하기 때문에 Windows 지표에는 계산이 사용되지 않습니다.

예: Linux에서 메모리 지표 계산

예를 들어 Linux 호스트에서 **`cat /proc/meminfo`** 명령을 입력하면 다음과 같은 결과가 표시된다고 가정해 보겠습니다.

```
MemTotal:       3824388 kB
MemFree:         462704 kB
MemAvailable:   2157328 kB
Buffers:         126268 kB
Cached:         1560520 kB
```

```
SReclaimable:    289080 kB>
```

이 예에서는 CloudWatch 에이전트가 다음 값을 수집합니다. CloudWatch 에이전트가 수집하고 보고하는 모든 값은 바이트 단위입니다.

- `mem_total`: 3916173312바이트
- `mem_available`: 2209103872바이트(MemFree + Cached)
- `mem_free`: 473808896바이트
- `mem_cached`: 1893990400바이트(`cached` + `SReclaimable`
- `mem_used`: 1419075584바이트(`MemTotal` – (`MemFree` + `Buffers` + (`Cached` + `SReclaimable`)))
- `mem_buffered`: 129667072바이트
- `mem_available_percent`: 56.41%
- `mem_used_percent`: 36.24% (`mem_used` / `mem_total`) * 100

CloudWatch 에이전트를 사용하는 일반적인 시나리오

다음 섹션에서는 CloudWatch 에이전트에 대한 일반적인 구성 및 사용자 지정 작업을 완료하는 방법을 간략하게 설명합니다.

주제

다른 사용자로 CloudWatch 에이전트 실행

Linux 서버에서 CloudWatch는 기본적으로 루트 사용자로 실행됩니다. 에이전트를 다른 사용자로 실행하려면 CloudWatch 에이전트 구성 파일의 `agent` 섹션에 있는 `run_as_user` 파라미터를 사용합니다. 이 옵션은 Linux 서버에서만 사용할 수 있습니다.

이미 루트 사용자로 에이전트를 실행 중인데 다른 사용자를 사용하도록 변경하려면 다음 절차 중 하나를 사용하세요.

Linux를 실행하는 EC2 인스턴스에서 CloudWatch 에이전트를 다른 사용자로 실행하려면

1. 새 CloudWatch 에이전트 패키지를 다운로드하여 설치합니다. 자세한 내용은 CloudWatch 에이전트 패키지 다운로드 단원을 참조하세요.
2. 새 Linux 사용자를 생성하거나 RPM 또는 DEB 파일이 생성한 기본 사용자 `cwagent`를 사용합니다.
3. 다음 중 한 가지 방법으로 이 사용자에 대한 자격 증명을 제공합니다.
   - `.aws/credentials` 파일이 루트 사용자의 홈 디렉터리에 있는 경우 CloudWatch 에이전트를 실행하는 데 사용할 사용자의 자격 증명 파일을 생성해야 합니다. 이 자격 증명 파일은 `/home/`*`username`*`/.aws/credentials`입니다. 그런 다음 `common-config.toml`에 있는 `shared_credential_file` 파라미터 값을 자격 증명 파일의 경로 이름으로 설정합니다. 자세한 내용은 (선택 사항) 프록시 또는 리전 정보에 대한 일반 구성 수정 단원을 참조하세요.
   - `.aws/credentials` 파일이 루트 사용자의 홈 디렉터리에 없는 경우 다음 중 하나를 수행할 수 있습니다.
     - CloudWatch 에이전트를 실행하는 데 사용할 사용자의 자격 증명 파일을 생성합니다. 이 자격 증명 파일은 `/home/`*`username`*`/.aws/credentials`입니다. 그런 다음 `common-config.toml`에 있는 `shared_credential_file` 파라미터 값을 자격 증명 파일의 경로 이름으로 설정합니다. 자세한 내용은 (선택 사항) 프록시 또는 리전 정보에 대한 일반 구성 수정 단원을 참조하세요.
     - 자격 증명 파일을 생성하는 대신 인스턴스에 IAM 역할을 연결합니다. 에이전트는 이 역할을 자격 증명 공급자로 사용합니다.
4. CloudWatch 에이전트 구성 파일에서 `agent` 섹션에 다음 줄을 추가합니다.

   ```
   "run_as_user": "username"
   ```

   필요에 따라 구성 파일을 추가로 수정합니다. 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.

5. 사용자에게 필수 권한을 부여하세요. 사용자는 수집할 로그 파일에 대한 읽기(r) 권한이 있어야 하며 로그 파일 경로의 모든 디렉터리에 대해 실행(x) 권한이 있어야 합니다.
6. 방금 수정한 구성 파일로 에이전트를 시작합니다.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
```

Linux를 실행하는 온프레미스 서버에서 CloudWatch 에이전트를 다른 사용자로 실행하려면

1. 새 CloudWatch 에이전트 패키지를 다운로드하여 설치합니다. 자세한 내용은 CloudWatch 에이전트 패키지 다운로드 단원을 참조하세요.
2. 새 Linux 사용자를 생성하거나 RPM 또는 DEB 파일이 생성한 기본 사용자 `cwagent`를 사용합니다.
3. 이 사용자의 자격 증명을 사용자가 액세스할 수 있는 경로(예: `/home/username/.aws/credentials`)에 저장합니다.
4. `common-config.toml`에 있는 `shared_credential_file` 파라미터 값을 자격 증명 파일의 경로 이름으로 설정합니다. 자세한 내용은 (선택 사항) 프록시 또는 리전 정보에 대한 일반 구성 수정 단원을 참조하세요.
5. CloudWatch 에이전트 구성 파일에서 `agent` 섹션에 다음 줄을 추가합니다.

```
"run_as_user": "username"
```

   필요에 따라 구성 파일을 추가로 수정합니다. 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.
6. 사용자에게 필수 권한을 부여하세요. 사용자는 수집할 로그 파일에 대한 읽기(r) 권한이 있어야 하며 로그 파일 경로의 모든 디렉터리에 대해 실행(x) 권한이 있어야 합니다.
7. 방금 수정한 구성 파일로 에이전트를 시작합니다.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
```

CloudWatch 에이전트가 희소 로그 파일을 처리하는 방법

희소 파일은 빈 블록과 실제 내용이 모두 포함된 파일입니다. 희소 파일은 블록을 구성하는 실제 null 바이트 대신 빈 블록을 나타내는 간단한 정보를 디스크에 작성하여 디스크 공간을 보다 효율적으로 사용합니다. 이렇게 하면 일반적으로 희소 파일의 실제 크기가 명백한 크기보다 훨씬 작아집니다.

그러나 CloudWatch 에이전트는 희소 파일을 일반 파일 처리 방법과 다르게 처리하지 않습니다. 에이전트가 희소 파일을 읽을 때 빈 블록은 null 바이트로 채워진 "실제" 블록으로 처리됩니다. 이 때문에 CloudWatch 에이전트는 희소 파일의 외관상 크기만큼의 바이트를 CloudWatch에 게시합니다.

희소 파일을 게시하도록 CloudWatch 에이전트를 구성하면 예상보다 높은 CloudWatch 비용이 발생할 수 있으므로 그렇게 하지 않는 것이 좋습니다. 예를 들어 Linux의 `/var/logs/lastlog`는 일반적으로 희소 파일이므로 CloudWatch에 게시하지 않는 것이 좋습니다.

CloudWatch 에이전트가 수집한 지표에 사용자 지정 측정기준 추가

에이전트에 의해 수집되는 지표에 태그와 같은 사용자 지정 측정기준을 추가하려면 해당 지표를 나열하는 에이전트 구성 파일의 섹션에 `append_dimensions` 필드를 추가하세요.

예를 들어, 구성 파일의 다음 예제 섹션에서는 `stackName`의 값이 포함된 `Prod`라는 사용자 지정 측정기준을 에이전트에 의해 수집된 `cpu` 및 `disk` 지표에 추가합니다.

```
"cpu":{
  "resources":[
    "*"
  ],
  "measurement":[
    "cpu_usage_guest",
    "cpu_usage_nice",
    "cpu_usage_idle"
  ],
  "totalcpu":false,
  "append_dimensions":{
    "stackName":"Prod"
  }
},
"disk":{
  "resources":[
    "/",
    "/tmp"
  ],
  "measurement":[
```

```
    "total",
    "used"
  ],
  "append_dimensions":{
    "stackName":"Prod"
  }
}
```

에이전트 구성 파일을 변경할 때마다 에이전트를 다시 시작하여 변경 사항을 적용해야 합니다.

여러 CloudWatch 에이전트 구성 파일

Linux 서버와 Windows 서버 모두에서, 여러 개의 구성 파일을 사용하도록 CloudWatch 에이전트를 설정할 수 있습니다. 예를 들어 인프라의 모든 서버에서 항상 수집하려는 일련의 지표, 로그, 추적을 수집하는 공통 구성 파일을 사용할 수 있습니다. 그런 다음 특정 애플리케이션이나 특정 상황에서 지표를 수집하는 추가 구성 파일을 사용할 수 있습니다.

이렇게 설정하려면 먼저 사용하려는 구성 파일을 생성합니다. 동일한 서버에서 함께 사용할 구성 파일은 파일 이름이 서로 달라야 합니다. 구성 파일을 서버 또는 파라미터 스토어에 저장할 수 있습니다.

`fetch-config` 옵션을 사용하여 CloudWatch 에이전트를 시작하고 첫 번째 구성 파일을 지정합니다. 실행 중인 에이전트에 두 번째 구성 파일을 추가하려면, 동일한 명령에 `append-config` 옵션을 사용합니다. 구성 파일에 나열된 모든 지표, 로그, 추적이 수집됩니다. 다음 명령 예제는 파일로 저장된 구성을 사용하는 이 시나리오를 보여줍니다. 첫 행은 `infrastructure.json` 구성 파일을 사용하여 에이전트를 시작하고, 둘째 행은 `app.json` 구성 파일을 추가합니다.

다음은 Linux용 명령 예제입니다.

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2
 -s -c file:/tmp/infrastructure.json
```

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a append-config -m
 ec2 -s -c file:/tmp/app.json
```

다음은 Windows Server용 명령 예제입니다.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a fetch-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
\infrastructure.json"
```

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a append-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
\app.json"
```

다음 예제 구성 파일은 이 기능의 사용을 보여줍니다. 첫 번째 구성 파일은 인프라의 모든 서버에 사용되며, 두 번째 구성 파일은 특정 애플리케이션의 로그만 수집하며, 해당 애플리케이션을 실행하는 서버에 추가됩니다.

infrastructure.json

```
{
  "metrics": {
    "metrics_collected": {
      "cpu": {
        "resources": [
          "*"
        ],
        "measurement": [
          "usage_active"
        ],
        "totalcpu": true
      },
      "mem": {
         "measurement": [
           "used_percent"
        ]
      }
    }
  },
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
            "log_group_name": "amazon-cloudwatch-agent.log"
          },
          {
            "file_path": "/var/log/messages",
            "log_group_name": "/var/log/messages"
          }
        ]
```

```
      }
    }
  }
}
```

app.json

```
{
    "logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "/app/app.log*",
                        "log_group_name": "/app/app.log"
                    }
                ]
            }
        }
    }
}
```

구성에 추가된 모든 구성 파일은 파일 이름이 서로 달라야 하며 초기 구성 파일의 이름과도 달라야 합니다. 파일 이름이 동일한 구성 파일이 있는 append-config를 에이전트에서 이미 사용 중인 구성 파일로 사용할 경우 추가 명령은 첫 번째 구성 파일에 추가되는 것이 아니라 해당 파일의 정보를 덮어씁니다. 파일 이름이 동일한 두 개의 구성 파일이 서로 다른 파일 경로에 있는 경우에도 마찬가지입니다.

앞의 예는 두 개의 구성 파일을 사용하는 것을 보여주지만, 에이전트 구성에 추가할 수 있는 구성 파일의 수는 제한되지 않습니다. 서버에 있는 구성 파일과 파라미터 스토어에 있는 구성을 함께 사용할 수도 있습니다.

CloudWatch 에이전트가 수집한 지표 집계 또는 롤업

에이전트에 의해 수집되는 지표를 집계하거나 롤업하려면 에이전트 구성 파일의 해당 지표에 대한 섹션에 aggregation_dimensions 필드를 추가하세요.

예를 들어, 다음 구성 파일 조각은 AutoScalingGroupName 측정기준에서 지표를 롤업합니다. 각 Auto Scaling 그룹의 모든 인스턴스에서 지표가 집계되어 전체적으로 표시될 수 있습니다.

```
"metrics": {
  "cpu":{...}
```

```
  "disk":{...}
  "aggregation_dimensions" : [["AutoScalingGroupName"]]
}
```

Auto Scaling 그룹 이름에서 롤업할 뿐 아니라 InstanceId 및 InstanceType 측정기준 각각의 조합을 따라 롤업하려면 다음을 추가합니다.

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId", "InstanceType"]]
}
```

대신 지표를 하나의 모음에 롤업하려면 []를 사용합니다.

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [[]]
}
```

에이전트 구성 파일을 변경할 때마다 에이전트를 다시 시작하여 변경 사항을 적용해야 합니다.

CloudWatch 에이전트로 고분해능 지표 수집

metrics_collection_interval 필드는 수집되는 지표의 시간 간격을 초 단위로 지정합니다. 이 필드에 60 미만의 값을 지정하면 지표가 고분해능 지표로 수집됩니다.

예를 들어, 모든 지표가 고해상도 지표이며 10초마다 수집되어야 하는 경우 agent 섹션의 metrics_collection_interval에서 글로벌 지표 수집 간격 값으로 10을 지정합니다.

```
"agent": {
  "metrics_collection_interval": 10
}
```

또는 다음 예에서는 cpu 지표는 1초마다 수집되도록 설정하고 다른 모든 지표는 1분마다 수집되도록 설정합니다.

```
"agent":{
  "metrics_collection_interval": 60
},
```

```
"metrics":{
  "metrics_collected":{
    "cpu":{
      "resources":[
        "*"
      ],
      "measurement":[
        "cpu_usage_guest"
      ],
      "totalcpu":false,
      "metrics_collection_interval": 1
    },
    "disk":{
      "resources":[
        "/",
        "/tmp"
      ],
      "measurement":[
        "total",
        "used"
      ]
    }
  }
}
```

에이전트 구성 파일을 변경할 때마다 에이전트를 다시 시작하여 변경 사항을 적용해야 합니다.

다른 계정에 지표, 로그, 추적 전송

CloudWatch 에이전트가 지표, 로그, 추적을 다른 계정에 전송하도록 하려면 전송 서버의 에이전트 구성 파일에서 `role_arn` 파라미터를 지정합니다. `role_arn` 값은 에이전트가 데이터를 대상 계정에 전송할 때 사용하는 대상 계정의 IAM 역할을 지정합니다. 지표 또는 로그를 대상 계정에 전달할 때는 이 역할을 통해 전송 계정이 대상 계정의 해당 역할을 맡을 수 있습니다.

에이전트 구성 파일에 지표를 보낼 때 사용할 문자열, 로그를 보낼 때 사용할 문자열, 추적을 보낼 때 사용할 문자열 등 `role_arn` 문자열을 별도로 지정할 수도 있습니다.

구성 파일의 `agent` 섹션 부분에 대한 다음의 예에서는 데이터를 다른 계정에 보낼 때 `CrossAccountAgentRole`을 사용하도록 에이전트를 설정합니다.

```
{
  "agent": {
```

```
    "credentials": {
      "role_arn": "arn:aws:iam::123456789012:role/CrossAccountAgentRole"
    }
  },
  .....
}
```

또는 다음 예에서는 지표, 로그 및 추적 전송에 사용할 전송 계정에 대해 서로 다른 역할을 설정합니다.

```
"metrics": {
    "credentials": {
     "role_arn": "RoleToSendMetrics"
    },
    "metrics_collected": {....
```

```
"logs": {
    "credentials": {
    "role_arn": "RoleToSendLogs"
    },
    ....
```

필요한 정책

에이전트 구성 파일에서 `role_arn`을 지정할 때 전송 및 대상 계정의 IAM 역할에 특정 정책이 있는지도 확인해야 합니다. 전송 계정과 대상 계정의 역할에는 모두 `CloudWatchAgentServerPolicy`가 있어야 합니다. 이 정책을 역할에 지정하는 방법에 대한 자세한 내용은 Amazon EC2 인스턴스에서 CloudWatch 에이전트와 함께 사용할 IAM 역할 생성 단원을 참조하세요.

전송 계정의 역할은 다음의 정책을 포함해야 합니다. 역할을 편집할 때 IAM 콘솔의 [권한(Permissions)] 탭에서 이 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
```

```
                "arn:aws:iam::target-account-ID:role/agent-role-in-target-account"
            ]
        }
    ]
}
```

대상 계정의 역할에는 전송 계정에서 사용하는 IAM 역할을 인식하도록 다음 정책이 포함되어야 합니다. 역할을 편집할 때 IAM 콘솔의 [신뢰 관계(Trust relationships)] 탭에서 이 정책을 추가합니다. 이 정책을 추가하는 대상 계정의 역할은 CloudWatch 에이전트와 함께 사용하기 위한 IAM 역할 및 사용자 생성에서 생성한 역할입니다. 이 역할은 전송 계정에서 사용한 정책의 *agent-role-in-target-account*에 지정된 역할입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::sending-account-ID:role/role-in-sender-account"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

통합 CloudWatch 에이전트와 이전 CloudWatch Logs 에이전트 간의 타임스탬프 차이

CloudWatch 에이전트는 이전 CloudWatch Logs 에이전트와 비교했을 때 타임스탬프 형식에 대해 다른 기호 집합을 지원합니다. 이러한 차이는 다음 표에 표시됩니다.

두 에이전트 모두에서 지원하는 기호	통합 CloudWatch 에이전트에서만 지원하는 기호	이전 CloudWatch Logs 에이전트에서만 지원하는 기호
%A, %a, %b, %B, %d, %f, %H, %I, %m, %M, %p, %S, %y, %Y, %Z, %z	%-d, %-l, %-m, %-M, %-S	%c,%j, %U, %W, %w

새 CloudWatch 에이전트에서 지원하는 기호의 의미에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 에이전트 구성 파일: 로그 섹션 단원을 참조하세요. CloudWatch Logs 에이전트에서 지원하는 기호에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 에이전트 구성 파일 단원을 참조하세요.

CloudWatch 에이전트 문제 해결

다음은 CloudWatch 에이전트 관련 문제를 해결하는 데 도움이 되는 정보입니다.

주제

CloudWatch 에이전트 명령줄 파라미터

CloudWatch 에이전트가 지원하는 전체 파라미터 목록을 보려면 에이전트가 설치된 컴퓨터의 명령줄에 다음과 같이 입력합니다.

```
amazon-cloudwatch-agent-ctl -help
```

Run Command를 사용한 CloudWatch 에이전트 설치 실패

Systems Manager Run Command를 사용하여 CloudWatch 에이전트를 설치하려면 대상 서버의 SSM Agent가 2.2.93.0 이상 버전이어야 합니다. SSM Agent가 올바른 버전이 아닌 경우 다음 메시지를 포함한 오류가 표시될 수 있습니다.

```
no latest version found for package AmazonCloudWatchAgent on platform linux
```

```
failed to download installation package reliably
```

SSM Agent 버전 업데이트에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 SSM Agent 설치 및 구성 단원을 참조하세요.

CloudWatch 에이전트가 시작되지 않음

CloudWatch 에이전트가 시작되지 않는 경우 구성에 문제가 있을 수 있습니다. 구성 정보는 `configuration-validation.log` 파일에 기록됩니다. 이 파일은 Linux 서버의 `/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log` 및 Windows Server가 실행되는 서버의 `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log`에 있습니다.

CloudWatch 에이전트가 실행 중인지 확인

CloudWatch 에이전트를 쿼리하여 에이전트가 실행 중인지 아니면 중지되었는지 확인할 수 있습니다. AWS Systems Manager를 사용하면 이 작업을 원격으로 수행할 수 있습니다. 명령줄을 사용할 수도 있지만, 로컬 서버를 확인하기 위해서만 사용해야 합니다.

Run Command를 사용하여 CloudWatch 에이전트의 상태를 쿼리하려면

1. https://console.aws.amazon.com/systems-manager/에서 시스템 관리자 콘솔을 엽니다.
2. 탐색 창에서 Run Command를 선택합니다.

   -또는-

   AWS Systems Manager 홈페이지가 열리면 아래로 스크롤하여 [Run Command 탐색(Explore Run Command)]을 선택합니다.
3. Run command(Run 명령)를 선택합니다.

4. 명령 문서 목록에서 AmazonCloudWatch-ManageAgent 옆의 버튼을 선택합니다.
5. 작업 목록에서 상태를 선택합니다.
6. Optional Configuration Source(구성 소스(선택 사항))에 대해 기본값을 선택하고 Optional Configuration Location(구성 위치(선택 사항))을 비워 둡니다.
7. 대상 영역에서 확인할 인스턴스를 선택합니다.
8. Run(실행)을 선택합니다.

에이전트가 실행 중인 경우 결과가 다음과 유사하게 표시됩니다.

```
{
       "status": "running",
       "starttime": "2017-12-12T18:41:18",
       "version": "1.73.4"
}
```

에이전트가 중지된 경우 `"status"` 필드에 `"stopped"`가 표시됩니다.

명령줄을 사용하여 로컬에서 CloudWatch 에이전트의 상태를 쿼리하려면

- Linux 서버의 경우 다음을 입력합니다.

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a
   status
  ```

  Windows Server가 실행되는 서버의 경우 관리자로서 PowerShell에 다음을 입력합니다.

  ```
  & $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m
   ec2 -a status
  ```

CloudWatch 에이전트가 시작되지 않고 오류에 Amazon EC2 리전이 언급되어 있음

에이전트가 시작되지 않고 오류 메시지에 Amazon EC2 리전 엔드포인트가 언급되어 있으면 액세스 권한을 부여하지 않고 Amazon EC2 엔드포인트에 액세스해야 하도록 에이전트를 구성했을 수 있습니다.

예를 들어 에이전트 구성 파일에서 Amazon EC2 메타데이터에 따라 달라지는 `append_dimensions` 파라미터의 값을 지정하고 프록시를 사용하는 경우 서버가 Amazon EC2의 엔드포인트에 액세스할 수

있는지 확인해야 합니다. 이러한 엔드포인트에 대한 자세한 내용은 Amazon Web Services 일반 참조의 Amazon Elastic Compute Cloud(Amazon EC2)를 참조하세요.

CloudWatch 에이전트가 Windows Server에서 시작되지 않음

Windows Server에 다음 오류가 표시될 수 있습니다.

```
Start-Service : Service 'Amazon CloudWatch Agent (AmazonCloudWatchAgent)' cannot be
 started due to the following
error: Cannot start service AmazonCloudWatchAgent on computer '.'.
At C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1:113
 char:12
+     $svc | Start-Service
+            ~~~~~~~~~~~~~
    + CategoryInfo          : OpenError:
 (System.ServiceProcess.ServiceController:ServiceController) [Start-Service],
   ServiceCommandException
    + FullyQualifiedErrorId :
 CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand
```

이 문제를 해결하려면 먼저, 서버 서비스가 실행 중인지 확인합니다. 서버 서비스가 실행되고 있지 않을 때 에이전트를 시작하려고 하면 이 오류가 표시될 수 있습니다.

서버 서비스가 이미 실행 중인 경우 다음이 문제일 수 있습니다. 경우에 따라 Windows Server에 설치된 CloudWatch 에이전트를 시작하는 데 30초 넘게 걸립니다. Windows Server는 기본적으로 서비스를 시작하는 데 30초만 허용하므로 다음과 유사한 오류가 발생하며 에이전트가 시작되지 않습니다.

이 문제를 해결하려면 서비스 제한 시간 값을 늘려야 합니다. 자세한 내용은 서비스가 시작되지 않으며 Windows 이벤트 로그에 이벤트 7000 및 7011이 기록됨을 참조하세요.

지표를 찾을 수 없음

CloudWatch 에이전트가 실행 중이지만 AWS Management Console 또는 AWS CLI에서 에이전트가 수집한 지표를 찾을 수 없는 경우 올바른 네임스페이스를 사용하고 있는지 확인합니다. 기본적으로 에이전트에 의해 수집되는 지표의 네임스페이스는 `CWAgent`입니다. 에이전트 구성 파일의 `metrics` 섹션에 있는 `namespace` 필드를 사용하여 이 네임스페이스를 사용자 지정할 수 있습니다. 원하는 지표가 표시되지 않는 경우 구성 파일을 점검하여 사용 중인 네임스페이스를 확인하세요.

CloudWatch 에이전트 패키지를 처음 다운로드할 때 에이전트 구성 파일은 `amazon-cloudwatch-agent.json`입니다. 이 파일은 구성 마법사를 실행한 디렉터리에 있습니다. 그렇지 않으면 이 파일을 다른 디렉터리로 옮긴 것일 수 있습니다. 구성 마법사를 사용하는 경우, 마법사의 에이전트 구성 파일

출력이 config.json으로 명명됩니다. namespace 필드를 포함한 구성 파일에 대한 자세한 내용은 CloudWatch 에이전트 구성 파일: 지표 섹션 단원을 참조하세요.

CloudWatch 에이전트가 컨테이너에서 실행되는 데 시간이 오래 걸리거나 홉 제한 오류가 로깅됩니다.

CloudWatch 에이전트를 컨테이너 서비스로 실행하고 Amazon EC2 지표 차원을 에이전트가 수집한 모든 지표에 추가하려는 경우, 에이전트의 버전 v1.247354.0에서 다음 오류가 표시될 수 있습니다.

```
2022-06-07T03:36:11Z E! [processors.ec2tagger] ec2tagger: Unable to retrieve Instance
 Metadata Tags. This plugin must only be used on an EC2 instance.
2022-06-07T03:36:11Z E! [processors.ec2tagger] ec2tagger: Please increase hop limit
 to 2 by following this document https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
configuring-instance-metadata-options.html#configuring-IMDS-existing-instances.
2022-06-07T03:36:11Z E! [telegraf] Error running agent: could not initialize processor
 ec2tagger: EC2MetadataRequestError: failed to get EC2 instance identity document
caused by: EC2MetadataError: failed to make EC2Metadata request
        status code: 401, request id:
caused by:
```

에이전트가 적절한 홉 제한 없이 컨테이너 내의 IMDSv2에서 메타데이터를 가져오려고 하면 이 오류가 표시될 수 있습니다. v1.247354.0 이전 버전의 에이전트에서는 로그 메시지가 표시되지 않지만 이 문제가 발생할 수 있습니다.

이 문제를 해결하려면 인스턴스 메타데이터 옵션 구성의 지침에 따라 홉 제한을 2로 늘리세요.

에이전트 구성을 업데이트했지만 CloudWatch 콘솔에 새 지표 또는 로그가 표시되지 않음

CloudWatch 에이전트 구성 파일을 업데이트한 경우 다음에 에이전트를 시작할 때 **fetch-config** 옵션을 사용해야 합니다. 예를 들어 로컬 컴퓨터에 업데이트된 파일을 저장한 경우 다음 명령을 입력합니다.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
s -m ec2 -c file:configuration-file-path
```

CloudWatch 에이전트 파일 및 위치

다음 표에는 CloudWatch 에이전트가 설치하고 사용하는 파일이 Linux 또는 Windows Server를 실행하는 서버에서의 해당 위치와 함께 나열되어 있습니다.

파일	Linux 위치	Windows Server 위치
에이전트의 시작, 중지 및 재시작을 제어하는 제어 스크립트입니다.	`/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl` 또는 `/usr/bin/amazon-cloudwatch-agent-ctl`	`$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1`
에이전트가 작성하는 로그 파일입니다. AWS Support에 문의할 경우 이 파일을 첨부해야 할 수도 있습니다.	`/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` 또는 `/var/log/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.log`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log`
에이전트 구성 검증 파일입니다.	`/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log` 또는 `/var/log/amazon/amazon-cloudwatch-agent/configuration-validation.log`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log`
마법사가 에이전트를 생성한 직후에 에이전트를 구성할 때 사용되는 JSON 파일입니다. 자세한 내용은 CloudWatch 에이전트 구성 파일 생성 단원을 참조하세요.	`/opt/aws/amazon-cloudwatch-agent/bin/config.json`	`$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\config.json`
이 구성 파일을 파라미터 스토어에서 다운로드한 경우 에이	`/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-c`

파일	Linux 위치	Windows Server 위치
전트를 구성하는 데 사용되는 JSON 파일.	-agent.json 또는 /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json	loudwatch-agent.json
시스템 기본값을 재정의하여 에이전트가 사용할 리전 및 보안 인증 정보를 지정하는 데 사용되는 TOML 파일입니다.	/opt/aws/amazon-cloudwatch-agent/etc/common-config.toml 또는 /etc/amazon/amazon-cloudwatch-agent/common-config.toml	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\common-config.toml
JSON 구성 파일의 변환된 내용을 포함하는 TOML 파일입니다. amazon-cloudwatch-agent-ctl 스크립트는 이 파일을 생성합니다. 사용자는 이 파일을 직접 수정해서는 안 됩니다. JSON에서 TOML로의 변환이 성공했는지 확인하는 데 유용할 수 있습니다.	/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml 또는 /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.toml	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.toml
JSON 구성 파일의 변환된 내용을 포함하는 YAML 파일입니다. amazon-cloudwatch-agent-ctl 스크립트는 이 파일을 생성합니다. 이 파일을 직접 수정해서는 안 됩니다. 이 파일은 JSON에서 YAML로의 변환이 성공했는지 확인하는 데 유용할 수 있습니다.	/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.yaml or /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.yaml	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.yaml

CloudWatch 에이전트 버전에 관한 정보 찾기

Linux 서버에서 CloudWatch 에이전트의 버전 번호를 찾으려면 다음 명령을 입력합니다.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a status
```

Windows Server에서 CloudWatch 에이전트의 버전 번호를 찾으려면 다음 명령을 입력합니다.

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2
 -a status
```

> **Note**
>
> 이 명령을 사용하면 CloudWatch 에이전트 버전을 올바르게 찾을 수 있습니다. 이와 달리 제어판의 [프로그램 및 기능(Programs and Features)]을 사용하는 경우 잘못된 버전 번호가 표시됩니다.

또한 에이전트의 최신 변경 사항에 대한 README 파일과 현재 다운로드할 수 있는 버전 번호를 나타내는 파일도 다운로드할 수 있습니다. 이러한 파일은 다음 위치에 있습니다.

- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/RELEASE_NOTES` 또는 `https://amazoncloudwatch-agent-`*`region`*`.s3.`*`region`*`.amazonaws.com/info/latest/RELEASE_NOTES`
- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/CWAGENT_VERSION` 또는 `https://amazoncloudwatch-agent-`*`region`*`.s3.`*`region`*`.amazonaws.com/info/latest/CWAGENT_VERSION`

CloudWatch 에이전트가 생성하는 로그

에이전트는 실행 중에 로그를 생성합니다. 이 로그에는 문제 해결 정보가 들어 있습니다. 이 로그는 `amazon-cloudwatch-agent.log` 파일입니다. 이 파일은 Linux 서버의 `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` 및 Windows Server가 실행되는 서버의 `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log`에 있습니다.

`amazon-cloudwatch-agent.log` 파일에 추가 세부 정보를 기록하도록 에이전트를 구성할 수 있습니다. 에이전트 구성 파일의 `agent` 섹션에서 `debug` 필드를 `true`로 설정한 후 CloudWatch 에이전트

를 다시 구성하고 다시 시작합니다. 이 추가 정보 기록을 비활성화하려면 debug 필드를 false로 설정합니다. 그런 다음, 에이전트를 다시 구성하고 다시 시작합니다. 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하십시오.

버전 1.247350.0 이상의 CloudWatch 에이전트에서는 다음 옵션 중 하나 이상을 지정하는 에이전트 구성 파일의 agent 섹션 중 aws_sdk_log_level필드를 선택적으로 설정할 수 있습니다. |문자를 사용하여 여러 옵션을 구분합니다..

- LogDebug
- LogDebugWithSigning
- LogDebugWithHTTPBody
- LogDebugRequestRetries
- LogDebugWithEventStreamBody

이러한 옵션에 대한 자세한 정보는 LogLevelType 단원을 참조하세요.

CloudWatch 에이전트 중지 및 다시 시작

AWS Systems Manager 또는 명령줄을 사용하여 CloudWatch 에이전트를 수동으로 중지할 수 있습니다.

Run Command를 사용하여 CloudWatch 에이전트를 중지하려면

1. https://console.aws.amazon.com/systems-manager/에서 시스템 관리자 콘솔을 엽니다.
2. 탐색 창에서 Run Command를 선택합니다.

   -또는-

   AWS Systems Manager 홈페이지가 열리면 아래로 스크롤하여 [Run Command 탐색(Explore Run Command)]을 선택합니다.
3. Run command(Run 명령)를 선택합니다.
4. 명령 문서 목록에서 AmazonCloudWatch-ManageAgent를 선택합니다.
5. [대상(Targets)] 영역에서 CloudWatch 에이전트를 설치한 인스턴스를 선택합니다.
6. 작업 목록에서 중지를 선택합니다.
7. Optional Configuration Source(구성 소스(선택 사항)) 및 Optional Configuration Location(구성 위치(선택 사항))을 비워둡니다.

8. Run(실행)을 선택합니다.

명령줄을 사용하여 로컬에서 CloudWatch 에이전트를 중지하려면

- Linux 서버의 경우 다음을 입력합니다.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a
 stop
```

  Windows Server가 실행되는 서버의 경우 관리자로서 PowerShell에 다음을 입력합니다.

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m
 ec2 -a stop
```

에이전트를 다시 시작하려면 CloudWatch 에이전트 시작 섹션의 지침을 따르세요.

로그 내에 지표 포함

CloudWatch 임베디드 지표 형식을 사용하면 CloudWatch Logs에 기록된 로그 형식으로 사용자 지정 지표를 비동기식으로 생성할 수 있습니다. 세부 로그 이벤트 데이터와 함께 사용자 지정 지표를 포함할 수 있으며, CloudWatch를 통해 사용자 지정 지표가 자동으로 추출되므로 실시간 인시던트 감지를 위해 해당 지표를 시각화하고 경보를 생성할 수 있습니다. 또한 CloudWatch Logs Insights를 사용하여 추출된 지표와 관련된 세부 로그 이벤트를 쿼리함으로써 운영 이벤트의 근본 원인을 심층적으로 분석할 수 있습니다.

임베디드 지표 형식을 사용하면 Lambda 함수 및 컨테이너와 같은 임시 리소스에서 작업 가능한 사용자 지정 지표를 생성할 수 있습니다. 임베디드 지표 형식을 사용하여 이러한 휘발성 리소스에서 로그를 전송함으로써, 이제 별도의 코드를 구성하거나 유지 관리할 필요 없이 손쉽게 사용자 지정 지표를 생성할 수 있으며 로그 데이터에 대한 강력한 분석 기능을 얻을 수 있습니다.

임베디드 지표 형식을 사용하기 위한 설정은 필요하지 않습니다. 임베디드 지표 형식 사양에 따라 로그를 구성하거나, 클라이언트 라이브러리를 사용하여 로그를 생성한 후 PutLogEvents API 또는 CloudWatch 에이전트를 사용하여 CloudWatch Logs로 전송하세요.

로그 수집 및 보관과 생성된 사용자 지정 지표에 대한 요금이 발생합니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

> **Note**
>
> 지표 추출을 구성할 때는 사용자 지정 지표 사용 및 해당 청구서에 영향을 미치므로 주의해야 합니다. 높은 카디널리티 차원(예: `requestId`)을 기반으로 지표를 실수로 생성하는 경우 기본적으로 임베디드 지표 형식은 각 고유 차원 조합에 해당하는 사용자 지정 지표를 생성합니다. 자세한 내용은 차원 단원을 참조하세요.

주제

- 임베디드 지표 형식을 사용하여 로그 게시
- 콘솔에서 지표 및 로그 보기
- 임베디드 지표 형식으로 만든 지표의 경보 설정

임베디드 지표 형식을 사용하여 로그 게시

다음 방법을 사용하여 임베디드 지표 형식 로그를 생성할 수 있습니다.

- 오픈 소스 클라이언트 라이브러리를 사용하여 로그를 생성하고 전송합니다.
- 임베디드 지표 형식 사양을 사용하여 수동으로 로그를 생성한 다음 CloudWatch 에이전트 또는 PutLogEvents API를 사용하여 로그를 전송합니다.

주제

- 클라이언트 라이브러리를 사용하여 임베디드 지표 형식 로그 생성
- 사양: 임베디드 지표 형식
- PutLogEvents API를 사용하여 수동으로 생성된 임베디드 지표 형식 로그 전송
- CloudWatch 에이전트를 사용하여 임베디드 지표 형식 로그 보내기
- AWS Distro for OpenTelemetry에서 임베디드 지표 형식 사용하기

클라이언트 라이브러리를 사용하여 임베디드 지표 형식 로그 생성

Amazon은 임베디드 지표 형식 로그를 생성하는 데 사용할 수 있는 오픈 소스 클라이언트 라이브러리를 제공합니다. 현재 이러한 라이브러리는 다음 목록에 있는 언어에 대해 사용할 수 있습니다. 다양한 설정에 대한 전체 예는 클라이언트 라이브러리의 /examples 아래에서 확인할 수 있습니다.

라이브러리 및 사용 방법에 대한 지침은 Github에 있습니다. 다음 링크를 사용합니다.

- Node.Js

> **Note**
>
> Node.js 의 경우 Lambda JSON 로그 형식과 함께 사용하려면 4.1.1+, 3.0.2+, 2.0.7+ 버전이 필요합니다. 이러한 Lambda 환경에서 이전 버전을 사용하면 지표 손실이 발생할 수 있습니다.
> 자세한 내용은 AWS Lambda에 대한 Amazon CloudWatch logs 액세스를 참조하세요.

- Python
- Java
- C#

클라이언트 라이브러리는 CloudWatch 에이전트와 함께 즉시 작동하도록 되어 있습니다. 생성된 임베디드 지표 형식 로그는 CloudWatch 에이전트로 전송되며, 이 에이전트는 이를 집계하여 CloudWatch 로그에 게시합니다.

> **Note**
>
> Lambda를 사용하는 경우 로그를 CloudWatch로 전송하는 데 에이전트가 필요하지 않습니다. STDOUT에 기록된 모든 내용은 Lambda 로깅 에이전트를 통해 CloudWatch 로그로 전송됩니다.

사양: 임베디드 지표 형식

CloudWatch 임베디드 지표 형식은 구조화된 로그 이벤트에 포함된 지표 값을 자동으로 추출하도록 CloudWatch Logs에 지시하는 데 사용되는 JSON 사양입니다. CloudWatch를 사용하여 추출된 지표 값에 대한 그래프를 작성하고 경보를 생성할 수 있습니다.

임베디드 지표 형식 사양 규칙

이 형식 사양의 “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY” 및 “OPTIONAL” 키워드는 키워드 RFC2119에 설명된 대로 해석됩니다.

이 형식 사양의 "JSON", "JSON text", "JSON value", "member", "element", "object", "array", "number", "string", "boolean", "true", "false" 및 "null" 용어는 JavaScript 객체 표기법 RFC8259에 정의된 대로 해석됩니다.

> **Note**
>
> 임베디드 지표 형식을 사용하여 만든 지표의 경보를 만들려는 경우 임베디드 지표 형식으로 만든 지표의 경보 설정에서 권장 사항을 참조하세요.

임베디드 지표 형식 문서 구조

이 섹션에서는 임베디드 지표 형식 문서의 구조에 대해 설명합니다. 임베디드 지표 형식 문서는 JavaScript 객체 표기법 RFC8259에 정의되어 있습니다.

별도의 언급이 없는 한 이 사양에 의해 정의된 객체에 추가 멤버가 포함되어서는 안 됩니다. 이 사양에 의해 인식되지 않는 멤버는 무시해야 합니다. 이 사양에 정의된 멤버는 대/소문자를 구분합니다.

임베디드 지표 형식은 표준 CloudWatch Logs 이벤트와 동일한 제한이 적용되며 최대 크기가 256KB로 제한됩니다.

포함된 지표 형식을 사용하면 계정의 AWS/Logs 네임스페이스에 게시된 지표별로 EMF 로그 처리를 추적할 수 있습니다. 이를 사용하여 EMF에서 실패한 지표 생성과 구문 분석 또는 검증으로 인한 오류 발생 여부를 추적할 수 있습니다. 자세한 내용을 알아보려면 CloudWatch 지표를 사용한 모니터링을 참조하세요.

루트 노드

LogEvent 메시지는 LogEvent 메시지 문자열의 시작 또는 끝에 추가 데이터가 없는 유효한 JSON 객체여야 합니다. LogEvent 구조에 대한 자세한 내용은 InputLogEvent를 참조하세요.

임베디드 지표 형식 문서는 루트 노드에 다음과 같은 최상위 멤버를 포함해야 합니다. 이는 메타데이터 객체 객체입니다.

```
{
 "_aws": {
    "CloudWatchMetrics": [ ... ]
  }
}
```

루트 노드에는 MetricDirective 객체의 참조로 정의된 모든 대상 멤버 멤버가 포함되어야 합니다.

루트 노드는 위의 요구 사항에 포함되지 않은 다른 멤버를 포함할 수 있습니다. 이러한 멤버의 값은 유효한 JSON 유형이어야 합니다.

메타데이터 객체

_aws 멤버는 다운스트림 서비스에 LogEvent를 처리하는 방법을 알려주는 페이로드에 대한 메타데이터를 나타내는 데 사용할 수 있습니다. 값은 객체여야 하며 다음 멤버를 포함해야 합니다.

- CloudWatchMetrics - LogEvent의 루트 노드에서 지표를 추출하도록 CloudWatch에 지시하는데 사용되는 MetricDirective 객체의 배열입니다.

  ```
  {
    "_aws": {
      "CloudWatchMetrics": [ ... ]
    }
  }
  ```

- Timestamp - 이벤트에서 추출된 지표에 사용되는 타임스탬프를 나타내는 숫자입니다. 값은 1970년 1월 1일 00:00:00 UTC 이후 경과된 시간(밀리초)으로 표시해야 합니다.

```
{
  "_aws": {
    "Timestamp": 1559748430481
  }
}
```

MetricDirective 객체

MetricDirective 객체는 추출하여 CloudWatch에 게시될 지표를 LogEvent가 포함하도록 다운스트림 서비스에 지시합니다. MetricDirectives에는 다음 멤버가 포함되어야 합니다.

- Namespace - 지표의 CloudWatch 네임스페이스를 나타내는 문자열입니다.
- Dimensions - DimensionSet 배열입니다.
- Metrics - MetricDefinition 객체의 배열입니다. 이 배열에 100개를 초과하는 MetricDefinition 객체가 포함되어서는 안 됩니다.

DimensionSet 배열

DimensionSet는 문서의 모든 지표에 적용될 차원 키를 포함하는 문자열 배열입니다. 또한 이 배열 내의 값은 대상 멤버라고 하는 루트 노드의 멤버여야 합니다.

DimensionSet에는 30개를 초과하는 차원 키가 포함되어서는 안 됩니다. DimensionSet는 비어 있을 수 있습니다.

대상 멤버에는 문자열 값이 있어야 합니다. 이 값은 1024자를 초과하는 문자를 포함해서는 안 됩니다. 대상 멤버는 지표 ID의 일부로 게시될 차원을 정의합니다. 사용된 모든 DimensionSet는 CloudWatch에 새 지표를 생성합니다. 차원에 대한 자세한 내용은 차원 및 차원을 참조하세요.

```
{
 "_aws": {
   "CloudWatchMetrics": [
     {
       "Dimensions": [ [ "functionVersion" ] ],
       ...
     }
   ]
 },
 "functionVersion": "$LATEST"
```

```
}
```

> **Note**
>
> 지표 추출을 구성할 때는 사용자 지정 지표 사용 및 해당 청구서에 영향을 미치므로 주의해야 합니다. 높은 카디널리티 차원(예: `requestId`)을 기반으로 지표를 실수로 생성하는 경우 기본적으로 임베디드 지표 형식은 각 고유 차원 조합에 해당하는 사용자 지정 지표를 생성합니다. 자세한 내용은 차원 단원을 참조하세요.

MetricDefinition 객체

MetricDefinition은 다음 멤버를 포함해야 하는 객체입니다.

- Name - 대상 멤버 지표에 대한 참조 값 문자열입니다. 지표 대상은 숫자 값 또는 숫자 값의 배열이어야 합니다.

MetricDefinition 객체에는 다음 멤버가 포함될 수 있습니다.

- Unit - 해당 지표에 대한 측정 단위를 나타내는 선택적 문자열 값입니다. 값은 유효한 CloudWatch 지표 단위여야 합니다. 유효한 단위에 대한 자세한 내용은 MetricDatum을 참조하세요. 값이 제공되지 않으면 기본값인 NONE으로 가정합니다.
- StorageResolution - 해당 지표에 대한 스토리지 해상도를 나타내는 선택적 정수 값입니다. 이 값을 1로 설정하면 이 지표가 고해상도 지표로 지정되므로 CloudWatch는 1분 미만의 해상도로 지표를 1초까지 저장합니다. 이 값을 60으로 설정하면 이 지표가 표준 해상도로 지정되며, CloudWatch는 이를 1분 해상도로 저장합니다. 값은 유효한 CloudWatch 지원 해상도(1 또는 60)여야 합니다. 값이 제공되지 않으면 기본값인 60으로 가정합니다.

  고분해능 지표에 대한 자세한 내용은 고분해능 지표 단원을 참조하세요.

> **Note**
>
> 임베디드 지표 형식을 사용하여 만든 지표의 경보를 만들려는 경우 임베디드 지표 형식으로 만든 지표의 경보 설정에서 권장 사항을 참조하세요.

```
{
```

```
  "_aws": {
    "CloudWatchMetrics": [
      {
        "Metrics": [
          {
            "Name": "Time",
            "Unit": "Milliseconds",
            "StorageResolution": 60
          }
        ],
        ...
      }
    ]
  },
  "Time": 1
}
```

참조 값

참조 값은 루트 노드의 대상 멤버 멤버를 참조하는 문자열 값입니다. 이러한 참조를 RFC6901에 설명된 JSON 포인터와 혼동해서는 안 됩니다. 대상 값은 중첩될 수 없습니다.

대상 멤버

유효한 대상은 루트 노드의 멤버여야 하며 중첩된 객체일 수 없습니다. 예를 들어 `"A.a"`의 _reference_ 값이 다음 멤버와 일치해야 합니다.

```
{ "A.a" }
```

중첩된 멤버와 일치하지 않아야 합니다.

```
{ "A": { "a" } }
```

대상 멤버의 유효한 값은 해당 멤버를 참조하는 내용에 따라 달라집니다. 지표 대상은 숫자 값 또는 숫자 값의 배열이어야 합니다. 숫자 배열 지표 대상에는 100개가 넘는 멤버가 있어서는 안 됩니다. 차원 대상에는 문자열 값이 있어야 합니다.

임베디드 지표 형식 예시 및 JSON 스키마

다음은 임베디드 지표 형식의 유효한 예입니다.

```
{
  "_aws": {
    "Timestamp": 1574109732004,
    "CloudWatchMetrics": [
      {
        "Namespace": "lambda-function-metrics",
        "Dimensions": [["functionVersion"]],
        "Metrics": [
          {
            "Name": "time",
            "Unit": "Milliseconds",
            "StorageResolution": 60
          }
        ]
      }
    ]
  },
  "functionVersion": "$LATEST",
  "time": 100,
  "requestId": "989ffbf8-9ace-4817-a57c-e4dd734019ee"
}
```

다음 스키마를 사용하여 임베디드 지표 형식 문서의 유효성을 검사할 수 있습니다.

```
{
    "type": "object",
    "title": "Root Node",
    "required": [
        "_aws"
    ],
    "properties": {
        "_aws": {
            "$id": "#/properties/_aws",
            "type": "object",
            "title": "Metadata",
            "required": [
                "Timestamp",
                "CloudWatchMetrics"
            ],
            "properties": {
                "Timestamp": {
                    "$id": "#/properties/_aws/properties/Timestamp",
                    "type": "integer",
```

```
                    "title": "The Timestamp Schema",
                    "examples": [
                        1565375354953
                    ]
                },
                "CloudWatchMetrics": {
                    "$id": "#/properties/_aws/properties/CloudWatchMetrics",
                    "type": "array",
                    "title": "MetricDirectives",
                    "items": {
                        "$id": "#/properties/_aws/properties/CloudWatchMetrics/items",
                        "type": "object",
                        "title": "MetricDirective",
                        "required": [
                            "Namespace",
                            "Dimensions",
                            "Metrics"
                        ],
                        "properties": {
                            "Namespace": {
                                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Namespace",
                                "type": "string",
                                "title": "CloudWatch Metrics Namespace",
                                "examples": [
                                    "MyApp"
                                ],
                                "pattern": "^(.*)$",
                                "minLength": 1,
                                "maxLength": 1024
                            },
                            "Dimensions": {
                                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Dimensions",
                                "type": "array",
                                "title": "The Dimensions Schema",
                                "minItems": 1,
                                "items": {
                                    "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Dimensions/items",
                                    "type": "array",
                                    "title": "DimensionSet",
                                    "minItems": 0,
                                    "maxItems": 30,
```

```
                                    "items": {
                                        "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Dimensions/items/items",
                                        "type": "string",
                                        "title": "DimensionReference",
                                        "examples": [
                                            "Operation"
                                        ],
                                        "pattern": "^(.*)$",
                                        "minLength": 1,
                                        "maxLength": 250
}
                                }
                            },
                            "Metrics": {
                                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Metrics",
                                "type": "array",
                                "title": "MetricDefinitions",
                                "items": {
                                    "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items",
                                    "type": "object",
                                    "title": "MetricDefinition",
                                    "required": [
                                        "Name"
                                    ],
                                    "properties": {
                                        "Name": {
                                            "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Name",
                                            "type": "string",
                                            "title": "MetricName",
                                            "examples": [
                                                "ProcessingLatency"
                                            ],
                                            "pattern": "^(.*)$",
                                            "minLength": 1,
                                            "maxLength": 1024
                                        },
                                        "Unit": {
                                            "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Unit",
                                            "type": "string",
```

```
                                        "title": "MetricUnit",
                                        "examples": [
                                            "Milliseconds"
                                        ],
                                        "pattern": "^(Seconds|Microseconds|
Milliseconds|Bytes|Kilobytes|Megabytes|Gigabytes|Terabytes|Bits|Kilobits|Megabits|
Gigabits|Terabits|Percent|Count|Bytes\\/Second|Kilobytes\\/Second|Megabytes\\/Second|
Gigabytes\\/Second|Terabytes\\/Second|Bits\\/Second|Kilobits\\/Second|Megabits\\/
Second|Gigabits\\/Second|Terabits\\/Second|Count\\/Second|None)$"
                                    },
                                    "StorageResolution": {
                                        "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/StorageResolution",
                                        "type": "integer",
                                        "title": "StorageResolution",
                                        "examples": [
                                            60
                                        ]
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

PutLogEvents API를 사용하여 수동으로 생성된 임베디드 지표 형식 로그 전송

CloudWatch Logs PutLogEvents API를 사용하여 임베디드 지표 형식 로그를 CloudWatch Logs에 전송할 수 있습니다. PutLogEvents를 호출할 때 지표를 추출해야 한다고 CloudWatch Logs에 지시하기 위해 다음 HTTP 헤더를 포함할 수도 있지만, 더 이상 포함할 필요는 없습니다.

```
x-amzn-logs-format: json/emf
```

다음은 AWS SDK for Java 2.x를 사용하는 전체 예입니다.

```
package org.example.basicapp;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.InputLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.PutLogEventsRequest;

import java.util.Collections;

public class EmbeddedMetricsExample {
        public static void main(String[] args) {

                final String usage = "To run this example, supply a Region code (eg.
 us-east-1), log group, and stream name as command line arguments"
                                + "Ex: PutLogEvents <region-id> <log-group-name>
 <stream-name>";

                if (args.length != 3) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String regionId = args[0];
                String logGroupName = args[1];
                String logStreamName = args[2];

                CloudWatchLogsClient logsClient =
 CloudWatchLogsClient.builder().region(Region.of(regionId)).build();

                // Build a JSON log using the EmbeddedMetricFormat.
                long timestamp = System.currentTimeMillis();
                String message = "{" +
                                "  \"_aws\": {" +
                                "    \"Timestamp\": " + timestamp  + "," +
                                "    \"CloudWatchMetrics\": [" +
                                "      {" +
                                "        \"Namespace\": \"MyApp\"," +
                                "        \"Dimensions\": [[\"Operation\"], [\"Operation
\", \"Cell\"]]," +
                                "        \"Metrics\": [{ \"Name\": \"ProcessingLatency
\", \"Unit\": \"Milliseconds\", \"StorageResolution\": 60 }]" +
                                "      }" +
                                "    ]" +
```

```
                                "  },\" +
                                "  \"Operation\": \"Aggregator\"," +
                                "  \"Cell\": \"001\"," +
                                "  \"ProcessingLatency\": 100" +
                                "}";
                InputLogEvent inputLogEvent = InputLogEvent.builder()
                        .message(message)
                        .timestamp(timestamp)
                        .build();

                // Specify the request parameters.
                PutLogEventsRequest putLogEventsRequest = PutLogEventsRequest.builder()
                        .logEvents(Collections.singletonList(inputLogEvent))
                        .logGroupName(logGroupName)
                        .logStreamName(logStreamName)
                        .build();

                logsClient.putLogEvents(putLogEventsRequest);

                System.out.println("Successfully put CloudWatch log event");
        }

}
```

> **Note**
>
> 포함된 지표 형식을 사용하면 계정의 AWS/Logs 네임스페이스에 게시된 지표별로 EMF 로그 처리를 추적할 수 있습니다. 이를 사용하여 EMF에서 실패한 지표 생성과 구문 분석 또는 검증으로 인한 오류 발생 여부를 추적할 수 있습니다. 자세한 내용을 알아보려면 CloudWatch 지표를 사용한 모니터링을 참조하세요.

CloudWatch 에이전트를 사용하여 임베디드 지표 형식 로그 보내기

이 방법을 사용하려면 먼저, 임베디드 지표 형식 로그를 전송하려는 서비스에 대한 CloudWatch 에이전트를 설치해야 합니다. 그런 다음, 이벤트 전송을 시작할 수 있습니다.

CloudWatch 에이전트는 1.230621.0 이상 버전이어야 합니다.

> **Note**
>
> Lambda 함수에서 로그를 전송하기 위해 CloudWatch 에이전트를 설치할 필요가 없습니다.

Lambda 함수 시간 초과는 자동으로 처리되지 않습니다. 즉, 지표가 플러시되기 전에 함수가 시간 초과되면 해당 호출에 대한 지표가 캡처되지 않습니다.

CloudWatch 에이전트 설치

임베디드 지표 형식 로그를 전송할 각 서비스에 대한 CloudWatch 에이전트를 설치합니다.

EC2에 CloudWatch 에이전트 설치하기

먼저, 인스턴스에 CloudWatch 에이전트를 설치합니다. 자세한 내용은 CloudWatch 에이전트 설치 단원을 참조하세요.

에이전트를 설치한 후에는 임베디드 지표 형식 로그에 대해 UDP 또는 TCP 포트에서 수신 대기하도록 에이전트를 구성합니다. 다음은 기본 소켓 `tcp:25888`에서 수신 대기하는 이 구성의 예입니다. 에이전트 구성에 대한 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하세요.

```
{
  "logs": {
    "metrics_collected": {
      "emf": { }
    }
  }
}
```

Amazon ECS에 CloudWatch 에이전트 설치하기

Amazon ECS에 CloudWatch 에이전트를 배포하는 가장 쉬운 방법은 에이전트를 사이드카로 실행하여 애플리케이션과 동일한 태스크 정의에서 정의하는 것입니다.

에이전트 구성 파일 생성

CloudWatch 에이전트 구성 파일을 로컬에 생성합니다. 이 예제에서 상대 파일 경로는 `amazon-cloudwatch-agent.json`입니다.

에이전트 구성에 대한 자세한 내용은 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집 단원을 참조하세요.

```
{
  "logs": {
```

```
    "metrics_collected": {
      "emf": { }
    }
  }
}
```

SSM 파라미터 스토어로 구성 푸시

다음 명령을 입력하여 CloudWatch 에이전트 구성 파일을 AWS Systems Manager(SSM) 파라미터 스토어로 푸시합니다.

```
aws ssm put-parameter \
    --name "cwagentconfig" \
    --type "String" \
    --value "`cat amazon-cloudwatch-agent.json`" \
    --region "{{region}}"
```

태스크 정의 구성

CloudWatch 에이전트를 사용하고 TCP 또는 UDP 포트를 노출하도록 태스크 정의를 구성합니다. 사용해야 하는 샘플 작업 정의는 네트워킹 모드에 따라 다릅니다.

webapp에서 AWS_EMF_AGENT_ENDPOINT 환경 변수를 지정합니다. 이 기능은 라이브러리에서 사용되며 에이전트가 수신 대기하는 엔드포인트를 가리켜야 합니다. 또한 cwagent에서는 CW_CONFIG_CONTENT를 이전 단계에서 생성한 SSM 구성을 가리키는 “valueFrom” 파라미터로 지정합니다.

이 단원에는 브리지 모드에 대한 예 하나와 호스트 또는 awsvpc 모드에 대한 예 하나가 포함되어 있습니다. Amazon ECS에서 CloudWatch 에이전트를 구성하는 방법에 대한 추가 예는 Github 샘플 리포지토리를 참조하세요.

다음은 브리지 모드의 예입니다. 브리지 모드 네트워킹을 사용 설정한 경우 links 파라미터를 사용하여 에이전트를 애플리케이션에 연결해야 하며 컨테이너 이름을 사용하여 에이전트 주소를 지정해야 합니다.

```
{
  "containerDefinitions": [
          {
              "name": "webapp",
              "links": [ "cwagent" ],
              "image": "my-org/web-app:latest",
```

```
                "memory": 256,
                "cpu": 256,
                "environment": [{
                  "name": "AWS_EMF_AGENT_ENDPOINT",
                  "value": "tcp://cwagent:25888"
                }],
            },
            {
                "name": "cwagent",
                "mountPoints": [],
                "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
                "memory": 256,
                "cpu": 256,
                "portMappings": [{
                    "protocol": "tcp",
                    "containerPort": 25888
                }],
                "environment": [{
                  "name": "CW_CONFIG_CONTENT",
                  "valueFrom": "cwagentconfig"
                }],
            }
        ],
}
```

다음은 호스트 모드 또는 awsvpc 모드의 예입니다. 이러한 네트워크 모드에서 실행할 경우 에이전트는 `localhost`를 통해 주소를 지정할 수 있습니다.

```
{
  "containerDefinitions": [
            {
                "name": "webapp",
                "image": "my-org/web-app:latest",
                "memory": 256,
                "cpu": 256,
                "environment": [{
                  "name": "AWS_EMF_AGENT_ENDPOINT",
                  "value": "tcp://127.0.0.1:25888"
                }],
            },
            {
                "name": "cwagent",
                "mountPoints": [],
```

```
                "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
                "memory": 256,
                "cpu": 256,
                "portMappings": [{
                    "protocol": "tcp",
                    "containerPort": 25888
                }],
                "environment": [{
                  "name": "CW_CONFIG_CONTENT",
                  "valueFrom": "cwagentconfig"
                }],
            }
        ],
}
```

> **Note**
>
> awsvpc 모드에서는 VPC에 퍼블릭 IP 주소를 제공하거나(Fargate만 해당) NAT 게이트웨이를 설정하거나 CloudWatch Logs VPC 엔드포인트를 설정해야 합니다. NAT 설정에 대한 자세한 내용은 NAT 게이트웨이를 참조하세요. CloudWatch Logs VPC 엔드포인트 설정에 대한 자세한 내용은 인터페이스 VPC 엔드포인트와 함께 CloudWatch Logs 사용 단원을 참조하세요. 다음은 Fargate 시작 유형을 사용하는 작업에 퍼블릭 IP 주소를 할당하는 방법의 예입니다.
>
> ```
> aws ecs run-task \
> --cluster {{cluster-name}} \
> --task-definition cwagent-fargate \
> --region {{region}} \
> --launch-type FARGATE \
> --network-configuration
>  "awsvpcConfiguration={subnets=[{{subnetId}}],securityGroups=[{{sgId}}],assignPublicIp=ENA
> ```

권한 확인

태스크를 실행하는 IAM 역할에 SSM 파라미터 스토어에서 읽을 수 있는 권한이 있는지 확인합니다. AmazonSSMReadOnlyAccess 정책을 연결하여 이 권한을 추가할 수 있습니다. 이를 위해 다음 명령을 입력합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess
 \
```

```
--role-name CWAgentECSExecutionRole
```

Amazon EKS에 CloudWatch 에이전트 설치하기

이 클러스터에 CloudWatch Container Insights를 이미 설치한 경우 이 프로세스의 일부를 건너뛸 수 있습니다.

권한

Container Insights를 아직 설치하지 않았다면 먼저, Amazon EKS 노드에 적절한 IAM 권한이 있는지 확인합니다. CloudWatchAgentServerPolicy가 연결되어 있어야 합니다. 자세한 내용은 사전 조건 확인 단원을 참조하세요.

ConfigMap 생성

에이전트에 대한 ConfigMap을 생성합니다. ConfigMap은 또한 TCP 또는 UDP 포트에서 수신 대기하도록 에이전트에 지시합니다. 다음 ConfigMap을 사용합니다.

```
# cwagent-emf-configmap.yaml
apiVersion: v1
data:
  # Any changes here must not break the JSON format
  cwagentconfig.json: |
    {
      "agent": {
        "omit_hostname": true
      },
      "logs": {
        "metrics_collected": {
          "emf": { }
        }
      }
    }
kind: ConfigMap
metadata:
  name: cwagentemfconfig
  namespace: default
```

Container Insights를 이미 설치한 경우 기존 ConfigMap에 다음 `"emf": { }` 줄을 추가합니다.

ConfigMap 적용

다음 명령을 입력하여 ConfigMap을 적용합니다.

```
kubectl apply -f cwagent-emf-configmap.yaml
```

에이전트 배포

CloudWatch 에이전트를 사이드카로 배포하려면 다음 예와 같이 포드 정의에 에이전트를 추가합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  namespace: default
spec:
  containers:
    # Your container definitions go here
    - name: web-app
      image: my-org/web-app:latest
    # CloudWatch Agent configuration
    - name: cloudwatch-agent
      image: public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest
      imagePullPolicy: Always
      resources:
        limits:
          cpu: 200m
          memory: 100Mi
        requests:
          cpu: 200m
          memory: 100Mi
      volumeMounts:
        - name: cwagentconfig
          mountPath: /etc/cwagentconfig
      ports:
  # this should match the port configured in the ConfigMap
        - protocol: TCP
          hostPort: 25888
          containerPort: 25888
  volumes:
    - name: cwagentconfig
      configMap:
        name: cwagentemfconfig
```

CloudWatch 에이전트를 사용하여 임베디드 지표 형식 로그 보내기

CloudWatch 에이전트를 설치하고 실행 중인 경우 TCP 또는 UDP를 통해 임베디드 지표 형식 로그를 전송할 수 있습니다. 에이전트를 통해 로그를 전송할 때 다음 두 가지 요구 사항이 있습니다.

- 로그에는 사용할 로그 그룹을 에이전트에 알려주는 `LogGroupName` 키가 있어야 합니다.
- 각 로그 이벤트는 한 줄에 있어야 합니다. 즉, 로그 이벤트에는 줄 바꿈 문자(\n)가 포함될 수 없습니다.

또한 로그 이벤트는 임베디드 지표 형식 사양을 따라야 합니다. 자세한 내용은 사양: 임베디드 지표 형식 섹션을 참조하세요.

임베디드 지표 형식을 사용하여 만든 지표의 경보를 만들려는 경우 임베디드 지표 형식으로 만든 지표의 경보 설정에서 권장 사항을 참조하세요.

다음은 Linux bash 셸에서 수동으로 로그 이벤트를 전송하는 예제입니다. 선택한 프로그래밍 언어에서 제공하는 UDP 소켓 인터페이스를 대신 사용할 수 있습니다.

```
echo '{"_aws":{"Timestamp":1574109732004,"LogGroupName":"Foo","CloudWatchMetrics":
[{"Namespace":"MyApp","Dimensions":[["Operation"]],"Metrics":
[{"Name":"ProcessingLatency","Unit":"Milliseconds","StorageResolution":60}]}]},"Operation":"Agg
 \
> /dev/udp/0.0.0.0/25888
```

> **Note**
>
> 포함된 지표 형식을 사용하면 계정의 `AWS/Logs` 네임스페이스에 게시된 지표별로 EMF 로그 처리를 추적할 수 있습니다. 이를 사용하여 EMF에서 실패한 지표 생성과 구문 분석 또는 검증으로 인한 오류 발생 여부를 추적할 수 있습니다. 자세한 내용을 알아보려면 CloudWatch 지표를 사용한 모니터링을 참조하세요.

AWS Distro for OpenTelemetry에서 임베디드 지표 형식 사용하기

임베디드 지표 형식을 OpenTelemetry 프로젝트의 일부로 사용할 수 있습니다. OpenTelemetry는 단일 사양 및 API 세트를 제공함으로써 추적, 로그, 지표에 대한 공급 업체별 형식 간의 경계 및 제한을 제거하는 오픈 소스 이니셔티브입니다. 자세한 내용은 OpenTelemetry를 참조하세요.

OpenTelemetry에서 임베디드 지표 형식을 사용하려면 두 가지 구성 요소, 즉 OpenTelemetry 호환 데이터 원본 및 CloudWatch 임베디드 지표 형식 로그와 함께 사용하도록 설정된 AWS Distro for OpenTelemetry Collector가 필요합니다.

온보딩을 최대한 쉽게 하기 위해 AWS에서 유지 관리하는 OpenTelemetry 구성 요소의 재배포를 미리 구성했습니다. 다른 AWS 서비스 외에도 임베디드 지표 형식으로 OpenTelemetry를 사용하는 방법에 대한 자세한 내용은 AWS Distro for OpenTelemetry를 참조하세요.

언어 지원 및 사용에 대한 추가 정보는 Github의 AWS 관찰 가능성을 참조하세요.

콘솔에서 지표 및 로그 보기

지표를 추출하는 임베디드 지표 형식 로그를 생성한 후 CloudWatch 콘솔을 사용하여 지표를 볼 수 있습니다. 임베디드 지표에는 로그를 생성할 때 지정한 차원이 있습니다. 또한 클라이언트 라이브러리를 사용하여 생성한 임베디드 지표에는 다음과 같은 기본 차원이 있습니다.

- ServiceType
- ServiceName
- LogGroup

임베디드 지표 형식 로그에서 생성된 지표를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 임베디드 지표를 생성할 때 해당 지표에 지정한 네임스페이스를 선택합니다. 클라이언트 라이브러리를 사용하여 지표를 생성하고 네임스페이스를 지정하지 않은 경우 aws-embedded-metrics를 선택합니다. 클라이언트 라이브러리를 사용하여 생성된 임베디드 지표의 기본 네임스페이스입니다.
4. 지표 차원(예: ServiceName)을 선택합니다.
5. 모든 지표 탭에 네임스페이스의 해당 측정기준에 대한 모든 지표가 표시됩니다. 다음을 수행할 수 있습니다.

    a. 테이블을 정렬하려면 열 머리글을 사용합니다.

    b. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택하세요. 모든 지표를 선택하려면 테이블의 머리글 행에 있는 확인란을 선택하세요.

    c. 리소스로 필터링하려면 리소스 ID를 선택한 후 검색에 추가를 선택합니다.

d. 지표로 필터링하려면 지표 이름을 선택한 후 검색에 추가를 선택합니다.

CloudWatch Logs Insights를 사용하여 로그 쿼리

CloudWatch Logs Insights를 사용하여 추출된 지표와 연결된 세부 로그 이벤트를 쿼리함으로써 운영 이벤트의 근본 원인에 대한 심층적인 인사이트를 제공할 수 있습니다. 로그에서 지표를 추출할 때 얻을 수 있는 이점 중 하나는 나중에 고유한 지표(지표 이름 및 고유한 차원 집합) 및 지표 값을 기준으로 로그를 필터링하여 집계된 지표 값에 기여한 이벤트에 대한 컨텍스트를 가져올 수 있다는 것입니다.

예를 들어 영향을 받는 요청 ID 또는 X-Ray 추적 ID를 얻으려면 CloudWatch Logs Insights에서 다음 쿼리를 실행하면 됩니다.

```
filter Latency > 1000 and Operation = "Aggregator"
| fields RequestId, TraceId
```

이벤트의 영향을 받는 고객을 찾는 등 높은 카디널리티 키에 대해 쿼리 시간 집계를 수행할 수도 있습니다. 다음 예는 이를 보여 줍니다.

```
filter Latency > 1000 and Operation = "Aggregator"
| stats count() by CustomerId
```

자세한 내용은 CloudWatch Logs Insights를 사용하여 로그 데이터 분석을 참조하세요.

임베디드 지표 형식으로 만든 지표의 경보 설정

일반적으로 임베디드 지표 형식으로 생성된 지표의 경보를 만드는 것은 다른 지표에 대한 경보를 만드는 것과 패턴이 동일합니다. 자세한 내용은 Amazon CloudWatch 경보 사용 섹션을 참조하세요.

임베디드 지표 형식 지표 생성은 로그 게시 흐름에 따라 달라지는데 이는 로그를 지표로 변환하려면 CloudWatch Logs에서 로그를 처리해야 하기 때문입니다. 따라서 경보가 평가되는 기간 내에 지표 데이터 포인트가 생성될 수 있도록 적시에 로그를 게시하는 것이 중요합니다.

임베디드 지표 형식을 사용하여 고분해능 지표를 전송하고 이러한 지표에 대한 경보를 생성하려는 경우, 일부 또는 누락된 데이터에 대한 경보를 일으킬 수 있는 추가 지연이 발생하지 않도록 5초 이하의 간격으로 로그를 CloudWatch 로그에 플러시할 것을 권장합니다. CloudWatch 에이전트를 사용하는 경우 CloudWatch 에이전트 구성 파일에서 `force_flush_interval` 파라미터를 설정하여 플러시 간격을 조정할 수 있습니다. 이 값의 기본값은 5초입니다.

로그 플러시 간격을 제어할 수 없는 다른 플랫폼에서 Lambda를 사용하는 경우, 경보에 사용되는 데이터 포인트 수를 제어하기 위해 'N 중 M' 경보를 사용하는 것을 고려하세요. 자세한 내용은 경보 평가 섹션을 참조하세요.

CloudWatch 지표를 게시하는 AWS 서비스

다음 AWS 서비스는 지표를 CloudWatch에 게시합니다. 지표 및 측정기준에 대한 자세한 내용은 지정된 설명서를 참조하세요.

Service	네임스페이스	설명서
AWS Amplify	`AWS/AmplifyHosting`	모니터링(Monitoring)
Amazon API Gateway	`AWS/ApiGateway`	Amazon CloudWatch를 사용한 API 실행 모니터링
Amazon AppFlow	`AWS/AppFlow`	Amazon CloudWatch로 Amazon AppFlow 모니터링
AWS 애플리케이션 마이그레이션 서비스	`AWS/MGN`	Amazon CloudWatch를 사용한 Application Migration Service 모니터링
AWS App Runner	`AWS/AppRunner`	CloudWatch에 보고된 App Runner 서비스 지표 보기
AppStream 2.0	`AWS/AppStream`	Amazon AppStream 2.0 리소스 모니터링
AWS AppSync	`AWS/AppSync`	CloudWatch 지표
Amazon Athena	`AWS/Athena`	CloudWatch 지표를 사용한 Athena 쿼리 모니터링
Amazon Aurora	`AWS/RDS`	Amazon Aurora 지표
AWS Backup	`AWS/Backup`	CloudWatch로 AWS 백업 지표 모니터링
Amazon Bedrock	`AWS/Bedrock`	Amazon CloudWatch를 사용한 Amazon Bedrock 모니터링

Service	네임스페이스	설명서
AWS Billing and Cost Management	AWS/Billing	알림을 사용한 요금 모니터링
Amazon Braket	AWS/Braket/By Device	Amazon CloudWatch를 사용한 Amazon Braket 모니터링
AWS Certificate Manager	AWS/CertificateManager	지원되는 CloudWatch 지표
AWS Private CA	AWS/ACMPrivateCA	지원되는 CloudWatch 지표
AWS Chatbot	AWS/Chatbot	Amazon CloudWatch를 사용한 AWS Chatbot 모니터링
Amazon Chime	AWS/ChimeVoiceConnector	Amazon CloudWatch를 사용한 Amazon Chime 모니터링
Amazon Chime SDK	AWS/ChimeSDK	서비스 지표
AWS Client VPN	AWS/ClientVPN	Amazon CloudWatch로 모니터링
Amazon CloudFront	AWS/CloudFront	CloudWatch를 사용한 CloudFront 활동 모니터링
AWS CloudHSM	AWS/CloudHSM	CloudWatch 지표 가져오기
Amazon CloudSearch	AWS/CloudSearch	Amazon CloudWatch를 사용한 Amazon CloudSearch 도메인 모니터링
AWS CloudTrail	AWS/CloudTrail	지원되는 CloudWatch 지표

Service	네임스페이스	설명서
CloudWatch 에이전트	`CWAgent` 또는 사용자 지정 네임스페이스	CloudWatch 에이전트가 수집하는 지표
CloudWatch 지표 스트림	`AWS/CloudWatch/MetricStreams`	CloudWatch 지표를 사용하여 지표 스트림 모니터링
CloudWatch RUM	`AWS/RUM`	CloudWatch RUM을 사용하여 수집할 수 있는 CloudWatch 지표
CloudWatch Synthetics	`CloudWatchSynthetics`	canary가 게시한 CloudWatch 지표
Amazon CloudWatch Logs	`AWS/Logs`	CloudWatch 지표를 사용한 사용량 모니터링
AWS CodeBuild	`AWS/CodeBuild`	AWS CodeBuild 모니터링
Amazon CodeGuru Reviewer		Amazon CloudWatch를 사용한 CodeGuru Reviewer 모니터링
Amazon Kendra		Amazon CloudWatch를 사용한 Amazon Kendra 모니터링
Amazon CodeWhisperer	`AWS/CodeWhisperer`	Amazon CloudWatch를 사용한 Amazon CodeWhisperer 모니터링
Amazon Cognito	`AWS/Cognito`	Amazon Cognito 모니터링
Amazon Comprehend	`AWS/Comprehend`	Amazon Comprehend 엔드포인트 모니터링
AWS Config	`AWS/Config`	AWS Config 사용량 및 성공 지표

Service	네임스페이스	설명서
Amazon Connect	AWS/Connect	Amazon CloudWatch 지표의 Amazon Connect 모니터링
Amazon Data Lifecycle Manager	AWS/DataLifecycleManager	Amazon CloudWatch를 사용하여 정책 모니터링
AWS DataSync	AWS/DataSync	작업 모니터링
Amazon DataZone		Amazon CloudWatch를 사용한 Amazon DataZone 모니터링
Amazon DevOps Guru	AWS/DevOps-Guru	Amazon CloudWatch를 사용한 Amazon DevOps Guru 모니터링
AWS Database Migration Service	AWS/DMS	AWS DMS 작업 모니터링
AWS Direct Connect	AWS/DX	Amazon CloudWatch로 모니터링
AWS Directory Service	AWS/DirectoryService	Amazon CloudWatch 지표를 사용하여 도메인 컨트롤러를 추가할 시기를 결정합니다.
Amazon DocumentDB	AWS/DocDB	Amazon DocumentDB 지표
Amazon DynamoDB	AWS/DynamoDB	DynamoDB 지표 및 측정기준
DynamoDB Accelerator(DAX)	AWS/DAX	DAX 지표 및 측정기준 보기
Amazon EC2	AWS/EC2	CloudWatch를 사용한 인스턴스 모니터링

Service	네임스페이스	설명서
Amazon EC2 Elastic Graphics	AWS/ElasticGPUs	CloudWatch 지표를 사용하여 Elastic Graphics 모니터링
Amazon EC2 스팟 플릿	AWS/EC2Spot	스팟 집합에 대한 CloudWatch 지표
Amazon EC2 Auto Scaling	AWS/AutoScaling	CloudWatch를 사용하여 Auto Scaling 그룹 및 인스턴스 모니터링
AWS Elastic Beanstalk	AWS/ElasticBeanstalk	환경에 대한 Amazon CloudWatch 사용자 지정 지표 게시
Amazon Elastic Block Store	AWS/EBS	Amazon EBS의 Amazon CloudWatch 지표
Amazon Elastic 컨테이너 레지스트리	AWS/ECR	Amazon ECR 리포지토리 지표
Amazon Elastic Container Service	AWS/ECS	Amazon ECS CloudWatch 지표
CloudWatch Container Insights를 통한 Amazon ECS	ECS/ContainerInsights	Amazon ECS Container Insights 지표
Amazon ECS 클러스터 auto scaling	AWS/ECS/ManagedScaling	Amazon ECS 클러스터 auto scaling
AWS Elastic Disaster Recovery		DRS에 대한 CloudWatch 지표

Service	네임스페이스	설명서
Amazon Elastic File System	AWS/EFS	CloudWatch를 사용한 모니터링
Amazon Elastic Inference	AWS/ElasticInference	CloudWatch 지표를 사용하여 Amazon Elastic Inference 모니터링
CloudWatch Container Insights를 통한 Amazon EKS	ContainerInsights	Amazon EKS 및 Kubernetes Container Insights 지표
Elastic Load Balancing	AWS/ApplicationELB	Application Load Balancer의 CloudWatch 지표
Elastic Load Balancing	AWS/NetworkELB	Network Load Balancer의 CloudWatch 지표
Elastic Load Balancing	AWS/GatewayELB	Gateway Load Balancer의 CloudWatch 지표
Elastic Load Balancing	AWS/ELB	Classic Load Balancer의 CloudWatch 지표
Amazon Elastic Transcoder	AWS/ElasticTranscoder	Amazon CloudWatch로 모니터링
Amazon ElastiCache for Memcached	AWS/ElastiCache	CloudWatch 지표를 사용한 사용량 모니터링
Amazon ElastiCache for Redis	AWS/ElastiCache	CloudWatch 지표를 사용한 사용량 모니터링

Service	네임스페이스	설명서
Amazon OpenSearch Service	AWS/ES	Amazon CloudWatch로 OpenSearch 클러스터 지표 모니터링
Amazon EMR	AWS/ElasticMapReduce	CloudWatch를 사용한 지표 모니터링
AWS Elemental MediaConnect	AWS/MediaConnect	Amazon CloudWatch를 사용한 MediaConnect 모니터링
AWS Elemental MediaConvert	AWS/MediaConvert	CloudWatch 지표를 사용하여 AWS Elemental MediaConvert 리소스 지표 보기
AWS Elemental MediaLive	AWS/MediaLive	Amazon CloudWatch 지표를 사용하여 활동 모니터링
AWS Elemental MediaPackage	AWS/MediaPackage	Amazon CloudWatch 지표를 사용한 AWS Elemental MediaPackage 모니터링
AWS Elemental MediaStore	AWS/MediaStore	Amazon CloudWatch 지표를 사용한 AWS Elemental MediaStore 모니터링
AWS Elemental MediaTailor	AWS/MediaTailor	Amazon CloudWatch를 사용한 AWS Elemental MediaTailor 모니터링
Amazon EventBridge	AWS/Events	Amazon EventBridge 모니터링
Amazon FinSpace		로깅 및 모니터링
Amazon Forecast		Amazon Forecast에 대한 CloudWatch 지표
Amazon Fraud Detector		Amazon CloudWatch를 사용한 Amazon Fraud Detector 모니터링

Service	네임스페이스	설명서
Amazon FSx for Lustre	AWS/FSx	Amazon FSx for Lustre 모니터링
Amazon FSx for OpenZFS	AWS/FSx	Amazon CloudWatch를 사용한 모니터링
Amazon FSx for Windows File Server	AWS/FSx	Amazon FSx for Windows File Server 모니터링
Amazon FSx for NetApp ONTAP CSI 드라이버	AWS/FSx	Amazon CloudWatch를 사용한 모니터링
Amazon FSx for OpenZFS	AWS/FSx	Amazon CloudWatch를 사용한 모니터링
Amazon GameLift	AWS/GameLift	CloudWatch를 사용한 Amazon GameLift 모니터링
AWS Global Accelerator	AWS/GlobalAccelerator	AWS Global Accelerator와 함께 Amazon CloudWatch 사용
AWS Glue	Glue	CloudWatch 지표를 사용한 AWS Glue 모니터링
AWS Ground Station	AWS/GroundStation	Amazon CloudWatch를 사용한 지표
AWS HealthLake	AWS/HealthLake	CloudWatch를 사용한 HealthLake 모니터링
Amazon Inspector	AWS/Inspector	CloudWatch를 사용하여 Amazon Inspector 모니터링
Amazon Interactive Video Service	AWS/IVS	Amazon CloudWatch에서 Amazon IVS 모니터링

Service	네임스페이스	설명서
Amazon Interactive Video Service	AWS/IVSChat	Amazon CloudWatch에서 Amazon IVS 모니터링
AWS IoT	AWS/IoT	AWS IoT 지표 및 측정기준
AWS IoT Analytics	AWS/IoTAnalytics	네임스페이스, 지표 및 측정기준
AWS IoT FleetWise	AWS/IoTFleetWise	Amazon CloudWatch로 AWS IoT FleetWise 모니터링
AWS IoT SiteWise	AWS/IoTSiteWise	Amazon CloudWatch 지표를 사용한 AWS IoT SiteWise 모니터링
AWS IoT TwinMaker	AWS/IoTTwinMaker	Amazon CloudWatch 지표를 사용한 AWS IoT 모니터링
AWS IoT 1-Click		Amazon CloudWatch를 사용한 AWS IoT 원클릭 모니터링
AWS Key Management Service	AWS/KMS	CloudWatch를 사용한 모니터링
Amazon Keyspaces (Apache Cassandra용)	AWS/Cassandra	Amazon Keyspaces 지표 및 측정기준
Amazon Kendra		Amazon CloudWatch를 사용한 Amazon Kendra 모니터링
Amazon Managed Service for Apache Flink	AWS/KinesisAnalytics	SQL 애플리케이션에 대한 Managed Service for Apache Flink: CloudWatch를 사용한 모니터링 Apache Flink에 대한 Managed Service for Apache Flink: Amazon Managed Service for Apache Flink 지표 및 측정기준 보기

Service	네임스페이스	설명서
Amazon Data Firehose	`AWS/Firehose`	CloudWatch 지표를 사용하여 Firehose 모니터링
Amazon Kinesis Data Streams	`AWS/Kinesis`	Amazon CloudWatch를 사용한 Amazon Kinesis Data Streams 모니터링
Amazon Kinesis Video Streams	`AWS/KinesisVideo`	CloudWatch를 사용한 Kinesis Video Streams 지표 모니터링
AWS Lambda	`AWS/Lambda`	AWS Lambda 지표
Amazon Lex	`AWS/Lex`	Amazon CloudWatch를 사용한 Amazon Lex 모니터링

AWS License Manager	`AWSLicenseManager/licenseUsage`
	
	`AWS/LicenseManager/LinuxSubscriptions`
		Amazon CloudWatch를 사용하여 라이선스 사용 모니터링
		
		Linux 구독에 대한 사용량 지표 및 Amazon CloudWatch 경보

Amazon Location Service	`AWS/Location`	Amazon CloudWatch에 내보낸 Amazon Location Service 지표
Amazon Lookout for Equipment	`AWS/lookoutequipment`	Amazon CloudWatch를 사용한 Lookout for Equipment 모니터링
Amazon Lookout for Metrics	`AWS/LookoutMetrics`	Amazon CloudWatch를 사용한 Lookout for Metrics 모니터링
Amazon Lookout for Vision	`AWS/LookoutVision`	Amazon CloudWatch를 사용한 Lookout for Vision 모니터링

Service	네임스페이스	설명서
AWS 메인프레임 현대화		Amazon CloudWatch를 사용한 AWS 메인프레임 현대화 모니터링
Amazon Machine Learning	AWS/ML	CloudWatch 지표를 사용한 Amazon ML 모니터링
Amazon Managed Blockchain	AWS/managedblockchain	Amazon Managed Blockchain에서 Hyperledger Fabric 피어 노드 지표 사용
Amazon Managed Service for Prometheus	AWS/Prometheus	Amazon CloudWatch 지표
Amazon Managed Streaming for Apache Kafka	AWS/Kafka	Amazon CloudWatch를 사용한 Amazon MSK 모니터링
Amazon Managed Streaming for Apache Kafka	AWS/KafkaConnect	MSK Connect 모니터링
Amazon Managed Workflows for Apache Airflow	AWS/MWAA	Amazon MWAA의 컨테이너, 대기열 및 데이터베이스 지표
Amazon MemoryDB for Redis	AWS/MemoryDB	CloudWatch 지표 모니터링

Service	네임스페이스	설명서
Amazon MQ	AWS/AmazonMQ	Amazon CloudWatch를 사용하여 Amazon MQ 브로커 모니터링
Amazon Neptune	AWS/Neptune	CloudWatch를 사용한 Neptune 모니터링
AWS Network Firewall	AWS/NetworkFirewall	Amazon CloudWatch의 AWS Network Firewall 지표
AWS Network Manager	AWS/NetworkManager	온프레미스 리소스에 대한 CloudWatch 지표
Amazon Nimble Studio	AWS/NimbleStudio	Amazon CloudWatch CloudWatch를 사용한 Nimble Studio 모니터링
AWS HealthOmics	AWS/Omics	Amazon CloudWatch를 사용한 AWS HealthOmics 모니터링
AWS OpsWorks	AWS/OpsWorks	Amazon CloudWatch를 사용하여 스택 모니터링
AWS Outposts	AWS/Outposts	AWS Outposts에 대한 CloudWatch 지표
AWS Panorama	AWS/PanoramaDeviceMetrics	Amazon CloudWatch를 사용한 어플라이언스 및 애플리케이션 모니터링
Amazon Personalize	AWS/Personalize	Amazon Personalize에 대한 CloudWatch 지표
Amazon Pinpoint	AWS/Pinpoint	CloudWatch에서 Amazon Pinpoint 지표 보기
Amazon Polly	AWS/Polly	CloudWatch와 Amazon Polly 통합

Service	네임스페이스	설명서
AWS PrivateLink	AWS/PrivateLinkEndpoints	AWS PrivateLink의 CloudWatch 지표
AWS PrivateLink	AWS/PrivateLinkServices	AWS PrivateLink의 CloudWatch 지표
AWS Private 5G	AWS/Private5G	Amazon CloudWatch 지표
Amazon QLDB	AWS/QLDB	Amazon QuickSight에서 데이터 모니터링
Amazon QuickSight	AWS/QuickSight	Amazon CloudWatch를 사용한 모니터링
Amazon Redshift	AWS/Redshift	Amazon Redshift 성능 데이터
Amazon Relational Database Service	AWS/RDS	Amazon CloudWatch로 Amazon RDS 지표 모니터링
Amazon Rekognition	AWS/Rekognition	Amazon CloudWatch를 사용한 Rekognition 모니터링
AWS re:Post Private	AWS/rePostPrivate	Amazon CloudWatch로 AWS re:Post Private 모니터링
AWS RoboMaker	AWS/Robomaker	Amazon CloudWatch를 사용한 AWS RoboMaker 모니터링
Amazon Route 53	AWS/Route53	Amazon Route 53 모니터링

Service	네임스페이스	설명서
Route 53 애플리케이션 복구 컨트롤러	AWS/Route53RecoveryReadiness	애플리케이션 복구 컨트롤러와 함께 Amazon CloudWatch 사용하기
Amazon SageMaker	AWS/SageMaker	CloudWatch를 사용한 SageMaker 모니터링
Amazon SageMaker 모델 구축 파이프라인	AWS/SageMaker/ModelBuildingPipeline	SageMaker 파이프라인 지표
AWS Secrets Manager	AWS/SecretsManager	Amazon CloudWatch를 사용한 Secrets Manager 모니터링
Amazon Security Lake	AWS/SecurityLake	Amazon Security Lake에 대한 CloudWatch 지표
서비스 카탈로그	AWS/ServiceCatalog	Service Catalog CloudWatch 지표
AWS Shield Advanced	AWS/DDoSProtection	CloudWatch를 사용한 모니터링
Amazon Simple Email Service	AWS/SES	CloudWatch에서 Amazon SES 이벤트 데이터 가져오기
AWS SimSpace Weaver	AWS/simspaceweaver	Amazon CloudWatch를 사용한 AWS SimSpace Weaver 모니터링
Amazon Simple Notification Service	AWS/SNS	CloudWatch를 사용한 Amazon SNS 모니터링
Amazon Simple Queue Service	AWS/SQS	CloudWatch를 사용한 Amazon SQS 대기열 모니터링

Service	네임스페이스	설명서
Amazon S3	AWS/S3	Amazon CloudWatch를 사용한 지표 모니터링
S3 Storage Lens	AWS/S3/Storage-Lens	CloudWatch에서 S3 Storage Lens 지표 모니터링
Amazon Simple Workflow Service	AWS/SWF	CloudWatch에 대한 Amazon SWF 지표
AWS Step Functions	AWS/States	CloudWatch를 사용한 Step Functions 모니터링
AWS Storage Gateway	AWS/StorageGateway	Amazon CloudWatch 지표 사용
AWS Systems Manager 명령 실행	AWS/SSM-RunCommand	CloudWatch를 사용하여 명령 실행 지표 모니터링
Amazon Textract	AWS/Textract	Amazon Textract의 CloudWatch 지표
Amazon Timestream	AWS/Timestream	Timestream 지표 및 측정기준
AWS Transfer for SFTP	AWS/Transfer	AWS SFTP CloudWatch 지표
Amazon Transcribe	AWS/Transcribe	Amazon CloudWatch를 사용한 Amazon Transcribe 모니터링
Amazon Translate	AWS/Translate	Amazon Translate에 대한 CloudWatch 지표 및 측정기준
AWS Trusted Advisor	AWS/TrustedAdvisor	CloudWatch를 사용한 Trusted Advisor 경보 생성

Service	네임스페이스	설명서
Amazon VPC	AWS/NATGateway	CloudWatch를 사용한 NAT 게이트웨이 모니터링
Amazon VPC	AWS/TransitGateway	Transit Gateway의 CloudWatch 지표
Amazon VPC	AWS/VPN	CloudWatch를 사용한 모니터링
Amazon VPC IP 주소 관리자	AWS/IPAM	Amazon CloudWatch로 경보 생성
AWS WAF	AWS/WAFV2 (AWS WAF 리소스의 경우) WAF(AWS WAF 클래식 리소스의 경우)	CloudWatch를 사용한 모니터링
Amazon WorkMail	AWS/WorkMail	Amazon CloudWatch를 사용한 Amazon WorkMail 모니터링
Amazon WorkSpaces	AWS/WorkSpaces	CloudWatch 지표를 사용하여 WorkSpaces 모니터링
Amazon WorkSpaces Web	AWS/WorkSpacesWeb	Amazon CloudWatch를 사용한 Amazon WorkSpaces Web 모니터링

AWS 사용량 지표

CloudWatch는 일부 AWS 리소스 및 API의 사용량을 추적하는 지표를 수집합니다. 이러한 지표는 AWS/Usage 네임스페이스에 게시됩니다. CloudWatch의 사용량 지표를 사용하면 CloudWatch 콘솔에서 지표를 시각화하고 사용자 지정 대시보드를 생성하며 CloudWatch 이상 탐지를 통해 활동의 변화를 감지하고 사용량이 임계값에 근접할 때 이를 알려 주는 경보를 구성함으로써 사용량을 사전에 관리할 수 있습니다.

일부 AWS 서비스는 이러한 사용량 지표를 Service Quotas와 통합합니다. 이러한 서비스의 경우 CloudWatch를 사용하여 계정의 서비스 할당량 사용을 관리할 수 있습니다. 자세한 내용은 서비스 할당량 시각화 및 경보 설정 섹션을 참조하세요.

주제

서비스 할당량 시각화 및 경보 설정

일부 AWS 서비스의 경우 사용량 지표를 사용하여 CloudWatch 그래프 및 대시보드에서 현재 서비스 사용량을 시각화할 수 있습니다. CloudWatch 지표 수학 함수를 사용하여 해당 리소스에 대한 서비스 할당량을 그래프에 표시할 수 있습니다. 사용량이 서비스 할당량에 가까워지면 경고하는 경보를 구성할 수도 있습니다. 서비스 할당량에 대한 자세한 내용은 Service Quotas 사용 설명서의 Service Quotas란 무엇인가요? 단원을 참조하세요.

CloudWatch 교차 계정 관찰성에서 모니터링 계정으로 설정된 계정에 로그인한 경우 해당 모니터링 계정을 사용하여 서비스 할당량을 시각화하고 해당 모니터링 계정에 연결된 소스 계정의 지표에 대한 경보를 설정할 수 있습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 섹션을 참조하세요.

현재 다음 서비스는 해당 사용량 지표를 Service Quotas와 통합합니다.

- AWS CloudHSM
- Amazon Chime SDK
- Amazon CloudWatch
- Amazon CloudWatch Logs
- Amazon DynamoDB

- Amazon EC2
- Amazon Elastic 컨테이너 레지스트리
- Elastic Load Balancing
- AWS Fargate
- AWS Fault Injection Service
- AWS Interactive Video Service
- AWS Key Management Service
- Amazon Data Firehose
- Amazon Location Service
- Amazon Managed Blockchain 쿼리
- AWS RoboMaker
- Amazon SageMaker

서비스 할당량을 시각화하고 선택적으로 경보를 설정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. All metrics(모든 지표) 탭에서 Usage(사용량)를 선택한 다음 AWS By Resource(리소스별)를 선택합니다.

   서비스 할당량 사용량 지표 목록이 나타납니다.
4. 지표 중 하나 옆에 있는 확인란을 선택합니다.

   그래프는 해당 AWS 리소스의 현재 사용량을 표시합니다.
5. 그래프에 서비스 할당량을 추가하려면 다음을 수행합니다.

   a. 그래프로 표시된 지표(Graphed metrics) 탭을 선택합니다.

   b. 수학 표현식(Math expression), 빈 표현식으로 시작(Start with an empty expression)을 선택합니다. 새 행의 세부 정보(Details)에 **`SERVICE_QUOTA(m1)`**를 입력합니다.

      그래프에 새 줄이 추가되어 지표에 표시된 리소스에 대한 서비스 할당량을 표시합니다.
6. 현재 사용량을 할당량의 백분율로 보려면 새 표현식을 추가하거나 현재 SERVICE_QUOTA 표현식을 변경합니다. 사용할 새 표현식은 **`m1/SERVICE_QUOTA(m1)*100`**입니다.
7. (선택 사항) 서비스 할당량에 접근하는 경우 알려주는 경보를 설정하려면 다음을 수행합니다.

a. **m1/SERVICE_QUOTA(m1)*100** 행의 작업(Actions)에서 경보 아이콘을 선택합니다. 이 아이콘은 종처럼 보입니다.

   경보 생성 페이지가 나타납니다.

b. 조건(Conditions)에서 임계값 유형(Threshold type)이 정적(Static)이고 표현식1이 해당할 때마다(Whenever Expression1 is)가 큼(Greater)으로 설정되었는지 확인합니다. 보다(than)에 **80**을 입력합니다. 이렇게 하면 사용량이 할당량의 80%를 초과할 경우 ALARM 상태가 되는 경보가 생성됩니다.

c. 다음(Next)을 선택합니다.

d. 다음 페이지에서 Amazon SNS 주제를 선택하거나 새 주제를 생성한 후 다음(Next)을 선택합니다. 경보가 ALARM 상태가 되면 선택한 주제에 알림이 전송됩니다.

e. 다음 페이지에서 경보의 이름과 설명을 입력하고 다음(Next)을 선택합니다.

f. 경보 생성(Create alarm)을 선택합니다.

AWS API 사용량 지표

AWS CloudTrail 로깅을 지원하는 대부분의 API는 CloudWatch에 사용량 지표도 보고합니다. CloudWatch의 API 사용량 지표를 사용하면 CloudWatch 콘솔에서 지표를 시각화하고 사용자 지정 대시보드를 생성하며 CloudWatch 이상 탐지를 통해 활동의 변화를 감지하고 사용량이 임계값에 근접할 때 이를 알려 주는 경보를 구성함으로써 API 사용량을 사전에 관리할 수 있습니다.

다음 표에는 API 사용량 지표를 CloudWatch에 보고하는 서비스 및 해당 서비스의 사용량 지표를 확인하는 데 `Service` 측정기준에 사용할 값이 나와 있습니다.

Service	**`Service`** 측정기준의 값
AWS Identity and Access Management Access Analyzer	`Access Analyzer`
AWS Account Management	`Account Management`
Alexa for Business	`A4B`
Amazon API Gateway	`API Gateway`
AWS App Mesh	`App Mesh`

Service	**Service** 측정기준의 값
AWS AppConfig	`AWS AppConfig`
Amazon AppFlow	`AppFlow`
Application Auto Scaling	`Application Auto Scaling`
Application Discovery Service	`Application Discovery Service`
Amazon 앱Stream	`AppStream`
AppStream 2.0 Image Builder	`Image Builder`
Amazon Athena	`Athena`
AWS Audit Manager	`Audit Manager`
AWS Backup	`Backup`
AWS Batch	`Batch`
Amazon Braket	`Braket`
AWS 예산	`Budgets`
AWS Certificate Manager	`Certificate Manager`
Amazon Chime SDK	`ChimeSDK`
Amazon Cloud Directory	`Cloud Directory`
AWS Cloud Map	`Cloud Map`
AWS CloudFormation	`CloudFormation`
AWS CloudHSM	`CloudHSM`
Amazon CloudSearch	`CloudSearch`
AWS CloudShell	`CloudShell`

Service	**Service** 측정기준의 값
AWS CloudTrail	`CloudTrail`
Amazon CloudWatch	`CloudWatch`
Amazon CloudWatch Logs	`Logs`
Amazon CloudWatch Application Insights	`CloudWatch Application Insights`
AWS CodeBuild	`CodeBuild`
AWS CodeCommit	`CodeCommit`
Amazon CodeGuru Profiler	`CodeGuru Profiler`
AWS CodePipeline	`CodePipeline`
AWS CodeStar	`CodeStar`
AWS CodeStar 알림	`CodeStar Notifications`
AWS CodeStar 연결	`CodeStar Connections`
Amazon Cognito 자격 증명 풀	`Cognito Identity Pools`
Amazon Cognito Sync	`Cognito Sync`
Amazon Comprehend	`Comprehend`
Amazon Comprehend Medical	`Comprehend Medical`
AWS Compute Optimizer	`ComputeOptimzier`
Amazon Connect	`Connect`
Amazon Connect Customer Profiles	`Customer Profiles`
AWS 비용 및 사용 보고서	`Cost and Usage Report`
AWS Cost Explorer	`Cost Explorer`

Service	**Service** 측정기준의 값
AWS Data Exchange	`Data Exchange`
AWS Data Lifecycle Manager	`Data Lifecycle Manager`
AWS Database Migration Service	`Database Migration Service`
AWS DataSync	`DataSync`
AWS DeepLens	`AWS DeepLens`
Amazon Detective	`Detective`
Device Advisor	`Device Advisor`
AWS Direct Connect	`Direct Connect`
AWS Directory Service	`Directory Service`
DynamoDB Accelerator	`DynamoDBAccelerator`
Amazon EC2	`EC2`
EC2 Auto Scaling	`EC2 Auto Scaling`
Amazon Elastic 컨테이너 레지스트리	`ECR Public`
Amazon Elastic Container Service	`ECS`
Amazon Elastic File System	`EFS`
Amazon Elastic Kubernetes 서비스	`EKS`
AWS Elastic Beanstalk	`Elastic Beanstalk`
Amazon Elastic Inference	`Elastic Inference`
Elastic Load Balancing	`Elastic Load Balancing`
Amazon EMR	`EMR Containers`

Service	**Service** 측정기준의 값
AWS Firewall Manager	`Firewall Manager`
Amazon FSx	`FSx`
Amazon GameLift	`GameLift`
AWS Glue DataBrew	`DataBrew`
Amazon Managed Grafana	`Grafana`
AWS IoT Greengrass	`Greengrass`
AWS Ground Station	`Ground Station`
AWS Health API 및 알림	`AWS Health APIs And Notifications`
Amazon Interactive Video Service	`IVS`
AWS IoT Core	`IoT`
AWS IoT 1-Click	`IoT 1-Click`
AWS IoT Events	`IoT Events`
AWS IoT RoboRunner	`IoT RoboRunner`
AWS IoT SiteWise	`IoT Sitewise`
AWS IoT Wireless	`IoT Wireless`
Amazon Kendra	`Kendra`
Amazon Keyspaces(Apache Cassandra용)	`Keyspaces`
Amazon Managed Service for Apache Flink	`Kinesis Analytics`
Amazon Data Firehose	`Firehose`
Kinesis Video Streams	`Kinesis Video Streams`

Service	**Service** 측정기준의 값
AWS Key Management Service	KMS
AWS Lambda	Lambda
AWS Launch Wizard	Launch Wizard
Amazon Lex	Amazon Lex
Amazon Lightsail	Lightsail
Amazon Location Service	Location
Amazon Lookout for Vision	Lookout for Vision
Amazon Machine Learning	Amazon Machine Learning
Amazon Macie	Macie
Amazon Managed Blockchain(AMB) 쿼리	Amazon Managed Blockchain Query
AWS Managed Services	AWS Managed Services
AWS Marketplace Commerce Analytics	Marketplace Analytics Service
AWS Elemental MediaConnect	MediaConnect
AWS Elemental MediaConvert	MediaConvert
AWS Elemental MediaLive	MediaLive
AWS Elemental MediaStore	Mediastore
AWS Elemental MediaTailor	MediaTailor
AWS Mobile Hub	Mobile Hub
AWS Network Firewall	Network Firewall
AWS OpsWorks	OpsWorks

Service	**Service** 측정기준의 값
AWS OpsWorks for Configuration Management	`OPsWorks CM`
AWS Outposts	`Outposts`
AWS Organizations	`Organizations`
Amazon RDS 성능 개선 도우미	`Performance Insights`
Amazon Pinpoint	`Pinpoint`
AWS Private Certificate Authority	`Private Certificate Authority`
Amazon Managed Service for Prometheus	`Prometheus`
AWS Proton	`Proton`
Amazon Quantum Ledger Database(QLDB)	`QLDB`
Amazon RDS	`RDS`
Amazon Redshift	`Redshift Data API`
Amazon Rekognition	`Rekognition`
AWS Resource Access Manager	`Resource Access Manager`
AWS Resource Groups	`Resource Groups`
AWS Resource Groups Tagging API	`Resource Groups Tagging API`
AWS RoboMaker	`RoboMaker`
Amazon Route 53 도메인	`Route 53 Domains`
Amazon Route 53 Resolver	`Route 53 Resolver`
Amazon S3	`S3`
Amazon S3 Glacier	`Amazon S3 Glacier`

Service	**Service** 측정기준의 값
Amazon SageMaker Runtime	`Sagemaker`
절감형 플랜	`Savings Plans`
AWS Secrets Manager	`Secrets Manager`
AWS Security Hub	`Security Hub`
AWS Server Migration Service	`AWS Server Migration Service`
AWS Service Catalog AppRegistry	`Service Catalog AppRegistry`
Service Quotas	`Service Quotas`
AWS Shield	`Shield`
AWS Signer	`Signer`
Amazon Simple Notification Service	`SNS`
Amazon Simple Email Service	`SES`
Amazon Simple Queue Service	`SQS`
아이덴티티 스토어	`Identity Store`
Storage Gateway	`Storage Gateway`
AWS Support	`Support`
Amazon Simple Workflow Service	`SWF`
Amazon Textract	`Textract`
AWS IoT Things Graph	`ThingsGraph`
Amazon Timestream	`Timestream`
Amazon Transcribe	`Transcribe`

Service	**Service** 측정기준의 값
Amazon Translate	`Translate`
Amazon Transcribe 스트리밍 트랜스크립션	`Transcribe Streaming`
AWS Transfer Family	`Transfer`
AWS WAF	`WAF`
Amazon WorkDocs	`Amazon WorkDocs`
Amazon WorkLink	`WorkLink`
Amazon WorkMail	`Amazon WorkMail`
Amazon WorkSpaces	`Workspaces`
AWS X-Ray	`X-Ray`

일부 서비스는 추가 API 사용량 지표도 보고합니다. API가 CloudWatch에 사용량 지표를 보고하는지 여부를 알아보려면 CloudWatch 콘솔을 사용하여 `AWS/Usage` 네임스페이스에서 해당 서비스가 보고한 지표를 확인합니다.

CloudWatch에 사용량 지표를 보고하는 서비스의 API 목록을 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. All metrics(모든 지표) 탭에서 Usage(사용량)를 선택한 다음 AWS By Resource(리소스별)를 선택합니다.
4. 지표 목록 근처의 검색 상자에 서비스 이름을 입력합니다. 입력한 서비스를 기준으로 지표가 필터링됩니다.

CloudWatch 사용량 지표

CloudWatch는 일부 AWS 리소스의 사용량을 추적하는 지표를 수집합니다. 이러한 지표는 AWS 서비스 할당량에 해당합니다. 이러한 지표를 추적하면 할당량을 사전 예방적으로 관리하는 데 도움이 될 수 있습니다. 자세한 내용은 서비스 할당량 시각화 및 경보 설정 섹션을 참조하세요.

서비스 할당량 사용량 지표는 AWS/Usage 네임스페이스에 있으며 1분마다 수집됩니다.

현재 CloudWatch가 게시하는 이 네임스페이스의 유일한 지표 이름은 CallCount입니다. 이 지표는 Resource, Service 및 Type 측정기준으로 게시됩니다. Resource 측정기준은 추적 중인 API 작업의 이름을 지정합니다. 예를 들어 "Service": "CloudWatch", "Type": "API" 및 "Resource": "PutMetricData" 측정기준과 함께 게시된 CallCount 지표는 계정에서 CloudWatch PutMetricData API 작업이 호출된 횟수를 나타냅니다.

CallCount 지표에는 지정된 단위가 없습니다. 지표에 가장 유용한 통계는 SUM이며 1분 동안의 총 작업 수를 나타냅니다.

지표

지표	설명
CallCount	계정에서 수행된 지정된 작업 수

측정기준

측정기준	설명
Service	리소스가 포함된 AWS 서비스의 이름 CloudWatch 사용량 지표의 경우 이 측정기준의 값은 CloudWatch 입니다.
Class	추적 중인 리소스의 클래스. CloudWatch API 사용량 지표는 이 측정기준을 None 값과 함께 사용합니다.
Type	추적 중인 리소스의 유형. 현재, Service 치수가 CloudWatch 인 경우 Type에 대한 유일한 유효한 값은 API입니다.
Resource	API 작업의 이름. 유효한 값은 다음과 같습니다. DeleteAlarms , DeleteDashboards , DescribeAlarmHistory , DescribeAlarms , GetDashboard , GetMetricData , GetMetricStatistics , ListMetrics , PutDashboard 및 PutMetricData

CloudWatch 자습서

다음 시나리오에서는 Amazon CloudWatch 사용을 보여 줍니다. 첫 번째 시나리오에서는 CloudWatch 콘솔을 사용하여 AWS 사용량을 추적하고 특정 지출 임계값을 초과했을 때 이를 알려 주는 결제 경보를 생성합니다. 좀 더 발전한 두 번째 시나리오에서는 AWS Command Line Interface(AWS CLI)를 사용하여 GetStarted라는 가상의 애플리케이션을 위한 단일 지표를 게시합니다.

시나리오

- 예상 요금 모니터링
- 지표 게시

시나리오: CloudWatch를 사용하여 예상 요금 모니터링

이 시나리오에서는 예상 요금을 모니터링하기 위한 Amazon CloudWatch 경보를 생성합니다. AWS 계정에 대한 예상 요금 모니터링을 사용 설정하면 예상 요금이 계산되어 지표 데이터로서 매일 여러 번 CloudWatch에 전송됩니다.

결제 지표 데이터는 미국 동부(버지니아 북부) 리전에 저장되며 전 세계 요금을 반영합니다. 이 데이터에는 사용한 AWS의 모든 서비스에 대한 예상 요금과 전반적인 총 AWS 예상 요금이 들어 있습니다.

요금이 특정 임계값을 초과한 경우 이메일로 알림을 받도록 선택할 수 있습니다. 이러한 알림은 CloudWatch에 의해 트리거되며, 메시지는 Amazon Simple Notification Service(Amazon SNS)를 사용하여 전송됩니다.

> **Note**
>
> 이미 청구된 CloudWatch 요금을 분석하는 방법은 CloudWatch 결제 및 비용 섹션을 참조하세요.

작업

- 1단계: 결제 알림 사용
- 2단계: 결제 알림 생성
- 3단계: 알림 상태 확인

1단계: 결제 알림 사용

예상 요금에 대한 경보를 생성할 수 있으려면 먼저 결제 알림을 활성화해야 합니다. 그래야만 예상되는 AWS 요금을 모니터링하고 결제 지표 데이터를 사용하여 경보를 생성할 수 있습니다. 결제 알림을 활성화한 후에는 데이터 수집을 비활성화할 수 없지만, 생성된 결제 알림은 무엇이든 삭제할 수 있습니다.

결제 알림을 처음 활성화하고 나서 결제 데이터를 확인하고 결제 경보를 설정할 수 있기까지 약 15분 정도의 시간이 걸립니다.

요구 사항

- 루트 사용자 보안 인증 정보를 사용하여 로그인하거나 청구 정보를 볼 수 있는 권한이 부여된 사용자로 로그인해야 합니다.
- 통합 결제 계정의 경우 결제 계정으로 로그인하면 연결된 각 계정에 대한 결제 데이터를 찾을 수 있습니다. 통합 계정에 대해서뿐만 아니라 연결된 각 계정에 대한 서비스별 총 예상 요금 및 예상 요금에 대한 결제 데이터를 볼 수 있습니다.
- 통합 결제 계정에서 멤버에 연결된 계정 지표는 지급인 계정이 결제 알림 받기 기본 설정을 사용하도록 설정한 경우에만 캡처됩니다. 관리/지급인 계정인 계정을 변경하는 경우 새 관리/지급인 계정에서 결제 알림을 사용해야 합니다.
- APN 계정에 대한 결제 지표는 CloudWatch에 게시되지 않으므로 계정이 Amazon 파트너 네트워크(APN)에 속하지 않아야 합니다. 자세한 내용은 AWS 파트너 네트워크를 참조하십시오.

예상 요금 모니터링을 활성화하려면

1. https://console.aws.amazon.com/billing/에서 AWS Billing 콘솔을 엽니다.
2. 탐색 창에서 결제 기본 설정(Billing preferences)을 선택합니다.
3. 알림 환경 설정에서 편집을 선택합니다.
4. CloudWatch 결제 알림 수신을 선택합니다.
5. 기본 설정 저장을 선택합니다.

2단계: 결제 알림 생성

> **Important**
>
> 결제 경보를 생성하기 전에 Region(리전)을 US East (N. Virginia)(미국 동부(버지니아 북부))로 설정해야 합니다. 결제 지표 데이터는 이 리전에 저장되어 전 세계 요금을 나타냅니다. 또한 계정 또는 관리/지급인 계정(통합 결제를 사용하는 경우)에 대한 결제 알림을 활성화해야 합니다. 자세한 내용을 알아보려면 1단계: 결제 알림 활성화를 참조하세요.

이 절차에서는 AWS에 대한 예상 요금이 정의된 임계값을 초과할 때 알림을 보내는 경보를 생성합니다.

CloudWatch 콘솔을 사용하여 결제 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 경보(Alarms)를 선택한 다음 모든 경보(All alarms)를 선택합니다.
3. 경보 생성(Create alarm)을 선택하세요.
4. 지표 선택(Select metric)을 선택하세요. Browse(찾아보기)에서 Billing(결제)을 선택한 다음 Total Estimated Charge(예상 요금 합계)를 선택합니다.

   > **Note**
   >
   > 결제/예상 요금 합계 지표가 표시되지 않으면 결제 알림을 활성화하고 리전을 미국 동부(버지니아 북부)로 변경합니다. 자세한 내용은 결제 알림 사용 설정 섹션을 참조하세요.

5. EstimatedCharges 지표 상자를 선택한 다음 Select metric(지표 선택)을 선택합니다.
6. 통계에서 최대를 선택합니다.
7. Period(기간)에서 6 hours(6시간)를 선택합니다.
8. 임계값 유형(Threshold type)에서 정적(Static)을 선택합니다.
9. Whenever EstimatedCharges is . . .(EstimatedCharges가 다음인 경우 항상…)에서 Greater(보다 큼)를 선택합니다.
10. . . . 보다에 대해 경보를 트리거하려는 값을 정의합니다. 예를 들어 USD **200**달러입니다.

    EstimatedCharges 지표 값은 미국 달러(USD)로만 표시되며, 통화 변환은 Amazon Services LLC에서 제공합니다. 자세한 내용은 AWS Billing란 무엇인가요?를 참조하세요.

> **Note**
>
> 임계값을 정의하면 미리 보기 그래프에 이번 달의 예상 요금이 표시됩니다.

11. 추가 구성을 선택하고 다음을 수행합니다.
    - Datapoints to alarm(경보를 보낼 데이터 포인트)에서 1 out of 1(1/1)을 지정합니다.
    - Missing data treatment(누락된 데이터 처리)에서 Treat missing data as missing(누락된 데이터를 누락으로 처리)을 선택합니다.
12. 다음(Next)을 선택합니다.
13. 알림에서 경보 내가 선택되어 있는지 확인하세요. 그런 다음 경보가 ALARM 상태일 때 알림을 받을 Amazon SNS 주제를 지정합니다. Amazon SNS 주제에 이메일 주소를 포함하면 청구 금액이 지정한 임계값을 초과할 때 이메일을 받을 수 있습니다.

    기존 Amazon SNS 주제를 선택하거나, 새 Amazon SNS 주제를 생성하거나, 주제 ARN을 사용하여 다른 계정에 알릴 수 있습니다. 경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 전송하도록 하려면 Add notification(알림 추가)을 선택합니다.
14. 다음(Next)을 선택합니다.
15. Name and description(이름 및 설명)에 경보 이름을 입력합니다.
    - (선택 사항) 경보에 대한 설명을 입력합니다.
16. 다음(Next)을 선택합니다.
17. Preview and create(미리 보기 및 생성)에서 구성이 올바른지 확인한 다음 Create alarm(경보 생성)을 선택합니다.

3단계: 알림 상태 확인

이제, 방금 만든 결제 경보의 상태를 확인합니다.

경보 상태를 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 필요한 경우 리전을 미국 동부(버지니아 북부)로 변경합니다. 결제 지표 데이터는 이 리전에 저장되며 전 세계 요금을 반영합니다.
3. 탐색 창에서 경보(Alarms)를 선택합니다.

4. 경보 옆의 확인란을 선택합니다. 구독이 확인되기 전까지 "확인 보류 중"으로 표시가 됩니다. 구독 확인 후에 콘솔을 새로 고쳐 업데이트된 상태를 보여줍니다.

4단계: 결제 알림 편집

예를 들어 매월 AWS에 사용하는 금액을 200 USD에서 400 USD로 늘리고 싶은 경우 기존 결제 경보를 편집하여 경보 트리거 전에 초과해야 하는 금액을 늘릴 수 있습니다.

결제 경보를 편집하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 필요한 경우 리전을 미국 동부(버지니아 북부)로 변경합니다. 결제 지표 데이터는 이 리전에 저장되며 전 세계 요금을 반영합니다.
3. 탐색 창에서 경보(Alarms)를 선택합니다.
4. 경보 옆의 확인란을 선택한 후 작업과 수정을 차례로 선택합니다.
5. Whenever my total AWS charges for the month exceed(해당 월 총 요금이 다음을 초과할 때마다)의 경우 알림이 트리거되고 이메일 알림이 전송되려면 초과되어야 하는 새 금액을 지정합니다.
6. 변경 사항 저장을 선택합니다.

5단계: 결제 알림 삭제

결제 경보가 더 이상 필요 없는 경우 해당 경보를 삭제할 수 있습니다.

결제 경보를 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 필요한 경우 리전을 미국 동부(버지니아 북부)로 변경합니다. 결제 지표 데이터는 이 리전에 저장되며 전 세계 요금을 반영합니다.
3. 탐색 창에서 경보(Alarms)를 선택합니다.
4. 경보 옆의 확인란을 선택하고 작업, 삭제를 차례로 선택합니다.
5. 확인 메시지가 나타나면 예, 삭제합니다(Yes, Delete)를 선택합니다.

시나리오: CloudWatch에 지표 게시

이 시나리오에서는 AWS Command Line Interface(AWS CLI)를 사용하여 GetStarted라는 가상의 애플리케이션을 위한 단일 지표를 게시합니다. AWS CLI를 아직 설치 및 구성하지 않은 경우 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface 설정 단원을 참조하세요.

작업

- 1단계: 데이터 구성 정의
- 2단계: CloudWatch에 지표 추가
- 3단계: CloudWatch에서 통계 가져오기
- 4단계: 콘솔을 사용하여 그래프 보기

1단계: 데이터 구성 정의

이 시나리오에서는 애플리케이션의 요청 지연 시간을 추적하는 데이터 요소를 게시합니다. 적절한 지표 및 네임스페이스의 이름을 선택합니다. 예를 들어 지표의 이름을 RequestLatency로 지정하고 모든 데이터 요소를 GetStarted 네임스페이스에 배치합니다.

3시간의 지연 시간 데이터를 총체적으로 나타내는 여러 데이터 요소를 게시합니다. 원시 데이터는 3시간에 걸쳐 분산된 요청 지연 시간 판독값 15개로 구성됩니다. 각 판독값은 다음과 같이 밀리초 단위입니다.

- 시간 1: 87, 51, 125, 235
- 시간 2: 121, 113, 189, 65, 89
- 시간 3: 100, 47, 133, 98, 100, 328

데이터를 단일 데이터 요소 또는 '통계 집합'이라고 하는 집계된 데이터 요소 집합으로 CloudWatch에 게시할 수 있습니다. 지표를 1분 정도로 낮게 세분화하여 집계할 수 있습니다. 집계된 데이터 요소를 미리 정의된 4개의 키(`Sum`, `Minimum`, `Maximum`, `SampleCount`)가 있는 통계 집합으로 CloudWatch에 게시할 수 있습니다.

시간 1의 데이터 요소를 단일 데이터 요소로 게시합니다. 시간 2 및 시간 3의 데이터의 경우 데이터 요소를 집계하여 각 시간에 대한 통계 세트를 게시합니다. 키 값은 다음 표에 표시됩니다.

시간	원시 데이터	합계	최소	최대	SampleCount
1	87				
1	51				
1	125				
1	235				
2	121, 113, 189, 65, 89	577	65	189	5
3	100, 47, 133, 98, 100, 328	806	47	328	6

2단계: CloudWatch에 지표 추가

데이터 구성을 정의하면 데이터 추가를 시작할 준비가 된 것입니다.

CloudWatch에 데이터 요소를 게시하려면

1. 명령 프롬프트에서 put-metric-data 명령을 실행하여 첫 번째 시간에 대한 데이터를 추가합니다. 예제 타임스탬프를 UTC 기준으로 2시간 전인 타임스탬프로 변경합니다.

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 87 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 51 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 125 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 235 --unit Milliseconds
```

2. 첫 번째 시간보다 1시간 늦은 타임스탬프를 사용하여 두 번째 시간에 대한 데이터를 추가합니다.

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T21:30:00Z --statistic-values
 Sum=577,Minimum=65,Maximum=189,SampleCount=5 --unit Milliseconds
```

3. 현재 시간에 대한 기본값에 대한 타임스탬프를 생략하고 세 번째 시간에 대한 데이터를 추가합니다.

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--statistic-values Sum=806,Minimum=47,Maximum=328,SampleCount=6 --unit Milliseconds
```

3단계: CloudWatch에서 통계 가져오기

CloudWatch에 지표를 게시했으므로 이제 다음과 같이 get-metric-statistics 명령을 사용하여 해당 지표를 기반으로 통계를 검색할 수 있습니다. 게시한 가장 빠른 타임스탬프를 포함하도록 지난 시간에서 `--start-time` 및 `--end-time`을 충분히 여유 있게 지정해야 합니다.

```
aws cloudwatch get-metric-statistics --namespace GetStarted --metric-name
 RequestLatency --statistics Average \
--start-time 2016-10-14T00:00:00Z --end-time 2016-10-15T00:00:00Z --period 60
```

다음은 예시 출력입니다.

```
{
 "Datapoints": [],
 "Label": "Request:Latency"
}
```

4단계: 콘솔을 사용하여 그래프 보기

CloudWatch에 지표를 게시했다면 CloudWatch 콘솔을 사용하여 통계 그래프를 볼 수 있습니다.

콘솔에서 통계 그래프를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표를 선택합니다.
3. 모든 지표 탭의 검색 상자에RequestLatency를 입력하고 Enter 키를 누릅니다.

4. RequestLatency 지표에 대한 확인란을 선택합니다. 지표 데이터의 그래프가 위쪽 창에 표시됩니다.

자세한 내용은 지표 그래프 작성 단원을 참조하세요.

AWS SDK에서 CloudWatch 사용

다양한 프로그래밍 언어에 대해 AWS 소프트웨어 개발 키트(SDK)을 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예시
AWS SDK for C++	AWS SDK for C++ 코드 예시
AWS CLI	AWS CLI 코드 예시
AWS SDK for Go	AWS SDK for Go 코드 예시
AWS SDK for Java	AWS SDK for Java 코드 예시
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예시
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예시
AWS SDK for .NET	AWS SDK for .NET 코드 예시
AWS SDK for PHP	AWS SDK for PHP 코드 예시
AWS Tools for PowerShell	Tools for PowerShell 코드 예시
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예시
AWS SDK for Ruby	AWS SDK for Ruby 코드 예시
AWS SDK for Rust	AWS SDK for Rust 코드 예시
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예시
AWS SDK for Swift	AWS SDK for Swift 코드 예시

CloudWatch와 관련된 예제는 AWS SDK를 사용한 CloudWatch 코드 예제 섹션을 참조하세요.

가용성 예제

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

AWS SDK를 사용한 CloudWatch 코드 예제

다음 코드 예제에서는 AWS 소프트웨어 개발 키트(SDK)에서 CloudWatch를 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

교차 서비스 예시는 여러 AWS 서비스 전반에서 작동하는 샘플 애플리케이션입니다.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 AWS SDK에서 CloudWatch 사용을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작하기

Hello CloudWatch

다음 코드 예제에서는 CloudWatch를 사용하여 시작하는 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace CloudWatchActions;
```

```
public static class HelloCloudWatch
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon CloudWatch service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatch>()
            ).Build();

        // Now the client is available for injection.
        var cloudWatchClient =
 host.Services.GetRequiredService<IAmazonCloudWatch>();

        // You can use await and any of the async methods to get a response.
        var metricNamespace = "AWS/Billing";
        var response = await cloudWatchClient.ListMetricsAsync(new
 ListMetricsRequest
        {
            Namespace = metricNamespace
        });
        Console.WriteLine($"Hello Amazon CloudWatch! Following are some metrics
 available in the {metricNamespace} namespace:");
        Console.WriteLine();
        foreach (var metric in response.Metrics.Take(5))
        {
            Console.WriteLine($"\tMetric: {metric.MetricName}");
            Console.WriteLine($"\tNamespace: {metric.Namespace}");
            Console.WriteLine($"\tDimensions: {string.Join(", ",
 metric.Dimensions.Select(m => $"{m.Name}:{m.Value}"))}");
            Console.WriteLine();
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListMetrics를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> System.out.println(" Retrieved metric is:
 " + metrics.metricName()));

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListMetrics를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <namespace>
        Where:
           namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListMetrics를 참조하십시오.

코드 예시

- AWS SDK를 사용한 CloudWatch 작업
  - AWS SDK 또는 CLI와 함께 DeleteAlarms 사용
  - AWS SDK 또는 CLI와 함께 DeleteAnomalyDetector 사용
  - AWS SDK 또는 CLI와 함께 DeleteDashboards 사용
  - AWS SDK 또는 CLI와 함께 DescribeAlarmHistory 사용
  - AWS SDK 또는 CLI와 함께 DescribeAlarms 사용
  - AWS SDK 또는 CLI와 함께 DescribeAlarmsForMetric 사용
  - AWS SDK 또는 CLI와 함께 DescribeAnomalyDetectors 사용
  - AWS SDK 또는 CLI와 함께 DisableAlarmActions 사용
  - AWS SDK 또는 CLI와 함께 EnableAlarmActions 사용
  - AWS SDK 또는 CLI와 함께 GetDashboard 사용
  - AWS SDK 또는 CLI와 함께 GetMetricData 사용
  - AWS SDK 또는 CLI와 함께 GetMetricStatistics 사용
  - AWS SDK 또는 CLI와 함께 GetMetricWidgetImage 사용
  - AWS SDK 또는 CLI와 함께 ListDashboards 사용
  - AWS SDK 또는 CLI와 함께 ListMetrics 사용
  - AWS SDK 또는 CLI와 함께 PutAnomalyDetector 사용
  - AWS SDK 또는 CLI와 함께 PutDashboard 사용
  - AWS SDK 또는 CLI와 함께 PutMetricAlarm 사용
  - AWS SDK 또는 CLI와 함께 PutMetricData 사용
- AWS SDK를 사용한 CloudWatch 시나리오
  - AWS SDK를 사용하여 CloudWatch 경보 시작하기
  - AWS SDK를 사용하여 CloudWatch 지표, 대시보드 및 경보 시작하기
  - AWS SDK를 사용하여 CloudWatch 지표 및 경보 관리
- AWS SDK를 사용한 CloudWatch에 대한 교차 서비스 예제
  - AWS SDK를 사용하여 Amazon DynamoDB의 성능 모니터링

AWS SDK를 사용한 CloudWatch 작업

다음 코드 예제에서는 AWS SDK에서 개별 CloudWatch 작업을 수행하는 방법을 보여줍니다. 이들 발췌문은 CloudWatch API를 직접적으로 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon CloudWatch API Reference(Amazon CloudWatch API 참조)에서 확인하세요.

예제

- AWS SDK 또는 CLI와 함께 DeleteAlarms 사용
- AWS SDK 또는 CLI와 함께 DeleteAnomalyDetector 사용
- AWS SDK 또는 CLI와 함께 DeleteDashboards 사용
- AWS SDK 또는 CLI와 함께 DescribeAlarmHistory 사용
- AWS SDK 또는 CLI와 함께 DescribeAlarms 사용
- AWS SDK 또는 CLI와 함께 DescribeAlarmsForMetric 사용
- AWS SDK 또는 CLI와 함께 DescribeAnomalyDetectors 사용
- AWS SDK 또는 CLI와 함께 DisableAlarmActions 사용
- AWS SDK 또는 CLI와 함께 EnableAlarmActions 사용
- AWS SDK 또는 CLI와 함께 GetDashboard 사용
- AWS SDK 또는 CLI와 함께 GetMetricData 사용
- AWS SDK 또는 CLI와 함께 GetMetricStatistics 사용
- AWS SDK 또는 CLI와 함께 GetMetricWidgetImage 사용
- AWS SDK 또는 CLI와 함께 ListDashboards 사용
- AWS SDK 또는 CLI와 함께 ListMetrics 사용
- AWS SDK 또는 CLI와 함께 PutAnomalyDetector 사용
- AWS SDK 또는 CLI와 함께 PutDashboard 사용
- AWS SDK 또는 CLI와 함께 PutMetricAlarm 사용
- AWS SDK 또는 CLI와 함께 PutMetricData 사용

AWS SDK 또는 CLI와 함께 **DeleteAlarms** 사용

다음 코드 예시는 DeleteAlarms의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 경보 시작하기
- 지표, 대시보드 및 경보 시작하기
- 지표 및 경보 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Delete a list of alarms from CloudWatch.
/// </summary>
/// <param name="alarmNames">A list of names of alarms to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteAlarms(List<string> alarmNames)
{
    var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
        new DeleteAlarmsRequest()
        {
            AlarmNames = alarmNames
        });

    return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteAlarms를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

경보를 삭제합니다.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DeleteAlarmsRequest request;
        request.AddAlarmNames(alarm_name);

        auto outcome = cw.DeleteAlarms(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteAlarms를 참조하십시오.

CLI

AWS CLI

경보를 삭제하는 방법

다음 예제에서는 `delete-alarms` 명령을 사용하여 'myalarm'이라는 Amazon CloudWatch 경보를 삭제합니다.

```
aws cloudwatch delete-alarms --alarm-names myalarm
```

출력:

```
This command returns to the prompt if successful.
```

- API 세부 정보는 AWS CLI Command Reference의 DeleteAlarms를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to delete (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_2;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        deleteCWAlarm(cw, alarmName);
        cw.close();
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.deleteAlarms(request);
            System.out.printf("Successfully deleted alarm %s", alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAlarms를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteAlarms를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmNames: ["Web_Server_CPU_Utilization"],
};

cw.deleteAlarms(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteAlarms를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteAlarms를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
```

```
        """
        self.cloudwatch_resource = cloudwatch_resource


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DeleteAlarms를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarms deleted.' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound .
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 DeleteAlarms를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteAnomalyDetector** 사용

다음 코드 예시는 DeleteAnomalyDetector의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
```

```
   public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
   {
       var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
           new DeleteAnomalyDetectorRequest()
           {
               SingleMetricAnomalyDetector = anomalyDetector
           });

       return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
   }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteAnomalyDetector를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void deleteAnomalyDetector(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                   .metricName(customMetricName)
```

```
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            DeleteAnomalyDetectorRequest request =
 DeleteAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAnomalyDetector를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
```

```
    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val request = DeleteAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- API 세부 정보는 Kotlin용 AWS SDK API 참조의 DeleteAnomalyDetector를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteDashboards** 사용

다음 코드 예시는 DeleteDashboards의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
_amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteDashboards를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void deleteDashboard(CloudWatchClient cw, String dashboardName)
{
        try {
            DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                    .dashboardNames(dashboardName)
                    .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

        } catch (CloudWatchException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDashboards를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest = DeleteDashboardsRequest {
        dashboardNames = listOf(dashboardName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteDashboards를 참조하십시오.

PowerShell

PowerShell용 도구

예시 1: 지정된 대시보드를 삭제하고 계속하기 전에 확인을 위해 승격합니다. 확인을 우회하려면 명령에 -Force 스위치를 추가합니다.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteDashboards를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeAlarmHistory** 사용

다음 코드 예시는 DescribeAlarmHistory의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
```

```
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeAlarmHistory를 참조하십시오.

CLI

AWS CLI

경보에 대한 기록을 검색하는 방법

다음 예제에서는 `describe-alarm-history` 명령을 사용하여 'myalarm'이라는 Amazon CloudWatch 경보에 대한 기록을 검색합니다.

```
aws cloudwatch describe-alarm-history --alarm-name "myalarm" --history-item-type
 StateUpdate
```

출력:

```
{
    "AlarmHistoryItems": [
        {
            "Timestamp": "2014-04-09T18:59:06.442Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"ALARM\",\"stateReason\":\"testing purposes\"},\"newState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were not
 greater than the threshold (70.0). The most recent datapoints: [38.958,
 40.292].\",\"stateReasonData\":{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
```

```
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}}}",
            "HistorySummary": "Alarm updated from ALARM to OK"
        },
        {
            "Timestamp": "2014-04-09T18:59:05.805Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were
 not greater than the threshold (70.0). The most recent datapoints:
 [38.839999999999996, 39.714].\",\"stateReasonData\":{\"version\":
\"1.0\",\"queryDate\":\"2014-03-11T22:45:41.569+0000\",\"startDate\":
\"2014-03-11T22:30:00.000+0000\",\"statistic\":\"Average\",\"period\":300,
\"recentDatapoints\":[38.839999999999996,39.714],\"threshold\":70.0}},\"newState
\":{\"stateValue\":\"ALARM\",\"stateReason\":\"testing purposes\"}}",
            "HistorySummary": "Alarm updated from OK to ALARM"
        }
    ]
}
```

- API 세부 정보는 AWS CLI Command Reference의 DescribeAlarmHistory를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void getAlarmHistory(CloudWatchClient cw, String fileName,
String date) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
```

```
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
            DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

            DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
            if (historyItems.isEmpty()) {
                System.out.println("No alarm history data found for " + alarmName
+ ".");
            } else {
                for (AlarmHistoryItem item : historyItems) {
                    System.out.println("History summary: " +
item.historySummary());
                    System.out.println("Time stamp: " + item.timestamp());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAlarmHistory를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAlarmHistory(fileName: String, date: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest = DescribeAlarmHistoryRequest {
        startDate = aws.smithy.kotlin.runtime.time.Instant(start)
        endDate = aws.smithy.kotlin.runtime.time.Instant(endDateVal)
        alarmName = alarmNameVal
        historyItemType = HistoryItemType.Action
    }

    CloudWatchClient { credentialsProvider = EnvironmentCredentialsProvider();
 region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- API 세부 정보는 Kotlin용 AWS SDK API 참조의 DescribeAlarmHistory를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeAlarms** 사용

다음 코드 예시는 DescribeAlarms의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 경보 시작하기
- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeAlarms를 참조하십시오.

CLI

AWS CLI

경보에 대한 정보를 나열하는 방법

다음 예제에서는 `describe-alarms` 명령을 사용하여 'myalarm'이라는 경보에 대한 정보를 제공합니다.

```
aws cloudwatch describe-alarms --alarm-names "myalarm"
```

출력:

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:myalarm",
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2012-12-27T00:49:54.032Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:123456789012:myHighCpuAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myalarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
```

```
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        }
    ]
}
```

- API 세부 정보는 AWS CLI Command Reference의 DescribeAlarms를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                    .alarmTypes(typeList)
                    .maxRecords(10)
                    .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm : alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " +
 alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAlarms를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest = DescribeAlarmsRequest {
        alarmTypes = typeList
        maxRecords = 10
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAlarms를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require "aws-sdk-cloudwatch"

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeAlarms를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_cwt->describealarms(                " oo_result is
 returned for testing purposes. "
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarms retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 DescribeAlarms를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 `DescribeAlarmsForMetric` 사용

다음 코드 예시는 `DescribeAlarmsForMetric`의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기
- 지표 및 경보 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeAlarmsForMetric을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

경보를 설명합니다.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DescribeAlarmsRequest request;
        request.SetMaxRecords(1);

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.DescribeAlarms(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to describe CloudWatch alarms:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left <<
                    std::setw(32) << "Name" <<
                    std::setw(64) << "Arn" <<
                    std::setw(64) << "Description" <<
                    std::setw(20) << "LastUpdated" <<
                    std::endl;
                header = true;
            }

            const auto &alarms = outcome.GetResult().GetMetricAlarms();
            for (const auto &alarm : alarms)
            {
                std::cout << std::left <<
```

```
                    std::setw(32) << alarm.GetAlarmName() <<
                    std::setw(64) << alarm.GetAlarmArn() <<
                    std::setw(64) << alarm.GetAlarmDescription() <<
                    std::setw(20) <<
                    alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                        SIMPLE_DATE_FORMAT_STR) <<
                    std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DescribeAlarmsForMetric을 참조하십시오.

CLI

AWS CLI

지표와 관련된 경보에 대한 정보를 표시하는 방법

다음 예제에서는 `describe-alarms-for-metric` 명령을 사용하여 Amazon EC2 CPUUtilization 지표 및 ID i-0c986c72의 인스턴스와 관련된 모든 경보에 대한 정보를 표시합니다.

```
aws cloudwatch describe-alarms-for-metric --metric-name CPUUtilization --
namespace AWS/EC2 --dimensions Name=InstanceId,Value=i-0c986c72
```

출력:

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 10,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm2",
            "StateUpdatedTimestamp": "2013-10-30T03:03:51.479Z",
            "AlarmConfigurationUpdatedTimestamp": "2013-10-30T03:03:50.865Z",
            "ComparisonOperator": "GreaterThanOrEqualToThreshold",
            "AlarmActions": [
```

```
                "arn:aws:sns:us-east-1:111122223333:NotifyMe"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2013-10-30T03:03:51.479+0000\",\"startDate\":\"2013-10-30T02:08:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":
[40.698,39.612,42.432,39.796,38.816,42.28,42.854,40.088,40.760000000000005,41.316],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm2",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 10 datapoints were not
 greater than or equal to the threshold (70.0). The most recent datapoints:
 [40.760000000000005, 41.316].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        },
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm",
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2014-04-09T22:26:05.958Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:HighCPUAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
```

```
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": false,
            "MetricName": "CPUUtilization"
        }
    ]
}
```

- API 세부 정보는 AWS CLI Command Reference의 DescribeAlarmsForMetric을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void checkForMetricAlarm(CloudWatchClient cw, String fileName)
{
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            while (!hasAlarm && retries > 0) {
                DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for " + customMetricName
+ " after 10 retries.");
            else
                System.out.println("Alarm state found for " + customMetricName +
".");

        } catch (CloudWatchException | IOException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAlarmsForMetric을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DescribeAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DescribeAlarmsForMetric을 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.describeAlarms({ StateValue: "INSUFFICIENT_DATA" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DescribeAlarmsForMetric을 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest = DescribeAlarmsForMetricRequest {
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) println("No Alarm state found for $customMetricName after
 10 retries.") else println("Alarm state found for $customMetricName.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DescribeAlarmsForMetric를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
 metric_name)
        return alarm_iter
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DescribeAlarmsForMetric를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts "-" * 16
      puts "Name:           " + alarm.alarm_name
      puts "State value:    " + alarm.state_value
      puts "State reason:   " + alarm.state_reason
      puts "Metric:         " + alarm.metric_name
      puts "Namespace:      " + alarm.namespace
      puts "Statistic:      " + alarm.statistic
      puts "Period:         " + alarm.period.to_s
      puts "Unit:           " + alarm.unit.to_s
      puts "Eval. periods:  " + alarm.evaluation_periods.to_s
      puts "Threshold:      " + alarm.threshold.to_s
      puts "Comp. operator: " + alarm.comparison_operator

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts "OK actions:"
        alarm.ok_actions.each do |a|
          puts "  " + a
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
```

```
        puts "Alarm actions:"
        alarm.alarm_actions.each do |a|
          puts "  " + a
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
          alarm.insufficient_data_actions.count.positive?
        puts "Insufficient data actions:"
        alarm.insufficient_data_actions.each do |a|
          puts "  " + a
        end
      end

      puts "Dimensions:"
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: " + d.name + ", Value: " + d.value
        end
      else
        puts "  None for this alarm."
      end
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ""

  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby cw-ruby-example-show-alarms.rb REGION"
    puts "Example: ruby cw-ruby-example-show-alarms.rb us-east-1"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = "us-east-1"
  # Otherwise, use the values as specified at the command prompt.
  else
```

```
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts "Available alarms:"
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeAlarmsForMetric을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DescribeAnomalyDetectors** 사용

다음 코드 예시는 DescribeAnomalyDetectors의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
```

```
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DescribeAnomalyDetectors를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
```

```
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
            List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
            for (AnomalyDetector detector : anomalyDetectorList) {
                System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: " + detector.stateValue());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.xAPI 참조의 DescribeAnomalyDetectors 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest = DescribeAnomalyDetectorsRequest {
        maxResults = 10
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- API 세부 정보는 Kotlin용 AWS SDK API 참조의 DescribeAnomalyDetectors를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DisableAlarmActions** 사용

다음 코드 예시는 DisableAlarmActions의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 경보 시작하기
- 지표 및 경보 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보에 대한 내용은 AWS SDK for .NET API 참조의 DisableAlarmActions를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

경보 작업 사용을 중지합니다.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::DisableAlarmActionsRequest
 disableAlarmActionsRequest;
        disableAlarmActionsRequest.AddAlarmNames(alarm_name);

        auto disableAlarmActionsOutcome =
 cw.DisableAlarmActions(disableAlarmActionsRequest);
        if (!disableAlarmActionsOutcome.IsSuccess())
        {
            std::cout << "Failed to disable actions for alarm " << alarm_name <<
                ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully disabled actions for alarm " <<
                alarm_name << std::endl;
        }
```

- API 세부 정보에 대한 내용은 AWS SDK for C++ API 참조의 DisableAlarmActions를 참조하십시오.

CLI

AWS CLI

경보에 대한 작업을 비활성화하는 방법

다음 예제에서는 `disable-alarm-actions` 명령을 사용하여 myalarm이라는 경보에 대한 모든 작업을 비활성화합니다.

```
aws cloudwatch disable-alarm-actions --alarm-names myalarm
```

이 명령은 성공하면 프롬프트로 돌아갑니다.

- API 세부 정보는 AWS CLI Command Reference의 DisableAlarmActions를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- API 세부 정보에 대한 내용은 AWS SDK for Java 2.x API 참조의 DisableAlarmActions를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DisableAlarmActionsCommand({
    AlarmNames: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
```

```
export const client = new CloudWatchClient({});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DisableAlarmActions을 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.disableAlarmActions(
  { AlarmNames: ["Web_Server_CPU_Utilization"] },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보에 대한 내용은 AWS SDK for JavaScript API 참조의 DisableAlarmActions를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun disableActions(alarmName: String) {

    val request = DisableAlarmActionsRequest {
        alarmNames = listOf(alarmName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DisableAlarmActions를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
```

```
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DisableAlarmActions를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  return true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  return false
end

# Example usage:
def run_me
  alarm_name = "ObjectsInBucket"
  alarm_description = "Objects exist in this bucket for more than 1 day."
  metric_name = "NumberOfObjects"
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ["arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic"]
```

```
namespace = "AWS/S3"
statistic = "Average"
dimensions = [
  {
    name: "BucketName",
    value: "doc-example-bucket"
  },
  {
    name: "StorageType",
    value: "AllStorageTypes"
  }
]
period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
unit = "Count"
evaluation_periods = 1 # More than one day.
threshold = 1 # One object.
comparison_operator = "GreaterThanThreshold" # More than one object.
# Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
region = "us-east-1"

cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

if alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  puts "Alarm '#{alarm_name}' created or updated."
else
  puts "Could not create or update alarm '#{alarm_name}'."
end

if alarm_actions_disabled?(cloudwatch_client, alarm_name)
  puts "Alarm '#{alarm_name}' disabled."
```

```
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보에 대한 내용은 AWS SDK for Ruby API 참조의 DisableAlarmActions를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    "Disables actions on the specified alarm. "
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarm actions disabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API에 대한 자세한 내용은 AWS SDK for SAP ABAP API 참조의 DisableAlarmActions를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **EnableAlarmActions** 사용

다음 코드 예시는 EnableAlarmActions의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표 및 경보 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 EnableAlarmActions를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

경보 작업을 사용합니다.

```
    Aws::CloudWatch::CloudWatchClient cw;
    Aws::CloudWatch::Model::PutMetricAlarmRequest request;
    request.SetAlarmName(alarm_name);
    request.SetComparisonOperator(
        Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
    request.SetEvaluationPeriods(1);
    request.SetMetricName("CPUUtilization");
    request.SetNamespace("AWS/EC2");
    request.SetPeriod(60);
    request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
    request.SetThreshold(70.0);
    request.SetActionsEnabled(false);
    request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
    request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
    request.AddAlarmActions(actionArn);

    Aws::CloudWatch::Model::Dimension dimension;
    dimension.SetName("InstanceId");
    dimension.SetValue(instanceId);
    request.AddDimensions(dimension);
```

```
    auto outcome = cw.PutMetricAlarm(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create CloudWatch alarm:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
    enable_request.AddAlarmNames(alarm_name);

    auto enable_outcome = cw.EnableAlarmActions(enable_request);
    if (!enable_outcome.IsSuccess())
    {
        std::cout << "Failed to enable alarm actions:" <<
            enable_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    std::cout << "Successfully created alarm " << alarm_name <<
        " and enabled actions on it." << std::endl;
```

- API 세부 정보는 AWS SDK for C++ API 참조의 EnableAlarmActions를 참조하십시오.

CLI

AWS CLI

경보에 대한 모든 작업을 활성화하는 방법

다음 예제에서는 `enable-alarm-actions` 명령을 사용하여 myalarm이라는 경보에 대한 모든 작업을 활성화합니다.

```
aws cloudwatch enable-alarm-actions --alarm-names myalarm
```

이 명령은 성공하면 프롬프트로 돌아갑니다.

- API 세부 정보는 AWS CLI Command Reference의 EnableAlarmActions를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to enable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
```

```
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request =
 EnableAlarmActionsRequest.builder()
                    .alarmNames(alarm)
                    .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 EnableAlarmActions를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { EnableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
```

```
import { client } from "../libs/client.js";

const run = async () => {
  const command = new EnableAlarmActionsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 EnableAlarmActions를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"],
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Alarm action added", data);
    var paramsEnableAlarmAction = {
      AlarmNames: [params.AlarmName],
    };
    cw.enableAlarmActions(paramsEnableAlarmAction, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Alarm action enabled", data);
      }
    });
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 EnableAlarmActions를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun enableActions(alarm: String) {

    val request = EnableAlarmActionsRequest {
        alarmNames = listOf(alarm)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 EnableAlarmActions를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CloudWatchWrapper:
```

```
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 EnableAlarmActions를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    "Enable actions on the specified alarm."
    TRY.
        lo_cwt->enablealarmactions(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarm actions enabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 EnableAlarmActions를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetDashboard** 사용

다음 코드 예시는 GetDashboard의 사용 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get information on a dashboard.
/// </summary>
/// <param name="dashboardName">The name of the dashboard.</param>
/// <returns>A JSON object with dashboard information.</returns>
public async Task<string> GetDashboard(string dashboardName)
{
    var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
        new GetDashboardRequest()
        {
            DashboardName = dashboardName
        });

    return dashboardResponse.DashboardBody;
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetDashboard를 참조하십시오.

PowerShell

PowerShell용 도구

예시 1: 지정된 대시보드의 arn 본문을 반환합니다.

```
Get-CWDashboard -DashboardName Dashboard1
```

출력:

```
DashboardArn                                            DashboardBody
```

```
------------                                                  -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetDashboard를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetMetricData** 사용

다음 코드 예시는 GetMetricData의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
```

```
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
 ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        await foreach (var data in paginatedMetricData.MetricDataResults)
        {
            metricData.Add(data);
        }
        return metricData;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetMetricData를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void getCustomMetricData(CloudWatchClient cw, String fileName)
{
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           // Set the date.
           Instant nowDate = Instant.now();

           long hours = 1;
           long minutes = 30;
           Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                   ChronoUnit.MINUTES);

           Metric met = Metric.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

           MetricStat metStat = MetricStat.builder()
                   .stat("Maximum")
                   .period(1)
                   .metric(met)
                   .build();

           MetricDataQuery dataQUery = MetricDataQuery.builder()
                   .metricStat(metStat)
                   .id("foo2")
                   .returnData(true)
                   .build();

           List<MetricDataQuery> dq = new ArrayList<>();
           dq.add(dataQUery);

           GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                   .maxDatapoints(10)
```

```
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(date2)
                    .metricDataQueries(dq)
                    .build();

            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
 item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetMetricData를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
```

```
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(
        minutes,
        ChronoUnit.MINUTES
    )

    val met = Metric {
        metricName = customMetricName
        namespace = customMetricNamespace
    }

    val metStat = MetricStat {
        stat = "Maximum"
        period = 1
        metric = met
    }

    val dataQUery = MetricDataQuery {
        metricStat = metStat
        id = "foo2"
        returnData = true
    }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq = GetMetricDataRequest {
        maxDatapoints = 10
        scanBy = ScanBy.TimestampDescending
        startTime = aws.smithy.kotlin.runtime.time.Instant(nowDate)
        endTime = aws.smithy.kotlin.runtime.time.Instant(date2)
        metricDataQueries = dq
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}
```

- API 세부 정보는 Kotlin용 AWS SDK API 참조의 GetMetricData를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetMetricStatistics** 사용

다음 코드 예시는 GetMetricStatistics의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기
- 지표 및 경보 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
```

```
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetMetricStatistics를 참조하십시오.

CLI

AWS CLI

EC2 인스턴스별 CPU 사용률을 가져오는 방법

다음 예제에서는 `get-metric-statistics` 명령을 사용하여 ID i-abcdef의 EC2 인스턴스에 대한 CPU 사용률을 가져옵니다.

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-time
 2014-04-08T23:18:00Z --end-time 2014-04-09T23:18:00Z --period 3600 --namespace
 AWS/EC2 --statistics Maximum --dimensions Name=InstanceId,Value=i-abcdef
```

출력:

```
{
    "Datapoints": [
        {
            "Timestamp": "2014-04-09T11:18:00Z",
            "Maximum": 44.79,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T20:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T19:18:00Z",
            "Maximum": 50.85,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T09:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T03:18:00Z",
            "Maximum": 76.84,
            "Unit": "Percent"
        },
        {
```

```
            "Timestamp": "2014-04-09T21:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T14:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T08:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T16:18:00Z",
            "Maximum": 45.55,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T06:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T13:18:00Z",
            "Maximum": 45.08,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T05:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T18:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T17:18:00Z",
            "Maximum": 52.08,
            "Unit": "Percent"
        },
```

```
        {
            "Timestamp": "2014-04-09T07:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T02:18:00Z",
            "Maximum": 51.23,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T12:18:00Z",
            "Maximum": 47.67,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-08T23:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T10:18:00Z",
            "Maximum": 51.91,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T04:18:00Z",
            "Maximum": 47.13,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T15:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T00:18:00Z",
            "Maximum": 48.16,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T01:18:00Z",
            "Maximum": 49.18,
            "Unit": "Percent"
```

```
        }
    ],
    "Label": "CPUUtilization"
}
```

여러 측정기준을 지정하는 방법

다음 예제는 여러 측정기준을 지정하는 방법을 보여줍니다. 각 측정기준은 이름과 값 사이에 쉼표가 있는 이름/값 페어로 지정됩니다. 여러 측정기준은 공백으로 구분됩니다. 단일 지표에 여러 개의 측정기준이 포함된 경우에는 정의된 모든 측정기준에 대해 값을 지정해야 합니다.

`get-metric-statistics` 명령을 사용하는 더 많은 예시는 Amazon CloudWatch 개발자 안내서의 지표에 대한 통계 얻기를 참조하세요.

```
aws cloudwatch get-metric-statistics --metric-name Buffers --
namespace MyNameSpace --dimensions Name=InstanceID,Value=i-abcdef
 Name=InstanceType,Value=m1.small --start-time 2016-10-15T04:00:00Z --end-time
 2016-10-19T07:00:00Z --statistics Average --period 60
```

- API 세부 정보는 AWS CLI Command Reference의 GetMetricStatistics를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
 nameSpace, String metVal,
            String metricOption, String date, Dimension myDimension) {
        try {
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();

            GetMetricStatisticsRequest statisticsRequest =
 GetMetricStatisticsRequest.builder()
                    .endTime(endDate)
```

```
                    .startTime(start)
                    .dimensions(myDimension)
                    .metricName(metVal)
                    .namespace(nameSpace)
                    .period(86400)
                    .statistics(Statistic.fromValue(metricOption))
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetMetricStatistics를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAndDisplayMetricStatistics(nameSpaceVal: String, metVal: String,
 metricOption: String, date: String, myDimension: Dimension) {
```

```
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest = GetMetricStatisticsRequest {
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
        dimensions = listOf(myDimension)
        metricName = metVal
        namespace = nameSpaceVal
        period = 86400
        statistics = listOf(Statistic.fromValue(metricOption))
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetMetricStatistics를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CloudWatchWrapper:
```

```
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period
must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
value
                           or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
```

```
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
            raise
        else:
            return stats
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetMetricStatistics를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetMetricWidgetImage** 사용

다음 코드 예시는 `GetMetricWidgetImage`의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
```

```
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetMetricWidgetImage를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void getAndOpenMetricImage(CloudWatchClient cw, String
fileName) {
       System.out.println("Getting Image data for custom metric.");
       try {
           String myJSON = "{\n" +
                   "  \"title\": \"Example Metric Graph\",\n" +
                   "  \"view\": \"timeSeries\",\n" +
                   "  \"stacked \": false,\n" +
                   "  \"period\": 10,\n" +
                   "  \"width\": 1400,\n" +
                   "  \"height\": 600,\n" +
                   "  \"metrics\": [\n" +
                   "    [\n" +
                   "      \"AWS/Billing\",\n" +
                   "      \"EstimatedCharges\",\n" +
                   "      \"Currency\",\n" +
                   "      \"USD\"\n" +
                   "    ]\n" +
                   "  ]\n" +
                   "}";

           GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
                   .metricWidget(myJSON)
                   .build();

           GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
```

```
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
            try (FileOutputStream outputStream = new
 FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetMetricWidgetImage를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
```

```
            ]
        ]
        }"""

    val imageRequest = GetMetricWidgetImageRequest {
        metricWidget = myJSON
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- API에 대한 세부 정보는 Kotlin용 AWS SDK API 참조의 GetMetricWidgetImage를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListDashboards** 사용

다음 코드 예시는 `ListDashboards`의 사용 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get a list of dashboards.
```

```
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListDashboards를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                    .flatMap(r -> r.dashboardEntries().stream())
                    .forEach(entry -> {
                        System.out.println("Dashboard name is: " +
entry.dashboardName());
                        System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
                    });

        } catch (CloudWatchException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDashboards를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListDashboards를 참조하십시오.

PowerShell

PowerShell용 도구

예시 1: 계정의 대시보드 컬렉션을 반환합니다.

```
Get-CWDashboardList
```

출력:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

예시 2: 이름이 접두사 'dev'로 시작하는 계정의 대시보드 컬렉션을 반환합니다.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListDashboards를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 `ListMetrics` 사용

다음 코드 예시는 `ListMetrics`의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기
- 지표 및 경보 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
```

```
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListMetrics를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
```

```
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

지표를 나열합니다.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::ListMetricsRequest request;

        if (argc > 1)
        {
            request.SetMetricName(argv[1]);
        }

        if (argc > 2)
        {
            request.SetNamespace(argv[2]);
        }

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.ListMetrics(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to list CloudWatch metrics:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left << std::setw(48) << "MetricName" <<
                    std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
                    std::endl;
                header = true;
            }

            const auto &metrics = outcome.GetResult().GetMetrics();
```

```
            for (const auto &metric : metrics)
            {
                std::cout << std::left << std::setw(48) <<
                    metric.GetMetricName() << std::setw(32) <<
                    metric.GetNamespace();
                const auto &dimensions = metric.GetDimensions();
                for (auto iter = dimensions.cbegin();
                    iter != dimensions.cend(); ++iter)
                {
                    const auto &dimkv = *iter;
                    std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
                    if (iter + 1 != dimensions.cend())
                    {
                        std::cout << ", ";
                    }
                }
                std::cout << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListMetrics를 참조하십시오.

CLI

AWS CLI

Amazon SNS에 대한 지표를 나열하는 방법

다음 `list-metrics` 예제는 Amazon SNS에 대한 지표를 표시합니다.

```
aws cloudwatch list-metrics \
    --namespace "AWS/SNS"
```

출력:

```
{
    "Metrics": [
        {
```

```
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfNotificationsFailed"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
```

```
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfNotificationsFailed"
        }
    ]
}
```

- API 세부 정보는 AWS CLI Command Reference의 ListMetrics를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        boolean done = false;
        String nextToken = null;

        try {
            while (!done) {

                ListMetricsResponse response;
                if (nextToken == null) {
                    ListMetricsRequest request = ListMetricsRequest.builder()
                            .namespace(namespace)
                            .build();

                    response = cw.listMetrics(request);
                } else {
                    ListMetricsRequest request = ListMetricsRequest.builder()
                            .namespace(namespace)
                            .nextToken(nextToken)
                            .build();

                    response = cw.listMetrics(request);
                }

                for (Metric metric : response.metrics()) {
                    System.out.printf("Retrieved metric %s",
metric.metricName());
                    System.out.println();
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    nextToken = response.nextToken();
```

```
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListMetrics를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { ListMetricsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

export const main = () => {
  // Use the AWS console to see available namespaces and metric names. Custom
 metrics can also be created.
  // https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
viewing_metrics_with_cloudwatch.html
  const command = new ListMetricsCommand({
    Dimensions: [
      {
        Name: "LogGroupName",
      },
    ],
    MetricName: "IncomingLogEvents",
    Namespace: "AWS/Logs",
  });
```

```
  return client.send(command);
};
```

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListMetrics를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  Dimensions: [
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};
```

```
cw.listMetrics(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListMetrics를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListMetrics를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def list_metrics(self, namespace, name, recent=False):
        """
        Gets the metrics within a namespace that have the specified name.
        If the metric has no dimensions, a single metric is returned.
        Otherwise, metrics for all dimensions are returned.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param recent: When True, only metrics that have been active in the last
                       three hours are returned.
        :return: An iterator that yields the retrieved metrics.
        """
        try:
            kwargs = {"Namespace": namespace, "MetricName": name}
            if recent:
                kwargs["RecentlyActive"] = "PT3H"  # List past 3 hours only
            metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
            logger.info("Got metrics for %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
            raise
        else:
```

```
            return metric_iter
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListMetrics를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts "    Dimensions:"
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts "No dimensions found."
      end
    end
  else
```

```
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      "Note that it could take up to 15 minutes for recently-added metrics " \
      "to become available."
  end
end

# Example usage:
def run_me
  metric_namespace = "SITE/TRAFFIC"
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisitors",
    "SiteName",
    "example.com",
    5_885.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisits",
    "SiteName",
    "example.com",
    8_628.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "PageViews",
    "PageURL",
    "example.html",
    18_057.0,
    "Count"
  )
```

```
  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListMetrics를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   "The following list-metrics example displays the metrics for Amazon
CloudWatch."
   TRY.
       oo_result = lo_cwt->listmetrics(            " oo_result is returned for
testing purposes. "
         iv_namespace = iv_namespace
       ).
       DATA(lt_metrics) = oo_result->get_metrics( ).
       MESSAGE 'Metrics retrieved.' TYPE 'I'.
     CATCH /aws1/cx_cwtinvparamvalueex .
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
   ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 ListMetrics를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutAnomalyDetector** 사용

다음 코드 예시는 PutAnomalyDetector의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /// <summary>
   /// Add an anomaly detector for a single metric.
   /// </summary>
   /// <param name="anomalyDetector">A single metric anomaly detector.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
   {
       var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
           new PutAnomalyDetectorRequest()
           {
               SingleMetricAnomalyDetector = anomalyDetector
           });

       return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
   }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 PutAnomalyDetector를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .stat("Maximum")
                   .build();

           PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                   .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                   .build();

           cw.putAnomalyDetector(anomalyDetectorRequest);
           System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

       } catch (CloudWatchException | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutAnomalyDetector를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val anomalyDetectorRequest = PutAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- API 세부 정보는 AWS Kotlin용 SDK API 참조의 PutAnomalyDetector를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutDashboard** 사용

다음 코드 예시는 PutDashboard의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
```

```
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
 newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
```

```
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 PutDashboard를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            PutDashboardResponse response = cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully created.");
            List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
            if (messages.isEmpty()) {
                System.out.println("There are no messages in the new Dashboard");
            } else {
```

```
                for (DashboardValidationMessage message : messages) {
                    System.out.println("Message is: " + message.message());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutDashboard를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createDashboardWithMetrics(dashboardNameVal: String, fileNameVal:
 String) {
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
```

```
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutDashboard를 참조하십시오.

PowerShell

PowerShell용 도구

예시 1: 지표 위젯 2개를 나란히 포함하도록 'Dashboard1'이라는 대시보드를 생성하거나 업데이트합니다.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
                      "CPUUtilization",
                      "InstanceId",
                      "i-012345"
                   ]
                ],
                "period":300,
                "stat":"Average",
                "region":"us-east-1",
                "title":"EC2 Instance CPU"
             }
        },
        {
             "type":"metric",
             "x":12,
             "y":0,
```

```
            "width":12,
            "height":6,
            "properties":{
               "metrics":[
                  [
                     "AWS/S3",
                     "BucketSizeBytes",
                     "BucketName",
                     "MyBucketName"
                  ]
               ],
               "period":86400,
               "stat":"Maximum",
               "region":"us-east-1",
               "title":"MyBucketName bytes"
            }
         }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

예시 2: 대시보드를 생성하거나 업데이트하여 대시보드를 설명하는 콘텐츠를 cmdlet에 전달합니다.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 PutDashboard를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutMetricAlarm** 사용

다음 코드 예시는 PutMetricAlarm의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 경보 시작하기
- 지표, 대시보드 및 경보 시작하기
- 지표 및 경보 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
    {
        try
        {
            var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
                new PutMetricAlarmRequest()
                {
                    AlarmActions = alarmActions,
```

```
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
                    ComparisonOperator = comparison,
                    Threshold = threshold,
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
 has already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
 alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 PutMetricAlarm을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

지표를 감시할 경보를 생성합니다.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::PutMetricAlarmRequest request;
        request.SetAlarmName(alarm_name);
        request.SetComparisonOperator(
            Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
        request.SetEvaluationPeriods(1);
        request.SetMetricName("CPUUtilization");
        request.SetNamespace("AWS/EC2");
        request.SetPeriod(60);
        request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
        request.SetThreshold(70.0);
        request.SetActionsEnabled(false);
        request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
        request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("InstanceId");
```

```
        dimension.SetValue(instanceId);

        request.AddDimensions(dimension);

        auto outcome = cw.PutMetricAlarm(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 PutMetricAlarm을 참조하십시오.

CLI

AWS CLI

CPU 사용률이 70%를 초과할 때 Amazon Simple Notification Service 이메일 메시지 보내기

다음 예제에서는 `put-metric-alarm` 명령을 사용하여 CPU 사용률이 70%를 초과할 때 Amazon Simple Notification Service 이메일 메시지를 보냅니다.

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70 percent" --metric-name CPUUtilization --namespace AWS/
EC2 --statistic Average --period 300 --threshold 70 --comparison-operator
 GreaterThanThreshold  --dimensions "Name=InstanceId,Value=i-12345678" --
evaluation-periods 2 --alarm-actions arn:aws:sns:us-east-1:111122223333:MyTopic
 --unit Percent
```

이 명령은 성공하면 프롬프트로 돌아갑니다. 같은 이름의 경보가 이미 있는 경우 새 경보가 해당 경보를 덮어씁니다.

여러 측정기준을 지정하는 방법

다음 예제는 여러 측정기준을 지정하는 방법을 보여줍니다. 각 측정기준은 이름과 값 사이에 쉼표가 있는 이름/값 페어로 지정됩니다. 여러 측정기준은 공백으로 구분됩니다.

```
aws cloudwatch put-metric-alarm --alarm-name "Default_Test_Alarm3" --alarm-
description "The default example alarm" --namespace "CW EXAMPLE METRICS" --
metric-name Default_Test --statistic Average --period 60 --evaluation-periods 3
 --threshold 50 --comparison-operator GreaterThanOrEqualToThreshold --dimensions
 Name=key1,Value=value1 Name=key2,Value=value2
```

- API 세부 정보는 AWS CLI 명령 참조의 PutMetricAlarm을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static String createAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                    .alarmActions(alarmActions)
                    .alarmDescription("Example metric alarm")
```

```
                    .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                    .threshold(100.00)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .evaluationPeriods(1)
                    .period(10)
                    .statistic("Maximum")
                    .datapointsToAlarm(1)
                    .treatMissingData("ignore")
                    .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName + " was successfully created!");
            return alarmName;

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutMetricAlarm을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";
```

```
const run = async () => {
  // This alarm triggers when CPUUtilization exceeds 70% for one minute.
  const command = new PutMetricAlarmCommand({
    AlarmName: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
    ComparisonOperator: "GreaterThanThreshold",
    EvaluationPeriods: 1,
    MetricName: "CPUUtilization",
    Namespace: "AWS/EC2",
    Period: 60,
    Statistic: "Average",
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: "Alarm when server CPU exceeds 70%",
    Dimensions: [
      {
        Name: "InstanceId",
        Value: process.env.EC2_INSTANCE_ID, // Set the value of EC_INSTANCE_ID to
 the Id of an existing Amazon EC2 instance.
      },
    ],
    Unit: "Percent",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 PutMetricAlarm을 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
```

```
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 PutMetricAlarm을 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun putMetricAlarm(alarmNameVal: String, instanceIdVal: String) {

    val dimensionOb = Dimension {
        name = "InstanceId"
        value = instanceIdVal
    }

    val request = PutMetricAlarmRequest {
        alarmName = alarmNameVal
        comparisonOperator = ComparisonOperator.GreaterThanThreshold
        evaluationPeriods = 1
        metricName = "CPUUtilization"
        namespace = "AWS/EC2"
        period = 60
        statistic = Statistic.fromValue("Average")
        threshold = 70.0
        actionsEnabled = false
        alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
        unit = StandardUnit.fromValue("Seconds")
        dimensions = listOf(dimensionOb)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
```

```
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutMetricAlarm를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
```

```
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over
the
                             alarm threshold before the alarm is set into an
alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the
threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 PutMetricAlarm를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
```

```
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
```

```
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  return true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  return false
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutMetricAlarm을 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        lo_cwt->putmetricalarm(
          iv_alarmname                = iv_alarm_name
          iv_comparisonoperator       = iv_comparison_operator
          iv_evaluationperiods        = iv_evaluation_periods
          iv_metricname               = iv_metric_name
          iv_namespace                = iv_namespace
          iv_statistic                = iv_statistic
          iv_threshold                = iv_threshold
          iv_actionsenabled           = iv_actions_enabled
          iv_alarmdescription         = iv_alarm_description
          iv_unit                     = iv_unit
          iv_period                   = iv_period
          it_dimensions               = it_dimensions
        ).
        MESSAGE 'Alarm created.' TYPE 'I'.
      CATCH /aws1/cx_cwtlimitexceededfault.
```

```
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 PutMetricAlarm을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutMetricData** 사용

다음 코드 예시는 PutMetricData의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 지표, 대시보드 및 경보 시작하기
- 지표 및 경보 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
```

```
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
        return customData;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 PutMetricData를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

지표에 데이터를 입력합니다.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("UNIQUE_PAGES");
        dimension.SetValue("URLS");

        Aws::CloudWatch::Model::MetricDatum datum;
        datum.SetMetricName("PAGES_VISITED");
        datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
        datum.SetValue(data_point);
        datum.AddDimensions(dimension);

        Aws::CloudWatch::Model::PutMetricDataRequest request;
        request.SetNamespace("SITE/TRAFFIC");
        request.AddMetricData(datum);

        auto outcome = cw.PutMetricData(request);
        if (!outcome.IsSuccess())
        {
```

```
            std::cout << "Failed to put sample metric data:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully put sample metric data" << std::endl;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 PutMetricData를 참조하십시오.

CLI

AWS CLI

Amazon CloudWatch에 사용자 지정 지표를 게시하는 방법

다음 예제에서는 `put-metric-data` 명령을 사용하여 Amazon CloudWatch에 사용자 지정 지표를 게시합니다.

```
aws cloudwatch put-metric-data --namespace "Usage Metrics" --metric-data file://
metric.json
```

지표 자체의 값은 JSON 파일인 `metric.json`에 저장됩니다.

해당 파일의 내용은 다음과 같습니다.

```
[
  {
    "MetricName": "New Posts",
    "Timestamp": "Wednesday, June 12, 2013 8:28:20 PM",
    "Value": 0.50,
    "Unit": "Count"
  }
]
```

자세한 내용은 Amazon CloudWatch 개발자 안내서의 사용자 지정 지표 게시를 참조하세요.

여러 측정기준을 지정하는 방법

다음 예제는 여러 측정기준을 지정하는 방법을 보여줍니다. 각 측정기준은 이름=값 페어로 지정됩니다. 여러 측정기준은 쉼표로 구분됩니다.

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace
 MyNameSpace --unit Bytes --value 231434333 --dimensions
 InstanceID=1-23456789,InstanceType=m1.small
```

- API 세부 정보는 AWS CLI Command Reference의 PutMetricData를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void addMetricDataForAlarm(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           // Set an Instant object.
           String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
           Instant instant = Instant.parse(time);

           MetricDatum datum = MetricDatum.builder()
                   .metricName(customMetricName)
                   .unit(StandardUnit.NONE)
                   .value(1001.00)
                   .timestamp(instant)
                   .build();
```

```
            MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric " +
customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutMetricData를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
```

```
import { client } from "../libs/client.js";

const run = async () => {
  // See https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/
API_PutMetricData.html#API_PutMetricData_RequestParameters
  // and https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
publishingMetrics.html
  // for more information about the parameters in this command.
  const command = new PutMetricDataCommand({
    MetricData: [
      {
        MetricName: "PAGES_VISITED",
        Dimensions: [
          {
            Name: "UNIQUE_PAGES",
            Value: "URLS",
          },
        ],
        Unit: "None",
        Value: 1.0,
      },
    ],
    Namespace: "SITE/TRAFFIC",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.

- API 세부 정보는 AWS SDK for JavaScript API 참조의 PutMetricData를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
    {
      MetricName: "PAGES_VISITED",
      Dimensions: [
        {
          Name: "UNIQUE_PAGES",
          Value: "URLS",
        },
      ],
      Unit: "None",
      Value: 1.0,
    },
  ],
  Namespace: "SITE/TRAFFIC",
};

cw.putMetricData(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", JSON.stringify(data));
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 PutMetricData를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1001.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val datum2 = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1002.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }
```

```
    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request = PutMetricDataRequest {
        namespace = customMetricNamespace
        metricData = metricDataList
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 PutMetricData를 참조하십시오.

PowerShell

PowerShell용 도구

예시 1: 새 MetricDatum 객체를 생성하고, Amazon Web Services CloudWatch 지표에 씁니다.

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 PutMetricData를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is
 given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
 name)
            raise
```

CloudWatch 지표에 데이터 집합을 추가합니다.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
 set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
```

```
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 PutMetricData를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require "aws-sdk-cloudwatch"

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
```

```
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  return true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  return false
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutMetricData를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 CloudWatch 시나리오

다음 코드 예제에서는 AWS SDK로 CloudWatch에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 CloudWatch 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여줍니다. 각 시나리오에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

예제

- AWS SDK를 사용하여 CloudWatch 경보 시작하기
- AWS SDK를 사용하여 CloudWatch 지표, 대시보드 및 경보 시작하기
- AWS SDK를 사용하여 CloudWatch 지표 및 경보 관리

AWS SDK를 사용하여 CloudWatch 경보 시작하기

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 경보를 생성합니다.
- 경보 작업을 비활성화합니다.
- 경보를 설명합니다.
- 경보를 삭제합니다.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.
```

```
    "Create an alarm"
    TRY.
        lo_cwt->putmetricalarm(
          iv_alarmname                = iv_alarm_name
          iv_comparisonoperator       = iv_comparison_operator
          iv_evaluationperiods        = iv_evaluation_periods
          iv_metricname               = iv_metric_name
          iv_namespace                = iv_namespace
          iv_statistic                = iv_statistic
          iv_threshold                = iv_threshold
          iv_actionsenabled           = iv_actions_enabled
          iv_alarmdescription         = iv_alarm_description
          iv_unit                     = iv_unit
          iv_period                   = iv_period
          it_dimensions               = it_dimensions
        ).
        MESSAGE 'Alarm created' TYPE 'I'.
      CATCH /aws1/cx_cwtlimitexceededfault.
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
    ENDTRY.

    "Create an ABAP internal table for the created alarm."
    CREATE OBJECT lo_alarmname EXPORTING iv_value = iv_alarm_name.
    INSERT lo_alarmname INTO TABLE lt_alarmnames.

    "Disable alarm actions."
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames               = lt_alarmnames
        ).
        MESSAGE 'Alarm actions disabled' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
        DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception-
>av_err_code }" - { lo_disablealarm_exception->av_err_msg }|.
        MESSAGE lv_disablealarm_error TYPE 'E'.
    ENDTRY.

    "Describe alarm using the same ABAP internal table."
    TRY.
        oo_result = lo_cwt->describealarms(                       " oo_result is
 returned for testing purpose "
          it_alarmnames               = lt_alarmnames
        ).
```

```
        MESSAGE 'Alarms retrieved' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
        DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
        MESSAGE lv_describealarms_error TYPE 'E'.
    ENDTRY.

    "Delete alarm."
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = lt_alarmnames
        ).
        MESSAGE 'Alarms deleted' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound .
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 다음 주제를 참조하십시오.
  - DeleteAlarms
  - DescribeAlarms
  - DisableAlarmActions
  - PutMetricAlarm

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 CloudWatch 지표, 대시보드 및 경보 시작하기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- CloudWatch 네임스페이스 및 지표를 나열합니다.
- 지표 및 예상 청구에 대한 통계를 가져옵니다.
- 대시보드를 생성하고 업데이트합니다.
- 데이터를 생성하여 지표에 추가합니다.
- 경보를 생성하고 트리거한 다음 경보 기록을 봅니다.
- 이상 탐지기를 추가합니다.
- 지표 이미지를 가져온 다음 리소스를 정리합니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
public class CloudWatchScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
    https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

    This .NET example performs the following tasks:
        1. List and select a CloudWatch namespace.
        2. List and select a CloudWatch metric.
        3. Get statistics for a CloudWatch metric.
        4. Get estimated billing statistics for the last week.
        5. Create a new CloudWatch dashboard with two metrics.
        6. List current CloudWatch dashboards.
        7. Create a CloudWatch custom metric and add metric data.
        8. Add the custom metric to the dashboard.
        9. Create a CloudWatch alarm for the custom metric.
       10. Describe current CloudWatch alarms.
       11. Get recent data for the custom metric.
       12. Add data to the custom metric to trigger the alarm.
       13. Wait for an alarm state.
       14. Get history for the CloudWatch alarm.
       15. Add an anomaly detector.
       16. Describe current anomaly detectors.
       17. Get and display a metric image.
       18. Clean up resources.
    */
```

```
   private static ILogger logger = null!;
   private static CloudWatchWrapper _cloudWatchWrapper = null!;
   private static IConfiguration _configuration = null!;
   private static readonly List<string> _statTypes = new List<string>
{ "SampleCount", "Average", "Sum", "Minimum", "Maximum" };
   private static SingleMetricAnomalyDetector? anomalyDetector = null!;

   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
           services.AddAWSService<IAmazonCloudWatch>()
           .AddTransient<CloudWatchWrapper>()
       )
       .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();

       logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
           .CreateLogger<CloudWatchScenario>();

       _cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();

       Console.WriteLine(new string('-', 80));
       Console.WriteLine("Welcome to the Amazon CloudWatch example scenario.");
       Console.WriteLine(new string('-', 80));

       try
       {
           var selectedNamespace = await SelectNamespace();
```

```
            var selectedMetric = await SelectMetric(selectedNamespace);
            await GetAndDisplayMetricStatistics(selectedNamespace,
selectedMetric);
            await GetAndDisplayEstimatedBilling();
            await CreateDashboardWithMetrics();
            await ListDashboards();
            await CreateNewCustomMetric();
            await AddMetricToDashboard();
            await CreateMetricAlarm();
            await DescribeAlarms();
            await GetCustomMetricData();
            await AddMetricDataForAlarm();
            await CheckForMetricAlarm();
            await GetAlarmHistory();
            anomalyDetector = await AddAnomalyDetector();
            await DescribeAnomalyDetectors();
            await GetAndOpenMetricImage();
            await CleanupResources();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources();
        }

    }

    /// <summary>
    /// Select a namespace.
    /// </summary>
    /// <returns>The selected namespace.</returns>
    private static async Task<string> SelectNamespace()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. Select a CloudWatch Namespace from a list of
Namespaces.");
        var metrics = await _cloudWatchWrapper.ListMetrics();
        // Get a distinct list of namespaces.
        var namespaces = metrics.Select(m => m.Namespace).Distinct().ToList();
        for (int i = 0; i < namespaces.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {namespaces[i]}");
        }
```

```
        var namespaceChoiceNumber = 0;
        while (namespaceChoiceNumber < 1 || namespaceChoiceNumber >
namespaces.Count)
        {
            Console.WriteLine(
                "Select a namespace by entering a number from the preceding
list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out namespaceChoiceNumber);
        }

        var selectedNamespace = namespaces[namespaceChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedNamespace;
    }

    /// <summary>
    /// Select a metric from a namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <returns>The metric name.</returns>
    private static async Task<Metric> SelectMetric(string metricNamespace)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. Select a CloudWatch metric from a namespace.");

        var namespaceMetrics = await
_cloudWatchWrapper.ListMetrics(metricNamespace);

        for (int i = 0; i < namespaceMetrics.Count && i < 15; i++)
        {
            var dimensionsWithValues = namespaceMetrics[i].Dimensions
                .Where(d => !string.Equals("None", d.Value));
            Console.WriteLine($"\t{i + 1}. {namespaceMetrics[i].MetricName} " +
                              $"{string.Join(", :", dimensionsWithValues.Select(d
=> d.Value))}");
        }

        var metricChoiceNumber = 0;
        while (metricChoiceNumber < 1 || metricChoiceNumber >
namespaceMetrics.Count)
        {
```

```
            Console.WriteLine(
                "Select a metric by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out metricChoiceNumber);
        }

        var selectedMetric = namespaceMetrics[metricChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedMetric;
    }

    /// <summary>
    /// Get and display metric statistics for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayMetricStatistics(string
metricNamespace, Metric metric)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. Get CloudWatch metric statistics for the last
day.");

        for (int i = 0; i < _statTypes.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {_statTypes[i]}");
        }

        var statisticChoiceNumber = 0;
        while (statisticChoiceNumber < 1 || statisticChoiceNumber >
_statTypes.Count)
        {
            Console.WriteLine(
                "Select a metric statistic by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out statisticChoiceNumber);
        }

        var selectedStatistic = _statTypes[statisticChoiceNumber - 1];
        var statisticsList = new List<string> { selectedStatistic };
```

```
        var metricStatistics = await
_cloudWatchWrapper.GetMetricStatistics(metricNamespace, metric.MetricName,
statisticsList, metric.Dimensions, 1, 60);

        if (!metricStatistics.Any())
        {
            Console.WriteLine($"No {selectedStatistic} statistics found for
{metric} in namespace {metricNamespace}.");
        }

        metricStatistics = metricStatistics.OrderBy(s => s.Timestamp).ToList();
        for (int i = 0; i < metricStatistics.Count && i < 10; i++)
        {
            var metricStat = metricStatistics[i];
            var statValue =
metricStat.GetType().GetProperty(selectedStatistic)!.GetValue(metricStat, null);
            Console.WriteLine($"\t{i + 1}. Timestamp
{metricStatistics[i].Timestamp:G} {selectedStatistic}: {statValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get and display estimated billing statistics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayEstimatedBilling()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Get CloudWatch estimated billing for the last
week.");

        var billingStatistics = await SetupBillingStatistics();

        for (int i = 0; i < billingStatistics.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. Timestamp
{billingStatistics[i].Timestamp:G} : {billingStatistics[i].Maximum}");
        }
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Create a dashboard with metrics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CreateDashboardWithMetrics()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Create a new CloudWatch dashboard with metrics.");
        var dashboardName = _configuration["dashboardName"];
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);
        var newDashboardString = JsonSerializer.Serialize(
            newDashboard,
            new JsonSerializerOptions
            {
                DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull
            });
```

```
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} was created.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List dashboards.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDashboards()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. List the CloudWatch dashboards in the current
account.");

        var dashboards = await _cloudWatchWrapper.ListDashboards();

        for (int i = 0; i < dashboards.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {dashboards[i].DashboardName}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and add data for a new custom metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateNewCustomMetric()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Create and add data for a new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
```

```
        var customMetricName = _configuration["customMetricName"];

        var customData = await PutRandomMetricData(customMetricName,
customMetricNamespace);

        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
```

```
        return customData;
    }

    /// <summary>
    /// Add the custom metric to the dashboard.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricToDashboard()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Add the new custom metric to the dashboard.");

        var dashboardName = _configuration["dashboardName"];

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var validationMessages = await SetupDashboard(customMetricNamespace,
customMetricName, dashboardName);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} updated with metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
```

```
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Create a CloudWatch alarm for the new metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateMetricAlarm()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Create a CloudWatch alarm for the new metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var alarmName = _configuration["exampleAlarmName"];
        var accountId = _configuration["accountId"];
        var region = _configuration["region"];
        var emailTopic = _configuration["emailTopic"];
        var alarmActions = new List<string>();

        if (GetYesNoResponse(
                $"\tAdd an email action for topic {emailTopic} to alarm
{alarmName}? (y/n)"))
        {
            _cloudWatchWrapper.AddEmailAlarmAction(accountId, region, emailTopic,
alarmActions);
        }

        await _cloudWatchWrapper.PutMetricEmailAlarm(
            "Example metric alarm",
            alarmName,
            ComparisonOperator.GreaterThanOrEqualToThreshold,
            customMetricName,
            customMetricNamespace,
            100,
            alarmActions);

        Console.WriteLine($"\tAlarm {alarmName} added for metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Describe Alarms.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAlarms()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Describe CloudWatch alarms in the current
account.");
```

```
        var alarms = await _cloudWatchWrapper.DescribeAlarms();
        alarms = alarms.OrderByDescending(a => a.StateUpdatedTimestamp).ToList();

        for (int i = 0; i < alarms.Count && i < 10; i++)
        {
            var alarm = alarms[i];
            Console.WriteLine($"\t{i + 1}. {alarm.AlarmName}");
            Console.WriteLine($"\tState: {alarm.StateValue} for
{alarm.MetricName} {alarm.ComparisonOperator} {alarm.Threshold}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the recent data for the metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetCustomMetricData()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. Get current data for new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var accountId = _configuration["accountId"];

        var query = new List<MetricDataQuery>
        {
            new MetricDataQuery
            {
                AccountId = accountId,
                Id = "m1",
                Label = "Custom Metric Data",
                MetricStat = new MetricStat
                {
                    Metric = new Metric
                    {
                        MetricName = customMetricName,
                        Namespace = customMetricNamespace,
                    },
                    Period = 1,
                    Stat = "Maximum"
```

```
                }
            }
        };

        var metricData = await _cloudWatchWrapper.GetMetricData(
            20,
            true,
            DateTime.UtcNow.AddMinutes(1),
            20,
            query);

        for (int i = 0; i < metricData.Count; i++)
        {
            for (int j = 0; j < metricData[i].Values.Count; j++)
            {
                Console.WriteLine(
                    $"\tTimestamp {metricData[i].Timestamps[j]:G} Value:
{metricData[i].Values[j]}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add metric data to trigger an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricDataForAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"12. Add metric data to the custom metric to trigger
an alarm.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var nowUtc = DateTime.UtcNow;
        List<MetricDatum> customData = new List<MetricDatum>
        {
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-2)
```

```
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-1)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc
            }
        };
        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");
        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check for a metric alarm using the DescribeAlarmsForMetric action.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckForMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"13. Checking for an alarm state.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var hasAlarm = false;
        var retries = 10;
        while (!hasAlarm && retries > 0)
        {
            var alarms = await
_cloudWatchWrapper.DescribeAlarmsForMetric(customMetricNamespace,
customMetricName);
            hasAlarm = alarms.Any(a => a.StateValue == StateValue.ALARM);
            retries--;
            Thread.Sleep(20000);
```

```
        }

        Console.WriteLine(hasAlarm
            ? $"\tAlarm state found for {customMetricName}."
            : $"\tNo Alarm state found for {customMetricName} after 10
retries.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get history for an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAlarmHistory()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"14. Get alarm history.");

        var exampleAlarmName = _configuration["exampleAlarmName"];

        var alarmHistory = await
_cloudWatchWrapper.DescribeAlarmHistory(exampleAlarmName, 2);

        for (int i = 0; i < alarmHistory.Count; i++)
        {
            var history = alarmHistory[i];
            Console.WriteLine($"\t{i + 1}. {history.HistorySummary}, time
{history.Timestamp:g}");
        }
        if (!alarmHistory.Any())
        {
            Console.WriteLine($"\tNo alarm history data found for
{exampleAlarmName}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an anomaly detector.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<SingleMetricAnomalyDetector> AddAnomalyDetector()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"15. Add an anomaly detector.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detector = new SingleMetricAnomalyDetector
        {
            MetricName = customMetricName,
            Namespace = customMetricNamespace,
            Stat = "Maximum"
        };
        await _cloudWatchWrapper.PutAnomalyDetector(detector);
        Console.WriteLine($"\tAdded anomaly detector for metric
{customMetricName}.");

        Console.WriteLine(new string('-', 80));
        return detector;
    }

    /// <summary>
    /// Describe anomaly detectors.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAnomalyDetectors()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"16. Describe anomaly detectors in the current
account.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detectors = await
_cloudWatchWrapper.DescribeAnomalyDetectors(customMetricNamespace,
customMetricName);

        for (int i = 0; i < detectors.Count; i++)
        {
            var detector = detectors[i];
            Console.WriteLine($"\t{i + 1}.
{detector.SingleMetricAnomalyDetector.MetricName}, state
{detector.StateValue}");
```

```
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Fetch and open a metrics image for a CloudWatch metric and namespace.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAndOpenMetricImage()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("17. Get a metric image from CloudWatch.");

        Console.WriteLine($"\tGetting Image data for custom metric.");
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var memoryStream = await
_cloudWatchWrapper.GetTimeSeriesMetricImage(customMetricNamespace,
customMetricName, "Maximum", 10);
        var file = _cloudWatchWrapper.SaveMetricImage(memoryStream,
"MetricImages");

        ProcessStartInfo info = new ProcessStartInfo();

        Console.WriteLine($"\tFile saved as {Path.GetFileName(file)}.");
        Console.WriteLine($"\tPress enter to open the image.");
        Console.ReadLine();
        info.FileName = Path.Combine("ms-photos://", file);
        info.UseShellExecute = true;
        info.CreateNoWindow = true;
        info.Verb = string.Empty;

        Process.Start(info);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up created resources.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
```

```
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"18. Clean up resources.");

        var dashboardName = _configuration["dashboardName"];
        if (GetYesNoResponse($"\tDelete dashboard {dashboardName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting dashboard.");
            var dashboardList = new List<string> { dashboardName };
            await _cloudWatchWrapper.DeleteDashboards(dashboardList);
        }

        var alarmName = _configuration["exampleAlarmName"];
        if (GetYesNoResponse($"\tDelete alarm {alarmName}? (y/n)"))
        {
            Console.WriteLine($"\tCleaning up alarms.");
            var alarms = new List<string> { alarmName };
            await _cloudWatchWrapper.DeleteAlarms(alarms);
        }

        if (GetYesNoResponse($"\tDelete anomaly detector? (y/n)") &&
 anomalyDetector != null)
        {
            Console.WriteLine($"\tCleaning up anomaly detector.");

            await _cloudWatchWrapper.DeleteAnomalyDetector(
                anomalyDetector);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
```

```
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

시나리오에서 CloudWatch 작업에 대해 사용하는 래퍼 메서드입니다.

```
/// <summary>
/// Wrapper class for Amazon CloudWatch methods.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;
    private readonly ILogger<CloudWatchWrapper> _logger;

    /// <summary>
    /// Constructor for the CloudWatch wrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch,
 ILogger<CloudWatchWrapper> logger)

    {
        _logger = logger;
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
 listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
 DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
```

```
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });
```

```
        return metricStatistics.Datapoints;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages;
    }


    /// <summary>
    /// Get information on a dashboard.
    /// </summary>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A JSON object with dashboard information.</returns>
    public async Task<string> GetDashboard(string dashboardName)
    {
        var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
            new GetDashboardRequest()
            {
                DashboardName = dashboardName
            });

        return dashboardResponse.DashboardBody;
    }
```

```
    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
```

```
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }
```

```
    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
 ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        await foreach (var data in paginatedMetricData.MetricDataResults)
        {
            metricData.Add(data);
        }
        return metricData;
    }
```

```
   /// <summary>
   /// Add a metric alarm to send an email when the metric passes a threshold.
   /// </summary>
   /// <param name="alarmDescription">A description of the alarm.</param>
   /// <param name="alarmName">The name for the alarm.</param>
   /// <param name="comparison">The type of comparison to use.</param>
   /// <param name="metricName">The name of the metric for the alarm.</param>
   /// <param name="metricNamespace">The namespace of the metric.</param>
   /// <param name="threshold">The threshold value for the alarm.</param>
   /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
       string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
   {
       try
       {
           var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
               new PutMetricAlarmRequest()
               {
                   AlarmActions = alarmActions,
                   AlarmDescription = alarmDescription,
                   AlarmName = alarmName,
                   ComparisonOperator = comparison,
                   Threshold = threshold,
                   Namespace = metricNamespace,
                   MetricName = metricName,
                   EvaluationPeriods = 1,
                   Period = 10,
                   Statistic = new Statistic("Maximum"),
                   DatapointsToAlarm = 1,
                   TreatMissingData = "ignore"
               });
           return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
       }
       catch (LimitExceededException lex)
       {
           _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
has already been reached.");
       }
```

```
        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
 alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }

    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
 null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
 _amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
```

```
        }
        return alarms;
    }

    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms;
    }

    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
```

```
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }

    /// <summary>
    /// Delete a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAlarms(List<string> alarmNames)
    {
        var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
            new DeleteAlarmsRequest()
            {
                AlarmNames = alarmNames
            });

        return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
```

```
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }

    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
_amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });
```

```
        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
```

```
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import
 software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import
 software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import
 software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                  <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage> \s

                Where:
                  myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)\s
```

```
                costDateWeek - The start date to use to get AWS/Billinget
statistics. (For example, 2023-01-11T18:35:24.00Z.)\s
                dashboardName - The name of the dashboard to create.\s
                dashboardJson - The location of a JSON file to use to create a
dashboard. (See Readme file.)\s
                dashboardAdd - The location of a JSON file to use to update a
dashboard. (See Readme file.)\s
                settings - The location of a JSON file from which various
values are read. (See Readme file.)\s
                metricImage - The location of a BMP file that is used to create
a graph.\s
               """;

       if (args.length != 7) {
           System.out.println(usage);
           System.exit(1);
       }

       Region region = Region.US_EAST_1;
       String myDate = args[0];
       String costDateWeek = args[1];
       String dashboardName = args[2];
       String dashboardJson = args[3];
       String dashboardAdd = args[4];
       String settings = args[5];
       String metricImage = args[6];

       Double dataPoint = Double.parseDouble("10.0");
       Scanner sc = new Scanner(System.in);
       CloudWatchClient cw = CloudWatchClient.builder()
               .region(region)
               .credentialsProvider(ProfileCredentialsProvider.create())
               .build();

       System.out.println(DASHES);
       System.out.println("Welcome to the Amazon CloudWatch example scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println(
               "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
       ArrayList<String> list = listNameSpaces(cw);
       for (int z = 0; z < 5; z++) {
```

```
            int index = z + 1;
            System.out.println("    " + index + ". " + list.get(z));
        }

        String selectedNamespace = "";
        String selectedMetrics = "";
        int num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            selectedNamespace = list.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List available metrics within the selected
namespace and select one from the list.");
        ArrayList<String> metList = listMets(cw, selectedNamespace);
        for (int z = 0; z < 5; z++) {
            int index = z + 1;
            System.out.println("    " + index + ". " + metList.get(z));
        }
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            selectedMetrics = metList.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + selectedMetrics);
        Dimension myDimension = getSpecificMet(cw, selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get statistics for the selected metric over the
last day.");
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
```

```
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
        }
        System.out.println("Select a metric statistic by entering a number from
the preceding list:");
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + metricOption);
        getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get CloudWatch estimated billing for the last
week.");
        getMetricStatistics(cw, costDateWeek);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new CloudWatch dashboard with metrics.");
        createDashboardWithMetrics(cw, dashboardName, dashboardJson);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List dashboards using a paginator.");
        listDashboards(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a new custom metric by adding data to
it.");
        createNewCustomMetric(cw, dataPoint);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Add an additional metric to the dashboard.");
        addMetricToDashboard(cw, dashboardAdd, dashboardName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Create an alarm for the custom metric.");
        String alarmName = createAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Describe ten current alarms.");
        describeAlarms(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Get current data for new custom metric.");
        getCustomMetricData(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Push data into the custom metric to trigger the
alarm.");
        addMetricDataForAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        checkForMetricAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Get alarm history for the new alarm.");
        getAlarmHistory(cw, settings, myDate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Add an anomaly detector for the custom metric.");
        addAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Describe current anomaly detectors.");
        describeAnomalyDetectors(cw, settings);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is
complete.");
        System.out.println(DASHES);
        cw.close();
    }

    public static void deleteAnomalyDetector(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();
```

```
            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.deleteAlarms(request);
            System.out.println("Successfully deleted alarm " + alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteDashboard(CloudWatchClient cw, String dashboardName)
{
        try {
            DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                    .dashboardNames(dashboardName)
                    .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getAndOpenMetricImage(CloudWatchClient cw, String
fileName) {
```

```
        System.out.println("Getting Image data for custom metric.");
        try {
            String myJSON = "{\n" +
                    "  \"title\": \"Example Metric Graph\",\n" +
                    "  \"view\": \"timeSeries\",\n" +
                    "  \"stacked \": false,\n" +
                    "  \"period\": 10,\n" +
                    "  \"width\": 1400,\n" +
                    "  \"height\": 600,\n" +
                    "  \"metrics\": [\n" +
                    "    [\n" +
                    "      \"AWS/Billing\",\n" +
                    "      \"EstimatedCharges\",\n" +
                    "      \"Currency\",\n" +
                    "      \"USD\"\n" +
                    "    ]\n" +
                    "  ]\n" +
                    "}";

            GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
                    .metricWidget(myJSON)
                    .build();

            GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
            try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
```

```
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
            List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
            for (AnomalyDetector detector : anomalyDetectorList) {
                System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: " + detector.stateValue());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.putAnomalyDetector(anomalyDetectorRequest);
            System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getAlarmHistory(CloudWatchClient cw, String fileName,
String date) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String alarmName = rootNode.findValue("exampleAlarmName").asText();

            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
            DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

            DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
```

```
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
            if (historyItems.isEmpty()) {
                System.out.println("No alarm history data found for " + alarmName
+ ".");
            } else {
                for (AlarmHistoryItem item : historyItems) {
                    System.out.println("History summary: " +
item.historySummary());
                    System.out.println("Time stamp: " + item.timestamp());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkForMetricAlarm(CloudWatchClient cw, String fileName)
{
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            while (!hasAlarm && retries > 0) {
                DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
```

```
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for " + customMetricName
+ " after 10 retries.");
            else
                System.out.println("Alarm state found for " + customMetricName +
".");

        } catch (CloudWatchException | IOException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addMetricDataForAlarm(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set an Instant object.
            String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

            MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
```

```
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric " +
customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCustomMetricData(CloudWatchClient cw, String fileName)
{
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set the date.
            Instant nowDate = Instant.now();

            long hours = 1;
            long minutes = 30;
            Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                    ChronoUnit.MINUTES);
```

```
            Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(1)
                    .metric(met)
                    .build();

            MetricDataQuery dataQUery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

            List<MetricDataQuery> dq = new ArrayList<>();
            dq.add(dataQUery);

            GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(date2)
                    .metricDataQueries(dq)
                    .build();

            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAlarms(CloudWatchClient cw) {
        try {
```

```
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                    .alarmTypes(typeList)
                    .maxRecords(10)
                    .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm : alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " +
alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                    .alarmActions(alarmActions)
                    .alarmDescription("Example metric alarm")
```

```
                    .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                    .threshold(100.00)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .evaluationPeriods(1)
                    .period(10)
                    .statistic("Maximum")
                    .datapointsToAlarm(1)
                    .treatMissingData("ignore")
                    .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName + " was successfully created!");
            return alarmName;

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully updated.");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void createNewCustomMetric(CloudWatchClient cw, Double
dataPoint) {
        try {
```

```
            Dimension dimension = Dimension.builder()
                    .name("UNIQUE_PAGES")
                    .value("URLS")
                    .build();

            // Set an Instant object.
            String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName("PAGES_VISITED")
                    .unit(StandardUnit.NONE)
                    .value(dataPoint)
                    .timestamp(instant)
                    .dimensions(dimension)
                    .build();

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace("SITE/TRAFFIC")
                    .metricData(datum)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric
PAGES_VISITED");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                    .flatMap(r -> r.dashboardEntries().stream())
                    .forEach(entry -> {
                        System.out.println("Dashboard name is: " +
entry.dashboardName());
                        System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
                    });
```

```
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            PutDashboardResponse response = cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully created.");
            List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
            if (messages.isEmpty()) {
                System.out.println("There are no messages in the new Dashboard");
            } else {
                for (DashboardValidationMessage message : messages) {
                    System.out.println("Message is: " + message.message());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }

    public static void getMetricStatistics(CloudWatchClient cw, String
costDateWeek) {
        try {
            Instant start = Instant.parse(costDateWeek);
            Instant endDate = Instant.now();
            Dimension dimension = Dimension.builder()
                    .name("Currency")
```

```
                    .value("USD")
                    .build();

            List<Dimension> dimensionList = new ArrayList<>();
            dimensionList.add(dimension);
            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .metricName("EstimatedCharges")
                    .namespace("AWS/Billing")
                    .dimensions(dimensionList)
                    .statistics(Statistic.MAXIMUM)
                    .startTime(start)
                    .endTime(endDate)
                    .period(86400)
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
            String metricOption, String date, Dimension myDimension) {
        try {
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();

            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
```

```
                    .endTime(endDate)
                    .startTime(start)
                    .dimensions(myDimension)
                    .metricName(metVal)
                    .namespace(nameSpace)
                    .period(86400)
                    .statistics(Statistic.fromValue(metricOption))
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static Dimension getSpecificMet(CloudWatchClient cw, String namespace)
{
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsResponse response = cw.listMetrics(request);
            List<Metric> myList = response.metrics();
            Metric metric = myList.get(0);
            return metric.dimensions().get(0);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return null;
    }

    public static ArrayList<String> listMets(CloudWatchClient cw, String
namespace) {
        try {
            ArrayList<String> metList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> metList.add(metrics.metricName()));

            return metList;

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
        try {
            ArrayList<String> nameSpaceList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> {
                        String data = metrics.namespace();
                        if (!nameSpaceList.contains(data)) {
                            nameSpaceList.add(data);
                        }
                    });

            return nameSpaceList;
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return null;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
```

```
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

 This Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 2. List available metrics within the selected namespace.
 3. Get statistics for the selected metric over the last day.
 4. Get CloudWatch estimated billing for the last week.
 5. Create a new CloudWatch dashboard with metrics.
 6. List dashboards using a paginator.
 7. Create a new custom metric by adding data for it.
 8. Add the custom metric to the dashboard.
 9. Create an alarm for the custom metric.
 10. Describe current alarms.
 11. Get current data for the new custom metric.
 12. Push data into the custom metric to trigger the alarm.
 13. Check the alarm state using the action DescribeAlarmsForMetric.
 14. Get alarm history for the new alarm.
 15. Add an anomaly detector for the custom metric.
 16. Describe current anomaly detectors.
 17. Get a metric image for the custom metric.
 18. Clean up the Amazon CloudWatch resources.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)
```

```
            costDateWeek - The start date to use to get AWS Billing and Cost
Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a
dashboard. (See Readme file.)
            settings - The location of a JSON file from which various values are
read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")
```

```
    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select
one from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList!![num - 1]
    } else {
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
```

```
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)

    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)
```

```
    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action
DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
```

```
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val request = DeleteAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest = DeleteDashboardsRequest {
        dashboardNames = listOf(dashboardName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest = GetMetricWidgetImageRequest {
        metricWidget = myJSON
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest = DescribeAnomalyDetectorsRequest {
        maxResults = 10
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val anomalyDetectorRequest = PutAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
```

```
    }
}

suspend fun getAlarmHistory(fileName: String, date: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest = DescribeAlarmHistoryRequest {
        startDate = aws.smithy.kotlin.runtime.time.Instant(start)
        endDate = aws.smithy.kotlin.runtime.time.Instant(endDateVal)
        alarmName = alarmNameVal
        historyItemType = HistoryItemType.Action
    }

    CloudWatchClient { credentialsProvider = EnvironmentCredentialsProvider();
 region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10
```

```
    val metricRequest = DescribeAlarmsForMetricRequest {
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) println("No Alarm state found for $customMetricName after
 10 retries.") else println("Alarm state found for $customMetricName.")
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1001.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val datum2 = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1002.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
```

```
    }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request = PutMetricDataRequest {
        namespace = customMetricNamespace
        metricData = metricDataList
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(
        minutes,
        ChronoUnit.MINUTES
    )

    val met = Metric {
        metricName = customMetricName
        namespace = customMetricNamespace
    }

    val metStat = MetricStat {
        stat = "Maximum"
        period = 1
        metric = met
    }
```

```
    val dataQUery = MetricDataQuery {
        metricStat = metStat
        id = "foo2"
        returnData = true
    }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq = GetMetricDataRequest {
        maxDatapoints = 10
        scanBy = ScanBy.TimestampDescending
        startTime = aws.smithy.kotlin.runtime.time.Instant(nowDate)
        endTime = aws.smithy.kotlin.runtime.time.Instant(date2)
        metricDataQueries = dq
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest = DescribeAlarmsRequest {
        alarmTypes = typeList
        maxRecords = 10
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}

suspend fun createAlarm(fileName: String): String {
```

```
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest = PutMetricAlarmRequest {
        alarmActions = alarmActionObs
        alarmDescription = "Example metric alarm"
        alarmName = alarmNameVal
        comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
        threshold = 100.00
        metricName = customMetricName
        namespace = customMetricNamespace
        evaluationPeriods = 1
        period = 10
        statistic = Statistic.Maximum
        datapointsToAlarm = 1
        treatMissingData = "ignore"
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}

suspend fun addMetricToDashboard(fileNameVal: String, dashboardNameVal: String) {
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
```

```
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension = Dimension {
        name = "UNIQUE_PAGES"
        value = "URLS"
    }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = "PAGES_VISITED"
        unit = StandardUnit.None
        value = dataPoint
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
        dimensions = listOf(dimension)
    }

    val request = PutMetricDataRequest {
        namespace = "SITE/TRAFFIC"
        metricData = listOf(datum)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

```
suspend fun createDashboardWithMetrics(dashboardNameVal: String, fileNameVal:
 String) {
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}

fun readFileAsString(file: String): String {
    return String(Files.readAllBytes(Paths.get(file)))
}

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension = Dimension {
        name = "Currency"
        value = "USD"
    }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest = GetMetricStatisticsRequest {
        metricName = "EstimatedCharges"
        namespace = "AWS/Billing"
        dimensions = dimensionList
        statistics = listOf(Statistic.Maximum)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
```

```
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        period = 86400
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun getAndDisplayMetricStatistics(nameSpaceVal: String, metVal: String,
 metricOption: String, date: String, myDimension: Dimension) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest = GetMetricStatisticsRequest {
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
        dimensions = listOf(myDimension)
        metricName = metVal
        namespace = nameSpaceVal
        period = 86400
        statistics = listOf(Statistic.fromValue(metricOption))
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
```

```
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
```

```
            }
        }
    }
    return nameSpaceList
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 CloudWatch 지표 및 경보 관리

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- CloudWatch 지표를 볼 수 있는 경보를 생성합니다.
- 데이터를 지표에 입력하고 경보를 트리거합니다.
- 경보에서 데이터를 가져옵니다.
- 경보를 삭제합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

CloudWatch 작업을 래핑하는 클래스를 만듭니다.

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
 set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
```

```
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
```

```
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over
the
                             alarm threshold before the alarm is set into an
alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the
threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm


    def put_metric_data(self, namespace, name, value, unit):
```

```
        """
        Sends a single data value to CloudWatch for a metric. This metric is
given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
name)
            raise


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period
must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
value
                           or maximum value.
```

```
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
            raise
        else:
            return stats


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
metric_name)
        return alarm_iter


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
```

```
            metric = self.cloudwatch_resource.Metric(metric_namespace,
 metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

래퍼 클래스를 사용하여 데이터를 지표에 넣고, 지표를 감시하는 경보를 트리거하고, 경보에서 데이터를 가져옵니다.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    cw_wrapper = CloudWatchWrapper(boto3.resource("cloudwatch"))

    minutes = 20
    metric_namespace = "doc-example-metric"
    metric_name = "page_views"
    start = datetime.utcnow() - timedelta(minutes=minutes)
    print(
        f"Putting data into metric {metric_namespace}.{metric_name} spanning the
 "
        f"last {minutes} minutes."
    )
    for offset in range(0, minutes):
        stamp = start + timedelta(minutes=offset)
        cw_wrapper.put_metric_data_set(
            metric_namespace,
            metric_name,
```

```
            stamp,
            "Count",
            {
                "values": [
                    random.randint(bound, bound * 2)
                    for bound in range(offset + 1, offset + 11)
                ],
                "counts": [random.randint(1, offset + 1) for _ in range(10)],
            },
        )

    alarm_name = "high_page_views"
    period = 60
    eval_periods = 2
    print(f"Creating alarm {alarm_name} for metric {metric_name}.")
    alarm = cw_wrapper.create_metric_alarm(
        metric_namespace,
        metric_name,
        alarm_name,
        "Maximum",
        period,
        eval_periods,
        100,
        "GreaterThanThreshold",
    )
    print(f"Alarm ARN is {alarm.alarm_arn}.")
    print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Sending data to trigger the alarm. This requires data over the
threshold "
        f"for {eval_periods} periods of {period} seconds each."
    )
    while alarm.state_value == "INSUFFICIENT_DATA":
        print("Sending data for the metric.")
        cw_wrapper.put_metric_data(
            metric_namespace, metric_name, random.randint(100, 200), "Count"
        )
        alarm.load()
        print(f"Current alarm state is: {alarm.state_value}.")
        if alarm.state_value == "INSUFFICIENT_DATA":
            print(f"Waiting for {period} seconds...")
            time.sleep(period)
        else:
```

```
            print("Wait for a minute for eventual consistency of metric data.")
            time.sleep(period)
            if alarm.state_value == "OK":
                alarm.load()
                print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Getting data for metric {metric_namespace}.{metric_name} during
timespan "
        f"of {start} to {datetime.utcnow()} (times are UTC)."
    )
    stats = cw_wrapper.get_metric_statistics(
        metric_namespace,
        metric_name,
        start,
        datetime.utcnow(),
        60,
        ["Average", "Minimum", "Maximum"],
    )
    print(
        f"Got {len(stats['Datapoints'])} data points for metric "
        f"{metric_namespace}.{metric_name}."
    )
    pprint(sorted(stats["Datapoints"], key=lambda x: x["Timestamp"]))

    print(f"Getting alarms for metric {metric_name}.")
    alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
    for alarm in alarms:
        print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

    print(f"Deleting alarms for metric {metric_name}.")
    cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

    print("Thanks for watching!")
    print("-" * 88)
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - DeleteAlarms
  - DescribeAlarmsForMetric

- DisableAlarmActions
- EnableAlarmActions
- GetMetricStatistics
- ListMetrics
- PutMetricAlarm
- PutMetricData

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 CloudWatch에 대한 교차 서비스 예제

다음 샘플 애플리케이션에서는 AWS SDK를 사용하여 CloudWatch를 를 다른 AWS 서비스와 결합합니다. 각 예시에는 애플리케이션을 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 GitHub 링크가 포함되어 있습니다.

예제

- AWS SDK를 사용하여 Amazon DynamoDB의 성능 모니터링

AWS SDK를 사용하여 Amazon DynamoDB의 성능 모니터링

다음 코드 예제는 성능 모니터링을 위해 애플리케이션의 DynamoDB 사용을 구성하는 방법을 보여줍니다.

Java

SDK for Java 2.x

이 예제는 DynamoDB의 성능을 모니터링하도록 Java 애플리케이션을 구성하는 방법을 보여줍니다. 애플리케이션은 성능을 모니터링할 수 있는 CloudWatch로 지표 데이터를 전송합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예시를 참조하세요.

이 예시에서 사용되는 서비스

- CloudWatch
- DynamoDB

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK에서 CloudWatch 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon CloudWatch의 보안

AWS에서는 클라우드 보안을 가장 중요하게 생각합니다. 여러분은 AWS 고객으로서 보안에 민감한 기관의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와(과) 귀하의 공동 책임입니다. 공동 책임 모델은(는) 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드의 보안: AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS는 안전하게 사용할 수 있는 서비스 또한 제공합니다. 서드 파티 감사자는 AWS 규정 준수 프로그램의 일환으로 보안 효과를 정기적으로 테스트하고 검증합니다. CloudWatch에 적용되는 규정 준수 프로그램에 대한 자세한 내용을 알아보려면 규정 준수 프로그램 제공 AWS 범위 내 서비스를 참조하세요.
- 클라우드 내 보안: 귀하의 책임은 귀하가 사용하는 AWS 서비스에 의해 결정됩니다. 또한 여러분은 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다

이 설명서는 Amazon CloudWatch를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 그리고 보안 및 규정 준수 목적에 맞게 Amazon CloudWatch를 구성하는 방법을 보여 줍니다. 또한 CloudWatch 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

내용

Amazon CloudWatch의 데이터 보호

AWS 공동 책임 모델은 Amazon CloudWatch의 데이터 보호에 적용됩니다. 이 모델이 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 Data Privacy FAQ(데이터 프라이버시 FAQ)를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS Shared Responsibility Model and GDPR 블로그 게시물을 참조하세요.

데이터를 보호하려면 AWS 계정보안 인증 정보를 보호하고 AWS IAM Identity Center또는 AWS Identity and Access Management(IAM)를 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신하세요. TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정하세요.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용합니다.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 FIPS(Federal Information Processing Standard) 140-2를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름(Name) 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 CloudWatch 또는 기타 AWS 서비스 서비스에서 콘솔, API, AWS CLI 또는 AWS SDK를 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 보안 인증를 URL에 포함시켜서는 안 됩니다.

전송 중 암호화

CloudWatch는 전송 중인 데이터의 엔드 투 엔드 암호화를 사용합니다.

Amazon CloudWatch의 Identity and Access Management

AWS Identity and Access Management(IAM)는 관리자가 AWS리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 CloudWatch 리소스를 사용하도록 인증(로그인) 및 권한(권한 있음)을 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

고객

AWS Identity and Access Management(IAM)를 사용하는 방법은 CloudWatch에서 수행하는 작업에 따라 달라집니다.

서비스 사용자 - CloudWatch 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증 정보와 권한을 관리자가 제공합니다. 더 많은 CloudWatch 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. CloudWatch의 기능에 액세스할 수 없는 경우 Amazon CloudWatch 자격 증명 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 - 회사에서 CloudWatch 리소스를 담당하고 있는 경우 CloudWatch에 대한 전체 액세스 권한을 보유하고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 CloudWatch 기능과 리소스를 결정합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해합니다. 회사가 CloudWatch에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amazon CloudWatch와 함께 IAM을 사용하는 방법 섹션을 참조하세요.

IAM 관리자 - IAM 관리자는 CloudWatch에 대한 액세스 권한 관리 정책 작성 방법에 대해 자세히 알고 있는 것이 좋습니다. IAM에서 사용할 수 있는 CloudWatch 자격 증명 기반 정책 예시를 보려면 Amazon CloudWatch에 대한 자격 증명 기반 정책 예시 섹션을 참조하세요.

자격 증명을 통한 인증

인증은 ID 보안 인증을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자나 IAM 사용자로 또는 IAM 역할을 수임하여 인증(AWS에 로그인)되어야 합니다.

자격 증명 소스 AWS IAM Identity Center를 통해 제공된 보안 인증 정보를 사용하여 연동 ID로 AWS에 로그인할 수 있습니다. (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 보안 인증이 페더레이션형 ID의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 연동을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라 AWS Management Console 또는AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법을 참조하세요.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 보안 인증 정보를 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청에 서명을 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS은 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity

Center 사용 설명서의 다중 인증 및 IAM 사용자 설명서의 AWS에서 다중 인증(MFA) 사용을 참조합니다.

AWS 계정 루트 사용자

AWS 계정을 생성할 때는 해당 계정의 모든 AWS 서비스및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 자격 증명은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 태스크를 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 태스크의 전체 목록은 IAM 사용자 안내서의 루트 사용자 보안 인증이 필요한 태스크를 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 사용자가 보안 인증 공급자와의 페더레이션을 사용하여 임시 보안 인증 정보를 사용하여 AWS 서비스에 액세스하도록 요구합니다.

페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, AWS Directory Service, Identity Center 디렉터리의 사용자 또는 자격 증명 소스를 통해 제공된 보안 인증을 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션 보안 인증은 AWS 계정에 액세스할 때 역할을 수임하고 역할은 임시 보안 인증을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 AWS 계정및 애플리케이션에서 사용하기 위해 고유한 자격 증명 소스의 사용자 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWSAPI 태스크를 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

임시 보안 인증 정보가 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 아이덴티티에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연동 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 자격 증명 공급자의 역할 만들기를 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 자격 증명이 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 정책을 리소스에 직접 연결할 수 있습니다(역할을 프록시로 사용하는 대신). 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.
- 교차 서비스 액세스 – 일부 AWS 서비스는 다른 AWS 서비스의 기능을 사용합니다. 예컨대, 어떤 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서

완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용자 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조합니다.
- 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.

- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용자 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조합니다.

정책을 사용한 액세스 관리

정책을 생성하고 AWSID 또는 리소스에 연결하여 AWS내 액세스를 제어합니다. 정책은 ID 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole`태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI또는 AWSAPI에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 또는 AWS 서비스가 포함될 수 있습니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACLs)

액세스 제어 목록(ACLs)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는 지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3, AWS WAF 및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS는 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 보안 인증 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 엔터티의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용자 설명서의 IAM 엔터티에 대한 권한 경계를 참조합니다.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations는 기업이 소유하는 여러 개의 AWS 계정을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자을 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 연합된 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할 자격 증명 기반 정책의 교차 및 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용자 설명서의 세션 정책을 참조합니다.

여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon CloudWatch와 함께 IAM을 사용하는 방법

IAM을 사용하여 CloudWatch에 대한 액세스를 관리하기 전에 CloudWatch와 함께 사용할 수 있는 IAM 기능을 알아보세요.

Amazon CloudWatch에서 사용할 수 있는 IAM 기능

IAM 특성	CloudWatch 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACLs	아니요
ABAC(정책 내 태그)	부분
임시 보안 인증	예
보안 주체 권한	예
서비스 역할	예
서비스 연결 역할	아니요

CloudWatch 및 기타 AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서에서 AWS IAM으로 작업하는 서비스를 참조하세요.

CloudWatch에 대한 자격 증명 기반 정책

ID 기반 정책 지원	예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업와 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

CloudWatch에 대한 자격 증명 기반 정책 예시

CloudWatch 자격 증명 기반 정책 예시를 보려면 Amazon CloudWatch에 대한 자격 증명 기반 정책 예시 섹션을 참조하세요.

CloudWatch 내 리소스 기반 정책

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 또는 AWS 서비스가 포함될 수 있습니다.

계정 간 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 엔터티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 서로 다른 AWS 계정에 있는 경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에도 리소스 액세스 권한을 부여해야만 합니다. 개체에 자격 증명 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

CloudWatch 정책 작업

정책 작업 지원	예

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWSAPI 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

CloudWatch 작업 목록을 보려면 서비스 승인 참조에서 Amazon CloudWatch에서 정의한 작업을 참조하세요.

CloudWatch의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
cloudwatch
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "cloudwatch:action1",
      "cloudwatch:action2"
         ]
```

CloudWatch 자격 증명 기반 정책 예시를 보려면 Amazon CloudWatch에 대한 자격 증명 기반 정책 예시 섹션을 참조하세요.

CloudWatch 정책 리소스

정책 리소스 지원	예

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 타입을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

CloudWatch 리소스 유형 및 해당 ARN 목록을 보려면 서비스 승인 참조에서 Amazon CloudWatch에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon CloudWatch에서 정의한 작업을 참조하세요.

CloudWatch 자격 증명 기반 정책 예시를 보려면 Amazon CloudWatch에 대한 자격 증명 기반 정책 예시 섹션을 참조하세요.

CloudWatch 정책 조건 키

서비스별 정책 조건 키 지원	예

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition`요소를 지정하거나 단일 `Condition`요소에서 여러 키를 지정하는 경우 AWS는 논리적 `AND`태스크를 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS는 논리적 `OR`태스크를 사용하여 조건을 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용자 가이드의 AWS 전역 조건 컨텍스트 키를 참조하세요.

CloudWatch 조건 키 목록을 보려면 서비스 승인 참조의 Amazon CloudWatch에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon CloudWatch에서 정의한 작업을 참조하세요.

CloudWatch 자격 증명 기반 정책 예시를 보려면 Amazon CloudWatch에 대한 자격 증명 기반 정책 예시 섹션을 참조하세요.

CloudWatch의 ACL

ACL 지원	아니요

액세스 제어 목록(ACLs)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는 지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

CloudWatch를 사용한 ABAC

ABAC(정책 내 태그) 지원	부분

속성 기반 액세스 제어(ABAC)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그를 기반으로 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 타입에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 타입에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 정보는 IAM 사용 설명서의 ABAC란 무엇인가요?를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

CloudWatch에서 임시 보안 인증 정보 사용

임시 보안 인증 지원	예

일부 AWS 서비스는 임시 보안 인증을 사용하여 로그인할 때 작동하지 않습니다. 임시 보안 인증으로 작동하는 AWS 서비스를 비롯한 추가 정보는 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 참조하세요.

사용자 이름과 암호를 제외한 다른 방법을 사용하여 AWS Management Console에 로그인하면 임시 보안 인증을 사용하는 것입니다. 예를 들어 회사의 Single Sign-On(SSO) 링크를 사용하여 AWS에 액세스하면 해당 프로세스에서 자동으로 임시 보안 인증을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 정보는 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 보안 인증을 수동으로 만들 수 있습니다 그런 다음 이러한 임시 보안 인증을 사용하여 AWS에 액세스할 수 있습니다. AWS에서는 장기 액세스 키를 사용하는 대신 임시 보안 인증을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 인증 섹션을 참조하세요.

CloudWatch의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

CloudWatch에 대한 서비스 역할

서비스 역할 지원	예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용자 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조합니다.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 CloudWatch 기능이 중단될 수 있습니다. CloudWatch에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집하세요.

Amazon CloudWatch에 대한 자격 증명 기반 정책 예시

기본적으로 사용자 및 역할에는 CloudWatch 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 AWSAPI를 사용해 태스크를 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 포함하여 CloudWatch에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 Amazon CloudWatch에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- CloudWatch 콘솔 사용

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 CloudWatch 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS managed policies(관리형 정책) 또는 AWS managed policies for job functions(직무에 대한 관리형 정책)를 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 Policies and permissions in IAM(IAM의 정책 및 권한)을 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 CloudFormation와 같이, 특정 AWS 서비스를 통해 사용되는 경

우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.

- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하tpdy.
- 다중 인증(MFA) 필요 – AWS 계정 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 설정합니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 Configuring MFA-protected API access(MFA 보호 API 액세스 구성)를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

CloudWatch 콘솔 사용

Amazon CloudWatch 콘솔에 액세스하려면 최소한의 권한 세트가 있어야 합니다. 이러한 권한은 AWS 계정에서 CloudWatch 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다. 최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 엔터티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 CloudWatch 콘솔을 여전히 사용할 수 있도록 하려면 CloudWatch *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책을 엔터티에 추가합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

CloudWatch 콘솔에 필요한 권한

CloudWatch 콘솔로 작업하는 데 필요한 전체 권한 세트는 아래에 나와 있습니다. 이러한 권한은 CloudWatch 콘솔에 대한 전체 쓰기 및 읽기 권한을 제공합니다.

- application-autoscaling:DescribeScalingPolicies
- autoscaling:DescribeAutoScalingGroups
- autoscaling:DescribePolicies
- cloudtrail:DescribeTrails

- cloudwatch:DeleteAlarms
- cloudwatch:DescribeAlarmHistory
- cloudwatch:DescribeAlarms
- cloudwatch:GetMetricData
- cloudwatch:GetMetricStatistics
- cloudwatch:ListMetrics
- cloudwatch:PutMetricAlarm
- cloudwatch:PutMetricData
- ec2:DescribeInstances
- ec2:DescribeTags
- ec2:DescribeVolumes
- es:DescribeElasticsearchDomain
- es:ListDomainNames
- events:DeleteRule
- events:DescribeRule
- events:DisableRule
- events:EnableRule
- events:ListRules
- events:PutRule
- iam:AttachRolePolicy
- iam:CreateRole
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:ListAttachedRolePolicies
- iam:ListRoles
- kinesis:DescribeStream
- kinesis:ListStreams
- lambda:AddPermission
- lambda:CreateFunction

- lambda:GetFunctionConfiguration
- lambda:ListAliases
- lambda:ListFunctions
- lambda:ListVersionsByFunction
- lambda:RemovePermission
- logs:CancelExportTask
- logs:CreateExportTask
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:DeleteLogGroup
- logs:DeleteLogStream
- logs:DeleteMetricFilter
- logs:DeleteRetentionPolicy
- logs:DeleteSubscriptionFilter
- logs:DescribeExportTasks
- logs:DescribeLogGroups
- logs:DescribeLogStreams
- logs:DescribeMetricFilters
- logs:DescribeQueries
- logs:DescribeSubscriptionFilters
- logs:FilterLogEvents
- logs:GetLogGroupFields
- logs:GetLogRecord
- logs:GetLogEvents
- logs:GetQueryResults
- logs:PutMetricFilter
- logs:PutRetentionPolicy
- logs:PutSubscriptionFilter
- logs:StartQuery
- logs:StopQuery

- logs:TestMetricFilter
- s3:CreateBucket
- s3:ListBucket
- sns:CreateTopic
- sns:GetTopicAttributes
- sns:ListSubscriptions
- sns:ListTopics
- sns:SetTopicAttributes
- sns:Subscribe
- sns:Unsubscribe
- sqs:GetQueueAttributes
- sqs:GetQueueUrl
- sqs:ListQueues
- sqs:SetQueueAttributes
- swf:CreateAction
- swf:DescribeAction
- swf:ListActionTemplates
- swf:RegisterAction
- swf:RegisterDomain
- swf:UpdateAction

또한 X-Ray 트레이스 맵을 보려면 `AWSXrayReadOnlyAccess`가 필요합니다.

Amazon CloudWatch 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 CloudWatch 및 IAM 작업 시 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

CloudWatch에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojacksonIAM 사용자가 콘솔을 사용하여 가상 *my-example-widget*리소스에 대한 세부 정보를 보려고 하지만 가상 cloudwatch:*GetWidget*권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cloudwatch:GetWidget on resource: my-example-widget
```

이 경우 cloudwatch:*GetWidget*작업을 사용하여 *my-example-widget*리소스에 액세스할 수 있도록 mateojackson사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 CloudWatch에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예시 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 CloudWatch에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사람이 내 CloudWatch 리소스에 액세스하도록 허용하고 싶음

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제

어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- CloudWatch에서 이러한 기능의 지원 여부를 알아보려면 Amazon CloudWatch와 함께 IAM을 사용하는 방법 섹션을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하세요.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 연동)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

CloudWatch 대시보드 권한 업데이트

2018년 5월 1일에 AWS는 CloudWatch 대시보드에 액세스하는 데 필요한 권한을 변경했습니다. 이제 CloudWatch 콘솔에서 대시보드에 액세스하려면 대시보드 API 작업을 지원하기 위해 2017년에 도입한 다음과 같은 권한이 필요합니다.

- cloudwatch:GetDashboard
- cloudwatch:ListDashboards
- cloudwatch:PutDashboard
- cloudwatch:DeleteDashboards

CloudWatch 대시보드에 액세스하려면 다음 중 하나가 필요합니다.

- AdministratorAccess 정책
- CloudWatchFullAccess 정책
- 다음과 같은 특정 권한 중 하나 이상을 포함하는 사용자 지정 정책:
  - 대시보드를 볼 수 있는 `cloudwatch:GetDashboard` 및 `cloudwatch:ListDashboards`
  - 대시보드를 생성하거나 수정할 수 있는 `cloudwatch:PutDashboard`
  - 대시보드를 삭제할 수 있는 `cloudwatch:DeleteDashboards`

정책을 사용하여 IAM 사용자의 권한을 변경하는 방법에 대한 자세한 내용은 IAM 사용자의 권한 변경을 참조하세요.

CloudWatch 권한에 대한 자세한 내용은 Amazon CloudWatch 권한 참조 단원을 참조하세요.

대시보드 API 작업에 대한 자세한 내용은 Amazon CloudWatch API 참조의 PutDashboard를 참조하세요.

CloudWatch에 대한 AWS 관리형(미리 정의된) 정책

AWS는 AWS에서 생성하고 관리하는 독립형 IAM 정책을 제공하여 많은 일반 사용 사례를 처리합니다. 이러한 AWS 관리형 정책은 사용자가 필요한 권한을 조사할 필요가 없도록 일반 사용 사례에 필요한 권한을 부여합니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 단원을 참조하세요.

계정의 사용자에게 연결할 수 있는 다음 AWS 관리형 정책은 CloudWatch에만 적용됩니다.

주제

- CloudWatchFullAccessV2
- CloudWatchFullAccess
- CloudWatchReadOnlyAccess
- CloudWatchActionsEC2Access
- CloudWatchAutomaticDashboardsAccess
- CloudWatchAgentServerPolicy
- CloudWatchAgentAdminPolicy
- CloudWatch 크로스 계정 관측성에 대한 AWS 관리형 (미리 정의된) 정책
- CloudWatch Synthetics에 대한 AWS 관리형(미리 정의된) 정책
- Amazon CloudWatch RUM에 대한 AWS 관리형(미리 정의된) 정책
- CloudWatch Evidently에 대한 AWS 관리형(미리 정의된) 정책
- AWS Systems Manager Incident Manager에 대한 AWS 관리형 정책

CloudWatchFullAccessV2

최근 AWS에 CloudWatchFullAccessV2 관리형 IAM 정책이 추가되었습니다. 이 정책은 CloudWatch 작업 및 리소스에 대한 전체 액세스 권한을 부여하는 동시에 Amazon SNS 및 Amazon EC2 Auto Scaling과 같은 다른 서비스에 부여된 권한의 범위를 더 적절하게 지정합니다.

CloudWatchFullAccess를 사용하는 대신 이 정책을 사용하는 것이 좋습니다. AWS에서는 가까운 시일 내에 CloudWatchFullAccess를 사용 중단할 계획입니다.

여기에는 사용자가 CloudWatch 콘솔의 Application Signals 아래에서 모든 기능에 액세스할 수 있도록 `application-signals:` 권한이 포함됩니다. 여기에는 일부 `autoscaling:Describe` 권한이 포함됩니다. 따라서 이 정책이 적용되는 사용자가 CloudWatch 경보와 연결된 Auto Scaling 작업을 볼 수 있습니다. 여기에는 일부 `sns` 권한이 포함됩니다. 따라서 이 정책이 적용되는 사용자가 Amazon SNS 주제를 검색, 생성하여 CloudWatch 경보와 연결할 수 있습니다. 여기에는 IAM 권한이 포함되며, 이 정책이 적용되는 사용자가 CloudWatch와 연결된 서비스 연결 역할에 대한 정보를 볼 수 있습니다. 여기에는 `oam:ListSinks` 및 `oam:ListAttachedLinks` 권한이 포함됩니다. 따라서 이 정책이 적용되는 사용자가 콘솔을 사용하여 CloudWatch 교차 계정 관측성에서 소스 계정의 공유된 데이터를 볼 수 있습니다.

사용자가 CloudWatch 서비스에 포함된 CloudWatch Synthetics, AWS X-Ray 및 CloudWatch RUM에 대한 전체 액세스 권한을 가질 수 있도록 여기에는 `rum`, `synthetics` 및 `xray` 권한이 포함되어 있습니다.

CloudWatchFullAccessV2의 내용은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchFullAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DescribeScalingPolicies",
                "application-signals:*",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribePolicies",
                "cloudwatch:*",
                "logs:*",
                "sns:CreateTopic",
                "sns:ListSubscriptions",
                "sns:ListSubscriptionsByTopic",
                "sns:ListTopics",
                "sns:Subscribe",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "oam:ListSinks",
                "rum:*",
```

```
                "synthetics:*",
                "xray:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsServiceLinkedRolePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "application-
signals.cloudwatch.amazonaws.com"
                }
            }
        },
        {
            "Sid": "EventsServicePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/events.amazonaws.com/
AWSServiceRoleForCloudWatchEvents*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "events.amazonaws.com"
                }
            }
        },
         {
            "Sid": "OAMReadPermissions",
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

CloudWatchFullAccess

CloudWatchFullAccess 정책은 사용 중단될 예정입니다. 따라서 사용을 중단하고 대신 CloudWatchFullAccessV2를 사용하는 것이 좋습니다.

CloudWatchFullAccess의 내용은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:Describe*",
                "cloudwatch:*",
                "logs:*",
                "sns:*",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "oam:ListSinks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/events.amazonaws.com/
AWSServiceRoleForCloudWatchEvents*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "events.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
```

```
}
```

CloudWatchReadOnlyAccess

[CloudWatchReadOnlyAccess] 정책은 CloudWatch에 대한 읽기 전용 액세스 권한을 부여합니다.

정책에는 몇 가지 `logs:` 권한이 포함되어 있습니다. 따라서 이 정책이 적용되는 사용자가 콘솔을 사용하여 CloudWatch Logs 정보와 보고 CloudWatch Logs Insights 쿼리를 볼 수 있습니다. 이 정책이 적용되는 사용자가 CloudWatch 경보와 연결된 Auto Scaling 작업을 볼 수 있도록 `autoscaling:Describe*`가 포함되어 있습니다. 여기에는 사용자가 Application Signals를 사용하여 서비스 상태를 모니터링할 수 있도록 `application-signals:` 권한이 포함됩니다. 이 정책이 적용되는 사용자가 Application Auto Scaling 정책에 대한 정보에 액세스할 수 있도록 `application-autoscaling:DescribeScalingPolicies`가 포함되어 있습니다. 이 정책이 적용되는 사용자가 CloudWatch 경보에 대한 알림을 수신하는 Amazon SNS 주제에 대한 정보를 검색할 수 있도록 `sns:Get*` 및 `sns:List*`가 포함되어 있습니다. 이 정책이 적용되는 사용자가 콘솔을 사용하여 CloudWatch 크로스 계정 관측성에서 소스 계정의 공유된 데이터를 볼 수 있도록 `oam:ListSinks` 및 `oam:ListAttachedLinks` 권한이 포함되어 있습니다. 여기에는 사용자가 CloudWatch Application Signals가 설정되었는지 확인할 수 있도록 `iam:GetRole` 권한이 포함됩니다.

사용자가 CloudWatch 서비스에 포함된 CloudWatch Synthetics, AWS X-Ray 및 CloudWatch RUM에 대한 읽기 전용 액세스 권한을 가질 수 있도록 여기에는 `rum`, `synthetics` 및 `xray` 권한이 포함되어 있습니다.

다음은 CloudWatchReadOnlyAccess 정책의 내용입니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchReadOnlyAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DescribeScalingPolicies",
                "application-signals:BatchGet*",
                "application-signals:Get*",
                "application-signals:List*",
                "autoscaling:Describe*",
                "cloudwatch:BatchGet*",
                "cloudwatch:Describe*",
                "cloudwatch:GenerateQuery",
                "cloudwatch:Get*",
```

```
                "cloudwatch:List*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:Describe*",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "logs:StartLiveTail",
                "logs:StopLiveTail",
                "oam:ListSinks",
                "sns:Get*",
                "sns:List*",
                "rum:BatchGet*",
                "rum:Get*",
                "rum:List*",
                "synthetics:Describe*",
                "synthetics:Get*",
                "synthetics:List*",
                "xray:BatchGet*",
                "xray:Get*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "OAMReadPermissions",
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        },
        {
            "Sid": "CloudWatchReadOnlyGetRolePermissions",
            "Effect": "Allow",
            "Action": "iam:GetRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals"
        }
    ]
}
```

CloudWatchActionsEC2Access

[CloudWatchActionsEC2Access] 정책은 Amazon EC2 메타데이터 외에도 CloudWatch 경보 및 지표에 대한 읽기 전용 액세스 권한을 부여합니다. 또한 EC2 인스턴스의 중지, 종료 및 재부팅 API 작업에 대한 액세스 권한도 부여합니다.

다음은 CloudWatchActionsEC2Access 정책의 내용입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:Describe*",
        "ec2:Describe*",
        "ec2:RebootInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": "*"
    }
  ]
}
```

CloudWatchAutomaticDashboardsAccess

[CloudWatch-CrossAccountAccess] 관리형 정책은 [CloudWatch-CrossAccountSharingRole] IAM 역할에 의해 사용됩니다. 이 역할 및 정책을 통해 교차 계정 대시보드 사용자가 대시보드를 공유하는 각 계정의 자동 대시보드를 볼 수 있습니다.

다음은 CloudWatchAutomaticDashboardsAccess 정책의 내용입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "cloudfront:GetDistribution",
        "cloudfront:ListDistributions",
        "dynamodb:DescribeTable",
        "dynamodb:ListTables",
```

```
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "ecs:DescribeClusters",
        "ecs:DescribeContainerInstances",
        "ecs:ListClusters",
        "ecs:ListContainerInstances",
        "ecs:ListServices",
        "elasticache:DescribeCacheClusters",
        "elasticbeanstalk:DescribeEnvironments",
        "elasticfilesystem:DescribeFileSystems",
        "elasticloadbalancing:DescribeLoadBalancers",
        "kinesis:DescribeStream",
        "kinesis:ListStreams",
        "lambda:GetFunction",
        "lambda:ListFunctions",
        "rds:DescribeDBClusters",
        "rds:DescribeDBInstances",
        "resource-groups:ListGroupResources",
        "resource-groups:ListGroups",
        "route53:GetHealthCheck",
        "route53:ListHealthChecks",
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "sns:ListTopics",
        "sqs:GetQueueAttributes",
        "sqs:GetQueueUrl",
        "sqs:ListQueues",
        "synthetics:DescribeCanariesLastRun",
        "tag:GetResources"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "apigateway:GET"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:apigateway:*::/restapis*"
      ]
    }
  ]
```

CloudWatchAgentServerPolicy

[CloudWatchAgentServerPolicy] 정책은 Amazon EC2 인스턴스에 연결된 IAM 역할에서 사용되어 CloudWatch 에이전트가 정보를 인스턴스에서 읽고 CloudWatch에 쓸 수 있게 합니다. 그 내용은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CWACloudWatchServerPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "ec2:DescribeVolumes",
                "ec2:DescribeTags",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CWASSMServerPermissions",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-*"
        }
    ]
}
```

CloudWatchAgentAdminPolicy

[CloudWatchAgentAdminPolicy] 정책은 Amazon EC2 인스턴스에 연결된 IAM 역할에서 사용할 수 있습니다. 이 정책을 통해 CloudWatch 에이전트가 정보를 인스턴스에서 읽고 CloudWatch에 쓸 수 있으며 파라미터 스토어에도 쓸 수 있습니다. 그 내용은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CWACloudWatchPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "ec2:DescribeTags",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CWASSMPermissions",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter",
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-*"
        }
    ]
}
```

> **Note**
>
> IAM 콘솔에 로그인하고 이 콘솔에서 특정 정책을 검색하여 이러한 권한 정책을 검토할 수 있습니다.

또한 CloudWatch 작업 및 리소스에 대한 권한을 허용하는 자체 사용자 지정 IAM 정책을 생성할 수도 있습니다. 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다.

CloudWatch 크로스 계정 관측성에 대한 AWS 관리형 (미리 정의된) 정책

이 섹션의 정책은 CloudWatch 크로스 계정 관측성과 관련된 권한을 부여합니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.

CloudWatchCrossAccountSharingConfiguration

CloudWatchCrossAccountSharingConfiguration 정책은 계정 간에 CloudWatch 리소스를 공유하기 위해 Observability Access Manager 링크를 생성 및 관리하고 볼 수 있는 액세스 권한을 부여합니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오. 내용은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:Link",
                "oam:ListLinks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        }
    ]
}
```

OAMFullAccess

OAMFullAccess 정책은 CloudWatch 크로스 계정 관측성에 사용되는 Observability Access Manager 싱크 및 링크를 생성 및 관리하고 볼 수 있는 액세스 권한을 부여합니다.

OAMFullAccess 정책 자체는 링크 간 관측성 데이터를 공유를 허용하지 않습니다. CloudWatch 지표를 공유하기 위한 링크를 생성하려면 CloudWatchFullAccess 또는 CloudWatchCrossAccountSharingConfiguration이 필요합니다. CloudWatch Logs 로그 그룹을 공유하기 위한 링크를 생성하려면 CloudWatchLogsFullAccess 또는 CloudWatchLogsCrossAccountSharingConfiguration이 필요합니다. X-Ray 추적을 공유하는 링크를 생성하려면 AWSXRayFullAccess 또는 AWSXRayCrossAccountSharingConfiguration이 필요합니다.

자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오. 내용은 다음과 같습니다.

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

OAMReadOnlyAccess

OAMReadOnlyAccess 정책은 CloudWatch 크로스 계정 관측성에 사용되는 Observability Access Manager 리소스에 대한 읽기 전용 액세스 권한을 부여합니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오. 내용은 다음과 같습니다.

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

CloudWatch Synthetics에 대한 AWS 관리형(미리 정의된) 정책

[CloudWatchSyntheticsFullAccess] 및 [CloudWatchSyntheticsReadOnlyAccess] AWS 관리형 정책은 CloudWatch Synthetics를 관리하거나 사용할 사용자에게 할당할 수 있습니다. 다음과 같은 추가 정책도 관련이 있습니다.

- [AmazonS3ReadOnlyAccess] 및 [CloudWatchReadOnlyAccess] – 이러한 정책은 CloudWatch 콘솔에서 모든 Synthetics 데이터를 읽는 데 필요합니다.
- [AWSLambdaReadOnlyAccess] – canary가 사용한 소스 코드를 볼 수 있습니다.
- CloudWatchSyntheticsFullAccess - canary를 생성할 수 있습니다. 또한 새 IAM 역할을 생성할 canary를 생성하고 삭제하려면 다음 인라인 정책 문도 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:CreatePolicy",
```

```
                "iam:DeletePolicy",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
                "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
            ]
        }
    ]
}
```

> **Important**
>
> 사용자에게 `iam:CreateRole`, `iam:DeleteRole`, `iam:CreatePolicy`, `iam:DeletePolicy`, `iam:AttachRolePolicy`, 및 `iam:DetachRolePolicy` 권한을 부여하면 해당 사용자에게 `arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*` 및 `arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*`과 일치하는 ARN이 있는 역할 및 정책을 생성, 연결 및 삭제할 수 있는 전체 관리 액세스 권한이 부여됩니다. 예를 들어 이러한 권한을 가진 사용자는 모든 리소스에 대한 전체 권한을 가진 정책을 생성하고 해당 정책을 ARN 패턴과 일치하는 모든 역할에 연결할 수 있습니다. 이러한 권한을 부여한 사람에게 매우 주의해야 합니다.

정책 연결 및 사용자에게 권한 부여에 대한 자세한 내용은 IAM 사용자의 권한 변경 및 사용자 또는 역할의 인라인 정책을 포함하려면을 참조하세요.

CloudWatchSyntheticsFullAccess

다음은 [CloudWatchSyntheticsFullAccess] 정책의 내용입니다.

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

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutEncryptionConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::cw-syn-results-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "xray:GetTraceSummaries",
                "xray:BatchGetTraces",
                "apigateway:GET"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::cw-syn-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::aws-synthetics-library-*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lambda.amazonaws.com",
                        "synthetics.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:Synthetics-*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:CreateFunction",
                "lambda:AddPermission",
                "lambda:PublishVersion",
                "lambda:UpdateFunctionCode",
                "lambda:UpdateFunctionConfiguration",
                "lambda:GetFunctionConfiguration",
                "lambda:DeleteFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:cwsyn-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:GetLayerVersion",
                "lambda:PublishLayerVersion",
                "lambda:DeleteLayerVersion"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:layer:cwsyn-*",
                "arn:aws:lambda:*:*:layer:Synthetics:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
```

```
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "arn:*:sns:*:*:Synthetics-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:*:*:key/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:*:*:key/*",
```

```
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "s3.*.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

CloudWatchSyntheticsReadOnlyAccess

다음은 [CloudWatchSyntheticsReadOnlyAccess] 정책의 내용입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:Describe*",
                "synthetics:Get*",
                "synthetics:List*",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon CloudWatch RUM에 대한 AWS 관리형(미리 정의된) 정책

AmazonCloudWatchRUMFullAccess 및 AmazonCloudWatchRUMReadOnlyAccess AWS 관리형 정책은 CloudWatch RUM을 관리하거나 사용할 사용자에게 배정할 수 있습니다.

AmazonCloudWatchRUMFullAccess

다음은 AmazonCloudWatchRUMFullAccess 정책의 내용입니다.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "rum:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/rum.amazonaws.com/
AWSServiceRoleForRealUserMonitoring"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/RUM-Monitor*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "cognito-identity.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "arn:aws:cloudwatch:*:*:alarm:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cognito-identity:CreateIdentityPool",
                "cognito-identity:ListIdentityPools",
                "cognito-identity:DescribeIdentityPool",
                "cognito-identity:GetIdentityPoolRoles",
                "cognito-identity:SetIdentityPoolRoles"
            ],
            "Resource": "arn:aws:cognito-identity:*:*:identitypool/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy",
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*RUMService*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:DescribeResourcePolicies"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
```

```
            "Resource": "arn:aws:logs:*:*:log-group::log-stream:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:describeCanaries",
                "synthetics:describeCanariesLastRun"
            ],
            "Resource": "arn:aws:synthetics:*:*:canary:*"
        }
    ]
}
```

AmazonCloudWatchRUMReadOnlyAccess

다음은 AmazonCloudWatchRUMReadOnlyAccess 정책의 내용입니다.

```
{

 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "rum:GetAppMonitor",
    "rum:GetAppMonitorData",
    "rum:ListAppMonitors",
    "rum:ListRumMetricsDestinations",
    "rum:BatchGetRumMetricDefinitions"
   ],
   "Resource": "*"
  }
 ]
}
```

AmazonCloudWatchRUMServiceRolePolicy

AmazonCloudWatchRUMServiceRolePolicy를 IAM 엔터티에 연결할 수 없습니다. 이 정책은 CloudWatch RUM이 모니터링 데이터를 다른 관련 AWS 서비스에 게시하도록 허용하는 서비스 연결 역할에 연결됩니다. 이 서비스 연결 역할에 대한 자세한 내용은 CloudWatch RUM에 서비스 연결 역할 사용 섹션을 참조하세요.

AmazonCloudWatchRUMServiceRolePolicy의 전체 내용은 다음과 같습니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "xray:PutTraceSegments"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringLike": {
     "cloudwatch:namespace": [
      "RUM/CustomMetrics/*",
      "AWS/RUM"
     ]
    }
   }
  }
 ]
}
```

CloudWatch Evidently에 대한 AWS 관리형(미리 정의된) 정책

CloudWatchEvidentlyFullAccess 및 CloudWatchEvidentlyReadOnlyAccess AWS 관리형 정책은 CloudWatch Evidently를 관리하거나 사용할 사용자에게 배정할 수 있습니다.

CloudWatchEvidentlyFullAccess

다음은 CloudWatchEvidentlyFullAccess 정책의 내용입니다.

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

```
                "evidently:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchRUMEvidentlyRole-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms",
                "cloudwatch:TagResource",
```

```
                "cloudwatch:UnTagResource"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudtrail:LookupEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:Evidently-Alarm-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "arn:*:sns:*:*:Evidently-*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

CloudWatchEvidentlyReadOnlyAccess(새 정책)

다음은 CloudWatchEvidentlyReadOnlyAccess 정책의 내용입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "evidently:GetExperiment",
                "evidently:GetFeature",
                "evidently:GetLaunch",
                "evidently:GetProject",
                "evidently:GetSegment",
                "evidently:ListExperiments",
                "evidently:ListFeatures",
                "evidently:ListLaunches",
                "evidently:ListProjects",
                "evidently:ListSegments",
                "evidently:ListSegmentReferencs"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Systems Manager Incident Manager에 대한 AWS 관리형 정책

[AWSCloudWatchAlarms_ActionSSMIncidentsServiceRolePolicy] 정책은 CloudWatch가 사용자를 대신하여 AWS Systems Manager Incident Manager에서 인시던트를 시작할 수 있게 허용하는 서비스 연결 역할에 연결됩니다. 자세한 내용은 CloudWatch 경보 Systems Manager Incident Manager 작업에 대한 서비스 연결 역할 권한 단원을 참조하세요.

정책에 다음 권한이 있습니다.

- ssm-incidents:StartIncident

고객 관리형 정책 예

이 단원에서는 다양한 CloudWatch 작업에 대한 권한을 부여하는 사용자 정책의 예를 확인할 수 있습니다. 이러한 정책은 CloudWatch API, AWS SDK 또는 AWS CLI를 사용하는 경우에 유효합니다.

예제

- 예 1: 사용자에게 CloudWatch에 대한 전체 액세스 허용
- 예 2: CloudWatch에 대한 읽기 전용 액세스 허용
- 예 3: Amazon EC2 인스턴스 중지 또는 종료

예 1: 사용자에게 CloudWatch에 대한 전체 액세스 허용

사용자에게 CloudWatch에 대한 전체 액세스 권한을 부여하려면 고객 관리형 정책을 생성하는 대신 사용자에게 [CloudWatchFullAccess] 관리형 정책을 부여하면 됩니다. [CloudWatchFullAccess]의 내용은 CloudWatchFullAccess 단원에 나와 있습니다.

예 2: CloudWatch에 대한 읽기 전용 액세스 허용

다음 정책은 사용자에게 CloudWatch에 대한 읽기 전용 액세스를 허용하고 Amazon EC2 Auto Scaling 작업, CloudWatch 지표, CloudWatch Logs 데이터, 경보 관련 Amazon SNS 데이터를 볼 수 있도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
        "cloudwatch:Describe*",
        "cloudwatch:Get*",
        "cloudwatch:List*",
        "logs:Get*",
        "logs:Describe*",
        "logs:StartQuery",
        "logs:StopQuery",
```

```
        "logs:TestMetricFilter",
        "logs:FilterLogEvents",
        "logs:StartLiveTail",
        "logs:StopLiveTail",
        "sns:Get*",
        "sns:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

예 3: Amazon EC2 인스턴스 중지 또는 종료

다음 정책은 CloudWatch 경보 작업이 EC2 인스턴스를 중지하거나 종료하도록 허용합니다. 아래 샘플에서 GetMetricData, ListMetrics 및 DescribeAlarms 작업은 선택 사항입니다. 인스턴스를 제대로 중지 또는 종료하려면 이러한 작업을 포함하는 것이 좋습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:DescribeAlarms"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": [
        "*"
```

```
      ],
      "Effect": "Allow"
    }
  ]
}
```

AWS 관리형 정책에 대한 CloudWatch 업데이트

아래에는 CloudWatch의 AWS 관리형 정책 업데이트에 관한 세부 정보가 나와 있습니다. 이 서비스가 해당 변경 사항을 추적하기 시작한 이후부터 설명되어 있습니다. 이 페이지의 변경 사항에 관한 자동 알림을 받으려면 CloudWatch 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
CloudWatchFullAccessV2 – 기존 정책에 대한 업데이트	CloudWatch에서 CloudWatchFullAccessV2 정책이 업데이트되었습니다. 사용자가 CloudWatch Application Signals를 사용하여 서비스 상태와 관련된 문제를 보고, 조사하며, 진단할 수 있도록 `application-signals:*` 를 추가하기 위해 `CloudWatchFullAccessPermissions` 정책 범위가 업데이트되었습니다.	2024년 5월 20일
CloudWatchReadOnlyAccess – 기존 정책에 대한 업데이트	CloudWatch에서 CloudWatchReadOnlyAccess 정책이 업데이트되었습니다. 사용자가 CloudWatch Application Signals를 사용하여 서비스 상태와 관련된 문제를 보고, 조사하며, 진단할 수 있도록 `application-signals:BatchGet*` ,	2024년 5월 20일

변경 사항	설명	날짜
	application-signals:List* , application-signals:Get* 을 추가하기 위해 CloudWatchReadOnlyAccessPermissions 정책 범위가 업데이트되었습니다. CloudWatch Application Signals가 설정되어 있는지 사용자가 확인할 수 있도록 iam:GetRole 작업을 추가하기 위해 CloudWatchReadOnlyGetRolePermissions 범위가 업데이트되었습니다.	
CloudWatchApplicationSignalsServiceRolePolicy – 기존 정책 업데이트	CloudWatch에서 CloudWatchApplicationSignalsServiceRolePolicy 정책이 업데이트되었습니다. 더 많은 아키텍처에서 Application Signals를 활성화하도록 logs:StartQuery 및 logs:GetQueryResults 권한의 범위가 변경되어 arn:aws:logs:*:*:log-group:/aws/appsignals/*:* 및 arn:aws:logs:*:*:log-group:/aws/application-signals/data:* ARN이 추가되었습니다.	2024년 4월 18일

변경 사항	설명	날짜
CloudWatchApplicationSignalsServiceRolePolicy – 기존 정책 업데이트	CloudWatch에서 CloudWatchApplicationSignalsServiceRolePolicy의 권한 범위가 변경되었습니다. Application Signals가 연결된 계정의 소스에서 지표를 검색할 수 있도록 `cloudwatch:GetMetricData` 권한 범위가 *로 변경되었습니다.	2024년 4월 8일
CloudWatchAgentServerPolicy - 기존 정책 업데이트	CloudWatch가 CloudWatchAgentServerPolicy에 대한 권한을 추가했습니다. `xray:PutTraceSegments` , `xray:PutTelemetryRecords` , `xray:GetSamplingRules` , `xray:GetSamplingTargets` , `xray:GetSamplingStatisticSummaries` 및 `logs:PutRetentionPolicy` 권한이 추가되어 CloudWatch 에이전트가 X-Ray 트레이스를 게시하고 로그 그룹 보존 기간을 수정할 수 있습니다.	2024년 2월 12일

변경 사항	설명	날짜
CloudWatchAgentAdminPolicy – 기존 정책 업데이트	CloudWatch가 CloudWatchAgentAdminPolicy에 대한 권한을 추가했습니다. `xray:PutTraceSegments` , `xray:PutTelemetryRecords` , `xray:GetSamplingRules` , `xray:GetSamplingTargets` , `xray:GetSamplingStatisticSummaries` 및 `logs:PutRetentionPolicy` 권한이 추가되어 CloudWatch 에이전트가 X-Ray 트레이스를 게시하고 로그 그룹 보존 기간을 수정할 수 있습니다.	2024년 2월 12일

변경 사항	설명	날짜
CloudWatchFullAccessV2 – 기존 정책에 대한 업데이트	CloudWatch는 CloudWatchFullAccessV2에 권한을 추가했습니다. 이 정책이 적용되는 사용자가 CloudWatch Application Signals를 관리할 수 있도록 CloudWatch Synthetics, X-Ray 및 CloudWatch RUM 작업에 대한 기존 권한과 CloudWatch Application Signals에 대한 새 권한이 추가되었습니다. CloudWatch Application Signals가 로그, 지표, 트레이스 및 태그에서 텔레메트리 데이터를 검색할 수 있도록 CloudWatch Application Signals 서비스 연결 역할을 생성할 수 있는 권한이 추가되었습니다.	2023년 12월 5일

변경 사항	설명	날짜
CloudWatchReadOnlyAccess – 기존 정책에 대한 업데이트	CloudWatch에서는 CloudWatchReadOnlyAccess에 권한을 추가했습니다. 이 정책이 적용되는 사용자가 CloudWatch Application Signals에서 보고하는 서비스 상태 문제를 분류하고 진단할 수 있도록 CloudWatch Synthetics, X-Ray, CloudWatch RUM 작업에 대한 기존 읽기 전용 권한과 CloudWatch Application Signals에 대한 새로운 읽기 전용 권한이 추가되었습니다. 이 정책이 적용되는 사용자가 자연어 프롬프트에서 CloudWatch Metrics Insights 쿼리 문자열을 생성할 수 있도록 `cloudwatch:GenerateQuery` 권한이 추가되었습니다.	2023년 12월 5일

변경 사항	설명	날짜
CloudWatchApplicationSignalsServiceRolePolicy – 새 정책	CloudWatch에 새 정책 CloudWatchApplicationSignalsServiceRolePolicy가 추가되었습니다. CloudWatchApplicationSignalsServiceRolePolicy은 예정된 기능에 CloudWatch Logs 데이터, X-Ray 추적 데이터, CloudWatch 지표 데이터, 태깅 데이터를 수집할 수 있는 권한을 부여합니다.	2023년 11월 9일
AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy – 새 정책	CloudWatch에 새 정책 AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy가 추가되었습니다. AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy는 사용자를 대신하여 데이터베이스에서 성능 개선 도우미 지표를 가져올 수 있는 권한을 CloudWatch에 부여합니다.	2023년 9월 20일

변경 사항	설명	날짜
CloudWatchReadOnlyAccess – 기존 정책에 대한 업데이트	CloudWatch Logs는 CloudWatchReadOnly Access에 대한 권한을 추가했습니다. 이 정책이 적용되는 사용자가 Application Auto Scaling 정책에 대한 정보에 액세스할 수 있도록 `application-autoscaling:DescribeScalingPolicies` 권한이 추가되었습니다.	2023년 9월 14일
CloudWatchFullAccessV2 – 새 정책	CloudWatch에 새 정책 CloudWatchFullAccessV2가 추가되었습니다. CloudWatchFullAccessV2는 CloudWatch 작업 및 리소스에 대한 전체 액세스 권한을 부여하는 동시에 Amazon SNS 및 Amazon EC2 Auto Scaling과 같은 다른 서비스에 부여된 권한의 범위를 더 효과적으로 지정합니다. 자세한 내용은 CloudWatchFullAccessV2를 참조하세요.	2023년 8월 1일

변경 사항	설명	날짜
AWSServiceRoleForInternetMonitor – 기존 정책 업데이트	Amazon CloudWatch Internet Monitor에 Network Load Balancer 리소스를 모니터링할 수 있는 새로운 권한이 추가되었습니다. Internet Monitor가 NLB 리소스에 대한 흐름 로그를 분석하여 고객의 Network Load Balancer 트래픽을 모니터링하려면 `elasticloadbalancing:DescribeLoadBalancers` 및 `ec2:DescribeNetworkInterfaces` 권한이 필요합니다. 자세한 내용은 Amazon CloudWatch Internet Monitor 사용 단원을 참조하십시오.	2023년 7월 15일
CloudWatchReadOnlyAccess – 기존 정책에 대한 업데이트	CloudWatch에서는 CloudWatchReadOnlyAccess에 권한을 추가했습니다. 이 정책이 있는 사용자가 콘솔을 사용하여 CloudWatch Logs Live Tail 세션을 시작하고 중지할 수 있도록 `logs:StartLiveTail` 및 `logs:StopLiveTail` 권한이 추가되었습니다. 자세한 내용은 Live Tail을 사용하여 거의 실시간으로 로그 보기를 참조하세요.	2023년 6월 6일

변경 사항	설명	날짜
CloudWatchCrossAccountSharingConfiguration — 새 정책	CloudWatch에서는 CloudWatch 지표를 공유하는 CloudWatch 크로스 계정 관측성 링크를 관리할 수 있는 새로운 정책을 추가했습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.	2022년 11월 27일
OAMFullAccess – 새 정책	CloudWatch에서는 CloudWatch 크로스 계정 관측성 링크 및 싱크를 완전하게 관리할 수 있는 새로운 정책을 추가했습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.	2022년 11월 27일
OAMReadOnlyAccess - 새 정책	CloudWatch에서는 CloudWatch 크로스 계정 관측성 링크 및 싱크에 대한 정보를 볼 수 있는 새로운 정책을 추가했습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 단원을 참조하십시오.	2022년 11월 27일

변경 사항	설명	날짜
CloudWatchFullAccess – 기존 정책에 대한 업데이트	CloudWatch는 CloudWatchFullAccess에 권한을 추가했습니다. 이 정책이 적용되는 사용자가 콘솔을 사용하여 CloudWatch 크로스 계정 관측성에서 소스 계정의 공유된 데이터를 볼 수 있도록 여기에는 `oam:ListSinks` 및 `oam:ListAttachedLinks` 권한이 포함되었습니다.	2022년 11월 27일
CloudWatchReadOnlyAccess – 기존 정책에 대한 업데이트	CloudWatch에서는 CloudWatchReadOnlyAccess에 권한을 추가했습니다. 이 정책이 적용되는 사용자가 콘솔을 사용하여 CloudWatch 크로스 계정 관찰성에서 소스 계정의 공유된 데이터를 볼 수 있도록 여기에는 `oam:ListSinks` 및 `oam:ListAttachedLinks` 권한이 포함되었습니다.	2022년 11월 27일

변경 사항	설명	날짜
AmazonCloudWatchRUMServiceRolePolicy - 기존 정책에 대한 업데이트	CloudWatch RUM은 AmazonCloudWatchRUMServiceRolePolicy에서 조건 키를 업데이트했습니다. `"Condition": { "StringEquals": { "cloudwatch:namespace": "AWS/RUM" } }` 조건 키가 다음과 같이 변경되어 CloudWatch RUM이 사용자 지정 지표를 사용자 지정 지표 네임스페이스로 보낼 수 있습니다. `"Condition": { "StringLike": { "cloudwatch:namespace": [ "RUM/CustomMetrics/*", "AWS/RUM" ] } }`	2023년 2월 2일

변경 사항	설명	날짜
AmazonCloudWatchRUMReadOnlyAccess – 업데이트된 정책	CloudWatch가 AmazonCloudWatchRUMReadOnlyAccess 정책에 권한을 추가했습니다. CloudWatch RUM이 CloudWatch 및 Evidently로 확장된 지표를 전송할 수 있도록 `rum:ListRumMetricsDestinations` 및 `rum:BatchGetRumMetricsDefinitions` 권한이 추가되었습니다.	2022년 10월 27일
AmazonCloudWatchRUMServiceRolePolicy - 기존 정책에 대한 업데이트	CloudWatch RUM이 AmazonCloudWatchRUMServiceRolePolicy에 권한을 추가했습니다. CloudWatch RUM이 확장된 지표를 CloudWatch로 전송할 수 있도록 `cloudwatch:PutMetricData` 권한이 추가되었습니다.	2022년 10월 26일

변경 사항	설명	날짜
CloudWatchEvidentlyReadOnlyAccess – 기존 정책 업데이트	CloudWatch Evidently는 CloudWatchEvidentlyReadOnlyAccess에 권한을 추가했습니다. `evidently:GetSegment` , `evidently:ListSegments` 및 `evidently:ListSegmentReferences` 권한이 추가되어 이 정책이 있는 사용자가 생성된 Evidently 대상 세그먼트를 볼 수 있습니다.	2022년 8월 12일
CloudWatchSyntheticsFullAccess – 기존 정책 업데이트	CloudWatch Synthetics는 CloudWatchSyntheticsFullAccess에 권한을 추가했습니다. Canary가 삭제될 때 CloudWatch Synthetics에서 관련 리소스를 삭제할 수 있도록 `lambda:DeleteFunction` 과 `lambda:DeleteLayerVersion` 권한이 추가되었습니다. 고객이 canary의 IAM 역할에 연결된 정책을 볼 수 있도록 `iam:ListAttachedRolePolicies` 가 추가되었습니다.	2022년 5월 6일

변경 사항	설명	날짜
AmazonCloudWatchRUMFullAccess(새 정책)	CloudWatch는 CloudWatch RUM을 완전히 관리할 수 있는 새 정책을 추가했습니다. CloudWatch RUM을 사용하면 웹 애플리케이션에 대한 실제 사용자 모니터링을 수행할 수 있습니다. 자세한 내용은 CloudWatch RUM 사용 단원을 참조하십시오.	2021년 11월 29일
AmazonCloudWatchRUMReadOnlyAccess(새 정책)	CloudWatch는 CloudWatch RUM에 대한 읽기 전용 액세스를 활성화하는 새로운 정책을 추가했습니다. CloudWatch RUM을 사용하면 웹 애플리케이션에 대한 실제 사용자 모니터링을 수행할 수 있습니다. 자세한 내용은 CloudWatch RUM 사용 단원을 참조하십시오.	2021년 11월 29일
CloudWatchEvidentlyFullAccess(새 정책)	CloudWatch는 CloudWatch Evidently를 완전히 관리할 수 있는 새 정책을 추가했습니다. CloudWatch Evidently를 사용하면 웹 애플리케이션에 대한 A/B 실험을 수행하고 점진적으로 배포할 수 있습니다. 자세한 내용은 CloudWatch Evidently를 통해 출시 및 A/B 실험 수행 단원을 참조하십시오.	2021년 11월 29일

변경 사항	설명	날짜
CloudWatchEvidentlyReadOnlyAccess(새 정책)	CloudWatch는 CloudWatch Evidently에 대한 읽기 전용 액세스를 활성화하는 새로운 정책을 추가했습니다. CloudWatch Evidently를 사용하면 웹 애플리케이션에 대한 A/B 실험을 수행하고 점진적으로 배포할 수 있습니다. 자세한 내용은 CloudWatch Evidently를 통해 출시 및 A/B 실험 수행 단원을 참조하십시오.	2021년 11월 29일
AWSServiceRoleForCloudWatchRUM – 새 관리형 정책	CloudWatch는 CloudWatch RUM이 모니터링 데이터를 다른 관련 AWS 서비스에 게시하도록 허용하는 새 서비스 연결 역할에 대한 정책을 추가했습니다.	2021년 11월 29일

변경 사항	설명	날짜
CloudWatchSyntheticsFullAccess – 기존 정책 업데이트	CloudWatch Synthetics는 CloudWatchSyntheticsFullAccess에 대한 권한을 추가하고 해당 권한의 범위도 변경했습니다. 사용자가 canary 아티팩트를 암호화하는 데 사용할 수 있는 사용 가능한 AWS KMS 키를 나열할 수 있도록 `kms:ListAliases` 권한이 추가되었습니다. 사용자가 canary 아티팩트를 암호화하는 데 사용할 키의 세부 정보를 확인할 수 있도록 `kms:DescribeKey` 권한이 추가되었습니다. 그리고 사용자가 canary 아티팩트를 복호화할 수 있도록 `kms:Decrypt` 권한이 추가되었습니다. 이 복호화 기능은 Amazon S3 버킷 내의 리소스에서 사용하도록 제한됩니다. `s3:GetBucketLocation` 권한의 `Resource` 범위가 `*`에서 `arn:aws:s3:::*` 로 변경되었습니다.	2021년 9월 29일

변경 사항	설명	날짜
CloudWatchSyntheticsFullAccess – 기존 정책 업데이트	CloudWatch Synthetics는 [CloudWatchSyntheticsFullAccess]에 권한을 추가했습니다. 이 정책이 있는 사용자가 canary의 런타임 버전을 변경할 수 있도록 `lambda:UpdateFunctionCode` 권한을 추가했습니다.	2021년 7월 20일
AWSCloudWatchAlarms_ActionSSMIncidentsServiceRolePolicy – 새 관리형 정책	CloudWatch는 새 관리형 IAM 정책을 추가하여 CloudWatch가 AWS Systems Manager Incident Manager에서 인시던트를 생성할 수 있도록 했습니다.	2021년 5월 10일
CloudWatchAutomaticDashboardsAccess – 기존 정책 업데이트	CloudWatch는 [CloudWatchAutomaticDashboardsAccess] 관리형 정책에 권한을 추가했습니다. 교차 계정 대시보드 사용자가 CloudWatch Synthetics canary 실행에 관한 세부 정보를 볼 수 있도록 이 정책에 `synthetics:DescribeCanariesLastRun` 권한을 추가했습니다.	2021년 4월 20일
CloudWatch에서 변경 사항 추적 시작	CloudWatch가 AWS 관리형 정책의 변경 사항 추적을 시작했습니다.	2021년 4월 14일

조건 키를 사용하여 CloudWatch 네임스페이스에 대한 액세스 제한

IAM 조건 키를 사용하여 사용자가 지정된 CloudWatch 네임스페이스에만 지표를 게시하도록 제한합니다.

하나의 네임스페이스에만 게시 허용

다음 정책은 사용자가 MyCustomNamespace라는 네임스페이스에만 지표를 게시하도록 제한합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Resource": "*",
        "Action": "cloudwatch:PutMetricData",
        "Condition": {
            "StringEquals": {
                "cloudwatch:namespace": "MyCustomNamespace"
            }
        }
    }
}
```

네임스페이스에서 게시 제외

다음 정책은 사용자가 CustomNamespace2를 제외한 모든 네임스페이스에 지표를 게시할 수 있도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData"
        },
        {
            "Effect": "Deny",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
```

```
                    "cloudwatch:namespace": "CustomNamespace2"
                }
            }
        }
    ]
}
```

조건 키를 사용하여 Contributor Insights 사용자의 로그 그룹 액세스 제한

Contributor Insights에서 규칙을 생성하고 그 결과를 확인하려면 사용자에게 `cloudwatch:PutInsightRule` 권한이 있어야 합니다. 기본적으로 이 권한이 있는 사용자는 CloudWatch Logs의 로그 그룹을 평가하는 Contributor Insights 규칙을 생성한 다음, 결과를 확인할 수 있습니다. 결과에는 해당 로그 그룹의 기여자 데이터가 포함될 수 있습니다.

조건 키를 사용해 IAM 정책을 생성하여 사용자에게 일부 로그 그룹에 대한 Contributor Insights 규칙을 쓸 수 있는 권한을 부여하는 동시에 다른 로그 그룹에 대한 규칙을 쓰고 이 데이터를 보는 것을 방지할 수 있습니다.

IAM 정책의 `Condition` 요소에 대한 자세한 내용은 IAM JSON 정책 요소: Condition 단원을 참조하세요.

특정 로그 그룹에 대해서만 규칙을 쓰고 결과를 볼 수 있는 액세스 권한 허용

다음 정책은 사용자에게 `AllowedLogGroup`이라는 로그 그룹과 이름이 `AllowedWildCard`로 시작하는 모든 로그 그룹에 대한 규칙을 쓰고 결과를 볼 수 있는 액세스 권한을 허용합니다. 다른 로그 그룹에 대한 규칙을 쓰거나 규칙 결과를 볼 수 있는 액세스 권한은 부여하지 않습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCertainLogGroups",
            "Effect": "Allow",
            "Action": "cloudwatch:PutInsightRule",
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*",
            "Condition": {
                "ForAllValues:StringEqualsIgnoreCase": {
                    "cloudwatch:requestInsightRuleLogGroups": [
                        "AllowedLogGroup",
                        "AllowedWildcard*"
                    ]
                }
```

```
            }
        }
    ]
}
```

특정 로그 그룹에 대한 규칙 쓰기는 거부하지만 다른 모든 로그 그룹에 대한 규칙 쓰기는 허용

다음 정책은 사용자에게 ExplicitlyDeniedLogGroup이라는 로그 그룹에 대한 규칙 쓰기 및 규칙 결과 보기 액세스 권한을 명시적으로 거부하지만 다른 모든 로그 그룹에 대한 규칙 쓰기 및 규칙 결과 보기는 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowInsightRulesOnLogGroupsByDefault",
            "Effect": "Allow",
            "Action": "cloudwatch:PutInsightRule",
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*"

        },
        {
            "Sid": "ExplicitDenySomeLogGroups",
            "Effect": "Deny",
            "Action": "cloudwatch:PutInsightRule",
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*",
            "Condition": {
                "ForAllValues:StringEqualsIgnoreCase": {
                    "cloudwatch:requestInsightRuleLogGroups": [
                        "/test/alpine/ExplicitlyDeniedLogGroup"
                    ]
                }
            }
        }
    ]
}
```

조건 키를 사용하여 경보 작업 제한

CloudWatch 경보의 상태가 변경되면 해당 경보는 EC2 인스턴스 중지 및 종료, Systems Manager 작업 수행과 같은 다양한 작업을 수행할 수 있습니다. 경보가 ALARM, OK 또는 INSUFFICIENT_DATA를 포함하여 어떤 상태로든 변경될 때 이러한 작업을 시작할 수 있습니다.

`cloudwatch:AlarmActions` 조건 키를 사용하면 사용자가 경보 상태 변경 시 지정된 작업만 수행할 수 있는 경보를 생성하도록 허용할 수 있습니다. 예를 들어 사용자가 EC2 작업이 아닌 작업만 수행할 수 있는 경보를 생성하도록 허용할 수 있습니다.

사용자가 Amazon SNS 알림 전송 또는 Systems Manager 작업 수행만 가능한 경보를 생성하도록 허용

다음 정책은 사용자가 Amazon SNS 알림 전송 및 Systems Manager 작업 수행만 가능한 경보를 생성하도록 제한합니다. 사용자는 EC2 작업을 수행하는 경보를 생성할 수 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateAlarmsThatCanPerformOnlySNSandSSMActions",
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricAlarm",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringLike": {
                    "cloudwatch:AlarmActions": [
                        "arn:aws:sns:*",
                        "arn:aws:ssm:*"
                    ]
                }
            }
        }
    ]
}
```

CloudWatch에 서비스 연결 역할 사용

Amazon CloudWatch는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 CloudWatch에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 CloudWatch에서 미리 정의하며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

CloudWatch의 한 서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요 없이 Amazon EC2 인스턴스를 종료, 중지 또는 재부팅할 수 있는 CloudWatch 경보를 설정할 수 있습니다. 또 다른 서비스 연결 역할을 사용하면 모니터링 계정이 지정된 다른 계정의 CloudWatch 데이터에 액세스하여 교차 계정 교차 리전 대시보드를 구축할 수 있습니다.

CloudWatch는 이러한 서비스 연결 역할의 권한을 정의합니다. 달리 정의되어 있지 않는 한, CloudWatch만 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 제한하면 리소스에 액세스할 수 있는 권한을 실수로 제거할 수 없기 때문에 CloudWatch가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대해 자세히 알아보려면 AWS Services That Work with IAM(IAM과 연동하는 서비스)을 참조하고 Service-Linked Role(서비스 연결 역할) 열에 Yes(예)라고 표시된 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

CloudWatch 경보 EC2 작업에 대한 서비스 연결 역할 권한

CloudWatch는 [AWSServiceRoleForCloudWatchEvents]라는 서비스 연결 역할을 사용합니다. CloudWatch는 이 서비스 연결 역할을 사용하여 Amazon EC2 경보 작업을 수행합니다.

AWSServiceRoleForCloudWatchEvents 서비스 연결 역할은 역할을 맡는 CloudWatch Events 서비스를 신뢰합니다. CloudWatch Events는 경보에 의해 호출될 때 인스턴스 종료, 중지 또는 재부팅 작업을 호출합니다.

AWSServiceRoleForCloudWatchEvents 서비스 연결 역할 권한 정책은 CloudWatch Events가 Amazon EC2 인스턴스에서 다음 작업을 완료하도록 허용합니다.

- `ec2:StopInstances`
- `ec2:TerminateInstances`
- `ec2:RecoverInstances`
- `ec2:DescribeInstanceRecoveryAttribute`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceStatus`

[AWSServiceRoleForCloudWatchCrossAccount] 서비스 연결 역할 권한 정책은 CloudWatch가 다음 작업을 완료하도록 허용합니다.

- `sts:AssumeRole`

CloudWatch Application Signals에 대한 서비스 연결 역할 권한

곧 출시될 기능에서는 AWSServiceRoleForCloudWatchApplicationSignals라는 서비스 연결 역할을 사용할 예정입니다. CloudWatch는 이 서비스 연결 역할을 사용하여 CloudWatch Logs 데이터, X-Ray 트레이스 데이터, CloudWatch 지표 데이터, 태깅 데이터를 수집합니다.

AWSServiceRoleForCloudWatchApplicationSignals 서비스 연결 역할은 CloudWatch Application Signals를 신뢰하여 역할을 수임합니다. Application Signals는 계정에서 로그, 트레이스, 지표 및 태그 데이터를 수집합니다.

AWSServiceRoleForCloudWatchApplicationSignals에는 IAM 정책이 첨부되어 있으며, 이 정책의 이름은 CloudWatchApplicationSignalsServiceRolePolicy입니다. 이 정책은 CloudWatch Application Signals에 다른 관련 AWS 서비스로부터 모니터링 및 태깅 데이터를 수집할 수 있는 권한을 부여합니다. 여기에는 Application Signals가 다음 작업을 완료할 수 있는 권한이 포함됩니다.

- `xray:GetServiceGraph`
- `logs:StartQuery`
- `logs:GetQueryResults`
- `cloudwatch:GetMetricData`
- `cloudwatch:ListMetrics`
- `tag:GetResources`

CloudWatchApplicationSignalsServiceRolePolicy의 전체 내용은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "XRayPermission",
            "Effect": "Allow",
            "Action": [
                "xray:GetServiceGraph"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
```

```
                }
            }
        },
        {
            "Sid": "CWLogsPermission",
            "Effect": "Allow",
            "Action": [
                "logs:StartQuery",
                "logs:GetQueryResults"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/appsignals/*:*",
                "arn:aws:logs:*:*:log-group:/aws/application-signals/data:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "CWListMetricsPermission",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "CWGetMetricDataPermission",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData"
            ],
            "Resource": [
                "*"
            ]
```

```
        },
        {
            "Sid": "TagsPermission",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        }
    ]
}
```

CloudWatch 경보 Systems Manager OpsCenter 작업에 대한 서비스 연결 역할 권한

CloudWatch는 [AWSServiceRoleForCloudWatchAlarms_ActionSSM]이라는 서비스 연결 역할을 사용합니다. CloudWatch는 이 서비스 연결 역할을 사용하여 CloudWatch 경보가 ALARM 상태가 될 때 Systems Manager OpsCenter 작업을 수행합니다.

AWSServiceRoleForCloudWatchAlarms_ActionSSM 서비스 연결 역할은 역할을 맡는 CloudWatch 서비스를 신뢰합니다. CloudWatch 경보는 경보에 의해 호출될 때 Systems Manager OpsCenter 작업을 호출합니다.

[AWSServiceRoleForCloudWatchAlarms_ActionSSM] 서비스 연결 역할 권한 정책은 Systems Manager가 다음 작업을 완료하도록 허용합니다.

- `ssm:CreateOpsItem`

CloudWatch 경보 Systems Manager Incident Manager 작업에 대한 서비스 연결 역할 권한

CloudWatch는 [AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents]라는 서비스 연결 역할을 사용합니다. CloudWatch는 이 서비스 연결 역할을 사용하여 CloudWatch 경보가 ALARM 상태가 될 때 Incident Manager 인시던트를 시작합니다.

[AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents] 서비스 연결 역할은 역할을 맡는 CloudWatch 서비스를 신뢰합니다. CloudWatch 경보는 경보에 의해 호출될 때 Systems Manager Incident Manager 작업을 호출합니다.

[AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents] 서비스 연결 역할 권한 정책은 Systems Manager가 다음 작업을 완료하도록 허용합니다.

- `ssm-incidents:StartIncident`

CloudWatch 교차 계정 교차 리전에 대한 서비스 연결 역할 권한

CloudWatch는 [AWSServiceRoleForCloudWatchCrossAccount]라는 서비스 연결 역할을 사용합니다. CloudWatch는 이 역할을 사용하여 지정된 다른 AWS 계정의 CloudWatch 데이터에 액세스합니다. SLR은 CloudWatch 서비스가 공유 계정의 역할을 맡을 수 있도록 역할 수임 권한만 제공합니다. 데이터에 대한 액세스 권한을 제공하는 공유 역할입니다.

[AWSServiceRoleForCloudWatchCrossAccount] 서비스 연결 역할 권한 정책은 CloudWatch가 다음 작업을 완료하도록 허용합니다.

- `sts:AssumeRole`

[AWSServiceRoleForCloudWatchCrossAccount] 서비스 연결 역할은 역할을 맡는 CloudWatch 서비스를 신뢰합니다.

CloudWatch 데이터베이스 성능 개선 도우미에 대한 서비스 연결 역할 권한

CloudWatch는 AWSServiceRoleForCloudWatchMetrics_DbPerfInsights라는 서비스 연결 역할을 사용합니다. – CloudWatch는 이 역할을 사용하여 경보 생성 및 스냅샷 생성을 위한 성능 개선 도우미 지표를 검색합니다.

AWSServiceRoleForCloudWatchMetrics_DbPerfInsights 서비스 연결 역할에는 `AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy` IAM 정책이 연결되어 있습니다. 해당 정책의 내용은 다음과 같습니다.

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

```
    "Action": [
     "pi:GetResourceMetrics"
    ],
    "Resource": "*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceAccount": "${aws:PrincipalAccount}"
     }
    }
   }
  ]
}
```

AWSServiceRoleForCloudWatchMetrics_DbPerfInsights 서비스 연결 역할은 CloudWatch 서비스를 신뢰하여 역할을 위임합니다.

CloudWatch에 대한 서비스 연결 역할 생성

이러한 서비스 연결 역할을 수동으로 생성할 필요가 없습니다. AWS Management Console, IAM CLI 또는 IAM API에서 경보를 처음 생성할 때 CloudWatch는 AWSServiceRoleForCloudWatchEvents와 AWSServiceRoleForCloudWatchAlarms_ActionSSM을 자동으로 생성합니다.

서비스 및 토폴로지 검색을 처음 활성화하면 Application Signals가 AWSServiceRoleForCloudWatchApplicationSignals를 생성합니다.

계정을 크로스 계정 크로스 리전 기능에 대한 모니터링 계정으로 처음 활성화할 때 CloudWatch는 AWSServiceRoleForCloudWatchCrossAccount를 자동으로 생성합니다.

`DB_PERF_INSIGHTS` 지표 수학 함수를 사용하는 경보를 처음 생성하면 CloudWatch가 AWSServiceRoleForCloudWatchMetrics_DbPerfInsights를 생성합니다.

자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 생성 섹션을 참조하세요.

CloudWatch에 대한 서비스 연결 역할 편집

CloudWatch에서는 AWSServiceRoleForCloudWatchEvents, AWSServiceRoleForCloudWatchAlarms_ActionSSM, AWSServiceRoleForCloudWatchCrossAccount 또는 AWSServiceRoleForCloudWatchMetrics_DbPerfInsights 역할을 편집하는 것을 허용하지 않습니다. 다양한 개체가 역할을 참조할 수 있으므로 이러한 역할을 생성한 후에는 이름을 변경할 수 없습니다. 그러나 IAM을 사용하여 이러한 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할 설명 편집(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 편집하려면(콘솔 사용)

1. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
2. 변경할 역할 이름을 선택합니다.
3. 역할 설명의 맨 오른쪽에서 편집을 선택합니다.
4. 상자에 새 설명을 입력하고 저장을 선택합니다.

서비스 연결 역할 설명 편집(AWS CLI)

AWS Command Line Interface에서 IAM 명령을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하려면(AWS CLI)

1. (옵션) 역할의 현재 설명을 보려면 다음 명령 중 하나를 사용합니다.

   ```
   $ aws iam get-role --role-name role-name
   ```

   AWS CLI 명령에서 역할을 참조하려면 ARN이 아니라 역할 이름을 사용해야 합니다. 예를 들어, 어떤 역할의 ARN이 `arn:aws:iam::123456789012:role/myrole`인 경우 참조할 역할은 **`myrole`**입니다.
2. 서비스 연결 역할의 설명을 업데이트하려면 다음 명령을 사용합니다.

   ```
   $ aws iam update-role-description --role-name role-name --description description
   ```

서비스 연결 역할 설명 편집(IAM API)

IAM API를 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하려면(API 사용)

1. (선택 사항) 역할의 현재 설명을 보려면 다음 명령을 사용합니다.

   GetRole

2. 역할 설명을 업데이트하려면 다음 명령을 사용합니다.

   UpdateRoleDescription

CloudWatch에 대한 서비스 연결 역할 삭제

자동으로 EC2 인스턴스를 중지, 종료 또는 재부팅하는 경보가 더 이상 없는 경우 AWSServiceRoleForCloudWatchEvents 역할을 삭제하는 것이 좋습니다.

Systems Manager OpsCenter 작업을 수행하는 경보가 더 이상 없는 경우 AWSServiceRoleForCloudWatchAlarms_ActionSSM 역할을 삭제하는 것이 좋습니다.

`DB_PERF_INSIGHTS` 지표 수학 함수를 사용하는 경보를 모두 삭제하는 경우 AWSServiceRoleForCloudWatchMetrics_DbPerfInsights 서비스 연결 역할을 삭제하는 것이 좋습니다.

따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 삭제 전에 서비스 연결 역할을 정리해야 합니다.

서비스 연결 역할 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에 활성 세션이 없는지 확인하고 역할에서 사용되는 리소스를 모두 제거해야 합니다.

IAM 콘솔에서 서비스 연결 역할에 활성 세션이 있는지 확인하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다. AWSServiceRoleForCloudWatchEvents 역할의 이름(확인란 아님)을 선택합니다.
3. 선택된 역할의 요약 페이지에서 Access Advisor(액세스 관리자)를 선택하고 서비스 연결 역할의 최근 활동을 검토합니다.

> **Note**
>
> CloudWatch가 AWSServiceRoleForCloudWatchEvents 역할을 사용하고 있는지 확실하지 않은 경우 역할을 삭제해 보세요. 서비스에서 역할을 사용하는 경우에는 삭제가 안 되어 역할이 사용 중인 리전을 볼 수 있습니다. 역할이 사용 중인 경우에는 세션이 종료될 때까지 기다렸다가 역할을 삭제해야 합니다. 서비스 연결 역할에 대한 세션은 취소할 수 없습니다.

서비스 연결 역할 삭제(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할을 삭제하는 방법(콘솔)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다. 그런 다음 삭제할 역할의 이름이나 행이 아닌 이름 옆에 있는 확인란을 선택합니다.
3. 역할 작업에 대해 역할 삭제를 선택합니다.
4. 확인 대화 상자가 나타나면 서비스 마지막 액세스 데이터를 검토합니다. 이 데이터는 선택한 각 역할이 AWS 서비스를 마지막으로 액세스한 일시를 보여 줍니다. 이를 통해 역할이 현재 활동 중인지를 확인할 수 있습니다. 계속하려면 예, 삭제를 선택합니다.
5. IAM 콘솔 알림을 보고 서비스 연결 역할 삭제 진행 상황을 모니터링합니다. IAM 서비스 연결 역할 삭제는 비동기이므로 삭제할 역할을 제출한 후에 삭제 태스크가 성공하거나 실패할 수 있습니다. 작업에 실패할 경우 알림의 세부 정보 보기 또는 View Resources(리소스 보기)를 선택하면 삭제 실패 이유를 확인할 수 있습니다. 역할에서 사용 중인 리소스가 서비스에 있기 때문에 삭제에 실패하는 경우, 실패 원인에 리소스 목록이 포함됩니다.

서비스 연결 역할 삭제(AWS CLI)

AWS Command Line Interface에서 IAM 명령을 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할을 삭제하려면(AWS CLI)

1. 서비스 연결 역할이 사용되지 않거나 연결된 리소스가 없는 경우에는 서비스 연결 역할을 삭제할 수 없으므로 삭제 요청을 제출해야 합니다. 이러한 조건이 충족되지 않으면 요청이 거부될 수 있습니다. 삭제 태스크 상태를 확인하려면 응답의 `deletion-task-id`(을)를 캡처해야 합니다. 다음 명령을 입력하여 서비스 연결 역할 삭제 요청을 제출합니다.

```
$ aws iam delete-service-linked-role --role-name service-linked-role-name
```

2. 다음 명령을 입력하여 삭제 작업의 상태를 확인합니다.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

삭제 태스크는 NOT_STARTED, IN_PROGRESS, SUCCEEDED 또는 FAILED 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

서비스 연결 역할 삭제(IAM API)

IAM API를 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할(API)을 삭제하는 방법

1. 서비스 연결 역할 삭제 요청을 제출하려면 DeleteServiceLinkedRole을 호출합니다. 요청에서 삭제할 역할 이름을 지정합니다.

   서비스 연결 역할이 사용되지 않거나 연결된 리소스가 없는 경우에는 서비스 연결 역할을 삭제할 수 없으므로 삭제 요청을 제출해야 합니다. 이러한 조건이 충족되지 않으면 요청이 거부될 수 있습니다. 삭제 태스크 상태를 확인하려면 응답의 DeletionTaskId(을)를 캡처해야 합니다.
2. 삭제 상태를 확인하려면 GetServiceLinkedRoleDeletionStatus를 호출합니다. 요청에 DeletionTaskId(을)를 지정합니다.

   삭제 태스크는 NOT_STARTED, IN_PROGRESS, SUCCEEDED 또는 FAILED 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

AWS 서비스 연결 역할에 대한 CloudWatch 업데이트

아래에는 CloudWatch의 AWS 관리형 정책 업데이트에 관한 세부 정보가 나와 있습니다. 이 서비스가 해당 변경 사항을 추적하기 시작한 이후부터 설명되어 있습니다. 이 페이지의 변경 사항에 관한 자동 알림을 받으려면 CloudWatch 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AWSServiceRoleForCloudWatchApplicationSignals – 서비스 연결 역할 정책의 권한 업데이트	CloudWatch에서 이 역할이 부여하는 logs:StartQuery 및 logs:GetQueryResults 권한의 범위에 더 많은 로그 그룹이 추가됩니다.	2024년 4월 24일

변경 사항	설명	날짜
AWSServiceRoleForCloudWatchApplicationSignals – 새 서비스 연결 역할	CloudWatch는 CloudWatch Application Signals가 CloudWatch Application Signals에 대해 활성화한 애플리케이션에서 CloudWatch Logs 데이터, X-Ray 트레이스 데이터, CloudWatch 지표 데이터 및 태그 지정 데이터를 수집할 수 있도록 이 새로운 서비스 연결 역할을 추가했습니다.	2023년 11월 9일
AWSServiceRoleForCloudWatchMetrics_DbPerfInsights – 새로운 서비스 연결 역할	CloudWatch는 CloudWatch가 경보 및 스냅샷 생성을 위해 성능 개선 도우미 지표를 가져올 수 있도록 이 새로운 서비스 연결 역할을 추가했습니다. IAM 정책은 이 역할에 연결되며, 정책은 CloudWatch에 사용자를 대신하여 성능 개선 도우미 지표를 가져올 수 있는 권한을 부여합니다.	2023년 9월 13일
AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents – 새 서비스 연결 역할	CloudWatch는 새 서비스 연결 역할을 추가하여 CloudWatch가 AWS Systems Manager Incident Manager에서 인시던트를 생성할 수 있도록 했습니다.	2021년 4월 26일
CloudWatch에서 변경 사항 추적 시작	CloudWatch가 서비스 연결 역할의 변경 사항 추적을 시작했습니다.	2021년 4월 26일

CloudWatch RUM에 서비스 연결 역할 사용

CloudWatch RUM은 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 RUM에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 RUM에서 사전 정의하며, 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

RUM에서 서비스 연결 역할 권한을 정의하므로 달리 정의되지 않은 한 RUM만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 역할의 관련 리소스를 삭제해야만 역할을 삭제할 수 있습니다. 이렇게 제한하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없으므로 RUM 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-linked roles) 열에 예(Yes)가 있는 서비스를 찾으십시오. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

RUM에 대한 서비스 연결 역할 권한

RUM은 AWSServiceRoleForCloudWatchRUM이라는 서비스 연결 역할을 사용합니다. X-Ray 추적을 활성화한 앱 모니터에 대해 이 역할을 사용하여 RUM이 AWS X-Ray 추적 데이터를 사용자 계정으로 전송할 수 있습니다.

AWSServiceRoleForCloudWatchRUM 서비스 연결 역할은 해당 역할을 맡는 X-Ray 서비스를 신뢰합니다. X-Ray는 추적 데이터를 계정으로 보냅니다.

AWSServiceRoleForCloudWatchRUM 서비스 연결 역할에는 AmazonCloudWatchRUMServiceRolePolicy에 연결된 IAM 정책이 있습니다. 이 정책은 CloudWatch RUM에 모니터링 데이터를 다른 관련 AWS 서비스에 게시할 수 있는 권한을 부여합니다. 여기에는 RUM이 다음 작업을 완료할 수 있도록 허용하는 권한이 포함됩니다.

- `xray:PutTraceSegments`
- `cloudwatch:PutMetricData`

AmazonCloudWatchRUMServiceRolePolicy의 전체 내용은 다음과 같습니다.

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

```
  {
   "Effect": "Allow",
   "Action": [
    "xray:PutTraceSegments"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringLike": {
     "cloudwatch:namespace": [
      "RUM/CustomMetrics/*",
      "AWS/RUM"
     ]
    }
   }
  }
 ]
}
```

RUM에 대한 서비스 연결 역할 생성

CloudWatch RUM에 대한 서비스 연결 역할은 수동으로 생성할 필요가 없습니다. X-Ray 추적이 활성화된 앱 모니터를 처음 생성하거나 X-Ray 추적을 사용하도록 앱 모니터를 처음 업데이트하면 RUM이 AWSServiceRoleForCloudWatchRUM을 생성합니다.

자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 생성 섹션을 참조하세요.

RUM에 대한 서비스 연결 역할 편집

CloudWatch RUM은 AWSServiceRoleForCloudWatchRUM 역할 편집을 허용하지 않습니다. 다양한 개체가 역할을 참조할 수 있으므로 이러한 역할을 생성한 후에는 이름을 변경할 수 없습니다. 그러나 IAM을 사용하여 이러한 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할 설명 편집(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 편집하려면(콘솔 사용)

1. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
2. 변경할 역할 이름을 선택합니다.
3. 역할 설명의 맨 오른쪽에서 편집을 선택합니다.
4. 상자에 새 설명을 입력하고 저장을 선택합니다.

서비스 연결 역할 설명 편집(AWS CLI)

AWS Command Line Interface에서 IAM 명령을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하려면(AWS CLI)

1. (옵션) 역할의 현재 설명을 보려면 다음 명령 중 하나를 사용합니다.

   ```
   $ aws iam get-role --role-name role-name
   ```

   AWS CLI 명령에서 역할을 참조하려면 ARN이 아니라 역할 이름을 사용해야 합니다. 예를 들어, 어떤 역할의 ARN이 `arn:aws:iam::123456789012:role/myrole`인 경우 참조할 역할은 **`myrole`**입니다.
2. 서비스 연결 역할의 설명을 업데이트하려면 다음 명령을 사용합니다.

   ```
   $ aws iam update-role-description --role-name role-name --description description
   ```

서비스 연결 역할 설명 편집(IAM API)

IAM API를 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하려면(API 사용)

1. (선택 사항) 역할의 현재 설명을 보려면 다음 명령을 사용합니다.

   GetRole
2. 역할 설명을 업데이트하려면 다음 명령을 사용합니다.

   UpdateRoleDescription

RUM에 대한 서비스 연결 역할 삭제

활성화된 X-Ray를 사용하는 앱 모니터가 더 이상 없는 경우 AWSServiceRoleForCloudWatchRUM 역할을 삭제하는 것이 좋습니다.

따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 삭제 전에 서비스 연결 역할을 정리해야 합니다.

서비스 연결 역할 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에 활성 세션이 없는지 확인하고 역할에서 사용되는 리소스를 모두 제거해야 합니다.

IAM 콘솔에서 서비스 연결 역할에 활성 세션이 있는지 확인하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다. AWSServiceRoleForCloudWatchRUM 역할의 이름(확인란 아님)을 선택합니다.
3. 선택된 역할의 요약 페이지에서 Access Advisor(액세스 관리자)를 선택하고 서비스 연결 역할의 최근 활동을 검토합니다.

> **Note**
>
> RUM이 AWSServiceRoleForCloudWatchRUM 역할을 사용하고 있는지 확실하지 않은 경우 역할을 삭제할 수 있습니다. 서비스에서 역할을 사용하는 경우에는 삭제가 안 되어 역할이 사용 중인 리전을 볼 수 있습니다. 역할이 사용 중인 경우에는 세션이 종료될 때까지 기다렸다가 역할을 삭제해야 합니다. 서비스 연결 역할에 대한 세션은 취소할 수 없습니다.

서비스 연결 역할 삭제(IAM 콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할을 삭제하는 방법(콘솔)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다. 그런 다음 삭제할 역할의 이름이나 행이 아닌 이름 옆에 있는 확인란을 선택합니다.
3. 역할 작업에 대해 역할 삭제를 선택합니다.

4. 확인 대화 상자가 나타나면 서비스 마지막 액세스 데이터를 검토합니다. 이 데이터는 선택한 각 역할이 AWS 서비스를 마지막으로 액세스한 일시를 보여 줍니다. 이를 통해 역할이 현재 활동 중인지를 확인할 수 있습니다. 계속하려면 예, 삭제를 선택합니다.
5. IAM 콘솔 알림을 보고 서비스 연결 역할 삭제 진행 상황을 모니터링합니다. IAM 서비스 연결 역할 삭제는 비동기이므로 삭제할 역할을 제출한 후에 삭제 태스크가 성공하거나 실패할 수 있습니다. 작업에 실패할 경우 알림의 세부 정보 보기 또는 View Resources(리소스 보기)를 선택하면 삭제 실패 이유를 확인할 수 있습니다. 역할에서 사용 중인 리소스가 서비스에 있기 때문에 삭제에 실패하는 경우, 실패 원인에 리소스 목록이 포함됩니다.

서비스 연결 역할 삭제(AWS CLI)

AWS Command Line Interface에서 IAM 명령을 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할을 삭제하려면(AWS CLI)

1. 서비스 연결 역할이 사용되지 않거나 연결된 리소스가 없는 경우에는 서비스 연결 역할을 삭제할 수 없으므로 삭제 요청을 제출해야 합니다. 이러한 조건이 충족되지 않으면 요청이 거부될 수 있습니다. 삭제 태스크 상태를 확인하려면 응답의 `deletion-task-id`(을)를 캡처해야 합니다. 다음 명령을 입력하여 서비스 연결 역할 삭제 요청을 제출합니다.

```
$ aws iam delete-service-linked-role --role-name service-linked-role-name
```

2. 다음 명령을 입력하여 삭제 작업의 상태를 확인합니다.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

삭제 태스크는 `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED` 또는 `FAILED` 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

서비스 연결 역할 삭제(IAM API)

IAM API를 사용하여 서비스 연결 역할을 삭제할 수 있습니다.

서비스 연결 역할(API)을 삭제하는 방법

1. 서비스 연결 역할 삭제 요청을 제출하려면 DeleteServiceLinkedRole을 호출합니다. 요청에서 삭제할 역할 이름을 지정합니다.

서비스 연결 역할이 사용되지 않거나 연결된 리소스가 없는 경우에는 서비스 연결 역할을 삭제할 수 없으므로 삭제 요청을 제출해야 합니다. 이러한 조건이 충족되지 않으면 요청이 거부될 수 있습니다. 삭제 태스크 상태를 확인하려면 응답의 `DeletionTaskId`(을)를 캡처해야 합니다.

2. 삭제 상태를 확인하려면 GetServiceLinkedRoleDeletionStatus를 호출합니다. 요청에 `DeletionTaskId`(을)를 지정합니다.

   삭제 태스크는 `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED` 또는 `FAILED` 상태일 수 있습니다. 삭제에 실패할 경우 문제를 해결할 수 있도록 실패 이유가 호출에 반환됩니다.

CloudWatch Application Insights에 서비스 연결 역할 사용

CloudWatch Application Insights는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 CloudWatch Application Insights에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 CloudWatch Application Insights에서 미리 정의하며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는 데 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 CloudWatch Application Insights를 더 쉽게 설정할 수 있습니다. CloudWatch Application Insights는 해당 서비스 연결 역할의 권한을 정의합니다. 달리 정의되어 있지 않는 한, CloudWatch Application Insights만 역할을 맡을 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 참조해 서비스 연결 역할(Service-Linked Role) 열이 예(Yes)인 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

CloudWatch Application Insights에 대한 서비스 연결 역할 권한

CloudWatch Application Insights는 [AWSServiceRoleForApplicationInsights]라는 서비스 연결 역할을 사용합니다. Application Insights는 이 역할을 사용하여 고객의 리소스 그룹 분석, 지표에 대한 경보 생성을 위한 CloudFormation 스택 생성, EC2 인스턴스에서 CloudWatch 에이전트 구성과 같은 작업을 수행합니다. 이 서비스 연결 역할에는 연결된 IAM 정책이 있으며, 이름은 `CloudwatchApplicationInsightsServiceLinkedRolePolicy`입니다. 이 정책에 대한 업데이트는 AWS 관리형 정책에 대한 Application Insights 업데이트 단원을 참조하세요.

역할 권한 정책은 CloudWatch Application Insights가 리소스에서 다음 작업을 완료하도록 허용합니다.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:DeleteAlarms",
        "cloudwatch:PutAnomalyDetector",
        "cloudwatch:DeleteAnomalyDetector",
        "cloudwatch:DescribeAnomalyDetectors"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents",
        "logs:DescribeLogStreams",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudFormation:CreateStack",
```

```
        "cloudFormation:UpdateStack",
        "cloudFormation:DeleteStack",
        "cloudFormation:DescribeStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudFormation:DescribeStacks",
        "cloudFormation:ListStackResources",
        "cloudFormation:ListStacks"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "tag:GetResources"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "resource-groups:ListGroupResources",
        "resource-groups:GetGroupQuery",
        "resource-groups:GetGroup"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "resource-groups:CreateGroup",
        "resource-groups:DeleteGroup"
```

```
      ],
      "Resource": [
        "arn:aws:resource-groups:*:*:group/ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:PutParameter",
        "ssm:DeleteParameter",
        "ssm:AddTagsToResource",
        "ssm:RemoveTagsFromResource",
        "ssm:GetParameters"
      ],
      "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-ApplicationInsights-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:CreateAssociation",
        "ssm:UpdateAssociation",
        "ssm:DeleteAssociation",
        "ssm:DescribeAssociation"
      ],
```

```
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ssm:*:*:association/*",
        "arn:aws:ssm:*:*:managed-instance/*",
        "arn:aws:ssm:*:*:document/AWSEC2-
ApplicationInsightsCloudwatchAgentInstallAndConfigure",
        "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
        "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsItem",
        "ssm:CreateOpsItem",
        "ssm:DescribeOpsItems",
        "ssm:UpdateOpsItem",
        "ssm:DescribeInstanceInformation"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:AddTagsToResource"
      ],
      "Resource": "arn:aws:ssm:*:*:opsitem/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:ListCommandInvocations",
        "ssm:GetCommandInvocation"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "ssm:SendCommand",
      "Resource": [
```

```
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ssm:*:*:document/AWSEC2-CheckPerformanceCounterSets",
        "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
        "arn:aws:ssm:*:*:document/AWSEC2-DetectWorkload",
        "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "ec2:DescribeVolumeStatus",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeNatGateways"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "rds:DescribeDBInstances",
        "rds:DescribeDBClusters"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "lambda:ListFunctions",
        "lambda:GetFunctionConfiguration",
        "lambda:ListEventSourceMappings"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",
        "events:DeleteRule"
      ],
      "Resource": [
        "arn:aws:events:*:*:rule/AmazonCloudWatch-ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "xray:GetServiceGraph",
        "xray:GetTraceSummaries",
        "xray:GetTimeSeriesServiceStatistics",
        "xray:GetTraceGraph"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:ListTables",
        "dynamodb:DescribeTable",
        "dynamodb:DescribeContributorInsights",
        "dynamodb:DescribeTimeToLive"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "application-autoscaling:DescribeScalableTargets"
      ],
      "Resource": [
        "*"
      ]
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetMetricsConfiguration",
        "s3:GetReplicationConfiguration"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:ListStateMachines",
        "states:DescribeExecution",
        "states:DescribeStateMachine",
        "states:GetExecutionHistory"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeClusters",
        "ecs:DescribeContainerInstances",
        "ecs:DescribeServices",
        "ecs:DescribeTaskDefinition",
        "ecs:DescribeTasks",
        "ecs:DescribeTaskSets",
        "ecs:ListClusters",
        "ecs:ListContainerInstances",
        "ecs:ListServices",
        "ecs:ListTasks"
```

```
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecs:UpdateClusterSettings"
      ],
      "Resource": [
        "arn:aws:ecs:*:*:cluster/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:DescribeCluster",
        "eks:DescribeFargateProfile",
        "eks:DescribeNodegroup",
        "eks:ListClusters",
        "eks:ListFargateProfiles",
        "eks:ListNodegroups",
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:GetSubscriptionAttributes",
        "sns:GetTopicAttributes",
        "sns:GetSMSAttributes",
        "sns:ListSubscriptionsByTopic",
        "sns:ListTopics"
      ],
      "Resource": [
        "*"
      ]
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "sqs:ListQueues"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DeleteSubscriptionFilter"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutSubscriptionFilter"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:*",
        "arn:aws:logs:*:*:destination:AmazonCloudWatch-ApplicationInsights-
LogIngestionDestination*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:DescribeFileSystems"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:GetHostedZone",
        "route53:GetHealthCheck",
        "route53:ListHostedZones",
        "route53:ListHealthChecks",
        "route53:ListQueryLoggingConfigs"
      ],
```

```
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53resolver:ListFirewallRuleGroupAssociations",
        "route53resolver:GetFirewallRuleGroup",
        "route53resolver:ListFirewallRuleGroups",
        "route53resolver:ListResolverEndpoints",
        "route53resolver:GetResolverQueryLogConfig",
        "route53resolver:ListResolverQueryLogConfigs",
        "route53resolver:ListResolverQueryLogConfigAssociations",
        "route53resolver:GetResolverEndpoint",
        "route53resolver:GetFirewallRuleGroupAssociation"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 작성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한 섹션을 참조하세요.

CloudWatch Application Insights에 대한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console에서 새 Application Insights 애플리케이션을 생성하면 CloudWatch Application Insights가 서비스 연결 역할을 자동으로 생성합니다.

이 서비스 연결 역할을 삭제한 다음 다시 생성하려는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 새 Application Insights 애플리케이션을 생성하면 CloudWatch Application Insights가 서비스 연결 역할을 다시 자동으로 생성합니다.

CloudWatch Application Insights에 대한 서비스 연결 역할 편집

CloudWatch Application Insights에서는 AWSServiceRoleForApplicationInsights 서비스 연결 역할을 편집할 수 없습니다. 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 편집을 참조하세요.

CloudWatch Application Insights에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 이렇게 하면 적극적으로 모니터링되거나 유지 관리되지 않는 미사용 개체를 피할 수 있습니다. 그러나 역할을 수동으로 삭제하기 전에 Application Insights에서 모든 애플리케이션을 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 CloudWatch Application Insights 서비스가 역할을 사용 중인 경우 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

AWSServiceRoleForApplicationInsights가 사용하는 CloudWatch Application Insights 리소스를 삭제하려면

- 모든 CloudWatch Application Insights 애플리케이션을 삭제합니다. 자세한 내용은 CloudWatch Application Insights 사용 설명서의 '애플리케이션 삭제' 단원을 참조하세요.

IAM을 사용하여 수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForApplicationInsights 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제 섹션을 참조하세요.

CloudWatch Application Insights 서비스 연결 역할에 지원되는 리전

CloudWatch Application Insights는 서비스를 사용할 수 있는 모든 AWS 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용은 CloudWatch Application Insights 리전 및 엔드포인트 단원을 참조하세요.

Amazon CloudWatch Application Insights에 대한 AWS 관리형 정책

AWS 관리형 정책은 AWS에 의해 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. AWS에서 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 보안 주체 엔터티(사용자, 그룹 및 역할)에도 업데이트가 적용됩니다. 새로운 AWS 서비스를 시작하거나 새로운 API 작업을 기존 서비스에 이용하는 경우 AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: CloudWatchApplicationInsightsFullAccess

`CloudWatchApplicationInsightsFullAccess` 정책을 IAM 자격 증명에 연결할 수 있습니다.

이 정책은 Application Insights 기능에 대한 전체 액세스를 허용하는 관리 권한을 부여합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `applicationinsights` – Application Insights 기능에 대한 전체 액세스를 허용합니다.
- `iam` – Application Insights가 서비스 연결 역할인 AWSServiceRoleForApplicationInsights를 생성하도록 허용합니다. 이 역할은 Application Insights가 고객의 리소스 그룹을 분석하고 CloudFormation 스택을 생성하여 지표에 대한 경보를 생성하며 EC2 인스턴스에서 CloudWatch 에이전트를 구성하는 등의 작업을 수행할 수 있도록 하는 데 필요합니다. 자세한 내용은 CloudWatch Application Insights에 서비스 연결 역할 사용 단원을 참조하세요.

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

```
      "Action": "applicationinsights:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "rds:DescribeDBInstances",
        "rds:DescribeDBClusters",
        "sqs:ListQueues",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth",
        "autoscaling:DescribeAutoScalingGroups",
        "lambda:ListFunctions",
        "dynamodb:ListTables",
        "s3:ListAllMyBuckets",
        "sns:ListTopics",
        "states:ListStateMachines",
        "apigateway:GET",
        "ecs:ListClusters",
        "ecs:DescribeTaskDefinition",
        "ecs:ListServices",
        "ecs:ListTasks",
        "eks:ListClusters",
        "eks:ListNodegroups",
        "fsx:DescribeFileSystems",
        "logs:DescribeLogGroups",
        "elasticfilesystem:DescribeFileSystems"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/application-insights.amazonaws.com/
AWSServiceRoleForApplicationInsights"
      ],
      "Condition": {
        "StringEquals": {
```

```
          "iam:AWSServiceName": "application-insights.amazonaws.com"
        }
      }
    }
  ]
}
```

AWS 관리형 정책: CloudWatchApplicationInsightsReadOnlyAccess

`CloudWatchApplicationInsightsReadOnlyAccess` 정책을 IAM 자격 증명에 연결할 수 있습니다.

이 정책은 모든 Application Insights 기능에 대한 읽기 전용 액세스를 허용하는 관리 권한을 부여합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `applicationinsights` – Application Insights 기능에 대한 읽기 전용 액세스를 허용합니다.

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

AWS 관리형 정책: CloudwatchApplicationInsightsServiceLinkedRolePolicy

CloudwatchApplicationInsightsServiceLinkedRolePolicy를 IAM 엔터티에 연결할 수 없습니다. 이 정책은 Application Insights가 고객 리소스를 모니터링할 수 있게 허용하는 서비스 연결 역할에 연결됩니다. 자세한 내용은 CloudWatch Application Insights에 서비스 연결 역할 사용 단원을 참조하세요.

AWS 관리형 정책에 대한 Application Insights 업데이트

아래에는 Application Insights의 AWS 관리형 정책 업데이트에 관한 세부 정보가 나와 있습니다. 이 서비스가 해당 변경 사항을 추적하기 시작한 이후부터 설명되어 있습니다. 이 페이지의 변경 사항에 관한 자동 알림을 받으려면 Application Insights 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights에 CloudFormation 스택을 나열할 수 있는 새로운 권한이 추가되었습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 CloudFormation 스택에 중첩된 AWS 리소스를 분석하고 모니터하는 데 필요합니다.	2023년 4월 24일
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights에서는 Amazon VPC 및 Route 53 리소스의 목록을 가져오는 새 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights에서 Amazon CloudWatch을(를) 사용하여 모범 사례 네트워크 모니터링을	2023년 1월 23일

변경 사항	설명	날짜
	자동으로 설정하는 데 필요합니다.	
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights에는 SSM 명령 호출 결과를 가져오는 새 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights에서 Amazon EC2에서 실행 중인 워크로드를 자동으로 탐지 및 모니터링하는 데 필요합니다.	2022년 12월 19일
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights에서는 Amazon VPC 및 Route 53 리소스를 설명하는 새 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 고객 Amazon VPC 및 Route 53 리소스 구성을 읽고 고객이 Amazon CloudWatch을(를) 사용해 모범 사례 네트워크 모니터링을 자동으로 설정하도록 지원하는 데 필요합니다.	2022년 12월 19일

변경 사항	설명	날짜
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights에는 EFS 리소스를 기술하는 새 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 Amazon EFS 고객 리소스 구성을 읽고 고객이 CloudWatch를 사용해 EFS 모니터링에 대한 모범 사례를 자동으로 설정하도록 지원하는 데 필요합니다.	2022년 10월 3일
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights에는 EFS 파일 리스템을 기술하는 새 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 계정에서 지원되는 모든 리소스를 쿼리하여 계정 기반 애플리케이션을 생성하는 데 필요합니다.	2022년 10월 3일
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights에서는 FSx 리소스에 대한 정보를 검색하는 새 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 기본 FSx 볼륨에 대한 충분한 정보를 검색하여 워크로드를 모니터링하는 데 필요합니다.	2022년 9월 12일

변경 사항	설명	날짜
AWS 관리형 정책: CloudWatchApplicationInsightsFullAccess - 기존 정책에 대한 업데이트	Application Insights에서는 로그 그룹을 설명하는 새 권한을 추가했습니다. 새 애플리케이션을 생성할 때 모니터링 로그 그룹에 대한 올바른 권한이 계정에 있는지 확인하려면 Amazon CloudWatch Application Insights에 이 권한이 필요합니다.	2022년 1월 24일
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights에서는 CloudWatch Log 구독 필터를 생성하고 삭제하는 새 권한을 추가했습니다. 구성한 애플리케이션 내에서 리소스의 로그 모니터링이 용이하도록 Amazon CloudWatch Application Insights에서 구독 필터를 생성하려면 이러한 권한이 필요합니다.	2022년 1월 24일
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights는 Elastic Load Balancer의 대상 그룹 및 대상 상태를 설명하는 새로운 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 계정에서 지원되는 모든 리소스를 쿼리하여 계정 기반 애플리케이션을 생성하는 데 필요합니다.	2021년 11월 4일

변경 사항	설명	날짜
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights는 Amazon EC2 인스턴스에 대한 `AmazonCloudWatch-ManageAgent` SSM 문서를 실행하는 새로운 권한을 추가했습니다. 이 권한은 Amazon CloudWatch Application Insights가 Application Insights에서 생성한 CloudWatch 에이전트 구성 파일을 정리하는 데 필요합니다.	2021년 9월 30일

변경 사항	설명	날짜
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights는 계정 기반 애플리케이션 모니터링을 지원하는 새로운 권한을 추가하여 계정에서 지원되는 모든 리소스를 온보딩하고 모니터링합니다. 이러한 권한은 Amazon CloudWatch Application Insights에서 쿼리하고, 리소스에 태깅하고, 해당 리소스에 대한 그룹을 생성하는 데 필요합니다. Application Insights는 SNS 주제 모니터링을 지원하는 새로운 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 SNS 리소스에서 메타데이터를 수집하여 SNS 주제에 대한 모니터링을 구성하는 데 필요합니다.	2021년 9월 15일
AWS 관리형 정책: CloudWatchApplicationInsightsFullAccess - 기존 정책에 대한 업데이트	Application Insights는 지원되는 리소스를 기술하고 나열하는 새 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 계정에서 지원되는 모든 리소스를 쿼리하여 계정 기반 애플리케이션을 생성하는 데 필요합니다.	2021년 9월 15일

변경 사항	설명	날짜
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights는 FSx 리소스를 기술하는 새 권한을 추가했습니다. 이러한 권한은 Amazon CloudWatch Application Insights가 고객 FSx 리소스 구성을 읽고 고객이 CloudWatch를 사용해 모범 사례 FSx 모니터링을 자동으로 설정하도록 지원하는 데 필요합니다.	2021년 8월 31일
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights는 ECS 및 EKS 서비스 리소스를 기술하고 나열하는 새 권한을 추가했습니다. 이 권한은 Amazon CloudWatch Application Insights가 고객 컨테이너 리소스 구성을 읽고 고객이 CloudWatch를 사용해 모범 사례 컨테이너 모니터링을 자동으로 설정하도록 지원하는 데 필요합니다.	2021년 5월 18일

변경 사항	설명	날짜
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 기존 정책 업데이트	Application Insights는 OpsCenter가 `opsitem` 리소스 유형의 리소스에서 `ssm:AddTagsToResource` 작업을 사용하여 OpsItem에 태그를 지정할 수 있게 허용하는 새 권한을 추가했습니다. 이 권한은 OpsCenter에 필요합니다. Amazon CloudWatch Application Insights는 고객이 AWS SSM OpsCenter를 사용하여 문제를 해결할 수 있도록 OpsItem을 생성합니다.	2021년 4월 13일
Application Insights에서 변경 사항 추적 시작	Application Insights가 AWS 관리형 정책의 변경 사항 추적을 시작했습니다.	2021년 4월 13일

Amazon CloudWatch 권한 참조

다음 테이블에는 각 CloudWatch API 작업과 이 작업을 수행할 권한을 부여할 수 있는 작업이 나와 있습니다. 정책의 `Action` 필드에서 작업을 지정하고, 정책의 `Resource` 필드에서 리소스 값으로 와일드카드 문자 (*)를 지정합니다.

CloudWatch 정책에서 AWS 전체 조건 키를 사용하여 조건을 표현할 수 있습니다. AWS 전체 키의 완전한 목록은 IAM 사용 설명서의 AWS 글로벌 및 IAM 조건 컨텍스트 키 단원을 참조하세요.

> **Note**
>
> 작업을 지정하려면 `cloudwatch:` 접두사 다음에 API 작업 이름을 사용합니다. 예를 들어 `cloudwatch:GetMetricData`, `cloudwatch:ListMetrics` 또는 `cloudwatch:*`(모든 CloudWatch 작업의 경우) 같이 지정합니다.

주제

CloudWatch API 작업 및 작업에 필요한 권한

CloudWatch API 작업	필요한 권한(API 작업)
DeleteAlarms	`cloudwatch:DeleteAlarms` 경보를 삭제하는 데 필요합니다.
DeleteDashboards	`cloudwatch:DeleteDashboards` 대시보드를 삭제하는 데 필요합니다.
DeleteMetricStream	`cloudwatch:DeleteMetricStream` 지표 스트림을 삭제하는 데 필요합니다.
DescribeAlarmHistory	`cloudwatch:DescribeAlarmHistory` 경보 기록을 보는 데 필요합니다. 복합 경보에 관한 정보를 검색하려면 `cloudwatch:DescribeAlarmHistory` 권한의 범위가 `*`여야 합니다. `cloudwatch:DescribeAlarmHistory` 권한의 범위가 더 좁은 경우 복합 경보에 관한 정보를 반환할 수 없습니다.

CloudWatch API 작업	필요한 권한(API 작업)
DescribeAlarms	`cloudwatch:DescribeAlarms` 경보에 관한 정보를 검색하는 데 필요합니다. 복합 경보에 관한 정보를 검색하려면 `cloudwatch:DescribeAlarms` 권한의 범위가 `*`여야 합니다. `cloudwatch:DescribeAlarms` 권한의 범위가 더 좁은 경우 복합 경보에 관한 정보를 반환할 수 없습니다.
DescribeAlarmsForMetric	`cloudwatch:DescribeAlarmsForMetric` 지표에 대한 경보를 보는 데 필요합니다.
DisableAlarmActions	`cloudwatch:DisableAlarmActions` 경보 작업을 비활성화하는 데 필요합니다.
EnableAlarmActions	`cloudwatch:EnableAlarmActions` 경보 작업을 활성화하는 데 필요합니다.
GetDashboard	`cloudwatch:GetDashboard` 기존 대시보드에 대한 데이터를 표시하는 데 필요합니다.
GetMetricData	`cloudwatch:GetMetricData` CloudWatch 콘솔에서 지표 데이터를 그래프로 표시하고 지표 데이터의 대규모 배치를 검색하며 해당 데이터에 대해 지표 수학을 수행하는 데 필요합니다.

CloudWatch API 작업	필요한 권한(API 작업)
GetMetricStatistics	`cloudwatch:GetMetricStatistics` CloudWatch 콘솔의 다른 부분과 대시보드 위젯에서 그래프를 보는 데 필요합니다.
GetMetricStream	`cloudwatch:GetMetricStream` 지표 스트림에 관한 정보를 보는 데 필요합니다.
GetMetricWidgetImage	`cloudwatch:GetMetricWidgetImage` CloudWatch 지표 1개 이상의 스냅샷 그래프를 비트맵 이미지로 가져오는 데 필요합니다.
ListDashboards	`cloudwatch:ListDashboards` 계정의 CloudWatch 대시보드 목록을 보는 데 필요합니다.
ListMetrics	`cloudwatch:ListMetrics` CloudWatch 콘솔 및 CLI에서 지표 이름을 보거나 검색하는 데 필요합니다. 대시보드 위젯에서 지표를 선택하는 데 필요합니다.
ListMetricStreams	`cloudwatch:ListMetricStreams` 계정의 지표 스트림 목록을 보거나 검색하는 데 필요합니다.

CloudWatch API 작업	필요한 권한(API 작업)
PutCompositeAlarm	`cloudwatch:PutCompositeAlarm` 복합 경보를 생성하는 데 필요합니다. 복합 경보를 생성하려면 `cloudwatch:PutCompositeAlarm` 권한의 범위가 `*`여야 합니다. `cloudwatch:PutCompositeAlarm` 권한의 범위가 더 좁은 경우 복합 경보에 관한 정보를 반환할 수 없습니다.
PutDashboard	`cloudwatch:PutDashboard` 대시보드를 생성하거나 기존 대시보드를 업데이트하는 데 필요합니다.
PutMetricAlarm	`cloudwatch:PutMetricAlarm` 경보를 생성 또는 업데이트하는 데 필요합니다.
PutMetricData	`cloudwatch:PutMetricData` 지표를 만드는 데 필요합니다.
PutMetricStream	`cloudwatch:PutMetricStream` 지표 스트림을 생성하는 데 필요합니다.
SetAlarmState	`cloudwatch:SetAlarmState` 경보 상태를 수동으로 설정하는 데 필요합니다.

CloudWatch API 작업	필요한 권한(API 작업)
StartMetricStreams	`cloudwatch:StartMetricStreams` 지표 스트림의 지표 흐름을 시작하는 데 필요합니다.
StopMetricStreams	`cloudwatch:StopMetricStreams` 지표 스트림의 지표 흐름을 일시적으로 중지하는 데 필요합니다.
TagResource	`cloudwatch:TagResource` 경보 및 Contributor Insights 규칙과 같은 CloudWatch 리소스에서 태그를 추가하거나 업데이트하는 데 필요합니다.
UntagResource	`cloudwatch:UntagResource` CloudWatch 리소스에서 태그를 제거하는 데 필요합니다.

CloudWatch Contributor Insights API 작업 및 작업에 필요한 권한

Important

사용자에게 `cloudwatch:PutInsightRule` 권한을 부여하면 기본적으로 해당 사용자는 CloudWatch Logs의 로그 그룹을 평가하는 규칙을 생성할 수 있습니다. 이러한 권한을 제한하는 IAM 정책 조건을 추가하여 사용자가 특정 로그 그룹을 포함하고 제외하도록 할 수 있습니다. 자세한 내용은 조건 키를 사용하여 Contributor Insights 사용자의 로그 그룹 액세스 제한 단원을 참조하세요.

CloudWatch Contributor Insights API 작업	필요한 권한(API 작업)
DeleteInsightRules	`cloudwatch:DeleteInsightRules` Contributor Insights 규칙을 삭제하는 데 필요합니다.
DescribeInsightRules	`cloudwatch:DescribeInsightRules` 계정의 Contributor Insights 규칙을 보는 데 필요합니다.
EnableInsightRules	`cloudwatch:EnableInsightRules` Contributor Insights 규칙을 사용 설정하는 데 필요합니다.
GetInsightRuleReport	`cloudwatch:GetInsightRuleReport` Contributor Insights 규칙이 수집한 시계열 데이터 및 기타 통계를 검색하는 데 필요합니다.
PutInsightRule	`cloudwatch:PutInsightRule` Contributor Insights 규칙을 생성하는 데 필요합니다. 이 표의 시작 부분에 있는 중요 참고 사항을 참조하세요.

CloudWatch Events API 작업 및 작업에 필요한 권한

CloudWatch Events API 작업	필요한 권한(API 작업)
DeleteRule	`events:DeleteRule`

CloudWatch Events API 작업	필요한 권한(API 작업)
	규칙을 삭제하는 데 필요합니다.
DescribeRule	`events:DescribeRule` 규칙에 대한 세부 사항을 나열하는 데 필요합니다.
DisableRule	`events:DisableRule` 규칙을 비활성화하는 데 필요합니다.
EnableRule	`events:EnableRule` 규칙을 활성화하는 데 필요합니다.
ListRuleNamesByTarget	`events:ListRuleNamesByTarget` 대상과 연관된 규칙을 나열하는 데 필요합니다.
ListRules	`events:ListRules` 계정에서 모든 그룹을 나열하는 데 필요합니다.
ListTargetsByRule	`events:ListTargetsByRule` 규칙과 연관된 모든 대상을 나열하는 데 필요합니다.
PutEvents	`events:PutEvents` 규칙에 일치시킬 수 있는 사용자 지정 이벤트를 추가하는 데 필요합니다.

CloudWatch Events API 작업	필요한 권한(API 작업)
PutRule	`events:PutRule` 규칙을 생성 또는 업데이트하는 데 필요합니다.
PutTargets	`events:PutTargets` 규칙에 대상을 추가하는 데 필요합니다.
RemoveTargets	`events:RemoveTargets` 규칙에서 대상을 제거하는 데 필요합니다.
TestEventPattern	`events:TestEventPattern` 특정 이벤트를 기준으로 이벤트 패턴을 테스트하는 데 필요합니다.

CloudWatch Logs API 작업 및 작업에 필요한 권한

CloudWatch Logs API 작업	필요한 권한(API 작업)
CancelExportTask	`logs:CancelExportTask` 보류 또는 실행 중인 내보내기 작업을 취소하는 데 필요합니다.
CreateExportTask	`logs:CreateExportTask` 로그 그룹에서 Amazon S3 버킷으로 데이터를 내보내는 데 필요합니다.
CreateLogGroup	`logs:CreateLogGroup`

CloudWatch Logs API 작업	필요한 권한(API 작업)
	새 보안 그룹을 생성하는 데 필요합니다.
CreateLogStream	`logs:CreateLogStream` 로그 그룹에서 로그 스트림을 새로 생성하는 데 필요합니다.
DeleteDestination	`logs:DeleteDestination` 로그 대상을 삭제하고 이에 대한 모든 구독 필터를 비활성화하는 데 필요합니다.
DeleteLogGroup	`logs:DeleteLogGroup` 로그 그룹과 보관되는 모든 연관 로그 이벤트를 삭제하는 데 필요합니다.
DeleteLogStream	`logs:DeleteLogStream` 로그 스트림과 보관되는 모든 연관 로그 이벤트를 삭제하는 데 필요합니다.
DeleteMetricFilter	`logs:DeleteMetricFilter` 로그 그룹과 연관된 지표 필터를 삭제하는 데 필요합니다.
DeleteQueryDefinition	`logs:DeleteQueryDefinition` CloudWatch Logs Insights에서 저장된 쿼리 정의를 삭제하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
DeleteResourcePolicy	`logs:DeleteResourcePolicy` CloudWatch Logs 리소스 정책을 삭제하는 데 필요합니다.
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` 로그 그룹의 보존 정책을 삭제하는 데 필요합니다.
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` 로그 그룹과 연관된 구독 필터를 삭제하는 데 필요합니다.
DescribeDestinations	`logs:DescribeDestinations` 계정과 연결된 모든 대상을 보는 데 필요합니다.
DescribeExportTasks	`logs:DescribeExportTasks` 계정과 연관된 모든 내보내기 작업을 보는 데 필요합니다.
DescribeLogGroups	`logs:DescribeLogGroups` 계정과 연관된 모든 로그 그룹들을 보는 데 필요합니다.
DescribeLogStreams	`logs:DescribeLogStreams` 로그 그룹과 연관된 모든 로그 스트림을 보는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
DescribeMetricFilters	`logs:DescribeMetricFilters` 로그 그룹과 연관된 모든 지표를 보는 데 필요합니다.
DescribeQueryDefinitions	`logs:DescribeQueryDefinitions` CloudWatch Logs Insights에 저장된 쿼리 정의의 목록을 확인하는 데 필요합니다.
DescribeQueries	`logs:DescribeQueries` 예약되었거나 실행 중이거나 최근에 실행된 CloudWatch Logs Insights 쿼리 목록을 확인하는 데 필요합니다.
DescribeResourcePolicies	`logs:DescribeResourcePolicies` CloudWatch Logs 리소스 정책 목록을 보는 데 필요합니다.
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` 로그 그룹과 연관된 모든 구독 필터를 보는 데 필요합니다.
FilterLogEvents	`logs:FilterLogEvents` 로그 그룹 필터 패턴에 따라 로그 이벤트를 정렬하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
GetLogEvents	`logs:GetLogEvents` 로그 스트림에서 로그 이벤트를 검색하는 데 필요합니다.
GetLogGroupFields	`logs:GetLogGroupFields` 로그 그룹의 로그 이벤트에 포함된 필드 목록을 검색하는 데 필요합니다.
GetLogRecord	`logs:GetLogRecord` 단일 로그 이벤트에서 세부 정보를 검색하는 데 필요합니다.
GetQueryResults	`logs:GetQueryResults` CloudWatch Logs Insights 쿼리 결과를 검색하는 데 필요합니다.
ListTagsLogGroup	`logs:ListTagsLogGroup` 로그 그룹과 연결된 태그를 나열하는 데 필요합니다.
PutDestination	`logs:PutDestination` 대상 로그 스트림(예: Kinesis 스트림)을 생성하거나 업데이트하는 데 필요합니다.
PutDestinationPolicy	`logs:PutDestinationPolicy` 기존 로그 대상과 연관된 액세스 정책을 생성 또는 업데이트하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
PutLogEvents	`logs:PutLogEvents` 로그 스트림에서 로그 이벤트 배치를 업로드하는 데 필요합니다.
PutMetricFilter	`logs:PutMetricFilter` 지표 필터를 생성 또는 업데이트하고 이를 로그 그룹과 연관시키는 데 필요합니다.
PutQueryDefinition	`logs:PutQueryDefinition` CloudWatch Logs Insights에 쿼리를 저장하는 데 필요합니다.
PutResourcePolicy	`logs:PutResourcePolicy` CloudWatch Logs 리소스 정책을 생성하는 데 필요합니다.
PutRetentionPolicy	`logs:PutRetentionPolicy` 로그 그룹에 로그 이벤트를 유지하는 일수(보존 일수)를 설정하는 데 필요합니다.
PutSubscriptionFilter	`logs:PutSubscriptionFilter` 구독 필터를 생성 또는 업데이트하고 이를 로그 그룹과 연관시키는 데 필요합니다.
StartQuery	`logs:StartQuery` CloudWatch Logs Insights 쿼리를 시작하는 데 필요합니다.

CloudWatch Logs API 작업	필요한 권한(API 작업)
StopQuery	`logs:StopQuery` 진행 중인 CloudWatch Logs Insights 쿼리를 중지하는 데 필요합니다.
TagLogGroup	`logs:TagLogGroup` 로그 그룹 태그를 추가하거나 업데이트하는 데 필요합니다.
TestMetricFilter	`logs:TestMetricFilter` 로그 이벤트 메시지 샘플을 기준으로 필터 패턴을 테스트하는 데 필요합니다.

Amazon EC2 API 작업 및 작업에 필요한 권한

Amazon EC2 API 작업	필요한 권한(API 작업)
DescribeInstanceStatus	`ec2:DescribeInstanceStatus` EC2 인스턴스 상태에 대한 세부 정보를 보는 데 필요합니다.
DescribeInstances	`ec2:DescribeInstances` EC2 인스턴스에 대한 세부 정보를 보는 데 필요합니다.
RebootInstances	`ec2:RebootInstances` EC2 인스턴스를 재부팅하는 데 필요합니다.

Amazon EC2 API 작업	필요한 권한(API 작업)
StopInstances	`ec2:StopInstances` EC2 인스턴스를 중지하는 데 필요합니다.
TerminateInstances	`ec2:TerminateInstances` EC2 인스턴스를 종료하는 데 필요합니다.

Amazon EC2 Auto Scaling API 작업 및 작업에 필요한 권한

Amazon EC2 Auto Scaling API 작업	필요한 권한(API 작업)
스케일링	`autoscaling:Scaling` Auto Scaling 그룹의 크기를 조정하는 데 필요합니다.
트리거	`autoscaling:Trigger` Auto Scaling 작업을 트리거하는 데 필요합니다.

Amazon CloudWatch 규정 준수 검증

서드 파티 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 Amazon CloudWatch의 보안 및 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램의 범위 내에 있는 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하세요.

Amazon CloudWatch 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 해당 법률 및 규정에 따라 결정됩니다. AWS는 규정 준수를 지원하기 위해 다음과 같은 리소스를 제공합니다.

- 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 준수 기술 백서 아키텍팅 - 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 콜렉션입니다.
- AWS Config 개발자 가이드의 규칙을 사용하여 리소스 평가 – AWS Config를 사용하여 리소스 구성이 내부 사례, 업계 지침, 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub - 이 AWS 서비스는 보안 산업 표준 및 모범 사례 규정 준수 여부를 확인하는 데 도움이 되도록 AWS 내 보안 상태를 종합적으로 보여줍니다.

Amazon CloudWatch의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 정보는 AWS 글로벌 인프라를 참조하세요.

Amazon CloudWatch의 인프라 보안

관리형 서비스인 Amazon CloudWatch는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 CloudWatch에 액세스합니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

네트워크 격리

Virtual Private Cloud(VPC)는 Amazon Web Services 클라우드에서 논리적으로 격리된 자체 영역에 있는 가상 네트워크입니다. 서브넷은 VPC의 IP 주소 범위입니다. VPC의 서브넷에 다양한 AWS 리소스를 배포할 수 있습니다. 예를 들어 서브넷에 Amazon EC2 인스턴스, EMR 클러스터, DynamoDB 테이블을 배포할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

CloudWatch가 퍼블릭 인터넷을 통하지 않고 VPC의 리소스와 통신할 수 있도록 하려면 AWS PrivateLink를 사용합니다. 자세한 내용은 인터페이스 VPC 엔드포인트와 함께 CloudWatch 및 CloudWatch Synthetics 사용하기 섹션을 참조하세요.

프라이빗 서브넷은 퍼블릭 인터넷에 대한 기본 경로가 없는 서브넷입니다. 프라이빗 서브넷에 AWS 리소스를 배포해도 Amazon CloudWatch가 리소스에서 기본 제공 지표를 수집하지 못하게 되지 않습니다.

프라이빗 서브넷에 AWS 리소스의 사용자 지정 지표를 게시해야 하는 경우 프록시 서버를 사용하여 게시할 수 있습니다. 프록시 서버는 이러한 HTTPS 요청을 CloudWatch의 퍼블릭 API 엔드포인트로 전달합니다.

AWS Security Hub

AWS 보안 허브를 사용하여 보안 모범 사례와 관련된 CloudWatch 사용 현황을 모니터합니다. Security Hub는 보안 제어를 사용하여 리소스 구성 및 보안 표준을 평가하여 다양한 규정 준수 프레임워크를 준수할 수 있도록 지원합니다. AWS 보안 허브를 사용하여 CloudWatch 리소스를 평가하는 방법에 대한 자세한 내용은 보안 허브 사용 가이드에서 Amazon CloudWatch controls를 참조하세요.

인터페이스 VPC 엔드포인트와 함께 CloudWatch 및 CloudWatch Synthetics 사용하기

Amazon Virtual Private Cloud(Amazon VPC)를 사용하여 AWS 리소스를 호스트하는 경우 VPC, CloudWatch 및 CloudWatch Synthetics 간에 프라이빗 연결을 설정할 수 있습니다. 이러한 연결을 사용하면 CloudWatch 및 CloudWatch Synthetics가 퍼블릭 인터넷을 통하지 않고 VPC의 리소스와 통신하도록 지원할 수 있습니다.

Amazon VPC는 직접 정의한 가상 네트워크에서 AWS 리소스를 시작하는 데 사용할 수 있는 AWS 서비스입니다. VPC가 있으면 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 등 네트워크 설정을 제어할 수 있습니다. VPC를 CloudWatch 또는 CloudWatch Synthetics에 연결하려면 '인터페이스 VPC 엔드포인트'를 정의하여 VPC를 AWS 서비스에 연결하면 됩니다. 엔드포인트는 인터넷 게이트웨이, 네트워크 주소 변환(NAT) 인스턴스 또는 VPN 연결 없이도 CloudWatch 또는 CloudWatch Synthetics에 대한 안정적이고 확장 가능한 연결을 제공합니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇인가요? 단원을 참조하세요.

인터페이스 VPC 엔드포인트는 프라이빗 IP 주소와 함께 탄력적 네트워크 인터페이스를 사용하여 AWS 서비스 간 프라이빗 통신을 사용할 수 있는 AWS 기술인 AWS PrivateLink에 의해 구동됩니다. 자세한 내용은 새 기능 – AWS 서비스를 위한 AWS PrivateLink 블로그 게시물을 참조하세요.

다음은 Amazon VPC 사용자를 위한 단계들입니다. 자세한 내용은 시작하기(출처: Amazon VPC 사용 설명서)를 참조하세요.

CloudWatch VPC 엔드포인트

현재 CloudWatch가 VPC 엔드포인트를 지원하는 AWS 리전은 다음과 같습니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 아시아 태평양(홍콩)
- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- Europe (London)
- 유럽(파리)
- 중동(UAE)

- 남아메리카(상파울루)
- AWS GovCloud(미국 동부)
- AWS GovCloud(미국 서부)

CloudWatch VPC 엔드포인트 생성하기

VPC에서 CloudWatch를 사용하려면 CloudWatch용 인터페이스 VPC 엔드포인트를 생성해야 합니다. 선택할 서비스 이름은 `com.amazonaws.region.monitoring`입니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성 단원을 참조하세요.

CloudWatch의 설정을 변경할 필요는 없습니다. CloudWatch는 퍼블릭 엔드포인트 또는 프라이빗 인터페이스 VPC 엔드포인트 중에서 현재 사용 중인 엔드포인트를 사용해 다른 AWS 서비스를 호출합니다. 예를 들어 CloudWatch용 인터페이스 VPC 엔드포인트를 생성할 때 VPC에 있는 리소스에서 CloudWatch로 흐르는 지표가 이미 있는 경우 이러한 지표는 기본적으로 인터페이스 VPC 엔드포인트를 통해 흐르기 시작합니다.

CloudWatch VPC 엔드포인트 액세스 제어하기

VPC 엔드포인트 정책은 엔드포인트를 만들거나 수정 시 엔드포인트에 연결하는 IAM 리소스 정책입니다. 엔드포인트를 생성할 때 정책을 연결하지 않으면 Amazon VPC는 서비스에 대한 전체 액세스를 허용하는 기본 정책을 자동으로 연결합니다. 엔드포인트 정책은 사용자 정책 또는 서비스별 정책을 무시하거나 교체하지 않습니다. 이는 엔드포인트에서 지정된 서비스로의 액세스를 제어하기 위한 별도의 정책입니다.

엔드포인트 정책은 JSON 형식으로 작성해야 합니다.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

다음은 CloudWatch에 대한 엔드포인트 정책의 예입니다. 이 정책은 VPC를 통해 CloudWatch에 연결하는 사용자가 지표 데이터를 CloudWatch에 전송하도록 허용하며 다른 CloudWatch 작업을 수행하지 못하게 방지합니다.

```
{
  "Statement": [
    {
      "Sid": "PutOnly",
      "Principal": "*",
      "Action": [
```

```
        "cloudwatch:PutMetricData"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

CloudWatch에 대한 VPC 엔드포인트 정책을 편집하려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. CloudWatch용 엔드포인트를 아직 생성하지 않았다면 [엔드포인트 생성(Create Endpoint)]을 선택합니다. com.amazonaws.***region***.monitoring을 선택한 다음 엔드포인트 생성을 선택합니다.
4. com.amazonaws.***region***.monitoring을 선택한 다음 정책 탭을 선택합니다.
5. 정책 편집을 선택한 다음 변경합니다.

CloudWatch Synthetics VPC 엔드포인트

현재 CloudWatch Synthetics가 VPC 엔드포인트를 지원하는 AWS 리전은 다음과 같습니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 아시아 태평양(홍콩)
- 아시아 태평양(뭄바이)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- Europe (London)

- 유럽(파리)
- 남아메리카(상파울루)

CloudWatch Synthetics VPC 엔드포인트 생성하기

VPC에서 CloudWatch Synthetics를 사용하려면 CloudWatch Synthetics용 인터페이스 VPC 엔드포인트를 생성해야 합니다. 선택할 서비스 이름은 `com.amazonaws.region.synthetics`입니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성 단원을 참조하세요.

CloudWatch Synthetics의 설정을 변경할 필요는 없습니다. CloudWatch Synthetics는 퍼블릭 엔드포인트 또는 프라이빗 인터페이스 VPC 엔드포인트 중에서 현재 사용 중인 엔드포인트를 사용해 다른 AWS 서비스와 통신합니다. 예를 들어 CloudWatch Synthetics용 인터페이스 VPC 엔드포인트를 생성할 때 Amazon S3용 인터페이스 엔드포인트가 이미 있는 경우 CloudWatch Synthetics는 기본적으로 인터페이스 VPC 엔드포인트를 통해 Amazon S3와 통신을 시작합니다.

CloudWatch Synthetics VPC 엔드포인트 액세스 제어하기

VPC 엔드포인트 정책은 엔드포인트를 만들거나 수정 시 엔드포인트에 연결하는 IAM 리소스 정책입니다. 엔드포인트를 만들 때 정책을 추가하지 않으면 서비스에 대한 모든 액세스를 허용하는 기본 정책이 추가됩니다. 엔드포인트 정책은 사용자 정책 또는 서비스별 정책을 무시하거나 교체하지 않습니다. 이는 엔드포인트에서 지정된 서비스로의 액세스를 제어하기 위한 별도의 정책입니다.

엔드포인트 정책은 VPC에서 비공개로 관리되는 canary에 영향을 줍니다. 프라이빗 서브넷에서 실행되는 canary에는 엔드포인트 정책이 필요하지 않습니다.

엔드포인트 정책은 JSON 형식으로 작성해야 합니다.

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

다음은 CloudWatch Synthetics에 대한 엔드포인트 정책의 예입니다. 이 정책을 사용하면 VPC를 통해 CloudWatch Synthetics에 연결하는 사용자가 canary 및 해당 실행에 관한 정보를 볼 수 있지만 canary를 생성, 수정 또는 삭제할 수는 없습니다.

```
{
    "Statement": [
        {
            "Action": [
                "synthetics:DescribeCanaries",
                "synthetics:GetCanaryRuns"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

CloudWatch Synthetics에 대한 VPC 엔드포인트 정책을 편집하려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. CloudWatch Synthetics용 엔드포인트를 아직 생성하지 않았다면 [엔드포인트 생성(Create Endpoint)]을 선택합니다. com.amazonaws.***region***.synthetics를 선택한 다음 엔드포인트 생성을 선택합니다.
4. com.amazonaws.***region***.synthetics 엔드포인트를 선택한 다음 정책 탭을 선택합니다.
5. 정책 편집을 선택한 다음 변경합니다.

Synthetics canary에 대한 보안 고려 사항

다음 단원에서는 Synthetics에서 카나리아를 생성 및 실행할 때 고려해야 할 보안 문제에 대해 설명합니다.

보안 연결 사용

canary 코드와 canary 테스트 실행의 결과에는 중요한 정보가 포함될 수 있으므로 암호화되지 않은 연결을 통해 엔드포인트에 canary를 연결하지 마십시오. `https://`로 시작하는 것과 같이 항상 암호화된 연결을 사용합니다.

canary 이름 지정 고려 사항

canary의 Amazon 리소스 이름(ARN)은 CloudWatch Synthetics 래퍼 라이브러리의 일부로 포함된 Puppeteer 기반 Chromium 브라우저에서 수행된 아웃바운드 호출의 일부로 user-agent 헤더에 포함됩니다. 이렇게 하면 CloudWatch Synthetics canary 트래픽을 식별하여 호출을 수행 중인 canary에 다시 연결할 수 있습니다.

canary ARN에는 canary 이름이 포함되어 있습니다. 독점 정보를 공개하지 않는 canary 이름을 선택하십시오.

또한 자신이 제어하는 웹 사이트 및 엔드포인트의 canary만 가리키도록 해야 합니다.

canary 코드의 비밀 및 민감한 정보

zip 파일을 사용하여 canary 코드를 canary에 직접 전달하면 스크립트의 내용을 AWS CloudTrail 로그에서 볼 수 있습니다.

canary 스크립트에 민감한 정보 또는 비밀(예: 액세스 키 또는 데이터베이스 자격 증명)이 있는 경우, zip 파일로 canary 코드를 전달하는 대신 스크립트를 Amazon S3에 버전이 지정된 객체로 저장하고 canary에 Amazon S3 위치를 전달하는 것이 좋습니다.

zip 파일을 사용하여 canary 스크립트를 전달하는 경우 canary 소스 코드에 비밀이나 민감한 정보를 포함하지 않는 것이 좋습니다. AWS Secrets Manager를 사용하여 암호를 안전하게 유지하는 방법에 대한 자세한 내용은 AWS Secrets Manager란 무엇입니까?를 참조하십시오.

권한 고려 사항

CloudWatch Synthetics가 생성하거나 사용하는 리소스에 대한 액세스를 제한하는 것이 좋습니다. canary가 테스트 실행 결과와 기타 아티팩트(예: 로그 및 스크린샷)를 저장하는 Amazon S3 버킷에 엄격한 권한을 사용해야 합니다.

마찬가지로 canary 소스 코드가 저장된 위치에 대한 권한을 엄격하게 유지하여 사용자가 canary에 사용된 Lambda 계층 또는 Lambda 함수를 실수로 또는 악의적으로 삭제하지 않도록 해야 합니다.

원하는 canary 코드를 실행할 수 있도록 canary 코드가 저장된 Amazon S3 버킷에서 객체 버전 관리를 사용할 수 있습니다. 그런 다음 이 코드를 canary로 실행하도록 지정하면 다음 예제와 같이 객체 `versionId`를 경로의 일부로 포함할 수 있습니다.

```
https://bucket.s3.amazonaws.com/path/object.zip?versionId=version-id
https://s3.amazonaws.com/bucket/path/object.zip?versionId=version-id
https://bucket.s3-region.amazonaws.com/path/object.zip?versionId=version-id
```

스택 추적 및 예외 메시지

기본적으로 CloudWatch Synthetics canary는 스크립트가 사용자 지정인지 아니면 블루프린트인지에 상관없이 canary 스크립트에서 발생하는 모든 예외를 캡처합니다. CloudWatch Synthetics는 예외 메시지와 스택 추적을 모두 다음 세 위치에 로그합니다.

- 테스트 실행을 설명할 때 디버깅 속도를 높이기 위해 CloudWatch Synthetics 서비스에 다시 로그
- Lambda 함수가 생성되는 구성에 따라 CloudWatch Logs에 로그

- canary의 `resultsLocation`에 대해 설정한 값으로 지정된 Amazon S3 위치에 업로드되는 일반 텍스트 파일인 Synthetics 로그 파일에 로그

더 적은 정보를 보내고 저장하려는 경우 예외가 CloudWatch Synthetics 래퍼 라이브러리로 반환되기 전에 예외를 캡처할 수 있습니다.

또한 오류에 요청 URL이 있을 수도 있습니다. CloudWatch Synthetics는 스크립트에서 발생한 오류의 URL을 검색하며 [restrictedUrlParameters] 구성에 따라 제한된 URL 파라미터를 수정합니다. 스크립트에 오류 메시지를 로그하는 경우 getSanitizedErrorMessage 를 사용하여 로깅 전에 URL을 수정할 수 있습니다.

IAM 역할의 범위를 좁게 지정

잠재적인 악성 URL 또는 엔드포인트를 방문하도록 canary를 구성하지 않는 것이 좋습니다. canary가 신뢰할 수 없거나 알 수 없는 웹 사이트 또는 엔드포인트를 가리키면 Lambda 함수 코드가 악의적인 사용자의 스크립트에 노출될 수 있습니다. 악성 웹 사이트가 Chromium에서 벗어날 수 있다고 가정하면 인터넷 브라우저를 사용하여 웹 사이트에 연결한 경우와 비슷한 방식으로 Lambda 코드에 액세스할 수 있습니다.

따라서 권한 범위가 축소된 IAM 실행 역할로 Lambda 함수를 실행해야 합니다. 이렇게 하면 Lambda 함수가 악성 스크립트에 의해 손상된 경우 canary의 AWS 계정으로 실행할 때 수행할 수 있는 작업이 제한됩니다.

CloudWatch 콘솔을 사용하여 canary를 생성하면 범위가 축소된 IAM 실행 역할로 생성됩니다.

민감한 데이터 수정

CloudWatch Synthetics는 요청 및 응답의 URL, 상태 코드, 실패 원인(있는 경우), 헤더 및 본문을 캡처합니다. 이를 통해 canary 사용자는 canary를 파악하고 모니터링하며 디버그할 수 있습니다.

다음 단원에서 설명하는 구성은 canary 실행의 어느 시점에서나 설정할 수 있습니다. 또한 다양한 Synthetics 단계에 서로 다른 구성을 적용하도록 선택할 수도 있습니다.

요청 URL

기본적으로 CloudWatch Synthetics는 canary 로그에 요청 URL, 상태 코드, 각 URL의 상태 이유를 로그합니다. 요청 URL은 canary 실행 보고서, HAR 파일 등에 표시될 수도 있습니다. 요청 URL에는 액세스 토큰 또는 암호와 같은 민감한 쿼리 파라미터가 포함될 수 있습니다. 민감한 정보를 CloudWatch Synthetics에서 로그하지 못하게 수정할 수 있습니다.

민감한 정보를 수정하려면 구성 속성인 [restrictedUrlParameters]를 설정합니다. 자세한 내용은 SyntheticsConfiguration 클래스 단원을 참조하세요. 이렇게 하면 CloudWatch Synthetics는 로깅 전에 [restrictedUrlParameters]에 따라 경로 및 쿼리 파라미터 값을 포함한 URL 파라미터를 수정합니다. 스크립트에 URL을 로그하는 경우 getSanitizedUrl(url, stepConfig = null)을 사용하여 로깅 전에 URL을 수정할 수 있습니다. 자세한 내용은 SyntheticsLogHelper 클래스 섹션을 참조하세요.

헤더

기본적으로 CloudWatch Synthetics는 요청 또는 응답 헤더를 로그하지 않습니다. UI canary의 경우 이는 런타임 버전 `syn-nodejs-puppeteer-3.2` 이상을 사용하는 canary의 기본 동작입니다.

헤더에 민감한 정보가 포함되어 있지 않으면 [includeRequestHeaders] 및 [includeResponseHeaders] 속성을 `true`로 설정하여 HAR 파일 및 HTTP 보고서에서 헤더를 사용 설정할 수 있습니다. 모든 헤더를 사용 설정할 수 있지만, 민감한 헤더 키의 값을 제한하도록 선택할 수 있습니다. 예를 들어 canary가 생성한 아티팩트의 `Authorization` 헤더만 수정하도록 선택할 수 있습니다.

요청 및 응답 본문

기본적으로 CloudWatch Synthetics는 canary 로그 또는 보고서에 요청 또는 응답 본문을 로그하지 않습니다. 이 정보는 API canary에 특히 유용합니다. Synthetics는 모든 HTTP 요청을 캡처하고 헤더, 요청 및 응답 본문을 표시할 수 있습니다. 자세한 내용은 executeHttpStep(stepName, requestOptions, [callback], [stepConfig]) 단원을 참조하세요. [includeRequestBody] 및 [includeResponseBody] 속성을 `true`로 설정하여 요청 및 응답 본문을 사용하도록 선택할 수 있습니다.

AWS CloudTrail으로 Amazon CloudWatch API 호출 로그하기

Amazon CloudWatch 및 CloudWatch Synthetics는 사용자, 역할 또는 AWS 서비스가 수행한 작업의 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 AWS 계정에 의해 또는 이 계정을 대신하여 수행된 API 호출을 캡처합니다. 캡처되는 호출에는 콘솔에서 수행한 호출과 API 작업에 대한 코드 호출이 포함됩니다.

'추적'을 생성하면 CloudWatch에 대한 이벤트를 포함하여 CloudTrail 이벤트를 S3 버킷에 지속적으로 전달할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail이 수집한 정보를 사용하여 CloudWatch에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사용자, 요청이 수행된 시간 및 기타 세부 정보를 확인할 수 있습니다.

구성 및 사용 방법을 포함하여 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 AWS Identity and Access Management (IAM) 사용자 보안 인증으로 했는지.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지.
- 다른 AWS 서비스에서 요청했는지.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

CloudWatch 및 CloudWatch Synthetics에 대한 이벤트를 포함하여 AWS 계정의 이벤트를 지속적으로 기록하려면 추적을 생성합니다. 추적은 CloudTrail이 S3 버킷으로 로그 파일을 전송할 수 있도록 합니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 S3 버킷으로 로그 파일을 전송합니다. 다른 AWS 서비스를 구성하여 CloudTrail 로그에 수집된 이벤트 데이터를 추가로 분석하고 조치를 취할 수 있습니다. 자세한 정보는 을(를) 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성

- 여러 지역으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

> **Note**
>
> CloudTrail에 기록된 CloudWatch Logs API 호출에 대한 자세한 내용은 CloudWatch Logs information in CloudTrail를 참조하세요.

주제

- CloudTrail의 CloudWatch 정보
- CloudTrail의 CloudWatch Internet Monitor
- CloudTrail의 CloudWatch Synthetics 정보

CloudTrail의 CloudWatch 정보

CloudWatch는 CloudTrail 로그 파일에 다음 작업을 이벤트로 로그하도록 지원합니다.

- DeleteAlarms
- DeleteAnomalyDetector
- DeleteDashboards
- DescribeAlarmHistory
- DescribeAlarms
- DescribeAlarmsForMetric
- DescribeAnomalyDetectors
- DisableAlarmActions
- EnableAlarmActions
- GetDashboard
- ListDashboards
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- SetAlarmState

예: CloudWatch 로그 파일 항목

다음은 PutMetricAlarm 작업을 설명하는 CloudTrail 로그 항목을 보여 주는 예제입니다.

```
{
    "Records": [{
        "eventVersion": "1.01",
        "userIdentity": {
            "type": "Root",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:root",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID"
        },
        "eventTime": "2014-03-23T21:50:34Z",
        "eventSource": "monitoring.amazonaws.com",
        "eventName": "PutMetricAlarm",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "threshold": 50.0,
            "period": 60,
            "metricName": "CloudTrail Test",
            "evaluationPeriods": 3,
            "comparisonOperator": "GreaterThanThreshold",
            "namespace": "AWS/CloudWatch",
            "alarmName": "CloudTrail Test Alarm",
            "statistic": "Sum"
        },
        "responseElements": null,
        "requestID": "29184022-b2d5-11e3-a63d-9b463e6d0ff0",
        "eventID": "b096d5b7-dcf2-4399-998b-5a53eca76a27"
    },
    ..additional entries
  ]
  }
```

다음 로그 파일 항목은 사용자가 CloudWatch Events PutRule 작업을 호출했음을 보여 줍니다.

```
{
         "eventVersion":"1.03",
         "userIdentity":{
```

```
            "type":"Root",
            "principalId":"123456789012",
            "arn":"arn:aws:iam::123456789012:root",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext":{
               "attributes":{
                  "mfaAuthenticated":"false",
                  "creationDate":"2015-11-17T23:56:15Z"
               }
            }
         },
         "eventTime":"2015-11-18T00:11:28Z",
         "eventSource":"events.amazonaws.com",
         "eventName":"PutRule",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"AWS Internal",
         "userAgent":"AWS CloudWatch Console",
         "requestParameters":{
            "description":"",
            "name":"cttest2",
            "state":"ENABLED",
            "eventPattern":"{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance
 State-change Notification\"]}",
            "scheduleExpression":""
         },
         "responseElements":{
            "ruleArn":"arn:aws:events:us-east-1:123456789012:rule/cttest2"
         },
         "requestID":"e9caf887-8d88-11e5-a331-3332aa445952",
         "eventID":"49d14f36-6450-44a5-a501-b0fdcdfaeb98",
         "eventType":"AwsApiCall",
         "apiVersion":"2015-10-07",
         "recipientAccountId":"123456789012"
}
```

다음 로그 파일 항목은 사용자가 CloudWatch Logs `CreateExportTask` 작업을 호출했음을 보여 줍니다.

```
{
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
```

```
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/someuser",
            "accountId": "123456789012",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "someuser"
        },
        "eventTime": "2016-02-08T06:35:14Z",
        "eventSource": "logs.amazonaws.com",
        "eventName": "CreateExportTask",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "destination": "yourdestination",
            "logGroupName": "yourloggroup",
            "to": 123456789012,
            "from": 0,
            "taskName": "yourtask"
        },
        "responseElements": {
            "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
        },
        "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
        "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
        "eventType": "AwsApiCall",
        "apiVersion": "20140328",
        "recipientAccountId": "123456789012"
}
```


CloudTrail의 CloudWatch Internet Monitor

CloudWatch Internet Monitor는 CloudTrail 로그 파일에 다음 작업을 이벤트로 로그하도록 지원합니다.

- CreateMonitor
- DeleteMonitor
- GetHealthEvent
- GetMonitor
- GetQueryResults
- GetQueryStatus
- ListHealthEvents

- ListMonitors
- ListTagsForResource
- StartQuery
- StopQuery
- UpdateMonitor

예: CloudWatch Internet Monitor 로그 파일 항목

다음 예제에서는 `ListMonitors` 작업에 대한 CloudTrail Internet Monitor 로그 항목을 보여줍니다.


```
{
        "eventVersion": "1.08",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::000000000000:assumed-role/role_name",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::000000000000:role/Admin",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
                },
                "webIdFederationData": {},
                "attributes": {
                    "creationDate": "2022-10-11T17:25:41Z",
                    "mfaAuthenticated": "false"
                }
            }
        },
        "eventTime": "2022-10-11T17:30:18Z",
        "eventSource": "internetmonitor.amazonaws.com",
        "eventName": "ListMonitors",
        "awsRegion": "us-east-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)",
        "requestParameters": null,
        "responseElements": null,
```

```
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
    }
```

다음 예제에서는 CreateMonitor 작업에 대한 CloudTrail Internet Monitor 로그 항목을 보여줍니다.

```
{
        "eventVersion": "1.08",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::000000000000:assumed-role/role_name",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::000000000000:role/Admin",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
                },
                "webIdFederationData": {},
                "attributes": {
                    "creationDate": "2022-10-11T17:25:41Z",
                    "mfaAuthenticated": "false"
                }
            }
        },
        "eventTime": "2022-10-11T17:30:08Z",
        "eventSource": "internetmonitor.amazonaws.com",
        "eventName": "CreateMonitor",
        "awsRegion": "us-east-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)",
        "requestParameters": {
            "MonitorName": "TestMonitor",
            "Resources": ["arn:aws:ec2:us-east-2:444455556666:vpc/vpc-febc0b95"],
```

```
            "ClientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
        },
        "responseElements": {
            "Arn": "arn:aws:internetmonitor:us-east-2:444455556666:monitor/ct-
onboarding-test",
            "Status": "PENDING"
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
    }
```


CloudTrail의 CloudWatch Synthetics 정보

CloudWatch Synthetics는 CloudTrail 로그 파일에 다음 작업을 이벤트로 로그하도록 지원합니다.

- CreateCanary
- DeleteCanary
- DescribeCanaries
- DescribeCanariesLastRun
- DescribeRuntimeVersions
- GetCanary
- GetCanaryRuns
- ListTagsForResource
- StartCanary
- StopCanary
- TagResource
- UntagResource
- UpdateCanary

예: CloudWatch Synthetics 로그 파일 항목

다음 CloudTrail Synthetics 로그 항목 예에서는 `DescribeCanaries` 작업을 보여 줍니다.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:47Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "DescribeCanaries",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "201ed5f3-15db-4f87-94a4-123456789",
    "eventID": "73ddbd81-3dd0-4ada-b246-123456789",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

다음 CloudTrail Synthetics 로그 항목 예에서는 UpdateCanary 작업을 보여 줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
               "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
       "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:47Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "UpdateCanary",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": {
        "Schedule": {
            "Expression": "rate(1 minute)"
        },
        "name": "sample_canary_name",
        "Code": {
            "Handler": "myOwnScript.handler",
            "ZipFile": "SAMPLE_ZIP_FILE"
        }
    },
    "responseElements": null,
```

```
    "requestID": "fe4759b0-0849-4e0e-be71-1234567890",
    "eventID": "9dc60c83-c3c8-4fa5-bd02-1234567890",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

다음 CloudTrail Synthetics 로그 항목 예에서는 GetCanaryRuns 작업을 보여 줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
       "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:30Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "GetCanaryRuns",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": {
        "Filter": "TIME_RANGE",
        "name": "sample_canary_name",
```

```
        "FilterValues": [
            "2020-04-08T23:00:00.000Z",
            "2020-04-08T23:10:00.000Z"
        ]
    },
    "responseElements": null,
    "requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
    "eventID": "52723fd9-4a54-478c-ac55-1234567890",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Amazon CloudWatch 리소스 태그 지정

태그는 사용자 또는 AWS에서 AWS 리소스에 할당하는 사용자 지정 속성 레이블입니다. 각 태그에는 다음 두 가지 부분이 있습니다.

- 태그 키 (예: `CostCenter`, `Environment` 또는 `Project`) 태그 키는 대/소문자를 구별합니다.
- 태그 값(예: `111122223333` 또는 `Production`)으로 알려진 선택적 필드 태그 값을 생략하는 것은 빈 문자열을 사용하는 것과 같습니다. 태그 키처럼 태그 값은 대/소문자를 구별합니다.

태그는 다음을 지원합니다.

- AWS 리소스를 식별하고 정리합니다. 많은 AWS 서비스가 태그 지정을 지원하므로 다른 서비스의 리소스에 동일한 태그를 할당하여 해당 리소스의 관련 여부를 나타낼 수 있습니다. 예를 들어 EC2 인스턴스에 할당하는 것과 동일한 태그를 CloudWatch 규칙에 할당할 수 있습니다.

다음 단원에서는 CloudWatch의 태그에 관한 추가 정보를 제공합니다.

CloudWatch에서 지원되는 리소스

CloudWatch의 다음 리소스는 태그 지정을 지원합니다.

- 경보 - tag-resource AWS CLI 명령 및 TagResource API를 사용하여 경보에 태그를 지정할 수 있습니다. CloudWatch 콘솔의 경보 세부 정보 페이지를 참조하여 경보 태그를 보고 관리할 수도 있습니다.
- canary - CloudWatch 콘솔을 사용하여 canary에 태그를 지정할 수 있습니다. 자세한 내용은 canary 생성 섹션을 참조하세요.
- Contributor Insights 규칙 - Contributor Insights 규칙을 생성할 때 put-insight-rule AWS CLI 명령 및 PutInsightRule API를 사용하여 해당 규칙에 태그를 지정할 수 있습니다. tag-resource AWS CLI 명령과 TagResource API를 사용하여 기존 규칙에 태그를 추가할 수 있습니다.
- 지표 스트림 – 지표 스트림을 생성할 때 put-metric-stream AWS CLI 명령 및 PutMetricStream API를 사용하여 지표 스트림에 태그를 지정할 수 있습니다. tag-resource AWS CLI 명령 및 TagResource API를 사용하여 기존 지표 스트림에 태그를 추가할 수 있습니다.

태그 추가 및 관리에 대한 자세한 내용은 태그 관리 단원을 참조하세요.

태그 관리

태그는 리소스의 `Key` 및 `Value` 속성으로 구성됩니다. CloudWatch 콘솔, AWS CLI 또는 CloudWatch API를 사용하여 이러한 속성의 값을 추가, 편집 또는 삭제할 수 있습니다. 태그 작업에 대한 자세한 내용은 다음을 참조하세요.

- Amazon CloudWatch API 참조의 TagResource, UntagResource, ListTagsForResource
- Amazon CloudWatch CLI 참조의 tag-resource, untag-resource, list-tags-for-resource
- Resource Groups 사용 설명서의 Tag Editor 작업

태그 이름 지정 및 사용 규칙

CloudWatch 리소스에서 태그를 사용할 때 다음과 같은 기본 이름 지정 및 사용 규칙이 적용됩니다.

- 각 리소스는 최대 50개의 태그를 보유할 수 있습니다.
- 각 리소스에 대해 각 태그 키는 고유하며 하나의 값만 가질 수 있습니다.
- 태그 키의 최대 길이는 UTF-8 형식의 유니코드 문자 128자입니다.
- 태그 값의 최대 길이는 UTF-8 형식의 유니코드 문자 256자입니다.
- 허용되는 문자는 UTF-8로 표현할 수 있는 문자, 숫자, 공백 및 . : + = @ _ / -(하이픈) 문자도 있습니다.
- 태그 키와 값은 대/소문자를 구분합니다. 모범 사례는 태그를 대문자로 사용할 것을 전략으로 결정하고 모든 리소스 유형에 대해 일관되게 해당 전략을 구현하는 것입니다. 예를 들어, `Costcenter`, `costcenter` 또는 `CostCenter`를 사용할지 결정하고 모든 태그에 대해 동일한 규칙을 사용합니다. 대/소문자가 일치하지 않는 유사한 태그를 사용하지 마세요.
- `aws:` 접두사는 AWS가 사용하도록 예약되어 있으므로 태그 사용이 금지되어 있습니다. 이 접두사가 지정된 태그 키나 값은 편집하거나 삭제할 수 없습니다. 이 접두사가 지정된 태그는 리소스당 태그 수 제한에 포함되지 않습니다.

Grafana 통합

Grafana 버전 6.5.0 이상을 사용하여 컨텍스트에 따라 CloudWatch 콘솔로 진행하고 와일드카드를 사용하여 동적 지표 목록을 쿼리할 수 있습니다. 이렇게 하면 Amazon Elastic Compute Cloud 인스턴스 또는 컨테이너와 같은 AWS 리소스에 대한 지표를 모니터링할 수 있습니다. Auto Scaling 이벤트의 일부로 새 인스턴스가 생성되면 자동으로 그래프에 나타납니다. 새 인스턴스 ID를 추적할 필요가 없습니다. 미리 구축된 대시보드를 사용하면 Amazon EC2, Amazon Elastic Block Store 및 AWS Lambda 리소스 모니터링을 위한 시작 환경을 간소화할 수 있습니다.

Grafana 버전 7.0 이상을 사용하여 CloudWatch Logs의 로그 그룹에 대한 CloudWatch Logs Insights 쿼리를 수행할 수 있습니다. 막대, 선 및 누적 그래프와 테이블 형식으로 쿼리 결과를 시각화할 수 있습니다. CloudWatch Logs Insights에 대한 자세한 내용은 CloudWatch Logs Insights를 사용한 로그 데이터 분석 단원을 참조하세요.

시작하는 방법에 대한 자세한 내용은 Grafana Labs 설명서의 Grafana에서 AWS CloudWatch 사용을 참조하세요.

교차 계정 교차 리전 CloudWatch 콘솔

지표, 로그 및 추적에 대해 가장 풍부한 크로스 계정 관측성 및 검색 경험을 얻으려면 CloudWatch 크로스 계정 관측성을 사용하는 것이 좋습니다. 자세한 내용은 CloudWatch 크로스 계정 관측성 섹션을 참조하세요.

CloudWatch는 크로스 계정, 크로스 리전 CloudWatch 대시보드도 제공합니다. 이 기능은 대시보드, 경보, 지표 및 자동 대시보드에 대한 크로스 계정 가시성을 제공합니다. 로그나 추적에 대한 크로스 계정 가시성은 제공하지 않습니다.

CloudWatch 크로스 계정 관측성도 사용하는 경우 이 크로스 계정 CloudWatch 대시보드의 한 가지 사용 사례는 CloudWatch 크로스 계정 관측성 소스 계정 중 하나가 다른 소스 계정의 지표를 볼 수 있도록 하는 것입니다.

이 섹션의 나머지 부분에서는 크로스 계정, 크로스 리전 대시보드에 대해 설명합니다. 여러 AWS 계정 및 여러 AWS 리전의 CloudWatch 데이터를 단일 대시보드에 요약하는 대시보드를 생성할 수 있습니다. 또한 다른 계정에 있는 지표를 감시하는 경보를 한 계정에서 생성할 수도 있습니다.

많은 조직에서는 결제 및 보안 경계를 제공하기 위해 AWS 리소스를 여러 계정에 배포하고 있습니다. 이 경우 하나 이상의 사용자 계정을 모니터링 계정으로 지정하고 이러한 계정에 교차 계정 대시보드를 구축하는 것이 좋습니다.

교차 계정 기능이 AWS Organizations와 통합되어 교차 계정 대시보드를 효율적으로 구축할 수 있습니다.

크로스 리전 기능

이제 교차 리전 기능이 자동으로 기본 제공됩니다. 별도의 단계가 필요 없이 동일한 그래프나 동일한 대시보드에서 단일 계정으로 여러 리전의 지표를 표시할 수 있습니다. 경보에 대한 교차 리전 기능을 지원하지 않으므로 다른 리전의 지표를 감시하는 리전에서는 경보를 생성할 수 없습니다.

주제

- CloudWatch에서 교차 계정 기능 사용 설정
- (선택 사항) AWS Organizations와 통합
- CloudWatch 교차 계정 설정 문제 해결
- 교차 계정 사용 후 사용 중지 및 정리

CloudWatch에서 교차 계정 기능 사용 설정

CloudWatch 콘솔에서 교차 계정 기능을 설정하려면 CloudWatch 콘솔을 사용하여 공유 계정 및 모니터링 계정을 설정합니다.

공유 계정 설정

모니터링 계정에서 데이터를 사용할 수 있도록 각 계정에서 공유를 활성화해야 합니다.

이렇게 하면 공유받는 계정의 사용자에게 해당 권한이 있는 경우 공유받는 계정의 교차 계정 대시보드를 보는 모든 사용자에게 5단계에서 선택한 읽기 전용 권한이 부여됩니다.

계정이 CloudWatch 데이터를 다른 계정과 공유할 수 있도록 하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. Share your CloudWatch data(CloudWatch 데이터 공유)에서 Configure(구성)를 선택합니다.
4. 공유(Sharing)에서 특정 계정(Specific accounts)을 선택하고 데이터를 공유할 계정의 ID를 입력합니다.

   여기에서 지정되는 해당 계정은 공유하는 계정의 CloudWatch 데이터를 볼 수 있습니다. 알고 신뢰하는 계정의 ID만 지정합니다.
5. 권한(Permissions)에 대해 다음 옵션 중 하나를 사용하여 데이터를 공유하는 방법을 지정합니다.

   - CloudWatch 지표, 대시보드 및 경보에 대한 읽기 전용 액세스 권한 제공(Provide read-only access to your CloudWatch metrics, dashboards, and alarms). 이 옵션을 사용하면 모니터링 계정이 계정의 CloudWatch 데이터가 포함된 위젯이 있는 교차 계정 대시보드를 생성할 수 있습니다.
   - CloudWatch 자동 대시보드 포함(Include CloudWatch automatic dashboards). 이 옵션을 선택하면 모니터링 계정의 사용자가 이 계정의 자동 대시보드에서 정보를 볼 수도 있습니다. 자세한 내용은 Amazon CloudWatch 시작하기 섹션을 참조하세요.
   - X-Ray 트레이스 맵에 대한 X-Ray 읽기 전용 액세스를 포함합니다. 이 옵션을 선택하면 모니터링 계정의 사용자가 이 계정의 X-Ray 트레이스 맵 및 X-Ray 트레이스 정보도 볼 수 있습니다. 자세한 내용은 X-Ray 트레이스 맵 사용을 참조하세요.
   - 사용자 계정의 모든 항목에 대한 전체 읽기 전용 액세스 권한(Full read-only access to everything in your account). 이 옵션을 사용하면 공유에 사용되는 계정이 계정의 CloudWatch 데이터가 포함된 위젯이 있는 교차 계정 대시보드를 생성할 수 있습니다. 또한 이러한 계정에서

사용자 계정을 더 자세히 살펴보고 다른 AWS 서비스의 콘솔에서 사용자 계정의 데이터를 볼 수 있습니다.

6. CloudFormation 템플릿 실행(Launch CloudFormation template)을 선택합니다.

   확인 화면에서 **`Confirm`**을 입력하고 템플릿 실행(Launch template)을 선택합니다.

7. ...확인합니다.(I acknowledge...) 확인란을 선택하고 스택 생성(Create stack)을 선택합니다.

조직 전체와 공유

위의 절차를 완료하면 IAM 역할이 생성됩니다. 이 역할을 사용하여 계정이 하나의 계정과 데이터를 공유할 수 있습니다. 조직의 모든 계정과 데이터를 공유하는 IAM 역할을 생성하거나 편집할 수 있습니다. 조직의 모든 계정을 알고 신뢰하는 경우에만 이 작업을 수행하십시오.

이렇게 하면 공유받는 계정의 사용자에게 해당 권한이 있는 경우 공유받는 계정의 교차 계정 대시보드를 보는 모든 사용자에게 위 절차의 5단계에 나와 있는 정책에 나열된 읽기 전용 권한이 부여됩니다.

조직의 모든 계정과 CloudWatch 계정 데이터를 공유하려면

1. 아직 수행하지 않은 경우 이전 절차를 완료하여 한 AWS 계정과 데이터를 공유하십시오.
2. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
3. 탐색 창에서 역할을 선택합니다.
4. 역할 목록에서 CloudWatch-CrossAccountSharingRole을 선택합니다.
5. 신뢰 관계(Trust relationships), 신뢰 관계 편집(Edit trust relationship)을 차례대로 선택합니다.

   다음과 같은 정책이 표시됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. 정책을 다음과 같이 변경하여 *org-id* 를 조직의 ID로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "org-id"
        }
      }

    }
  ]
}
```

7. 신뢰 정책 업데이트를 선택합니다.

모니터링 계정 설정

교차 계정 CloudWatch 데이터를 보려는 경우 각 모니터링 계정을 사용 설정합니다.

다음 절차를 완료하면 CloudWatch는 다른 계정에서 공유한 데이터에 액세스하는 데 사용하는 모니터링 계정의 서비스 연결 역할을 생성합니다. 이 서비스 연결 역할을 AWSServiceRoleForCloudWatchCrossAccount라고 합니다. 자세한 내용은 CloudWatch에 서비스 연결 역할 사용 섹션을 참조하십시오.

계정이 교차 계정 CloudWatch 데이터를 볼 수 있도록 사용 설정하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 설정(Settings)을 선택한 다음 교차 계정 교차 리전(Cross-account cross-region) 섹션에서 구성(Configure)을 선택합니다.
3. 교차 계정 교차 리전 보기(View cross-account cross-region) 섹션에서 사용 설정(Enable)을 선택한 다음 콘솔에 선택기 표시(Show selector in the console) 확인란을 선택하여 지표를 그래프로 표시하거나 경보를 생성할 때 CloudWatch 콘솔에 계정 선택기가 표시되도록 합니다.

4. 교차 계정 교차 리전 보기(View cross-account cross-region)에서 다음 옵션 중 하나를 선택합니다.

   - 계정 ID 입력(Account Id Input). 이 옵션은 교차 계정 데이터를 볼 때 계정을 전환할 때마다 계정 ID를 수동으로 입력하라는 메시지를 표시합니다.
   - AWS Organization account selector. 이 옵션을 사용하면 Organizations와의 교차 계정 통합을 완료할 때 지정한 계정이 표시됩니다. 다음에 콘솔을 사용하면 CloudWatch는 교차 계정 데이터를 볼 때 선택할 수 있도록 이러한 계정의 드롭다운 목록을 표시합니다.

     이렇게 하려면 먼저, 조직 관리 계정을 사용하여 CloudWatch가 조직의 계정 목록을 확인할 수 있도록 해야 합니다. 자세한 내용은 (선택 사항) AWS Organizations와 통합 섹션을 참조하세요.
   - 사용자 지정 계정 선택기(Custom account selector). 이 옵션은 계정 ID 목록을 입력하라는 메시지를 표시합니다. 다음에 콘솔을 사용하면 CloudWatch는 교차 계정 데이터를 볼 때 선택할 수 있도록 이러한 계정의 드롭다운 목록을 표시합니다.

     또한 보려는 계정을 선택할 때 계정을 식별하는 데 도움이 되도록 각 계정에 대한 레이블을 입력할 수도 있습니다.

     여기서 사용자가 지정하는 계정 선택기 설정은 모니터링 계정의 다른 모든 사용자가 아니라 해당 사용자에 대해서만 유지됩니다.
5. 사용(Enable)을 선택합니다.

이 설정을 완료한 후 교차 계정 대시보드를 생성할 수 있습니다. 자세한 내용은 교차 계정 교차 리전 대시보드 섹션을 참조하세요.

(선택 사항) AWS Organizations와 통합

교차 계정 기능을 AWS Organizations와 통합하려면 조직의 모든 계정 목록을 모니터링 계정에서 사용할 수 있도록 만들어야 합니다.

교차 계정 CloudWatch 기능을 사용 설정하여 조직의 모든 계정 목록에 액세스하려면

1. 조직의 관리 계정에 로그인합니다.
2. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
3. 탐색 창에서 설정(Settings)을 선택한 다음 구성(Configure)을 선택합니다.
4. 조직의 계정 목록을 볼 수 있는 권한 부여(Grant permission to view the list of accounts in the organization)에 대해 특정 계정(Specific accounts)을 선택하여 계정 ID 목록을 입력하라는 메시지를 표시합니다. 조직의 계정 목록은 여기에서 지정한 계정과만 공유됩니다.

5. 조직 계정 목록 공유(Share organization account list)를 선택합니다.
6. CloudFormation 템플릿 실행(Launch CloudFormation template)을 선택합니다.

   확인 화면에서 **`Confirm`**을 입력하고 템플릿 실행(Launch template)을 선택합니다.

CloudWatch 교차 계정 설정 문제 해결

이 단원에는 CloudWatch의 교차 계정 콘솔 배포에 대한 문제 해결 팁이 포함되어 있습니다.

교차 계정 데이터를 표시하는 데 액세스 거부 오류가 발생했습니다.

다음을 확인하세요.

- 모니터링 계정에는 AWSServiceRoleForCloudWatchCrossAccount라는 역할이 있어야 합니다. 그렇지 않은 경우 이 역할을 생성해야 합니다. 자세한 내용은 Set Up a Monitoring Account 섹션을 참조하세요.
- 각 공유 계정에는 CloudWatch-CrossAccountSharingRole이라는 역할이 있어야 합니다. 그렇지 않은 경우 이 역할을 생성해야 합니다. 자세한 내용은 Set Up A Sharing Account 섹션을 참조하세요.
- 공유 역할은 모니터링 계정을 신뢰해야 합니다.

CloudWatch 교차 계정 콘솔에 대해 역할이 올바르게 설정되었는지 확인하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 목록에서 필요한 역할이 있는지 확인합니다. 공유 계정에서 CloudWatch-CrossAccountSharingRole을 찾습니다. 모니터링 계정에서 AWSServiceRoleForCloudWatchCrossAccount를 찾습니다.
4. 공유 계정에 CloudWatch-CrossAccountSharingRole이 이미 있는 경우 CloudWatch-CrossAccountSharingRole을 선택합니다.
5. 신뢰 관계(Trust relationships), 신뢰 관계 편집(Edit trust relationship)을 차례대로 선택합니다.
6. 정책에 모니터링 계정의 계정 ID 또는 모니터링 계정이 포함된 조직의 조직 ID가 나열되어 있는지 확인합니다.

콘솔에 계정 드롭다운이 표시되지 않음

먼저, 앞의 문제 해결 단원에서 설명한 대로 올바른 IAM 역할을 생성했는지 확인합니다. 이러한 항목이 올바르게 설정된 경우 Enable Your Account to View Cross-Account Data에 설명된 대로 이 계정에서 교차 계정 데이터를 볼 수 있도록 활성화해야 합니다.

교차 계정 사용 후 사용 중지 및 정리

CloudWatch에 대한 교차 계정 기능을 사용 중지하려면 다음 단계를 따릅니다.

1단계: 크로스 계정 스택 또는 역할 제거

가장 좋은 방법은 교차 계정 기능을 사용 설정하는 데 사용된 AWS CloudFormation 스택을 제거하는 것입니다.

- 각 공유 계정에서 [CloudWatch-CrossAccountSharingRole] 스택을 제거합니다.
- AWS Organizations를 사용하여 조직의 모든 계정에서 교차 계정 기능을 사용 설정한 경우 조직의 관리 계정에서 [CloudWatch-CrossAccountListAccountsRole] 스택을 제거합니다.

교차 계정 기능을 사용 설정하는 데 AWS CloudFormation 스택을 사용하지 않은 경우 다음을 수행합니다.

- 각 공유 계정에서 [CloudWatch-CrossAccountSharingRole] IAM 역할을 삭제합니다.
- AWS Organizations를 사용하여 조직의 모든 계정에서 교차 계정 기능을 사용 설정한 경우 조직의 관리 계정에서 [CloudWatch-CrossAccountSharing-ListAccountsRole] IAM 역할을 삭제합니다.

2단계: 서비스 연결 역할 제거

모니터링 계정에서 [AWSServiceRoleForCloudWatchCrossAccount] 서비스 연결 IAM 역할을 삭제합니다.

CloudWatch 서비스 할당량

CloudWatch에는 지표, 경보, API 요청, 경보 이메일 알림에 대해 다음과 같은 할당량이 있습니다.

> **Note**
>
> CloudWatch를 포함한 일부 AWS 서비스의 경우 CloudWatch 사용량 지표를 사용하여 현재 서비스 사용량을 CloudWatch 그래프와 대시보드에서 시각화할 수 있습니다. CloudWatch 지표 수학 함수를 사용하여 해당 리소스에 대한 서비스 할당량을 그래프에 표시할 수 있습니다. 사용량이 서비스 할당량에 가까워지면 경고하는 경보를 구성할 수도 있습니다. 자세한 내용은 서비스 할당량 시각화 및 경보 설정 단원을 참조하십시오.

Resource	기본 할당량
경보 작업	5/경보. 이 할당량은 변경할 수 없습니다.
경보 평가 기간	경보 기간에 사용된 평가 기간 수를 곱하여 계산한 최댓값은 1일(86,400초)입니다. 이 할당량은 변경할 수 없습니다.
개 경보	고객당 월 10회 무료. 추가 경보에는 요금이 부과됩니다. 계정당 총 경보 수에는 제한이 없습니다. 지표 수학 표현식 기반 경보에는 지표가 최대 10개까지 허용됩니다. 리전당 200개 Metrics Insights 경보. 할당량 증가를 요청할 수 있습니다.
이상 탐지 모델	계정마다 리전당 500개
API 요청	고객당 월 1,000,000회 무료.
카나리아	계정당 리전별 200개. 할당량 증가를 요청할 수 있습니다.

Resource	기본 할당량
Contributor Insights API 요청	다음 API의 할당량은 리전당 20TPS(초당 트랜잭션)입니다. • DescribeInsightRules 할당량은 변경할 수 없습니다. • GetInsightRuleReport 할당량 증가를 요청할 수 있습니다. 다음 API의 할당량은 리전당 5TPS입니다. 이 할당량은 변경할 수 없습니다. • DeleteInsightRules • PutInsightRule 다음 API의 할당량은 리전당 1TPS입니다. 이 할당량은 변경할 수 없습니다. • DisableInsightRules • EnableInsightRules
Contributor Insights 규칙	계정별 리전당 규칙 100개입니다. 할당량 증가를 요청할 수 있습니다.
사용자 지정 지표	할당량이 없습니다.

Resource	기본 할당량
대시보드	대시보드 당 최대 500개 위젯 대시보드 위젯당 최대 500개 지표 전체 위젯에서 대시보드당 최대 2500개 지표 이 할당량에는 지표 수학 함수에서 사용하기 위해 검색한 모든 지표가 포함됩니다. 이러한 지표가 그래프에 표시되지 않는 경우에도 그렇습니다. 이러한 할당량은 변경할 수 없습니다.
DescribeAlarms	리전당 9TPS(초당 트랜잭션)입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 할당량 증가를 요청할 수 있습니다.
DeleteAlarms 요청 DescribeAlarmHistory 요청 DisableAlarmActions 요청 EnableAlarmActions 요청 SetAlarmState 요청	이러한 각 작업에 대해 리전당 3TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 이러한 할당량은 변경할 수 없습니다.
DescribeAlarmsForMetric 요청	리전당 9TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 이 할당량은 변경할 수 없습니다.
DeleteDashboards 요청 GetDashboard 요청 ListDashboards 요청 PutDashboard 요청	이러한 각 작업에 대해 리전당 10TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 이러한 할당량은 변경할 수 없습니다.
PutAnomalyDetector DescribeAnomalyDetectors	리전당 10TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다.

Resource	기본 할당량
DeleteAnomalyDetector	리전당 5TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다.
차원	30/미터법. 이 할당량은 변경할 수 없습니다.
GetMetricData	Metrics Insights 쿼리가 포함된 작업의 경우 리전당 10TPS. Metrics Insights 쿼리가 포함되지 않은 작업의 경우 할당량은 리전당 50TPS입니다. 이는 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 할당량 증가를 요청할 수 있습니다. Metrics Insights 쿼리를 포함하는 `GetMetricData` 작업의 할당량은 가장 최근 3시간 동안 4,300,000초당 데이터 포인트(DPS)입니다. 이는 지표를 10,000개 이하로 포함할 수 있는 쿼리에서 스캔한 총 데이터 포인트 수에 대해 계산됩니다. API 요청에서 사용되는 `StartTime` 이 현재 시간과 같거나 3시간 적을 경우 초당 데이터포인트(DPS)가 180,000입니다. `StartTime` 이 현재 시간보다 3시간 많을 경우 DPS는 396,000입니다. 이것은 조절 없이 하나 이상의 API 호출을 사용하여 1초마다 요청할 수 있는 데이터포인트의 최대 수입니다. 이 할당량은 변경할 수 없습니다. DPS는 실제 데이터 포인트가 아닌 예상 데이터 포인트를 기반으로 계산됩니다. 데이터 포인트 추정치는 요청된 시간 범위, 기간 및 보존 기간을 사용하여 계산됩니다. 즉, 요청된 지표의 실제 데이터 포인트가 희박하거나 비어 있는 경우에도 예상 데이터 포인트가 할당량을 초과하면 조절이 이루어집니다. DPS 할당량은 리전당입니다.

Resource	기본 할당량
GetMetricData	단일 GetMetricData 호출에는 다음이 포함될 수 있습니다. • 최대 500개의 MetricDataQuery 구조. • 최대 100개의 SERVICE_QUOTA() 함수. • 최대 100개의 SEARCH()함수. • 최대 5개의 LAMBDA()함수. 이러한 할당량은 변경할 수 없습니다.
GetMetricStatistics	리전당 400TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 할당량 증가를 요청할 수 있습니다.
GetMetricWidgetImage	이미지당 최대 500개의 지표. 이 할당량은 변경할 수 없습니다. 리전당 20TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 할당량 증가를 요청할 수 있습니다.
ListMetrics	리전당 25TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 할당량 증가를 요청할 수 있습니다.
지표 데이터 값	지표 데이터 포인트의 값은 -2^360에서 2^360 사이의 범위에 있어야 합니다. 특수 값(예: NaN, +무한대, -무한대)은 지원되지 않습니다. 이 할당량은 변경할 수 없습니다.
MetricDatum 항목	1000/PutMetricData 요청. MetricDatum 객체는 단일 값을 포함하거나 여러 값을 나타내는 StatisticSet 개체를 포함할 수 있습니다. 이 할당량은 변경할 수 없습니다.

Resource	기본 할당량
지표	고객당 월 10회 무료.
Metrics Insights 쿼리	단일 쿼리는 10,000개 이하의 지표를 처리할 수 있습니다. 즉, SELECT, FROM, WHERE 절이 10,000개 이상의 지표와 일치하면 발견된 해당 지표 중 처음 10,000개만 쿼리에 의해 처리됩니다. 단일 쿼리는 500개 이하의 시계열을 반환할 수 있습니다. 가장 최근 3시간의 데이터만 쿼리할 수 있습니다.
Observability Access Manager(OAM) API 요청 속도입니다.	PutSinkPolicy의 경우 지역당 1TPS입니다. CloudWatch OAM API 각각에 대해 리전당 10TPS입니다. 이러한 할당량은 제한 없이 초당 수행할 수 있는 최대 작업 요청 수를 반영합니다. 이러한 할당량은 변경할 수 없습니다.
OAM 소스 계정 링크	각 소스 계정은 최대 5개의 모니터링 계정에 연결할 수 있습니다. 이 할당량은 변경할 수 없습니다.
OAM 싱크	계정별 리전당 싱크 1개 이 할당량은 변경할 수 없습니다.
복합 알람 설정 요청	리전당 3TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 할당량 증가를 요청할 수 있습니다.
PutMetricAlarm 요청	리전당 3TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 할당량 증가를 요청할 수 있습니다.

Resource	기본 할당량
PutMetricData 요청	HTTP POST 요청에 대하여 1MB. PutMetricData는 500건의 초당 트랜잭션(TPS)을 처리할 수 있으며, 이는 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. `PutMetricData` 은(는) 요청당 1,000개 지표를 처리할 수 있습니다. 할당량 증가를 요청할 수 있습니다.
Amazon SNS 이메일 알림	고객당 월 1,000회 무료.
Synthetics Groups	계정당 20개 이 할당량은 변경할 수 없습니다.
TagResource	리전당 20TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 이 할당량은 변경할 수 없습니다.
UntagResource	리전당 20TPS입니다. 조절 없이 초당 수행할 수 있는 최대 작업 요청 수입니다. 이 할당량은 변경할 수 없습니다.

문서 기록

다음 표에서는 2018년 6월부터 적용되는 Amazon CloudWatch '사용 설명서'의 각 릴리스에서 변경된 중요 사항을 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
CloudWatch Application Signals 서비스 맵은 canary, RUM 클라이언트 및 AWS 서비스 종속성 그룹화를 지원합니다.	Application Signals 미리 보기 릴리스에서는 동일한 유형의 canary, RUM 클라이언트 및 AWS 서비스 종속성에 대한 기본 그룹화가 서비스 맵에 추가되었습니다. 이번 변경 내용으로 서비스 맵 기본 보기의 아이콘 수가 줄어들어 보기 및 탐색이 더 쉬워졌습니다.	2024년 5월 21일
CloudWatchReadOnlyAccess IAM 정책이 업데이트됨	CloudWatch에서 CloudWatchReadOnlyAccess의 권한 범위가 변경되었습니다. 사용자가 CloudWatch Application Signals를 사용하여 서비스 상태와 관련된 문제를 보고, 조사하며, 진단할 수 있도록 정책 범위에서 `application-signals:BatchGet*` , `application-signals:Get*` , `application-signals:List*` 작업이 추가되었습니다. 또한 CloudWatch에서는 사용자가 Application Signals가 설정되었는지 확인할 수 있도록 `iam:GetRole` 작업을 추가했습니다.	2024년 5월 17일
CloudWatchFullAccessV2 IAM 정책 업데이트됨	CloudWatch에서 CloudWatchFullAccessV2의 권한 범위	2024년 5월 17일

	가 변경되었습니다. 사용자가 CloudWatch Application Signals를 사용하여 서비스 상태와 관련된 문제를 보고, 조사하며, 진단할 수 있도록 정책 범위에서 `application-signals:*` 가 추가되었습니다.	
Lambda Insights는 AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부) 지원	CloudWatch Lambda Insights에서 AWS GovCloud(미국 동부) 및 AWS GovCloud(미국 서부) 리전에 대한 지원이 추가되었습니다.	2024년 4월 29일
CloudWatch 교차 계정 관찰 기능의 리소스 필터 지원	이제 계정 간에 링크를 생성할 때 소스 계정에서 모니터링 계정으로 공유할 지표 네임스페이스와 로그 그룹을 지정하는 필터를 생성할 수 있습니다.	2024년 4월 26일
CloudWatch Application Signals 업데이트	Application Signals 미리 보기 릴리스에 세 가지 기능이 추가되었습니다. 이제 Application Signals는 Python 애플리케이션을 지원합니다. Amazon EKS 아키텍처 기반 애플리케이션에 대해 더욱 간단한 활성화 프로세스를 제공합니다. 또한 수집된 지표의 카디널리티를 관리하는 데 사용할 수 있는 새로운 구성도 포함되어 있습니다.	2024년 4월 26일

Amazon EKS의 향상된 관찰 기능을 갖춘 CloudWatch Container Insights는 AWS Elastic Fabric Adapter(EFA) 지표 수집 가능	이제 Amazon EKS의 향상된 관찰 기능을 갖춘 CloudWatch Container Insights를 사용하여 Amazon EKS 클러스터에서 AWS Elastic Fabric Adapter(EFA) 지표를 수집할 수 있습니다.	2024년 4월 23일
업데이트된 IAM 정책	CloudWatch에서 CloudWatchApplicationSignalsServiceRolePolicy 정책이 업데이트되었습니다. 더 많은 아키텍처에서 Application Signals를 활성화하도록 이 정책의 `logs:StartQuery` 및 `logs:GetQueryResults` 권한의 범위가 변경되어 `arn:aws:logs:*:*:log-group:/aws/appsignals/*:*` 및 `"arn:aws:logs:*:*:log-group:/aws/application-signals/data:*"` 가 추가되었습니다. 이 정책은 AWSServiceRoleForCloudWatchApplicationSignals 서비스 연결 역할에 연결됩니다.	2024년 4월 18일
Internet Monitor에서 인증된 AWS 고객에게 글로벌 인터넷 웨더 맵 제공	Amazon CloudWatch Internet Monitor는 인증된 모든 AWS 고객에게 콘솔에서 제공되는 글로벌 인터넷 웨더 맵을 표시합니다. 맵을 보려면 Amazon CloudWatch 콘솔에서 Internet Monitor로 이동합니다.	2024년 4월 16일

Amazon EKS의 향상된 관찰 기능을 갖춘 CloudWatch Container Insights는 AWS Neuron 지표 수집 가능	이제 Amazon EKS의 향상된 관찰 기능을 갖춘 CloudWatch Container Insights를 사용하여 Amazon EKS 클러스터에서 AWS Neuron 지표를 수집할 수 있습니다.	2024년 4월 16일
CloudWatch Application Signals에 진단에 도움이 되는 서비스 개요 탭 등 추가	새로운 서비스 개요 탭에는 작업 수, 종속성, Synthetics 및 클라이언트 페이지를 비롯한 서비스 개요가 표시됩니다. 탭에는 전체 서비스, 상위 작업 및 종속성에 대한 주요 지표가 표시됩니다. 또한 이제 결함, 오류 및 대기 시간 문제를 비롯한 문제와 상관관계가 있는 X-Ray 트레이스를 볼 수 있습니다.	2024년 4월 16일
CloudWatch의 Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights에 Windows 지원 추가	이제 Amazon EKS의 향상된 관찰 기능을 갖춘 CloudWatch Container Insights를 사용하여 Amazon EKS 클러스터의 Windows 워커 노드에서 지표를 수집할 수 있습니다.	2024년 4월 10일
CloudWatchApplicationSignalsServiceRolePolicy IAM 정책 업데이트	CloudWatch에서 CloudWatchApplicationSignalsServiceRolePolicy의 권한 범위가 변경되었습니다. Application Signals가 연결된 계정의 소스에서 지표를 검색할 수 있도록 `cloudwatch:GetMetricData` 권한 범위가 *로 변경되었습니다.	2024년 4월 8일

Amazon CloudWatch Internet Monitor는 이제 계정 간 관찰성을 지원합니다.	이제 Internet Monitor의 교차 계정 관찰성을 사용하여 단일 AWS 리전 내의 여러 AWS 계정에 걸쳐 있는 애플리케이션을 모니터링할 수 있습니다.	2024년 3월 29일
CloudWatchAgentServerPolicy 및 CloudWatchAgentAdminPolicy 정책 업데이트됨	CloudWatch가 `CloudWatchAgentServerPolicy` 및 `CloudWatchAgentAdminPolicy` 정책에 대한 권한을 추가하여 CloudWatch 에이전트가 X-Ray 트레이스를 게시하고 로그 그룹 보존 기간을 수정할 수 있도록 허용했습니다. 두 정책 모두에 `xray:PutTraceSegments` , `xray:PutTelemetryRecords` , `xray:GetSamplingRules` , `xray:GetSamplingTargets` , `xray:GetSamplingStatisticSummaries` , `logs:PutRetentionPolicy` 권한 추가됨	2024년 2월 12일

CloudWatch Network Monitor에 대한 새 서비스 연결 역할 및 IAM 정책	CloudWatch에 AWSServiceRoleForNetworkMonitor라는 새로운 서비스 연결 역할이 추가되었습니다. CloudWatch는 소스 서브넷과 대상 IP 주소 간의 네트워크 지표를 가져오는 모니터를 생성할 수 있도록 이 새로운 서비스 연결 역할을 추가했습니다. 새 CloudWatchNetworkMonitorServiceRolePolicy IAM 정책은 이 역할에 연결되며, 정책은 CloudWatch에 사용자를 대신하여 네트워크 지표를 가져올 수 있는 권한을 부여합니다.	2023년 12월 22일
CloudWatch에서 Amazon CloudWatch Network Monitor 출시	CloudWatch는 Amazon CloudWatch 네트워크 모니터라는 새로운 기능을 출시했습니다. Amazon CloudWatch 네트워크 모니터는 AWS 네트워크 또는 회사 네트워크 내에 네트워크 문제가 있는지 확인하는 새로운 능동형 네트워크 모니터링 서비스입니다.	2023년 12월 22일

CloudWatchReadOnlyAccess 정책이 업데이트됨	CloudWatch는 이 정책이 적용되는 사용자가 CloudWatch Application Signals에서 보고하는 서비스 상태 문제를 분류하고 진단할 수 있도록 CloudWatch Synthetics, X-Ray, CloudWatch RUM에 대한 기존 읽기 전용 권한과 CloudWatch Application Signals에 대한 새로운 읽기 전용 권한을 CloudWatchReadOnlyAccess에 추가했습니다. 이 정책이 적용되는 사용자가 자연어 프롬프트에서 CloudWatch Metrics Insights 쿼리 문자열을 생성할 수 있도록 `cloudwatch:GenerateQuery` 권한이 추가되었습니다.	2023년 12월 5일
CloudWatchFullAccessV2 정책 업데이트됨	CloudWatch는 이 정책이 적용되는 사용자가 Application Signals를 완벽하게 관리하여 서비스 상태의 문제를 파악하고 진단할 수 있도록 CloudWatchFullAccessV2 for CloudWatch Synthetics, X-Ray 및 CloudWatch RUM에 기존 권한을 추가하고 CloudWatch Application Signals에 대한 새 권한을 추가했습니다.	2023년 12월 5일

새 서비스 연결 역할 및 새 IAM 정책	CloudWatch에 AWSServiceRoleForCloudWatchApplicationSignals라는 새로운 서비스 연결 역할이 추가되었습니다. CloudWatch는 CloudWatch Application Signals가 CloudWatch Application Signals에 대해 활성화한 애플리케이션에서 CloudWatch Logs 데이터, X-Ray 트레이스 데이터, CloudWatch 지표 데이터 및 태그 지정 데이터를 수집할 수 있도록 이 새로운 서비스 연결 역할을 추가했습니다. 새로운 CloudWatchApplicationSignalsServiceRolePolicy IAM 정책이 이 역할에 연결됩니다. 이 정책은 CloudWatch Application Signals에 권한을 부여하여 다른 관련 AWS 서비스에서 모니터링 및 태그 지정 데이터를 수집합니다.	2023년 11월 30일
CloudWatch에서 Application Signals의 평가판 릴리스 출시	CloudWatch Application Signals는 평가판입니다. Application Signals를 사용하면 현재 애플리케이션 상태를 모니터링하고, 서비스 수준 목표(SLO)를 생성하고, 비즈니스 목표에 대한 장기 애플리케이션 성능을 추적할 수 있도록 AWS에서 애플리케이션을 계측할 수 있습니다. 자세한 내용은 Application Signals를 참조하세요.	2023년 11월 30일

CloudWatch에서 다른 데이터 소스 쿼리를 위한 지원 추가	CloudWatch를 사용하여 다른 데이터 소스의 지표를 쿼리하고 시각화하고 그에 대한 경보를 생성할 수 있습니다. 자세한 내용은 다른 데이터 소스의 지표 쿼리를 참조하세요.	2023년 11월 26일
CloudWatch Metrics Insights에서 자연어 쿼리 생성 지원	CloudWatch Metrics Insights는 쿼리를 생성하고 업데이트하기 위한 자연어 쿼리를 지원합니다. 자세한 내용은 Use natural language to generate and update CloudWatch Metric Insights queries를 참조하세요.	2023년 11월 26일
CloudWatch, Amazon EKS의 향상된 관찰 기능을 갖춘 Container Insights 출시	CloudWatch에서 Container Insights의 새 버전을 출시했습니다. 이 버전은 Amazon EKS 클러스터의 향상된 관찰 기능을 지원하며 Amazon EKS에서 실행되는 클러스터에서 더 자세한 지표를 수집할 수 있습니다. 설치 후에는 Amazon EKS 클러스터에 대한 상세한 인프라 텔레메트리 및 컨테이너 로그를 자동으로 수집합니다. 그런 다음 즉시 사용할 수 있는 엄선된 대시보드를 사용하여 애플리케이션 및 인프라 텔레메트리를 자세히 살펴볼 수 있습니다.	2023년 11월 6일

CloudWatch 지표 스트림에 빠른 파트너 설정 추가	CloudWatch 지표 스트림은 이제 빠른 파트너 설정 옵션을 제공합니다. 이 옵션을 사용하여 일부 타사 공급자에 대한 지표 스트림을 빠르게 설정할 수 있습니다.	2023년 10월 17일
CloudWatch, 경보 권장 사항 발표	CloudWatch Synthetics는 이제 다른 AWS 서비스의 지표에 대한 경보 권장 사항을 제공합니다. 이러한 권장 사항은 관련 서비스 모니터링의 모범 사례를 따르기 위해 경보를 설정해야 하는 지표를 식별하는 데 도움이 됩니다.	2023년 10월 16일
CloudWatch Synthetics, 런타임 syn-nodejs-puppeteer-6.0 출시	CloudWatch Synthetics가 런타임 `syn-nodejs-puppeteer-6.0` 을 출시했습니다.	2023년 9월 26일
교차 계정 애플리케이션에 대한 Amazon CloudWatch Application Insights 지원 추가	이제 계정 경계를 넘어 CloudWatch Application Insights 애플리케이션을 공유할 수 있습니다.	2023년 9월 26일

새 서비스 연결 역할 및 새 IAM 정책	CloudWatch에 AWSServiceRoleForCloudWatchMetrics_DbPerfInsights라는 새로운 서비스 연결 역할이 추가되었습니다. 이 새 서비스 연결 역할을 추가하여 CloudWatch가 경보, 이상 탐지, 스냅샷을 위한 성능 개선 도우미 지표를 가져올 수 있도록 했습니다. 이 역할에는 새로운 AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy IAM 정책이 첨부되며, 이 정책은 사용자를 대신하여 성능 개선 도우미 지표를 가져올 수 있는 권한을 CloudWatch에 부여합니다.	2023년 9월 20일
새 지표 수학 함수 추가	CloudWatch에는 경보, 이상 탐색, 스냅샷을 위해 AWS 데이터베이스 서비스에서 성능 개선 도우미 지표를 가져오는 데 사용할 수 있는 새로운 지표 수학 함수 `DB_PERF_INSIGHTS` 가 추가되었습니다.	2023년 9월 20일
CloudWatchReadOnlyAccess 정책이 업데이트됨	이 정책이 적용되는 사용자가 애플리케이션 Auto Scaling 정책에 대한 정보에 액세스할 수 있도록 CloudWatchReadOnlyAccess에 `application-autoscaling:DescribeScalingPolicies` 권한을 추가했습니다.	2023년 9월 14일
CloudWatch 에이전트, AL2023에 대한 지원이 추가됨	CloudWatch 에이전트는 AL2023을 지원합니다.	2023년 8월 8일

새 관리형 IAM 정책, CloudWatchFullAccessV2	CloudWatch에 새 정책 CloudWatchFullAccessV2가 추가되었습니다. 이 정책은 CloudWatch 작업 및 리소스에 대한 전체 액세스 권한을 부여하는 동시에 Amazon SNS 및 Amazon EC2 Auto Scaling과 같은 다른 서비스에 부여된 권한의 범위를 더 잘 지정합니다.	2023년 8월 1일
Amazon CloudWatch Internet Monitor에 대한 서비스 연결 역할 업데이트 - 기존 정책에 대한 업데이트	특정 Network Load Balancer 리소스에 대한 트래픽 모니터링을 지원하기 위해 Internet Monitor의 서비스 연결 역할에 새 권한 `elasticloadbalancing:DescribeLoadBalancers` 및 `ec2:DescribeNetworkInterfaces` 를 추가합니다.	2023년 7월 25일
Amazon CloudWatch Internet Monitor에서 Network Load Balancer 리소스에 대한 지원이 추가됨	특정 Network Load Balancer 리소스를 사용하여 Internet Monitor에서 모니터를 생성할 수 있는 지원이 추가되어 애플리케이션에 대한 보다 세분화된 수준의 관찰성을 제공합니다.	2023년 7월 25일

대시보드 변수 특성	CloudWatch는 대시보드 내에서 하나의 입력 필드를 설정하는 방법에 따라 다양한 콘텐츠를 빠르게 표시하는 유연한 대시보드를 생성하는 데 사용할 수 있는 대시보드 변수를 출시했습니다. 예를 들어 서로 다른 Lambda 함수 또는 Amazon EC2 인스턴스 ID 사이를 빠르게 전환하는 대시보드 또는 서로 다른 AWS 리전으로 전환하는 대시보드를 만들 수 있습니다. 자세한 내용은 Create flexible dashboards with dashboard variables를 참조하세요.	2023년 6월 28일
이제 Internet Monitor에서 상태 이벤트의 임곗값 사용자 지정 지원	Internet Monitor에 전역 성능 점수 또는 가용성 점수가 상태 이벤트를 트리거하는 경우의 임곗값을 사용자 지정하는 기능이 추가되었습니다. 자세한 내용은 Tracking real-time performance and availability in Amazon CloudWatch Internet Monitor를 참조하세요.	2023년 6월 26일
Internet Monitor, 이제 모든 상업 리전 지원	Internet Monitor에 7개의 새로운 AWS 리전가 추가되었으며 이제 모든 상업 리전을 지원합니다.	2023년 6월 19일

새 Lambda 인사이트 확장 버전	CloudWatch는 x86-64 플랫폼과 ARM64 플랫폼 모두에 대해 1.0.229.0 버전의 Lambda Insights 확장 기능을 추가했습니다. 자세한 내용은 Available versions of the Lambda Insights extension을 참조하세요.	2023년 6월 12일
CloudWatchReadOnlyAccess 정책이 업데이트됨	CloudWatch에서는 CloudWatchReadOnlyAccess에 권한을 추가했습니다. 이 정책이 있는 사용자가 콘솔을 사용하여 CloudWatch Logs Live Tail 세션을 시작하고 중지할 수 있도록 `logs:StartLiveTail` 및 `logs:StopLiveTail` 권한이 추가되었습니다. 자세한 내용은 Live Tail을 사용하여 거의 실시간으로 로그 보기를 참조하세요.	2023년 6월 6일
CloudWatch RUM에 사용자 지정 지표에 대한 지원 추가	CloudWatch RUM 앱 모니터를 사용하여 사용자 지정 지표를 생성하고 이를 CloudWatch 및 CloudWatch Evidently로 전송할 수 있습니다. 이 기능에는 AmazonCloudWatchRUMServiceRolePolicy 관리형 IAM 정책에 대한 업데이트가 포함되어 있습니다. 이 정책에서 조건 키가 변경되어 CloudWatch RUM이 사용자 지정 지표 네임스페이스로 사용자 지정 지표를 보낼 수 있습니다.	2023년 2월 9일

CloudWatch에 대한 신규 및 업데이트된 관리형 정책	CloudWatch 크로스 계정 관측성을 지원하기 위해 `CloudWatchFullAccess` 및 `CloudWatchReadOnlyAccess` 정책이 업데이트되었으며, `CloudWatchCrossAccountSharingConfiguration`, `OAMFullAccess` 및 `OAMReadOnlyAccess` 등과 같은 새로운 관리형 정책이 추가되었습니다. 자세한 내용은 AWS 관리형 정책에 대한 CloudWatch 업데이트를 참조하세요.	2023년 2월 7일
CloudWatch Application Insights 서비스 연결 역할 정책 업데이트 — 기존 정책에 대한 업데이트.	CloudWatch Application Insights에서 기존 AWS 서비스 연결 역할 정책을 업데이트했습니다.	2022년 12월 19일
Container Insights 콘솔의 컨테이너화된 애플리케이션 및 마이크로서비스에 대한 Amazon CloudWatch Application Insights 지원.	Container Insights 대시보드에 Amazon ECS 및 Amazon EKS에서 감지된 CloudWatch Application Insights 문제를 표시할 수 있습니다.	2021년 11월 17일
SAP HANA 데이터베이스에 대한 Amazon CloudWatch Application Insights 모니터링.	Application Insights를 사용하여 SAP HANA 데이터베이스를 모니터링할 수 있습니다.	2021년 11월 15일
계정의 모든 리소스 모니터링을 위한 Amazon CloudWatch Application Insights 지원.	계정의 모든 리소스를 온보딩하고 모니터링할 수 있습니다.	2021년 9월 15일

Amazon FSx에 대한 Amazon CloudWatch Application Insights 지원.	Amazon FSx에서 검색된 지표를 모니터링할 수 있습니다.	2021년 8월 31일
SDK 지표가 더 이상 지원되지 않음	CloudWatch SDK 지표가 더 이상 지원되지 않습니다.	2021년 8월 25일
컨테이너 모니터링 설정을 위한 Amazon CloudWatch Application Insights 지원.	Amazon CloudWatch Application Insights와 함께 모범 사례를 사용하여 컨테이너를 모니터링할 수 있습니다.	2021년 5월 18일
지표 스트림 정식 출시	지표 스트림을 사용하여 원하는 대상에 CloudWatch 지표를 지속적으로 스트리밍할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 지표 스트림 단원을 참조하세요.	2021년 3월 31일
Amazon RDS 및 Amazon EC2의 Oracle 데이터베이스에 대한 Amazon CloudWatch Application Insights 모니터링.	Amazon CloudWatch Application Insights를 사용하여 Oracle에서 검색된 지표 및 로그를 모니터링할 수 있습니다.	2021년 1월 16일
Lambda Insights 정식 출시	CloudWatch Lambda Insights는 AWS Lambda에서 실행되는 서버리스 애플리케이션에 대한 모니터링 및 문제 해결 솔루션입니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Lambda Insights 사용 단원을 참조하세요.	2020년 12월 3일

Prometheus JMX Exporter 지표에 대한 Amazon CloudWatch Application Insights 모니터링.	Amazon CloudWatch Application Insights를 사용하여 Prometheus JMX Exporter에서 검색된 지표를 모니터링할 수 있습니다.	2020년 11월 20일
CloudWatch Synthetics에서 새로운 런타임 버전 출시	CloudWatch Synthetics는 새 런타임 버전을 릴리스했습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 canary 런타임 버전 단원을 참조하세요.	2020년 9월 11일
Amazon RDS 및 Amazon EC2의 Postgre SQL에 대한 Amazon CloudWatch Application Insights 모니터링.	Amazon RDS 또는 Amazon EC2에서 실행되는 PostgreSQL을 사용하여 구축된 애플리케이션을 모니터링할 수 있습니다.	2020년 9월 11일
CloudWatch에서 대시보드 공유 지원	이제 조직 및 AWS 계정 외부의 사용자와 CloudWatch 대시보드를 공유할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 대시보드 공유 단원을 참조하세요.	2020년 9월 10일
CloudWatch Application Insights와 함께 백엔드의 SQL Server를 사용하여 .NET 애플리케이션에 대한 모니터 설정	설명서 튜토리얼을 활용함으로써 CloudWatch Application Insights와 함께 백엔드의 SQL Server를 사용하여 .NET 애플리케이션에 대한 모니터를 설정할 수 있습니다.	2020년 8월 19일

Amazon CloudWatch Application Insights 애플리케이션에 대한 AWS CloudFormation 지원	AWS CloudFormation 템플릿에서 직접 애플리케이션, 데이터베이스 및 웹 서버로 주요 지표 및 원격 측정을 포함한 CloudWatch Application Insights 모니터링을 추가할 수 있습니다.	2020년 7월 30일
Aurora for MySQL 데이터베이스 클러스터에 대한 Amazon CloudWatch Application Insights 모니터링.	Amazon CloudWatch Application Insights를 사용하여 Aurora for MySQL 데이터베이스 클러스터(RDS Aurora)를 모니터링할 수 있습니다.	2020년 7월 2일
CloudWatch Contributor Insights 공식 출시	이제 CloudWatch Contributor Insights가 정식 출시되었습니다. Contributor Insights를 사용하면 로그 데이터를 분석하고 기고자 데이터를 표시하는 시계열을 생성할 수 있습니다. 상위 N개의 기고자, 총 고유 기고자 수 및 사용량에 대한 지표를 볼 수 있습니다. 자세한 내용을 알아보려면 Amazon CloudWatch 사용 설명서의https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/ContributorInsights.htmlContributor Insights를 사용하여 카디널리티가 높은 데이터 분석하기를 참조하세요.	2020년 4월 2일

CloudWatch Synthetics 공개 미리 보기	이제 CloudWatch Synthetics는 공개 평가판입니다. Synthetics를 사용하면 카나리아를 생성하여 엔드포인트와 API를 모니터링할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 canary 사용 단원을 참조하세요.	2019년 11월 25일
CloudWatch Contributor Insights 공개 미리 보기	이제 CloudWatch Contributor Insights는 공개 평가판입니다. Contributor Insights를 사용하면 로그 데이터를 분석하고 기고자 데이터를 표시하는 시계열을 생성할 수 있습니다. 상위 N개의 기고자, 총 고유 기고자 수 및 사용량에 대한 지표를 볼 수 있습니다. 자세한 내용을 알아보려면 Amazon CloudWatch 사용 설명서의https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/ContributorInsights.htmlContributor Insights를 사용하여 카디널리티가 높은 데이터 분석하기를 참조하세요.	2019년 11월 25일
CloudWatch에서 ServiceLens 기능 출시	ServiceLens는 추적, 지표, 로그 및 경보를 한 곳에 통합할 수 있게 하여 서비스 및 애플리케이션의 관찰 가능성을 향상시킵니다. ServiceLens는 CloudWatch를 AWS X-Ray와 통합하여 애플리케이션에 대한 전체적인 보기를 제공합니다.	2019년 11월 21일

CloudWatch를 사용하여 사전에 AWS Service Quotas 관리	CloudWatch를 사용하여 AWS 서비스 할당량을 사전에 관리할 수 있습니다. CloudWatch 사용량 지표는 계정의 리소스 및 API 작업 사용량에 대한 가시성을 제공합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Service Quotas 통합 및 사용량 지표 단원을 참조하세요.	2019년 11월 19일
경보가 상태가 변경되면 CloudWatch에서 이벤트 전송	이제 CloudWatch는 CloudWatch 경보 상태가 변경될 때 Amazon EventBridge에 이벤트를 전송합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 경보 이벤트 및 EventBridge 단원을 참조하세요.	2019년 10월 8일
Container Insights	이제 CloudWatch Container Insights가 정식 출시되었습니다. 컨테이너 인사이트는 컨테이너 애플리케이션 및 마이크로서비스의 지표 및 로그를 수집하고 종합하며 요약합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Container Insights 사용 단원을 참조하세요.	2019년 8월 30일

Amazon EKS 및 Kubernetes의 Container Insights 미리 보기 지표 업데이트	Amazon EKS 및 Kubernetes의 Container Insights 공개 평가판이 업데이트되었습니다. 이제 InstanceId가 클러스터 EC2 인스턴스에 대한 측정기준으로 포함되어 있습니다. 이로써 이 지표에서 생성된 경보가 중지, 종료, 재부팅 또는 복구와 같은 EC2 작업을 트리거할 수 있게 되었습니다. 뿐만 아니라 이제 Pod 및 서비스 지표를 Kubernetes 네임스페이스에서 보고하여 지표에 대한 네임스페이스의 모니터링 및 경고가 간소화되었습니다.	2019년 8월 19일
AWS Systems Manager OpsCenter 통합에 대한 업데이트	CloudWatch Application Insights가 Systems Manager OpsCenter와 통합되는 방법에 대한 업데이트입니다.	2019년 8월 7일
CloudWatch 사용량 지표	CloudWatch 사용량 지표를 통해 CloudWatch 리소스 사용량을 추적하고 서비스 한도 내에서 유지할 수 있습니다. 자세한 내용은 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Usage-Metrics.html 단원을 참조하십시오.	2019년 8월 6일

CloudWatch Container Insights 공개 미리 보기	이제 CloudWatch Container Insights는 공개 평가판입니다. 컨테이너 인사이트는 컨테이너 애플리케이션 및 마이크로서비스의 지표 및 로그를 수집하고 종합하며 요약합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Container Insights 사용 단원을 참조하세요.	2019년 7월 9일
CloudWatch 이상 탐지 공개 미리 보기	이제 CloudWatch 이상 탐지는 공개 평가판입니다. CloudWatch에서는 머신 러닝 알고리즘을 지표의 과거 데이터에 적용하여 지표의 기대값 모델을 생성합니다. 이 모델을 시각화 및 경보 설정에 사용할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 이상 탐지 사용 단원을 참조하세요.	2019년 7월 9일
.NET 및 SQL Server용 CloudWatch Application Insights	.NET 및 SQL Server용 CloudWatch Application Insights에서 .NET 및 SQL Server 애플리케이션의 상태를 확인할 수 있습니다. 애플리케이션 리소스에 대한 최상의 모니터를 설정하고 데이터를 분석하여 애플리케이션 문제의 징후가 있는지 지속적으로 확인할 수 있습니다.	2019년 6월 21일

CloudWatch 에이전트 섹션 재구성	특히 에이전트를 설치하고 구성하는 데 명령줄을 사용하는 고객을 위해 CloudWatch 에이전트 설명서를 더욱 명확하게 재작성했습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집 단원을 참조하세요.	2019년 3월 28일
지표 수학 표현식에 SEARCH 함수 추가	이제 지표 수학 표현식에서 검색 기능을 사용할 수 있습니다. 이 기능을 통해 검색 쿼리와 일치하는 새 리소스가 생성되는 대로 자동으로 업데이트되는 대시보드를 생성할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 그래프에서 검색 표현식 사용 단원을 참조하세요.	2019년 3월 21일
Enterprise Support를 위한 AWS SDK 지표	SDK 지표를 사용하여 AWS 서비스의 상태를 평가하고 계정 사용 제한 도달 또는 서비스 중단으로 인해 발생하는 대기 시간을 진단할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 AWS SDK 지표를 사용하여 애플리케이션 모니터링 단원을 참조하세요.	2018년 12월 11일

수학 표현식에 대한 경보	CloudWatch는 지표 수학 표현식을 기반으로 하는 경보 생성을 지원합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 수학 표현식에 대한 경보 단원을 참조하세요.	2018년 11월 20일
새 CloudWatch 콘솔 홈 페이지	Amazon은 CloudWatch 콘솔에 사용 중인 모든 AWS 서비스에 대한 주요 지표 및 경보를 자동으로 표시하는 새로운 홈페이지를 만들었습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 시작하기 단원을 참조하세요.	2018년 11월 19일
CloudWatch 에이전트용 AWS CloudFormation 템플릿	Amazon은 CloudWatch 에이전트를 설치하고 업데이트하는 데 사용할 수 있는 AWS CloudFormation 템플릿을 업로드했습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 AWS CloudFormation을 사용하여 새 인스턴스에 CloudWatch 에이전트 설치 단원을 참조하세요.	2018년 11월 9일

CloudWatch 에이전트의 기능 향상	CloudWatch 에이전트가 StatsD 및 collectd 프로토콜 모두에서 작동하도록 업데이트되었습니다. 또한 교차 계정에 대한 지원도 개선되었습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 StatsD를 사용하여 사용자 지정 지표 검색, collectd를 사용하여 사용자 지정 지표 검색, 다른 AWS 계정에 지표 및 로그 전송 단원을 참조하세요.	2018년 9월 28일
Amazon VPC 엔드포인트에 대한 지원	이제 VPC와 CloudWatch 간에 프라이빗 연결을 설정할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 인터페이스 VPC 엔드포인트와 함께 CloudWatch 사용 단원을 참조하세요.	2018년 6월 28일

다음 표에서는 2018년 6월 이전 Amazon CloudWatch '사용 설명서'에서 변경된 중요 사항을 설명합니다.

변경 사항	설명	릴리스 날짜
지표 수식	이제 CloudWatch 지표에서 수학 표현식을 실행하여 대시보드의 그래프에 추가할 수 있는 새로운 시계열을 생성할 수 있습니다. 자세한 내용은 지표 수학 사용 단원을 참조하십시오.	2018년 4월 4일
"N 중 M" 경보	이제는 경보 평가 간격에서 "N 중 M" 데이터 포인트를 기반으로 경보가 트리거되도록 구성할 수 있습니다. 자세한 내용은 경보 평가 단원을 참조하십시오.	2017년 12월 8일
CloudWatch 에이전트	새로운 통합 CloudWatch 에이전트가 릴리스되었습니다. 통합 다중 플랫폼 에이전트를 사용하여	2017년 9월 7일

변경 사항	설명	릴리스 날짜
	Amazon EC2 인스턴스 및 온프레미스 서버에서 사용자 지정 시스템 지표와 로그 파일을 모두 수집할 수 있습니다. 새로운 에이전트는 Windows Server 및 Linux를 모두 지원하며 CPU당 코어와 같은 하위 리소스 지표를 포함하여 수집할 지표를 사용자 지정할 수 있습니다. 자세한 내용은 CloudWatch 에이전트를 사용하여 지표, 로그, 추적 수집 단원을 참조하십시오.	
NAT 게이트웨이 지표	Amazon VPC NAT 게이트웨이에 대한 지표가 추가되었습니다.	2017년 9월 7일
고분해능 지표	선택적으로, 이제 사용자 지정 지표를 최대 1초 세분화의 고분해능 지표로 설정할 수 있습니다. 자세한 내용은 고분해능 지표 단원을 참조하십시오.	2017년 7월 26일
대시보드 API	이제 API 및 AWS CLI를 사용하여 대시보드를 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 CloudWatch 대시보드 생성 단원을 참조하십시오.	2017년 6월 7일
AWS Direct Connect 지표	AWS Direct Connect에 대한 지표가 추가되었습니다.	2017년 6월 29일
Amazon VPC VPN 지표	Amazon VPC VPN에 대한 지표가 추가되었습니다.	2017년 5월 15일
AppStream 2.0 지표	AppStream 2.0에 대한 지표가 추가되었습니다.	2017년 3월 8일
CloudWatch 콘솔 Color Picker	대시보드 위젯에서 각 지표에 대한 색상을 선택할 수 있습니다. 자세한 내용은 CloudWatch 대시보드에서 그래프 편집 단원을 참조하십시오.	2017년 2월 27일
대시보드의 경보	경보를 대시보드에 추가할 수 있습니다. 자세한 내용은 CloudWatch 대시보드에서 경보 위젯 추가 또는 제거 단원을 참조하십시오.	2017년 2월 15일

변경 사항	설명	릴리스 날짜
Amazon Polly에 대한 지표 추가	Amazon Polly에 대한 지표가 추가되었습니다.	2016년 1월 12일
Amazon Managed Service for Apache Flink에 대한 지표 추가	Amazon Managed Service for Apache Flink에 대한 지표가 추가되었습니다.	2016년 1월 12일
백분위수 통계에 대한 지원 추가	소수점 두 자리까지 사용하여 백분위 수를 지정할 수 있습니다(예: p95.45). 자세한 내용은 백분위수 단원을 참조하십시오.	2016년 11월 17일
Amazon Simple Email Service에 대한 지표 추가	Amazon Simple Email Service에 대한 지표가 추가되었습니다.	2016년 11월 2일
지표 보존 기간 업데이트	Amazon CloudWatch는 이제 14일이 아닌 15개월 동안 지표 데이터를 유지합니다.	2016년 11월 1일
지표 콘솔 인터페이스 업데이트	CloudWatch 콘솔은 기존 기능을 개선하고 새 기능을 추가하는 등 업데이트되었습니다.	2016년 11월 1일
Amazon Elastic Transcoder에 대한 지표 추가	Amazon Elastic Transcoder에 대한 지표가 추가되었습니다.	2016년 9월 20일
Amazon API Gateway에 대한 지표 추가	Amazon API Gateway에 대한 지표가 추가되었습니다.	2016년 9월 9일
AWS Key Management Service에 대한 지표 추가	AWS Key Management Service에 대한 지표가 추가되었습니다.	2016년 9월 9일

변경 사항	설명	릴리스 날짜
Elastic Load Balancing에서 지원하는 새로운 Application Load Balancer에 대한 지표 추가	Application Load Balancer에 대한 지표가 추가되었습니다.	2016년 8월 11일
Amazon EC2에 대한 새로운 NetworkPacketsIn 및 NetworkPacketsOut 측정치를 추가함.	Amazon EC2에 대한 새로운 NetworkPacketsIn 및 NetworkPacketsOut 측정치를 추가함.	2016년 3월 23일
Amazon EC2 스팟 플릿에 대한 새 지표 추가	Amazon EC2 스팟 플릿에 대한 새 지표가 추가되었습니다.	2016년 3월 21일
새 CloudWatch Logs 지표 추가	새 CloudWatch Logs 지표가 추가되었습니다.	2016년 3월 10일
Amazon OpenSearch Service 및 AWS WAF 지표 및 차원 추가	Amazon OpenSearch Service 및 AWS WAF 지표 및 차원이 추가되었습니다.	2015년 10월 14일
CloudWatch 대시보드에 대한 지원 추가	대시보드는 CloudWatch 콘솔에서 사용자 지정이 가능한 홈페이지로, 단일 보기에서 리소스(다양한 리전에 분산되어 있는 리소스 포함)를 모니터링하는 데 사용할 수 있습니다. 자세한 내용은 Amazon CloudWatch 대시보드 사용 단원을 참조하십시오.	2015년 10월 8일

변경 사항	설명	릴리스 날짜
AWS Lambda 지표 및 측정기준 추가	AWS Lambda 지표 및 측정기준이 추가되었습니다.	2015년 9월 4일
Amazon Elastic Container Service 지표 및 측정기준 추가	Amazon Elastic Container Service 지표 및 측정기준이 추가되었습니다.	2015년 8월 17일
Amazon Simple Storage Service 지표 및 측정기준 추가	Amazon Simple Storage Service 지표 및 측정기준이 추가되었습니다.	2015년 7월 26일
새로운 기능: 경보 작업 재부팅	재부팅 경보 작업과 경보 작업에 사용할 새로운 IAM 역할이 추가되었습니다. 자세한 내용은 EC2 인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성 단원을 참조하십시오.	2015년 7월 23일
Amazon WorkSpaces 지표 및 측정기준 추가	Amazon WorkSpaces 지표 및 측정기준이 추가되었습니다.	2015년 4월 30일
Amazon Machine Learning 지표 및 측정기준 추가	Amazon Machine Learning 지표 및 측정기준이 추가되었습니다.	2015년 4월 9일
새로운 기능: Amazon EC2 인스턴스 복구 경보 작업	새로운 EC2 인스턴스 복구 작업이 포함되도록 경보 작업이 업데이트되었습니다. 자세한 내용은 EC2 인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성 단원을 참조하십시오.	2015년 3월 12일

변경 사항	설명	릴리스 날짜
Amazon CloudFront 및 Amazon CloudSearch 지표 및 측정기준 추가	Amazon CloudFront 및 Amazon CloudSearch 지표 및 측정기준이 추가되었습니다.	2015년 3월 6일
Amazon Simple Workflow Service 지표 및 측정기준 추가	Amazon Simple Workflow Service 지표 및 측정기준이 추가되었습니다.	2014년 5월 9일
AWS CloudTrail에 대한 지원을 추가하도록 설명서 업데이트	AWS CloudTrail을 사용하여 Amazon CloudWatch에서 활동을 로그하는 방법을 설명하는 새 주제가 추가되었습니다. 자세한 내용은 AWS CloudTrail으로 Amazon CloudWatch API 호출 로그하기 단원을 참조하십시오.	2014년 4월 30일
새 AWS Command Line Interface(AWS CLI)를 사용하도록 설명서 업데이트	AWS CLI는 간단한 설치, 통합 구성, 일관된 명령 줄 구문을 특징으로 하는 교차 서비스 CLI입니다. AWS CLI는 Linux/Unix, Windows 및 Mac에서 지원됩니다. 이 설명서의 CLI 예제는 새 AWS CLI를 사용하도록 업데이트되었습니다. 새 AWS CLI를 설치 및 구성하는 방법에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 인터페이스 설정 단원을 참조하세요.	2014년 21월 2일
Amazon Redshift 및 AWS OpsWorks 지표 및 측정기준 추가	Amazon Redshift 및 AWS OpsWorks 지표 및 측정기준이 추가되었습니다.	2013년 7월 16일

변경 사항	설명	릴리스 날짜
Amazon Route 53 지표 및 측정 기준 추가	Amazon Route 53 지표 및 측정기준이 추가되었습니다.	2013년 6월 26일
새로운 기능: Amazon CloudWatch 경보 작업	Amazon Elastic Compute Cloud 인스턴스를 중지하거나 종료하는 데 사용할 수 있는 Amazon CloudWatch 경보 작업을 문서화하는 새 단원이 추가되었습니다. 자세한 내용은 EC2 인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성 단원을 참조하십시오.	2013년 1월 8일
EBS 지표가 업데이트됨	프로비저닝 IOPS 볼륨에 대해 새 지표 두 개를 포함하도록 EBS 지표가 업데이트되었습니다.	2012년 11월 20일
새로운 결제 알림	이제 Amazon CloudWatch 지표를 사용하여 AWS 요금을 모니터링하고 지정된 임곗값을 초과했을 때 이를 알려 주는 경보를 생성할 수 있습니다. 자세한 내용은 청구 경보를 생성하여 예상 AWS 요금을 모니터링하세요. 단원을 참조하십시오.	2012년 5월 10일
새로운 지표	이제 다양한 HTTP 응답 코드 수를 제공하는 6가지 새로운 Elastic Load Balancing 지표에 액세스할 수 있습니다.	2011년 10월 19일
새 기능	이제 Amazon EMR의 지표에 액세스할 수 있습니다.	2011년 6월 30일
새 기능	이제 Amazon Simple Notification Service 및 Amazon Simple Queue Service의 지표에 액세스할 수 있습니다.	2011년 7월 14일
새 기능	`PutMetricData` API를 사용하여 사용자 지정 지표를 게시하는 방법에 대한 정보가 추가되었습니다. 자세한 내용은 사용자 지정 지표 게시 단원을 참조하십시오.	2011년 5월 10일

변경 사항	설명	릴리스 날짜
지표 보존 기간 업데이트	이제 Amazon CloudWatch는 6주가 아닌 2주 동안 경보 기록을 유지합니다. 따라서 경보의 유지 기간이 지표 데이터의 유지 기간과 일치하게 되었습니다.	2011년 4월 7일
새 기능	지표가 임곗값을 초과할 때 Amazon Simple Notification Service 또는 Auto Scaling 알림을 보내는 기능이 추가되었습니다. 자세한 내용은 경보 단원을 참조하십시오.	2010년 12월 2일
새 기능	이제 여러 CloudWatch 작업에 MaxRecords 및 NextToken 파라미터가 포함됩니다. 이러한 파라미터를 사용하면 표시할 결과 페이지를 제어할 수 있습니다.	2010년 12월 2일
새 기능	이 서비스는 이제 AWS Identity and Access Management(IAM)와 통합됩니다.	2010년 12월 2일