사용자 가이드

AWS Elemental MediaStore

AWS Elemental MediaStore: 사용자 가이드

Table of Contents

지원 종료 공지: 2025년 11월 13일에 AWS 는 AWS Elemental MediaStore에 대한 지원을 중단합니다. 2025년 11월 13일 이후에는 더 이상 MediaStore 콘솔 또는 MediaStore 리소스에 액세스할 수 없습니다. 자세한 내용은 이 블로그 게시물을 참조하세요.

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

AWS Elemental MediaStore란 무엇인가요?

AWS Elemental MediaStore는 라이브 제작에 필요한 우수한 성능과 즉각적 일관성을 제공하는 비디오 제작 및 스토리지 서비스입니다. MediaStore를 사용하여 비디오 자산을 컨테이너에서 객체로 관리함으로써 안전적인 클라우드 기반 미디어 워크플로를 구축할 수 있습니다.

서비스를 사용하려면 MediaStore에서 인코더나 데이터 피드와 같은 소스의 객체를 만든 컨테이너에 업로드합니다.

MediaStore는 완벽한 일관성, 최소 지연 시간의 읽기 및 쓰기, 많은 양의 동시 요청을 처리할 수 있는 능력이 요구될 때 조각화된 비디오 파일을 저장할 수 있는 훌륭한 수단입니다. 라이브 스트리밍 비디오를 전송하지 않을 경우 Amazon Simple Storage Service(Amazon S3) 를 대신 사용해 보십시오.

주제

AWS Elemental MediaStore 개념 및 용어

ARN

Amazon 리소스 이름.

Body

객체에 업로드할 데이터.

(바이트) 범위

주소 지정할 객체 데이터 하위 집합. 자세한 내용은 HTTP 사양의 범위를 참조하세요.

컨테이너

객체를 포함하는 네임스페이스. 컨테이너에는 객체를 쓰고 검색하고, 액세스 정책을 연결하기 위해 사용할 수 있는 엔드포인트가 있습니다.

엔드포인트

MediaStore 서비스에 대한 진입점(HTTPS 루트 URL로 지정)

ETag

객체 데이터의 해시인 개체 태그.

폴더

컨테이너의 분할 요소. 폴더는 객체와 다른 폴더를 포함할 수 있습니다.

Item

객체와 폴더를 나타내는 데 사용하는 용어.

객체

Amazon S3 객체와 비슷한 자산 객체는 MediaStore에 저장되는 기본 객체입니다. 이 서비스는 모든 유형의 파일을 허용합니다.

제작 서비스

MediaStore는 미디어 콘텐츠 전송을 위한 배포 지점이라는 점에서 제작 서비스로 간주됩니다.

경로

객체나 폴더에 대한 고유 식별자로서, 컨테이너에서의 위치를 나타냅니다.

부품

객체 데이터의 하위 집합(청크)

정책

IAM 정책.

리소스

작업에 사용할 수 있는 AWS의 엔터티. 각 AWS 리소스에는 고유한 식별자인 Amazon 리소스 이름(ARN)이 할당됩니다. MediaStore에서 리소스 및 해당 ARN 형식은 다음과 같습니다.

- 컨테이너: `aws:mediastore:region:account-id:container/:containerName`

관련 서비스

- Amazon CloudFront는 최종 사용자에게 데이터와 비디오를 안전하게 전송하는 글로벌 콘텐츠 배포 네트워크(CDN) 서비스입니다. CloudFront를 사용하여 최고의 성능으로 콘텐츠가 제공됩니다. 자세한 내용은 Amazon CloudFront 개발자 안내서를 참조하세요.

- CloudFormation는 AWS 리소스를 모델링하고 설정하는 데 도움이 되는 서비스입니다. 원하는 모든 리소스(예: MediaStore 컨테이너)를 AWS 설명하는 템플릿을 생성하고 해당 리소스를 CloudFormation 프로비저닝하고 구성합니다. AWS 리소스를 개별적으로 생성 및 구성하고이 모든 것을 CloudFormation 처리하는 것이 무엇인지 파악할 필요가 없습니다. 자세한 내용은 AWS CloudFormation 사용 설명서를 참조하십시오.
- AWS CloudTrail는 AWS Management Console 및 AWS CLI기타 서비스의 호출을 포함하여 계정의 CloudTrail API에 대한 호출을 모니터링할 수 있는 서비스입니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.
- Amazon CloudWatch는 AWS 클라우드 리소스 및 실행 중인 애플리케이션에 대한 모니터링 서비스입니다 AWS. CloudWatch Events를 사용하여 MediaStore의 컨테이너와 객체 상태에 대한 변화를 추적할 수 있습니다. 자세한 내용은 Amazon CloudWatch 설명서를 참조하세요.
- AWS Identity and Access Management (IAM)는 사용자의 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 되는 웹 서비스입니다. IAM을 사용하여 AWS 리소스를 사용할 수 있는 사용자(인증)와 사용자가 어떤 방식으로 사용할 수 있는 리소스(권한 부여)를 제어합니다. 자세한 내용은 AWS Elemental MediaStore 설정 단원을 참조하십시오.
- Amazon Simple Storage Service(Amazon S3)는 어디서든 원하는 양의 데이터를 저장하고 검색할 수 있도록 구축된 객체 스토리지입니다. 자세한 내용은 Amazon S3 설명서를 참조하십시오.

AWS Elemental MediaStore에 액세스

다음 방법 중 하나를 사용하여 MediaStore에 액세스할 수 있습니다.

- AWS Management Console - 이 설명서의 절차에서는 AWS Management Console을 사용하여 MediaStore에 대한 작업을 수행하는 방법을 설명합니다. 콘솔을 사용하여 MediaStore에 액세스하기

```
https://<region>.console.aws.amazon.com/mediastore/home
```

- AWS Command Line Interface – 자세한 내용은 AWS Command Line Interface 사용 설명서를 참조하세요. CLI 엔드포인트를 사용하여 MediaStore에 액세스하기

```
aws mediastore
```

- MediaStore API - SDK가 제공되지 않는 프로그래밍 언어를 사용하는 경우, AWS Elemental MediaStore API 참조에서 API 작업에 대한 정보와 API 요청을 수행하는 방법을 참조하세요. REST API 엔드포인트를 사용하여 MediaStore에 액세스하기

```
https://mediastore.<region>.amazonaws.com
```

- AWS SDK - AWS가 SDK를 제공하는 프로그래밍 언어를 사용하는 경우, SDK를 사용하여 MediaStore에 액세스할 수 있습니다. SDK는 인증을 간편하게 만들고, 개발 환경에 쉽게 통합되며, MediaStore 명령어에 쉽게 액세스할 수 있게 해줍니다. 자세한 내용은 Amazon Web Services용 도구를 참조하세요.
- AWS Tools for Windows PowerShell - 자세한 내용은 AWS Tools for PowerShell 사용 설명서를 참조하세요.

AWS Elemental MediaStore 요금

다른 AWS 제품과 마찬가지로 MediaStore 사용에 대한 계약이나 최소 약정은 없습니다. 서비스에 콘텐츠가 제공될 때 GB당 수집 요금이 청구되고, 서비스에 저장하는 콘텐츠에 대해 GB당 월 요금이 청구됩니다. 자세한 내용은 AWS Elemental MediaStore 요금을 참조하세요.

AWS Elemental MediaStore에 사용되는 리전 및 엔드포인트

애플리케이션에서 데이터 지연 시간을 줄이기 위해 MediaStore는 요청을 수행하는 리전 엔드포인트를 제공합니다.

```
https://mediastore.<region>.amazonaws.com
```

MediaStore를 사용할 수 있는 AWS 리전 전체 목록은 AWS 일반 참조에서 AWS Elemental MediaStore 엔드포인트 및 할당량을 참조하세요.

AWS Elemental MediaStore 설정

이 섹션에서는 사용자를 구성하여 AWS Elemental MediaStore에 액세스하는 데 필요한 단계를 안내합니다. MediaStore용 자격 증명 및 액세스 관리에 대한 배경 설명 및 추가적인 정보는 AWS Elemental MediaStore에 대한 자격 증명 및 액세스 관리 섹션을 참조하세요.

AWS Elemental MediaStore를 사용하기 전에 먼저 다음 단계를 완료합니다.

주제

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

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

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

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

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

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

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS Sign-In User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

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

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

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

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

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

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS Sign-In 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.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

AWS Elemental MediaStore 시작하기

이 시작하기 튜토리얼에서는 AWS Elemental MediaStore를 사용하여 컨테이너를 만들고 객체를 업로드하는 방법을 보여 줍니다.

주제

1단계: AWS Elemental MediaStore에 액세스

AWS 계정을 설정하고 사용자와 역할을 만들었으면 AWS Elemental MediaStore용 콘솔에 로그인합니다.

AWS Elemental MediaStore에 액세스하기

- 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/mediastore/ MediaStore 콘솔을 엽니다.

> **Note**
>
> 이 계정에 대해 만든 IAM 자격 증명을 사용하여 로그인할 수 있습니다. IAM 자격 증명 만들기에 대한 자세한 내용은 AWS Elemental MediaStore 설정 섹션을 참조하세요.

2단계: 컨테이너 생성

AWS Elemental MediaStore의 컨테이너를 사용하여 폴더와 객체를 저장합니다. 디렉터리로 파일 시스템의 파일을 그룹화하는 것처럼 컨테이너를 사용하여 관련 객체를 그룹화할 수 있습니다. 컨테이너를 만들 때에는 요금이 청구되지 않습니다. 컨테이너에 객체를 업로드해야만 요금이 청구됩니다.

컨테이너 생성

1. 컨테이너 페이지에서 컨테이너 생성을 선택합니다.

2. 컨테이너 이름(Container name)에 컨테이너 이름을 입력합니다. 자세한 내용은 컨테이너 이름에 대한 규칙 단원을 참조하십시오.
3. 컨테이너 생성을 선택합니다. AWS Elemental MediaStore는 새 컨테이너를 컨테이너 목록에 추가합니다. 처음에 컨테이너의 상태는 생성 중이고 그 다음에 활성으로 변경됩니다.

3단계: 객체 업로드

컨테이너 또는 컨테이너의 폴더에 객체(각각 최대 25MB)를 업로드할 수 있습니다. 폴더에 객체를 업로드하려면 폴더 경로를 지정합니다. 폴더가 이미 있을 경우 AWS Elemental MediaStore는 폴더에 객체를 저장합니다. 폴더가 없으면 폴더를 만든 후 그 폴더에 객체를 저장합니다.

> **Note**
>
> 객체 파일 이름에는 문자, 숫자, 마침표(.), 밑줄(_), 물결 기호(~), 하이픈(-)만 사용할 수 있습니다.

객체 업로드

1. 컨테이너 페이지에서 방금 만든 컨테이너 이름을 선택합니다. 컨테이너의 세부 정보 페이지가 나타납니다.
2. 객체 업로드를 선택합니다.
3. 대상 경로에 폴더 경로를 입력합니다. 예를 들어 `premium/canada`입니다. 이 경로에 폴더가 존재하지 않으면 AWS Elemental MediaStore는 해당 폴더를 자동으로 만듭니다.
4. 객체에서 찾아보기를 선택합니다.
5. 해당 폴더로 이동한 후 업로드할 객체를 하나 선택합니다.
6. 열기를 선택한 후 업로드를 선택합니다.

4단계: 객체 액세스

지정한 엔드포인트로 객체를 다운로드할 수 있습니다.

1. 컨테이너 페이지에서, 다운로드하려는 객체가 들어 있는 컨테이너 이름을 선택합니다.
2. 다운로드하려는 객체가 하위 폴더에 있으면 해당 객체가 보일 때까지 폴더 이름을 계속 선택합니다.

3. 객체 이름을 선택합니다.
4. 객체에 대한 세부 정보 페이지에서 다운로드를 선택합니다.

AWS Elemental MediaStore의 컨테이너

MediaStore의 컨테이너를 사용하여 폴더와 객체를 저장합니다. 디렉터리를 사용하여 파일 시스템의 파일을 그룹화하듯이 관련 객체를 컨테이너에 그룹화할 수 있습니다. 컨테이너를 만들 때에는 요금이 청구되지 않습니다. 컨테이너에 객체를 업로드해야만 요금이 청구됩니다. 요금에 대한 자세한 내용은 AWS Elemental MediaStore 요금을 참조하세요.

주제

컨테이너 이름에 대한 규칙

컨테이너의 이름을 선택할 때 다음에 유의하세요.

- 이름은 현재 AWS 리전의 현재 계정에서 고유해야 합니다.
- 이름은 대문자, 소문자, 숫자, 밑줄(_)을 포함할 수 있습니다.
- 이름은 길이가 1자~255자여야 합니다.
- 이름은 대/소문자를 구분합니다. 예를 들어 `myContainer`라는 컨테이너와 `mycontainer`라는 폴더는 이름이 고유하므로 둘 다 둘 수 있습니다.
- 컨테이너를 만든 후 이름을 변경할 수 없습니다.

컨테이너 생성

AWS 계정당 최대 100개의 컨테이너를 만들 수 있습니다. 폴더가 컨테이너 내에서 10개를 초과하는 수준으로 중첩되지 않는 한 원하는 개수만큼 폴더를 생성할 수 있습니다. 또한 각 컨테이너에 원하는 개수만큼 객체를 업로드할 수 있습니다.

> **Tip**
>
> CloudFormation 템플릿을 사용하여 컨테이너를 자동으로 생성할 수도 있습니다. CloudFormation 템플릿은 컨테이너 생성, 액세스 로깅 설정, 기본 컨테이너 정책 업데이트, CORS(교차 원본 리소스 공유) 정책 추가, 객체 수명 주기 정책 추가 등의 5가지 API 작업에 대한 데이터를 관리합니다. 자세한 내용은 AWS CloudFormation 사용 설명서를 참조하십시오.

컨테이너를 만들려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 컨테이너 생성을 선택합니다.
3. 컨테이너 이름에 컨테이너 이름을 입력합니다. 자세한 내용은 컨테이너 이름에 대한 규칙 단원을 참조하십시오.
4. 컨테이너 생성을 선택합니다. AWS Elemental MediaStore는 새 컨테이너를 컨테이너 목록에 추가합니다. 처음에 컨테이너의 상태는 생성 중이고 그 다음에 활성으로 변경됩니다.

컨테이너를 만들려면(AWS CLI)

- 에서 `create-container` 명령을 AWS CLI사용합니다.

```
aws mediastore create-container --container-name ExampleContainer --region us-
west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "Container": {
        "AccessLoggingEnabled": false,
        "CreationTime": 1563557265.0,
        "Name": "ExampleContainer",
        "Status": "CREATING",
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
    }
}
```

컨테이너에 대한 세부 정보 보기

컨테이너 세부 정보에는 컨테이너 정책, 엔드포인트, ARN, 생성 시간이 포함됩니다.

컨테이너에 대한 세부 정보를 보려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 컨테이너 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다. 이 페이지는 두 섹션으로 나뉘어 있습니다.

   - 객체 섹션에는 컨테이너의 객체와 폴더가 나열됩니다.
   - 컨테이너 정책 섹션에는 해당 컨테이너와 연결된 리소스 기반 정책이 표시됩니다. 리소스 정책에 대한 자세한 내용은 컨테이너 정책 섹션을 참조하세요.

컨테이너에 대한 세부 정보를 보려면(AWS CLI)

- 에서 `describe-container` 명령을 AWS CLI사용합니다.

```
aws mediastore describe-container --container-name ExampleContainer --region us-
west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "Container": {
        "CreationTime": 1563558086.0,
        "AccessLoggingEnabled": false,
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
        "Status": "ACTIVE",
        "Name": "ExampleContainer",
        "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com"
    }
}
```

컨테이너 목록 보기

계정과 연결된 모든 컨테이너의 목록을 볼 수 있습니다.

컨테이너 목록을 보려면(콘솔)

- https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.

  해당 계정과 연결된 모든 컨테이너가 나열된 컨테이너 페이지가 나타납니다.

컨테이너 목록을 보려면(AWS CLI)

- 에서 `list-containers` 명령을 AWS CLI사용합니다.

```
aws mediastore list-containers --region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "Containers": [
        {
            "CreationTime": 1505317931.0,
            "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
            "AccessLoggingEnabled": false,
            "Name": "ExampleLiveDemo"
        },
        {
            "CreationTime": 1506528818.0,
            "Endpoint": "https://fffggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
            "AccessLoggingEnabled": false,
            "Name": "ExampleContainer"
        }
    ]
```

```
}
```

컨테이너 삭제

객체가 없는 컨테이너만 삭제할 수 있습니다.

컨테이너를 삭제하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 해당 컨테이너 이름 왼쪽의 옵션을 선택합니다.
3. 삭제를 선택합니다.

컨테이너를 삭제하려면(AWS CLI)

- 에서 `delete-container` 명령을 AWS CLI사용합니다.

  ```
  aws mediastore delete-container --container-name=ExampleLiveDemo --region us-west-2
  ```

  이 명령은 반환 값이 없습니다.

AWS Elemental MediaStore 정책

AWS Elemental MediaStore 컨테이너에 다음 정책을 하나 이상 적용할 수 있습니다.

- 컨테이너 정책 - 컨테이너 내의 모든 폴더와 객체에 대한 액세스 권한을 설정합니다. MediaStore는 사용자가 컨테이너에서 모든 MediaStore 작업을 수행할 수 있도록 허용하는 기본 정책을 설정합니다. 이 정책은 모든 작업이 HTTPS를 통해 수행되도록 지정합니다. 컨테이너를 생성한 후 컨테이너 정책을 편집할 수 있습니다.
- 교차 오리진 리소스 공유(CORS) 정책 - 한 도메인의 클라이언트 웹 애플리케이션이 다른 도메인의 리소스와 상호 작용할 수 있도록 허용합니다. MediaStore는 기본 CORS 정책을 설정하지 않습니다.
- 지표 정책 - MediaStore가 Amazon CloudWatch에 지표를 전송하도록 허용합니다. MediaStore는 기본 지표 정책을 설정하지 않습니다.
- 객체 수명 주기 정책 - 객체가 MediaStore 컨테이너에 남아 있는 기간을 제어합니다. MediaStore는 기본 객체 수명 주기 정책을 설정하지 않습니다.

AWS Elemental MediaStore의 컨테이너 정책

각 컨테이너에는 해당 컨테이너의 모든 폴더와 객체에 대한 액세스를 관리하는 리소스 기반 정책이 있습니다. 모든 새 컨테이너에 자동으로 연결되는 기본 정책은 컨테이너에서 모든 AWS Elemental MediaStore 작업에 대한 액세스를 허용합니다. 이러한 액세스는 작업에 대해 HTTPS를 요구하는 조건을 갖도록 지정됩니다. 컨테이너를 만든 후 해당 컨테이너에 연결되는 정책을 편집할 수 있습니다.

또한 컨테이너에서 객체의 만료 날짜를 관리하는 객체 수명 주기 정책을 지정할 수 있습니다. 객체가 지정된 최대 수명에 도달하면 서비스가 해당 객체를 컨테이너에서 삭제합니다.

항목

- 컨테이너 정책 보기
- 컨테이너 정책 편집
- 컨테이너 정책 예제

컨테이너 정책 보기

콘솔 또는를 사용하여 컨테이너의 리소스 기반 정책을 AWS CLI 볼 수 있습니다.

컨테이너 정책을 보려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 컨테이너 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다. 컨테이너 정책 섹션에 정책이 표시됩니다.

컨테이너 정책을 보려면(AWS CLI)

- 에서 `get-container-policy` 명령을 AWS CLI사용합니다.

```
aws mediastore get-container-policy --container-name ExampleLiveDemo --region us-
west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
  "Policy": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "PublicReadOverHttps",
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::111122223333:root",
        },
        "Action": [
          "mediastore:GetObject",
          "mediastore:DescribeObject",
        ],
        "Resource": "arn:aws:mediastore:us-
west-2:111122223333:container/ExampleLiveDemo/*",
        "Condition": {
          "Bool": {
            "aws:SecureTransport": "true"
          }
        }
      }
    ]
  }
}
```

컨테이너 정책 편집

기본 컨테이너 정책의 권한을 편집하거나, 기본 정책을 대체할 새 정책을 만들 수 있습니다. 새 정책이 적용되려면 최대 5분이 걸립니다.

컨테이너 정책을 편집하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 컨테이너 이름을 선택합니다.
3. 정책 편집을 선택합니다. 다양한 권한을 설정하는 방법을 보여 주는 예제는 the section called “컨테이너 정책 예제” 단원을 참조하십시오.
4. 정책을 적절히 변경하고 저장을 선택합니다.

컨테이너 정책을 편집하려면(AWS CLI)

1. 다음과 같이 컨테이너 정책을 정의하는 파일을 만듭니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadOverHttps",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal": "*",
      "Resource": "arn:aws:mediastore:us-
west-2:111122223333:container/ExampleLiveDemo/*",
      "Condition": {
        "Bool": {
            "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

2. 에서 `put-container-policy` 명령을 AWS CLI사용합니다.

```
aws mediastore put-container-policy --container-name ExampleLiveDemo --
policy file://ExampleContainerPolicy.json --region us-west-2
```

이 명령은 반환 값이 없습니다.

컨테이너 정책 예제

다음 예제는 여러 사용자 그룹에 대해 구성된 컨테이너 정책을 보여 줍니다.

항목

- 컨테이너 정책 예제: 기본
- 컨테이너 정책 예제: HTTPS를 통해 퍼블릭 읽기 액세스
- 컨테이너 정책 예제: HTTP 또는 HTTPS를 통한 퍼블릭 읽기 액세스
- 컨테이너 정책 예제: 교차 계정 읽기 액세스-HTTP 활성화
- 컨테이너 정책 예제: HTTPS를 통한 교차 계정 읽기 액세스
- 컨테이너 정책 예제: 역할에 대한 교차 계정 읽기 액세스
- 컨테이너 정책 예제: 역할에 대한 교차 계정 전체 액세스
- 컨테이너 정책 예제: 특정 IP 주소로 제한된 액세스

컨테이너 정책 예제: 기본

컨테이너를 만들면 AWS Elemental MediaStore에서 다음 리소스 기반 정책을 자동으로 연결합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "MediaStoreFullAccess",
            "Action": [
                "mediastore:*"
            ],
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:root"
```

```
            },
            "Effect": "Allow",
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

이 정책은 서비스에 포함되므로 따로 만들 필요가 없습니다. 그러나 기본 정책의 권한이 컨테이너에 사용하려는 권한과 일치하지 않는 경우 컨테이너에서 정책을 편집할 수 있습니다.

모든 새 컨테이너에 할당되는 기본 정책은 컨테이너에서 모든 MediaStore 작업에 대한 액세스를 허용합니다. 이러한 액세스는 작업에 대해 HTTPS를 요구하는 조건을 갖도록 지정됩니다.

컨테이너 정책 예제: HTTPS를 통해 퍼블릭 읽기 액세스

이 예제 정책은 사용자가 HTTPS 요청을 통해 객체를 검색할 수 있도록 허용합니다. 이 정책은 모든 사용자, 즉 인증된 사용자와 익명 사용자(로그인하지 않은 사용자)에게 보안 SSL/TLS 연결을 통한 읽기 액세스를 허용합니다. 구문에 `PublicReadOverHttps` 이름이 있습니다. 모든 객체(리소스 경로 끝에 * 기호로 지정)에 대해 `GetObject` 및 `DescribeObject` 작업에 대한 액세스를 허용합니다. 이러한 액세스는 작업에 대해 HTTPS를 요구하는 조건을 갖도록 지정됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadOverHttps",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": "*",
```

```
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

컨테이너 정책 예제: HTTP 또는 HTTPS를 통한 퍼블릭 읽기 액세스

이 예제 정책은 모든 객체(리소스 경로 끝에 * 기호로 지정)에 대해 `GetObject` 및 `DescribeObject` 작업에 대한 액세스를 허용합니다. 모든 인증된 사용자와 익명 사용자(로그인하지 않은 사용자)를 포함하여 모든 사용자에게 읽기 액세스를 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadOverHttpOrHttps",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": "*",
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

컨테이너 정책 예제: 교차 계정 읽기 액세스-HTTP 활성화

이 예제 정책은 사용자가 HTTP 요청을 통해 객체를 검색할 수 있도록 허용합니다. 교차 계정 액세스 권한을 가진 인증된 사용자에 대해 이 액세스를 허용합니다. 객체가 SSL/TLS 인증서를 사용하여 서버에서 호스트될 필요는 없습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountReadOverHttpOrHttps",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:root"
            },
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

컨테이너 정책 예제: HTTPS를 통한 교차 계정 읽기 액세스

이 예제 정책은 지정한 <다른 계정 번호>의 루트 사용자가 소유한 모든 객체(리소스 경로 끝에 * 기호로 지정)에 대해 `GetObject` 및 `DescribeObject` 작업에 대한 액세스를 허용합니다. 이러한 액세스는 작업에 대해 HTTPS를 요구하는 조건을 갖도록 지정됩니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountReadOverHttps",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:root"
            },
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

컨테이너 정책 예제: 역할에 대한 교차 계정 읽기 액세스

이 예제 정책은 <소유자 계정 번호>가 소유한 모든 객체(리소스 경로 끝에 * 기호로 지정)에 대해 GetObject 및 DescribeObject 작업에 대한 액세스를 허용합니다. <역할 이름>에 지정된 역할을 해당 계정이 가지고 있을 경우 <다른 계정 번호>의 모든 사용자에 대해 이 액세스를 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountRoleRead",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal":{
        "AWS": "arn:aws:iam::<other acct number>:role/<role name>"},
```

```
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
    }
  ]
}
```

컨테이너 정책 예제: 역할에 대한 교차 계정 전체 액세스

이 예제 정책은 계정의 객체를 업데이트할 수 있는 교차 계정 액세스 권한을 허용합니다. 단, 사용자가 HTTP를 통해 로그인해야 합니다. 또한 지정된 역할을 맡은 계정에 HTTP 또는 HTTPS를 통해 객체를 삭제, 다운로드 및 설명할 수 있는 교차 계정 액세스 권한을 허용합니다.

- 첫째 구문은 CrossAccountRolePostOverHttps입니다. 이 구문은 모든 객체에 대해 PutObject 작업에 대한 액세스를 허용하고, 지정한 계정이 <역할 이름>에 지정된 역할을 가지고 있을 경우 해당 계정의 사용자에 대해 이 액세스를 허용합니다. 이러한 액세스는 작업에 대해 HTTPS를 요구하는 조건을 갖도록 지정됩니다. PutObject에 대한 액세스를 제공할 때 이 조건이 항상 포함되어야 합니다.

  즉, 교차 계정 액세스 권한을 가진 보안 주체는 PutObject에 액세스할 수 있지만 HTTPS를 통해야만 합니다.
- 두 번째 구문은 CrossAccountFullAccessExceptPost입니다. 이 구문은 객체에 대해 PutObject를 제외한 모든 작업에 대한 액세스를 허용합니다. <역할 이름>에 지정된 역할을 해당 계정이 가지고 있을 경우 해당 계정의 사용자에 대해 이 액세스를 허용합니다. 이러한 액세스는 작업에 대해 HTTPS를 요구하는 조건을 갖지 않습니다.

  즉, 교차 계정 액세스 권한을 가진 계정은 DeleteObject, GetObject 등(PutObject 제외)에 액세스할 수 있고, HTTP 또는 HTTPS를 통해 이 작업을 수행할 수 있습니다.

  두 번째 구문에서 PutObject를 제외시키지 않으면 구문이 유효하지 않게 됩니다. PutObject를 포함시킬 경우 조건으로 HTTPS를 명시적으로 설정해야 하기 때문입니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountRolePostOverHttps",
            "Effect": "Allow",
```

```
            "Action": "mediastore:PutObject",
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:role/<role name>"
            },
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        },
        {
            "Sid": "CrossAccountFullAccessExceptPost",
            "Effect": "Allow",
            "NotAction": "mediastore:PutObject",
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:role/<role name>"
            },
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*"
        }
    ]
}
```

컨테이너 정책 예제: 특정 IP 주소로 제한된 액세스

이 예제 정책은 지정된 컨테이너의 객체에 대한 모든 AWS Elemental MediaStore 작업에 액세스하도록 허용합니다. 하지만 조건에 지정된 IP 주소 범위에서만 요청을 허용해야 합니다.

이 문의 조건은 허용되는 IPv4(인터넷 프로토콜 버전 4) IP 주소인 198.51.100.* 범위를 식별합니다(한 가지 예외: 198.51.100.188).

`Condition` 블록은 `IpAddress` 및 `NotIpAddress` 조건과 AWS 전체 범위 조건 키인 `aws:SourceIp` 조건 키를 사용합니다. `aws:sourceIp` IPv4 값은 표준 CIDR 표기법을 사용합니다. 자세한 내용은 IAM 사용 설명서의 IP 주소 조건 연산자를 참조하세요.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AccessBySpecificIPAddress",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": "*",
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "198.51.100.0/24"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": "198.51.100.188/32"
                }
            }
        }
    ]
}
```

AWS Elemental MediaStore의 교차 오리진 리소스 공유(CORS) 정책

교차 오리진 리소스 공유(CORS) 한 도메인에서 로드되어 다른 도메인에 있는 리소스와 상호 작용하는 클라이언트 웹 애플리케이션에 대한 방법을 정의합니다. AWS Elemental MediaStore의 CORS 지원을 통해 MediaStore로 다양한 기능의 클라이언트 측 웹 애플리케이션을 구축하고, MediaStore 리소스에 대한 cross-origin 액세스를 선택적으로 허용할 수 있습니다.

> **Note**
>
> Amazon CloudFront를 사용하여 CORS 정책을 포함하는 컨테이너에서 콘텐츠를 배포하는 경우 반드시 AWS Elemental MediaStore에 대해 배포를 구성해야 합니다(CORS 설정을 위한 캐시 동작을 편집하는 단계 포함).

이 섹션에서는 CORS 개요를 다룹니다. 하위 주제로 AWS Elemental MediaStore 콘솔을 사용하여 CORS를 활성화하거나, 프로그래밍 방식으로 MediaStor REST API 및 AWS SDK를 사용하는 방법을 설명합니다.

주제

CORS 사용 사례 시나리오

다음은 CORS 사용에 대한 예제 시나리오입니다.

- 시나리오 1: LiveVideo라는 AWS Elemental MediaStore 컨테이너에서 라이브 스트리밍 비디오를 배포한다고 가정해 보겠습니다. 사용자가 `http://livevideo.mediastore.ap-southeast-2.amazonaws.com`과 같은 특정 오리진에서 비디오 매니페스트 엔드포인트 `www.example.com`을 로드합니다. JavaScript 비디오 플레이어를 사용하여 이 컨테이너에서 제공하는 비디오를, 인증되지 않은 GET 요청과 PUT 요청을 통해 액세스하려고 합니다. 일반적으로 브라우저에서 이러한 요청을 허용하지 않도록 JavaScript를 차단하지만, 컨테이너에 CORS 정책을 설정하여 `www.example.com`으로부터의 이 요청을 명시적으로 허용할 수 있습니다.
- 시나리오 2: 시나리오 1과 같은 라이브 스트림을 MediaStore 컨테이너에서 호스팅하려 하며, 오리진으로부터의 요청을 허용하려 한다고 가정해 보겠습니다. 와일드카드(*) 오리진을 허용하도록 CORS 정책을 구성하여 모든 오리진으로부터의 요청이 비디오를 액세스하게 할 수 있습니다.

컨테이너에 CORS 정책 추가

이 섹션에서는 AWS Elemental MediaStore 컨테이너에 교차 오리진 리소스 공유(CORS) 구성을 추가하는 방법을 설명합니다. CORS는 한 도메인에서 로드되어 다른 도메인에 있는 리소스와 상호 작용하는 클라이언트 웹 애플리케이션을 허용합니다.

cross-origin 요청을 허용하도록 컨테이너를 구성하려면 컨테이너에 CORS 정책을 추가합니다. CORS 정책은 컨테이너에 대한 액세스를 허용할 오리진과 각 오리진에 대해 지원되는 작업(HTTP 메서드)을 식별하는 규칙과 기타 작업별 정보를 정의합니다.

컨테이너에 CORS 정책을 추가하면 컨테이너 정책(컨테이너에 대한 액세스 권한을 관리하는 정책)이 계속 적용됩니다.

CORS 정책을 추가하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, CORS 정책을 만들려는 컨테이너의 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다.
3. 컨테이너 CORS 정책 섹션에서 CORS 정책 생성을 선택합니다.
4. JSON 형식으로 정책을 삽입한 후 저장을 선택합니다.

CORS 정책을 추가하려면(AWS CLI)

1. 다음과 같이 CORS 정책을 정의하는 파일을 만듭니다.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

2. 에서 `put-cors-policy` 명령을 AWS CLI사용합니다.

```
aws mediastore put-cors-policy --container-name ExampleContainer --cors-policy
 file://corsPolicy.json --region us-west-2
```

이 명령은 반환 값이 없습니다.

CORS 정책 보기

교차 오리진 리소스 공유(CORS) 한 도메인에서 로드되어 다른 도메인에 있는 리소스와 상호 작용하는 클라이언트 웹 애플리케이션에 대한 방법을 정의합니다.

CORS 정책을 보려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, CORS 정책을 보려는 컨테이너의 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타나고 컨테이너 CORS 정책 섹션에 CORS 정책이 표시됩니다.

CORS 정책을 보려면(AWS CLI)

- 에서 `get-cors-policy` 명령을 AWS CLI사용합니다.

```
aws mediastore get-cors-policy --container-name ExampleContainer --region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "CorsPolicy": [
        {
            "AllowedMethods": [
                "GET",
                "HEAD"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedOrigins": [
                "*"
            ],
            "AllowedHeaders": [
                "*"
            ]
        }
    ]
}
```

CORS 정책 편집

교차 오리진 리소스 공유(CORS) 한 도메인에서 로드되어 다른 도메인에 있는 리소스와 상호 작용하는 클라이언트 웹 애플리케이션에 대한 방법을 정의합니다.

CORS 정책을 편집하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, CORS 정책을 편집하려는 컨테이너의 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다.
3. 컨테이너 CORS 정책 섹션에서 CORS 정책 편집을 선택합니다.
4. 정책을 변경하고 저장을 선택합니다.

CORS 정책을 편집하려면(AWS CLI)

1. 다음과 같이 업데이트된 CORS 정책을 정의하는 파일을 만듭니다.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://www.example.com"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

2. 에서 `put-cors-policy` 명령을 AWS CLI사용합니다.

```
aws mediastore put-cors-policy --container-name ExampleContainer --cors-policy
 file://corsPolicy2.json --region us-west-2
```

   이 명령은 반환 값이 없습니다.

CORS 정책 삭제

교차 오리진 리소스 공유(CORS) 한 도메인에서 로드되어 다른 도메인에 있는 리소스와 상호 작용하는 클라이언트 웹 애플리케이션에 대한 방법을 정의합니다. 컨테이너에서 CORS 정책을 삭제하면 cross-origin 요청에 대한 권한이 제거됩니다.

CORS 정책을 삭제하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, CORS 정책을 삭제하려는 컨테이너의 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다.
3. 컨테이너 CORS 정책 섹션에서 CORS 정책 삭제를 선택합니다.
4. 계속을 선택하여 확인한 다음 저장을 선택하세요.

CORS 정책을 삭제하려면(AWS CLI)

- 에서 `delete-cors-policy` 명령을 AWS CLI사용합니다.

  ```
  aws mediastore delete-cors-policy --container-name ExampleContainer --region us-west-2
  ```

  이 명령은 반환 값이 없습니다.

CORS 문제 해결

CORS 정책이 있는 컨테이너에 액세스할 때 예기치 않은 동작이 발생할 경우 다음 단계에 따라 문제를 해결하세요.

1. 버킷에 CORS 정책이 연결되어 있는지 확인합니다.

   지침은 the section called “CORS 정책 보기” 단원을 참조하십시오.
2. 원하는 도구(예: 브라우저의 개발자 콘솔)를 사용하여 완료 요청 및 응답을 캡처합니다. 컨테이너에 연결된 CORS 정책에 해당 요청의 데이터와 일치하는 CORS 규칙이 한 개 이상 포함되어 있는지를 다음과 같이 확인합니다.

   a. 요청에 `Origin` 헤더가 있는지 확인합니다.

헤더가 없으면 AWS Elemental MediaStore는 요청을 cross-origin 요청으로 처리하지 않고, 응답에 CORS 응답 헤더를 돌려 보내지 않습니다.

b. 요청의 `Origin` 헤더가 해당 `AllowedOrigins`의 `CORSRule` 요소 중 최소 하나와 일치하는지 확인합니다.

   `Origin` 요청 헤더의 체계, 호스트, 포트 값이 `AllowedOrigins`의 `CORSRule`와 일치해야 합니다. 예를 들어 `CORSRule`을 설정하여 `http://www.example.com` 오리진을 허용한 경우, 요청의 `https://www.example.com` 오리진과 `http://www.example.com:80` 오리진은 해당 구성의 허용되는 오리진과 일치하지 않습니다.

c. 요청의 메서드(preflight 요청의 경우 `Access-Control-Request-Method`에 지정된 메서드)가 동일한 `AllowedMethods`의 `CORSRule` 요소의 메서드 중 하나인지 확인합니다.

d. preflight 요청의 경우 요청에 `Access-Control-Request-Headers` 헤더가 있을 경우, `CORSRule` 헤더에 각 값에 대한 `AllowedHeaders` 항목이 `Access-Control-Request-Headers`에 포함되었는지 확인합니다.

CORS 정책 예제

다음 예제는 CORS(Cross-Origin Resource Sharing) 정책을 보여 줍니다.

주제

- CORS 정책 예제: 모든 도메인에 대한 읽기 액세스 권한
- CORS 정책 예제: 특정 도메인에 대한 읽기 액세스 권한

CORS 정책 예제: 모든 도메인에 대한 읽기 액세스 권한

다음 정책은 모든 도메인의 웹 페이지에서 AWS Elemental MediaStore 컨테이너의 콘텐츠를 가져올 수 있도록 허용합니다. 이 요청은 발신 도메인의 모든 HTTP 헤더를 포함하고, 서비스는 발신 도메인의 HTTP GET 요청과 HTTP HEAD 요청에만 응답합니다. 요청은 3,000초 동안 캐싱되며, 그 후에는 새로운 결과 집합이 전송됩니다.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
```

```
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

CORS 정책 예제: 특정 도메인에 대한 읽기 액세스 권한

다음 정책은 `https://www.example.com`의 웹 페이지에서 AWS Elemental MediaStore 컨테이너의 콘텐츠를 가져올 수 있도록 허용합니다. 이 요청은 `https://www.example.com`의 모든 HTTP 헤더를 포함하고, 서비스는 `https://www.example.com`의 HTTP GET 요청과 HTTP HEAD 요청에만 응답합니다. 요청은 3,000초 동안 캐싱되며, 그 후에는 새로운 결과 집합이 전송됩니다.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://www.example.com"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

AWS Elemental MediaStore의 객체 수명 주기 정책

각 컨테이너에 대해 해당 컨테이너에 저장된 객체가 유지되는 기간을 관리하는 객체 수명 주기 정책을 만들 수 있습니다. 객체가 지정된 최대 수명에 도달하면 AWS Elemental MediaStore가 해당 객체를 삭제합니다. 스토리지 비용을 절약하기 위해 더 이상 필요하지 않은 객체를 삭제할 수 있습니다.

특정 기간이 지나면 MediaStore는 객체를 IA(액세스 빈도 낮음) 스토리지 클래스로 이동하도록 지정할 수도 있습니다. IA 스토리지 클래스에 저장된 객체는 저장 및 검색 속도가 표준 스토리지 클래스에 저장된 객체와 다릅니다. 자세한 내용은 MediaStore 요금을 참조하세요.

객체 수명 주기 정책은 하위 폴더별로 객체의 수명을 지정하는 규칙을 포함합니다. (개별 객체에 객체 수명 주기 정책을 할당할 수는 없습니다.) 각 컨테이너에 하나의 객체 수명 주기 정책만 연결할 수 있지만, 각 객체 수명 주기 정책에 최대 10개의 규칙을 추가할 수 있습니다. 자세한 내용은 객체 수명 주기 정책의 구성 요소 단원을 참조하십시오.

주제

객체 수명 주기 정책의 구성 요소

객체 수명 주기 정책은 AWS Elemental MediaStore 컨테이너에서 객체가 유지되는 기간을 관리합니다. 각 객체 수명 주기 정책은 객체의 수명을 지정하는 하나 이상의 규칙으로 구성됩니다. 각 규칙은 한 폴더, 여러 폴더 또는 전체 컨테이너에 적용될 수 있습니다.

각 객체 수명 주기 정책을 한 컨테이너에 연결할 수 있고 각 객체 수명 주기 정책은 최대 10개의 규칙을 포함할 수 있습니다. 개별 객체에 객체 수명 주기 정책을 할당할 수는 없습니다.

객체 수명 주기 정책의 규칙

세 가지 유형의 규칙을 만들 수 있습니다.

- 임시 데이터
- 객체 삭제
- 수명 주기 전환

임시 데이터

임시 데이터 규칙은 객체가 수 초 내에 만료되도록 설정합니다. 이 유형의 규칙은 정책 실행 후 컨테이너에 추가된 객체에만 적용됩니다. MediaStore가 새 정책을 컨테이너에 적용하는 데 최대 20분이 걸립니다.

일시적인 데이터 규칙의 예는 다음과 같습니다:

```
        {
            "definition": {
                "path": [ {"wildcard": "Football/index*.m3u8"} ],
                "seconds_since_create": [
                    {"numeric": [">", 120]}
                ]
            },
            "action": "EXPIRE"
        },
```

임시 데이터 규칙에는 다음 세 부분이 있습니다.

- `path`: 항상 `wildcard`로 설정 이 부분을 사용하여 삭제할 객체를 정의합니다. 별표(*)로 표시되는 하나 이상의 와일드카드를 사용할 수 있습니다. 각 와일드카드는 0개 이상의 문자 조합을 나타냅니다. 예를 들어, `"path": [ {"wildcard": "Football/index*.m3u8"} ],`은 `index*.m3u8`의 패턴(예: index.m3u8, index1.m3us8, index123456.m3u8)과 일치하는 `Football` 폴더의 모든 파일에 적용됩니다. 단일 규칙에 최대 10개의 경로를 포함할 수 있습니다.
- `seconds_since_create`: 항상 `numeric`로 설정 1초~300초의 값을 지정할 수 있습니다. 연산자를 초과(>) 또는 이상(>=)으로 설정할 수도 있습니다.
- `action`: 항상 `EXPIRE`로 설정

임시 데이터 규칙(객체가 몇 초 내에 만료됨)의 경우 객체 만료와 객체 삭제 사이에 지연이 없습니다.

> **Note**
>
> 임시 규칙이 적용되는 객체는 `list-items` 응답에 포함되지 않습니다. 또한 임시 데이터 규칙으로 인해 만료되는 객체는 만료 시 CloudWatch 이벤트를 생성하지 않습니다.

객체 삭제

객체 삭제 규칙은 객체가 며칠 내에 만료되도록 설정합니다. 이러한 유형의 규칙은 정책이 생성되기 전에 컨테이너에 추가된 경우에도 컨테이너의 모든 객체에 적용됩니다. MediaStore가 새 정책을 적용하는 데 최대 20분이 걸리되지만 컨테이너에서 객체를 지우려면 최대 24시간이 걸릴 수 있습니다.

두 가지 객체 삭제 규칙의 예는 다음과 같습니다.

```
        {
            "definition": {
                "path": [ { "prefix": "FolderName/" }  ],
                "days_since_create": [
                    {"numeric": [">" , 5]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [ { "wildcard": "Football/*.ts" }  ],
                "days_since_create": [
                    {"numeric": [">" , 5]}
                ]
            },
            "action": "EXPIRE"
        }
```

객체 삭제 규칙에는 다음 세 부분이 있습니다.

- path: prefix 또는 wildcard로 설정합니다. 같은 규칙에서 prefix와 wildcard를 함께 사용할 수 없습니다. 두 가지 모두를 사용하려면 위의 예와 같이 prefix에 대한 규칙과 wildcard에 대한 규칙을 별도로 생성해야 합니다.
  - prefix - 특정 폴더 내의 모든 객체를 삭제하려면 경로를 prefix로 설정합니다. 매개 변수가 비어있는 경우("path": [{ "prefix": "" }],) 대상은 현재 컨테이너 내의 아무 곳에나 저장된 모든 객체입니다. 단일 규칙에 최대 10개의 prefix 경로를 포함할 수 있습니다.
  - wildcard - 파일 이름 및/또는 파일 형식에 따라 특정 객체를 삭제하려면 경로를 wildcard로 설정합니다. 별표(*)로 표시되는 하나 이상의 와일드카드를 사용할 수 있습니다. 각 와일드카드는 0개 이상의 문자 조합을 나타냅니다. 예를 들어, "path": [{"wildcard": "Football/*.ts"}],는 *.ts의 패턴(예: filename.ts, filename1.ts, filename123456.ts)과 일치하는

Football 폴더의 모든 파일에 적용됩니다. 단일 규칙에 최대 10개의 wildcard 경로를 포함할 수 있습니다.

- days_since_create: 항상 numeric로 설정 1일~36,500일의 값을 지정할 수 있습니다. 연산자를 초과(>) 또는 이상(>=)으로 설정할 수도 있습니다.
- action: 항상 EXPIRE로 설정

객체 삭제 규칙(객체가 며칠 내에 만료됨)의 경우 객체 만료와 객체 삭제 사이에 약간의 지연이 있을 수 있습니다. 그러나 객체가 만료되자마자 결제에 변경 사항이 발생합니다. 예를 들어 수명 주기 규칙에서 days_since_create를 10으로 지정하는 경우 계정은 객체가 아직 삭제되지 않았어도 객체 생성 후 10일이 지나면 객체에 대한 요금이 부과되지 않습니다.

수명 주기 전환

수명 주기 전환 규칙은 며칠 단위로 측정하여 일정 기간이 지난 객체를 IA(액세스 빈도 낮음) 스토리지 클래스로 이동하도록 설정합니다. IA 스토리지 클래스에 저장된 객체는 저장 및 검색 속도가 표준 스토리지 클래스에 저장된 객체와 다릅니다. 자세한 내용은 MediaStore 요금을 참조하세요.

객체를 IA 스토리지 클래스로 이동한 후에는 표준 스토리지 클래스로 다시 이동할 수 없습니다.

수명 주기 전환 규칙은 정책이 생성되기 전에 컨테이너에 추가되었더라도 컨테이너의 모든 객체에 적용됩니다. MediaStore가 새 정책을 적용하는 데 최대 20분이 걸리지만 컨테이너에서 객체를 지우려면 최대 24시간이 걸릴 수 있습니다.

수명 주기 전환 규칙의 예는 다음과 같습니다.

```
{
    "definition": {
        "path": [
            {"prefix": "AwardsShow/"}
        ],
        "days_since_create": [
            {"numeric": [">=" , 30]}
        ]
    },
    "action": "ARCHIVE"
}
```

수명 주기 전환 규칙은 다음 세 부분으로 이루어집니다.

- path: prefix 또는 wildcard로 설정합니다. 같은 규칙에서 prefix와 wildcard를 함께 사용할 수 없습니다. 둘 다 사용하려면 prefix의 규칙 하나와 wildcard의 규칙 하나를 별도로 만들어야 합니다.
  - prefix - 특정 폴더 내의 모든 객체를 IA 스토리지 클래스로 전환하려면 경로를 prefix로 설정합니다. 이 파라미터가 비어 있는 경우("path": [{ "prefix": "" }],), 위치와 관계없이 현재 컨테이너에 저장되어 있는 모든 객체가 대상이 됩니다. 단일 규칙에 최대 10개의 prefix 경로를 포함할 수 있습니다.
  - wildcard - 파일 이름 및/또는 파일 형식에 따라 특정 객체를 IA 스토리지 클래스로 전환하려면 경로를 wildcard로 설정합니다. 별표(*)로 표시되는 하나 이상의 와일드카드를 사용할 수 있습니다. 각 와일드카드는 0개 이상의 문자 조합을 나타냅니다. 예를 들어, "path": [{"wildcard": "Football/*.ts"}],는 *.ts의 패턴(예: filename.ts, filename1.ts, filename123456.ts)과 일치하는 Football 폴더의 모든 파일에 적용됩니다. 단일 규칙에 최대 10개의 wildcard 경로를 포함할 수 있습니다.
- days_since_create: 항상 "numeric": [">=" , 30]로 설정
- action: 항상 ARCHIVE로 설정

예제

예를 들어 컨테이너 LiveEvents에 4개의 하위 폴더 Football, Baseball, Basketball, AwardsShow가 있습니다. LiveEvents 폴더에 할당된 객체 수명 주기 정책은 다음과 같을 수 있습니다.

```
{
    "rules": [
         {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                ],
                "days_since_create": [
                    {"numeric": [">" , 28]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
```

```
                "path": [ { "prefix": "AwardsShow/" }  ],
                "days_since_create": [
                    {"numeric": [">=" , 15]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [ { "prefix": "" }  ],
                "days_since_create": [
                    {"numeric": [">" , 40]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [ { "wildcard": "Football/*.ts" }  ],
                "days_since_create": [
                    {"numeric": [">" , 20]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "Football/index*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [">" , 15]}
                ]
            },
            "action": "EXPIRE"
        },
        {
                "definition": {
                    "path": [
                        {"prefix": "Program/"}
                    ],
                    "days_since_create": [
                        {"numeric": [">=" , 30]}
                    ]
```

```
                },
                "action": "ARCHIVE"
            }
    ]
}
```

위 정책은 다음을 지정합니다.

- 첫 번째 규칙은 `LiveEvents/Football` 폴더 및 `LiveEvents/Baseball` 폴더에 저장된 객체가 28일을 경과하면 AWS Elemental MediaStore가 해당 객체를 삭제하도록 지시합니다.
- 두 번째 규칙은 `LiveEvents/AwardsShow` 폴더에 저장된 객체가 15일을 경과하면 서비스가 해당 객체를 삭제하도록 지시합니다.
- 세 번째 규칙은 `LiveEvents` 컨테이너에 저장된 객체가 40일을 경과하면 서비스가 해당 객체를 삭제하도록 지시합니다. 이 규칙은 `LiveEvents` 컨테이너에 직접 저장된 객체뿐 아니라 컨테이너의 4개 하위 폴더에 저장된 객체에도 적용됩니다.
- 네 번째 규칙은 `*.ts` 패턴과 일치하는 `Football` 폴더의 객체가 20일을 경과하면 서비스가 해당 객체를 삭제하도록 지시합니다.
- 다섯 번째 규칙은 `Football` 패턴과 일치하는 `index*.m3u8` 폴더의 객체가 15초를 경과하면 서비스가 해당 객체를 삭제하도록 지시합니다. MediaStore는 이러한 파일을 컨테이너에 배치한 후 16초 후에 삭제합니다.
- 여섯 번째 규칙은 30일이 지난 `Program` 폴더의 객체를 IA 스토리지 클래스로 이동하도록 서비스에 지시합니다.

객체 수명 주기 정책의 자세한 예는 객체 수명 주기 정책의 예 단원을 참조하십시오.

컨테이너에 객체 수명 주기 정책 추가

객체 수명 주기 정책을 사용하면 객체가 컨테이너에 저장되는 기간을 저장할 수 있습니다. 만료 날짜를 설정하면 만료 날짜 후 AWS Elemental MediaStore가 해당 객체를 삭제합니다. 서비스가 새 정책을 컨테이너에 적용하는 데 최대 20분 걸립니다.

수명 주기 정책을 구성하는 방법에 대한 자세한 내용은 객체 수명 주기 정책의 구성 요소 단원을 참조하십시오.

> **Note**
>
> 객체 삭제 규칙(객체가 며칠 내에 만료됨)의 경우 객체 만료와 객체 삭제 사이에 약간의 지연이 있을 수 있습니다. 그러나 객체가 만료되자마자 결제에 변경 사항이 발생합니다. 예를 들어 수

명 주기 규칙에서 `days_since_create`를 10으로 지정하는 경우 계정은 객체가 아직 삭제되지 않았어도 객체 생성 후 10일이 지나면 객체에 대한 요금이 부과되지 않습니다.

객체 수명 주기 정책을 추가하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, 객체 수명 주기 정책을 만들려는 컨테이너의 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다.
3. 객체 수명 주기 정책 섹션에서 객체 수명 주기 정책 생성을 선택합니다.
4. JSON 형식으로 정책을 삽입한 후 저장을 선택합니다.

객체 수명 주기 정책 추가(AWS CLI)

1. 객체 수명 주기 정책을 정의하는 파일을 만듭니다.

```
{
    "rules": [
         {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                ],
                "days_since_create": [
                    {"numeric": [">" , 28]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "AwardsShow/index*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [">" , 8]}
                ]
            },
            "action": "EXPIRE"
```

```
        }
    ]
}
```

2. 에서 put-lifecycle-policy 명령을 AWS CLI사용합니다.

```
aws mediastore put-lifecycle-policy --container-name LiveEvents --lifecycle-
policy file://LiveEventsLifecyclePolicy.json --region us-west-2
```

이 명령은 반환 값이 없습니다. 서비스가 지정된 정책을 컨테이너에 연결합니다.

객체 수명 주기 정책 보기

객체 수명 주기 정책은 객체를 컨테이너에 유지할 기간을 지정합니다.

객체 수명 주기 정책 조회(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, 객체 수명 주기 정책을 보려는 컨테이너의 이름을 선택합니다.

   객체 수명 주기 정책 섹션에 객체 수명 주기 정책과 함께 컨테이너 세부 정보 페이지가 나타납니다.

객체 수명 주기 정책을 보려면(AWS CLI)

- 에서 get-lifecycle-policy 명령을 AWS CLI사용합니다.

```
aws mediastore get-lifecycle-policy --container-name LiveEvents --region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "LifecyclePolicy": "{
        "rules": [
            {
                "definition": {
                    "path": [
                        {"prefix": "Football/"},
                        {"prefix": "Baseball/"}
                    ],
```

```
                    "days_since_create": [
                        {"numeric": [">" , 28]}
                    ]
                },
             "action": "EXPIRE"
            }
        ]
    }"
}
```

객체 수명 주기 정책 편집

기존 객체 수명 주기 정책을 편집할 수는 없습니다. 하지만 대체 정책을 업로드하면 기존 정책을 변경할 수 있습니다. 서비스가 업데이트된 정책을 컨테이너에 적용하는 데 최대 20분 걸립니다.

객체 수명 주기 정책 편집(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, 객체 수명 주기 정책을 편집하려는 컨테이너의 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다.
3. 객체 수명 주기 정책 섹션에서 객체 수명 주기 정책 편집을 선택합니다.
4. 정책을 변경하고 저장을 선택합니다.

객체 수명 주기 정책을 편집하려면(AWS CLI)

1. 업데이트된 객체 수명 주기 정책을 정의하는 파일을 만듭니다.

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                    {"prefix": "Basketball/"}
                ],
                "days_since_create": [
                    {"numeric": [">" , 28]}
                ]
```

```
            },
            "action": "EXPIRE"
        }
    ]
}
```

2. 에서 `put-lifecycle-policy` 명령을 AWS CLI사용합니다.

```
aws mediastore put-lifecycle-policy --container-name LiveEvents --lifecycle-
policy file://LiveEvents2LifecyclePolicy --region us-west-2
```

이 명령은 반환 값이 없습니다. 서비스가 지정된 정책을 이전 정책 대신 컨테이너에 연결합니다.

객체 수명 주기 정책 삭제

객체 수명 주기 정책을 삭제할 경우 서비스가 변경 사항을 컨테이너에 적용하려면 최대 20분이 걸립니다.

객체 수명 주기 정책을 삭제하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, 객체 수명 주기 정책을 삭제하려는 컨테이너의 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다.
3. 객체 수명 주기 정책 섹션에서 객체 수명 주기 정책 삭제를 선택합니다.
4. 계속을 선택하여 확인한 다음 저장을 선택하세요.

객체 수명 주기 정책을 삭제하려면(AWS CLI)

- 에서 `delete-lifecycle-policy` 명령을 AWS CLI사용합니다.

```
aws mediastore delete-lifecycle-policy --container-name LiveEvents --region us-
west-2
```

이 명령은 반환 값이 없습니다.

객체 수명 주기 정책의 예

다음은 객체 수명 주기 정책의 예입니다.

주제

객체 수명 주기 정책의 예: 몇 초 안에 만료

다음 정책은 다음 조건을 모두 충족하는 객체를 삭제하도록 MediaStore에 지시합니다.

- 정책이 적용되면 객체가 컨테이너에 추가됩니다.
- 객체가 `Football` 폴더에 저장됩니다.
- 객체의 파일 확장자는 `m3u8`입니다.
- 객체가 컨테이너에 20초 이상 있었습니다.

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "Football/*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [ ">", 20 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

객체 수명 주기 정책의 예: 며칠 안에 만료

다음 정책은 다음 조건을 모두 충족하는 객체를 삭제하도록 MediaStore에 지시합니다.

- 객체가 `Program` 폴더에 저장됩니다.
- 이 객체의 파일 확장자는 `ts`입니다.
- 객체가 컨테이너에 5일 이상 있었습니다.

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "Program/*.ts"}
                ],
                "days_since_create": [
                    {"numeric": [ ">", 5 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

객체 수명 주기 정책의 예: 액세스 빈도가 낮은 스토리지 클래스로 전환

다음 정책은 30일이 지난 객체를 IA(액세스 빈도 낮음) 스토리지 클래스로 이동하도록 MediaStore에 지시합니다. IA 스토리지 클래스에 저장된 객체는 저장 및 검색 속도가 표준 스토리지 클래스에 저장된 객체와 다릅니다.

`days_since_create` 필드를 `"numeric": [">=" ,30]`으로 설정해야 합니다.

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                ],
```

```
                "days_since_create": [
                    {"numeric": [">=" , 30]}
                ]
            },
            "action": "ARCHIVE"
        }
    ]
}
```

객체 수명 주기 정책의 예: 복수의 규칙

다음 정책은 다음을 수행하도록 MediaStore에 지시합니다.

- AwardsShow 폴더에 저장된 객체를 30일 후 IA(액세스 빈도 낮음) 스토리지 클래스로 이동
- 파일 확장자가 m3u8이고 Football 폴더에 저장된 객체를 20초 후 삭제
- April 폴더에 저장된 객체를 10일 후 삭제
- 파일 확장자가 ts이고 Program 폴더에 저장된 객체를 5일 후 삭제

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"prefix": "AwardsShow/"}
                ],
                "days_since_create": [
                    {"numeric": [ ">=" , 30 ]}
                ]
            },
            "action": "ARCHIVE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "Football/*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [ ">", 20 ]}
                ]
            },
            "action": "EXPIRE"
```

```
        },
        {
            "definition": {
                "path": [
                    {"prefix": "April"}
                ],
                "days_since_create": [
                    {"numeric": [ ">", 10 ]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "Program/*.ts"}
                ],
                "days_since_create": [
                    {"numeric": [ ">", 5 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

객체 수명 주기 정책의 예: 빈 컨테이너

다음 객체 수명 주기 정책은 컨테이너에 객체가 추가되고 1일 후 폴더 및 하위 폴더를 포함하여 컨테이너에 있는 모든 객체를 삭제하도록 MediaStore에 지시합니다. 이 정책이 적용되기 전에 컨테이너에 객체가 있었다면 MediaStore는 정책이 적용되고 1일 후에 해당 개체를 삭제합니다. 서비스가 새 정책을 컨테이너에 적용하는 데 최대 20분 걸립니다.

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "*"}
                ],
                "days_since_create": [
                    {"numeric": [ ">=", 1 ]}
                ]
```

```
            },
            "action": "EXPIRE"
        }
    ]
}
```

AWS Elemental MediaStore의 지표 정책

각 컨테이너의 경우 지표 정책을 추가하여 AWS Elemental MediaStore가 Amazon CloudWatch에 지표를 전송할 수 있습니다. 새 정책이 적용되려면 최대 20분이 걸립니다. 각 MediaStore 지표에 대한 설명은 MediaStore 지표 섹션을 참조하세요.

지표 정책에는 다음이 포함됩니다.

- 컨테이너 수준에서 지표를 활성화 또는 비활성화하는 설정입니다.
- 객체 수준에서 지표를 활성화하는 0에서 5개의 규칙 정책에 규칙이 포함되어 있는 경우 각 규칙에 다음 사항이 모두 포함되어야 합니다.
  - 그룹에 포함할 객체를 정의하는 객체 그룹입니다. 정의는 경로 또는 파일 이름일 수 있지만 900자를 초과할 수 없습니다. 유효한 문자는 a-z, A-Z, 0-9, _(밑줄), =(같음), :(콜론), .(마침표), -(하이픈), ~(물결표), /(슬래시) 및 *(별표)입니다. 와일드카드(*)를 사용할 수 있습니다.
  - 객체 그룹을 참조할 수 있는 객체 그룹 이름입니다. 이름은 30자를 초과할 수 없습니다. 유효한 문자는 a-z, A-Z, 0-9 및 –(밑줄)입니다.

객체가 여러 규칙과 일치하는 경우 CloudWatch는 일치하는 각 규칙에 대한 데이터 포인트를 표시합니다. 예를 들어 객체가 `rule1` 및 `rule2`라는 두 개의 규칙과 일치하는 경우 CloudWatch는 이러한 규칙에 대한 두 개의 데이터 포인트를 표시합니다. 첫 번째는 차원이 `ObjectGroupName=rule1`이고 두 번째는 차원이 `ObjectGroupName=rule2`입니다.

주제

- 지표 정책 추가
- 지표 정책 보기
- 지표 정책 편집
- 지표 정책 예제

지표 정책 추가

지표 정책에는 AWS Elemental MediaStore가 Amazon CloudWatch로 전송하는 지표를 지정하는 규칙이 포함되어 있습니다. 지표 정책의 예제는 지표 정책 예제 섹션을 참조하세요.

지표 정책을 추가하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 지표 정책을 추가할 컨테이너의 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다.
3. 지표 정책 섹션에서 지표 정책 생성을 선택합니다.
4. JSON 형식으로 정책을 삽입한 후 저장을 선택합니다.

지표 정책 보기

콘솔 또는를 사용하여 컨테이너의 지표 정책을 AWS CLI 볼 수 있습니다.

지표 정책을 보려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 컨테이너 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다. 정책이 지표 정책 섹션에 표시됩니다.

지표 정책 편집

지표 정책에는 AWS Elemental MediaStore가 Amazon CloudWatch로 전송하는 지표를 지정하는 규칙이 포함되어 있습니다. 기존 지표 정책을 편집할 때 새 정책이 적용되기까지 최대 20분이 걸립니다. 지표 정책의 예제는 지표 정책 예제 섹션을 참조하세요.

지표 정책을 편집하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 컨테이너 이름을 선택합니다.
3. 지표 정책 섹션에서 지표 정책 편집을 선택합니다.
4. 정책을 적절히 변경하고 저장을 선택합니다.

지표 정책 예제

다음 예제는 여러 사용 사례에 대해 구성된 지표 정책을 보여줍니다.

주제

지표 정책 예제: 컨테이너 수준 지표

이 정책 예제는 AWS Elemental MediaStore가 컨테이너 수준에서 지표를 Amazon CloudWatch에 전송해야 함을 나타냅니다. 예를 들어 컨테이너에 대한 `Put` 요청 수를 계산하는 `RequestCount` 지표가 포함됩니다. 또는 이를 `DISABLED`로 설정할 수 있습니다.

이 정책에는 규칙이 없으므로 MediaStore에서는 경로 수준에서 지표를 전송하지 않습니다. 예를 들어 이 컨테이너 내의 특정 폴더에 대한 `Put` 요청 수를 볼 수 없습니다.

```
{
   "ContainerLevelMetrics": "ENABLED"
}
```

지표 정책 예제: 경로 수준 지표

이 정책 예제는 AWS Elemental MediaStore가 컨테이너 수준에서 지표를 Amazon CloudWatch에 전송하지 않아야 함을 나타냅니다. 또한 MediaStore에서는 `baseball/saturday` 및 `football/saturday`의 두 폴더의 객체에 대한 지표를 전송해야 합니다. MediaStore 요청에 대한 지표는 다음과 같습니다.

- `baseball/saturday` 폴더에 대한 요청은 CloudWatch 차원이 `ObjectGroupName=baseballGroup`입니다.
- `football/saturday` 폴더에 대한 요청은 차원이 `ObjectGroupName=footballGroup`입니다.

```
{
```

```
   "ContainerLevelMetrics": "DISABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "baseball/saturday",
       "ObjectGroupName": "baseballGroup"
     },
     {
       "ObjectGroup": "football/saturday",
       "ObjectGroupName": "footballGroup"
     }
   ]
}
```

지표 정책 예제: 컨테이너 수준 및 경로 수준 지표

이 정책 예제는 AWS Elemental MediaStore가 컨테이너 수준에서 지표를 Amazon CloudWatch에 전송해야 함을 나타냅니다. 또한 MediaStore에서는 `baseball/saturday` 및 `football/saturday`의 두 폴더의 객체에 대한 지표를 전송해야 합니다. MediaStore 요청에 대한 지표는 다음과 같습니다.

- `baseball/saturday` 폴더에 대한 요청은 CloudWatch 차원이 `ObjectGroupName=baseballGroup`입니다.
- `football/saturday` 폴더에 대한 요청은 CloudWatch 차원이 `ObjectGroupName=footballGroup`입니다.

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "baseball/saturday",
       "ObjectGroupName": "baseballGroup"
     },
     {
       "ObjectGroup": "football/saturday",
       "ObjectGroupName": "footballGroup"
     }
   ]
}
```

지표 정책 예제: 와일드카드를 사용하는 경로 수준 지표

이 정책 예제는 AWS Elemental MediaStore가 컨테이너 수준에서 지표를 Amazon CloudWatch에 전송해야 함을 나타냅니다. 또한 MediaStore는 해당 파일 이름을 기준으로 객체에 대한 지표를 전송해야 합니다. 와일드카드는 객체가 컨테이너의 아무 곳에나 저장될 수 있으며 확장명이 `.m3u8`로 끝나는 한 모든 파일 이름을 가질 수 있음을 나타냅니다.

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "*.m3u8",
       "ObjectGroupName": "index"
     }
   ]
 }
```

지표 정책 예제: 중복 규칙이 있는 경로 레벨 지표

이 정책 예제는 AWS Elemental MediaStore가 컨테이너 수준에서 지표를 Amazon CloudWatch에 전송해야 함을 나타냅니다. 또한 MediaStore에서는 `sports/football/saturday` 및 `sports/football`의 두 폴더에 대한 지표를 전송해야 합니다.

`sports/football/saturday` 폴더에 대한 MediaStore 요청의 지표는 CloudWatch 차원이 `ObjectGroupName=footballGroup1`입니다. `sports/football` 폴더에 저장된 객체는 두 규칙과 일치하므로, CloudWatch는 이러한 객체에 대한 두 개의 데이터 포인트를 표시합니다. 하나는 차원이 `ObjectGroupName=footballGroup1`이고 다른 하나는 차원이 `ObjectGroupName=footballGroup2`입니다.

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "sports/football/saturday",
       "ObjectGroupName": "footballGroup1"
     },
     {
       "ObjectGroup": "sports/football",
       "ObjectGroupName": "footballGroup2"
     }
   ]
```

```
}
```

AWS Elemental MediaStore의 폴더

폴더는 컨테이너를 분할합니다. 파일 시스템에서 폴더에 하위 폴더를 만들어 폴더를 나누듯이 폴더를 사용하여 컨테이너를 나눕니다. 최대 10개 수준의 폴더(해당 컨테이너 자체는 포함되지 않음)를 생성할 수 있습니다.

폴더는 선택 사항입니다. 객체를 폴더 대신 컨테이너에 직접 업로드하도록 선택할 수 있습니다. 하지만 폴더는 객체를 정리하기 위한 간편한 방법입니다.

폴더에 객체를 업로드하려면 폴더 경로를 지정합니다. 폴더가 이미 있을 경우 AWS Elemental MediaStore는 폴더에 객체를 저장합니다. 폴더가 없으면 폴더를 만든 후 그 폴더에 객체를 저장합니다.

예를 들어 `movies`라는 이름이 지정된 컨테이너가 있고 경로 `mlaw.ts`가 포함된 `premium/canada`라는 이름의 파일을 업로드한다고 가정해 보겠습니다. AWS Elemental MediaStore는 프리미엄 폴더의 캐나다 하위 폴더에 객체를 저장합니다. 폴더가 없으면 서비스는 `premium` 폴더와 `canada` 하위 폴더를 생성한 후 `canada` 하위 폴더에 객체를 저장합니다. 경로 없이 `movies` 컨테이너만 지정한 경우, 서비스는 객체는 컨테이너에 직접 저장합니다.

해당 폴더에서 마지막 객체를 삭제하면 AWS Elemental MediaStore는 폴더를 자동으로 삭제합니다. 또한 서비스는 해당 폴더 위의 빈 폴더를 삭제합니다. 예를 들어 premium이라는 폴더가 있고 이 폴더에는 파일이 없으며 `canada`라는 하위 폴더만 한 개 있다고 가정해 보겠습니다. `canada` 하위 폴더에는 `mlaw.ts`라는 파일이 한 개 있습니다. `mlaw.ts` 파일을 삭제하면 서비스는 `premium` 폴더와 `canada` 폴더를 모두 삭제합니다. 이 자동 삭제는 폴더에만 적용됩니다. 서비스는 빈 컨테이너를 삭제하지 않습니다.

주제

폴더 이름에 대한 규칙

폴더의 이름을 선택할 때 다음에 유의하세요.

- 이름에는 대문자(A~Z), 소문자(a~z), 숫자(0~9), 마침표(.), 하이픈(-), 물결표(~), 밑줄(_), 콜론(:)과 같은 문자만 포함할 수 있습니다.

- 이름은 1자 이상 있어야 합니다. 빈 폴더 이름(예:`folder1//folder3/`)은 허용되지 않습니다.
- 이름은 대/소문자를 구분합니다. 예를 들어 `myFolder`라는 폴더와, `myfolder`라는 폴더를 동일한 컨테이너나 폴더에 둘 수 있습니다. 해당 이름이 고유하기 때문입니다.
- 이름은 상위 컨테이너 또는 폴더에서만 고유하면 됩니다. 예를 들어 `myfolder`라는 폴더를 2개의 서로 다른 컨테이너인 `movies/myfolder`와 `sports/myfolder`에 만들 수 있습니다.
- 이름은 상위 컨테이너의 이름과 같을 수 있습니다.
- 폴더를 만든 후 이름을 변경할 수 없습니다.

폴더 생성

객체를 업로드할 때 폴더를 만들 수 있습니다. 폴더에 객체를 업로드하려면 폴더 경로를 지정합니다. 폴더가 이미 있을 경우 AWS Elemental MediaStore는 폴더에 객체를 저장합니다. 폴더가 없으면 폴더를 만든 후 그 폴더에 객체를 저장합니다.

자세한 내용은 the section called “객체 업로드” 단원을 참조하십시오.

폴더 삭제

폴더가 빈 경우에만 폴더를 삭제할 수 있으며 객체가 들어 있는 폴더는 삭제할 수 없습니다.

해당 폴더에서 마지막 객체를 삭제하면 AWS Elemental MediaStore는 폴더를 자동으로 삭제합니다. 또한 서비스는 해당 폴더 위의 빈 폴더를 삭제합니다. 예를 들어 `premium`이라는 폴더가 있고 이 폴더에는 파일이 없으며 `canada`라는 하위 폴더만 한 개 있다고 가정해 보겠습니다. `canada` 하위 폴더에는 `mlaw.ts`라는 파일이 한 개 있습니다. `mlaw.ts` 파일을 삭제하면 서비스는 `premium` 폴더와 `canada` 폴더를 모두 삭제합니다. 이 자동 삭제는 폴더에만 적용됩니다. 서비스는 빈 컨테이너를 삭제하지 않습니다.

자세한 내용은 객체 삭제 단원을 참조하십시오.

AWS Elemental MediaStore의 객체

AWS Elemental MediaStore 자산을 객체라고 합니다. 컨테이너 또는 컨테이너의 폴더에 객체를 업로드할 수 있습니다.

MediaStore에서 객체를 업로드 및 다운로드하고 삭제할 수 있습니다.

- 업로드 – 컨테이너나 폴더에 객체를 추가합니다. 객체를 만드는 것과는 다릅니다. 객체를 MediaStore에 업로드하려면 먼저 로컬에 객체를 만들어야 합니다.
- 다운로드 – MediaStore의 객체를 다른 위치로 복사합니다. MediaStore에서 객체가 제거되지는 않습니다.
- 삭제 – MediaStore에서 객체를 완전히 제거합니다. 객체를 개별적으로 삭제할 수도 있고 객체 수명 주기 정책을 추가하여 지정된 지속 시간 이후 자동으로 컨테이너의 객체를 삭제할 수도 있습니다.

MediaStore는 모든 파일 유형을 허용합니다.

주제

객체 업로드

객체를 컨테이너 또는 컨테이너의 폴더로 업로드할 수 있습니다. 폴더에 객체를 업로드하려면 폴더 경로를 지정합니다. 폴더가 이미 있을 경우 AWS Elemental MediaStore는 폴더에 객체를 저장합니다. 폴더가 없으면 폴더를 만든 후 그 폴더에 객체를 저장합니다. 폴더에 대한 자세한 내용은 AWS Elemental MediaStore의 폴더 섹션을 참조하세요.

MediaStore 콘솔 또는를 사용하여 객체를 업로드 AWS CLI 할 수 있습니다.

MediaStore는 객체의 청크 분할 전송을 지원합니다. 이 기능은 객체를 업로드하는 동안에도 다운로드가 가능하게 만들어 지연 시간을 단축합니다. 이 기능을 사용하려면 객체의 업로드 가용성을 `streaming`으로 설정합니다. API를 사용하여 객체를 업로드할 때 이 헤더의 값을 설정할 수 있습니다.

요청에서 이 헤더를 지정하지 않으면 MediaStore가 객체 업로드 가용성으로 기본값 `standard`을 지정합니다.

표준 업로드 가용성에서는 객체 크기가 25MB를 초과할 수 없고 스트리밍 업로드 가용성은 10MB입니다.

> **Note**
>
> 객체 파일 이름에는 문자, 숫자, 마침표(.), 밑줄(_), 물결 기호(~), 하이픈(-), 등호(=), 콜론(:)만 사용할 수 있습니다.

객체를 업로드하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 컨테이너 이름을 선택합니다. 컨테이너의 세부 정보 창이 나타납니다.
3. 객체 업로드를 선택합니다.
4. 대상 경로에 폴더 경로를 입력합니다. 예를 들어 `premium/canada`입니다. 지정한 경로에 폴더가 존재하지 않으면 해당 폴더가 자동으로 생성됩니다.
5. 객체 섹션에서 찾아보기를 선택합니다.
6. 해당 폴더로 이동한 후 업로드할 객체를 하나 선택합니다.
7. 열기를 선택한 후 업로드를 선택합니다.

> **Note**
>
> 이름이 같은 파일이 해당 폴더에 이미 있으면, 업로드된 파일이 원래 파일을 겹쳐 씁니다.

객체를 업로드하려면(AWS CLI)

- 에서 `put-object` 명령을 AWS CLI사용합니다. 다음과 같은 파라미터를 포함시킬 수 있습니다. `content-type`, `cache-control`(호출자가 객체의 캐시 동작을 제어하도록 허용), `path`(컨테이너 내에 있는 폴더의 객체 저장).

> **Note**
>
> 객체를 업로드한 후에는 content-type, cache-control 또는 path를 변경할 수 없습니다.

```
aws mediastore-data put-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --body README.md --path /
folder_name/README.md --cache-control "max-age=6, public" --content-type binary/
octet-stream --region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "ContentSHA256":
 "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",
    "StorageClass": "TEMPORAL",
    "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"
}
```

객체 목록 보기

AWS Elemental MediaStore 콘솔을 사용하여 컨테이너나 폴더의 최상위에 저장된 항목(객체 및 폴더)을 볼 수 있습니다. 현재 컨테이너나 폴더의 하위 폴더에 저장된 항목은 표시되지 않습니다. 컨테이너 내에 있는 폴더 또는 하위 폴더의 수에 관계없이 AWS CLI 를 사용하여 컨테이너 내의 객체 및 폴더 목록을 볼 수 있습니다.

특정 컨테이너에 있는 객체의 목록을 보려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, 보려는 폴더가 들어 있는 컨테이너의 이름을 선택합니다.
3. 목록에서 폴더의 이름을 선택합니다.

   세부 정보 페이지가 나타나고, 해당 폴더에 저장된 모든 폴더와 객체가 표시됩니다.

특정 폴더에 있는 객체의 목록을 보려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, 보려는 폴더가 들어 있는 컨테이너의 이름을 선택합니다.

   세부 정보 페이지가 나타나고, 해당 컨테이너에 저장된 모든 폴더와 객체가 표시됩니다.

특정 컨테이너에 있는 객체 및 폴더의 목록을 보려면(AWS CLI)

- 에서 `list-items` 명령을 AWS CLI사용합니다.

```
aws mediastore-data list-items --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "Items": [
        {
            "ContentType": "image/jpeg",
            "LastModified": 1563571859.379,
            "Name": "filename.jpg",
            "Type": "OBJECT",
            "ETag":
 "543ab21abcd1a234ab123456a1a2b12345ab12abc12a1234abc1a2bc12345a12",
            "ContentLength": 3784
        },
        {
            "Type": "FOLDER",
            "Name": "ExampleLiveDemo"
        }
    ]
}
```

> **Note**
>
> `seconds_since_create` 규칙이 적용되는 객체는 `list-items` 응답에 포함되지 않습니다.

특정 폴더에 있는 객체 및 폴더의 목록을 보려면(AWS CLI)

- 에서 요청 끝에 지정된 폴더 이름과 함께 `list-items` 명령을 AWS CLI사용합니다.

```
aws mediastore-data list-items --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name --
region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "Items": [
        {
            "Type": "FOLDER",
            "Name": "folder_1"
        },
        {
            "LastModified": 1563571940.861,
            "ContentLength": 2307346,
            "Name": "file1234.jpg",
            "ETag":
 "111a1a22222a1a1a222abc333a444444b55ab1111ab2222222222ab333333a2b",
            "ContentType": "image/jpeg",
            "Type": "OBJECT"
        }
    ]
}
```

> **Note**
>
> `seconds_since_create` 규칙이 적용되는 객체는 `list-items` 응답에 포함되지 않습니다.

객체 세부 정보 보기

객체를 업로드하면 AWS Elemental MediaStore가 세부 정보(수정일, 콘텐츠 길이, ETag(엔터티 태그), 콘텐츠 유형 등)를 저장합니다. 객체의 메타데이터가 사용되는 방법에 대한 자세한 내용은 MediaStore의 HTTP 캐시와의 상호 작용 단원을 참조하십시오.

객체의 세부 정보를 보려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, 보려는 객체가 들어 있는 컨테이너의 이름을 선택합니다.
3. 보려는 객체가 폴더 내에 있으면 해당 객체가 보일 때까지 폴더 이름을 계속 선택합니다.
4. 객체 이름을 선택합니다.

   세부 정보 창이 나타나고 객체에 대한 정보가 표시됩니다.

객체의 세부 정보를 보려면(AWS CLI)

- 에서 `describe-object` 명령을 AWS CLI사용합니다.

```
aws mediastore-data describe-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name/
file1234.jpg --region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentLength": "2307346",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

객체 다운로드

콘솔을 사용하여 객체를 다운로드할 수 있습니다. AWS CLI 를 사용하여 객체를 다운로드하거나 객체의 일부만 다운로드할 수 있습니다.

객체를 다운로드하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서, 다운로드하려는 객체가 들어 있는 컨테이너 이름을 선택합니다.
3. 다운로드하려는 객체가 폴더 내에 있으면 해당 객체가 보일 때까지 폴더 이름을 계속 선택합니다.
4. 객체 이름을 선택합니다.

5. 객체 세부 정보 페이지에서 다운로드를 선택합니다.

객체를 다운로드하려면(AWS CLI)

- 에서 `get-object` 명령을 AWS CLI사용합니다.

```
aws mediastore-data get-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path=/folder_name/
README.md README.md --region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "ContentLength": "2307346",
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",
    "StatusCode": 200
}
```

객체의 일부를 다운로드하려면(AWS CLI)

- 에서 `get-object` 명령을 AWS CLI사용하고 범위를 지정합니다.

```
aws mediastore-data get-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name/
README.md --range="bytes=0-100" README2.md --region us-west-2
```

다음 예제는 반환 값을 보여줍니다.

```
{
    "StatusCode": 206,
    "ContentRange": "bytes 0-100/2307346",
    "ContentLength": "101",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentType": "image/jpeg",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

객체 삭제

AWS Elemental MediaStore는 컨테이너에서 객체를 삭제하는 다양한 옵션을 제공합니다.

- 개별 객체를 삭제합니다. 요금이 부과되지 않습니다.
- 컨테이너 내의 모든 객체를 한 번에 삭제하도록 컨테이너를 비웁니다. 이 프로세스는 API 호출을 사용하므로 일반 API 요금이 적용됩니다.
- 특정 수명에 도달할 때 객체를 삭제하도록 객체 수명 주기 정책을 추가합니다. 요금이 부과되지 않습니다.

객체 삭제

콘솔 또는 AWS CLI를 사용하여 객체를 개별적으로 삭제할 수 있습니다. 또는 컨테이너에서 특정 수명에 도달하면 객체를 자동으로 삭제하도록 객체 수명 주기 정책을 추가하거나 해당 컨테이너 내의 모든 객체를 삭제하도록 컨테이너를 비우기할 수 있습니다.

> **Note**
>
> 폴더에 있는 유일한 객체를 삭제하면 AWS Elemental MediaStore에서 해당 폴더와 그 상위의 빈 폴더를 자동으로 삭제합니다. 예를 들어 `premium`이라는 폴더가 있고 이 폴더에는 파일이 없으며 `canada`라는 하위 폴더만 한 개 있다고 가정해 보겠습니다. `canada` 하위 폴더에는 `mlaw.ts`라는 파일이 한 개 있습니다. `mlaw.ts` 파일을 삭제하면 서비스는 `premium` 폴더와 `canada` 폴더를 모두 삭제합니다.

객체를 삭제하려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 삭제하려는 객체가 들어 있는 컨테이너 이름을 선택합니다.
3. 삭제하려는 객체가 폴더 내에 있으면 해당 객체가 보일 때까지 폴더 이름을 계속 선택합니다.
4. 객체 이름 왼쪽에 있는 옵션을 선택합니다.
5. 삭제를 선택합니다.

객체를 삭제하려면(AWS CLI)

- 에서 `delete-object` 명령을 AWS CLI사용합니다.

예시

```
aws mediastore-data --region us-west-2 delete-object --endpoint=https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path=/folder_name/
README.md
```

이 명령은 반환 값이 없습니다.

컨테이너 비우기

컨테이너를 비워 컨테이너 내에 저장된 모든 객체를 삭제할 수 있습니다. 또는 특정 기간이 지난 객체를 컨테이너에서 자동으로 삭제하는 객체 수명 주기 정책을 추가하거나, 객체를 개별적으로 삭제할 수 있습니다.

컨테이너를 비우려면(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 비우려는 컨테이너에 대한 옵션을 선택합니다.
3. 컨테이너 비우기를 선택합니다. 확인 메시지가 표시됩니다.
4. 텍스트 필드에 컨테이너 이름을 입력하여 컨테이너를 비우는지 확인한 다음 비움을 선택합니다.

AWS Elemental MediaStore 내 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. AWS Elemental MediaStore에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스 규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 MediaStore 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 MediaStore를 구성하는 방법을 보여줍니다. 또한 MediaStore 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

항목

AWS Elemental MediaStore의 데이터 보호

AWS 공동 책임 모델 AWS Elemental MediaStore의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라

에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 MediaStore 또는 기타 AWS 서비스 에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

데이터 암호화

MediaStore는 업계 표준 AES-256 알고리즘을 사용하여 저장 중인 컨테이너와 개체를 암호화합니다. 다음과 같은 방법으로 MediaStore를 사용하여 데이터를 보호하는 것이 좋습니다.

- 컨테이너 정책을 생성하여 해당 컨테이너의 모든 폴더와 개체에 대한 액세스 권한을 제어합니다. 자세한 내용은 the section called “컨테이너 정책” 단원을 참조하십시오.

- 교차 오리진 리소스 공유(CORS 정책)은 MediaStore 리소스에 대한 교차 출처 액세스를 선택적으로 허용합니다. CORS를 사용하면 한 도메인에서 로드된 클라이언트 웹 애플리케이션이 다른 도메인에 있는 리소스와 상호 작용하도록 허용할 수 있습니다. 자세한 내용은 the section called “CORS 정책” 단원을 참조하십시오.

AWS Elemental MediaStore에 대한 자격 증명 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 누가 MediaStore 리소스를 사용하도록 인증되고(로그인됨) 권한이 부여되는지(권한 있음)를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한을 요청합니다(AWS Elemental MediaStore 자격 증명 및 액세스 문제 해결 참조).
- 서비스 관리자 - 사용자 액세스를 결정하고 권한 요청을 제출합니다(AWS Elemental MediaStore가 IAM과 작업하는 방법 참조).
- IAM 관리자 - 액세스를 관리하기 위한 정책을 작성합니다(AWS Elemental MediaStore에 대한 자격 증명 기반 정책 참조).

ID를 통한 인증

인증은 AWS 자격 증명으로에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS Sign-In 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 자격 증명

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 자격 증명은 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 IAM Identity Center란 무엇인가요? 섹션을 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 자격 증명입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 권한 관리를 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 위탁자가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 이를 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 해당 권한을 정의합니다.

ID 기반 정책

자격 증명 기반 정책은 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

자격 증명 기반 정책은 인라인 정책(단일 자격 증명에 직접 포함) 또는 관리형 정책(여러 자격 증명에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한

정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 타입

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - 자격 증명 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) - 계정 내 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 - 역할 또는 페더레이션 사용자에 대해 임시 세션을 생성할 때 파라미터 자격으로 전달되는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS Elemental MediaStore가 IAM과 작업하는 방법

IAM을 사용하여 MediaStore에 대한 액세스를 관리하기 전에 MediaStore와 함께 사용할 수 있는 IAM 기능을 알아보십시오.

AWS Elemental MediaStore와 함께 사용할 수 있는 IAM 기능

IAM 기능	MediaStore 지원
ID 기반 정책	예
리소스 기반 정책	예

IAM 기능	MediaStore 지원
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACLs	아니요
ABAC(정책 내 태그)	부분적
임시 자격 증명	예
위탁자 권한	예
서비스 역할	예
서비스 연결 역할	아니요

MediaStore 및 기타 AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

MediaStore에 대한 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

MediaStore에 대한 자격 증명 기반 정책 예제

MediaStore 자격 증명 기반 정책의 예를 보려면 AWS Elemental MediaStore에 대한 자격 증명 기반 정책 섹션을 참조하세요.

MediaStore 내 리소스 기반 정책

리소스 기반 정책 지원: 예

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 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 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 위탁자로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

> **Note**
>
> MediaStore는 또한 컨테이너에서 작업을 수행할 수 있는 보안 주체 엔터티(계정, 사용자, 역할 및 페더레이션 사용자)를 정의하는 컨테이너 정책을 지원합니다. 자세한 내용은 컨테이너 정책 단원을 참조하십시오.

MediaStore에 대한 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action` 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

MediaStore 작업 목록을 보려면 서비스 승인 참조의 AWS Elemental MediaStore에서 정의한 작업을 참조하세요.

MediaStore의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
mediastore
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "mediastore:action1",
      "mediastore:action2"
         ]
```

MediaStore 자격 증명 기반 정책의 예를 보려면 AWS Elemental MediaStore에 대한 자격 증명 기반 정책 섹션을 참조하세요.

MediaStore에 대한 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

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

MediaStore 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 승인 참조의 AWS Elemental MediaStore에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS Elemental MediaStore에서 정의한 작업을 참조하세요.

MediaStore 컨테이너 리소스의 ARN은 다음과 같습니다.

```
arn:${Partition}:mediastore:${Region}:${Account}:container/${containerName}
```

ARN 형식에 대한 자세한 내용은 Amazon 리소스 이름(ARNs) 및 AWS 서비스 네임스페이스를 참조하세요.

예를 들어, 문에서 `AwardsShow` 컨테이너를 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:mediastore:us-east-1:111122223333:container/AwardsShow"
```

MediaStore에 대한 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

MediaStore 조건 키 목록을 보려면 서비스 승인 참조의 AWS Elemental MediaStore의 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 AWS Elemental MediaStore에서 정의한 작업을 참조하세요.

MediaStore 자격 증명 기반 정책의 예를 보려면 AWS Elemental MediaStore에 대한 자격 증명 기반 정책 섹션을 참조하세요.

MediaStore의 ACL

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

MediaStore를 사용한 ABAC

ABAC 지원(정책의 태그): 부분적

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

MediaStore에서 임시 보안 인증 정보 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션을 사용하거나 역할을 전환할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM 임시 보안 자격 증명 및 IAM으로 작업하는 AWS 서비스 섹션을 참조하세요.

MediaStore의 서비스 간 보안 주체 권한

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

MediaStore의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 MediaStore 기능이 중단될 수 있습니다. MediaStore에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

MediaStore에 대한 서비스 연결 역할

서비스 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 나타나 AWS 계정 며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 Yes이(가) 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

AWS Elemental MediaStore에 대한 자격 증명 기반 정책

기본적으로 사용자 및 역할에는 MediaStore리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 포함하여 MediaStore에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 인증 참조에서 AWS Elemental MediaStore에 대한 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- MediaStore 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 MediaStore 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.

- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

MediaStore 콘솔 사용

AWS Elemental MediaStore 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은에서 MediaStore 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 여전히 MediaStore 콘솔을 사용할 수 있도록 하려면 MediaStore *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책을 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Elemental MediaStore 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 MediaStore및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- MediaStore에서 작업을 수행할 권한이 없습니다
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 MediaStore 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

MediaStore에서 작업을 수행할 권한이 없습니다

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 mediastore:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 mediastore:GetWidget on resource: my-example-widget
```

이 경우, mediastore:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 MediaStore에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 MediaStore에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
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 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 MediaStore 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제

어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- MediaStore에서 이러한 기능을 지원하는지 여부를 알아보려면 AWS Elemental MediaStore가 IAM과 작업하는 방법 섹션을 참조하세요.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 교차 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

에서 로깅 및 모니터링 AWS Elemental MediaStore

이 단원에는 보안을 위한 AWS Elemental MediaStore 의 로깅 및 모니티링 옵션에 대한 개요가 나와 있습니다. MediaStore의 로깅 및 모니터링에 대한 자세한 내용은 AWS Elemental MediaStore의 모니터링 및 태깅 섹션을 참조하세요.

모니터링은 AWS Elemental MediaStore 및 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 다중 지점 장애가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분에서 모니터링 데이터를 수집해야 합니다.는 MediaStore 리소스를 모니터링하고 잠재적 인시던트에 대응하기 위한 여러 도구를 AWS 제공합니다.

Amazon CloudWatch 경보

CloudWatch 경보를 사용하면 지정한 기간 동안 단일 지표를 감시할 수 있습니다. 지표가 지정된 임계값을 초과하면 Amazon SNS 주제 또는 AWS Auto Scaling 정책으로 알림이 전송됩니다. CloudWatch 경보는 특정 상태에 있다고 해서 작업을 호출하지 않습니다. 대신, 상태가 변경되어 지정된 기간 동안 유지되어야 합니다. 자세한 내용은 CloudWatch를 사용하여 모니터링 단원을 참조하십시오.

AWS CloudTrail 로그

CloudTrail은에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공합니다 AWS Elemental MediaStore. CloudTrail에서 수집한 정보를 사용하여 MediaStore에 수행된 요청, 요청이 수

행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다. 자세한 내용은 CloudTrail을 사용하여 API 직접 호출 로깅 단원을 참조하십시오.

AWS Trusted Advisor

Trusted Advisor 는 수십만 명의 AWS 고객에게 서비스를 제공하여 학습한 모범 사례를 활용합니다.는 AWS 환경을 Trusted Advisor 검사한 다음 비용 절감, 시스템 가용성 및 성능 개선 또는 보안 격차 해소를 위한 기회가 존재할 때 권장 사항을 제시합니다. 모든 AWS 고객은 Trusted Advisor 검사 5회에 액세스할 수 있습니다. 비즈니스 또는 엔터프라이즈 지원 플랜을 보유한 고객은 모든 Trusted Advisor 검사를 볼 수 있습니다.

자세한 내용은 AWS Trusted Advisor 단원을 참조하십시오.

AWS Elemental MediaStore에 대한 규정 준수 검증

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 내 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

AWS Elemental MediaStore의 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 지연 시간이 짧고 처리량이 많으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공합니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS 글로벌 인프라 외에도 MediaStore는 데이터 복원력 및 백업 요구 사항을 지원하는 데 도움이 되는 여러 기능을 제공합니다.

AWS Elemental MediaStore의 인프라 보안

관리형 서비스인 AWS Elemental MediaStore는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 MediaStore에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 AWS Elemental MediaStore가 다른 서비스에 리소스에 부여하는 권한을 제한하는 것을 권장합니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 `aws:SourceArn`을 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 `aws:SourceAccount`을(를) 사용합니다.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 `aws:SourceArn` 전역 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드 문자(*)를 포함한 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용합니다. 예를 들어 `arn:aws:servicename:*:123456789012:*`입니다.

만약 aws:SourceArn 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 글로벌 조건 컨텍스트 키를 모두 사용해야 합니다.

aws:SourceArn의 값은 MediaStore가 해당 리전 및 계정에 CloudWatch 로그를 게시하는 구성이어야 합니다.

다음 예는 MediaStore에서 aws:SourceArn 및 aws:SourceAccount 글로벌 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ConfusedDeputyPreventionExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "mediastore.amazonaws.com"
      },
      "Action": "mediastore:CreateContainer",
      "Resource": [
        "arn:aws:mediastore:us-east-2:333333333333:container/ResourceName/*"
      ],
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:mediastore:*:333333333333:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "333333333333"
        }
      }
    }
  ]
}
```

AWS Elemental MediaStore의 모니터링 및 태깅

모니터링은 AWS Elemental MediaStore 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다.는 MediaStore를 모니터링하고, 이상이 있을 때 이를 보고하고, 필요한 경우 자동 조치를 취할 수 있도록 다음과 같은 모니터링 도구를 AWS 제공합니다.

- AWS CloudTrail는 AWS 계정에서 또는 계정을 대신하여 수행한 API 호출 및 관련 이벤트를 캡처하고 사용자가 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 호출한 사용자 및 계정 AWS, 호출이 수행된 소스 IP 주소, 호출이 발생한 시기를 식별할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.
- Amazon CloudWatch는 AWS 리소스와 AWS 실행 중인 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.
- Amazon CloudWatch Events는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트 스트림을 제공합니다. 일반적으로 AWS 서비스는 몇 초 만에 CloudWatch Events에 이벤트 알림을 전송하지만 경우에 따라 1분 이상 걸릴 수 있습니다. CloudWatch Events를 사용하면 특정 이벤트를 감시하고 이러한 이벤트가 발생할 때 다른 AWS 서비스에서 자동화된 작업을 트리거하는 규칙을 작성할 수 있으므로 자동화된 이벤트 기반 컴퓨팅이 가능합니다. 자세한 내용은 Amazon CloudWatch Events 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.

MediaStore 컨테이너에 태그 형태로 메타데이터를 지정할 수도 있습니다. 각 태그는 사용자가 정의하는 키와 값으로 구성되는 레이블입니다. 태그를 사용하면 리소스를 보다 쉽게 관리, 검색 및 필터링할 수 있습니다. 태그를 사용하여 AWS 관리 콘솔에서 AWS 리소스를 구성하고, 모든 리소스에 대한 사용 및 결제 보고서를 생성하고, 인프라 자동화 활동 중에 리소스를 필터링할 수 있습니다 AWS .

항목

- AWS Elemental MediaStore 리소스에 태그 지정

AWS CloudTrail을 사용한 AWS Elemental MediaStore API 호출 로깅

AWS Elemental MediaStore는 MediaStore에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 MediaStore 콘솔의 호출 및 MediaStore API에 대한 코드 호출을 포함하여 MediaStore에 대한 API 호출의 하위 세트를 이벤트로 캡처합니다. 추적을 생성하면 MediaStore에 대한 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 MediaStore에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 등을 확인할 수 있습니다.

구성 및 사용 방법을 포함하여 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

항목

- CloudTrail의 AWS Elemental MediaStore 정보
- 예: AWS Elemental MediaStore 로그 파일 항목

CloudTrail의 AWS Elemental MediaStore 정보

AWS 계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. AWS Elemental MediaStore에서 지원되는 이벤트 활동이 발생하면 해당 활동은 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 설명은 CloudTrail 이벤트 기록으로 이벤트 보기를 참조하세요.

MediaStore에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 지역에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 정보는 다음의 주제를 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합

- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

AWS Elemental MediaStore는 CloudTrail 로그 파일에 있는 이벤트로 다음 작업의 로깅을 지원합니다.

- `CreateContainer`
- `DeleteContainer`
- `DeleteContainerPolicy`
- `DeleteCorsPolicy`
- `DescribeContainer`
- `GetContainerPolicy`
- `GetCorsPolicy`
- `ListContainers`
- `PutContainerPolicy`
- `PutCorsPolicy`

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트 사용자로 했는지 사용자 보안 인증으로 했는지 여부
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

예: AWS Elemental MediaStore 로그 파일 항목

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함하고 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출에 대한 순서 지정된 스택 추적이 아니기 때문에 특정 순서로 표시되지 않습니다.

다음 예제는 `CreateContainer` 작업을 보여주는 CloudTrail 로그 항목입니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEFGHIJKL123456789",
        "arn": "arn:aws:iam::111122223333:user/testUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "testUser",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-07-09T12:55:42Z"
            }
        },
        "invokedBy": "signin.amazonaws.com"
    },
    "eventTime": "2018-07-09T12:56:54Z",
    "eventSource": "mediastore.amazonaws.com",
    "eventName": "CreateContainer",
    "awsRegion": "ap-northeast-1",
    "sourceIPAddress": "54.239.119.16",
    "userAgent": "signin.amazonaws.com",
    "requestParameters": {
        "containerName": "TestContainer"
    },
    "responseElements": {
        "container": {
            "status": "CREATING",
            "creationTime": "Jul 9, 2018 12:56:54 PM",
            "name": " TestContainer ",
            "aRN": "arn:aws:mediastore:ap-northeast-1:111122223333:container/
TestContainer"
        }
    },
    "requestID":
 "MNCTGH4HRQJ27GRMBVDPIVHEP4LO2BN6MUVHBCPSHOAWNSOKSXCO24B2UEOBBND5DONRXTMFK3TOJ4G7AHWMESI",
    "eventID": "7085b140-fb2c-409b-a329-f567912d704c",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Amazon CloudWatch를 사용한 AWS Elemental MediaStore 모니터링

원시 데이터를 수집하여 읽을 수 있는 지표로 처리하는 CloudWatch를 사용하면 AWS Elemental MediaStore를 모니터링할 수 있습니다. CloudWatch는 통계를 15개월간 보관하므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 특정 임곗값을 주시하다가 해당 임곗값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

AWS 는 MediaStore를 모니터링하고, 이상이 있을 때 보고하고, 적절한 경우 자동 조치를 취할 수 있는 다음과 같은 모니터링 도구를 제공합니다.

- Amazon CloudWatch Logs는 AWS Elemental MediaStore와 같은 AWS 서비스에서 로그 파일을 모니터링, 저장 및 액세스할 수 있게 해줍니다. CloudWatch Logs를 사용하여 로그 데이터로 애플리케이션 및 시스템을 모니터링할 수 있습니다. 예를 들어 CloudWatch Logs에서는 애플리케이션 로그에서 발생하는 오류의 수를 추적하고 오류 비율이 지정한 임계값을 초과할 때마다 알림을 전송할 수 있습니다. CloudWatch Logs는 모니터링하는 데 로그 데이터를 사용하므로 코드를 변경할 필요가 없습니다. 예를 들어 애플리케이션 로그에서 특정 리터럴(예: "ValidationException")을 모니터링하거나 일정 기간 동안 생성된 `PutObject` 요청을 계수할 수 있습니다. 검색할 단어가 발견되면 CloudWatch Logs는 지정한 CloudWatch 지표로 데이터를 보고합니다. 로그 데이터는 전송 시는 물론 저장 시에도 암호화됩니다.
- Amazon CloudWatch Events는 MediaStore 객체와 같은 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트를 제공합니다. 일반적으로 AWS 서비스는 몇 초 만에 CloudWatch Events에 이벤트 알림을 전송하지만 경우에 따라 1분 이상 걸릴 수 있습니다. 규칙을 설정하면 일치하는 이벤트(예: `DeleteObject` 요청)를 검색하고 하나 이상의 대상 함수 또는 스트림으로 이벤트를 라우팅할 수 있습니다. CloudWatch Events는 운영 변경 사항이 발생할 때 이를 인식하게 됩니다. 또한 CloudWatch Events는 환경에 응답하기 위한 메시지를 전송하고 함수를 활성화하고 변경을 수행하고 상태 정보를 기록하는 등 이러한 운영 변경 사항에 응답하고 필요에 따라 교정 조치를 취합니다.

CloudWatch Logs

액세스 로깅은 컨테이너의 객체에 대해 이루어진 요청의 상세 레코드를 제공합니다. 액세스 로그는 보안 및 액세스 감사와 같은 여러 애플리케이션에 유용합니다. 또한 고객층을 파악하고 MediaStor 결제 요금을 확인할 수 있습니다. CloudWatch 로그는 다음과 같이 분류됩니다.

- 로그 스트림은 동일한 소스를 공유하는 로그 이벤트 시퀀스입니다.

- 로그 그룹은 동일한 보존 기간, 모니터링 및 액세스 제어 설정을 공유하는 로그 스트림 그룹입니다. 컨테이너에서 액세스 로깅에 액세스하면 MediaStore가 `/aws/mediastore/MyContainerName`과 같은 이름으로 로그 그룹을 생성합니다. 로그 그룹을 정의하고 각 그룹에 배치할 스트림을 지정할 수 있습니다. 하나의 로그 그룹에서 포함할 수 있는 로그 스트림의 수에는 할당량이 없습니다.

기본적으로 로그는 무기한으로 저장되고 만료 기간이 없습니다. 로그 그룹별로 보존 정책을 조정하여 무기한으로 보존하거나 1일부터 10년까지 보존 기간을 선택할 수 있습니다.

Amazon CloudWatch에 대한 권한 설정

AWS Identity and Access Management (IAM)을 사용하여 AWS Elemental MediaStore에 Amazon CloudWatch에 대한 액세스 권한을 부여하는 역할을 생성합니다. 계정에 대해 CloudWatch Logs를 게시하려면 다음 단계를 수행해야 합니다. CloudWatch는 계정에 대한 지표를 자동으로 게시합니다.

CloudWatch에 대한 MediaStore 액세스를 허용하기

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 정책(Policies)을 선택한 후 정책 생성(Create policy)을 선택합니다.
3. JSON 탭을 선택하고 다음 정책을 붙여넣습니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
```

```
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/mediastore/*"
        }
    ]
}
```

이 정책을 통해 MediaStore는 AWS 계정 내 모든 리전의 모든 컨테이너에 대한 로그 그룹 및 로그 스트림을 생성할 수 있습니다.

4. 정책 검토를 선택합니다.
5. 정책 검토 페이지에서 이름에 **MediaStoreAccessLogsPolicy**를 입력한 다음 정책 생성을 선택합니다.
6. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.
7. Another AWS account(다른 AWS 계정) 역할 유형을 선택합니다.
8. 계정 ID에 AWS 계정 ID를 입력합니다.
9. 다음: 권한을 선택합니다.
10. 검색 상자에 **MediaStoreAccessLogsPolicy**을(를) 입력합니다.
11. 새 정책 옆에 있는 확인란을 선택한 다음, 다음: 태그를 선택합니다.
12. 다음: 검토를 선택하여 새 사용자를 미리 봅니다.
13. 역할 이름에 **MediaStoreAccessLogs**를 입력한 다음 역할 생성을 선택합니다.
14. 확인 메시지에서 방금 생성한 역할의 이름(**MediaStoreAccessLogs**)을 선택합니다.
15. 역할의 요약 페이지에서 신뢰 관계 탭을 선택합니다.
16. 신뢰 관계 편집을 선택합니다.
17. 정책 문서에서 보안 주체부터 MediaStore 서비스까지 변경합니다. 형식은 다음과 같아야 합니다.

```
"Principal": {
   "Service": "mediastore.amazonaws.com"
},
```

전체 결과는 다음과 같습니다.

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "mediastore.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

18. 신뢰 정책 업데이트를 선택합니다.

컨테이너에 대한 액세스 로깅 활성화

기본적으로 AWS Elemental MediaStore는 액세스 로그를 수집하지 않습니다. 컨테이너에서 액세스 로깅을 활성화하면 MediaStore가 해당 컨테이너에 저장된 객체의 액세스 로그를 Amazon CloudWatch로 전송합니다. 액세스 로그는 컨테이너에 저장된 객체에 대해 이루어진 요청의 상세 레코드를 제공합니다. 이 정보에는 요청 유형, 요청에 지정된 리소스, 요청이 처리된 시간 및 날짜가 포함될 수 있습니다.

> **Important**
>
> MediaStore 컨테이너에서 액세스 로깅을 활성화하더라도 별도 요금이 부과되지 않습니다. 단, 이 서비스가 사용자에게 전달하는 로그 파일에 대해서는 일반적인 스토리지 요금이 발생합니다. (로그 파일은 언제든지 삭제할 수 있습니다.) AWS 는 로그 파일 전달에 따른 데이터 전송 요금이 발생하지는 않지만 로그 파일 액세스에 따른 일반 데이터 전송 요금은 부과됩니다.

액세스 로깅을 활성화하려면(AWS CLI)

- 에서 `start-access-logging` 명령을 AWS CLI사용합니다.

  ```
  aws mediastore start-access-logging --container-name LiveEvents --region us-west-2
  ```

  이 명령은 반환 값이 없습니다.

컨테이너에 대한 액세스 로깅 비활성화

컨테이너에서 액세스 로깅을 비활성화하면 AWS Elemental MediaStore가 액세스 로그를 Amazon CloudWatch로 전송하지 않습니다. 이러한 액세스 로그는 저장되지 않으며 따라서 검색할 수 없습니다.

액세스 로그를 비활성화하려면(AWS CLI)

- 에서 `stop-access-logging` 명령을 AWS CLI사용합니다.

```
aws mediastore stop-access-logging --container-name LiveEvents --region us-west-2
```

이 명령은 반환 값이 없습니다.

AWS Elemental MediaStore의 액세스 로깅 문제 해결

AWS Elemental MediaStore 액세스 로그가 Amazon CloudWatch에 표시되지 않을 경우 다음 표에서 예상 원인 및 해결책을 참조하세요.

> **Note**
>
> 문제 해결 프로세스를 지원하려면 AWS CloudTrail 로그를 활성화해야 합니다.

증상	예상 원인	해결책
CloudTrail 로그가 활성화되었지만 CloudTrail 이벤트가 전혀 보이지 않습니다.	IAM 역할이 존재하지 않거나 그 이름, 권한 또는 신뢰 정책이 잘못되었습니다.	올바른 이름, 권한 및 신뢰 정책으로 역할을 생성합니다. the section called “CloudWatch에 대한 권한 설정”을(를) 참조하세요.
`DescribeContainer` API 요청을 제출했지만 응답에 `AccessLoggingEnabled` 파라미터가 `False`의 값을 가지는 것으로 나타납니다. 또한 성공적인 `DescribeLogGroup` , `CreateLogGroup` , `DescribeL`	IAM 역할이 존재하지 않거나 그 이름, 권한 또는 신뢰 정책이 잘못되었습니다.	올바른 이름, 권한 및 신뢰 정책으로 역할을 생성합니다. the section called “CloudWatch에 대한 권한 설정”을(를) 참조하세요.

증상	예상 원인	해결책
ogStream 또는 CreateLog Stream 호출을 생성한 MediaStor eAccessLogs 역할에 대한 CloudTrail 이벤트가 전혀 보이지 않습니다.	컨테이너에서 액세스 로깅이 활성화되지 않았습니다.	컨테이너에서 액세스 로그를 활성화합니다. the section called “액세스 로깅 활성화”을(를) 참조하세요.

CloudTrai 콘솔에서 MediaStor eAccessLogs 역할과 관련된 액세스 거부 오류 이벤트가 표시됩니다. 이 CloudTrail 이벤트는 다음과 같은 행을 포함할 수 있습니다.
 	
`"eventSource": "logs.amazonaws.com",`
 	
`"errorCode": "AccessDenied",`
 	
`"errorMessage": "User: arn:aws:sts::111122223333:assumed-role/MediaStoreAccessLogs/MediaStoreAccessLogsSession is not authorized to perform: logs:DescribeLogGroups on resource: arn:aws:logs:us-west-2:111122223333:log-group::log-stream:",`
	IAM 역할에는 AWS Elemental MediaStore에 대해 올바른 권한이 없습니다.	IAM 역할을 업데이트하여 올바른 권한 및 신뢰 정책을 부여합니다. the section called “CloudWatch에 대한 권한 설정”을(를) 참조하세요.

증상	예상 원인	해결책
전체 컨테이너에 대해 로그가 전혀 보이지 않습니다.	계정이 리전별 계정당 로그 그룹에 대한 CloudWatch 할당량을 초과했을 수 있습니다. Amazon CloudWatch Logs 사용 설명서의 로그 그룹 할당량을 참조하세요.	CloudWatch 콘솔에서 계정이 로그 그룹에 대한 CloudWatch 할당량에 도달했는지 확인합니다. 필요한 경우 할당량 증가를 요청합니다.
CloudWatch에서 일부 로그가 보이지만 예상대로 모든 로그가 보이지는 않습니다.	계정이 리전별 계정당 초당 트랜잭션에 대한 CloudWatch 할당량을 초과했을 수 있습니다. Amazon CloudWatch Logs 사용 설명서의 PutLogEvents 할당량을 참조하세요.	리전별 계정당 초당 CloudWatch 트랜잭션의 할당량 증가를 요청합니다.

액세스 로그 형식

액세스 로그 파일은 일련의 JSON 형식 로그 레코드로 구성되며, 각 로그 레코드마다 한 요청이 표시됩니다. 로그 안의 필드 순서는 다를 수 있습니다. 다음은 2개의 로그 레코드로 구성된 로그의 예입니다.

```
{
  "Path": "/FootballMatch/West",
  "Requester": "arn:aws:iam::111122223333:user/maria-garcia",
  "AWSAccountId": "111122223333",
  "RequestID":
 "aaaAAA111bbbBBB222cccCCC333dddDDD444eeeEEE555fffFFF666gggGGG777hhhHHH888iiiIII999jjjJJJ",
  "ContainerName": "LiveEvents",
  "TotalTime": 147,
  "BytesReceived": 1572864,
  "BytesSent": 184,
  "ReceivedTime": "2018-12-13T12:22:06.245Z",
```

```
  "Operation": "PutObject",
  "ErrorCode": null,
  "Source": "192.0.2.3",
  "HTTPStatus": 200,
  "TurnAroundTime': 7,
  "ExpiresAt": "2018-12-13T12:22:36Z"
}
 {
  "Path": "/FootballMatch/West",
  "Requester": "arn:aws:iam::111122223333:user/maria-garcia",
  "AWSAccountId": "111122223333",
  "RequestID":
 "dddDDD444eeeEEE555fffFFF666gggGGG777hhhHHH888iiiIII999jjjJJJ000cccCCC333bbbBBB222aaaAAA",
  "ContainerName": "LiveEvents",
  "TotalTime": 3,
  "BytesReceived": 641354,
  "BytesSent": 163,
  "ReceivedTime": "2018-12-13T12:22:51.779Z",
  "Operation": "PutObject",
  "ErrorCode": "ValidationException",
  "Source": "198.51.100.15",
  "HTTPStatus": 400,
  "TurnAroundTime": 1,
  "ExpiresAt": null
}
```

다음 목록에서는 로그 레코드 필드에 대해 설명합니다.

AWSAccountId

요청을 수행하는 데 사용된 AWS 계정의 계정 ID입니다.

BytesReceived

MediaStore 서버가 수신하는 요청 본문의 바이트 수입니다.

BytesSent

MediaStore 서버가 송신하는 요청 본문의 바이트 수입니다. 이 값은 서버 응답과 함께 포함되는 `Content-Length` 헤더의 값과 동일한 경우가 자주 있습니다.

ContainerName

요청을 수신한 컨테이너의 이름입니다.

ErrorCode

MediaStore 오류 코드(예: `InternalServerError`) 발생한 오류가 없는 경우 - 문자가 표시됩니다. 상태 코드가 200(닫힌 연결 또는 서버가 응답을 스트리밍하기 시작한 후 오류를 나타냄)이라도 오류 코드가 표시될 수 있습니다.

ExpiresAt

객체의 만료 날짜 및 시간입니다. 이 값은 컨테이너에 적용되는 수명 주기 정책의 transient data rule에서 설정한 만료 기간을 기반으로 합니다. 이 값은 `ISO-8601` 날짜 시간이며 요청을 처리하는 호스트의 시스템 클록을 기준으로 합니다. 수명 주기 정책에 객체에 적용되는 임시 데이터 규칙이 없거나 컨테이너에 적용된 수명 주기 정책이 없는 경우 이 필드의 값은 `null`입니다. 이 필드는 `PutObject`, `GetObject`, `DescribeObject`, `DeleteObject` 작업에만 적용됩니다.

HTTPStatus

응답의 숫자 HTTP 상태 코드.

Operation

수행된 작업입니다(예: `PutObject` 또는 `ListItems`).

경로

컨테이너에서 객체가 저장된 경로. 작업이 경로 파라미터를 사용하지 않을 경우 - 문자가 표시됩니다.

ReceivedTime

요청이 수신된 시간입니다. 이 값은 `ISO-8601` 날짜 시간이며 요청을 처리하는 호스트의 시스템 클록을 기준으로 합니다.

요청자

요청을 생성하는 데 사용된 계정의 사용자 Amazon Resource Name(ARN). 인증되지 않은 요청은 이 값이 `anonymous`입니다. 인증이 완료되기 전에 요청이 실패하는 경우 이 필드가 로그에서 누락되었을 수 있습니다. 이러한 요청의 경우 `ErrorCode`에서 승인 문제를 식별할 수 있습니다.

RequestID

각 요청을 고유하게 식별하기 위해 AWS Elemental MediaStore에서 생성한 문자열입니다.

소스

호출을 생성한 AWS 서비스의 요청자 또는 서비스 보안 주체의 명백한 인터넷 주소. 중간 프록시 또는 방화벽이 요청을 생성한 시스템의 주소를 가릴 경우 이 값이 null로 설정됩니다.

TotalTime

서버 관점에서 요청이 플라이트 상태를 유지한 밀리초(ms) 단위 시간. 이 값은 서비스가 사용자로부터 요청을 수신한 시간에서 응답의 마지막 바이트를 전송한 시간까지 측정됩니다. 이 값은 클라이언트 관점에서 측정될 경우 네트워크 지연 시간에 의해 영향을 받으므로 서버 관점에서 측정됩니다.

TurnAroundTime

MediaStore가 요청을 처리하는 데 소비한 시간(밀리초)입니다. 이 값은 요청의 마지막 바이트가 수신된 시간부터 응답의 첫 바이트가 전송된 시간까지 측정됩니다.

로그 안의 필드 순서는 다를 수 있습니다.

로깅 상태 변경 시 일정 기간에 걸쳐 단계적으로 반영됨

로깅 상태를 변경한 후 실제 로그 파일의 전송에 반영되려면 어느 정도 시간이 지나야 합니다. 예를 들어, 컨테이너 A에 대해 로깅을 활성화할 경우 이후 1시간 동안 이루어진 요청 중 일부는 기록되지만 일부는 기록되지 않을 수도 있습니다. 컨테이너 B에 대해 로깅을 비활성화할 경우 이후 1시간 동안 일부 로그가 계속 전송될 수 있지만 다른 로그는 그렇지 않을 수 있습니다. 어떤 경우에도 사용자에 의한 추가 조치 없이 새로운 설정이 적용됩니다.

서버 로그 전송이 항상 보장되지는 않음

액세스 로그 레코드는 최대한 전송하겠지만 항상 모든 레코드가 전송된다고 보장할 수는 없습니다. 컨테이너에 대해 적절히 로깅이 구성된 대부분의 요청은 로그 레코드가 전송됩니다. 대부분 기록된 지 몇 시간 내로 로그 레코드가 전송되지만 더 자주 전송될 수 있습니다.

모든 액세스 로깅이 제때 전송될 것이라고 보장할 수는 없습니다. 특정 요청에 대한 로그 레코드는 요청이 실제로 처리된 후에 오랫동안 전송되거나 전혀 전송되지 않을 수도 있습니다. 액세스 로그는 컨테이너에 대한 트래픽의 특성을 파악할 용도로 제공되며, 실제로 로그 레코드가 누락되는 경우는 매우 드물지만 액세스 로깅 자체가 모든 요청을 완벽하게 기록할 목적으로 제공되는 것이 아닙니다.

완벽한 전송을 보장할 수 없는 액세스 로깅의 특성에 따라 AWS 포털에 제공되는 사용 보고서(AWS Management Console의 청구 및 비용 관리 보고서)에는 전송된 액세스 로그에 포함되지 않은 액세스 요청이 하나 이상 포함될 수 있습니다.

액세스 로그 형식 관련 프로그래밍 고려 사항

수시로 새로운 필드를 추가하여 액세스 로그 형식을 확장할 수 있습니다. 액세스 로그를 파싱하는 코드가 인식하지 못하는 추가 필드를 처리하도록 코드를 작성해야 합니다.

CloudWatch Events

Amazon CloudWatch Events를 사용하면 AWS 서비스를 자동화하고 애플리케이션 가용성 문제 또는 리소스 변경과 같은 시스템 이벤트에 자동으로 대응할 수 있습니다. 원하는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 작업을 지정할 수 있습니다.

> **Important**
>
> 일반적으로 AWS 서비스는 몇 초 만에 CloudWatch Events에 이벤트 알림을 전송하지만 경우에 따라 1분 이상 걸릴 수 있습니다.

파일을 컨테이너에 업로드하거나 컨테이너에서 제거할 경우 CloudWatch 서비스에서 두 가지 이벤트가 연속하여 수행됩니다.

1. the section called “객체 상태 변경 이벤트”
2. the section called “컨테이너 상태 변경 이벤트”

이러한 이벤트를 구독하는 자세한 방법은 Amazon CloudWatch를 참조하세요.

자동으로 트리거할 수 있는 태스크는 다음과 같습니다.

- AWS Lambda 함수 호출
- Amazon EC2 Run Command 호출
- Amazon Kinesis Data Streams로 이벤트 릴레이
- AWS Step Functions 상태 시스템 활성화
- Amazon SNS 주제 또는 AWS SMS 대기열 알림

CloudWatch Events를 AWS Elemental MediaStore에 사용하는 몇 가지 예는 다음과 같습니다.

- 컨테이너를 생성할 때마다 Lambda 함수를 실행
- 객체가 삭제될 때 Amazon SNS 주제를 알림

자세한 내용은 Amazon CloudWatch Events 사용 설명서를 참조하세요.

항목

- AWS Elemental MediaStore 객체 상태 변경 이벤트

- AWS Elemental MediaStore 컨테이너 상태 변경 이벤트

AWS Elemental MediaStore 객체 상태 변경 이벤트

이 이벤트는 객체의 상태가 변경되면(객체가 업로드되거나 삭제되면) 게시됩니다.

> **Note**
>
> 임시 데이터 규칙으로 인해 만료되는 객체는 만료 시 CloudWatch 이벤트를 생성하지 않습니다.

이 이벤트를 구독하는 자세한 방법은 Amazon CloudWatch를 참조하세요.

객체 업데이트됨

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Object State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:MondayMornings/Episode1/
Introduction.avi"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "UPDATE",
    "Path":"TVShow/Episode1/Pilot.avi",
    "ObjectSize":123456,
    "URL": "https://a832p1qeaznlp9.files.mediastore-us-west-2.com/Movies/
MondayMornings/Episode1/Introduction.avi"
  }
}
```

객체 제거됨

```
{
```

```
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Object State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:Movies/MondayMornings/Episode1/
Introduction.avi"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "REMOVE",
    "Path":"Movies/MondayMornings/Episode1/Introduction.avi",
    "URL": "https://a832p1qeaznlp9.files.mediastore-us-west-2.com/Movies/
MondayMornings/Episode1/Introduction.avi"
  }
}
```

AWS Elemental MediaStore 컨테이너 상태 변경 이벤트

이 이벤트는 컨테이너의 상태가 변경되면(컨테이너가 추가되거나 삭제되면) 게시됩니다. 이 이벤트를 구독하는 자세한 방법은 Amazon CloudWatch를 참조하세요.

컨테이너 생성됨

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Container State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:container/Movies"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "CREATE"
    "Endpoint": "https://a832p1qeaznlp9.mediastore-us-west-2.amazonaws.com"
  }
```

```
}
```

컨테이너 제거됨

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Container State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:container/Movies"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "REMOVE"
  }
}
```

Amazon CloudWatch 지표를 사용한 AWS Elemental MediaStore 모니터링

원시 데이터를 수집하여 읽을 수 있는 지표로 처리하는 CloudWatch를 사용하면 AWS Elemental MediaStore를 모니터링할 수 있습니다. CloudWatch는 통계를 15개월간 보관하므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 특정 임곗값을 주시하다가 해당 임곗값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

AWS Elemental MediaStore의 경우 `BytesDownloaded`를 감시하다가 지표가 특정 임계값에 도달하면 본인에게 이메일을 보낼 수 있습니다.

CloudWatch 콘솔을 사용하여 지표를 보려면

지표는 먼저 서비스 네임스페이스별로 그룹화된 다음 각 네임스페이스 내에서 다양한 차원 조합별로 그룹화됩니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. 모든 지표 아래에서 AWS/MediaPackage 네임스페이스를 선택합니다.

4. 지표를 보려면 지표 차원을 선택합니다. 예를 들어 컨테이너로 전송된 다양한 유형의 요청에 대한 지표를 보려면 `Request metrics by container`를 선택합니다.

를 사용하여 지표를 보려면 AWS CLI

- 명령 프롬프트에서 다음 명령을 사용합니다.

```
aws cloudwatch list-metrics --namespace "AWS/MediaStore"
```

AWS Elemental MediaStore 지표

다음 표에는 AWS Elemental MediaStore가 CloudWatch로 전송하는 지표가 나와 있습니다.

> **Note**
>
> 지표를 보려면 컨테이너에 지표 정책을 추가하여 MediaStore가 Amazon CloudWatch에 지표를 전송하도록 허용해야 합니다.

지표	Description
`RequestCount`	작업 유형(`Put`, `Get`, `Delete`, `Describe`, `List`)으로 구분된 MediaStore 컨테이너에 대한 총 HTTP 요청 수입니다. 단위: 개 유효한 차원: • 컨테이너 이름 • 객체 그룹 이름 • 요청 유형 유효 통계: Sum
`4xxErrorCount`	4xx 오류를 초래한 MediaStore에 대한 HTTP 요청 수입니다. 단위: 개

지표	Description
	유효한 차원: • 컨테이너 이름 • 객체 그룹 이름 • 요청 유형 유효 통계: Sum
5xxErrorCount	5xx 오류를 초래한 MediaStore에 대한 HTTP 요청 수입니다. 단위: 개 유효한 차원: • 컨테이너 이름 • 객체 그룹 이름 • 요청 유형 유효 통계: Sum
BytesUploaded	MediaStore 컨테이너에 대한 요청에 대해 업로드된 바이트 수로, 여기서 요청에는 본문이 포함됩니다. 단위: 바이트 유효한 차원: • 컨테이너 이름 • 객체 그룹 이름 유효한 통계: Average(요청당 바이트), Sum(기간당 바이트), Sample Count, Min(P0.0과 동일), Max(p100과 동일), p0.0과 p99.9 사이의 모든 백분위수

지표	Description
BytesDownloaded	응답에 본문이 포함된 MediaStore 컨테이너에 대한 요청에 대해 다운로드된 바이트 수입니다. 단위: 바이트 유효한 차원: • 컨테이너 이름 • 객체 그룹 이름 유효한 통계: Average(요청당 바이트), Sum(기간당 바이트), Sample Count, Min(P0.0과 동일), Max(p100과 동일), p0.0과 p99.9 사이의 모든 백분위수
TotalTime	서버 관점에서 요청이 플라이트 상태를 유지한 밀리초 단위 시간. 이 값은 MediaStore가 요청을 수신한 시간부터 응답의 마지막 바이트를 전송한 시간까지 측정됩니다. 이 값은 클라이언트 관점에서 측정될 경우 네트워크 지연 시간에 의해 영향을 받으므로 서버 관점에서 측정됩니다. 단위: 밀리초 유효한 차원: • 컨테이너 이름 • 객체 그룹 이름 • 요청 유형 유효한 통계: Average, Min(P0.0과 동일), Max(p100과 동일), p0.0과 p100 사이의 모든 백분위수

지표	Description
`TurnaroundTime`	MediaStore가 요청을 처리하는 데 소비한 시간(밀리초)입니다. 이 값은 MediaStore가 요청의 마지막 바이트를 수신한 시간부터 응답의 첫 번째 바이트를 전송한 시간까지 측정됩니다. 단위: 밀리초 유효한 차원: • 컨테이너 이름 • 객체 그룹 이름 • 요청 유형 유효한 통계: Average, Min(P0.0과 동일), Max(p100과 동일), p0.0과 p100 사이의 모든 백분위수
`ThrottleCount`	제한된 MediaStore에 대한 HTTP 요청 수입니다. 단위: 개 유효한 차원: • 컨테이너 이름 • 객체 그룹 이름 • 요청 유형 유효 통계: Sum

AWS Elemental MediaStore 리소스에 태그 지정

태그는 사용자가 할당하거나가 AWS 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다. 각 태그는 두 부분으로 구성됩니다.

- 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 구성합니다. 많은 AWS 서비스가 태그 지정을 지원합니다. 따라서 서로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 서로 관련이 있음을 나타낼 수 있습니다. 예를 들어 AWS Elemental MediaLive 입력에 할당한 것과 동일한 태그를 AWS Elemental MediaStore ####에 할당할 수 있습니다.
- AWS 비용을 추적합니다. AWS 결제 및 비용 관리 대시보드에서 이러한 태그를 활성화합니다. AWS는 태그를 사용하여 비용을 분류하고 월별 비용 할당 보고서를 제공합니다. 자세한 내용은 AWS Billing 사용 설명서의 비용 할당 태그 사용을 참조하세요.

다음 섹션에서는 AWS Elemental MediaStore의 태그에 대한 추가 정보를 제공합니다.

AWS Elemental MediaStore의 지원 리소스

AWS Elemental MediaStore의 다음 리소스는 태그 지정을 지원합니다.

- ####

태그 추가 및 관리에 대한 자세한 설명은 태그 관리 섹션을 참조하세요.

AWS Elemental MediaStore는 AWS Identity and Access Management (IAM)의 태그 기반 액세스 제어 기능을 지원하지 않습니다.

태그 명칭 지정 및 사용 규칙

다음 기본 이름 지정 및 사용 규칙은 AWS Elemental MediaStore 리소스와 함께 태그를 사용할 때 적용합니다.

- 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. Amazon EC2 리소스는 모든 문자를 허용합니다.
- 태그 키와 값은 대소문자를 구분합니다. 태그를 대문자로 사용하는 전략을 세우고 이러한 전략을 모든 리소스 타입에 대해 일관되게 구현하는 것이 가장 좋습니다. 예컨대, `Costcenter`, `costcenter`

또는 CostCenter를 사용할지 결정하고 모든 태그에 대해 동일한 규칙을 사용합니다. 대/소문자가 일치하지 않는 유사한 태그를 사용하지 마십시오.

- aws: 접두사는 태그에 사용할 수 없으며 AWS 사용을 위해 예약되어 있습니다. 이 접두사가 지정된 태그 키나 값은 편집하거나 삭제할 수 없습니다. 이 접두사가 지정된 태그는 리소스당 태그 수 할당량에 포함되지 않습니다.

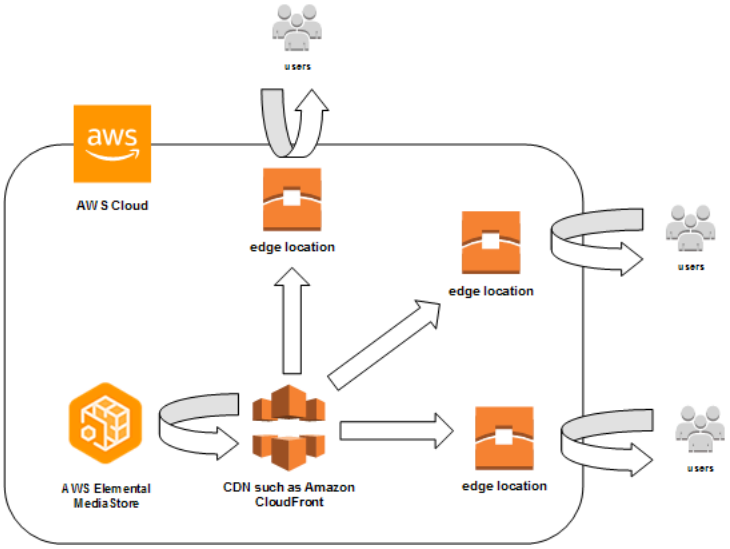
태그 관리

태그는 리소스의 Key 및 Value 속성으로 구성됩니다. AWS CLI 또는 MediaStore API를 사용하여 이러한 속성의 값을 추가, 편집 또는 삭제할 수 있습니다. 태그 작업에 대한 자세한 내용은 AWS Elemental MediaStore API 참조에서 다음 섹션을 참조하세요.

- CreateContainer
- ListTagsForResource
- 리소스
- TagResource
- UntagResource

CDN(콘텐츠 전송 네트워크) 작업

Amazon CloudFront 같은 콘텐츠 전송 네트워크(CDN)를 사용하여 AWS Elemental MediaStore에 저장된 콘텐츠를 제공할 수 있습니다. CDN은 전역적으로 배포된 서버 세트로 비디오 등의 콘텐츠를 캐싱합니다. 사용자가 콘텐츠를 요청하면 CDN은 이 요청을 지연 시간이 가장 짧은 엣지 로케이션으로 라우팅합니다. 콘텐츠가 해당 엣지 로케이션의 캐시에 이미 저장되어 있는 경우, CDN은 즉시 콘텐츠를 제공합니다. 엣지 로케이션에 콘텐츠가 없으면 CDN은 오리진(예: MediaStore 컨테이너)에서 콘텐츠를 가져와 사용자에게 배포합니다.



항목

Amazon CloudFront가 AWS Elemental MediaStore 컨테이너에 액세스하도록 허용

Amazon CloudFront를 사용하면 AWS Elemental MediaStore의 컨테이너에 저장한 콘텐츠를 제공할 수 있습니다. 이 작업을 다음 중 한 가지 방법으로 수행할 수 있습니다.

- 원본 액세스 제어(OAC) 사용 - (권장)가 CloudFront의 OAC 기능을 AWS 리전 지원하는 경우이 옵션을 사용합니다.
- 공유 암호 사용 - AWS 리전 가 CloudFront의 OAC 기능을 지원하지 않는 경우이 옵션을 사용합니다.

원본 액세스 제어(OAC) 사용

Amazon CloudFront의 원본 액세스 제어(OAC) 기능을 사용하여 AWS Elemental MediaStore 오리진을 향상된 보안으로 보호할 수 있습니다. MediaStore 오리진에 대한 CloudFront 요청에서 AWS Signature Version 4(SigV4)를 활성화하고 CloudFront가 요청에 서명해야 하는 시기와 서명 여부를 설정할 수 있습니다. 콘솔, API, SDK 또는 CLI를 통해 CloudFront의 OAC 기능에 액세스할 수 있으며 사용에 따른 추가 요금은 없습니다.

MediaStore를 사용하여 OAC 기능을 사용하는 방법에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서의 MediaStore 오리진에 대한 액세스 제한을 참조하세요.

공유 암호 사용

AWS 리전 가 Amazon CloudFront의 OAC 기능을 지원하지 않는 경우 CloudFront에 읽기 액세스 이상의 권한을 부여하는 정책을 AWS Elemental MediaStore 컨테이너에 연결할 수 있습니다.

> **Note**
>
> 에서 AWS 리전 지원하는 경우 OAC 기능을 사용하는 것이 좋습니다. 다음 절차에서는 MediaStore 컨테이너에 대한 액세스를 제한하기 위해 공유 암호로 MediaStore 및 CloudFront를 구성해야 합니다. 모범 보안 관행을 따르려면 이 수동 구성에 암호를 주기적으로 교체해야 합니다. MediaStore 오리진에서 OAC를 사용하면 CloudFront가 SigV4를 사용하여 요청에 서명하고 서명 매칭을 위해 MediaStore에 전달하도록 지시할 수 있으므로 암호를 사용하고 교체할 필요가 없습니다. 이렇게 하면 미디어 콘텐츠가 제공되기 전에 요청이 자동으로 확인되므로 MediaStore와 CloudFront를 통해 미디어 콘텐츠를 더 간단하고 안전하게 전송할 수 있습니다.

CloudFront가 컨테이너에 액세스하도록 허용하기(콘솔)

1. https://console.aws.amazon.com/mediastore/에서 MediaStore 콘솔을 엽니다.
2. 컨테이너 페이지에서 컨테이너 이름을 선택합니다.

   컨테이너 세부 정보 페이지가 나타납니다.
3. 컨테이너 정책 섹션에서 읽기 이상의 액세스 권한을 Amazon CloudFront에 부여하는 정책을 연결합니다.

Example

HTTPS를 통해 퍼블릭 읽기 액세스에 대한 예제 정책과 유사한 다음 예제 정책은 HTTPS를 통해 요청을 제출하는 누구든 `GetObject` 및 `DescribeObject` 명령을 허용하므로 이러한 요구 사항과 일치합니다. 또한 다음 예제 정책은 HTTPS 연결을 통해 요청이 발생하고 올바른 레퍼러 헤더를 포함하는 경우에만 CloudFront가 MediaStore 객체에 액세스할 수 있도록 허용하므로 워크플로의 보안을 더욱 강화합니다.

4. 컨테이너 CORS 정책 섹션에서 적절한 액세스 수준을 허용하는 정책을 할당합니다.

> **Note**
>
> CORS 정책은 브라우저 기반 플레이어에 액세스를 제공하려는 경우에만 필요합니다.

5. 다음과 같은 세부 정보를 기록해 두십시오.

   - 컨테이너에 할당된 데이터 엔드포인트. 이 정보는 컨테이너 페이지의 정보 섹션에서 확인할 수 있습니다. CloudFront에서 데이터 엔드포인트를 오리진 도메인 이름이라고 합니다.
   - 객체가 저장되는 컨테이너의 폴더 구조. CloudFront에서는 이를 오리진 경로라고 합니다. 이 설정은 선택 사항입니다. 오리진 경로에 대한 자세한 내용을 알아보려면 Amazon CloudFront 개발자 안내서를 참조하세요.

6. In CloudFront에서 AWS Elemental MediaStore의 서비스 콘텐츠를 제공하도록 구성된 배포를 만듭니다. 앞 단계에서 수집한 정보가 필요합니다.

정책을 MediaStore 컨테이너에 연결한 후에는 원본 요청에 HTTPS 연결만 사용하도록 CloudFront를 구성하고 올바른 보안 값이 포함된 사용자 지정 헤더도 추가해야 합니다.

리퍼러 헤더(콘솔)에 대한 비밀 값을 사용하여 HTTPS 연결을 통해 컨테이너에 액세스하도록 CloudFront를 구성하기

1. CloudFront 콘솔을 엽니다.
2. 오리진 페이지에서 MediaStore 오리진을 선택합니다.
3. 편집을 선택합니다.
4. 프로토콜에 대해 HTTP만 선택합니다.
5. 사용자 지정 헤더 추가 섹션에서 헤더 추가를 선택합니다.
6. 이름의 경우 리퍼러를 선택합니다. 값의 경우 컨테이너 정책에서 사용한 것과 동일한 *<secretValue>* 문자열을 사용하세요.

7. 저장을 선택하고 변경 내용을 배포합니다.

AWS Elemental MediaStore와 HTTP 캐시의 상호 작용

AWS Elemental MediaStore에서는 Amazon CloudFront와 같은 콘텐츠 전송 네트워크(CDN)를 통해 객체를 정확하고 효율적으로 캐싱할 수 있도록 객체를 저장합니다. 최종 사용자 또는 CDN이 MediaStore에서 개체를 검색할 때 서비스는 개체의 캐싱 동작에 영향을 주는 HTTP 헤더를 반환합니다. HTTP 1.1 캐싱 동작에 대한 표준은 RFC2616 섹션 13에서 찾을 수 있습니다. 이러한 헤더는 다음과 같습니다.

- **`ETag`**(사용자 지정할 수 없음) - 엔터티 태그 헤더는 MediaStore에서 보내는 응답의 고유 식별자입니다. 표준을 준수하는 CDN 및 웹 브라우저는 이 태그를 객체를 캐시하는 키로 사용합니다. MediaStore는 각 `ETag` 객체를 업로드할 때 각 객체에 대해 자동으로 생성합니다. 객체의 세부 정보를 보고 ETag 값을 결정할 수 있습니다.
- **`Last-Modified`**(사용자 지정할 수 없음) - 이 헤더의 값은 객체가 수정된 날짜 및 시간을 나타냅니다. MediaStore는 객체가 업로드될 때 이 값을 자동으로 생성합니다.
- **`Cache-Control`**(사용자 지정 가능) - 이 헤더의 값은 CDN이 객체가 수정되었는지 확인하기 전에 개체를 캐싱해야 하는 기간을 제어합니다. CLI 또는 API를 사용하여 MediaStore 컨테이너에 객체를 업로드할 때 이 헤더를 임의의 값으로 설정할 수 있습니다. 전체 유효한 값은 HTTP/1.1 설명서에 기술되어 있습니다. 객체를 업로드할 때 이 값을 설정하지 않으면 MediaStore에서는 객체를 검색할 때 이 헤더를 반환하지 않습니다.

  Cache-Control 헤더의 일반적인 사용 사례는 개체를 캐싱하는 기간을 지정하는 것입니다. 예를 들어, 인코더에서 자주 덮어쓰는 비디오 매니페스트 파일이 있다고 가정합니다. `max-age`를 10으로 설정하여 객체가 10초 동안만 캐싱되어야 함을 나타낼 수 있습니다. 또는 덮어쓰지 않는 저장된 비디오 세그먼트가 있다고 가정합니다. 이 객체에 대한 `max-age`를 31536000으로 설정하여 약 1년 동안 캐싱할 수 있습니다.

조건부 요청

MediaStore에 대한 조건부 요청

MediaStore는 조건부 요청(RFC7232의 설명에 따라 `If-None-Match` 및 `If-Modified-Since` 등의 요청 헤더 사용)과 무조건적인 요청에 동일하게 응답합니다. 즉, MediaStore에 유효한 `GetObject` 요청이 접수되면 클라이언트가 이미 개체를 가지고 있더라도 서비스는 항상 객체를 반환합니다.

CDN에 대한 조건부 요청

MediaStore를 대신하여 콘텐츠를 제공하는 CDN은 RFC7232 섹션 4.1에 설명된 대로 `304 Not Modified`을 반환하여 조건부 요청을 처리할 수 있습니다. 이는 요청자가 조건부 요청과 일치하는 객체를 이미 가지고 있기 때문에 전체 객체 컨텐츠를 전송할 필요가 없음을 나타냅니다.

CDN(및 HTTP/1.1을 준수하는 기타 캐시)은 오리진 서버에서 전달하는 `ETag` 및 `Cache-Control` 헤더를 기반으로 이러한 결정을 내립니다. 반복적으로 검색된 객체에 대한 업데이트를 위해 CDN이 MediaStore 오리진 서버를 쿼리하는 빈도를 제어하려면 객체를 MediaStore에 업로드할 때 해당 객체에 대한 `Cache-Control` 헤더를 설정합니다.

AWS SDK에서이 서비스 사용

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

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

이 서비스 관련 예제는 AWS SDKs를 사용하는 MediaStore의 코드 예제를 참조하세요.

예제 사용 가능 여부

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

AWS SDKs를 사용하는 MediaStore의 코드 예제

다음 코드 예제에서는 MediaStore를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDKs 사용하는 MediaStore의 기본 예제
  - AWS SDKs 사용하는 MediaStore에 대한 작업
    - AWS SDK 또는 CLI와 CreateContainer 함께 사용
    - AWS SDK 또는 CLI와 DeleteContainer 함께 사용
    - AWS SDK와 DeleteObject 함께 사용
    - AWS SDK 또는 CLI와 DescribeContainer 함께 사용
    - AWS SDK 또는 CLI와 GetObject 함께 사용
    - AWS SDK 또는 CLI와 ListContainers 함께 사용
    - AWS SDK 또는 CLI와 PutObject 함께 사용

AWS SDKs 사용하는 MediaStore의 기본 예제

다음 코드 예제에서는 AWS Elemental MediaStore SDKs에서의 기본 사항을 AWS 사용하는 방법을 보여줍니다.

예시

- AWS SDKs 사용하는 MediaStore에 대한 작업
  - AWS SDK 또는 CLI와 CreateContainer 함께 사용
  - AWS SDK 또는 CLI와 DeleteContainer 함께 사용
  - AWS SDK와 DeleteObject 함께 사용
  - AWS SDK 또는 CLI와 DescribeContainer 함께 사용
  - AWS SDK 또는 CLI와 GetObject 함께 사용

- AWS SDK 또는 CLI와 ListContainers 함께 사용
- AWS SDK 또는 CLI와 PutObject 함께 사용

AWS SDKs 사용하는 MediaStore에 대한 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 MediaStore 작업을 수행하는 방법을 보여줍니다. 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 AWS Elemental MediaStore API 참조를 참조하세요.

예시

- AWS SDK 또는 CLI와 CreateContainer 함께 사용
- AWS SDK 또는 CLI와 DeleteContainer 함께 사용
- AWS SDK와 DeleteObject 함께 사용
- AWS SDK 또는 CLI와 DescribeContainer 함께 사용
- AWS SDK 또는 CLI와 GetObject 함께 사용
- AWS SDK 또는 CLI와 ListContainers 함께 사용
- AWS SDK 또는 CLI와 PutObject 함께 사용

AWS SDK 또는 CLI와 **`CreateContainer`** 함께 사용

다음 코드 예시는 `CreateContainer`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

컨테이너 생성

다음 `create-container` 예시에서는 빈 새 컨테이너를 생성합니다.

```
aws mediastore create-container --container-name ExampleContainer
```

출력:

```
{
    "Container": {
        "AccessLoggingEnabled": false,
        "CreationTime": 1563557265,
        "Name": "ExampleContainer",
        "Status": "CREATING",
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
    }
}
```

자세한 내용은 AWS Elemental MediaStore 사용자 안내서의 컨테이너 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateContainer를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * 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 CreateContainer {
```

```
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
        try {
            CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }
```

```
            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateContainer를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **`DeleteContainer`** 함께 사용

다음 코드 예시는 `DeleteContainer`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

컨테이너 삭제

다음 `delete-container` 예시에서는 지정된 컨테이너를 삭제합니다. 객체가 없는 컨테이너만 삭제할 수 있습니다.

```
aws mediastore delete-container \
    --container-name=ExampleLiveDemo
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS Elemental MediaStore 사용자 안내서의 컨테이너 삭제를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteContainer를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * 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 CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteContainer를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK와 **DeleteObject** 함께 사용

다음 코드 예시는 DeleteObject의 사용 방법을 보여 줍니다.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * 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 DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """
```

```
                Usage:    <completePath> <containerName>

                Where:
                   completePath - The path (including the container) of the item
to delete.
                   containerName - The name of the container.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData,
String completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder()
                    .path(completePath)
                    .build();

            mediaStoreData.deleteObject(deleteObjectRequest);

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
```

```
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteObject를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeContainer** 함께 사용

다음 코드 예시는 DescribeContainer의 사용 방법을 보여 줍니다.

CLI

AWS CLI

컨테이너 세부 정보 보기

다음 describe-container 예시에서는 지정된 컨테이너의 세부 정보를 표시합니다.

```
aws mediastore describe-container \
    --container-name ExampleContainer
```

출력:

```
{
    "Container": {
```

```
        "CreationTime": 1563558086,
        "AccessLoggingEnabled": false,
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
        "Status": "ACTIVE",
        "Name": "ExampleContainer",
        "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com"
    }
}
```

자세한 내용은 AWS Elemental MediaStore 사용자 안내서의 컨테이너 세부 정보 보기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DescribeContainer를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * 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 DescribeContainer {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
DescribeContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            DescribeContainerResponse containerResponse =
mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
containerResponse.container().name());
            System.out.println("The container ARN is " +
containerResponse.container().arn());
            return containerResponse.container().status().toString();

        } catch (MediaStoreException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeContainer를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 `GetObject` 함께 사용

다음 코드 예시는 `GetObject`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

객체 다운로드

다음 `get-object` 예시에서는 지정된 엔드포인트에 객체를 다운로드합니다.

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path=/folder_name/README.md README.md
```

출력:

```
{
    "ContentLength": "2307346",
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",
    "StatusCode": 200
}
```

객체의 일부 다운로드

다음 `get-object` 예시에서는 지정된 엔드포인트에 객체의 일부를 다운로드합니다.

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path /folder_name/README.md \
    --range="bytes=0-100" README2.md
```

출력:

```
{
    "StatusCode": 206,
    "ContentRange": "bytes 0-100/2307346",
    "ContentLength": "101",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentType": "image/jpeg",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

자세한 내용은 AWS Elemental MediaStore 사용자 안내서의 객체 다운로드를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetObject를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
```

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * 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 GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName> <savePath>

                Where:
                   completePath - The path of the object in the container (for
 example, Videos5/sampleVideo.mp4).
                   containerName - The name of the container.
                   savePath - The path on the local drive where the file is
 saved, including the file name (for example, C:/AWS/myvid.mp4).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
```

```
                .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }

    public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .path(completePath)
                    .build();

            // Write out the data to a file.
            ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
            byte[] buffer = new byte[data.available()];
            data.read(buffer);

            File targetFile = new File(savePath);
            OutputStream outStream = new FileOutputStream(targetFile);
            outStream.write(buffer);
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
```

```
        return response.container().endpoint();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObject를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListContainers** 함께 사용

다음 코드 예시는 ListContainers의 사용 방법을 보여 줍니다.

CLI

AWS CLI

컨테이너 목록 보기

다음 list-containers 예시에서는 계정에 연결된 모든 컨테이너의 목록을 표시합니다.

```
aws mediastore list-containers
```

출력:

```
{
    "Containers": [
        {
            "CreationTime": 1505317931,
            "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
            "AccessLoggingEnabled": false,
            "Name": "ExampleLiveDemo"
        },
        {
            "CreationTime": 1506528818,
            "Endpoint": "https://fffgggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
```

```
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
            "AccessLoggingEnabled": false,
            "Name": "ExampleContainer"
        }
    ]
}
```

자세한 내용은 AWS Elemental MediaStore 사용자 안내서의 컨테이너 목록 보기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListContainers를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * 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 ListContainers {
```

```
    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
 mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListContainers를 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **PutObject** 함께 사용

다음 코드 예시는 PutObject의 사용 방법을 보여 줍니다.

CLI

AWS CLI

객체 업로드

다음 put-object 예시에서는 지정된 컨테이너에 객체를 업로드합니다. 객체가 컨테이너 내에 저장될 폴더 경로를 지정할 수 있습니다. 폴더가 이미 있는 경우 AWS Elemental MediaStore는 폴더에 객체를 저장합니다. 폴더가 없으면 폴더를 만든 후 그 폴더에 객체를 저장합니다.

```
aws mediastore-data put-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --body README.md \
    --path /folder_name/README.md \
    --cache-control "max-age=6, public" \
    --content-type binary/octet-stream
```

출력:

```
{
    "ContentSHA256":
 "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",
    "StorageClass": "TEMPORAL",
    "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"
}
```

자세한 내용은 AWS Elemental MediaStore 사용자 안내서의 객체 업로드를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutObject를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
```

```
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * 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 PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = """

                To run this example, supply the name of a container, a file
 location to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
```

```
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .path(completePath)
                    .contentType("video/mp4")
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutObject를 참조하십시오.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요AWS SDK에서이 서비스 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS Elemental MediaStore 내 할당량

Service Quotas 콘솔은 AWS Elemental MediaStore 할당량에 대한 정보를 제공합니다. 기본값 할당량을 볼 수 있을 뿐만 아니라 Service Quotas 콘솔을 사용하여 조정 가능한 할당량에 대한 할당량 증가를 요청할 수 있습니다.

다음 표에는 AWS Elemental MediaStore의 할당량(이전에는 한도라고 함)이 설명되어 있습니다. 할당량은 AWS 계정의 최대 서비스 리소스 또는 작업 수입니다.

> **Note**
>
> 계정 내 개별 컨테이너에 할당량을 할당하려면 AWS Support 또는 계정 관리자에게 문의하세요. 이 옵션을 사용하면 컨테이너 간에 계정 수준 한도를 나누어 하나의 컨테이너가 전체 할당량을 사용하지 않도록 할 수 있습니다.

리소스 또는 작업	기본 할당량	설명
컨테이너	100	계정당 생성할 수 있는 컨테이너의 최대 수입니다.
폴더 수준	10	컨테이너에서 생성할 수 있는 폴더 수준의 최대 수입니다. 폴더가 컨테이너 내에서 10개를 초과하는 수준으로 중첩되지 않는 한 원하는 개수만큼 폴더를 생성할 수 있습니다.
폴더	무제한	폴더가 컨테이너 내에서 10개를 초과하는 수준으로 중첩되지 않는 한 원하는 개수만큼 폴더를 생성할 수 있습니다.
객체 크기	25MB	단일 객체의 최대 파일 크기
Objects	무제한	계정의 폴더 또는 컨테이너에 원하는 개수만큼 객체를 업로드할 수 있습니다.
DeleteObject 요청 비율	100	초당 생성할 수 있는 작업 요청의 최대 수입니다. 추가 요청은 제한됩니다. 할당량 증가를 요청할 수 있습니다.

리소스 또는 작업	기본 할당량	설명
DeleteObject API 요청 비율	1,000	초당 생성할 수 있는 작업 요청의 최대 수입니다. 추가 요청은 제한됩니다. 할당량 증가를 요청할 수 있습니다.
표준 업로드 가용성에 대한 GetObject API 요청 비율	1,000	초당 생성할 수 있는 작업 요청의 최대 수입니다. 추가 요청은 제한됩니다. 할당량 증가를 요청할 수 있습니다.
스트리밍 업로드 가용성에 대한 GetObject API 요청 비율	25	초당 생성할 수 있는 작업 요청의 최대 수입니다. 추가 요청은 제한됩니다. 할당량 증가를 요청할 수 있습니다.
ListItems API 요청 비율	5	초당 생성할 수 있는 작업 요청의 최대 수입니다. 추가 요청은 제한됩니다. 할당량 증가를 요청할 수 있습니다.
청크 전송 인코딩에 대한 PutObject API 요청 비율(스트리밍 업로드 가용성이라고도 함)	10	초당 생성할 수 있는 작업 요청의 최대 수입니다. 추가 요청은 제한됩니다. 할당량 증가를 요청할 수 있습니다. 요청에서 요청한 TPS 및 평균 객체 크기를 지정합니다.
표준 업로드 가용성에 대한 PutObject API 요청 비율	100	초당 생성할 수 있는 작업 요청의 최대 수입니다. 추가 요청은 제한됩니다. 할당량 증가를 요청할 수 있습니다. 요청에서 요청한 TPS 및 평균 객체 크기를 지정합니다.
지표 정책의 규칙	10	지표 정책에 포함할 수 있는 최대 규칙 수입니다.
객체 수명 주기 정책의 규칙	10	단일 객체 수명 주기 정책에 포함할 수 있는 규칙의 최대 수입니다.

AWS Elemental MediaStore 관련 정보

다음 표에는 AWS Elemental MediaStore를 사용할 때 참조할 수 있는 관련 리소스가 나와 있습니다.

- 클래스 및 워크숍 - AWS 기술을 연마하고 실용적인 경험을 얻는 데 도움이 되는 자기 주도형 실습 외에도 역할 기반 및 특수 과정으로 연결되는 링크입니다.
- AWS 개발자 센터 - 자습서를 살펴보고, 도구를 다운로드하고, AWS 개발자 이벤트에 대해 알아봅니다.
- AWS 개발자 도구 - AWS 애플리케이션 개발 및 관리를 위한 개발자 도구, SDKs, IDE 도구 키트 및 명령줄 도구에 대한 링크입니다.
- 리소스 센터 시작하기 -를 설정하고 AWS 계정, AWS 커뮤니티에 가입하고, 첫 번째 애플리케이션을 시작하는 방법을 알아봅니다.
- 실습 튜토리얼 – 단계별 튜토리얼에 따라 AWS에서 첫 번째 애플리케이션을 시작합니다.
- AWS 백서 - 아키텍처, 보안 및 경제와 같은 주제를 다루고 Solutions Architects 또는 기타 기술 전문가가 작성한 AWS 포괄적인 기술 AWS 백서 목록으로 연결되는 링크입니다.
- AWS Support 센터 - AWS Support 사례를 생성하고 관리하기 위한 허브입니다. 포럼, 기술 FAQs, 서비스 상태 및 같은 기타 유용한 리소스에 대한 링크도 포함되어 있습니다 AWS Trusted Advisor.
- 지원 - 클라우드에서 애플리케이션을 구축하고 실행하는 데 도움이 지원되는 one-on-one 빠른 응답 지원 채널에 대한 정보를 제공하는 기본 웹 페이지입니다.
- Contact Us(문의처) - AWS 결제, 계정, 이벤트, 침해 및 기타 문제에 대해 문의할 수 있는 중앙 연락 창구입니다.
- AWS 사이트 약관 - 저작권 및 상표, 계정, 라이선스 및 사이트 액세스, 기타 주제에 대한 자세한 정보입니다.

사용 설명서에 대한 문서 이력

다음 표는 본 AWS Elemental MediaStore 릴리스 관련 설명서를 소개합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
지원 종료 알림	지원 종료 공지: 2025년 11월 13일에는 AWS Elemental MediaStore에 대한 지원을 중단할 AWS 예정입니다. 2025년 11월 13일 이후에는 더 이상 MediaStore 콘솔 또는 MediaStore 리소스에 액세스할 수 없습니다. 자세한 내용은 이 블로그 게시물을 참조하세요.	2024년 11월 12일
원본 액세스 제어(OAC) 개선	AWS Elemental MediaStore와 함께 OAC를 사용하는 방법에 대한 정보를 추가했습니다.	2023년 4월 17일
할당량 업데이트	`Rules in a Metric Policy`에 대한 할당량 값 및 설명을 수정했습니다.	2022년 10월 25일
ExpiresAt 필드	이제 액세스 로그에는 컨테이너 수명 주기 정책의 임시 데이터 규칙을 기반으로 객체의 만료 날짜 및 시간을 나타내는 `ExpiresAt` 필드가 포함됩니다.	2020년 7월 16일
수명 주기 전환 규칙	이제 객체 수명 주기 정책에 수명 주기 전환 규칙을 추가하여 특정 기간이 지난 객체를 IA(액세스 빈도 낮음) 스토리지 클래	2020년 4월 20일

	스로 이동하도록 설정할 수 있습니다.	
컨테이너 비우기	이제 컨테이너 내의 모든 객체를 한 번에 삭제할 수 있습니다.	2020년 4월 7일
Amazon CloudWatch 지표 지원	지표 정책을 설정하여 MediaStore가 CloudWatch로 전송하는 지표를 지정할 수 있습니다.	2020년 3월 30일
객체 삭제 규칙의 와일드카드	이제 객체 수명 주기 정책에서 객체 삭제 규칙에 와일드카드를 사용할 수 있습니다. 이를 통해 특정 기간(일)이 지난 후 서비스에서 삭제할 파일을 파일 이름 또는 확장명을 기준으로 지정할 수 있습니다.	2019년 12월 20일
객체 수명 주기 정책	이제 수명을 초 단위로 나타내는 만료를 나타내는 규칙을 객체 수명 주기 정책에 추가할 수 있습니다.	2019년 9월 13일
CloudFormation 지원	이제 CloudFormation 템플릿을 사용하여 컨테이너를 자동으로 생성할 수 있습니다. CloudFormation 템플릿은 컨테이너 생성, 액세스 로깅 설정, 기본 컨테이너 정책 업데이트, CORS(교차 원본 리소스 공유) 정책 추가, 객체 수명 주기 정책 추가 등의 5가지 API 작업에 대한 데이터를 관리합니다.	2019년 5월 17일

스트리밍 업로드 가용성 할당량	스트리밍 업로드 가용성을 사용하는 객체의 경우(객체의 청크 분할 전송) `PutObject` 작업이 10TPS를 초과할 수 없고 `GetObject` 작업이 25TPS를 초과할 수 없습니다.	2019년 4월 8일
객체의 청크 분할 전송	객체의 청크 분할 전송에 대한 지원이 추가되었습니다. 이 기능은 객체가 완전히 업로드되기 전에 객체를 다운로드하도록 지정할 수 있게 해줍니다.	2019년 4월 5일
액세스 로깅	AWS Elemental MediaStore는 이제 컨테이너의 객체에 대해 이루어진 요청의 상세 레코드를 제공하는 액세스 로깅을 지원합니다.	2019년 2월 25일
객체 수명 주기 정책	객체 수명 주기 정책에 대한 지원이 추가되었습니다. 이 정책은 현재 컨테이너 안의 객체의 만료 날짜를 관리합니다.	2018년 12월 12일
객체 크기 할당량 증가	객체 크기의 할당량은 현재 25MB입니다.	2018년 10월 10일
객체 크기 할당량 증가	객체 크기의 할당량은 현재 20MB입니다.	2018년 9월 6일
AWS CloudTrail 통합	CloudTrail 통합 콘텐츠가 업데이트되어 CloudTrail 서비스에 대한 최신 변경 사항과 부합됩니다.	2018년 7월 12일

CDN 협업	Amazon CloudFront와 같은 콘텐츠 전송 네트워크(CDN)를 AWS Elemental MediaStore와 함께 사용하는 방법에 대한 정보를 추가했습니다.	2018년 4월 14일
CORS 구성	AWS Elemental MediaStore는 이제 교차 오리진 리소스 공유(CORS)를 지원합니다. 따라서 한 도메인에 로드된 클라이언트 웹 애플리케이션이 다른 도메인의 리소스와 상호 작용할 수 있습니다.	2018년 2월 7일
새로운 서비스 및 안내서	비디오 제작 및 스토리지 서비스인 AWS Elemental MediaStore와 AWS Elemental MediaStore 사용 설명서의 첫 릴리스입니다.	2017년 11월 27일

Note

- AWS 미디어 서비스는 애플리케이션의 사용 또는 생명 안전 운영, 탐색 또는 통신 시스템, 항공 교통 관제 또는 서비스의 사용 불가, 중단 또는 장애로 인해 사망, 개인 부상, 재산 손상 또는 환경 손상이 발생할 수 있는 생명 유지 시스템과 같이 안전하지 않은 성능이 필요한 상황에서 사용하도록 설계되거나 의도되지 않았습니다.

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.